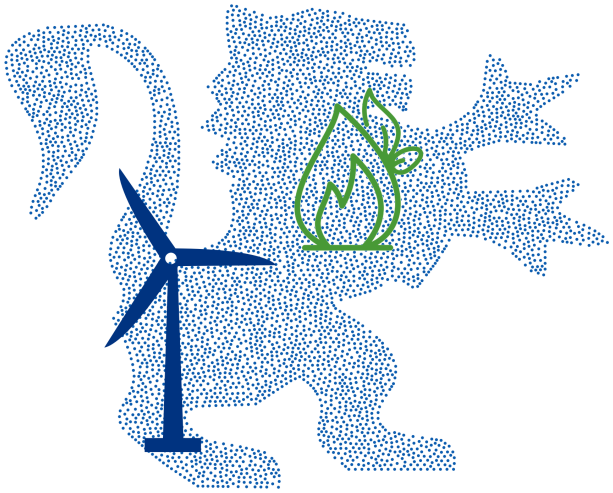
ÉCOLE CENTRALE LYON

Initiation à la modélisation mathématique

Grégory Vial



Formation par alternance

ORGANISATION DU COURS

Cours : introduction à la modélisation mathématique

— Cours 1 : Ajustement polynomial : application à la production d'une éolienne.

— Cours 2 : Un modèle d'équations différentielles pour le chémostat.

— Cours 3 : Optimisation d'un réseau éolien.

— Cours 4 : Compression d'images et analyse de Fourier.

— Cours 5 : Représentation géométrique d'une pale d'éolienne.

SUPPORTS DE COURS

POUR LES ÉLÈVES

Formation par alternance – Mathématiques 2

Ajustement polynomial

Problématique

Le tableau 1 ci-dessous fournit une estimation de la production annuelle d'énergie pour des petites éoliennes, en fonction de la vitesse du vent.

Vitesse moyenne annuelle du vent (m/s)	**Production annuelle d'énergie (kWh/m^2)**
4	130
5	260
6	410
7	570
8	770
9	1020

Table 1 – Production moyenne d'électricité d'une petite éolienne en fonction du vent.

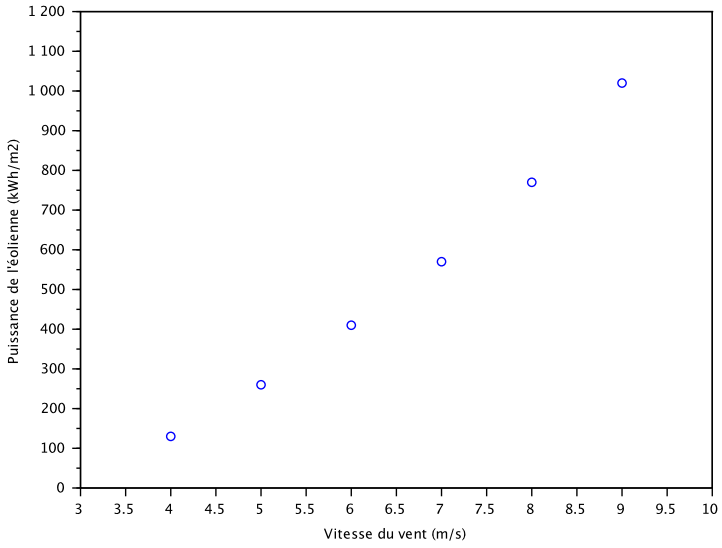


Figure 1 – Le nuage de points des données

Sur la figure 1, on a représenté les données sous la forme d'un nuage de points. La croissance de la puissance vis-à-vis de la vitesse du vent est clairement sur-linéaire. On va proposer un modèle pour *expliquer* la relation entre les deux variables.

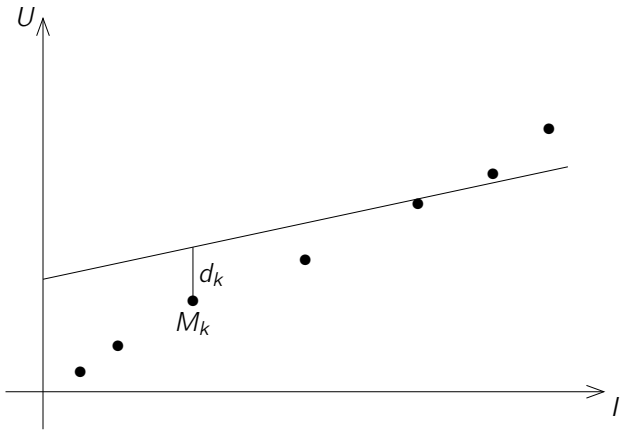


Figure 2 – Mesures aux bornes d'un résistor.

L'exemple simple de l'ajustement linéaire

Prenons une situation plus simple où les données sont représentées par un nuage de points grossièrement alignés. C'est le cas, par exemple, lorsque l'on mesure l'intensité I et la tension U aux bornes d'un résistor, voir Figure 2.

Notre objectif est de proposer une méthode *systématique* pour déterminer une droite qui s'ajuste au mieux au nuage de points. Afin de préciser ce que signifie *au mieux*, on doit proposer un critère d'optimalité que la droite devra satisfaire. Une droite étant donnée par son équation $U = aI + b$, on introduit le nombre

$$\sum_{k=1}^{N} d_k^2,$$

où d_k désigne la distance verticale entre le point M_k – de coordonnées (I_k, U_k) – du nuage, et la droite. Notons que d_k n'est pas la distance géométrique du point M_k à la droite. Ce choix permet l'expression simple suivante :

$$d_k = |U_k - aI_k - b|.$$

La quantité qui mesure la qualité d'ajustement de la droite au nuage est donc la fonction

$$\varphi(a, b) = \sum_{k=1}^{N} (U_k - aI_k - b)^2.$$

Il s'agit donc de déterminer les nombres a et b qui minimisent la fonction φ. Plusieurs questions se posent :

— de tels paramètres a et b existent-ils ?

— si oui, le couple (a, b) est-il unique ?

— si oui, comment les déterminer en pratique ?

On admet ici qu'il existe une unique solution au problème. Pour la détermination pratique de a et b, on calcule les dérivées partielles de la fonction φ :

$$\frac{\partial \varphi}{\partial a}(a, b) = -2 \sum_{k=1}^{N} (U_k - aI_k - b) I_k \quad \text{et} \quad \frac{\partial \varphi}{\partial b}(a, b) = -2 \sum_{k=1}^{N} (U_k - aI_k - b).$$

L'annulation des dérivées partielles conduit au système linéaire

$$\begin{pmatrix} \sum_{k=1}^{N} I_k^2 & \sum_{k=1}^{N} I_k \\ \sum_{k=1}^{N} I_k & N \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{N} U_k I_k \\ \sum_{k=1}^{N} U_k \end{pmatrix}.$$

Sur la figure 3, on a représenté la droite obtenue à l'aide de la résolution du système linéaire précédent.

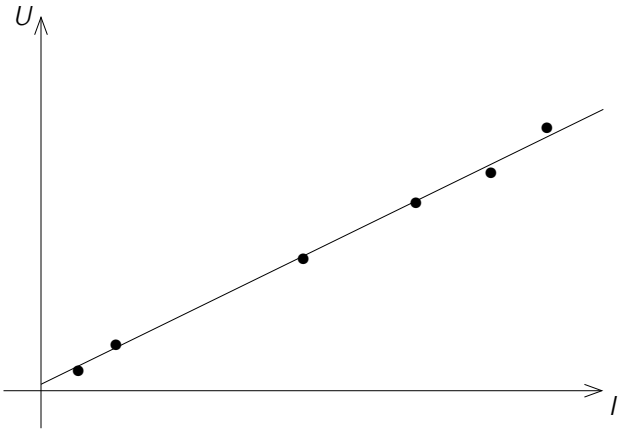


Figure 3 – Ajustement des mesures de la figure 2.

Un modèle quadratique pour le problème de l'éolienne

On cherche à exprimer la production annuelle P comme une fonction de la vitesse v du vent, sous la forme

$$P(v) = a + bv + cv^2,$$

où les coefficients a, b et c sont inconnus. Un critère pour déterminer ces valeurs consiste à minimiser la fonctionnelle

$$\phi(a, b, c) = \sum_{i=1}^{6} \left[a + bv_i + cv_i^2 - P_i\right]^2,$$

où (v_i, P_i) sont les données contenues dans le tableau. De la même manière que précédemment, on calcule les dérivées partielles :

$$\begin{aligned} \frac{\partial \phi}{\partial a} &= 2 \sum_{i=1}^{6} \left[a + bv_i + cv_i^2 - P_i\right] \\ \frac{\partial \phi}{\partial b} &= 2 \sum_{i=1}^{6} v_i \left[a + bv_i + cv_i^2 - P_i\right] \\ \frac{\partial \phi}{\partial c} &= 2 \sum_{i=1}^{6} v_i^2 \left[a + bv_i + cv_i^2 - P_i\right]. \end{aligned}$$

Le système linéaire obtenu s'écrit

$$M \begin{pmatrix} a \\ b \\ c \end{pmatrix} = b,$$

avec

$$M = \begin{pmatrix} 6 & \sum_{i=1}^{6} v_i & \sum_{i=1}^{6} v_i^2 \\ \sum_{i=1}^{6} v_i & \sum_{i=1}^{6} v_i^2 & \sum_{i=1}^{6} v_i^3 \\ \sum_{i=1}^{6} v_i^2 & \sum_{i=1}^{6} v_i^3 & \sum_{i=1}^{6} v_i^4 \end{pmatrix} \quad \text{et} \quad b = \begin{pmatrix} \sum_{i=1}^{6} P_i \\ \sum_{i=1}^{6} P_i v_i \\ \sum_{i=1}^{6} P_i v_i^2 \end{pmatrix}.$$

Le script `Scilab` suivant met en œuvre la méthode décrite plus haut

```
v=4:9;
P=[130,260,410,570,770,1020];
M=[6,sum(v),sum(v.^2);
   sum(v),sum(v.^2),sum(v.^3);
   sum(v.^2),sum(v.^3),sum(v.^4)];
F=[sum(P);sum(v.*P);sum(P.*v.^2)];
x=M\F;

plot(v,P,'o')
a=get("current_axes");
a.data_bounds=[3,100;10,1100];
xlabel('Vitesse du vent (m/s)')
ylabel('Puissance de l''eolienne (kWh/m2)')
t=linspace(3,10,100);
plot(t,x(1)+x(2)*t+x(3)*t.^2)
sqrt(sum((P-x(1)-x(2)*v-x(3)*v.^2).^2))
```

Sur la figure 4 est représenté le résultat de ce script.

