PROJET DE PHYSIQUE P6-3

STPI 2011

N°5 : Diffusion de la chaleur dans une barre

Étudiants :

Clément OBERT
Hamza HAJ HAMOU
Bich Ngoc HO
Arnaud KHUN
Grégoire PERCHET
Laurent SOK

Enseignant-responsable du projet :

M. Bernard GLEYSE

Date de remise du projet :
18 juin 2011

Référence du projet:
STPI/P6-3/2011- n°5

Intitulé du projet :
Diffusion de la chaleur dans une barre

Type de projet :
Modélisation, expérimental

Objectifs du projet :
Le problème de la diffusion de la chaleur se ramenant rapidement à résoudre une équation aux différentielles partielles, l'équation de la chaleur, le premier objectif de notre projet fût donc de trouver les différentes formes que peut prendre cette équation suivant les conditions du problème étudié. Cela fait, notre second objectif fût de trouver les principales méthodes de résolution de l'équation de la chaleur. Finalement, notre dernier objectif consista à appliquer les méthodes trouvées à des cas simples, suivi d'une implémentation informatique afin d'en avoir une visualisation.

Mots-clefs du projet :
Chaleur – Discrétisation – Stationnaire - Unidimensionnel

Table des matières

1. Introduction

Contexte d'apparition

Les phénomènes de transfert (ou transport), dits irréversibles, sont causés par l'hétérogénéité d'une grandeur physique intensive entre deux systèmes. Le paramètre de température T est une de ces grandeurs qui va être responsable d'un phénomène de transport particulier. Il va s'agir ici d'un transfert d'énergie thermique ou *chaleur* qui peut s'effectuer suivant trois processus : *conduction, convection et rayonnement*. Notre étude porte sur la conduction thermique et plus précisément dans une barre.

Par définition, *« la conduction thermique*[1] *(ou diffusion thermique) est un des trois modes de transfert thermique (d'énergie) provoqué par une différence de température entre deux régions d'un même milieu, ou entre deux milieux en contact, et se réalisant sans déplacement global de matière (à l'échelle macroscopique). »*.

Ce phénomène de diffusion de la chaleur fut découvert et modélisé pour la première fois en 1811, par le célèbre physicien et mathématicien *Joseph Fourier*. C'est à Grenoble qu'il conduit ses expériences sur la propagation de la chaleur qui lui permettront de modéliser l'évolution de la température au travers de séries trigonométriques (séries de Fourier).

Pourquoi étudier ce phénomène?

Par extrapolation des principes physiques et mathématiques qui ont été abordés et étudiés et des résultats de nos travaux, on est en mesure de voir l'intérêt d'un tel projet. En effet, notre étude a globalement été une initiation à la modélisation mathématique et numérique de la diffusion thermique grâce à l'équation de la chaleur, cette dernière ayant également permis un développement et des recherches plus approfondies sur la diffusion en générale dans des domaines tels que la chimie, la métallurgie, et la fabrication micro électronique.

Pourquoi avoir choisi ce projet?

D'un point de vue personnel, il est à souligner que ce projet fut pour la moitié du groupe (soit 3 personnes sur 6) leur premier vœu dans le choix de projet à réaliser de l'EC P6-3. C'est à la fois la dimension calculatoire et de modélisation de ce phénomène naturel récurrent qui a suscité un vif intérêt chez les trois personnes concernées ci-avant.

Ce phénomène étant très complexe et son étude un véritable défi mathématique, il était impératif pour nous de délimiter notre étude afin d'approfondir au mieux les différents axes de recherche. Ces derniers peuvent se regrouper en deux points :

– recherche, calcul, modélisation du phénomène dans le cas unidimensionnel de l'équation de la chaleur;
– recherche, calcul, modélisation dans le cas stationnaire de cette équation.

1. Méthodologie et organisation du travail

[1]. *Définition extraite de l'encyclopédie libre en-ligne Wikipédia.*

Comme tout travail de groupe, sur un sujet qui en l'occurrence n'a jamais été vu ni traité, les trois premières séances étaient essentiellement de la recherche sur internet, une initiation au sujet posé grâce aux explications de M. Bernard Gleyse, et une familiarisation avec les principes physiques mis en jeu et les outils mathématiques à manipuler.

Suite à ce travail préliminaire, nous étions en mesure d'établir nos deux principales voies de recherches énoncées précédemment et nous nous sommes répartis le travail à réaliser. Afin de mieux visualiser notre organisation, nous avons érigé le tableau récapitulatif ci-contre :

TRAVAUX RÉALISÉS	DATE (semaine de cours du semestre 4)	MEMBRES RESPONSABLES
Recherche et **familiarisation** avec le sujet posé	SEMAINE 1 À 3	TOUS
Étude théorique (modélisation mathématique) A) cas unidimensionnel **[1]** B) cas stationnaire **[2]** Début programmation C++ **[3]**	SEMAINE 4 À 7	**[1]** Clément; Grégoire, Bich Ngoc **[2]** Clément, Arnaud Laurent, Hamza **[3]** Grégoire, Bich Ngoc
Élaboration informatique (modélisation numérique) - programmation du A) **[4]** - programmation du B) **[5]**	SEMAINE 8 à 12 (9 exclue)	**[4]** Grégoire, Bich Ngoc, Hamza **[5]** M. Bernard Gleyse, [Clément, Arnaud]
Réalisation du travail écrit	SEMAINE 13 et 14	TOUS
Mise en forme et dernières modifications du travail écrit; **Réalisation du travail pour l'oral** PowerPoint de présentation	SEMAINE 14	TOUS Laurent

Au cours de la progression du projet, nous avons été confrontés à des problèmes d'ordres techniques, à savoir la difficulté de transférer rapidement des informations et des fichiers. Pour y remédier, nous avons choisi d'essayer dans un premier temps le bureau virtuel Open Virtual Desktop (OVD) lancé par la start-up Ultéo, qui s'est malheureusement avéré difficile d'utilisation (compatibilité Windows OS). Nous avons donc opté pour la solution plus traditionnelle : l'utilisation d'un File Transfert Protocol chez l'hébergeur Free.

