Interpolation : Splines

7 juillet 2014

Université de Ouagadougou

Unité de Formation et de Recherche en Sciences Exactes et Appliquées

LAboratoire National d'Informatique et de BIOmathématiques (LANIBIO)

Mathématiques Appliquées Informatique et Modélisation en Environnement (MAIME)

MASTER 1

Enseignant : Pr Ousséni SO

Projet MATLAB numéro 16 : Interpolation : Splines

Etudiant : Kevin Cédric Yiwènin BAMOUNI.

Chapitre 1

Présentation théorique

Notations

- $L^2([a,b])$: Ensemble des fonctions f telles que $\int_a^b (f''(x))^2 dx$ est défini.
- $D^{-2}L^2([a,b])$: Ensemble des fonctions dont la dérivée seconde existe et est dans $L^2([a,b])$.
- $0:n$ de *zéro* à n par pas de 1.
- $\omega_j = (x_j, y_j)_{j=0:n}$ les point ω de coordonnées (x,y).

1.1 Introduction à l'interpolation polynomiale

Pour évaluer une fonction, il est fréquent de ne disposer que de ses valeurs sur un ensemble fini de $(n+1)$ points $\omega_j = (x_j, y_j)_{j=0:n}$. Il est souvent nécéssaire de pouvoir évaluer, approcher cette fonction en un point se situant à l'intérieur d'un intervalle élémentaire borné de deux points consécutifs ω_j et ω_{j+1}, une procédure d'approximation, consiste à approcher la fonction par une autre fonction plus simple, notamment un polynôme : c'est l'interpolation polynomiale.

Mathématiquement, l'interpolation polynomiale de $(n+1)$ points $\omega_j = (x_j, y_j)_{j=0:n}$ consiste à trouver un polynôme p de dégré n tel que :

$$p(x_j) = y_j, \forall\, j = 0:n. \tag{1.1.1}$$

Ainsi le polynôme p d'interpolation sera :

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \tag{1.1.2}$$

Avec

$$y_j = p(x_j) = a_0 + a_1 x_j + a_2 x_j^2 + a_3 x_j^3 + \ldots + a_n t_j^n \quad 0 \leq j \leq n \tag{1.1.3}$$

où les $a_0, a_1, a_2, \ldots, a_n$ sont les coeffiscients du polynôme (pouvant être déterminés par différentes méthodes d'interpolation polynomiale).

1.2 Interpolation polynomiale par morceaux : Splines

Le but de cette approche est de conserver la forme polymiale, tout en évitant toute augmentation du degré du polynôme. le principe est de fabriquer une succession de polynôme, appelés **splines**, de degré peu élevé k, pour le cas générale, tels que les raccordements entre les polynômes soient le plus régulier possible. L'interpolation polynomiale la plus basique consiste à utiliser des polynômes de degré $k = 1$, ce qui reviendrais à relier les points ω_j par des segments. Cependant l'interpolant est juste continue mais pas dérivable.

Ainsi pour obtenir une fonction plus régulière tout en ayant un degré des polynômes peu élevé, une implémentation du principe d'interpolation polynomiale par morceaux consiste à utiliser des polynômes de degré $k = 3$, appelés **spline cubiques d'interpolation.**

Définition. (approche polynomiale par morceaux)

On appelle spline cubique σ de noeud $(x_j)_{j=0:n}$ toute fonction réelle de variable réelle, qui est deux fois continûment dérivable sur $\mathbb{R}$ et dont le restriction à chaque intervalle $[x_j, x_{j+1}]$, ainsi que $]-\infty, x_0]$ et $[x_n, +\infty[$ est un polynôme de degré $k = 3$.

Ainsi pour toute suite de points $\omega_j = (x_j, y_j)_{j=0:n}$ on peut toujours trouver une et une seule fonction σ qui vérifie $\forall j \in [0, n]\,,\ \sigma(x_j) = y_j$.

En effet considérons la restriction de σ à $[x_0, x_1]$; c'est un polynôme de degré 3, qui vérifie (1) : $\sigma(x_0) = y_0$, (2) : $\sigma'(x_0) = y'_0$, (3) : $\sigma''(x_0) = y''_0$ (supposés connus) et (4) : $\sigma(x_1) = y_1$. Les trois premières conditions permettent d'affirmer que $\forall x \in [x_0, x_1]\,, \quad \sigma(x) = y_0 + y'_0(x - x_0) + \frac{y''_0}{2}(x - x_0)^2 + \alpha(x - x_0)^3$ où α est une constante déterminé à l'aide de la condition (4) : $\sigma(x_1) = y_1$, (si $h_0 = x_1 - x_0,\ \alpha = (y_1 - y_0)/h_0^3 - y'_0/h_0^2 - y''_0/(2h_0)$). On connait donc σ sur $[x_0, x_1]$, et comme par hypothèse σ' et σ'' sont continues en x_1, on connait aussi σ' et σ'' à gauche et à droite de x_1. Ainsi en appliquant le raisonnement tenu sur $[x_0, x_1]$, successivement sur les intervalles jusqu'à $[x_{n-1}, x_n]$, on en déduit σ sur $[x_0, x_n]$. En résumé, d'une part peuvent être fixés arbitrairement et d'autre part il faut déterminer un critère pour définir σ sur $]-\infty, x_0]$et sur $[x_n, +\infty[$.