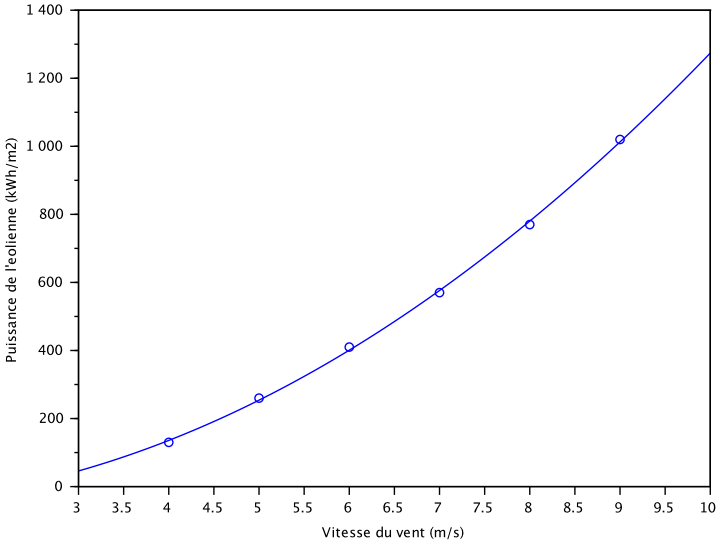


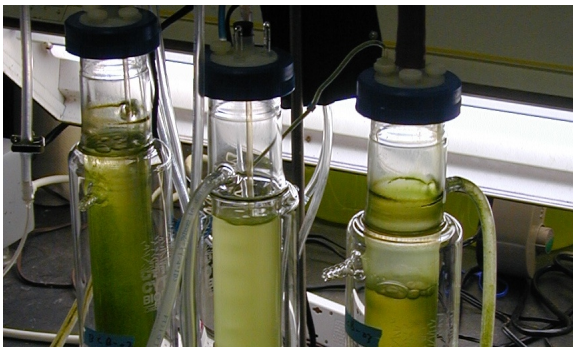
Figure 4 – L'ajustement quadratique pour les données de production éolienne.

Formation par alternance – Mathématiques 2

Modèle mathématique EDO pour le chemostat

Problématique

La production de biogaz est une alternative à l'exploitation des ressources fossiles dans les domaines du chauffage ou du transport. Le biogaz est un mélange, principalement constitué de méthane, obtenu par la fermentation anaérobie de matière organique par un ensemble de bactéries. On s'intéresse ici à un modèle mathématique de la production de biogaz dans un *chemostat*, aussi appelé *bioréacteur*, utilisé en laboratoire.



N.B. *La modélisation qui suit est simplifiée et ne prend en compte que les facteurs les plus importants des phénomènes considérés.*

Modélisation mathématique

Inconnues du problème

Le chemostat est un dispositif expérimental dans lequel on cultive de manière contrôlée des organismes. Les nutriments sont apportés par l'entrée, et on récupère le biogaz à la sortie.

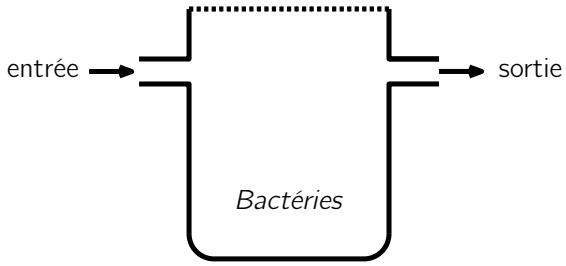


Figure 1 – Principe du chemostat.

La population de bactéries va évoluer au cours du temps, de même que la quantité de nutriment à l'intérieur du chemostat. Cette dernière est directement reliée à la quantité de biogaz produite. On propose ici un modèle dont les inconnues sont

— La population bactérienne dans le chemostat à l'instant t, notée $N(t)$,

— La quantité de nutriment à l'intérieur du chemostat à l'instant t, notée $c(t)$.

Il est important de noter que les fonctions $t \mapsto N(t)$ et $t \mapsto c(t)$ sont supposées *à valeurs réelles*. Ainsi, $N(t)$ ne représente pas un *nombre* de bactéries, mais une *densité* continue. Des modèles discrets pourraient être envisagées pour des populations de faible taille, mais ce n'est pas pertinent dans la situation considérée.

Si l'on note $K(t)$ la quantité de nutriments introduite depuis le début de l'expérience, et jusqu'à l'instant t, dans le chemostat, la quantité de biogaz $B(t)$ produite jusqu'à l'instant t est proportionnelle à la quantité de nutriment consommée :

$$B(t) = \lambda\big(K(t) - c(t)\big), \tag{1}$$

où λ est un coefficient dépendant de la composition du nutriment, supposé connu.

Mise en équations

La dynamique des quantités $N(t)$ et $c(t)$ est régie par les principes suivants :

1. Les bactéries ont un taux de croissance proportionnel à la quantité de nutriment présent, avec saturation vers un taux maximal (on note $r(c)$ la fonction taux).
2. Les bactéries ont un taux de mortalité constant q.
3. Le nutriment est apporté à vitesse constante v_0 par l'entrée du chemostat.
4. Le nutriment est consommé avec un taux constant p par les bactéries.

Ces considérations conduisent aux équations suivantes :

$$\begin{cases} N'(t) &= r\big(c(t)\big)N(t) - qN(t), \\ c'(t) &= v_0 - pr\big(c(t)\big)N(t). \end{cases} \quad \text{pour } t > 0. \tag{2}$$

La fonction $c \mapsto r(c)$ est donnée par

$$r(c) = \frac{Rc}{\beta + c},$$

où R et β sont des constantes fixées (R/β représente le coefficient de proportionnalité entre taux de croissance des bactéries et quantité de nutriment, pour de faibles quantités ; R est le taux maximal). Le programme Scilab suivant permet de représenter graphiquement la fonction c, voir Figure 2.

```
// Parametres
beta=1;
R=2;
// Graphe
c=linspace(0,10,100);
r=R*c./(beta+c);
plot(c,r)
xlabel('Concentration en nutriment c')
ylabel('Taux r(c)')
```

Les équations (2) sont assorties de conditions intiales :

$$N(0) = N_0 \quad \text{et} \quad c(0) = 0. \tag{3}$$

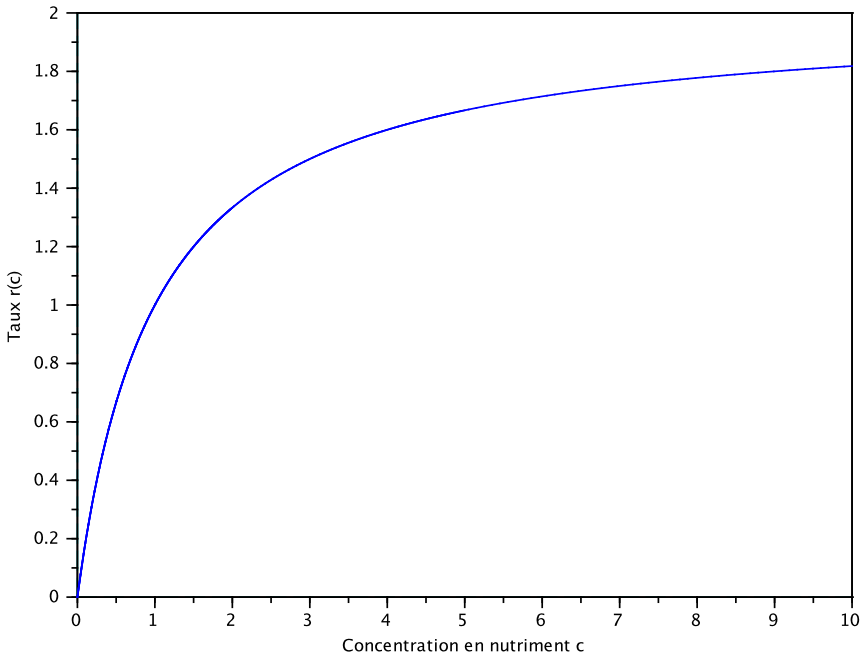


Figure 2 – Fonction $c \mapsto r(c)$ pour $R = 2$ et $\beta = 1$.

On obtient ainsi un système de deux équations différentielles ordinaires (EDO) en les inconnues N et c. Il n'est pas possible de déterminer une expression analytique (même compliquée) des solutions. On doit donc recourir à une simulation numérique pour approcher la solution.