1. Travaux réalisés et résultats

2.

3.

3.1) Équation de la chaleur : cas unidimensionnel

$\forall (x,t) \in]0,L[\times]0,M[$on considère l'équation $(1)\frac{\partial T}{\partial t}(x,t) - a\frac{\partial^2 T}{\partial x^2}(x,t) = f(x,t)$associée aux conditions suivantes :

Conditions limites : $T(x,0) = 0\frac{\partial T}{\partial x}(L,t) = 0$

Condition initiale : $T(x,0) = U(x)$

a) Système **sans** source de chaleur

b)

On se place dans le cas où $f(x,t) = 0$. On peut alors résoudre littéralement l'équation (1). On utilisera ici la méthode de résolution par séparation des variables.

On suppose donc qu'il existe X(x) et F(t) telles que $T(x,t) = X(x) \times F(t)$

On calcule alors les dérivées qui nous intéressent :

$$\frac{\partial^2 T}{\partial x^2}(x,t) = X''(x) \times F(t)$$
$$\frac{\partial T}{\partial t}(x,t) = F'(t) \times X(x)$$

puis on remplace dans (1), qui devient alors : $(2)F'(t) \times X(x) - a \times F(t) \times X''(x) = 0$

On cherche une solution non nulle au problème, donc $\exists (x_0,t_0) telque X(x_0) \neq 0 et F(t_0) \neq 0$

On se place en (x_0,t):$\frac{F\prime(t)}{F(t)} = a\frac{X\prime\prime(x_0)}{X(x_0)} = K$puis en (x,t_0):$\frac{F\prime(t_0)}{F(t_0)} = a\frac{X\prime\prime(x)}{X(x)} = K$

On doit alors résoudre le système différentiel suivant :

$$(3)F'(t) - K.F(t) = 0$$
$$(4)X''(x) - \frac{K}{a}.X(x) = 0$$

On commence par résoudre (4) : 3 solutions envisageables selon le signe de

K :
$$1: X(x) = Ae^{x\sqrt{\frac{K}{a}}} + Be^{-x\sqrt{\frac{K}{a}}} si K > 0$$
$$2: X(x) = Ax + B si K = 0$$
$$3: X(x) = A\cos(x\sqrt{\frac{-K}{a}}) + B\cos(x\sqrt{\frac{-K}{a}}) si K < 0$$

On utilise les conditions aux limites pour éliminer des solutions :

$T(0,t) = 0 \Leftrightarrow X(0) \times F(t) = 0 \Leftrightarrow X(0) = 0 (onchercheTnonnulle)$

$\frac{\partial T}{\partial x}(L,t) = 0 \Leftrightarrow X'(L) \times F(t) = 0 \Leftrightarrow X'(L) = 0 (mêmeargument)$

$$1 : X(0) = A + B = 0 et T(L) = \sqrt{(\frac{K}{a})} A \times \mathrm{e}^{L\sqrt{(\frac{K}{a})}} - \sqrt{(\frac{K}{a})} A \times \mathrm{e}^{-L\sqrt{(\frac{K}{a})}} = 0$$

$$\Leftrightarrow A = B = 0 absurde$$

$$2 : X(0) = B = 0 et X'(L) = A \times L + B = 0$$

$$\Leftrightarrow A = B = 0 absurde$$

$$3 : X(0) = A \times \cos(0) + B \times \sin(0) = A = 0$$

$$X'(L) = B\sqrt{(\frac{K}{a})} \times \cos(L\sqrt{(\frac{K}{a})}) = 0 \Leftrightarrow B = 0 ou \cos(L\sqrt{(\frac{K}{a})}) = 0$$

$$B = 0 \Rightarrow absurde d'où \cos(L\sqrt{(\frac{K}{a})}) = 0$$

$$\Leftrightarrow L\sqrt{(\frac{K}{a})} = \frac{\pi(1+2\mathrm{n})}{2} \Leftrightarrow \sqrt{(\frac{K}{a})} = \frac{\pi(1+2\mathrm{n})}{2\mathrm{L}} = I_n$$

On obtient alors la famille de fonctions solutions suivantes :

$X_n(x) = B_n \sin(xI_n)$

On résout alors (3) :

$$F'(t) - K \times F(t) = 0 \Leftrightarrow F(t) = C\mathrm{e}^{Kt} avec K = -a(I_n)^2 < 0$$

$$\Leftrightarrow F_n(t) = C_n \mathrm{e}^{-a(I_n)^2 t}$$

Finalement, on obtient les solutions suivantes :

$T_n(x,t) = X_n(x) \times F_n(t) = D_n \times \sin(xI_n) \times \mathrm{e}^{-a(I_n)^2 t} avec D_n = C_n \times B_n$

Les solutions étant linéaires, on peut alors écrire la solution générale sous la forme convergente suivante :

$$T_g(x,t) = \sum_{n=1}^{+\infty} T_n(x,t)$$

On utilise maintenant la condition

On initialise afin de déterminer une solution particulière :

$$T_p(x,0) \sum_{n=1}^{+\infty} T_n(x,0) = \sum_{n=1}^{+\infty} D_n \times \sin(xI_n) = U(x)$$

On obtient donc une fonction développée en une série de Fourier réelle, de période 2L.

On détermine alors la solution finale suivante :

$$T(x,t) = \sum_{n=1}^{+\infty} D_n(U) \times \sin(xI_n) \times \mathrm{e}^{-a(I_n)^2 t} avec D_n(U) = \frac{2}{L}\int_0^L U(x) \times \sin(xI_n)dx$$

Enfin :

$$T(x,t) = \sum_{n=1}^{+\infty} (\frac{2}{L}\int_0^L U(x)\sin(xI_n)dx) \times \sin(xI_n) \times \mathrm{e}^{-a(I_n)^2 t} avec I_n = \frac{\pi(1+2\mathrm{n})}{2\mathrm{L}}$$

a) Système avec source de chaleur

On arrive maintenant au cas où $f(x,t) \neq 0$. Il est alors impossible dans la plupart des cas de résoudre littéralement l'équation (1). On recourt alors à la méthode des différences finies.

