République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ MOHAMED KHIDER, BISKRA

FACULTÉ des SCIENCES EXACTES et des SCIENCES de la NATURE et de la VIE

DÉPARTEMENT DE MATHÉMATIQUES

Mémoire présenté en vue de l'obtention du Diplôme **:**

MASTER en Mathématiques

Option :**Analyse**

Par

Berri Oussama

Titre :

Système tridiagonaux et application à la recherche de fonction splines

Membres du Comité d'Examen :

Dr.	**Houda HASSOUNA**	**UMKB**	**Président**
Pr.	**Naceur KHELIL**	**UMKB**	**Encadreur**
Dr.	**Fatima OUAR**	**UMKB**	**Examinateur**

Juin 2020

Dédicace

Avec l'expression de reconnaissance Je dédie ce modeste travail :

À mon cher grand-père, ma chère grand-mère, ma chère mère, mon cher père, mes chères tantes et mes chers oncles qui m'ont toujours soutenu, leur amour a fait de moi ce que je suis aujourd'hui.

À tous mes chers frères et soeurs .

À tous qui m'ont aidé à affronter les difficultés, qui m'ont soutenu dans la vie et qui ont partagé mes joies et mes peines.

À tous les amis dont je suis fier.

À tous mes professeurs de tous niveaux académiques pour leurs conseils utiles, patience et soutien.....

REMERCIEMENTS

Je tiens tout d'abord à remercier Dr. Naceur KHELIL , professeur à l'Université de Biskra, qui m'a donné ce sujet et m'a encadré pendant mon Master. Il a toujours été à mon écoute et son point de vue complémentaire est souvent été trés utile.

Je suis trés reconnaissant envers Dr Houda HASSOUNA et Dr.Fatima OUAR d'avoir manisfesté de l'interêt pour mon travail en acceptant de participer à ce jury. Leurs remarques et commentaires constructifs m'ont permis d'en améliorer le manuscrit.

Enfin, que toutes les personnes qui ont partcipé de prés ou de loin à la réalisation de ce travail trouvent ici l'expression de mes sincères remerciements.

Mes derniers remerciements vont à mes parents, mes frères et soeurs et tous les amis.

ملخص

الاستقطاب مفيد عند استعمال عدد قليل من المعطيات، فنتحصل على علاقة لها خواص كثير حدود منخفض الدرجة
عند الإخلال بهذا الشرط، يمكن لكثير الحدود هذا آخذ شكل غير واقعي لتمثيل هذه المعطيات
و لتفادي هذا الإشكال نقترح استعمال الشرائح التكعيبية

كلمات مفتاحية: الاستقطاب، جملة ذات ثلاثة أقطار، الشرائح التكعيبية، الشرائح التكعيبية المثبتة

RESUME

L'interpolation est utile si peu de donnees sont utilisées et elles ont egalement un comportement polynomial, de sorte que sa representation est un polynôme approprie et de faible degré. Si ces conditions ne sont pas remplises, le polynôme peut prendre une forme inacceptable pour représenter les données. Pour eviter cette forme, une meilleure option consiste a utiliser les splines cubiques.

Mots clés : Interpolation, système tridiagonal, spline cubique, spline cubique attaché

Table des matières

Dédicace i

Remerciements ii

Table des matières iv

Liste des figures v

Introduction 1

1 Systèmes tridiagonaux 4

1.1 Formulation mathématique et algorithme 4

1.2 Instrumentation informatique 7

2 La spline cubique naturelle 9

2.1 Formulation de la spline naturelle 9

2.2 Instrumentation informatique de la spline cubique naturelle 14

3 La spline cubique (attaché) 19

3.1 Instrumentation informatique de la spline cubique (attaché) 21

Annexe A : Polynômes d'interpolation de Lagrange 29

3.1.1 Construction du polynôme 29

Table des figures

2.1 Polynôme pour chaque intervalle 10
2.2 La pente entre deux polynômes consecutifs 13
2.3 Spline naturelle 17
2.4 Comparaison de l'interpolation et la spline naturelle 18

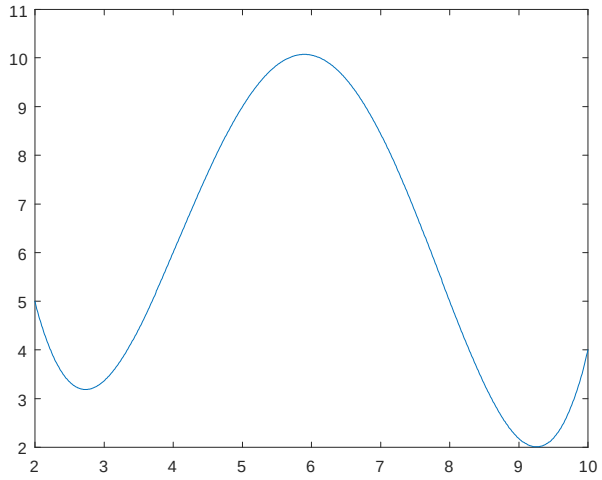
3.1 Graphique de la spline cubique (attaché) avec les points donnés : 24
3.2 Comparaison graphique de l'interpolation, spline naturelle, spline et spline fournit par Matlab 26
3.3 Graphe du polynome d'interploation, pour l'exemple d'application 31

Introduction

Motivation

Le polynôme d'interpolation est utile si peu de données sont utilisées et elles ont également un comportement polynomial, de sorte que sa représentation est un polynôme approprié et de faible degré. Si ces conditions ne sont pas remplies, le polynôme peut prendre une forme inacceptable pour représenter les données, comme illustré dans l'exemple suivant : Le polynôme d'interpolation qui inclut les données suivantes $(2, 5), (4, 6), (5, 9), (8, 5),(10, 4)$ avec la méthode de Lagrange (3.1) est $p(x) = \frac{19}{288}x^4 - \frac{151}{96}x^3 + \frac{1823}{144}x^2 - \frac{118}{3}x + \frac{401}{9}$

Son graphe est illustré ci-dessous



Interpolation de Lagrange

On observe que dans les intervalles $(2, 4)$ et $(8, 10)$ la forme du polynôme n'est pas appropriée pour exprimer la tendance des données.