Algorithme numérique de résolution du système d'EDO

Pour une équation différentielle scalaire

On considère le cas de la dimension 1 :

$$\begin{cases} y'(t) = f\big(y(t)\big), \\ y(0) = y_0, \end{cases} \tag{4}$$

où la fonction $f : \mathbb{R} \to \mathbb{R}$ est de classe $\mathscr{C}^1$. Afin d'approcher numériquement la solution $y(t)$, on se place sur un intervalle $[0, T]$ borné, qu'on *discrétise* : soit $K \in \mathbb{N}$ et $\Delta t = T/K$, on pose

$$t_n = n\Delta t, \qquad n = 0, 1, \ldots, K.$$

On cherche à construire une approximation y_n de $y(t_n)$. Pour cela, on part de la formule de Taylor

$$y'(t_n) \simeq \frac{y(t_{n+1}) - y(t_n)}{\Delta t}.$$

Comme $y'(t_n) = f\big(y(t_n)\big)$, il est naturel de définir les (y_n) par la récurrence

$$f(y_n) = \frac{y_{n+1} - y_n}{\Delta t} \quad \text{soit encore} \quad y_{n+1} = y_n + \Delta t\, f(y_n).$$

Ainsi, partant de y_0, on est capable de calculer tous les (y_n) pour $n \leq K$. C'est la méthode d'Euler.

Pour un système d'équations différentielles

La méthode d'Euler se généralise très simplement au cas multi-dimensionnel. Pour le problème qui nous intéresse ici, elle s'écrit

$$\left\{ \begin{array}{rcl} N_{n+1} & = & N_n + \Delta t \left(r(c_n) N_n - q N_n \right), \\ c_{n+1} & = & c_n + \Delta t \left(v_0 - p r(c_n) N_n \right). \end{array} \right.$$

Le script Scilab suivant programme la méthode d'Euler :

```
// Parametres
B=1;
R=2;
v0=0.7;
q=1;
p=1;

// Initialisation
N=2;
c=0;

// Methode d'Euler
Nt=500;
T=30;
dt=T/Nt;

lN=N;lc=c;t=0;
for n=1:Nt
    r=R*c/(B+c);
    c=c+dt*(v0-p*r*N);
    N=N+dt*(r*N-q*N);
        lN=[lN,N];
        lc=[lc,c];
        t=t+dt;
end

// Graphe
t=linspace(0,T,Nt+1);
plot(t,lN,t,lc)
xlabel('temps')
legend(['N(t)','c(t)'])
```

La figure 3 présente les courbes $t \mapsto N(t)$ et $t \mapsto c(t)$ obtenues pour un pas de temps $\Delta t = 10^{-2}$. On vérifie que les fonctions N et c restent positives au cours du temps (ce qui est souhaitable car elles représentent des quantités). On observe aussi qu'elles convergent toutes-deux lorsque $t \to +\infty$ vers un état stationnaire.

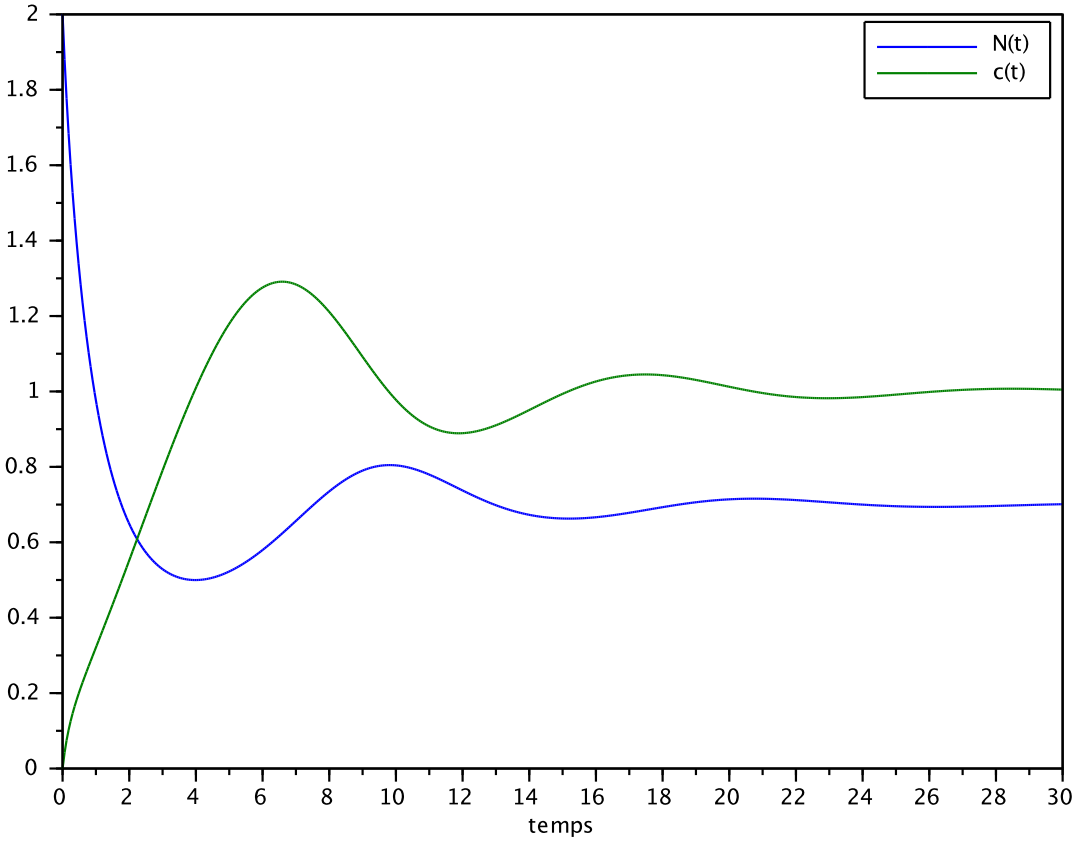


Figure 3 – Résultat obtenu à l'aide de la méthode d'Euler pour $\Delta t = 10^{-2}$.

ÉCOLE CENTRALELYON

2017-2018

Formation par alternance – Mathématiques 2

Optimisation d'un réseau éolien

Problématique

Une entreprise travaillant dans le secteur des énergies renouvelables souhaite implanter un champ d'éoliennes offshore. Chaque éolienne est reliée aux autres par un câble électrique afin d'assurer le transport de l'énergie électrique, le tout étant enfin relié au réseau. Afin de réduire le coût du câblage, il est naturel de réduire la distance entre deux éoliennes. Toutefois, celles-ci sont trop proches, des phénomènes de turbulence peuvent apparaître, réduisant le rendement du système de production.

On cherche donc à optimiser la topologie du réseau éolien dans le but de minimiser le coût d'amortissement de l'installation.

N.B. *Insistons sur le fait que notre approche est simplifiée à l'extrême. Elle n'a pas pour but de répondre de manière efficace à la question pratique, mais de dégager la démarche scientifique et les outils mathématiques pour la résolution de problèmes similaires. En particulier, les phénomènes physiques ne sont pris en compte que de manière phénoménologique, et la manière de les quantifier n'est pas réaliste.*

Modélisation mathématique

Comme toujours lorsqu'on met en place un *modèle*, on commence par faire des hypothèses simplificatrices et on essaie d'identifier les *paramètres* (ou *degrés de liberté*) qui caractérisent le problème étudié.

On se restreint ici à une seule ligne composée de $N+2$ éoliennes (N est fixé pour notre étude). Les éoliennes extrémales sont supposées fixes, et on cherche à optimiser la position des N autres. Pour simplifier encore, on considère que les éoliennes sont situées sur des axes parallèles équidistants, et seule leur position sur ces axes reste indéterminée. Après introductions d'un repère orthonormé adapté, l'éolienne E_i est située aux coordonnées (ih, y_i), où h est la distance entre deux axes successifs, cf. Figure 1. Les variables du modèle sont donc les ordonnées y_i ($i = 1, 2, \ldots, N$) des éoliennes internes (on suppose que les éoliennes extrémales sont sur l'axe des ordonnées : $y_0 = y_{N+1} = 0$). On note $Y \in \mathbb{R}^N$ le vecteur des paramètres :

$$Y = (y_1, y_2, \ldots, y_n).$$

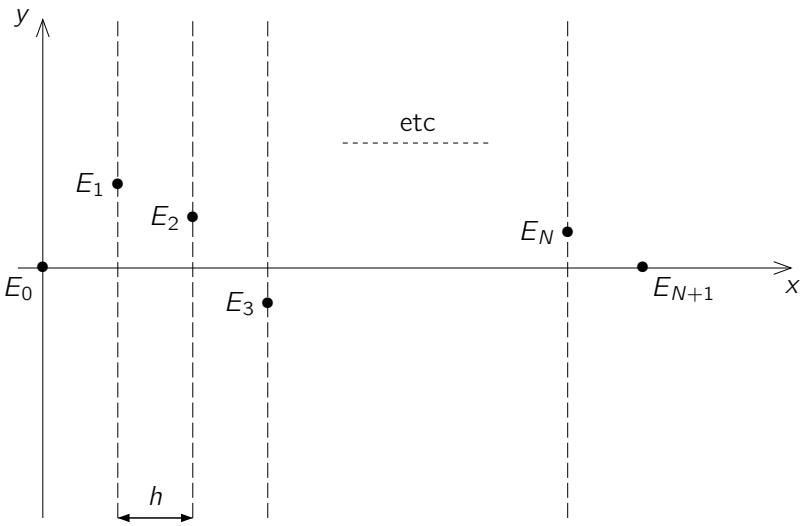


Figure 1 – Positions des éoliennes

Une configuration Y étant fixée, la longueur du câble reliant les éoliennes entre-elles est la longueur de la ligne brisée constituée par le réseau :

$$L(Y) = \sum_{i=0}^{N} d(E_i, E_{i+1}) = \sum_{i=0}^{N} \sqrt{h^2 + (y_{i+1} - y_i)^2}.$$

Si c désigne le prix d'une unité de câble, le coût de câblage est $c \times L(Y)$.

