Projet d'analyse numérique

Résolution d'equations différentielles à l'aide des splines

Céline CATTOEN
Bénédicte LIENHARDT

4 avril 2002

Table des matières

Remerciements

Nous tenons à remercier vivement M.RABUT, notre responsable de projet, pour sa trés grande disponibilité. Nous n'oublions pas non plus les responsables en informatique qui ont toujours su nous consacrer du temps et nous aider dans la recherche de nos erreurs.

Introduction

Dans les études scientifiques ou dans les sciences de l'ingénieur, la résolution d'équations ou plus généralement de systèmes d'équations différentielles ou aux dérivées partielles est trés importante. Cependant, certaines équations ne peuvent pas être résolues explicitement à l'aide d'expressions comportant les fonctions usuelles de l'analyse. Il est donc nécessaire de recourir à des méthodes numériques qui exploitées sur l'ordinateur vont donner une résolution de ces équations. Néanmoins, cette résolution ne sera qu'approchée dans la plupart des cas.

Notre projet consiste à décrire de manière approchée une solution particulière d'une équation differentielle linéaire par une foction spline de noeuds equirépartis.

Le rapport présente de manière détaillée la façon dont nous avons résolu ce problème et les choix que nous avons faits. En annexe sont fournis les courbes permettant de visualiser les résultats obtenus ainsi que le programme.

Chapitre 1

Présentation du projet

1.1 Présentation du Sujet

Notre projet consiste donc à décrire de manière apprchée une solution particulière d'une équation differentielle $E(f) = 0$ par une spline cubique de noeuds equirépartis $(x_i)_{i=0,..,n}$.
Pour cela, on va chercher la solution sous forme d'une combinaison linéaire de fonctions simples, les Bsplines, les inconnues étant les a_i dans

$$\sigma(x) \ = \sum_{i=-1}^{n+1} a_i B(\frac{x-\alpha}{h} - i)$$

On peut évaluer l'éxactitude de la solution approchée grâce à des comparaisons graphiques et au calcul de $D(h) = \frac{1}{N} * \sum_{j=1}^{N} (\sigma(x_j) - f(x_j))^2$ qui permet de comparer l'écart entre la solution approchée et la solution exacte pour un nombre de noeuds qui varie.

1.2 Equations proposées

Les équations à résoudre numériquement sur $[\alpha, \beta]$ sont les suivantes:

- $E(f)(x) \ = \ f''(x) + f'(x)$ avec comme conditions limites $f(0) = 0$ et $f(1) = 1$
- $E(f)(x) \ = \ f''(x) - 4f'(x) + 3f(x) - 10 sinx$ avec comme conditions limites $f(0) = 0$ et $f(1) = -1$
- $E(f)(x) \ = \ f''(x) - 4f'(x) + 4f(x) - 5coshx + 4sinhx$ avec comme conditions limites $f(0) = 0$ et $f(1) = cosh 1$

Afin de comparer la solution approchée avec la solution exacte, il est nécessaire de résoudre chacune de ces trois équations au préalable.

1.2.1 Première équation

$$E(f)(x) = f''(x) + f'(x)$$
$$f(0) = 0$$
$$f(1) = 1$$

La solution générale de l'équation homogène $f''(x) + f'(x) \ = \ 0$ est:

$$f(x) \ = \ Acosx + Bsinx$$

Les conditions initiales donnent:

$$\begin{cases} A \ = \ 0 \\ Acos1 + Bsin1 \ = \ 1 \end{cases}$$

On trouve donc la solution particulière de la première équation:

$$f(x) \ = \ \frac{sin1}{sinx}$$

1.2.2 Deuxième équation

$$E(f)(x) = f''(x) - 4f'(x) + 3f(x) - 10sinx$$
$$f(0) = 0$$
$$f(1) = -1$$

La solution générale de l'équation homogène $f''(x) - 4f'(x) + 3f(x) \ = \ 0$ est:

$$f(x) \ = \ Ae^x + Be^{3x}$$

La solution particulière de l'équation complète $f''(x)-4f'(x)+3f(x) \ = \ 10sinx$ est:

$$f_0(x) \ = \ 2cosx + sinx$$

Les conditons initiales donnent:

$$\begin{cases} A + B + 2 \ = \ 0 \\ Ae + Be^3 + 2cos1 + sin1 \ = \ -1 \end{cases}$$