Une option pourrait être de calculer des polynômes d'interpolation dans des sous intervalles. Par exemple, un polynôme de deuxième degré avec les points $(2, 5), (4.6)$, (5.9) et un autre polynôme du deuxième degré avec les points (5.9), (8.5), (10.4). Cependant, au point intermédiaire $(5, 9)$ où les deux polynômes du deuxième degré se rejoindraient, il y aurait un changement de pente inacceptable.

Une meilleure option consiste à utiliser les splines cubiques. Ce dispositif mathématique est équivalent à la règle flexible que certains dessinateurs utilisent, et qui la permet d'être adaptée pour suivre le chemin des points sur un plan en douceur.

Organisation du mémoire

La partie principale de ce mémoire est composée de trois chapitres, Le mémoire comporte également, un résumé, une introduction et une annexe 3.1
Le chapitre 1 présente une méthode directe pour résoudre un système tridiagonal avec une efficacité de premier ordre : $T(n) = O(n)$(Algorithme de Thomas).

Dans le chapitre 2 nous présentons les splines cubiques naturelles, c'est à dire avec des extrémités détachées. Ce dispositif mathématique est équivalent à la règle flexible que certains dessinateurs utilisent, et qui la permet d'être adaptée pour suivre le chemin des points sur un plan en douceur.
En général dans la spline cubique, les extrémités ne sont plus détachées mais attachées et à une inclinaison spécifiée. Par conséquent, la définition de la spline naturelle ne tient plus, Le chapitre 3 est dédié à cette éude.
Et on termine par une conclusion 3.1.

Chapitre 1

Systèmes tridiagonaux

Dans un système tridiagonal, la matrice des coefficients contient toutes ses composantes égales à zéro sauf dans les trois diagonales principales. Ces systèmes sont présentés dans l'application de certains types de méthodes numériques comme le cas des splines cubiques (l'objet de ma mémoire) et dans la résolution des des équations différentielles par différences finies.

Une méthode directe peut être conçue pour résoudre un système tridiagonal avec une efficacité de premier ordre : $T(n) = O(n)$ ce qui représente une énorme amélioration par rapport aux méthodes directes générales pour résoudre des systèmes d'équations linéaires, dont l'efficacité est $T(n) = O(n^3)$

1.1 Formulation mathématique et algorithme

Un système tridiagonal de n équations exprimées en notation matricielle :

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & 0 & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}$$

Pour obtenir la formulation, on ne peut considérer qu'un système de trois équations et ensuite l'étendre au cas général. Les transformations sont appliquées à la matrice augmentée :

$$\begin{pmatrix} b_1 & c_1 & 0 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ 0 & a_3 & b_3 & d_3 \end{pmatrix}$$

1. soit $w_1 = b_1$.On divise la première ligne par w_1.

$$\begin{pmatrix} 1 & \frac{c_1}{w_1} & 0 & \frac{d_1}{w_1} \\ a_2 & b_2 & c_2 & d_2 \\ 0 & a_3 & b_3 & d_3 \end{pmatrix}$$

2. soit $g_1 = \frac{d_1}{w_1}$. On soustraye de la deuxième ligne, la première ligne multipliée par a_2.

$$\begin{pmatrix} 1 & \frac{c_1}{w_1} & 0 & g_1 \\ 0 & b_2 - a_2\frac{c_1}{w_1} & c_2 & d_2 - a_2 g_1 \\ 0 & a_3 & b_3 & d_3 \end{pmatrix}$$

3. soit $w_2 = b_2 - a_2\frac{c_1}{w_1}$. On divise la deuxième ligne par w_2.

$$\begin{pmatrix} 1 & \frac{c_1}{w_1} & 0 & g_1 \\ 0 & 1 & \frac{c_2}{w_2} & \frac{d_2 - a_2 g_1}{w_2} \\ 0 & a_3 & b_3 & d_3 \end{pmatrix}$$

4. Soit $g_2 = \frac{d_2 - a_2 g_1}{w_2}$. On soustraye de la troisième ligne, la deuxième ligne multipliée par a_3.

$$\begin{pmatrix} 1 & \frac{c_1}{w_1} & 0 & g_1 \\ 0 & 1 & \frac{c_2}{w_2} & \frac{d_2 - a_2 g_1}{w_2} \\ 0 & 0 & b_3 - a_3 \frac{c_2}{w_2} & d_3 - a_3 g_2 \end{pmatrix}$$

5. Soit $w_3 = b_3 - a_3 \frac{c_2}{w_2}$. On divise la troisième ligne par w_3.

$$\begin{pmatrix} 1 & \frac{c_1}{w_1} & 0 & g_1 \\ 0 & 1 & \frac{c_2}{w_2} & g_2 \\ 0 & 0 & 1 & \frac{d_3 - a_3 g_2}{w_3} \end{pmatrix}.$$

6. Soit $g_3 = \frac{d_3 - a_3 g_2}{w_3}$. Enfin, on obtient :

$$\begin{pmatrix} 1 & \frac{c_1}{w_1} & 0 & g_1 \\ 0 & 1 & \frac{c_2}{w_2} & g_2 \\ 0 & 0 & 1 & g_3 \end{pmatrix}$$

Où la solution peut être trouvée directement :

$$x_3 = g_3$$

$$x_2 = g_2 - \frac{c_2}{w_2} x_3$$

$$x_1 = g_1 - \frac{c_1}{w_1} x_2$$

La formulation s'étend au cas général (algorithme de Thomas)

Transformation matricielle d'un système tridiagonal de n équations linéaires

$w_1 = b_1$

$g_1 = \frac{d_1}{w_1}$

$w_i = b_i - \frac{a_i c_{i-1}}{w_{i-1}}, \quad i = 2, 3, ..., n$

$g_i = \frac{d_i - a_i g_{i-1}}{w_i}, \quad i = 2, 3, ..., n.$

Obtention de la solution

$x_n = g_n$

$x_i = g_i - \frac{c_i x_{i+1}}{w_i}, \quad i = n-1, n-2, ..., 2, 1$

L'algorithme comprend un cycle en fonction de la taille du problème n pour réduire la matrice et un autre cycle externe dépendant de n pour obtenir la solution. C'est donc un algorithme efficace $T(n) = O(n)$.