Afin de prendre en compte les pertes de productivité liées à la proximité des éoliennes, on suppose que seules les éoliennes adjacentes interviennent, et que cette perte est proportionnelle à l'inverse de la distance entre deux éoliennes voisines :

$$P(Y) = k \left[\frac{1}{d(E_0, E_1)} + \sum_{i=1}^{N} \left(\frac{1}{d(E_{i-1}, E_i)} + \frac{1}{d(E_i, E_{i+1})} \right) + \frac{1}{d(E_N, E_{N+1})} \right].$$

On voit facilement que

$$P(Y) = 2k \sum_{i=0}^{N} \frac{1}{d(E_i, E_{i+1})} = 2k \sum_{i=0}^{N} \frac{1}{\sqrt{h^2 + (y_{i+1} - y_i)^2}}$$

Notre problème d'optimisation s'écrit donc

$$\min \left\{ P(Y) + c \times L(Y) \; ; \; Y \in \mathbb{R}^N \right\}, \tag{1}$$

avec la convention $y_0 = y_{N+1} = 0$ dans les formules donnant $L(Y)$ et $P(Y)$.

Dans la suite, on notera $F(Y) = P(Y) + c \times L(Y)$ la fonction *objectif*.

Analyse théorique du modèle

Un calcul explicite

Dans le cas où $N = 1$ (une seule éolienne interne dont on souhaite optimiser la position), l'analyse peut-être effectuée complètement. En effet, si on note $y_1 = y$, la fonction à optimiser devient

$$F(Y) = f(y) = \frac{4k}{\sqrt{h^2 + y^2}} + 2c \times \sqrt{h^2 + y^2}.$$

Une simple étude de fonction continue montre que f tend vers $+\infty$ en $\pm\infty$, et qu'elle admet donc un minimum. L'étude des points critiques fournit :

$$f'(y) = 0 \iff -\frac{4ky}{(h^2+y^2)^{\frac{3}{2}}} + \frac{2cy}{\sqrt{h^2+y^2}} = 0 \iff \frac{2y}{(h^2+y^2)^{\frac{3}{2}}}\left(-2k + ch^2 + cy^2\right) = 0.$$

Ainsi, si $ch^2 \geq 2k$, seule $y = 0$ est solution. Pour $ch^2 < 2k$ il y a trois solutions :

$$y = 0 \quad \text{et} \quad y = y_{\pm}^* \stackrel{\text{def.}}{=} \pm\sqrt{\frac{2k - ch^2}{c}}.$$

À l'aide du programme Scilab `Opt1D.sce` en annexe, on peut tracer la courbe $y \mapsto f(y)$ dans les deux cas, cf. Figure 2.

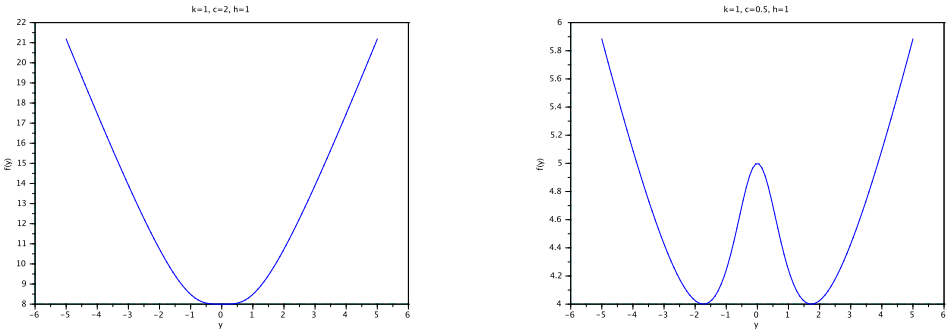


Figure 2 – Allure de la fonction f dans le cas $ch^2 \geq 2k$ (gauche) et $ch^2 < 2k$ (droite).

L'étude des variations de la fonction f prouve que le minimum recherché est atteint en $y = 0$ lorsque $ch^2 \geq 2k$, et en $y_{\pm}^*$ lorsque $ch^2 < 2k$. Remarquons que lorsque le câble est cher (c grand), alors on a intérêt à minimiser la longueur du réseau ($y = 0$).

Une étude graphique

Dès que N est supérieur ou égal à 2, il n'est plus facile d'effectuer tous les calculs explicitement. À l'aide du programme Scilab `Opt2D.sce` (cf. annexe), on peut tracer la surface $(y_1, y_2) \longmapsto F(Y)$.

La figure 3 présente la surface obtenue pour le jeu de paramètres ($k = 1$, $c = 0.5$, $h = 1$). Il apparaît que le minimum sur la grille considérée (100 points pour y_1 et y_2, répartis uniformément entre -5 et 5), est atteint pour

$$y_1^* = -1.2626263 \quad \text{et} \quad y_2^* = 1.2626263,$$

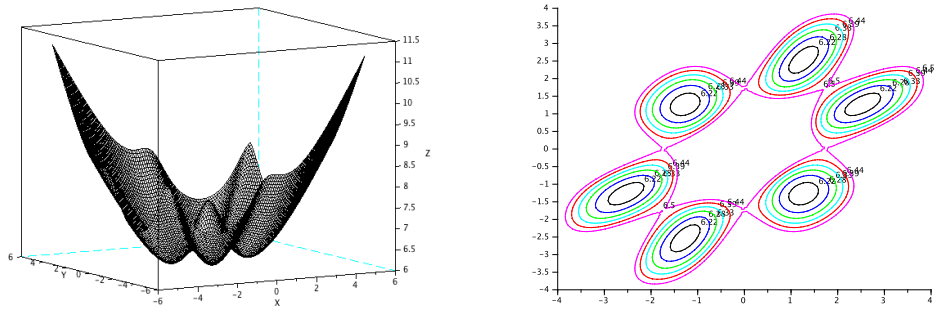


Figure 3 – Allure de la fonction f pour $N = 2$ et $k = 1$, $c = 0.5$, $h = 1$.

avec une valeur $F(y_1^*, y_2^*) = 6.1885014$.

Pour des raisons de symétrie, $(-y_1^*, -y_2^*)$ est aussi solution. Toutefois, le tracé des lignes de niveau laisser penser que le minimum est en fait atteint en 6 points.

Algorithmes numériques

On a vu que, pour $N = 2$, les calculs explicites étaient délicats à mener. Il est nécessaire de s'orienter vers une méthode numérique.

Méthode du gradient

Il s'agit de construire une suite $(Y_n)_n$ de vecteurs de $\mathbb{R}^N$, dont on espère qu'elle converge vers une solution du problème de minimisation. La méthode consiste à suivre la ligne de *plus grande pente* à chaque étape :

$$Y_{n+1} = Y_n - \rho \nabla F(Y_n),$$

où $\rho > 0$ est un paramètre de l'algorithme, appelé *pas* de la méthode.

Cette méthode nécessite le calcul du gradient de la fonction F, donné ci-dessous

$$\frac{\partial F}{\partial y_i}(Y) = c\left(\frac{y_i - y_{i+1}}{d_i} + \frac{y_i - y_{i-1}}{d_{i-1}}\right) + 2k\left(\frac{y_{i+1} - y_i}{d_i^3} + \frac{y_{i-1} - y_i}{d_{i-1}^3}\right),$$

où $d_i = \sqrt{h^2 + (y_{i+1} - y_i)^2}$ désigne la distance $d(E_i, E_{i+1})$.

Le programme Scilab `Gradient2D.sce` (cf. annexe) met en œuvre cette méthode pour le problème bidimensionnel ($N = 2$). On peut faire les remarques suivantes à l'issue des simulations numériques :

- Le choix du pas est crucial : s'il est trop grand, la méthode peut ne pas converger.
- En cas de convergence, selon le point de départ choisi, on n'obtient pas la même limite. Par exemple, on peut effectuer les tests suivants :
  - $Y_0 = (1, -1)$: l'algorithme converge vers $(1.2621681, -1.2621681)$, où la fonction F vaut 6.1885012.
  - $Y_0 = (1.5, 2.5)$: l'algorithme converge vers $(1.2621707, 2.5243403)$, où la fonction F vaut 6.1885012.

— $Y_0 = (2.5, 1.5)$: l'algorithme converge vers $(2.5243403, 1.2621707)$, où la fonction F vaut 6.1885012.

— $Y_0 = (1, 1)$: l'algorithme converge vers $(1.7320489, 1.7320489)$, où la fonction F vaut 6.5.

Ainsi, les trois premiers points (et leurs opposés) fournissent des points de minimum, comme observé sur la représentation graphique de la surface $Y \mapsto F(Y)$. En revanche, le dernier point est bien un point d'annulation du gradient, mais pas un point de minimum : il s'agit en fait d'un point selle (comme permettrait de le justifier l'étude de la hessienne).

— Le critère d'arrêt utilisé est lié à l'incrément $Y_{n+1} - Y_n$, ce qui revient à arrêter les itérations lorsque $\rho \nabla F(Y_n)$ est inférieur à la tolérance prescrite. Il faut aussi veiller à imposer un nombre maximal d'itérations, qui permet d'arrêter l'algorithme en cas de non convergence.

La figure 4 présente – à symétrie près – les solutions du problème d'optimisation. Insistons sur l'absence d'unicité pour notre problème, qui persistera très vraisemblablement pour $N > 2$. En effet seules les distances entre éoliennes interviennent dans la fonction à minimiser, et les trois configurations trouvées possèdent les mêmes distances entre deux éoliennes voisines, dont on a seulement perturbé l'ordre.