On trouve donc la solution particulière de la deuxième équation:

$$\begin{cases} f(x) \ = \ Ae^x + Be^{3x} + 2cosx + sinx \\ B = -\frac{1+2cos1+sin1-2e}{e^3-e} \\ A = 2 - B \end{cases}$$

1.2.3 Troisième équation

$$E(f)(x) = f''(x) - 4f'(x) + 4f(x) - 5chx + 4shx$$
$$f(0) = 0$$
$$f(1) = ch1$$

La solution générale de l'équation homogène $f''(x) - 4f'(x) + 4f(x) = 0$ est:

$$f(x) = (A + Bx)e^{2x}$$

La solution particulière de l'équation complète $f''(x)-4f'(x)+4f(x) = 5chx-4shx$ est:

$$f_0(x) = chx$$

Les conditions initiales donnent:

$$\begin{cases} A + 1 = 0 \\ (A + B)e^2 + ch1 = ch1 \end{cases}$$

On trouve donc la solution particulière de la troisième équation:

$$f(x) = (-1 + x)e^{2x} + chx$$

Chapitre 2

Détermination de la solution approchée

2.1 Mise en place du système à résoudre

On définit N points équirépartis $(X_j)_{i=1,..,N}$ sur notre intervalle de résolution $[\alpha, \beta]$, N étant supérieur voir largement supérieur à n et on cherche les $(a_i)_{i=-1,..,n}$ qui minimisent le critère:

$$e(a) \ = \sum_{j=1}^{N} E(f)^2(X_j)$$

sous la contrainte de satisfaire les conditions aux limites.

Pour minimiser $e(a)$, le raisonnement est le même que pour le pôlynome des moindres carrés discrets.
Ainsi,

$$\frac{\partial e(a)}{\partial a_j} = 0 \ \forall j \ \epsilon[1, n-1]$$

est une condition necéssaire, e etant une fonction des n-1 variables $(a_i)_{i=1,..,n-1}$ et c'est une condition suffisante. e est quadratique en chaque (a_j) et les coefficients des $(a_j)^2$ sont tous positifs, il s'agit donc bien d'un minimum.

D'àprès les propriétés des approximations B-spline, on connait a_0 et a_n grâce aux conditions aux limites, en effet $\sigma(x)$ passe par les extrémités. De plus, on a les prolongements suivants: $a_{-1} = 2 * a_0 - a_1$ et $a_{n+1} = 2 * a_n - a_{n-1}$. Par conséquent, le problème se ramène à déterminer les $(a_i)_{i=1,..,n-1}$.

On écrit donc $e(a)$ en remplaçant a_{-1} et a_{n+1} par leurs expressions respectives et on regroupe les termes en fonction des a_i.

$$e(a) \ = \sum_{j=1}^{N} (\sum_{i=0}^{n} (a_i * coeff_{a_i}) - SM(X_j))$$

avec, en notant SM la fonction représentant le second membre et L la fonction représentant la partie linéaire de l'équation différentielle:

$$coeff_{a_i} = L(B_i)(x_j)\ \forall i\ \epsilon [2, n-2]$$
$$coeff_{a_0} = (2 * L(B_{-1}) + L(B_0))(x_j)$$
$$coeff_{a_1} = (-L(B_{-1}) + L(B_1))(x_j)$$
$$coeff_{a_{n-1}} = (-L(B_{n+1}) + L(B_{n-1}))(x_j)$$
$$coeff_{a_n} = (2 * L(B_{n+1}) + L(B_n))(x_j)$$

On dérive alors $e(a)$ par rapport à $(a_{i_0})_{i_0=1,..,n-1}$. E étant linéaire dans le cadre de ce projet, on obtient alors un système linéaire en a_i.