On commence donc par réaliser une discrétisation en temps et en espace du problème, en imposant un pas constant :

$\forall p, t_{p+1} - t_p = \tau$
$\forall n, x_{n+1} - x_n = h$

Avec l'approximation de T(x,t) par une suite : $T(x_n, t_p) \approx u_n^p$

On approxime les dérivées au voisinage des points u_n^p:

$$T(x_n + h, t_p + \tau) = T(x_n, t_p) + h\frac{\partial T}{\partial x}(x_n, t_p) + \tau\frac{\partial T}{\partial t}(x_n, t_p) + \frac{h^2}{2}\frac{\partial^2 T}{\partial x^2}(x_n, t_p) + h^2\varepsilon(h,\tau)$$

$$T(x_n - h, t_p + \tau) = T(x_n, t_p) - h\frac{\partial T}{\partial x}(x_n, t_p) + \tau\frac{\partial T}{\partial t}(x_n, t_p) + \frac{h^2}{2}\frac{\partial^2 T}{\partial x^2}(x_n, t_p) + h^2\varepsilon(h,\tau)$$

Ainsi,

$$T(x_n + h, t_p + \tau) + T(x_n - h, t_p + \tau) = 2\mathrm{T}(x_n, t_p) + 2\tau\frac{\partial T}{\partial t}(x_n, t_p) + h^2\frac{\partial^2 T}{\partial x^2}(x_n, t_p) + h^2\varepsilon(h,\tau)$$

De plus,

$$T(x_n, t_p + \tau) = \frac{1}{2}(T(x_n + h, t_p + \tau) + T(x_n - h, t_p + \tau))$$

On en déduit alors :

$$\frac{\partial T}{\partial t}(x_n, t_p) \approx \frac{1}{\tau}(T(x_n, t_{p+1}) - T(x_n, t_p))$$

$$\frac{\partial^2 T}{\partial x^2}(x_n, t_p) \approx \frac{1}{h^2}(T(x_{n+1}, t_p) - 2\mathrm{T}(x_n, t_p) + T(x_{n-1}, t_p))$$

Enfin, on remplace dans l'équation (1) :

$$\frac{1}{\tau}(u_n^{p+1} - u_n^p) - \frac{a}{h^2}(u_{n+1}^p - 2\mathrm{u}_n^p + u_{n-1}^p) = f_n^p$$

On obtient alors l'équation discrétisée du problème.

On sait que $u_0^p = 0, u_n^0 = U(x_n) et u_N^p - u_{N-1}^p = 0$donc par conséquent on peut calculer tous les éléments de la suite :

$$u_n^{p+1} = u_n^{p1} + \frac{\tau a}{h^2}(u_{n+1}^p - 2u_n^p + u_{n-1}^p) + \tau f_n^p$$

3.2) Équation de la chaleur : cas stationnaire

On se place dans un cas ou le temps est considéré comme n´intervenant plus dans le phénomène de diffusion au bout d´un certain laps de temps. Le problème devient alors stationnaire, et on se retrouve avec une équation de la forme suivante : $(1) - a\Delta u(x,y,z) = f(x,y,z)$

où f désigne une source de chaleur.

Il s´agit donc ici d´une équation de Poisson dans le cas d´une source non nulle, et d´une équation de Laplace dans le cas contraire.

Ici, on réduira le problème à 2 dimensions, dans un domaine fini : carré de cote d. L'équation (1) devient alors $(2) - a\frac{\partial^2 u}{\partial x^2}(x,y) - a\frac{\partial^2 u}{\partial y^2}(x,y) = f(x,y)$

a) Discrétisation du problème

On pose un même pas constant en x et en y :

$\forall p, y_{p+1} - y_p = h$
$\forall n, x_{n+1} - x_n = h$

On utilise les mêmes approximations des dérivées que pour le cas unidimensionnel. On obtient ainsi $\frac{\partial^2 u}{\partial x^2}(x_n,y_p) \approx \frac{1}{h^2}(u(x_{n+1},y_p) - 2u(x_n,y_p) + u(x_{n-1},y_p))$
$\frac{\partial^2 u}{\partial y^2}(x_n,y_p) \approx \frac{1}{h^2}(u(x_n,y_{p+1}) - 2u(x_n,t_p) + u(x_{n-1},y_{p+1}))$

En approximant u(x,y) et f(x,y) par :$u(x_n,y_p) \approx u_n^p et f(x_n,y_p) \approx f_n^p$,

on remplace dans (2) : $-\frac{a}{h^2}(u_{n+1}^p - 2u_n^p + u_{n-1}^p) - \frac{a}{h^2}(u_n^{p+1} - 2u_n^p + u_n^{p-1}) = f_n^p$

Et finalement : $\frac{a}{h^2}(4u_n^p - u_{n+1}^p - u_{n-1}^p - u_n^{p+1} - u_n^{p-1}) = f_n^p$

b) Formation du système linéaire

L'équation discrétisée $-(\Delta_{hk}u)_{ij}=f_{ij}$ ($1\leq i\leq n$ et $1\leq j\leq m$) est un ensemble de nxm équations scalaires linéaires. Donc on peut l'écrire sous la forme matricielle : AU=F.
Problème : U ne peut pas être aisément défini car c'est un vecteur à une colonne contenant nxm éléments alors que la notation « géométrique u_{ij} indique une matrice rectangulaire. Il faut donc numéroter le double indice (i,j) par un mono-indice l. On choisit : l(i,j)=(i-1)m+j, ce qui revient à numéroter les points du domaine de calcul colonne par colonne en partant de gauche.

