

## Conduction thermique



Enseignant responsable  
Bernard GLEYSE

Étudiants :  
Anne Charlotte PASQUET  
Marie Prud'homme  
Jean-François ROBIN  
Yao YAO



---

**Date de remise du rapport :** 15/06/15

**Référence du projet :** STPI<sup>1</sup>/P6/2016 – 14

**Intitulé du projet :** Conduction thermique

**Type de projet :** Simulation, modélisation

**Objectifs du projet :**

Notre projet a pour objectif d'étudier la conduction thermique, et de comparer différentes méthodes de calcul pour évaluer la température en un point donné et à un temps donné, le long d'une barre chauffée, en régime permanent et soumise à des contraintes extérieures constantes (un certain temps est nécessaire pour que l'équilibre s'effectue dans la barre). Pour cela, nous devons mettre en place différentes méthodes de résolutions informatiques et mathématiques telles que l'implémentation de la méthode explicite ou encore le calcul de la solution exacte, puis nous devons comparer ses résultats obtenus par ces différentes méthodes avec les résultats expérimentaux.

**Mots-clefs du projet :** Équation de la chaleur, méthode explicite et implicite, expérience

---

1. INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE ROUEN  
DÉPARTEMENT SCIENCES ET TECHNIQUES POUR L'INGÉNIEUR  
685 AVENUE DE L'UNIVERSITÉ BP 08- 76801 SAINT-ETIENNE-DU-ROUVRAY  
TÉL : 33 2 32 95 66 21 - FAX : 33 2 32 95 66 31

# Table des matières

# Introduction

Au cours de notre 2<sup>ème</sup> année de STPI au sein de l'INSA de Rouen, nous avons réalisé un projet physique. Ce projet s'est déroulé durant le 4<sup>ème</sup> semestre, en groupe de 4 personnes, pour notre part. Nous avons effectué ce projet sur la conduction thermique.

La conduction thermique est un mode de transfert thermique provoqué par une différence de température entre deux régions d'un même milieu, se réalisant sans déplacement de matière. En effet, il existe aussi la convection et le rayonnement comme autres modes de transfert thermique qui font respectivement appel à un déplacement de matière et au rayonnement électromagnétique émis par un corps, mais ces derniers ne feront pas l'objet de notre étude.

En effet, nous allons étudier plus particulièrement le cas de la diffusion thermique à travers une barre métallique où seule la conduction thermique intervient. En effet, nous avons dû restreindre notre étude tant ce sujet est particulièrement riche.

Nous allons essayer, dans ce rapport, de démontrer ce phénomène et de l'expliquer à l'aide d'équations, de programmes...

Pour nous aider, nous avons eu accès aux rapports des années précédentes. Cependant, nous avons adapté le sujet à notre vision.

Ce projet nous a aussi permis de travailler en équipe, ce qui est une bonne formation pour notre futur métier. Communication, entraide et organisation sont les mots-clés dans le travail de l'ingénieur et dans le cadre de ce projet.

# Chapitre 1

## Méthodologie, organisation du travail

Nous avons commencé l'étude de ce sujet par des recherches informatiques. Nous avons mis un certain temps à assimiler le sujet. En effet, la lecture et la compréhension des rapports des années précédentes a pris quelques semaines. Ensuite, nous avons déterminé les aspects du sujet que nous voulions approfondir, puis, nous nous sommes répartis les tâches. Ce sujet allie plusieurs compétences. Ainsi chacun a pu y trouver son compte. Le tableau suivant explique notre répartition du travail.