On se retrouve avec des équations de la forme:

$$2 * \sum_{j=1}^{N} coeff_{a_{i_0}} * (\sum_{i=-1}^{n+1} a_i * coeff_{a_i} - SM(x_j)) = 0$$

or $coeff_{a_{i_0}}$ est une combinaison lineaire de B_spline et on le multiplie avec $\sum_{i=-1}^{n+1} a_i * coeff_{a_i}$ contenant elle même des B_splines ce qui permet de simplifier.

En effet, si on dérive par rapport à (a_{i_0}), on a pour $4 \leq |i_0 - k|\ B_k(x) * B_{i_0}(x) = 0$, une des deux splines étant forcément nulle. On a donc des expressions du type:

$$2 * \sum_{j=1}^{N} \sum_{i=i_0-3}^{i_0+3} (coeff_{a_{i_0} * a_i * coeff_{a_i}}) - \sum_{j=1}^{N} coeff_{a_{i_0}} * SM(x_j) = 0$$

$$\Longleftrightarrow \sum_{j=1}^{N} \sum_{i=i_0-3}^{i_0+3} (coeff_{a_{i_0}} * a_i * coeff_{a_i}) = \sum_{j=1}^{N} coeff_{a_{i_0}} * SM(x_j)$$

On a ainsi un système linéaire d'ordre n-1.

2.2 Résolution du système

Les systèmes linéaires multidiagonaux sont résolus par la méthode du pivot de Gauss qui se déroule en deux phases:

- la triangulation du système linéaire
- la résolution du système triangulaire

2.2.1 Triangulation du système linéaire

Pour se ramener à un système triangulaire, on va mettre des zéros sous les ièmes termes diagonaux dans la matrice du premier membre, mais on va aussi modifier les termes de la matrice du second membre en procédant de la sorte:

$$M = \begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & & & \cdots & m_{1,p} \\ 0 & \ddots & m_{\dots} & m_{\dots} & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & 0 & m_{i,i} & \ddots & & \vdots \\ \vdots & \text{\huge 0} & & m_{i+1,i} & \ddots & \ddots & \\ \vdots & & & \vdots & \ddots & & \vdots \\ 0 & \cdots & 0 & m_{r,i} & \cdots & & m_{r,p} \end{pmatrix} \text{ et } N = \begin{pmatrix} n_1 \\ n_2 \\ \vdots \\ n_i \\ n_{i+1} \\ \vdots \\ n_r \end{pmatrix}$$

au ième terme diagonal on effectue les calculs sur le terme diagonal $i+1$ suivant:

- $m_{i+1,i+1} = m_{i+1,i+1} - \frac{m_{i+1,i} * m_{i,i+1}}{m_{i,i}}$
- $n_{i+1} = n_{i+1} - \frac{m_{i+1,i} * n_i}{m_{i,i}}$

$m_{i,i}$ est appelé le pivot, et il ne doit pas être nul.

2.2.2 Résolution du système triangulaire

En partant du dernier terme diagonal de la matrice du premier membre, on peut déterminer le terme du vecteur cherché, puis de proche en proche calculer tous les termes en remontant.

$$MA = N$$
$$\Longleftrightarrow$$
$$\begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & & & & \cdots & m_{1,p} \\ 0 & \ddots & m_{\dots} & m_{\dots} & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & & 0 & m_{i,i} & \ddots & & & \vdots \\ \vdots & \text{\huge 0} & & 0 & \ddots & \ddots & & \\ \vdots & & & \vdots & \ddots & & & \vdots \\ 0 & \cdots & 0 & & \cdots & & 0 & m_{r,p} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{i+1} \\ \vdots \\ \vdots \\ a_r \end{pmatrix} = \begin{pmatrix} n_1 \\ n_2 \\ \vdots \\ n_i \\ \vdots \\ \vdots \\ n_r \end{pmatrix}$$

Et donc au ième terme du vecteur solution on trouve, connaissant le terme $(i+1)$:

$a_i = n_i - \frac{m_{i,i+1} * a_{i+1}}{m_{i,i}}$

Chapitre 3

Algorithme de programmation

Pour un maximum de précision, nous avons utilisé la double précision afin d'éviter les problèmes d'arrondis.