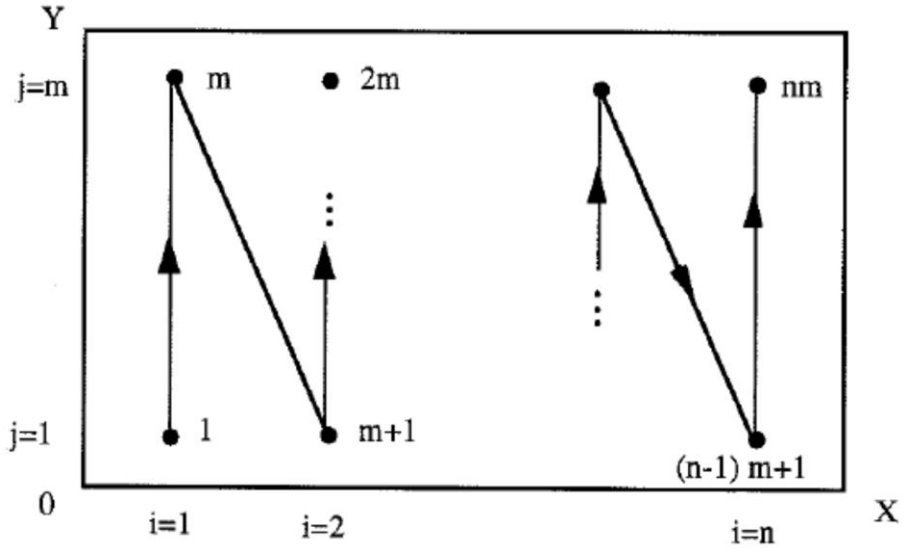

Schéma de la numérotation

Le vecteur U_k des inconnues est donc donné par : $U=\begin{pmatrix} u_{11} \\ u_{12} \\ \vdots \\ u_{1m} \\ u_{21} \\ \vdots \\ u_{2m} \\ \vdots \\ u_{i1} \\ \vdots \\ u_{im} \\ \vdots \\ u_{n1} \\ \vdots \\ u_{nm} \end{pmatrix}$.

Il faut trouver A avec la relation $-(\Delta_{hk}u)_{ij}=-\frac{1}{h^2}u_{i-1,j}-\frac{1}{k^2}u_{i,j-1}+2\left(\frac{1}{h^2}+\frac{1}{k^2}\right)u_{i,j}-\frac{1}{k^2}u_{i,j+1}-\frac{1}{h^2}u_{i+1,j}$.

Remplaçons l'indice double ij par le mono-indice l précédemment défini. Alors : (i,j)↔l, (i,j+1)↔l+1, (i,j-1)↔l-1, (i+1,j)↔l+m, (i-1,j)↔l-m, et $-(\Delta_{hk}u)_{ij}=f_{ij}\leftrightarrow -(\Delta_{hk}u)_l=f_l$ avec $-(\Delta_{hk}u)_l\equiv-\frac{1}{h^2}u_{l-m}-\frac{1}{k^2}u_{l-1}+2\left(\frac{1}{h^2}+\frac{1}{k^2}\right)u_l-\frac{1}{k^2}u_{l+1}-\frac{1}{h^2}u_{l+m}$.

La matrice A est ainsi décomposée en nxn blocs de taille mxm. De plus A est tridiagonale par blocs (seuls les blocs diagonaux ainsi que les blocs directement supérieurs et directement inférieurs aux blocs diagonaux sont non nuls). Les coefficients $\frac{-1}{h^2}$, $\frac{-1}{k^2}$et $2\frac{1}{h^2}+\frac{1}{k^2}$ ne dépendent pas de l, donc il faut identifier un bloc diagonal, un bloc de la diagonale supérieure et un bloc de la diagonale inférieure.

Sachant que u_{l-m} et u_{l+m} ont le même coefficient, les deux blocs non diagonaux sont égaux.

La matrice A est donc de la forme : $A=\begin{pmatrix} C & J & 0 & \cdots & 0 \\ J & C & J & \ddots & \vdots \\ 0 & J & C & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & J \\ 0 & \cdots & 0 & J & C \end{pmatrix}$ (tridiagonale de nxn blocs, chacun des blocs C et J étant lui-même une matrice mxm)

On en déduit : $J=\frac{-1}{h^2}\begin{pmatrix}1 & 0 & \cdots & 0\\ 0 & 1 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \cdots & 0 & 1\end{pmatrix}=\frac{-1}{h^2}I_m$ et $C=\begin{pmatrix}2\frac{1}{h^2}+\frac{1}{k^2} & \frac{-1}{k^2} & 0 & \cdots & 0\\ \frac{-1}{k^2} & \ddots & \ddots & \ddots & \vdots\\ 0 & \ddots & \ddots & \ddots & 0\\ \vdots & \ddots & \ddots & \ddots & \frac{-1}{k^2}\\ 0 & \cdots & 0 & \frac{-1}{k^2} & C\end{pmatrix}$

La matrice A est creuse, elle a à priori 5 éléments non nuls par ligne (sauf pour les points du bord où il y en a moins). On a donc au minimum 5nm termes non nuls au lieu de $(nm)^2$ pour une matrice de cet ordre.
A est symétrique définie positive (valeurs propres strictement positives).

c) Résolution par la méthode de Cholesky

A est symétrique définie positive, donc elle peut être mise sous la forme du produit d'une matrice triangulaire supérieure droite S et de sa transposée S^T.

$$\begin{matrix}A_{11} & A_{12} & A_{13} & A_{14}\\ A_{12} & A_{22} & A_{23} & A_{24}\\ A_{13} & A_{23} & A_{33} & A_{34}\\ A_{14} & A_{24} & A_{34} & A_{44}\end{matrix}=\begin{matrix}S_{11} & 0 & 0 & 0\\ S_{12} & S_{22} & 0 & 0\\ S_{13} & S_{23} & S_{33} & 0\\ S_{14} & S_{24} & S_{34} & S_{44}\end{matrix}\begin{matrix}S_{11} & S_{12} & S_{13} & S_{14}\\ 0 & S_{22} & S_{23} & S_{24}\\ 0 & 0 & S_{33} & S_{34}\\ 0 & 0 & 0 & S_{44}\end{matrix}$$