1.2 Instrumentation informatique

Avec la formulation précédente, une fonction est écrite pour résoudre un système tridiagonal de n équations linéaires. La fonction reçoit les coefficients et les constantes dans les vecteurs a, b, c, d. La solution est livrée dans le vecteur x.

```
function x = tridiagonal (a, b, c, d)
n = length(d);
w (1) = b (1);
g (1) = d (1) / w (1);
for i = 2 : n       % Transformation matricielle
w (i) = b (i) - a (i) * c (i-1) / w (i-1);
g (i) = (d(i) - a (i) * g (i-1)) / w (i);
end
x (n) = g (n);      % Obtention de la solution
for i = n-1 : -1 : 1
x (i) = g (i) - c (i) * x (i + 1) / w (i);
end
```

Application

Résoudre le système tridiagonal d'équations linéaires suivant en utilisant la fonction ci-dessus

$$\begin{bmatrix} 7 & 5 & & \\ 2 & -8 & 1 & \\ & 6 & 4 & 3 \\ & & 9 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 6 \\ 5 \\ 7 \\ 8 \end{bmatrix}$$

```
>> a=[0 2 6 9];
>> b=[7 -8 4 8];
>> c=[5 1 3 0];
>> d=[6 5 7 8];
>> x=tridiagonal(a,b,c,d)
x =
0.7402
0.1637
4.8288
-4.4324
```

Chapitre 2

La spline cubique naturelle

Étant donné les points $(x_i, y_i), i = 1, 2, ..., n$, la spline cubique naturelle est un ensemble de $n-1$ polynômes de degré **trois** placés un par un entre chaque paire de points consécutifs, de telle manière qu'il y a continuité, en conservant la même pente et la même courbure avec les polynômes d'intervalles adjacents.

Définition 2.0.1 *spline cubique naturelle* $T(x)$

$$T(x) = \begin{cases} a_1 (x - x_1)^3 + b_1 (x - x_1)^2 + c_1 (x - x_1) + d_1, ...x_1 \leq x \leq x_2 \\ a_2 (x - x_2)^3 + b_2 (x - x_2)^2 + c_2 (x - x_2) + d_2, ...x_2 \leq x \leq x_3 \\ \\ a_{n-1} (x - x_{n-1})^3 + b_{n-1} (x - x_{n-1})^2 + c_2 (x - x_{n-1}) + d_{n-1}, ...x_{n-1} \leq x \leq x_n \end{cases}$$

2.1 Formulation de la spline naturelle

Polynôme pour chaque intervalle :

$$y = p(x) = a_i (x - x_i)^3 + b_i (x - x_i)^2 + c_i (x - x_i) + d_i \qquad x_i \leq x \leq x_{i+1} \quad \forall i = 1, 2..., n-1 \tag{2.1}$$

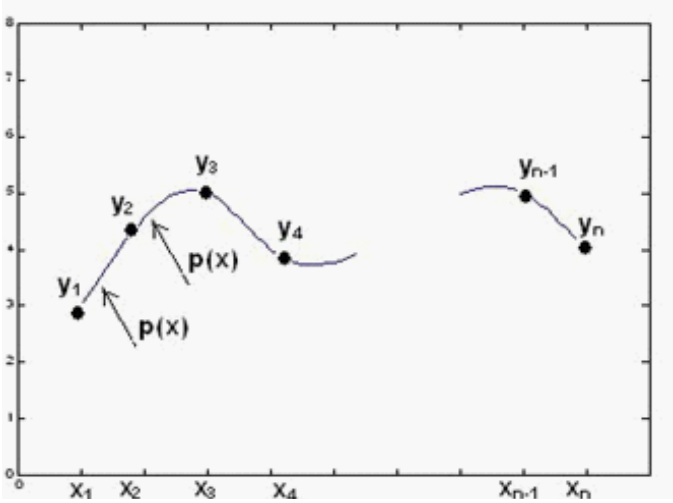


FIG. 2.1 – Polynôme pour chaque intervalle

Pour chacun de ces polynômes, les coefficients doivent être déterminés

$$a_i, b_i, c_i, d_i, i = 1, 2, ..., n-1$$

Les points ne sont pas nécessairement régulièrement espacés, donc donnons un nom pour chacune des distances entre des points consécutifs :

$$h_i = x_{i+1} - x_i, \quad i = 1, 2, ..., n-1$$

Le développement suivant basé sur les conditions requises pour $p(x)$ permet d'obtenir les coefficients polynomiaux.