3.1 Gestion des données

La saisie des données occupe une place relativement importante dans la programmation, en effet, six modules sont concernés pour gérer l'utilisation des données: m_var_globale.f90, m_choix_donnees.f90, m_equa1.f90, m_equa2.f90, m_equa3.f90, m_equa.f90.

Le module m_var_globale.f90 contient toutes les variables globales utilisées dans la majorité des modules.

Les possibilités de conditions initiales pour exécuter le programme sont multiples, l'utilisateur peut ainsi choisir entre trois équations différentielles différentes à résoudre et a aussi le choix concernant le nombre de noeuds de la spline, le nombre de points de la spline, les bornes de l'intervalle de résolution $[\alpha, \beta]$, l'intervalle de variation du pas pour le calcul de l'erreur $D(h)$, etc ... Le module m_choix_donnees.f90 gère donc la saisie de multiples données nécessaires à l'exécution du programme et propose aussi trois choix standards qui simplifient son utilisation.

Les modules m_equa1.f90, m_equa2.f90, m_equa3.f90 stockent les trois équations différentielles sous forme de premier membre, deuxième membre, conditons aux limites et contiennent également la fonction exacte de l'équation différentielle (ce qui permet ensuite d'effectuer une comparaison graphique avec la solution approchée). Le module m_equa.f90 facilite l'accès aux trois équations proposées dans l'ensemble du programme. Suivant le choix de l'utilisateur les fonctions générales appelées dans ce module se redirigent vers les fontions du module de l'équation choisie.

3.2 Fonctions particulières utilisées

Dans le module m_B_spline, la fonction $B(x)$ définit l'expression sur l'ensemble des reéls de la B spline cubique naturelle d'interpolation des points $(-2,0)$ $(-1,1/6)$ $(0,2/3)$ $(1,1/6)$ $(2,0)$.
Cette fonction est par la suite utilisée pour le remplissage des matrices du système linéaire, pour le calcul des solutions approchées ainsi que pour effectuer une comparaison graphique entre la solution exacte et la solution approchée σ.

Dans le module m_B_derivée, la fonction deriv permet de déterminer la dérivée piéme de B((x-alpha)/h-k).
Cette fonction est par la suite utilisée pour le remplissage des matrices du système linéaire.

3.3 Travail sur les matrices du système

3.3.1 Mise en place du système

Dans le module m_système, le sous-programme remplissage permet de remplir les matrices du premier membre et du second membre du systeme linéaire découlant de la minimisation de $e(a)$.
On a présenté dans le chapitre 2, partie 1 les matrices permettant de discrétiser le système à résoudre.
Notre travail consistait alors à traduire de manière informatique nos équations pour pouvoir les résoudre.

Sur le plan informatique, il n'est pas judicieux de stocker la matrice M du premier membre de dimension $(n-1, n-1)$, car elle n'a que sept diagonales non nulles.

$$M = \begin{pmatrix} m & m & m & m & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ m & m & m & m & m & 0 & \cdots & \cdots & \cdots & 0 \\ m & m & m & m & m & m & 0 & \cdots & \cdots & 0 \\ m & m & m & m & m & m & m & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \text{\huge 0} & \vdots \\ \vdots & \text{\huge 0} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & m & m & m & m & m & m & m \\ 0 & \cdots & \cdots & 0 & m & m & m & m & m & m \\ 0 & \cdots & \cdots & \cdots & 0 & m & m & m & m & m \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & m & m & m & m \end{pmatrix}$$

On utilise donc la matrice $M1$ telle que $M1(i,j) = M(i, i+j)$ pour $i = 1, ..., n-1$ et $j = -3, ..., 3$.

$$M1 = \begin{pmatrix} 0 & 0 & 0 & m & m & m & m \\ 0 & 0 & m & m & m & m & m \\ 0 & m & m & m & m & m & m \\ m & m & m & m & m & m & m \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m & m & m & m & m & m & m \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m & m & m & m & m & m & m \\ m & m & m & m & m & m & 0 \\ m & m & m & m & m & 0 & 0 \\ m & m & m & m & 0 & 0 & 0 \end{pmatrix}$$

Pour le second membre, on utilise le vecteur M2 à n-1 lignes, tel que $M1 * A = M2$, A étant un vecteur colonne contenant les $(a_i)_{i=1,..,n-1}$.