Un critère simple pour prolonger σ sur $]-\infty, x_0]$et sur $[x_n, +\infty[$ est de choisir des polynômes de plus bas degré possible tels que $\sigma \in C^2$ (un polynôme de degré 2 par exemple). En ce qui concerne les conditions aux deux bouts x_0 et x_n trois types de conditions sont couramment utilisés et conduisent à des fonctions stables :

1. **"Splines périodiques" :** on utilise les conditions $\sigma''(x_0) = \sigma''(x_n)$ et $\sigma'(x_0) = \sigma'(x_n)$ pour obtenir une fonction périodique $C'(\mathbb{R})$

2. **"Splines not a knot" :** Ces splines consistent à relier x_0à x_2 par un seul polynôme de degré 3 au lieu de deux polynômes, de même pour la restriction de σ à $[x_{n-2}, x_n]$. l'expression anglophone "not a knot" s'explique par le fait que ces splines considèrent qu'il y'ait pas de noeud en $x_1$et en x_{n-1} , bien qu'il y ait des conditions d'interpolation. Remarquons au passage que , par defaut, Matlab utilise cette condition pour interpoler à l'aide de fonction splines.
3. **"Splines cubiques naturelles" :** Cela consiste à imposer les deux conditions $\sigma''(x_0) = 0\, et\, \sigma''(x_n) = 0$. Ces conditions conduisent à une propriété de minimisation très intéréssante, qui est à l'origine du vocabulaire "spline" et "naturelle", et justifie l'utilisation extensive des fonctions splines pour l'interpolation et l'approximation de données. Ainsi notre problème d'interpolation sera exclusivement résolu à l'aide des splines cubiques naturelles.

1.3 Splines cubiques naturelle d'interpolation

Théorème. ***Carl de Boor, 1963***

Soient $a = x_0$ *et* $b = x_n$.

1. Parmi toutes les fonctions f deux fois dérivables, dont la dérivée seconde est dans $L^2([a,b])$, et qui interpolent les points $(x_j, y_j)_{j=0:n}$, la spline cubique naturelle est celle qui minimise $\int_a^b (f''(x))^2 dx$.
2. Parmi toutes les fonctions f deux fois dérivables, dont la dérivée seconde est dans $L^2(\mathbb{R})$, et qui interpolent les points $(x_j, y_j)_{j=0:n}$, la spline cubique naturelle est celle qui minimise $\int_{\mathbb{R}} (f''(x))^2 dx$.

Définition. spline cubique naturelle d'interpolation

On appelle *spline cubique naturelle d'interpolation* des points $(x_j, y_j)_{j=0:n}$ l'unique fonction σ de $D^{-2}L^2(\mathbb{R})$ telle que :

1. $\forall j \in \mathbb{N},\ \sigma(x_j) = y_j$
2. $\forall f \in D^{-2}L^2(\mathbb{R}), \quad (\forall j \in [0,n]\,,\ f(x_j) = y_j) \Rightarrow \int_{\mathbb{R}} (\sigma''(x))^2 dx \leq \int_{\mathbb{R}} (f''(x))^2 dx$

Propriété. La spline cupique naturelle d'interpolation des point $(x_j, y_j)_{j=0:n}$, vérifie :

- Sur $]-\infty, x_0]$et sur $[x_n, +\infty[$, σ est un polynôme de degré 1.
- Sur chaque $]x_j, x_{j+1}]$, σ est un polynôme de degré 3.
- σ, σ', *et* σ'' sont définies et continues sur $\mathbb{R}$.
- $\sigma''(x_0) = \sigma''(x_n) = 0$

Réciproquement. Si une fonction satisfait les conditions ci-dessus et $\forall j \in \mathbb{N},\ \sigma(x_j) = y_j$, alors σ est la spline cubique naturelle d'interpolation des points $(x_j, y_j)_{j=0:n}$,

On dit que σ est un polynôme de degré 3 par morceaux, dont la dérivée seconde est continue et nulle en x_0 et x_n.

Expression des splines cubiques

Il y a plusieurs façons d'écrire les splines cubiques, cependant, la plus simple et la plus rapide du point de vue des calculs, est certainement de considérer que, localement, la fonction spline est un polynôme p de degré 3. Pour un x donné, il faut donc déterminer dans quel intervalle $[x_j, x_{j+1}]$ il se trouve, puis calculer le polynôme de degré 3 auquel la spline est égale sur cet intervalle. Pour cela, on écrit naturellement le polynôme p sous la forme locale, liée à l'intervalle $[x_j, x_{j+1}]$ c'est à dire à l'aide de son developpement limité enx_j :

$$p(x) = p(x_j) + (x - x_j)p'(x_j) + \frac{(x - x_j)^2}{2}p''(x_j) + \frac{(x - x_j)^3}{6}p'''(x_j) \tag{1.3.1}$$

puisque p et σ sont identiques sur $[x_j, x_{j+1}]$ on a bien sûr $p(x_j) = \sigma(x_j)$, $p'(x_j) = \sigma'(x_j)$, $p''(x_j) = \sigma''(x_j)$, $p'''(x_j) = \sigma'''(x_j+) = lim_{x\to x_j, x\succ x_j}\sigma'''(x_j)$ (σ''' est une fonction constante par morceaux car σ est de degré 3) avec les $\sigma(x_j)$, $\sigma'(x_j)$, $\sigma''(x_j)$, $\sigma'''(x_j+)$ supposées calculables.

Dans la suite on notera, pour tout j dans $[0, n]$:

$$\sigma_j \, pour \, \sigma(x_j),\ \sigma'_j \, pour \, \sigma'(x_j),\ \sigma''_j \, pour \, \sigma''(x_j),\ \sigma'''_j \, pour \, \sigma'''(x_j+).$$

Ainsi l'écriture de σ entre x_0 et x_n peut se résumer, de façon analytique, sous la forme :

$$\forall j \in [0, n-1]\,,\ \forall x \in [x_j, x_{j+1}]\,,\ \sigma(x) = \sigma_j + (x - x_j)\sigma'_j + \frac{(x - x_j)^2}{2}\sigma''_j + \frac{(x - x_j)^3}{6}\sigma'''_j \tag{1.3.2}$$

Remarquons que si $x \in]-\infty, x_0]$ on a $\sigma(x) = \sigma_0 + (x - x_0)\sigma'_0$ et si $x \in [x_n, +\infty[$ on a $\sigma(x) = \sigma_n + (x - x_n)\sigma'_0$.