Effectuons littéralement le produit matriciel, soit : $A_{11}=S_{11}^2$, $A_{12}=S_{11}S_{12}$, $A_{13}=S_{11}S_{13}$, $A_{14}=S_{11}S_{14}$, $A_{22}=S_{12}^2+S_{22}^2$, $A_{23}=S_{12}S_{13}+S_{22}S_{23}$, $A_{24}=S_{12}S_{14}+S_{22}S_{24}$, $A_{33}=S_{13}^2+S_{23}^2+S_{33}^2$, $A_{34}=S_{13}S_{14}+S_{23}S_{24}+S_{33}S_{34}$, $A_{44}=S_{14}^2+S_{24}^2+S_{34}^2+S_{44}^2$.
Ces relations permettent de déterminer successivement les termes de la matrice triangulaire :
$S_{11}=\sqrt{A_{11}}$, $S_{12}=\frac{A_{12}}{S_{11}}$, $S_{13}=\frac{A_{13}}{S_{11}}$, $S_{14}=\frac{A_{14}}{S_{11}}$, $S_{22}=\sqrt{A_{22}-S_{12}^2}$, $S_{23}=\frac{A_{23}-S_{12}S_{13}}{S_{22}}$, $S_{24}=\frac{A_{24}-S_{12}S_{14}}{S_{22}}$, $S_{33}=\sqrt{A_{33}-S_{13}^2-S_{23}^2}$, $S_{34}=\frac{A_{34}-S_{13}S_{14}-S_{23}S_{24}}{S_{33}}$,
$S_{44}=\sqrt{A_{44}-S_{14}^2-S_{24}^2-S_{34}^2}$.

Sous forme matricielle condensée, le système Ax=B devient $S^TSx=B$. On définit le vecteur auxiliaire y=Sx. Le système peut alors être résolu par deux applications successives de la méthode de substitution, tout d'abord à $S^Ty=B \rightarrow y$ puis à $Sx=y \rightarrow x$.
La méthode de Cholesky est plus rapide que la méthode d'élimination de Gauss. La question est de savoir, dans chaque cas, si on est en droit de l'utiliser. Pour s'en assurer, il suffit de vérifier, au fur et à mesure de la factorisation, que l'argument des racines carrées est toujours strictement positif. D'autre part, de nombreux problèmes physiques reviennent à chercher le minimum d'une *forme quadratique*, de la forme : $F=\sum_i\sum_j A_{ij}x_ix_j$.

Pour annuler les dérivées partielles par rapport à chacune des variables indépendantes, on doit résoudre un système linéaire dont la matrice est nécessairement symétrique définie positive. Le nombre d'opérations élémentaires et le temps de calcul augmentent comme un polynôme du 3è degré en l'ordre *n* du système.

3.3) Modélisation informatique

2.

3.

a) Cas unidimensionnel

*Voir programme joint en annexe **(ANNEXE 1)**.*

Nous avons précédemment vu le cas unidimensionnel dans son aspect mathématique. Informatiquement, une représentation de la chaleur dans une barre est possible à l'aide d'une programmation informatique. L'objectif du programme est de calculer deux solutions de U, l'une à l'aide de la fonction propre à U et une seconde discrétisée obtenue à l'aide d'une formule fournie lors du premier cours par notre enseignant. Puis étudier les différences obtenues entre les valeurs des deux méthodes.

Pour programmer, le choix a été fait d'utiliser le C++. En effet, ce langage que nous avions déjà tous pratiqué se révèle être un langage de programmation adapté pour un programme constitué de nombreuses boucles « if » et d'un grand nombre incrémentations.

Nous avons cependant regretté l'absence d'une interface graphique simplifiée avec ce langage de programmation alors que d'autres tels que Maple ou le Java l'aurait proposé moyennant néanmoins certains procédés de programmation plus complexes.

D'abord, à l'aide des explications fournies par notre enseignant lors du premier cours, il fallut identifier la nature et le nombre des différentes variables qui allaient être utilisées (int ou double). Évidemment d'autres furent ajoutées plus tard mais ce premier travail permis de ne pas revenir constamment au début du programme pour changer la nature des variables qui le composait.

Par la suite vint la programmation de toutes les boucles permettant de mettre en place les conditions limite et les conditions au bord. Cette partie, consistant à reformuler informatiquement les formules mathématiques du cours, fut la plus délicate car la moindre erreur dans les paramètres de la boucle entraînait un effet boule de neige sur la suite du programme en créant davantage de termes que ceux voulus ou en renvoyant des valeurs erronées en certains points.

Ainsi des incrémentations se trouvèrent défaillantes dans notre programme entraînant la présence d'une dizaine de valeurs nulles pour l'une des fonctions calculées. De même, lorsque la boucle se prolongeait jusqu'à n+1 au lieu de s'arrêter à n, le nombre de valeurs obtenues se trouvait être faux.

Enfin si cette erreur n'était pas une erreur de syntaxe, elle n'était pas détectée lors de la compilation du programme ce qui compliquait davantage sa détection et faisait perdre beaucoup de temps dans l'avancement du programmation.

Après avoir établi les formules de calcul du U propres aux deux méthodes employées et corrigé les erreurs existantes après les avoir détectées dans ces méthodes de calcul, nous

imposé le respect d'une condition dite de « stabilité » à l'utilisateur sous laquelle le programme est autorisé a fonctionner. Le programme ne s'exécutant dans l'hypothèse où cette condition, imposée par l'équation de la chaleur, n'est pas respectée.