Grâce au résultat de $\frac{\partial e(a)}{\partial a_{i_0} = 0}$, on peut remplir les matrices M1 et M2 à la ligne i_0. De plus, comme on travaille avec des sommes finies sur i_0 et j, on peut les permuter.

De façon générale, on a pour la matrice du premier membre :

$$M1(i_0, l) = \sum_{j=1}^{N} (coeff_{a_{i_0}} * coeff_{a_{i_0+l}}) \ \forall i_0 \ \epsilon [1, n-1]$$

Pour la matrice du second membre, il ne faut pas oublier que l'on doit passer $a_0 * coeff_{a_0}$ et $a_n * coeff_{a_n}$ de l'autre côté du système linéaire, a_0 et a_n étant connus. On remplit donc la matrice du second membre du système linéaire ainsi:

$$M2(i_0) = \sum_{j=1}^{N} (coeff_{a_{i_0}} * (SM(x_j) - a_0 * coeff_{a_0}))\forall i_0 \ \epsilon[1,3]$$

et

$$M2(i_0) = \sum_{j=1}^{N} (coeff_{a_{i_0}} * SM(x_j))\forall i_0 \ \epsilon[4,n-4]$$

et

$$M2(i_0) = \sum_{j=1}^{N} (coeff_{a_{i_0}} * (SM(x_j) - a_n * coeff_{a_n}))\forall i_0 \ \epsilon[n-3,n-1]$$

Remarque : Il faut gérer à part les équations contenant a_0, a_1, a_{n-1} et a_n, leurs coefficients respectifs étant particuliers.

3.3.2 Résolution du système

Les conditions aux limites imposent des conditions sur deux termes du vecteur solution A:

- $A(0) = f(0)$
- $A(n) = f(1)$

où f est la fonction exacte de l'équation différentielle.
Pour satisfaire les conditions imposées par les B-splines on a aussi:

- $A(-1) = 2A(0) - A(1)$
- $A(n+1) = 2A(n) - A(n-1)$

Le système à résoudre va donc s'effectuer sur la matrice A des lignes 1 à $n-1$.

La matrice du premier membre étant redressée, il faut prendre en compte les changements d'indices pour les différentes phases de la résolution du système linéaire. La diagonale qui contient les pivôts pour la méthode de Gauss devient la colonne d'indice 0 dans la matrice du premier membre dont les colonnes vont de -3 à $+3$.
La triangulation du système linéaire devient en tenant compte des changements d'indices:

- $M(j, i+1-j : kfin-j) = M(j, i+1-j : kfin-j) - \frac{M(j,i-j)M(i,1:kfin-i)}{M(i,0)}$ avec $kfin = min(i + Nsurdiag, size(N))$ où $Nsurdiag$ est le nombre de sur diagonal de la matrice du premier membre et $size(N))$ le nombre de lignes du système.

- $N(j) = N(j) - \frac{M(j,i-j)N(i)}{M(i,0)}$

La résolution du système triangulaire devient alors:

- $A(i) = N(i) - \frac{M(i,1:min(Nsurdiag,size(N)-i)A(i+1:min(i+Nsurdiag,size(N)))}{M(i,0)}$