Soit $y = p(x)$ le polynôme dans tout intervalle $i, i = 1, 2, ..., n-1$

Ce polynôme doit inclure les extrémités de chaque intervalle i :

$$x = x_i : \; y_i = a_i (x - x_i)^3 + b_i (x - x_i)^2 + c_i (x - x_i) + d_i \implies \quad d_i = \; y_i \tag{2.2}$$

$$\begin{cases} x = x_{i+1} : \; y_{i+1} = a_i \left(x_{i+1} - x_i\right)^3 + b_i \left(x_{i+1} - x_i\right)^2 + c_i \left(x_{i+1} - x_i\right) + d_i \\ \qquad\qquad\qquad\qquad = \; a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i \end{cases} \tag{2.3}$$

Les deux premières dérivées de $y = p(x)$

$$y' = 3a_i \left(x - x_i\right)^2 + 2b_i \left(x - x_i\right) + c_i \tag{2.4}$$

$$y'' = 6a_i \left(x - x_i\right) + 2b_i \tag{2.5}$$

Par souci de simplicité, la notation suivante est utilisée pour la dérivée seconde

$$y'' = S = 6a_i \left(x - x_i\right) + 2b_i$$

Nous évaluons la dérivée seconde aux extrémités de l'intervalle i :

$$x = x_i : \; y_i'' = S_i = 6a_i \left(x_i - x_i\right) + 2b_i = 2b_i \implies \quad b_i = \frac{S_i}{2} \tag{2.6}$$

$$x = x_{i+1} : \; y_{i+1}'' = S_{i+1} = 6a_i \left(x_{i+1} - x_i\right) + 2b_i = 6a_i h_i + 2b_i$$

D'où, il vient

$$a_i = \frac{S_{i+1} - S_i}{6h_i} \tag{2.7}$$

Substituons 2.2, 2.6 et 2.7 dans 2.3

$y_{i+1} = \frac{S_{i+1}-S_i}{6h_i}h_i^3 + \frac{S_i}{2}h_i^2 + c_i h_i + y_i$

D'où vous obtenez

$$c_i = \frac{y_{i+1} - y_i}{h_i} \; - \frac{2h_i S_i + h_i S_{i+1}}{6} \tag{2.8}$$

Ainsi les coefficients de $p(x)$ sont exprimés par les données données et les valeurs des dérivées secondes S

Coefficients de la spline naturelle

$$a_i = \frac{S_{i+1} - S_i}{6h_i}, \quad b_i = \frac{S_i}{2} \; , \quad c_i = \frac{y_{i+1} - y_i}{h_i} \; - \frac{2h_i S_i + h_i S_{i+1}}{6} \tag{2.9}$$

Au point intermédiaire entre deux intervalles adjacents, la pente des polynômes doit être la même :

Pente dans l'intervalle $[x_i, x_{i+1}]$, à partir de 2.4 :

$y' = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i$

Evaluons en l'extrimité gauche

$x = x_i : \quad y_i' = 3a_i(x_i - x_i)^2 + 2b_i(x_i - x_i) + c_i = c_i$

Pente dans l'intervalle $[x_{i-1}, x_i]$, à partir de 2.4 :

$y' = 3a_{i-1}(x - x_{i-1})^2 + 2b_i(x - x_{i-1}) + c_{i-1}$

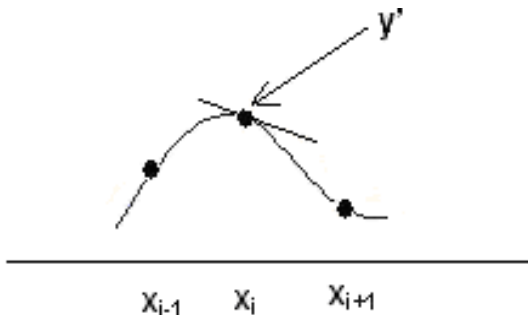


FIG. 2.2 – La pente entre deux polynômes consecutifs

Evaluons en l'extrimité droite

$x = x_i : \quad y_i' = 3a_{i-1}(x_i - x_{i-1})^2 + 2b_{i-1}(x_i - x_{i-1}) + c_{i-1} = 3a_{i-1}h_{i-1}^2 + 2b_{i-1}h_{i-1} + c_{i-1}$

Au point x_i, les deux pentes doivent avoir la même valeur :

$c_i = 3a_{i-1}h_{i-1}^2 + 2b_{i-1}h_{i-1} + c_{i-1}$

Enfin, les définitions de sont remplacées c_i, a_{i-1}, b_{i-1}, c_{i-1}

$$\frac{y_{i+1}-y_i}{6h_i} - \frac{2h_iS_i+h_iS_{i+1}}{6} = 3\left(\frac{S_{i+1}-S_i}{6h_i}\right)h_{i-1}^2 + 2\left(\frac{S_{i-1}}{2}\right)h_{i-1} + \frac{y_{i+1}-y_i}{h_i} - \frac{2h_{i-1}S_{i-1}+h_{i-1}S_i}{6}$$

Après avoir simplifié, vous obtenez :

$$h_{i-1}S_{i-1}+2\left(h_{i-1}+h_i\right)S_i+h_iS_{i+1} = 6\left(\frac{y_{i+1}-y_i}{h_i} - \frac{y_i-y_{i-1}}{h_{i-1}}\right), \quad i = 2,3,...,n-1 \quad (2.10)$$

Cette équation doit être évaluée avec les données données, avec lesquelles un système n - 2 équations linéaires avec les n variables : $S_1, S_2, ..., S_n$ est obtenu.

Pour obtenir deux données supplémentaires, on considère que dans la spline naturelle, les points finaux initial et final sont lâches et n'ont donc pas de courbure. Avec cette hypothèse, la valeur de la dérivée seconde a une valeur nulle aux extrêmes et peut s'écrire :

$$S_1 = 0, S_n = 0 \quad (2.11)$$

Algorithme de la spline cubique naturelle

Compte tenu des points : $(x_i, y_i), i = 1, 2, ..., n$

1. Avec l'équation 2.10 et en remplaçant les valeurs données en 2.11, obtenir un système d'équations linéaires $n-2$ avec les inconnues $S_2, S_3, ..., S_{n-1}$, **(Système tridiagonal d'équations linéaires).**
2. Résoudre le système et obtenir les valeurs de $S_2, S_3, ..., S_{n-1}$
3. Avec les définitions données en 2.9, obtenir les coefficients du traceur cubique.
4. Remplacer les coefficients dans la définition donnée en 2.1 et obtenir le polynôme de la spline cubique dans chacun des intervalles.