Pour compléter le programme, nous avons mis au point une méthode simple consistant en la soustraction des deux U pour obtenir l'écart entre les valeurs calculées. Ensuite nous l'avons affiché et avec les valeurs obtenues, nous avons vérifié que le programme fonctionnait correctement,

Afin de rendre le programme plus ouvert à l'usage par des utilisateurs extérieurs, l'interface à été simplifiée avec l'ajout de textes expliquant à l'utilisateur la démarche à suivre pour utiliser le programme et afficher clairement les valeurs obtenues.

De petits éléments d'amélioration visuelle furent apportés à la fin dans le programme lui-même tels que l'introduction de commentaires, la mise en place de tabulations pour comprendre la succession des boucles et surtout la mise en place de texte lors de l'affichage des résultats indiquant à quoi se rapporte chaque nombre affiché.

b) Cas stationnaire

*Voir programme joint en annexe **(ANNEXE 2)**.*

Comme pour le cas unidimensionnel ci-dessus, l'objectif du programme réalisé ici était de calculer les solutions de l'équation discrétisée de la chaleur, mais en régime stationnaire. Cependant, il n'était pas possible ici de calculer terme par terme la suite doublement indicée obtenue par la discrétisation, Il fallait poser les équations obtenues de manière à obtenir un système de plusieurs équations linéaires afin de faire apparaître une forme matricielle.

Le problème revenait alors à résoudre un système linéaire de la forme Ax=b, avec A une matrice carrée, et x et b des vecteurs. De nombreuses méthodes de résolutions de ce système étant possibles, nous avons regardé les spécificités de la matrice A. Celle-ci était en effet dans notre problème définie positive et tridiagonale par blocs. Nous avons par conséquent choisi la méthode de résolution de Cholesky.

Cette méthode a pour but de décomposer la matrice A en un produit de la forme Lt.L, où L

est une matrice triangulaire et Lt sa transposée. Le système devenait alors Lt.L.x=b. On pose alors y = Lx, et on trouve y en résolvant Lt.y=b, puis y en résolvant la première équation.

Le programme réalisé ici est donc une implémentation de la méthode de Cholesky, appliqué à un exemple : un carré de 1x1, avec un pas h=1/4 en x et en y (soit 16 points), avec f(x,y)=-16.

1. Conclusions et perspectives

Tout d'abord, nous tenions à dire que ce projet nous a, à tous, appris beaucoup plus que ce que nous espérions, et que nous voulions remercier particulièrement Monsieur Gleyse, pour son aide durant tout notre projet.

Ce projet nous a également servi d'expérience unique. En effet, nous n'étions pas spécialement proches à la base, mais nous avons tout au long du projet appris à nous connaître, et nous n'avons eu aucun mal à travailler en groupe. Certains travaillaient plus chez eux qu'en cours et inversement, mais le travail a toujours été réparti équitablement, et ce, à l'étonnement de la plupart, sans aucun problème. Cette expérience nous a donc appris à s'adapter avec les qualités et les défauts de chacun et à trouver comment travailler efficacement pour mener à bien notre projet.

Nous sommes cependant conscients que notre projet avec quelques semaines supplémentaires aurait pu être amélioré.

Tout d'abord, les programmes permettant de modéliser la diffusion de la chaleur dans une barre auraient pu être améliorés. Par exemple, nous aurions pu faire en sorte de rendre le programme plus « attractif », en insérant quelques touches de java dans ce dernier.

De même, avec le temps et l'équipement adéquat, il aurait été possible d'effectuer une expérience réelle en chauffant une barre de métal dans un laboratoire. Puis, à l'aide de sondes apposées et d'un ordinateur, il aurait été possible d'enregistrer le modèle obtenu afin de le comparer à la diffusion telle qu'elle est définie mathématiquement et au modèle fourni par nos programmes. Nous aurions pu voir à quel point notre étude théorique s'éloignait de l'étude pratique, et ainsi modifier notre projet pour qu'il se rapproche plus de la pratique.

Enfin, dans notre étude, nous nous sommes limités à des cas précis (unidimensionnel, stationnaire), et pour une étude plus poussée nous aurions pu travailler sur d'autres cas que ces derniers.

2. Bibliographie

Livres

1. **Théorie analytique de la chaleur** - Jean-Baptiste-Joseph Fourier (1822) – Édition F. Diderot
   (Consulté brièvement le 4 avril 2011)

2. ***C++ pour les nuls*** - *Stephen-Randy Davis et Matthew Telles* - Broché

Sites Internet

3. **Site du zéro** – Langage C++
Site disponible sur : http://www.siteduzero.com/

4. **Comment ça marche** – Les structures conditionnelles et les types de données
Site disponible sur : http://www.commentcamarche.net/
(page consultée le 21 mars 2011)

5. **Wikipédia, l'encyclopédie libre** – Erreur de segmentation (+ Segmentation Fault)
Site disponible sur : http://fr.wikipedia.org/
(page consultée le 4 avril 2011)

6. **Comment ça marche** – Qu'est ce qu'une erreur de segmentation
Site disponible sur : http://www.commentcamarche.net
(page consultée le 4 avril 2011)

7. **Wikipédia** – Séparation des variables
Site disponible sur : http://fr.wikipedia.org/wiki/S
%C3%A9paration_des_variables#.C3.89quation_aux_d.C3.A9riv.C3.A9es_partielles
(page consultée le 7 mars 2011)

8. **Wikipédia** – Equation de la chaleur
Site disponible sur :http://fr.wikipedia.org/wiki/%C3%89quation_de_la_chaleur
(page consultée le 7 mars 2011)

9. **Unité de Formation et de Recherche en Physique et Génie Electrique** – Travaux Pratiques
Site disponible sur : http://physique-eea.ujf-grenoble.fr/
(pageconsultée le 21 mars 2011)

10. **RÉSOLUTION NUMÉRIQUE DE L'ÉQUATION DE LAPLACE (1)**
Site disponible sur :http://perso.ensc-rennes.fr/jimmy.roussel/apprendre/laplace/index.html
(page consultée le 14 mars 2011)