1.4 Méthode de détermination de $\sigma(x)$: système linéaire en $\sigma_j^{(2)}(x)$

On notera $h_j = x_{j+1} - x_j$

L'idée de la méthode est d'utiliser au maximum la forme localement polynomiale de la spline (formulation (1.3.1)).

Théorème. *Cas générale : Méthode de détermination de la spline cubique d'interpolation.*

La spline cubique naturelle d'interpolation des point $(x_j, y_j)_{j=0:n}$ peut être déterminée par la méthode suivante :

– Détermination des σ_j : le vecteur $(\sigma_j)_{j=0:n}$ est déterminé par la relation $\forall j \in [0, n]\,,\ \sigma_j = y_j$.

– Détermination des σ''_j : le vecteur $(\sigma''_j)_{j=0:n}$ est déterminé en résolvant le système linéaire suivant :

$$\begin{cases} \sigma''_0 = \sigma''_n = 0 \ (\textit{conditions pour spline naturelle}) \\ \forall j \in [1, n-1]\,, \ , \frac{h_{j-1}}{6}\sigma''_{j-1} + \frac{h_{j-1}+h_j}{3}\sigma''_j + \frac{h_j}{6}\sigma''_{j+1} = \frac{\sigma_{j+1}-\sigma_j}{h_j} - \frac{\sigma_j - \sigma_{j-1}}{h_{j-1}} \end{cases} \tag{1.4.1}$$

le système ci-dessus peut s'écrire sous forme matricielle :

$$\begin{pmatrix} \frac{h_0+h_1}{3} & \frac{h_1}{6} & 0 & & & \\ \frac{h_1}{6} & \frac{h_1+h_2}{3} & \frac{h_2}{6} & & 0 & \\ 0 & \frac{h_2}{6} & \frac{h_2+h_3}{3} & \frac{h_3}{6} & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \frac{h_{n-3}}{6} & \frac{h_{n-3}+h_{n-2}}{3} & \frac{h_{n-2}}{6} \\ & & & & \frac{h_{n-2}}{6} & \frac{h_{n-2}+h_{n-1}}{3} \end{pmatrix} \begin{pmatrix} \sigma''_1 \\ \sigma''_2 \\ . \\ . \\ . \\ \sigma''_{n-1} \end{pmatrix} = \begin{pmatrix} \frac{\sigma_2-\sigma_1}{h_1} - \frac{\sigma_1-\sigma_0}{h_0} \\ \frac{\sigma_3-\sigma_2}{h_2} - \frac{\sigma_2-\sigma_1}{h_1} \\ \\ \\ \\ \frac{\sigma_n-\sigma_{n-1}}{h_{n-1}} - \frac{\sigma_{n-1}-\sigma_{n-2}}{h_{n-2}} \end{pmatrix}$$

c'est un système linéaire tridiagonale.

– Détermination des σ'_j et σ'''_j : les vecteurs $(\sigma'_j)_{j=0:n}$ et $(\sigma'''_j)_{j=0:n}$ sont alors déterminés par les relations :

$$\forall j \in [0, n-1]\,, \ \ \sigma'''_j = \frac{(\sigma''_{j+1} - \sigma''_j)}{h_j} \tag{1.4.2}$$

$$\forall j \in [0, n-1]\,, \qquad \sigma'_j = \frac{\sigma_{j+1} - \sigma_j}{h_j} - \frac{h_j}{6}(\sigma''_{j+1} + 2\sigma''_j) \tag{1.4.3}$$

enfin $\sigma'''_n = 0$ et $\sigma'_n = \sigma'_{n-1} + h_{n-1}\sigma''_{n-1} + \frac{h^2_{n-1}}{3}\sigma'''_{n-1}$.