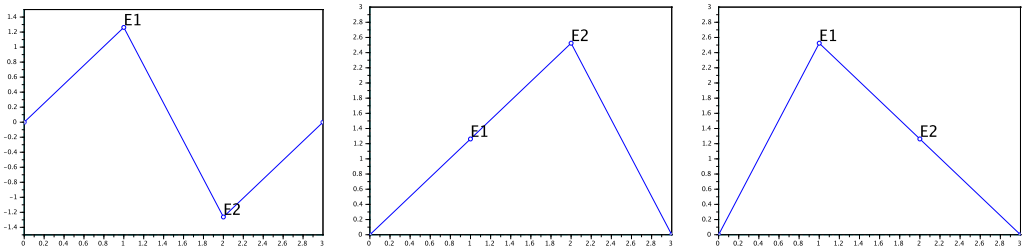


Figure 4 – Position optimale des deux éoliennes internes ($N = 2$).

.

La méthode peut-être facilement généralisée à d'autres valeurs de N, cf. programme Scilab `GradientND` en annexe. La figure 5 présente les configurations obtenues pour $N = 5$. La valeur commune de la fonction objectif est 12. Elles sont obtenues en partant de vecteurs initiaux Y_0 choisis au hasard (les composantes, indépendantes, suivent une loi uniforme sur $[-1, 1]$). Notons que l'algorithme converge aussi parfois vers des minima locaux, comme présenté sur la figure 6, où la valeur obtenue pour F à convergence est 12.377002.

Annexe : programmes Scilab

```
// Opt1D.sce
k=1;c=.5;h=1;

y=linspace(-5,5,100);
f=4*k./sqrt(h^2+y.^2)+2*c*sqrt(h^2+y.^2);
clf
plot(y,f)
xlabel('y')
ylabel('f(y)')
title('k=1, c=0.5, h=1')
```

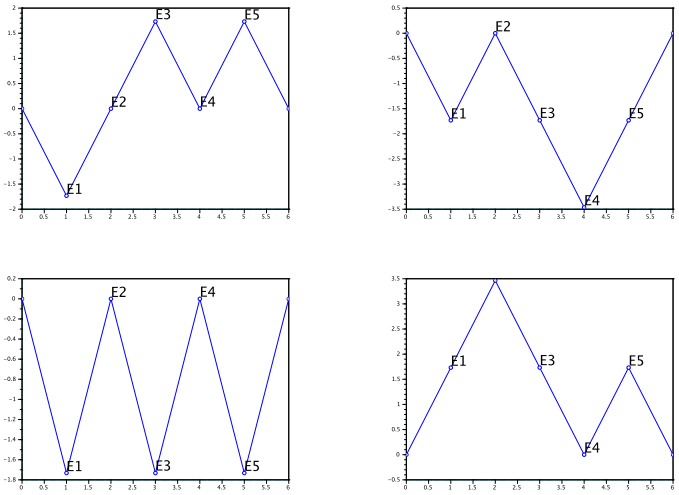


Figure 5 – Position optimale des éoliennes ($N = 5$).

.

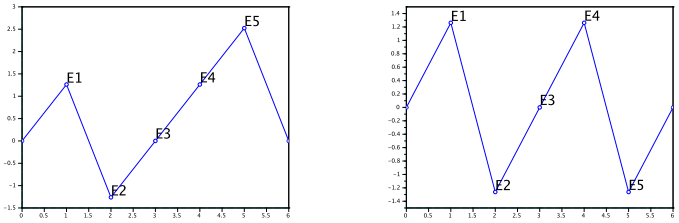


Figure 6 – Minima locaux ($N = 5$).

.

```
// Opt2D.sce
k=1;c=0.5;h=1;

y1=linspace(-5,5,100);
y2=linspace(-5,5,100);

[Y1,Y2]=meshgrid(y1,y2);
d0=sqrt(h^2+Y1.^2);
d1=sqrt(h^2+(Y2-Y1).^2);
d2=sqrt(h^2+Y2.^2);
L=d0+d1+d2;
P=(1)./d0+(1)./d1+(1)./d2;
F=2*k*P+c*L;
title('k=1, c=0.5, h=1')

clf
mesh(Y1,Y2,F)

[Val,Ind1]=min(F)
Y1(Ind1(1),Ind1(2))
Y2(Ind1(1),Ind1(2))

scf

contour(y1,y2,F,linspace(6,6.5,10))
```

```
// Gradient2D.sce
k=1;h=1;c=.5;

function val=F(Y)
N=length(Y);
Z=[0;Y;0];
val=0;
for i=0:N
    di=sqrt(h^2+(Z(i+2)-Z(i+1))^2);
    val=val+c*di+2*k/di;
end

endfunction

//

function grad=gradF(Y)
N=length(Y);
Z=[0;Y;0];
grad=zeros(N,1);
for i=1:N
    di=sqrt(h^2+(Z(i+2)-Z(i+1))^2);
    dm=sqrt(h^2+(Z(i+1)-Z(i))^2);
    grad(i)=c*(-(Z(i+2)-Z(i+1))/di+(Z(i+1)-Z(i))/dm);
    grad(i)=grad(i)+2*k*((Z(i+2)-Z(i+1))/di^3-(Z(i+1)-Z(i))/dm^3);
end

endfunction;

//
```

```
N=2;
tol=1e-8;
rho=1e-2;
Y=[1;-1];
//Y=[1.5;2.5];
//Y=[2.5;1.5];
//Y=[1;1];
iter=0;
itermax=1e5;
err=1+tol;
while (err>tol & iter<itermax)
    Yt=Y-rho*gradF(Y);
    err=norm(Yt-Y);
    iter=iter+1;
    Y=Yt;
end

clf
plot([0;h;2*h;3*h],[0;Y;0],'o-')
xstring(h,Y(1),'E1')
t=get("hdl");t.font_size=5;
xstring(2*h,Y(2),'E2')
t=get("hdl");t.font_size=5;
```

```
// Gradient2D.sce
k=1;h=1;c=.5;

function val=F(Y)
N=length(Y);
Z=[0;Y;0];
val=0;
for i=0:N
    di=sqrt(h^2+(Z(i+2)-Z(i+1))^2);
    val=val+c*di+2*k/di;
end

endfunction

//

function grad=gradF(Y)
N=length(Y);
Z=[0;Y;0];
grad=zeros(N,1);
for i=1:N
    di=sqrt(h^2+(Z(i+2)-Z(i+1))^2);
    dm=sqrt(h^2+(Z(i+1)-Z(i))^2);
    grad(i)=c*(-(Z(i+2)-Z(i+1))/di+(Z(i+1)-Z(i))/dm);
    grad(i)=grad(i)+2*k*((Z(i+2)-Z(i+1))/di^3-(Z(i+1)-Z(i))/dm^3);
end

endfunction;

//
```

```
N=5;
tol=1e-8;
rho=1e-2;
Y=2*rand(N,1)-1
iter=0;
itermax=1e5;
err=1+tol;
while (err>tol & iter<itermax)
    Yt=Y-rho*gradF(Y);
    err=norm(Yt-Y);
    iter=iter+1;
    Y=Yt;
end

clf
plot(h*(0:N+1),[0;Y;0],'o-')
for i=1:N
    xstring(i*h,Y(i),'E'+string(i));
    t=get("hdl");t.font_size=5;
end
```

ÉCOLE CENTRALELYON

2017-2018

Formation par alternance – Mathématiques 2

Compression d'images et analyse de Fourier

Problématique

On s'intéresse au problème de la compression de fichiers informatiques représentant des images. De nombreuses formats existent pour réduire la taille des images, par exemple `jpeg` ou `jpeg2000`. Les techniques utilisées reposent sur l'analyse multi-résolution et la théorie des ondelettes. On s'attache ici à adopter une stratégie similaire à l'aide des séries de Fourier. Soulignons que les séries trigonométriques ne sont pas bien adaptées à la représentation d'images, et c'est d'ailleurs une des raisons de l'introduction des ondelettes.

Figure 1 – L'image utilisée pour les tests.

Modélisation mathématique

On considère l'image en niveaux de gris ci-dessus. Elle peut être représentée par un tableau dont chaque entrée correspond au niveau de gris d'un pixel :

$$M_{p,q} \in \{0, 1, \ldots, 255\} \qquad (0 = \text{noir}, 255 = \text{blanc}).$$

L'image de la figure 1 contient 256×256 pixels, si bien que la matrice M possède 256 lignes et 256 colonnes.

Ainsi, le stockage de la matrice ainsi représentée nécessite 256×256 entiers entre 0 et 255. Un tel entier est stocké sur 1 octet, donc la taille du fichier (en dehors d'autres informations stockées par le logiciel ou le système d'exploitation) vaudra 64 ko. Cette taille peut paraître faible, mais si l'on considère une image de même taille avec 16 millions de couleurs plutôt que 256 niveaux de gris, la taille devient 1.5 Mo, ce qui est très important pour une si petite image. Il est donc important de mettre au point des procédés de compression.

On ne s'intéresse pas ici aux procédés informatiques de réduction de la taille des fichiers sans perte d'information (cf. formats `ZIP`, `gz`, etc), mais à des procédés adaptés à des fichiers d'images, admettant une perte d'information si elle n'est pas trop visible à l'œil nu.

Les images présentées dans ce document ont été affichées à l'aide de `Scilab` à partir d'une matrice M telle que définie plus haut. La fonction suivante a été utilisée.

```
function image(M)
s=scf();
s.color_map=graycolormap(256);
a=gca();a.isoview='on';
Matplot(M');
a.axes_visible='off';
```

Par ailleurs, la matrice correspondant à l'image de la figure 1 est stockée dans un fichier `ITII.mat` qui peut être chargé dans `Scilab` à l'aide de la commande

```
M=fscanfMat('ITII.mat');
```

On dispose alors de la matrice M, de taille 256×256 qui peut être utilisée comme n'importe quelle variable `Scilab`.