2.2 Instrumentation informatique de la spline cubique naturelle

La formulation de la spline cubique naturelle a été instrumentée dans MATLAB en utilisant une fonction appelée SCN qui fournit un vecteur avec des points de la courbe ou les vecteurs avec les coefficients : a, b, c, d. Il existe une propre version de MATLAB équivalente à cette dernière fonction et elle est appelée spline

Le système résultant étant tridiagonal, une méthode spécifique très efficace est utilisée pour résoudre ces systèmes. Il doit y avoir au moins 4 points de données.

function[a,b,c,d] = SCN (x,y,z)

% spline cubique naturelle : $a(i)(x-x(i))\hat{}3+b(i)(x-x(i))\hat{}2+c(i)(x-x(i))+d(i), n > 3$

% z est facultatif : c'est le vecteur de points pour évaluer la spline

% Renvoie les points de la spline ou les coefficients des polynômes segmentaires

n = length(x) ;

clear A B C D ;

if n < 4

```
return
end
for i = 1 : n-1
h(i) = x (i + 1)-x (i) ;
end
s(1) = 0 ;
s (n) = 0 ;
B (1) = 2 * (h(1) + h(2)) ;
C (1) = h(2) ;
D (1) = 6 * ((y(3) -y(2)) / h(2) - (y(2) -y(1)) / h(1)) - h(1)* s( 1) ;
for i = 2 : n-3      % Système tridiagonal pour obtenir S
A(i) = h(i) ;
B(i) = 2*(h(i) + h(i+1)) ;
C(i) = h(i+1) ;
D(i) = 6 *((y(i+2)-y(i+1)) / h(i+1)-(y(i+1)-y(i))/h(i)) ;
end
A(n-2)= h(n-2) ;
B (n-2) = 2*(h(n-2)+h(n-1)) ;
D(n-2) = 6*((y(n) -y(n-1)) / h(n-1)-(y(n-1)-y(n-2)) / h(n -2))- h (n-1) * s (n) ;
u = tridiagonal(A,B,C,D) ;
for i = 2 : n-1
s (i) = u(i-1) ;
end
for i= 1 :n-1      % Coefficients du traceur cubique naturel
a(i) = (s(i + 1)-s (i)) / (6*h(i)) ;
b(i) = s(i) / 2 ;
c (i) = (y(i+1) -y(i)) / h(i)-(2* h(i)* s(i) + h(i) * s(i + 1)) / 6 ;
```

```
d(i) = y(i);
end
if nargin == 3      % Points de traceur cubique naturel
p = [];
m = length(z);
for k = 1 : m
t = z(k);
for i = 1 : n-1
if t>= x(i) & t <= x(i+1)
p(k) = a(i)*(t-x(i))^3 + b(i)*(t-x(i))^2+c(i)*(t-x(i))+d(i);
end
end
end
if m> 1
k = m; i=n-1;
p(k) = a(i) * (t-x(i))^3 + b(i)*(t-x(i))^2+ c(i)*(t-x(i))+ d(i);
end
clear a b c d;
a = p;
end
```

Application

```
x = [2 4 5 8 10];
y = [5 6 9 5 4];
z = [2 : 0.01 : 10];
p = SCN (x, y, z);
plot (x, y, 'o');
```

hold on ;

plot(z,p)

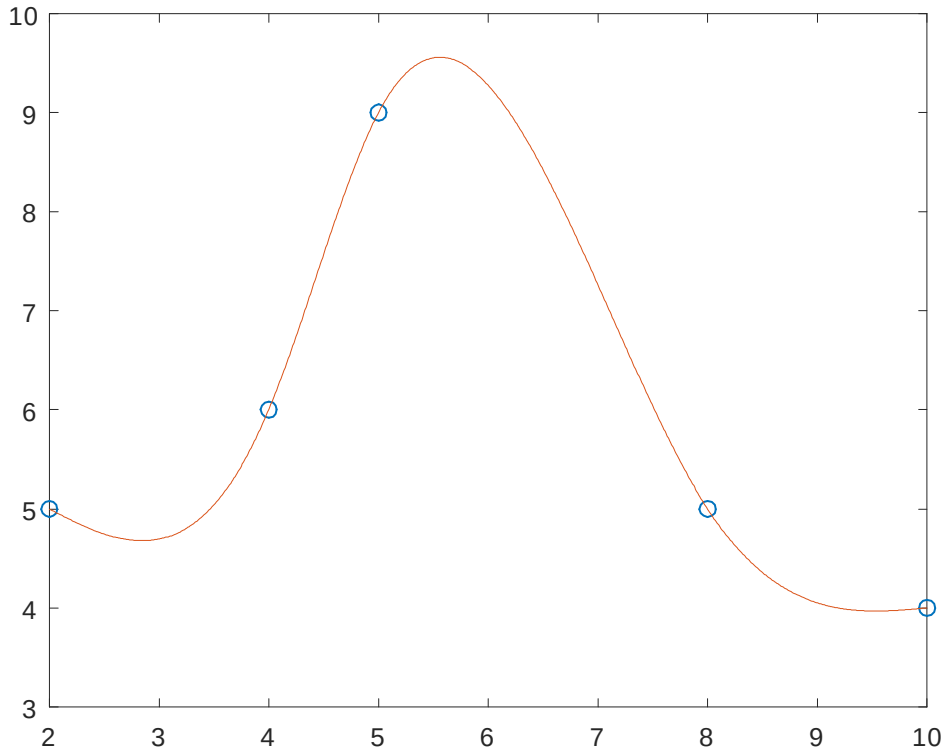


FIG. 2.3 – Spline naturelle

[a, b, c, d] = SCN (x, y) ;