Chapitre 2

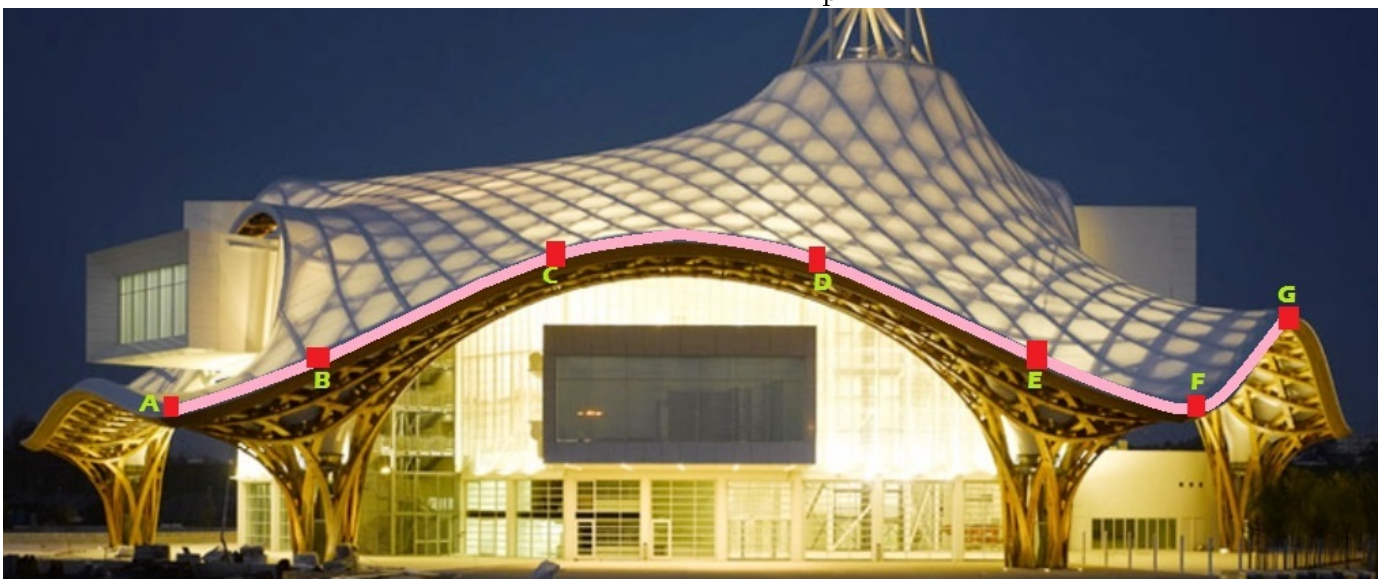
Problème

2.1 Enoncé

Dans le domaine du design, les splines sont utilisées pour approcher des contours complexes. Leur simplicité d'implémentation les rend très populaires et elles sont fréquemment utilisées dans les logiciels de dessin.

On cherche à simuler sous Matlab le design du bord du toit du Centre Pompidou de Metz (tel un logiciel de dessin spécialisé) comme sur la figure ci dessous.

FIGURE 2.1.1 – Le Centre Pompidou-Metz.



Algorithme 2.1 Algorithme de résolution du problème

1. Choix stratégique points pour l'interpolation afin d'avoir l'allure voulu de la courbe d'interpolation
2. Récupération des coordonnés des points d'interpolation d'abcisses $(x_j)_{j=0:n}$ et d'ordonnées $(y_j)_{j=0:n}$
3. Construire le vecteur des points qui seront interpolés : On choisi les abcisses $(x'_i)_{i=0:k} / x'_i < x'_{i+1}$ des points à interpoler avec $x'_0 \geq x_0$ et $x'_k \leq x_n$
4. Construire la matrice des intervalles entre deux noeuds consécutifs : $h_j = x_{j+1} - x_j$
5. Construire le vecteur *sigma* qui contient les différents noeuds, c'est à dire les abcisses de jonction entre deux polynômes cubiques. $(\sigma_j)_{j=0:n} = (x_j)_{j=0:n}$;
6. Construction du second membre du système triadiagonale :
$$\begin{pmatrix} \frac{\sigma_2-\sigma_1}{h_1} - \frac{\sigma_1-\sigma_0}{h_0} \\ \frac{\sigma_3-\sigma_2}{h_2} - \frac{\sigma_2-\sigma_1}{h_1} \\ \\ \\ \\ \frac{\sigma_n-\sigma_{n-1}}{h_{n-1}} - \frac{\sigma_{n-1}-\sigma_{n-2}}{h_{n-2}} \end{pmatrix}$$
7. Construction de la matrice tridiagonale du système :
$$\begin{pmatrix} \frac{h_0+h_1}{3} & \frac{h_1}{6} & 0 & & & \\ \frac{h_1}{6} & \frac{h_1+h_2}{3} & \frac{h_2}{6} & & 0 & \\ 0 & \frac{h_2}{6} & \frac{h_2+h_3}{3} & \frac{h_3}{6} & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \frac{h_{n-3}}{6} & \frac{h_{n-3}+h_{n-2}}{3} & \frac{h_{n-2}}{6} \\ & & & & \frac{h_{n-2}}{6} & \frac{h_{n-2}+h_{n-1}}{3} \end{pmatrix}$$
8. On résoud le système tridiagonale, pour orbtenir le vecteur *sigmasecond*, on a :
   (a) $$\begin{pmatrix} \sigma''_1 \\ \sigma''_2 \\ . \\ . \\ . \\ \sigma''_{n-1} \end{pmatrix} = \begin{pmatrix} \frac{\sigma_2-\sigma_1}{h_1} - \frac{\sigma_1-\sigma_0}{h_0} \\ \frac{\sigma_3-\sigma_2}{h_2} - \frac{\sigma_2-\sigma_1}{h_1} \\ \\ \\ \\ \frac{\sigma_n-\sigma_{n-1}}{h_{n-1}} - \frac{\sigma_{n-1}-\sigma_{n-2}}{h_{n-2}} \end{pmatrix} \times \begin{pmatrix} \frac{h_0+h_1}{3} & \frac{h_1}{6} & 0 & & & \\ \frac{h_1}{6} & \frac{h_1+h_2}{3} & \frac{h_2}{6} & & 0 & \\ 0 & \frac{h_2}{6} & \frac{h_2+h_3}{3} & \frac{h_3}{6} & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \frac{h_{n-3}}{6} & \frac{h_{n-3}+h_{n-2}}{3} & \frac{h_{n-2}}{6} \\ & & & & \frac{h_{n-2}}{6} & \frac{h_{n-2}+h_{n-1}}{3} \end{pmatrix}^{-1}$$
   (b) on ajoute les conditions "naturelles" aux bords :$\sigma''_0 = \sigma''_n = 0$
9. On construit le vecteur *sigmater*, on a :
   (a) $\forall j \in [0, n-1]\,,\ \ \sigma'''_j = \frac{(\sigma''_{j+1}-\sigma''_j)}{h_j}$ On
   (b) ajoute la condition au bord : $\sigma'''_n = 0$
10. On construit le vecteur sigmaprime, on a :
    (a) $\forall j \in [0, n-1]\,, \qquad \sigma'_j = \frac{\sigma_{j+1}-\sigma_j}{h_j} - \frac{h_j}{6}(\sigma''_{j+1} + 2\sigma''_j)$
    (b) Avec la condition au bord $\sigma'_n = \sigma'_{n-1} + h_{n-1}\sigma''_{n-1} + \frac{h^2_{n-1}}{3}\sigma'''_{n-1}$.
11. Pour trouver les ordonnés $(y'_i)_{i=0:k}$ chaque point interpollé avec les $(x'_i)_{i=0:k}$ on calcul :
    (a) $\forall j \in [0, n-1]\,, \forall i \in [0, k]\,,\ \forall x'_i \in [x_j, x_{j+1}]\,,\ y'_i = \sigma_j + (x'_i - x_j)\sigma'_j + \frac{(x'_i-x_j)^2}{2}\sigma''_j + \frac{(x'_i-x_j)^3}{6}\sigma'''_j$ **(l'expression locale entre deux noeuds du poplynome cubique.)**
12. On trace la coube d'interpolation avec les points $(x'_i, y'_i)_{i=0:k}$ et on place les points d'interpolations $(x_j, y_j)_{j=0:n}$.