Un procédé de compression naïf

Une première solution pour réduire le stockage consiste à « oublier » une ligne et une colonne sur deux. On ne conserve ainsi que 25% de l'information présente initialement. On peut augmenter la compression en ne conservant qu'une ligne et colonne sur 4, 8 ou 16, etc.

Voici le programme Scilab permettant d'effectuer ce travail.

```
// Compression par effacement
// M est la matrice de l'image de depart

// 1 ligne sur 2
image(M(1:2:$,1:2:$));
// 1 ligne sur 4
image(M(1:4:$,1:4:$));
// 1 ligne sur 8
image(M(1:8:$,1:8:$));
// 1 ligne sur 16
image(M(1:16:$,1:16:$));
```

Les résultats obtenus sont consignés dans la figure 2. Au bas de chaque image est indiqué le pourcentage d'information contenu par rapport à l'image initiale. Ainsi, lorsqu'on ne conserve qu'une ligne et une colonne sur deux, on ne conserve que 25% ; c'est aussi le taux de compression, i.e. le rapport de la taille du nouveau fichier sur celle du fichier initial.

Indiquons que les images obtenues ont été lissées automatiquement par Scilab, ce qui explique l'effet de flou observé. De plus, les images apparaissent toutes avec la même taille alors qu'en réalité la dernière ne contient que 16×16 pixels.

Filtre passe-bas par séries de Fourier

La méthode que nous présentons ici nécessite d'abstraire le problème. On suppose que la matrice représentant l'image n'est que la discrétisation d'une version continue de l'image. Cela est à rapprocher des différentes tailles de matrices obtenues à partir de diverses résolutions d'une même image. On considère donc une fonction

$$f : [0,1] \times [0,1] \to \mathbb{R}$$

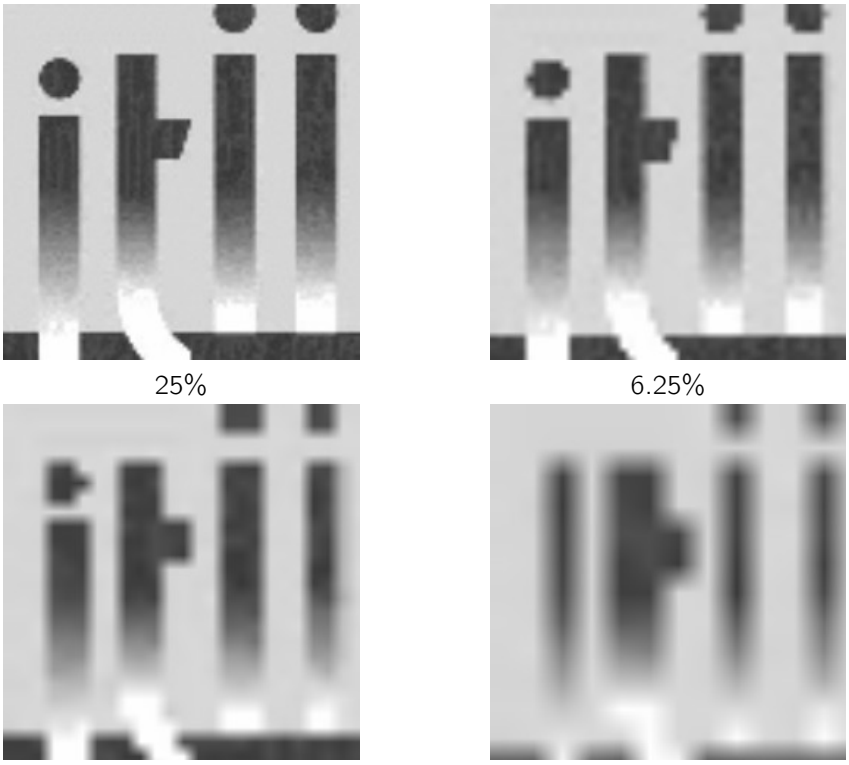


Figure 2 – Compression par effacement.

et l'on suppose que la matrice M, de taille $n \times n$ est l'évaluation de f en les points d'une grille :

$$M_{p,q} = f\left(\frac{p-1}{n-1}, \frac{q-1}{n-1}\right), \qquad 1 \leq p, q \leq n. \tag{1}$$

Si l'on est capable de stocker une approximation de f avec moins d'espace que celui occupé par la matrice M, on aura obtenu un procédé de compression. Bien sûr, il faudra veiller à ce que l'approximation de f fournisse un résultat graphique satisfaisant pour l'œil.

L'idée que nous allons mettre en œuvre utilise les séries de Fourier : $y \in [0,1]$ étant fixé, on peut considérer la fonction d'une variable

$$f_y : x \mapsto f(x,y).$$

Vue comme fonction 1-périodique, elle se décompose en série de Fourier :

$$f_y(x) = \sum_{k\in\mathbb{Z}} c_k(y) \exp(2ik\pi x),$$

où le k-ième coefficient de Fourier c_k est donné par

$$c_k(y) = \int_0^1 f_y(x) \exp(-2ik\pi x)\,\mathrm{d}x.$$

Il dépend bien sûr du paramètre y et est donc noté comme une fonction de y. Cette fonction peut alors elle-même être décomposée en série de Fourier selon y :

$$c_k(y) = \sum_{\ell\in\mathbb{Z}} c_{k,\ell} \exp(2i\ell\pi y),$$

où les coefficients $c_{k,\ell}$ sont ici des nombres réels :

$$c_{k,\ell} = \int_0^1 c_k(y) \exp(-2i\ell\pi y)\,\mathrm{d}y.$$

Ainsi, on a obtenu le développement en deux variables suivant :

$$f(x,y) = \sum_{k\in\mathbb{Z}} \sum_{\ell\in\mathbb{Z}} c_{k,\ell} \exp(2ik\pi x + 2i\ell\pi y), \tag{2}$$

où les coefficients $c_{k,\ell}$ sont donnés par

$$c_{k,\ell} = \int_0^1 \int_0^1 f(x,y) \exp(-2ik\pi x - 2i\ell\pi y)\,\mathrm{d}x\,\mathrm{d}y. \tag{3}$$

Ainsi, la connaissance de la fonction f pour tout $(x,y) \in [0,1] \times [0,1]$ revient à celles des coefficients $c_{k,\ell}$ pour $k, \ell \in \mathbb{Z}$.

Si la fonction f n'est connue que sur une grille de points, comme c'est le cas pour une image, cf. formule (1), on a un équivalent discret de l'analyse en série de Fourier. Pour $1 \leq k, \ell \leq n$, on définit la *transformée de Fourier discrète* par

$$\widehat{M}_{k,\ell} = \sum_{p=1}^{n} \sum_{q=1}^{n} M_{p,q} \exp\left(-\frac{2i\pi}{n}(p-1)(k-1) - \frac{2i\pi}{n}(q-1)(\ell-1)\right). \tag{4}$$

La matrice M peut alors être retrouvée par la formule d'inversion

$$M_{q,p} = \frac{1}{n^2} \sum_{k=1}^{n} \sum_{\ell=1}^{n} \widehat{M}_{k,\ell} \exp\left(\frac{2i\pi}{n}(p-1)(k-1) + \frac{2i\pi}{n}(q-1)(\ell-1) \right). \tag{5}$$

Ainsi, la connaissance des n^2 coefficients $(\widehat{M}_{k,\ell})$ est équivalente à celle des n^2 valeurs $(M_{p,q})$. On ne gagne bien sûr rien en terme de stockage ! Toutefois, on peut se demander si une stratégie d'effacement similaire à celle présentée plus haut, mais appliquée aux coefficients de Fourier, plutôt qu'aux valeurs, serait plus efficace. Une première idée consiste à ne conserver que les coefficients de Fourier de plus basse fréquence, postulant (ce qui est contestable) que l'essentiel de l'information y est contenu.

On forme donc la somme suivant les plus basses fréquences :

$$\frac{1}{n^2} \sum_{k\in\mathcal{B}} \sum_{\ell\in\mathcal{B}} \widehat{M}_{k,\ell} \exp\left(\frac{2i\pi}{n}(p-1)(k-1) + \frac{2i\pi}{n}(q-1)(\ell-1) \right),$$

où $\mathcal{B}$ représente la bande de fréquences retenue :

$$\mathcal{B} = \left\{1, 2, 3, \ldots, \frac{n}{d} - 1\right\} \cup \left\{n - \frac{n}{d}, n - \frac{n}{d} + 1, \ldots, n\right\}.$$

On n'a donc conservé qu'un coefficient sur d^2. Dans la suite, on prendre $d = 2, 4, 8, 16$ si bien qu'on retrouvera les mêmes taux de compression que pour la méthode d'effacement. Le script `Scilab` suivant effectue ce travail.

```
// Compression par Fourier passe-bas
N=size(M,1);
F=fft(M);
// 25%
ind = [1,2:N/4-1,N-N/4:N];
P=real(ifft(F(ind,ind)))/4;
image(P);
// 6%
ind = [1,2:N/8-1,N-N/8:N];
P=real(ifft(F(ind,ind)))/16;
image(P);
// 1.5%
ind = [1,2:N/16-1,N-N/16:N];
P=real(ifft(F(ind,ind)))/64;
image(P);
// 0.4%
ind = [1,2:N/32-1,N-N/32:N];
P=real(ifft(F(ind,ind)))/256;
image(P);
```

Les résultats obtenus sont représentés sur la figure 3. On remarque que la forme principale de l'image est assez bien conservée, mais que les détails sont rapidement dégradés. En comparaison avec la figure 2, le résultat peut être considéré comme à peine meilleur.