Répartition du travail :

Tâche à effectuer	Membres en charge	Semaines
Familiarisation avec le sujet	Tout le monde	1 à 5
Travail sur l'équation de la chaleur	Tous	5
Approche mathématique : Dérivée, coefficient de diffusion	Marie, Anne Charlotte et Yao	6
Intégration du coefficient de diffusion dans les programmes	Anne Charlotte, Marie	7 à 9
Recherche de solutions qui se rapprochent le plus du modèle expérimental :	Marie et Anne Charlotte	9 à 10
Méthode de résolution explicite en 1D + compréhension méthode explicite	Jean François, Marie et Anne Charlotte	8 à 10
Recherche de la valeur exacte de l'équation de la chaleur	Yao	8 à 12
Transformation du code pascal en Python	Jean François	8 à 12
Exploitation de nos résultats sous Maple	Yao	12 à 13
Modélisation	Anne Charlotte et Jean François	12 à 13
Rédaction du rapport	Anne Charlotte et Marie	9 à 13

# Chapitre 2

## Définition de l'équation de la chaleur et méthodes de résolution

### 2.1 Définition de l'équation de la chaleur

Nous considérons une barre comme un volume conducteur, solide et homogène. Nous lui appliquons le premier principe de la thermodynamique (vu en P1) :  $dU = \delta W + \delta Q$  (où  $dU$  est la variation d'énergie interne dans l'intervalle entre  $t$  et  $t + dt$ , et où  $\delta W$  et  $\delta Q$  sont respectivement les formes différentielles du travail et de l'énergie thermique, toutes ces grandeurs étant exprimées en J). Nous considérons la barre comme un système isochore, d'où  $dV = 0$  et donc  $\delta W = -PdV = 0$ . Nous avons donc :

$$dU = \delta Q$$

$$\text{Et } U(t) = \int \int \int_V \rho c T(t) dV + f(V)$$

Avec :

$\rho$  la masse volumique (exprimée en  $kg.m^{-3}$ )

$c$  la chaleur spécifique massique du matériau (exprimée en  $J.kg^{-1}.K^{-1}$ )

$f(V)$  une fonction du volume

$$\text{Nous avons donc : } dU(t) = U(t + dt) - U(t) = \int \int \int_V \rho c dT dV.$$

$T$  étant fonction du temps, nous pouvons écrire  $dT = \frac{\partial T}{\partial t} dt$ , ce qui donne quand nous le remplaçons dans la formule ci-dessus :

$$dU(t) = \int \int \int_V \rho c \frac{\partial T}{\partial t} dt dV$$

La chaleur reçue est la somme de la chaleur gagnée par échange avec l'extérieur, c'est à dire le flux de chaleur  $\phi$  en  $J.s^{-1}$ , et de celle produite dans la barre, que l'on écrit  $F(M)$  en  $J.s^{-1}$ .

On écrit donc :

$$\delta Q = \phi dt + F(M) dt.$$

Or  $\phi = - \int \int_{\Sigma} \varphi dS$  avec  $\varphi$  la densité de flux, exprimée en  $J.m^{-2}.s^{-1}$  (le flux est négatif, ce qui explique la présence du signe -).

En appliquant la loi de Fourier, on obtient donc  $\phi = - \int \int_{\Sigma} -k_{tr} \overrightarrow{\text{grad}}(T) dS$ , avec  $-k_{tr}$  la conductivité thermique du milieu, exprimée en  $J.m^{-1}.K^{-1}.s^{-1}$ .

On utilise alors le théorème de Green-Ostrogradsky, ou théorème de flux-divergence, qui donne une intégrale triple sur le volume pour  $\phi$ .

$$\phi = \int \int \int_V \text{div}(k_{tr} \overrightarrow{\text{grad}}(T)) dV$$

On pose ensuite  $F(M) = \int \int \int_V f(M) dV$ , où  $f(M)$  est la fonction densité de source de chaleur, exprimée en  $J.s^{-1}$ . En combinant tous les résultats, nous avons alors :

$$\int \int \int_V \rho c \frac{\partial T}{\partial t} dt dV = \int \int \int_V k_{tr} \Delta(T) dV + \int \int \int_V f(M) dV$$

et donc

$$\rho c \frac{\partial T}{\partial t} = k_{tr} \Delta(T) + f(M)$$

L'équation de la chaleur est donc donnée par :

$$f(M) = \rho c \frac{\partial T}{\partial t} - k_{tr} \Delta(T)$$

## 2.2 Méthodes de résolution

Il s'agit de méthodes de résolution d'équations différentielles de la forme :

$$\frac{\partial F}{\partial t} = A \frac{\partial^2 F}{\partial x^2} + B \frac{\partial F}{\partial x} + CF$$