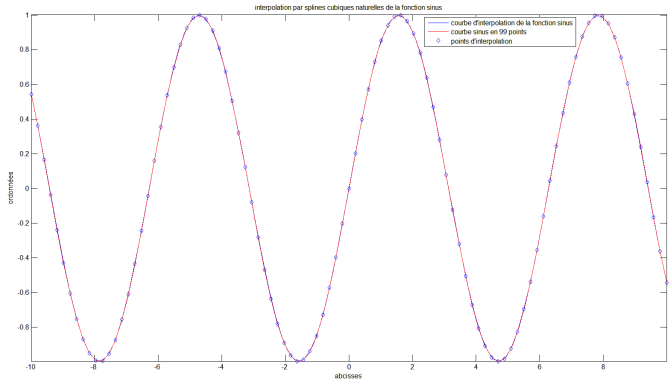
2.2 Algorithmes de résolution

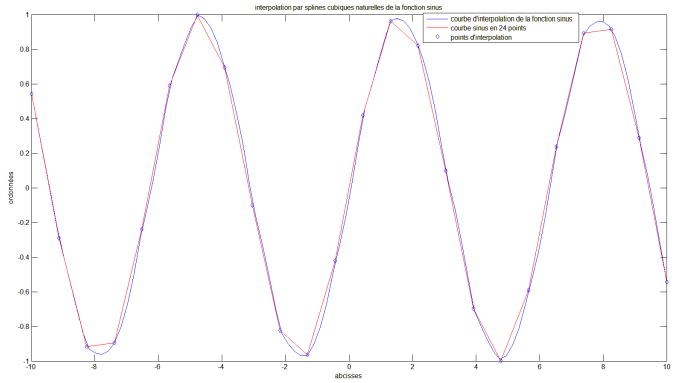
2.3 Figures

2.3.1 Tests

D'abord on teste notre programme en interpolant la fonction *sinus* sur l'interval $[-10, 10]$ à l'aide de deux scripts Matlab *interpolationsinus* et *interpolationsinus*100 .On obtient les figures suivantes.

FIGURE 2.3.1 – Interpolation de la fonction sinus par splines cubiques naturelles