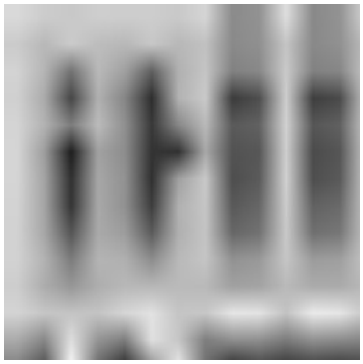
Un effet d'écho est observé au voisinage des changements brutaux de couleur (sur la frontière des lettres, et de la ligne noire du bas). Cela correspond à une mauvaise approximation des discontinuités, connu sous le nom de *phénomène de Gibbs*.

N.B. *Les calculs sont effectués à partir des fonctions* `fft` *et* `ifft` *de* `Scilab`*, qui fournissent un algorithme rapide de calcul des quantités définies plus haut. Signalons que cet*

25%

6.25%

1.6%



0.4%

Figure 3 – Compression par filtre passe-bas.

algorithme (dû à Cooley et Tukey, 1965) a un coût de calcul en $n \log n$ alors que les formules (5) *et* (4) *requièrent* a priori n^2 *opérations.*

Compression par suppression des coefficients de faible amplitude

En s'appuyant toujours sur la représentation de Fourier (5), on peut adopter une stratégie différente : ne conserver que les coefficients « grands » dans la somme. Cela revient à choisir les indices non plus dans une bande $\mathcal{B} \times \mathcal{B}$ mais selon

$$\mathcal{I} = \left\{(k,\ell)\ ;\ |\widehat{M}_{k,\ell}| \geq \alpha\right\}.$$

Ainsi, la somme considérée pour reconstruire l'image est

$$\frac{1}{n^2} \sum_{(k,\ell)\in\mathcal{I}} \widehat{M}_{k,\ell} \exp\left(\frac{2i\pi}{n}(p-1)(k-1) + \frac{2i\pi}{n}(q-1)(\ell-1)\right).$$

le seuil α étant choisi pour obtenir un taux de compression prédéfini. Le script suivant met en œuvre cette méthode. Les résultats sont visibles sur la figure 4.

```
// Compression par Fourier grandes amplitudes
N=size(M,1);
T=mtlb_sort(matrix(abs(F),N*N,1));
// 25%
MatInd = abs(F)>T(floor((N*N)*(1-1/4)));
P=real(ifft(F.*MatInd));image(P)
// 6%
MatInd = abs(F)>T(floor((N*N)*(1-1/16)));
P=real(ifft(F.*MatInd));image(P)
// 1.5%
MatInd = abs(F)>T(floor((N*N)*(1-1/64)));
P=real(ifft(F.*MatInd));image(P)
// 0.4%
MatInd = abs(F)>T(floor((N*N)*(1-1/256)));
P=real(ifft(F.*MatInd));image(P)
```

On observe un résultat meilleur que précédemment pour des taux de compression de 25% et 6.25%, mais pour des compressions plus fortes le résultat est encore assez mauvais, quoique de manière différente.

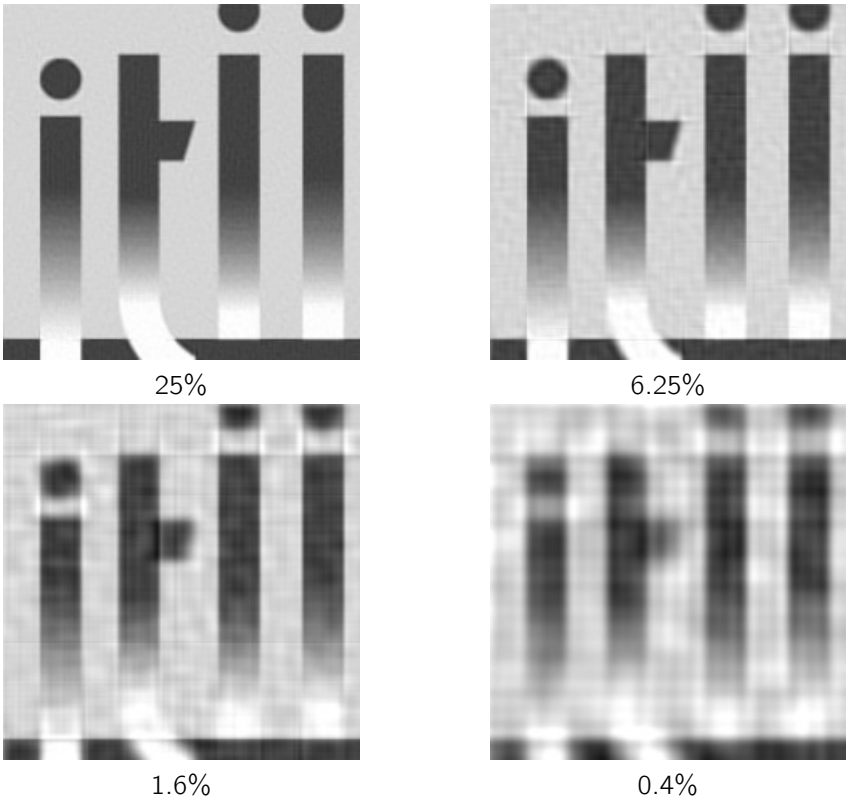


Figure 4 – Compression par suppression des coefficients de faible amplitude.

ÉCOLE CENTRALELYON

2017-2018

Formation par alternance – Mathématiques 2

Représentation géométrique d'une pale d'éolienne

Problématique

La forme donnée aux pales d'une éolienne est un élément important dans la résistance mécanique du dispositif, et sa performance en terme de récupération d'énergie. La figure 1 fournit trois exemples d'éoliennes en production qui montrent que la forme des pales peut être très variée (ainsi que leur nombre).

Figure 1 – Différentes éoliennes.

Il ne s'agit pas ici de s'intéresser à l'optimisation de la forme de la pale en vue d'un critère à définir (résistance, rendement, etc), mais seulement à la description géométrique du bord de la pale, cf. figure 2.

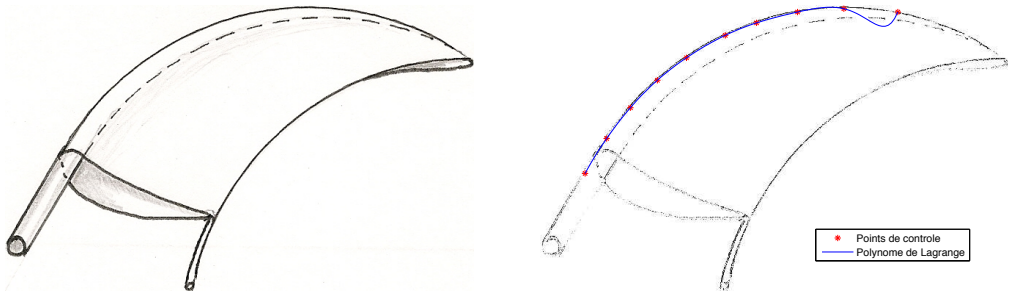


Figure 2 – Coupe d'une pale d'éolienne.

Plus précisément, étant repérés un certain nombre de points sur le bord de la courbe, comment *interpoler* pour obtenir une courbe continue ? L'interpolation de Lagrange est une réponse possible : si des points (x_i, y_i) sont donnés pour $i = 0, 2, \ldots, N$, alors il existe un unique polynôme p, de degré inférieur ou égal à N satisfaisant

$$\forall i = 0, 1, \ldots, N, \quad y_i = p(x_i).$$

Toutefois, il est facile de se convaincre avec quelques essais, que cette solution n'est pas satisfaisante pour la problématique qui nous intéresse, cf. figure 2 à droite.

Modélisation mathématique

Nous allons adopter une autre approche pour le problème d'interpolation. Pour fixer les idées, on suppose que les points sont des clous, et que la courbe recherchée est une tige élastique qu'on positionne successivement entre ceux-là, cf. figure 3.

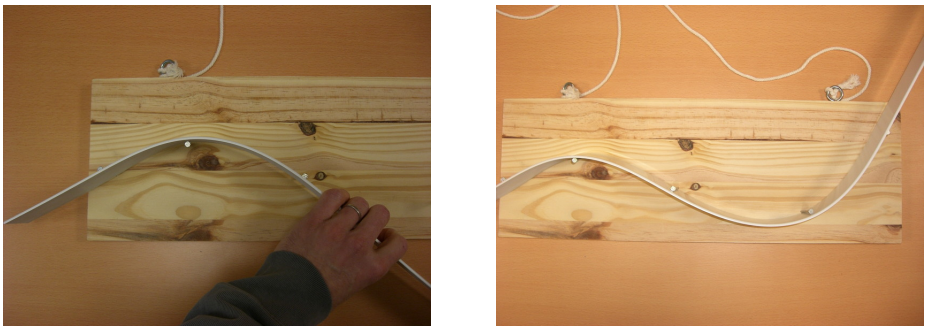


Figure 3 – L'expérience de la tige élastique sur la planche à clous.

La tige prend alors une forme qui paraît satisfaisante comme interpolatrice à partir des points de contrôle. La question est de savoir comment caractériser mathématiquement cette courbe, et comment la calculer numériquement.