11. **ÉQUATIONS DIFFÉRENTIELLES – GEN-0135**
PDF disponible sur http://web2.uqat.ca/lerene/Webcours/gen-0135/manuel/m41-0135.pdf
(page consultée le 7 mars 2011)

12. **Transfert thermique dans une barre calorifugée**
PDF disponible sur http://www.physique.enscachan.fr/
(page consultée le 7 mars 2011)

13. **Différences finies pour l'équation de Poisson à deux dimensions d'espace**
PDF disponible sur http://www.math.u-psud.fr/~fdubois/cours/idef/idef-ch5.pdf
(page consultée le 2 mai 2011)

3. Annexes

ANNEXE 1 :

Programme en C++ calculant la diffusion de la chaleur dans le cas unidimensionnel.

```
#include<stdlib.h>
#include<stdio.h>
#include<unistd.h>
#include<math.h>
main()
```

```
{

int p, n, Ng, Pg;

printf("Veuillez introduire un entier P :\n");                  // On demande à l'utilisateur d'entrer la valeur de l'ordonnée P
scanf("%d", &Pg);

printf("Veuillez introduire un entier N, avec N²≤P :\n");        // On demande à l'utilisateur d'entrer la valeur de l'abscisse N
scanf("%d", &Ng);

double u[Ng][Pg], tho, T, l, h, t[Pg], x[Ng], a, f[Ng][Pg], uo[Ng], dux[Ng][Pg], duxx[Ng][Pg], dut[Ng][Pg], v[Ng][Pg],
Ug[Ng][Pg], ecart[Ng][Pg];

l=1; T=0.5; a=1;                // On donne des valeurs aux constantes

tho=T/Pg;

h=l/Ng;

// Début de la boucle

if((Ng*Ng)<=Pg)
        {

        for(p=0;p<Pg; p++)
        {
       t[p]=p*tho;
       }

        for(n=0;n<Ng; n++)
                {
                x[n]=n*h;
                }


        for(p=0;p<Pg; p++)
                {
                u[0][p]=0;
                }

        for(n=0;n<Ng; n++)
                {
                u[n][0]=uo[n];
                }

        for(p=0;p<Pg; p++)
                {
                u[Ng][p]=u[Ng-1][p];
                }

        printf("Voici les valeurs de u\n");

        for(n=0;n<Ng; n++)
                {
                for(p=0;p<Pg; p++)
                        {
                        u[n][p]=x[n]*(1-x[n])*(1-x[n])*(1-3*x[n]*t[p])/(1+4*t[p]*t[p]);
                        dux[n][p]=(1-x[n])*(1-6*x[n]*t[p]-3*x[n]+12*x[n]*x[n]*t[p])/(1+4*t[p]*t[p]);
                        duxx[n][p]=-2*(2-18*x[n]*t[p]-3*x[n]+18*x[n]*x[n]*t[p]+3*t[p])/(1+4*t[p]*t[p]);
                        dut[n][p]=x[n]*(x[n]-1)*(x[n]-1)*(-3*x[n]+12*x[n]*t[p]*t[p]-8*t[p])/((1+4*t[p]*t[p])*(1+4*t[p]*t[p]));
                        f[n][p]=dut[n][p]-a*duxx[n][p];
                        printf("    u[%d][%d]= %lf\n",n,p,u[n][p]);                    // Affichage des valeurs de u
                        }
                }
```

```
	for(n=0;n<Ng; n++)
		{
		for(p=0;p<Pg; p++)
			{
			v[0][p+1]=0;
			v[n+1][p+1]=v[n+1][p]+tho*a*((v[n+2][p]-2*v[n+1][p]+v[n][p])+tho*f[n+1][p])/(h*h);
			v[n+1][p]=v[n][p];
			}
		}

	for(p=0;p<Pg; p++)
		{
		Ug[0][p]=0;
		}

	printf("Voici les valeurs de Ug\n");

	for(n=0;n<Ng; n++)
		{
		for(p=0;p<Pg; p++)
			{
			Ug[n+1][p+1]=Ug[n+1][p]+((tho*a)/(h*h))*(Ug[n+2][p]-2*(Ug[n+1][p])+Ug[n][p])+tho*f[n+1][p];
			printf("    Ug[%d][%d]= %lf\n",n,p,Ug[n][p]);                    // Affichage des valeurs de Ug
			}
		}

	printf("Voici les ecarts\n");

	for(n=0;n<Ng;n++)
		{
		for(p=0;p<Pg;p++)
			{
			ecart[n][p]=u[n][p]-Ug[n][p];
			printf("    ecart[%d][%d]= %lf\n",n,p,ecart[n][p]);              // Affichage des valeurs des
écarts entre les valeurs de u et les valeurs de Ug
			}
		}

	printf("Le resultat est stable. Merci !\n");

	}

else
	{
	printf("Le resultat n'est pas assez stable. Veuillez réessayer avec N²≤P !\n");
	} // Fin de la boucle
}
```

ANNEXE 2 :