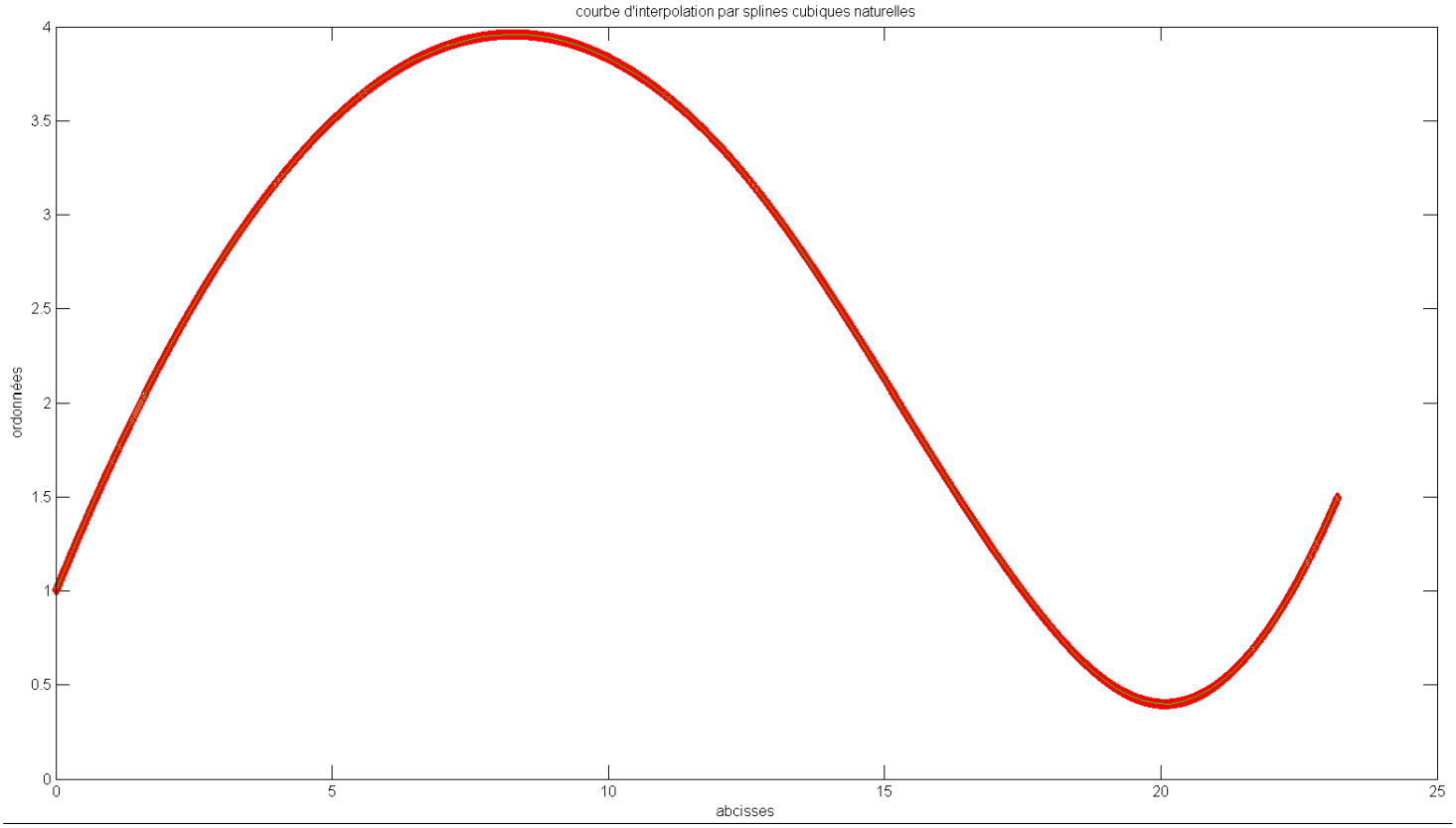




2.3.2 Figures de la résolution

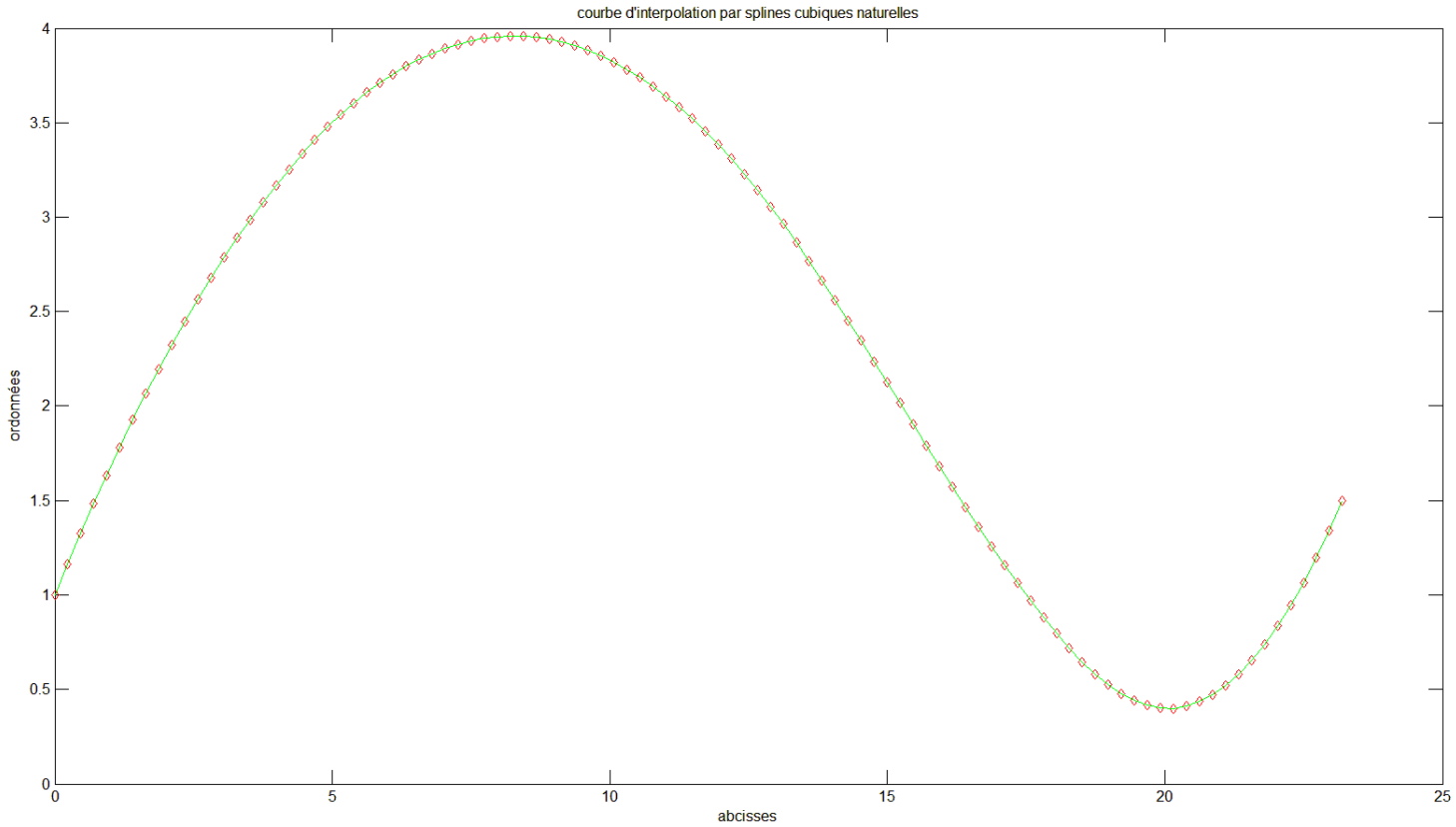
Algorithme 2.2 Resolution à l'aide du script reso

10.000 points interpolés, à exécuter avec précautions car le temps de calcul peut être important pour un processeur peur performant. Ceci a été exécuté sur un processeur INTEL INSIDE i7.



Algorithme 2.3 Resolution à l'aide du script reso2

100 points interpolés



Chapitre 3

Annexe

3.1 Bibliographie

1. “Modélisation” de Christophe Rabut, INSA Toulouse, Département Sciences et Techniques Pour l'Ingénieur, PO MIC 3ème année, 2009-2010.