On note $\varphi : [0,1] \to \mathbb{R}$ dont le graphe représente la position de la tige à l'équilibre. Ainsi, on a bien sûr

$$\forall i = 0, 1, \dots N, \quad \varphi(x_i) = y_i.$$

Pour simplifier les notations, on suppose dans la suite que $0 = x_0 < x_1 < \cdots < x_N = 1$, ce qui n'est pas une restriction. Par ailleurs, il est naturel de supposer la fonction φ régulière, disons de classe $\mathscr{C}^2$ sur l'intervalle $[0,1]$. Enfin, la tige est rectiligne en dehors de l'intervalle $[0,1]$, ce qui s'exprime par

$$\varphi''(0) = \varphi''(1) = 0.$$

On note donc $\mathcal{A}$ l'ensemble (affine) :

$$\mathcal{A} = \Big\{ \varphi \in \mathscr{C}^2([0,1]) \ ; \ \varphi''(0) = \varphi''(1) = 0 \text{ et } \forall i = 0, 1, \dots N, \quad \varphi(x_i) = y_i \Big\}.$$

Énergie élastique

La position d'équilibre de la tige est caractérisée par la minimisation de son énergie élastique donnée, en première approximation, par

$$\mathcal{E}(\varphi) = \int_0^1 \varphi''(x)^2 \, \mathrm{d}x.$$

Ainsi, la fonction φ recherchée est solution du problème d'optimisation

$$\min \big\{ \mathcal{E}(\varphi) \ ; \ \varphi \in \mathcal{A} \big\}. \tag{1}$$

Condition d'optimalité

N.B. *Cette partie peut être ignorée en première lecture, il suffit d'en retenir que la solution est polynomiale de degré inférieur ou égal à 3 par morceaux, sur chaque intervalle* $[x_i, x_{i+1}]$.

On considère alors une fonction $\psi \in \mathscr{C}^2([0,1])$ satisfaisant

$$\forall i,\ \psi(x_i) = 0 \quad \text{et} \quad \psi''(0) = \psi''(1) = 0. \tag{2}$$

Pour tout réel t, la fonction $\varphi + t\psi$ est dans $\mathcal{A}$ et on peut écrire

$$\int_0^1 \left|\varphi''(x)\right|^2 \mathrm{d}x \leq \int_0^1 \left|\varphi''(x) + t\psi''(x)\right|^2 \mathrm{d}x.$$

En développant, on obtient un trinôme du second degré t de signe constant :

$$t^2 \int_0^1 \left|\psi''(x)\right|^2 \mathrm{d}x + 2t \int_0^1 \varphi''(x)\psi''(x)\,\mathrm{d}x \geq 0.$$

Cela n'est possible pour tout réel t que si le terme d'ordre 1 est nul, ce qui signifie :

$$\int_0^1 \varphi''(x)\psi''(x)\,\mathrm{d}x = 0.$$

Si φ vérifie la condition de régularité par morceaux supplémentaire

$$\forall i = 0, 1, \ldots, N-1, \quad \varphi \in \mathscr{C}^4([x_i, x_{i+1}]),$$

alors on peut intégrer deux fois par parties pour obtenir

$$\int_0^1 \varphi^{(4)}(x)\psi(x)\,\mathrm{d}x = 0.$$

On peut montrer que cette dernière égalité implique que la fonction $\varphi^{(4)}$ est nulle sur chaque intervalle $]x_i, x_{i+1}[$.

On a donc montré que si φ est solution du problème d'optimisation (1) et est de classe $\mathscr{C}^4$ par morceaux, alors elle est polynomiale de degré 3 par morceaux. Cette remarque va nous permettre de réduire la recherche d'une solution à un espace de dimension finie.

Interpolation de Hermite et splines cubiques

Il est facile de vérifier qu'il existe un unique polynôme p (dit de Hermite), de degré inférieur ou égal à 3, satisfaisant

$$p(a) = \alpha, \quad p(b) = \beta, \quad p'(a) = \gamma, \quad p''(a) = \delta, \tag{3}$$

quels que soient les réels $a, b, \alpha, \beta, \gamma, \delta$.

Ainsi, si l'on fixe $\theta \in \mathbb{R}$, le problème de trouver $\varphi_\theta \in \mathscr{C}^2([0,1])$, polynomiale de degré 3 par morceaux et satisfaisant

$$\forall i,\ \varphi_\theta(x_i) = y_i, \quad \varphi_\theta''(0) = 0, \quad \varphi_\theta'(0) = \theta \tag{4}$$

admet une unique solution. En effet, si l'on se place sur l'intervalle $[x_0, x_1]$, il s'agit de trouver un polynôme p_0 de degré inférieur ou égal à 3 satisfaisant

$$p_0(x_0) = y_0, \quad p_0(x_1) = y_1, \quad p_0'(x_0) = \theta, \quad p_0''(x_0) = 0.$$

Ce polynôme p_0 étant fixé, on considère l'intervalle $[x_1, x_2]$ sur lequel φ_θ doit coïncider avec un polynôme p_1 de degré inférieur ou égal à 3 satisfaisant

$$p_1(x_1) = y_1, \quad p_1(x_2) = y_2, \quad p_1'(x_1) = p_0'(x_1), \quad p_1''(x_1) = p_0''(x_1),$$

les deux dernières conditions provenant de la régularité $\mathscr{C}^2$ imposée à la fonction φ_θ. La preuve s'achève par une récurrence immédiate.

Ainsi, pour chaque θ fixé, on a trouvé (de manière constructive, par ailleurs) une fonction φ_θ qui satisfait les conditions (4). Il s'agit maintenant d'ajuster θ afin d'obtenir la condition terminale $\varphi_\theta''(1) = 0$. On peut montrer que cela est possible, assez facilement de surcroît car l'application $F : \mathbb{R} \to \mathbb{R}$ définie par

$$F(\theta) = \varphi_\theta''(1).$$

est affine (non constante). Elle s'annule donc pour une unique valeur de θ.

La figure 4 présente la spline obtenue avec les points de contrôle utilisés précédemment sur le schéma de la pale. On observe ici un bien meilleur résultat qu'avec l'interpolation de Lagrange.

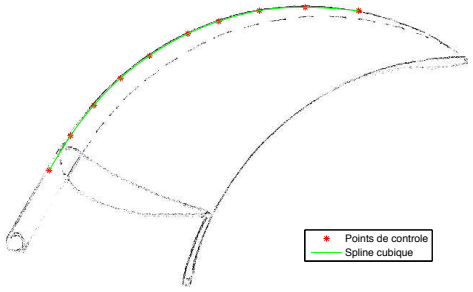


Figure 4 – Interpolation par fonction spline.

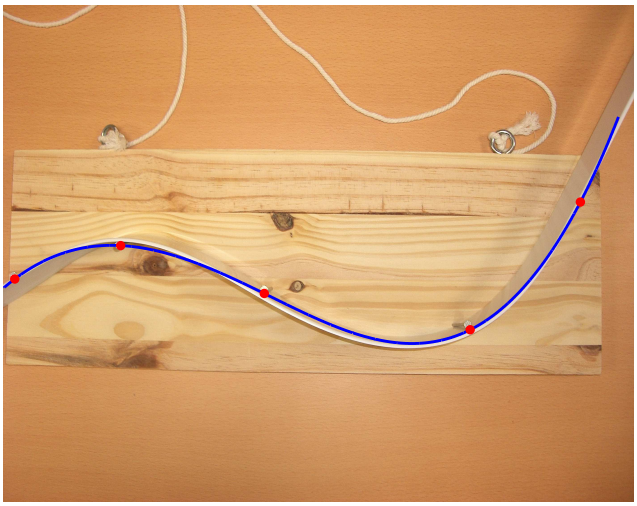


Figure 5 – Comparaison de la spline interpolante et de la tige élastique.

Annexe : programmes Scilab

Calcul du polynôme de Hermite satisfaisant (3)**.**

```
function p=hermite(a,b,alpha,bet,gamm,delta)
Mat=[a^3,a^2,a,1;
b^3,b^2,b,1;
3*a^2,2*a,1,0;
6*a,2,0,0];
scmb=[alpha;bet;gamm;delta];
p=Mat\scmb;
```

Évaluation d'un polynôme de degré inférieur ou éal à 3**.**

```
function val=polyval(p,t)
val=p(1)*t.^3+p(2)*t.^2+p(3)*t+p(4);
```

Calcul de la spline cubique par la méthode de tirs.

```
// les vecteurs x et y sont donnes

N=length(x);

// Calcul de phi_0
d=0;
dd=0;
for i=1:N-1
    a=x(i);b=x(i+1);
    p=hermite(a,b,y(i),y(i+1),d,dd);
    d=3*p(1)*b^2+2*p(2)*b+p(3);
    dd=6*p(1)*b+2*p(2);
end
DDphi_0=dd;

// Calcul de psi_1
d=1;
dd=0;
for i=1:N-1
    a=x(i);b=x(i+1);
    p=hermite(a,b,0,0,d,dd);
    d=3*p(1)*b^2+2*p(2)*b+p(3);
    dd=6*p(1)*b+2*p(2)
end
DDpsi_1=dd;

// Calcul de theta_etoile
theta_etoile=-DDphi_0/DDpsi_1;

// Calcul de phi
d=theta_etoile;
dd=0;
plot(x,y,'r*')
for i=1:N-1
    a=x(i);b=x(i+1);
    p=hermite(a,b,y(i),y(i+1),d,dd);
    d=3*p(1)*b^2+2*p(2)*b+p(3);
```

```
    dd=6*p(1)*b+2*p(2);
    // Trace
    t=linspace(x(i),x(i+1),20);
    Pol=poly(p($:-1:1),'x',"coeff");
    val=horner(Pol,t);
    norm(val-t.^3-t)/norm(val)
    plot(xx(1)+(xx($)-xx(1))*t,yy(1)-mean(yy)*val,'r','LineWidth',2);
end
```