Programme en C++ calculant la diffusion de la chaleur dans le cas stationnaire en 2 dimensions d'espace.

```
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

#define N2 3

const int n=9;
void matriceinf(double [n][n],int ,double [n][n]);
void transp_mat(double [n][n],double [n][n]);
void det_aux(double [n][n],double [n][n],int ,int );
double determinant(double [n][n],int );
```

```
double expo(int);
void coffacteur(double [n][n],double [n][n],int);
void inverse(double [n][n],double [n][n],int );
void multi_R(double ,double [n][n],double [n][n]);
void produit(double[n][n],double[],double[]);

int main() // Résolution du système Ax = b par la méthode de cholesky.
{
	// Blocs de la matrice A (tridiagonale par blocs).
	const double C[N2][N2] = {{4,-1,0},{-1,4,-1},{0,-1,4}};
	const double J[N2][N2] = {{-1,0,0},{0,-1,0},{0,0,-1}};
	double A[n][n];
	double L[n][n];
	double Lt[n][n];
	double b[n] = {1,1,1,1,1,1,1,1,1};
	double x[n];
	double y[n];

	for(int i=0;i<n;i++) // On construit la matrice A
	{
		for(int j=0;j<n;j++)
		{
			if(int(i/N2) == int(j/N2))
				A[i][j] = C[i-N2*int(i/N2)][j-N2*int(j/N2)];
			else if(int(i/N2) == int(j/N2) + 1 || int(i/N2) == int(j/N2) - 1)
				A[i][j] = J[i-N2*int(i/N2)][j-N2*int(j/N2)];
			else
				A[i][j]=0;
		}
	}

   printf("\nMatrice A\n");
	for(int i=0;i<n;i++) // On affiche la matrice A
	{
		for(int j=0;j<n;j++)
		{
			printf("%.2f ",A[i][j]);
		}
		printf("\n");
	}
   // On calcule et affiche la matrice L (triangulaire inférieure) telle que Lt*L=A
	printf("\nMatrice L\n");
	matriceinf(A,n,L);
	for(int i=0;i<n;i++)
	{
		for(int j=0;j<n;j++)
		{
			printf("%.2f ",L[i][j]);
		}
		printf("\n");
	}
	// On calcule et affiche la matrice Lt (transposee de L)
	transp_mat(L,Lt);
	printf("\n\nTransposee de L\n");
	for(int i=0;i<n;i++)
	{
		for(int j=0;j<n;j++)
		{
			printf("%.2f ",Lt[i][j]);
		}
		printf("\n");
	}
	double Lm2[n][n];
	inverse(L,Lm2,n);
	produit( Lm2,y,b);
```

```
	printf("\ny\n");
	for(int i=0;i<n;i++)
	{
		printf("%f \n",y[i]);

	}
	double Lm1[n][n];
	inverse(Lt,Lm1,n);
	produit( Lm1,x,y);

	printf("\nx\n");
	for(int i=0;i<n;i++)
	{
		printf("%f \n",x[i]);
	}
	printf("\n");

	printf("Carre discretise :\n");
	for(int i=0;i<4;i++)
	{
		for(int j=0;j<4;j++)
		{
			if(j == 0 || i == 3)
				printf("0.0000 ");
			else
				printf("%.4f ",x[3*(2-i)+j-1]);
		}
		printf("\n\n");
	}
	return 0;
}

void matriceinf(double mat[n][n],int N,double L[n][n])  //on determine la matrice inferieur L de mat
{
	double somme=0;
	for(int j=0;j<n;j++)
	{
		for(int i=0;i<n;i++)
		{
			if(i>j)
			{
				L[j][i]=0;
			}
			else
			{
				if(i==j)
				{
					for(int k=0;k<=i-1;k++)
					{
						somme=somme+L[i][k]*L[i][k];

					}
					L[i][i]=sqrt(mat[i][i]-somme);
				}
				else
				{
					for(int k=0;k<=i-1;k++)
					{
						somme=somme+L[j][k]*L[i][k];
					}
					L[j][i]=(mat[j][i]-somme)/L[i][i];
				}
			}
		somme=0;
		}
```

```
        }
}

/* Pour calculer l'inverse on utilise la methode transposee de la co matrice */


 void transp_mat(double A[n][n],double B[n][n]) //on opere la transposee de la matrice A
{
        for (int i=0;i<n;i++)
        {
                for (int j=0;j<n;j++)
                {
                        B[j][i]=A[i][j];
                }
        }
}

 void det_aux(double A[n][n],double B[n][n],int l,int c) //on calcul le mineur
{
        int d,e=0;
        for(int i=0;i<n;i++)
        {
                d=0;
                if(i!=l)
                {
                        for(int j=0;j<n;j++)
                        {
                                if(j!=c)
                                {
                                        B[e][d]=A[i][j];
                                        d++;
                                }
                        }
                        e++;
                }
        }
}

 double expo(int n) // on determine le signe
{
        if((n%2)==0) //il y avait differen avant
        {
                return (1);
        }
        else
        {
                return (-1);
        }
}

 double determinant(double A[n][n],int l) // on calcule le determinant
{
        double B[n][n],x=0;
        if(l==1)
        {
                return A[0][0];
        }
        for(int i=0;i<l;i++)
        {
                det_aux(A,B,i,0);
                x=x+(expo(i)*A[i][0]*determinant(B,(l-1)));
        }
        return x;
}
// Remarque: det(A)= produit des L(i,i)^2
```

```
 void multi_R(double a,double A[n][n],double B[n][n]) // on multiple l inverse du determinant par la matrice A pour avoir B
linervse
{
            for(int i=0;i<n;i++)
      {
                  for(int j=0;j<n;j++)
            {
                        B[i][j]=A[i][j]*a;
                  }
            }
}

 void coffacteur(double A[n][n],double B[n][n],int l) // on determine le coffacteur
{
            double C[n][n];
            if (l==1)
            {
                        B[0][0]=1;
            }
            else
            {
                        for (int i=0;i<l;i++)
                        {
                                    for (int j=0;j<l;j++)
                                    {
                                                det_aux(A,C,i,j);
                                                B[i][j]=expo(i+j)*determinant(C,(l - 1));
                                    }
                        }
            }
}

 void inverse(double A[n][n],double B[n][n],int l) //on determine l inverse
{
            double C[n][n],D[n][n],d;
            d=(1/determinant(A,l));
            coffacteur(A,C,l);
            transp_mat(C,D);
            multi_R(d,D,B);
}


void produit(double A[n][n],double p[],double b[]) //on fait le produit des deux matrices
{
            double som=0;
            int i,j;
            for(i=0;i<=n-1;i++)
            {
                        som=0;
                        for(j=0;j<=n-1;j++)
                        {
                                    som=som+A[i][j]*b[j];
                        }
                        p[i]=som;
            }
}
```