a =

0,2684 -1,2580 0,3403 -0.1498

b =

0 1,6106 -2,1635 0,8990

c =

-0,5737 2,6474 2.0946 -1.6987

d =

5 6 9 5

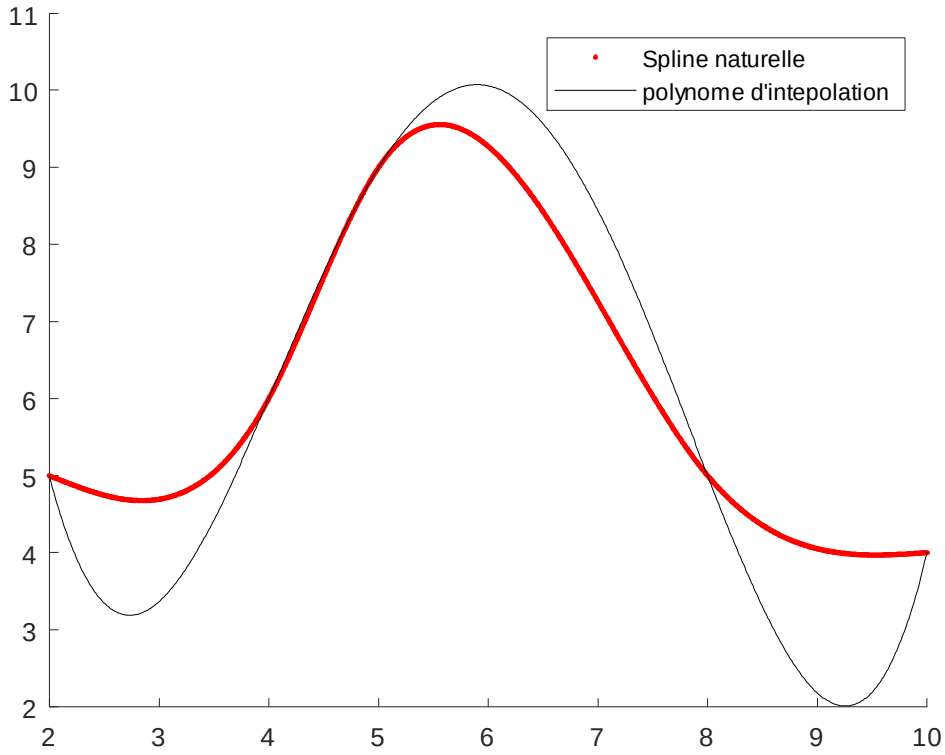


FIG. 2.4 – Comparaison de l'interpolation et la spline naturelle

Chapitre 3

La spline cubique (attaché)

En général dans la spline cubique, les extrémités ne sont plus détachées mais attachées et à une inclinaison spécifiée. Par conséquent, la définition ci-dessus (2.11) : $S_1 = 0, S_n = 0$ ne tient plus

Compte tenu des points : $(x_i, y_i), i = 1, 2, ..., n$. De plus, l'inclinaison de la spline aux extrémités spécifiée comme suit :

$$\begin{cases} y\prime(x_1) = u \\ y\prime(x_n) = v \end{cases}$$

Nous utilisons l'expression (2.4) de l'étude précédente :

$$y' = 3a_i (x - x_i)^2 + 2b_i (x - x_i) + c_i$$

Nous substituons les données données aux polynômes dans le premier et le dernier intervalle :

Dans le premier intervalle :

$$x = x_1 : \qquad y\prime(x_1) = u = 3a_1(x_1 - x_1)2 + 2b_1(x_1 - x_1) + c_1 = c_1$$

La définition de c_1 et on a :

$$u = \frac{y_2 - y_1}{h_1} - \frac{2h_1S_1 + h_1S_2}{6}$$

Finalement,

$$-\frac{1}{3}h_1S_1 - \frac{1}{6}h_1S_2 = u - \frac{y_2 - y_1}{h_1} \tag{3.1}$$

Dans le dernier intervalle :

$$x = x_n : \qquad y\prime(x_n) = v = 3a_{n-1}(x_n - x_{n-1})2 + 2b_{n-1}(x_n - x_{n-1}) + c_{n-1}$$

$$v = 3a_{n-1}h_{n-1}^2 + 2b_{n-1}h_{n-1} + c_{n-1}$$

Les définitions des coefficients sont remplacées :

$$v = 3(\frac{S_n - S_{n-1}}{6h_{n-1}})h_{n-1}^2 + 2(\frac{S_{n-1}}{2})h_{n-1} + \frac{y_n - y_{n-1}}{h_{n-1}} - \frac{2h_{n-1}S_{n-1} + h_{n-1}S_n}{6}$$

Alors

$$v = (\frac{S_n - S_{n-1}}{2})h_{n-1} + S_{n-1}h_{n-1} + \frac{y_n - y_{n-1}}{h_{n-1}} - \frac{2h_{n-1}S_{n-1} + h_{n-1}S_n}{6}$$

Finalement

$$\frac{1}{6}S_{n-1}h_{n-1} + \frac{1}{3}h_{n-1}S_n = v - \frac{y_n - y_{n-1}}{h_{n-1}} \tag{3.2}$$

Le équations (3.1) et (3.2) ainsi que les équations obtenues à partir de (2.10) constituent un système tridiagonal de n équations linéaires avec n variables $S_1, S_2, ..., S_n$.

Le reste de la procédure est similaire à celui correspondant au spline cubique naturelle.