3.4 Les differentes courbes tracées

Les différentes courbes tracées permettent de visualiser le travail effectué par le programme. On peut donc observer la solution approchée et la solution exacte sur un même graphe, puis comparer les résultats obtenus pour des valeurs de pas h différentes soit pour des nombres de noeuds différents.
La résolution du système linéaire multidiagonal (obtenu par minimisation du critère $e(a) = \sum_{j=1}^{N} E(f)^2(X_j)$) et donc la connaissance des a_i dans

$$\sigma(x) \ = \sum_{i=-1}^{n+1} a_i B(\frac{x-\alpha}{h} - i)$$

permet de calculer la solution approchée $\sigma(x)$ de l'équation différentielle choisie pour N points équirépartis sur $[\alpha, \beta]$. La solution approchée $\sigma(x)$ et la fonction exacte $f(x)$ sont toutes les deux calculées aux mêmes points dans le module m_calcul_s_f.f90.
La comparaison graphique sous la forme de $D(h) \ = \frac{1}{N} * \sum_{j=1}^{N} (\sigma(x_j) - f(x_j))^2$ nécessite beaucoup plus de calculs. En effet à chaque nouvelle valeur du pas h soit du nombre de noeuds n il faut résoudre un nouveau système linéaire multidiagonal qui permet ensuite le calcul de $\sigma(x)$ dépendant de h soit du nombre de noeuds, et enfin il est possible de calculer $D(h)$. Ce travail est effectué dans le module m_comparaison.f90.
Le tracé de ces trois courbes est géré avec Gnuplot dans le module m_trace.f90. Les courbes de la solution approhée et de la solution exacte sont tracées avec une échelle normale alors que la comparaison graphique est effectuée avec une échelle logarithmique pour une plus grande clarté . Les points et les valeurs en ces points sont écrits dans trois fichiers différents:

- sol_approchée
- sol_exacte
- tracé_Dh

Gnuplot utilise donc les données contenues dans ces fichiers pour générer les graphes. A chaque fois, on propose à l'utilisateur d'imprimer le tracé obtenu.

Chapitre 4

Bilan

4.1 Conception d'un programme

Initialement nous avons analysé les différentes étapes générales nécessaires à la mise en oeuvre du projet. Cette première approche a guidé l'orientation de notre travail et a permis très rapidement un partage équitable des modules à concevoir au sein du binôme (7 et 8 modules). Tout en gardant un contact et une communication primordiale permanents, nous avons pu travailler séparément à l'élaboration du projet. Nous nous sommes très vites aperçues de l'interêt des échanges qui nous ont permis d'avancer, de rectifier, même de reconcevoir la circulation des données à l'intérieur du programme.
Ainsi des modules non initialement prévus ont vu le jour tandis que d'autres ont vu leurs arguments en appel changer plusieurs fois au cours de la conception du projet.

La liste des modules qui composent le projet est la suivante:

- makefile
- m_var_globale.f90
- m_choix_donnees.f90
- m_equa1.f90, m_equa2.f90, m_equa3.f90, m_equa.f90
- m_B_spline.f90
- m_B_derivee.f90
- m_systeme.f90
- m_resolution.f90
- m_calcul_s_f.f90

- m_comparaison.f90
- m_trace.f90
- principal.f90

4.2 Résultats

Les conditions aux limites sont respectées par la solution approchée si elles correspondent aux bornes extrèmes de l'intervalle de résolution $[\alpha;\beta]$. Si ce n'est pas le cas, les courbes obtenues sont trés éloignées de la solution exacte.

La précision des solutions approchées augmente bien sûr avec le nombre de noeuds des splines. On peut facilement évaluer cette caractéristique en analysant la comparaison grahique de l'erreur ($D(h)$), qui présente l'écart entre la solution exacte et la solution approchée pour différents nombres de noeuds n,(les variables n et h sont reliés par la relation: $h = (\beta - \alpha)/n$).

Quand on compare les résultats obtenus pour les trois équations différentielles, on constate que pour un même nombre de noeuds donnés l'exactitude de la solution approchée diminue avec l'augmentation de la complexité de l'équation différentielle, c'est à dire la courbure de la solution exacte.

Les résultats sont donc très positifs quant à l'utilisation des splines pour la résolution graphique d'équations différentielles avec conditions initiales. En effet, la solution approchée de la première équation $E(f)(x) = f''(x) + f'(x)$ avec comme conditions limites $f(0) = 0$ et $f(1) = 1$, recouvre la solution exacte $f(x) = \frac{sin1}{sinx}$ à partir de $n = 10$ noeuds. Ensuite, la courbe $D(h)$ permet d'observer que l'erreur varie en h^5 sur une échelle logarithmique. Les équations différentielles plus complexes nécessitent un nombre de noeuds des splines plus important, cependant les résultats obtenus peuvent atteindre une précision tout aussi remarquable.