3.2 Webographie

www.wikipedia.fr

3.3 Programmes

Algorithme 3.1 Fonction Constructinterpolation

```
function [ interpolation ] = Constructinterpolation( X,NBP )
%UNTITLED6 Summary of this function goes here
% Detailed explanation goes here
interpolation1=linspace(X(1),X(length(X)),NBP) ;
%On insere les points d'interpolation dans le vecteur des points à
%interpoler
for i=1 :(length(X))
for j=1 :(length(interpolation1))
if X(i)<=interpolation1(j)
interpolation(j)=X(i) ;
else
if X(i)>interpolation1(j)
interpolation(j)=interpolation1(j) ;
end
end
end
end
end
```

Algorithme 3.2 Fonction construcinterval

```
function [interval] = construcinterval( x )
%%/*Construction du vecteur h qui represente les longueurs entres les noeuds
%/*D'apres la méthode h_{j}=x_{j+1}-x_{j}
% Detailed explanation goes here
for i=1 :(length(x)-1)
interval(i)=x(i+1)-x(i) ;
end
end
```

Algorithme 3.3 fonction constructtridiagonale

```
function [ tridiagonale ] = constructtridiagonale( interval )
%*construction des diagonales 1 et 2
% /*construction des diagonales 1 et 2
%/*d12 le vecteur des diagonales 1 et 2
for j=1 :(length(interval)-2)
d12(j)=interval(j)/6 ;
end
%/*construction de la diagonale principale de vecteur d
for j=1 :(length(interval)-1)
d(j)=(interval(j)+interval(j+1))/3 ;
end
%/*construction de la matrice tridiagonale A
%/*diagonale secondaire
for j=1 :(length(interval)-2)
tridiagonale(j,j+1)=d12(j) ;
tridiagonale(j+1,j)=d12(j) ;
end
%/*diagonale principale
for j=1 :(length(interval)-1)
tridiagonale(j,j)=d(j) ;
end
end
```

Algorithme 3.4 fonction constructsigmasecond

```
function [ sigmasecond ] = constructsigmasecond( secondmembre,tridiagonale )
%/*resolution du système tridiagonale, calcul de sigmasecond
% /*resolution du système tridiagonale, calcul de sigmasecond
sigmasecond=secondmembre/tridiagonale ;
sigmasecond=[0 sigmasecond 0] ;
end
```

Algorithme 3.5 Fonction constructsigmater

```
function [ sigmater ] = constructsigmater( sigmasecond,interval )
%%Construction du vecteur sigma-ter
% %Construction du vecteur sigma-ter
for j=1 :(length(interval))
sigmater(j)=(sigmasecond(j+1)-sigmasecond(j))/interval(j) ;
sigmater=[sigmater 0] ;
end
end
```

Algorithme 3.6 Fonction constructsigmaprime

```
function [ sigmaprime ] = constructsigmaprime( sigma,sigmasecond,sigmater,interval )
%
% %Construction du vecteur sigma-prime
for j=1 :(length(interval))
sigmaprime(j)=((sigma(j+1)-sigma(j))/interval(j))-((interval(j)/6)*(sigmasecond(j+1)+2*sigmasecond(j)));
end
ns=length(sigmaprime);
nh=length(interval);
nd=length(sigmasecond)-1;
nt=length(sigmater)-1;
sigmaprimen=sigmaprime(ns)+interval(nh)*sigmasecond(nd)+(((interval(nh)^2)/2)*sigmater(nt));
sigmaprime=[sigmaprime sigmaprimen];
end
```

Algorithme 3.7 Fonction polyx

```
function [ sigmax ] = polyx( X,interpolation,sigma,sigmaprime,sigmasecond,sigmater )
%UNTITLED3 Summary of this function goes here
% Detailed explanation goes here
for j=1 :length(interpolation)
for i=1 :length(X)-1
if interpolation(j)>=X(i) && interpolation(j)<=X(i+1)
sigmax(j,1)=1*sigma(i);
sigmax(j,2)=(interpolation(j)-X(i))*sigmaprime(i);
sigmax(j,3)=((interpolation(j)-X(i))^2/2)*sigmasecond(i);
sigmax(j,4)=((interpolation(j)-X(i))^3/6)*sigmater(i);
end
end
end
```

Algorithme 3.8 Fonction pspline

```
function [interpolation,interpole] = pspline(X,Y,NBP)
%Fonction PSPLINE/interpolation :splines/projet16 Matlab/MASTER MAIME2013-2014/BCKY
% La fonction PSPLINE, fait de l'interpolation polynomiale par morceaux, par la
% méthode des splines cubiques naturelles
% La syntaxe générale est PSPLINE(X,Y,NBP)
% X : est un vecteur/abcisses des point par lesquels passe la courbes d'interpolation
% Y : est unvecteur de meme taille que Y/ordonees associées à X
% NBP : est un scalaire ou un vecteur/Le vecteur contenant les abcisses des points à interpoler ou le Nombre
de point entre le minimum de X et le maximum de X qui seront interpolés
% La fonction renvoie les points interpolés, à savoir un vecteur qui
% contient leurs abcisses et un autre qui contient leurs ordonnees et trace
% la courbe d'interpolation
%Construction de interpolation les abcices des points qui seront
%interpolées par les splines cubiques
if size(NBP)==[1 1]
interpolation = Constructinterpolation( X,NBP );
else
%On insere les points d'interpolation dans le vecteur des points à
%interpoler
interpolation1=NBP;
for i=1 :(length(X))
for j=1 :(length(interpolation1))
if X(i)<=interpolation1(j)
interpolation(j)=X(i);
else
if X(i)>interpolation1(j)
interpolation(j)=interpolation1(j);
end
```