Algorithme de la spline cubique

Compte tenu des points : $(x_i, y_i), i = 1, 2, ..., n$

1. Avec les équations (2.10), (3.1) et (3.2) obtenir un système linéaire de n équations avec les inconnues $S_1, S_3, ..., S_n$, **(Système tridiagonal d'équations linéaires).**
2. Résoudre le système et obtenir les valeurs de $S_1, S_3, ..., S_n$
3. Avec les définitions données en 2.9, obtenir les coefficients du splinec ubique.
4. Remplacer les coefficients dans la définition donnée en 2.1 et obtenir le polynôme de la spline cubique dans chacun des intervalles.

En définitive :

$$\begin{cases} -\frac{1}{3}h_1S_1 - \frac{1}{6}h_1S_2 = u - \frac{y_2-y_1}{h_1} \\ h_{i-1}S_{i-1} + 2\left(h_{i-1}+h_i\right)S_i + h_iS_{i+1} = 6\left(\frac{y_{i+1}-y_i}{h_i} - \frac{y_i-y_{i-1}}{h_{i-1}}\right) \ , \quad i = 2,3,...,n-1 \\ \frac{1}{6}S_{n-1}h_{n-1} + \frac{1}{3}h_{n-1}S_n = v - \frac{y_n-y_{n-1}}{h_{n-1}} \\ a_i = \frac{S_{i+1}-S_i}{6h_i} \\ b_i = \frac{S_i}{2} \ , \\ c_i = \frac{y_{i+1}-y_i}{h_i} \ - \frac{2h_iS_i+h_iS_{i+1}}{6} \\ d_i = y_i \quad i = 1,2,...,n-1 \\ y = p(x) = a_i\left(x-x_i\right)^3 + b_i\left(x-x_i\right)^2 + c_i\left(x-x_i\right) + d_i \quad x_i \le x \le x_{i+1} \ \ \forall i = 1,2...,n-1 \end{cases}$$

3.1 Instrumentation informatique de la spline cubique (attaché)

La formulation du spline cubique en question a été instrumentée dans MATLAB en utilisant une fonction appelée SCG qui fournit un vecteur avec des points de la spline ou des vecteurs avec des coefficients : a b c d.

```
function [a, b, c, d] = SCG (x, y, u, v, z)
 % Spline cubique
 %u, v sont les pentes aux extrêmes
 % a (i) (zx (i)) ^3 + b (i) (zx (i)) ^2 + c (i) (zx (i)) + d ( i), n> 3
 % z est optionnel : c'est le vecteur de points pour évaluer la spline
```

```
% Retourne les points du spline ou les coefficients des polynômes segmentaires
n = 5 %longueur (x);
clear A B C D;
if n < 4
return
end
for i = 1 : n-1
h (i) = x (i + 1) -x (i);
end
B (1) = - 2 * h (1) / 6;
C (1) = - h (1) / 6;
D (1) = u- (y (2) -y (1)) / h (1);
for i = 2 : n-1
A (i) = h (i-1);
B (i)= 2 * (h (i-1) + h (i)); C (i) = h (i);
D (i) = 6 * ((y(i + 1) -y (i)) / h (i) - (y(i) -y(i-1)) / h(i-1));
end
A (n) = h(n-1) / 6;
B(n) = h(n-1) / 3;
D (n) = v-(y(n) -y(n-1)) / h(n-1);
s = tridiagonal (A, B, C, D);
for i = 1 : n-1 % Coefficients du spline cubique en question
a (i) = (s (i + 1) -s (i)) / (6 * h (i)); b (i) = s(i) / 2;
c (i) = (y (i + 1) -y (i)) / h (i) - (2 * h (i) * s(i) + h (i) * s (i + 1)) / 6;
d (i) = y (i);
end
if nargin == 5
```

```
% Points du spline cubique
p = [];
m=length(z);;
for k = 1 : m
t = z (k);
for i = 1 : n-1
if t>= x (i) & t <= x (i + 1)
p(k) = a(i)*(t-x(i))^3 + b (i) * (t-x (i))^2 + c (i) *(t-x (i))+d (i);
end
end
end
if m>1
k = m; i = n-1;
p (k) = a (i) * (t-x (i))^3 + b (i)*(t-x (i))^2 + c(i)*(t-x (i))+d(i);
end
clear a b c d;
a = p;
end
```

Exemple. Trouvez la spline cubique en utilisant la fonction ci-dessus pour les points suivants. (2, 5), (4.6), (5.9), (8.5), (10.4). À l'extrêmité gauche doit se pencher de 45° et il devrait se terminer horizontalement à l'extrême droite

```
>> x = [2 4 5 8 10];
y = [5 6 9 5 4];
z = [2 : 0.01 : 10]; %Points pour évaluer la spline
```

p = SCG(x, y, 1, 0, z) ; %points de la spline

plot(x, y, 'o') ; %graphique des points

hold on

plot(z, p) %graphique de la spline

axis([0,10,0,10]) ;

Graphique de la spline cubique attaché avec les points donnés :

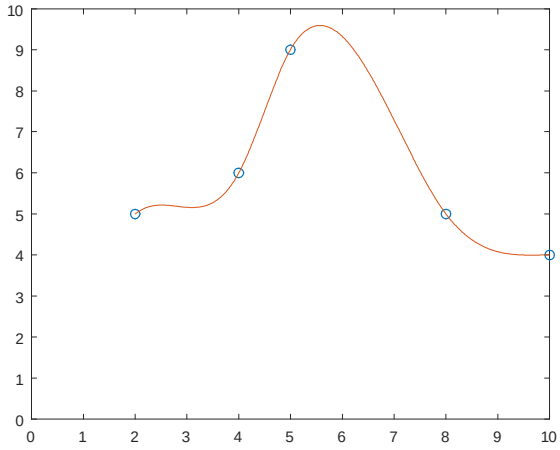


FIG. 3.1 – Graphique de la spline cubique (attaché) avec les points donnés :