4.3 Difficultés rencontrées

Au cours de ce projet, nous avons eu à gérer des fonctions en argument de sous-programme et donc à faire des interfaces pour ces fonctions, ce que nous n'avions jamais fait auparavant.
La circulation des données à l'intérieur du programme a demandé beaucoup d'attention notament au niveau des arguments en appel. Au fur et à mesure de l'avancement du projet de nombreux arguments ont été modifiés ce qui pouvait créer de nouvelles sources d'erreurs.

Par ailleurs nous avons été confrontées à la recherche de nos erreurs a partir des courbes obtenues, ce qui nécessitait la compréhension des erreurs à partir des résultats obtenus, et ce qui n'a pas été toujours facile. Nous avons dû travailler en essayant d'utiliser à la fois nos connaissances mathématiques et informatiques pour résoudre nos problèmes.

Conclusion

Ce projet consistait en une mise en oeuvre informatique de nos connaissances d'analyse numérique sur le problème particulier de la résolution d'équations différentielles linéaires.
Nous avons, en effet, déterminer des solutions particulières à partir d'équations en utilisant des splines cubiques pour des problèmes autres que l'interpolation.

De plus, nous avons pu examiner l'évolution de l'écart entre les solutions approchées et les solutions exactes pour différentes valeurs du pas h, ainsi que la vitesse de convergence. Les résultats sont donc très positifs quant à l'utilisation des splines pour la résolution graphique d'équations différentielles avec conditions initiales.

Enfin, ce projet a permis un partage équitable des modules à concevoir au sein du binôme. La conception du projet s'est déroulée en deux phases: une phase s'est effectuée en commun, l'autre phase a nécessité un travail plus personnel. Nous nous sommes réparties les tâches et avons travaillé séparément pour discuter ensuite des résultats obtenus et des problèmes rencontrés.
Nous avons de ce fait pu appréhender le travail de groupe qui nécessite échanges, écoute et cohésion.

Chapitre 5

Annexes

Sont répertoriés dans les annexes les graphes et le programme. Il est préférable toutefois de visualiser les courbes en exécutant le programme, les couleurs permettent en effet de mieux apprécier les résultats obtenus.

Voici la liste des différentes annexes :

- 1 Résultats obtenus avec l'équation différentielle 1 et les données suivantes:
- $E(f)(x) = f''(x) + f'(x)$ avec comme conditions limites $f(0) = 0$ et $f(1) = 1$.
  La spline utilisée a 10 noeuds et est calculée sur 100 points. L'intervalle de resolution est $[0.0; 1.0]$. La valeur du pivot minimum accepté est 0.000001.

- 2 Résultats obtenus avec l'équation différentielle 2 et les données suivantes:
- $E(f)(x) = f''(x) - 4f'(x) + 3f(x) - 10sinx$ avec comme conditions limites $f(0) = 0$ et $f(1) = -1$.
  La spline utilisée a 20 noeuds et est calculée sur 200 points. L'intervalle de resolution est $[0.0; 1.0]$. La valeur du pivot minimum accepté est 0.000001. Sont aussi représentés les résultats avec une spline de 50 noeuds puis 100 noeuds.

- 3 Résultats obtenus avec l'équation différentielle 3 et les données suivantes:
- $E(f)(x) = f''(x) - 4f'(x) + 4f(x) - 5coshx + 4sinhx$ avec comme conditions limites $f(0) = 0$ et $f(1) = cosh1$
  La spline utilisée a 30 noeuds et est calculée sur 300 points. L'intervalle de resolution est $[0.0; 1.0]$. La valeur du pivot minimum accepté est 0.000001. Sont aussi représentés les résultats avec une spline de 50 noeuds puis 100 noeuds.
- 4 Programme