Algorithme 3.9 Fonction pspline (Suite)

```
end
end
end
end
%Construction du vecteur sigma qui contient les coordonnees des noeuds, où
%c'est à dire les abcisses de raccordement des splines cubiques
sigma=constructsigma(Y) ;
%construction du vecteur contenant les distances entre deux noeuds consécutifs x(j+1)-x(j)
interval=construcinterval(X) ;
%Construction du second membre du systeme tridiagonale a resoudre
secondmembre=construcsecondmembre(sigma,interval) ;
%Construction de la matrice tridiagonale du systeme tridiagonale a resoudre
tridiagonale = constructtridiagonale( interval ) ;
%Construction du vecteur sigma-second par resolution du systeme tridiagonale
sigmasecond = constructsigmasecond(secondmembre,tridiagonale) ;
%Construction du vecteur sigma-ter
sigmater = constructsigmater( sigmasecond,interval ) ;
%Construction du vecteur sigma-prime
sigmaprime = constructsigmaprime( sigma,sigmasecond,sigmater,interval ) ;
%Recherche de l'intervalle et calcul de la matrice
%[s,(x-xj)s',s"(x-xj)^2/2),s"'((x-xj)^3/6)]
sigmax = polyx( X,interpolation,sigma,sigmaprime,sigmasecond,sigmater) ;
%Construction de interpole/ on somme ligne par ligne pour obtenir
%s+(x-xj)s'+s"(x-xj)^2/2)+s"'((x-xj)^3/6)
for j=1 :length(sigmax( :,1))
interpole(j)= sum(sigmax(j, :)) ;
end
%**********TRACE, GRAPHIQUE*************
plot(X,Y,'rd',interpolation,interpole,'g') ;
xlabel('abcisses') ;
ylabel('ordonnées') ;
title('courbe d"interpolation par splines cubiques naturelles') ;
end
```

Algorithme 3.12 script reso

```
%On utilise la fonction spline défini dans matlab afin d'avoir une certaine
%comparaison avec notre fonction spline et nous donner les points
%d'interpolation
%le vecteur x et y donne les points d'interpolation que nous avons choisi
%afin d'avoir l'allule de la courbe désiré
%ce script est à exécuter avec précaution, car les points interpolés
%sont au nombre de 10000 et peut prendre du temps à s'exécuter
%Nous avons pu l'exécuter sur notre machine ayant un processeur 7 coeurs
%(intel inside i7)
l=linspace(0,23.2,10000);
x= [0 5.0000 12.5000 17.5000 20.0000 21.0000 23.2000];
y= [1.0000 3.5000 3.2000 1.0000 0.4000 0.5000 1.5000];
s=spline(x,y,l);
[xx,yy]= pspline(l,s,10000);
```

Algorithme 3.10 Script interpolationsinus

```
x=linspace(-10,10,24);
y=sin(x);
[xx,yy]=pspline(x,y,100);
plot(xx,yy,'b',x,y,'r',x,y,'d');
%plot();
xlabel('abcisses');
ylabel('ordonnées');
title('interpolation par splines cubiques naturelles de la fonction sinus');
legend('courbe d"interpolation de la fonction sinus','courbe sinus en 24 points','points d"interpolation');
```

Algorithme 3.11 Script interpolationsinus100

```
x=linspace(-10,10,99);
y=sin(x);
[xx,yy]=pspline(x,y,100);
plot(xx,yy,'b',x,y,'r',x,y,'d');
%plot();
xlabel('abcisses');
ylabel('ordonnées');
title('interpolation par splines cubiques naturelles de la fonction sinus');
legend('courbe d"interpolation de la fonction sinus','courbe sinus en 99 points','points d"interpolation');
```

Algorithme 3.13 script reso2

```
%On utilise la fonction spline défini dans matlab afin d'avoir une certaine
%comparaison avec notre fonction spline et nous donner les points
%d'interpolation
%le vecteur x et y donne les points d'interpolation que nous avons choisi
%afin d'avoir l'allule de la courbe désirée
l=linspace(0,23.2,100) ;
x= [0 5.0000 12.5000 17.5000 20.0000 21.0000 23.2000] ;
y= [1.0000 3.5000 3.2000 1.0000 0.4000 0.5000 1.5000] ;
s=spline(x,y,l) ;
[xx,yy]= pspline(l,s,100) ;
```

FIGURE 3.3.1 – Indication sur les programmes

Fonction	Paramètres	Indication
Constructinterpolation(X,NBP)	**X** : vecteur, nœuds des polynômes **NBP** : scalaire ou vecteur, nombre de points interpolés ou vecteur des abscisses des points à interpoler, *dont les extrémités sont égales à celles de X*	*Renvoi tous les points qui seront interpolés par splines cubiques naturelles. Les éléments des vecteurs en paramètre doivent être croissants*
construcinterval(X)	**X** : vecteur de taille n	*Renvoi un vecteur (n-1) qui contient les longueurs des intervalles entre deux éléments successifs de X.*
constructtridiagonale(interval)	**Interval** : vecteur	*Renvoi une matrice tridiagonale*
constructsigmasecond(secondmembre,tridiagonale)	**secondmembre** : vecteur de taille n **Tridiagonale** : matrice tridiagonale n*n	*Renvoi le vecteur secondmembre*inverse(tridiagonale).*
constructsigmater(sigmasecond,interval)	**Sigmasecond** : vecteur de taille n **Interval** : vecteur de taille n-1	*Renvoi un vecteur construit à partir de la formule de l'algorithme*
constructsigmaprime(sigma,sigmasecond,sigmater,interval)	Tous des vecteurs de la même taille	*Renvoi un vecteur calculé avec la formule de l'algorithme*
polyx(X,interpolation,sigma,sigmaprime,sigmasecond,sigmater);	Tous des vecteur de même taille	*Renvoi une matrice dont les éléments sont calculés suivant la forme locale du polynôme d'interpolation.*
pspline(X,Y,NBP)	**X** : vecteur de taille n **Y** : vecteur de taille n **NBP** : scalaire ou vecteur dont les extrémités sont égales à celles de X.	*Renvoi deux vecteurs contenant abscisses et ordonnées des points interpolés puis trace la courbe d'interpolation, elle utilise l'ensemble des fonctions précédemment décrites.*