Coefficients des polynômes segmentaires de la spline cubique (attaché) :

[a, b, c, d] = SCG (x, y, 1, 0)

a =

0.5870 − 1.4461 0.3535 − 0.1728

b =

−1.4240 2.0980 − 2.2402 0.9412

c =

1.0000 2.3480 2.2059 − 1.6912

d =

5 6 9 5

Exemple comparatif

Comparaison graphique des interpolations polynomiaux et cubiques d'interpolation, y compris le spline cubique fourni par la fonction spline de MATLAB

Données : (2, 7), (4, 8), (5, 9), (6, 8), (8.6)

x = [2 4 5 6 8] ;

y = [7 8 9 8 6] ;

plot (x, y, 'o'), grid on, hold on

p = lagrange(x, y) ;

ezplot (p, [2,8])

z = [2 : 0.01 : 8] ;

v = SCN (x, y, z) ;

plot (z, v)

w = spline (x, y, z) ;

plot (z, w)

t = SCG (x, y, 0,0, z) ;

plot (z, t)

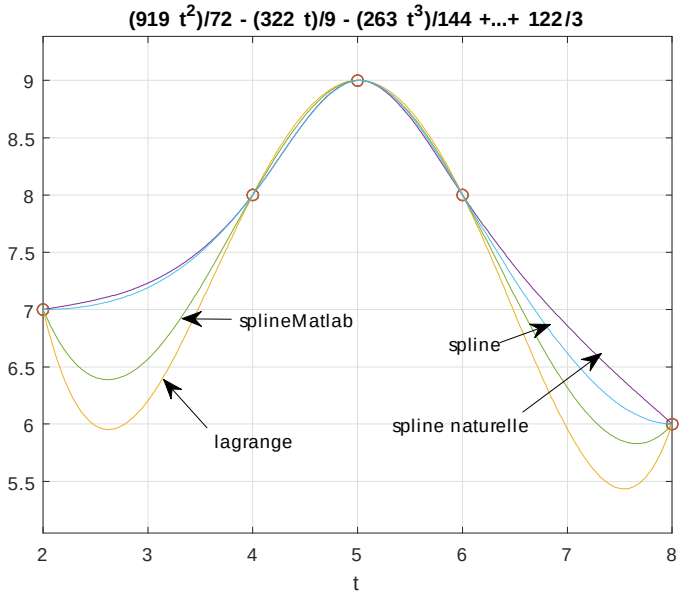


FIG. 3.2 – Comparaison graphique de l'interpolation, spline naturelle, spline et spline fournit par Matlab

Conclusion

En conclusion, lorsque on utilise la spline cubique naturelle, on observe une amélioration remarquable de la représentation des données données.

De plus la spline cubique maintient une courbure lisse et continue et suit mieux la tendance des points.

Bibliographie

[1] Gerard, C., Aplied Numerical Analysis. Addison-Wesley. Publishing Company.

[2] Camahan B., Luther H., Wilkes J., Applied Numerical Methods, John Wily & Sons, Inc.

[3] The Mathworks, Inc. Using MATLAB Computation, Visualisation, Programming

Annexe A : Polynômes d'interpolation de Lagrange

3.1.1 Construction du polynôme

Supposons que pour $(n+1)$ valeurs distincts de l'argument $x_0, x_1, ...x_n$ données sur le segment $[a, b]$ on connaisse les valeurs correspondantes de la fonction $y = f(x)$; $f(x_0) = y_0$, $f(x_1) = y_1$,...,$f(x_n) = y_n$.

Il existe un polynôme unique $L_n(x) = \sum_{i=0}^{n} P_i(x) f(x_i)$ vérifiant $L_n(x_i) = y_i$ pour $i = \overline{0, n}$ (**Conditions d'interpolation**), avec $P_i(x) = \frac{\prod_{k=0}^{n}(x - x_k)}{\prod_{k=0}^{n}(x_i - x_k)}, k \neq i.$

Programme qui calcule le polynôme de Lagrange

% x, f : points de base pour l'interpolation

% v : valeur à interpoler (paramètre facultatif). Cela peut être un vecteur

function p = lagrange (x, f, v)

n=length(x) ;

syms t ; % Variable pour le polynôme

p = 0 ;

for i = 1 : n

L = 1 ;

for j = 1 : n

```
if i ~= j
L = L * (t-x (j)) / (x (i) -x (j));
end
end
p = p + L * f (i);        % obtention de p(t) symboliquement
end
p = expand (p);        % simplification algébrique
if nargin == 3        % vérifie s'il y a un paramètre supplémentaire
t = v;
p = eval(p);        % obtention du résultat de p évalué en v
end
```

Application

```
x = [2 4 5 6 8];
y = [7 8 9 8 6];
p = lagrange(x, y);
ezplot (p, [2,8])
```

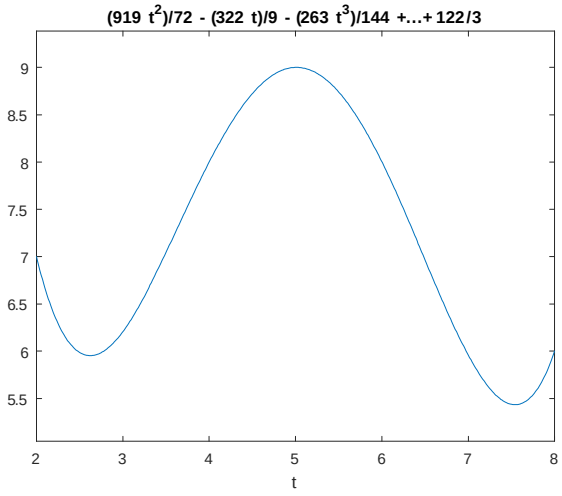


FIG. 3.3 – Graphe du polynome d'interploation, pour l'exemple d'application