### 2.2.1 Méthode explicite

Résolution de l'équation de la chaleur en une dimension

Pour l'ensemble de nos modélisations, nous posons les conditions aux limites suivantes, basées sur l'expérience de l'année 2013/2014 :

$$T_0^t = 27,1$$

$$T_N^{t+1} = 1,8 + T_{N-1}^{t+1}$$

et la condition initiale :

$$T_n^0 = 25,6^\circ\text{C} \text{ (la température ambiante)}$$

On pose  $\forall n, x_n - x_{n-1} = h$  avec  $h$  le pas de l'espace  
 et  $\forall p, t_n - t_{n-1} = \tau$  avec  $\tau$  le pas de temps.

La méthode explicite permet de calculer la température le long de la barre à tout instant  $t$  à partir des température à l'instant précédent  $t-1$ . Cette méthode repose sur une relation de récurrence. On cherche donc à approximer  $T(x, t)$  par une suite  $T(x_n, t_p) \approx T_n^p$ .

Nous voulons donc approximer les dérivées présentes dans l'équation de la chaleur au voisinage des points  $T_n^p$ . Nous commençons alors par la formule de Taylor-Young à l'ordre 2 :

$$T(x_n + h, t_p - \tau) = T(x_n, t_p) + h \frac{\partial T}{\partial x_n}(x_n, t_p) - \tau \frac{\partial T}{\partial t_p}(x_n, t_p) + \frac{1}{2} \left( \frac{\partial^2 T}{\partial x_n^2}(x_n, t_p) h^2 - h \tau \frac{\partial^2 T}{\partial t_p \partial x_n}(x_n, t_p) + \tau^2 \frac{\partial^2 T}{\partial t_p^2}(x_n, t_p) \right) - \tau h \epsilon(\tau h)$$



Or  $\frac{1}{2}(-h\tau\frac{\partial^2 T}{\partial t_p \partial x_n}(x_n, t_p) + \tau^2\frac{\partial^2 T}{\partial t_p^2}(x_n, t_p)) - \tau h\epsilon(\tau h)$  est négligeable devant  $T(x_n, t_p)$  et ces termes n'interviennent pas dans l'équation de la chaleur nous les négligeons donc. D'où :

$$T(x_n + h, t_p - \tau) = T(x_n, t_p) + h\frac{\partial T}{\partial x_n}(x_n, t_p) - \tau\frac{\partial T}{\partial t_p}(x_n, t_p) + \frac{1}{2}h^2\frac{\partial^2 T}{\partial x_n^2}(x_n, t_p) \quad (2.1)$$

En procédant de même, nous obtenons :

$$T(x_n - h, t_p - \tau) = T(x_n, t_p) - h\frac{\partial T}{\partial x_n}(x_n, t_p) - \tau\frac{\partial T}{\partial t_p}(x_n, t_p) + \frac{1}{2}h^2\frac{\partial^2 T}{\partial x_n^2}(x_n, t_p) \quad (2.2)$$

Ensuite, nous additionnons les deux formules de Taylor-Young (2.1) et (2.2) :

$$T(x_n + h, t_p - \tau) + T(x_n - h, t_p - \tau) = 2T(x_n, t_p) - 2\tau\frac{\partial T}{\partial t_p}(x_n, t_p) + h^2\frac{\partial^2 T}{\partial x_n^2}(x_n, t_p) \quad (2.3)$$

Pour retrouver la différentielle d'ordre 1 présente dans l'équation de la chaleur, nous posons  $h=0$  et nous obtenons :

$$\frac{\partial T}{\partial t_p}(x_n, t_p) = \frac{T(x_n, t_p) - T(x_n, t_p - \tau)}{\tau}$$

Puis pour trouver la différentielle d'ordre 2 présente dans l'équation de la chaleur, nous remplaçons cette différentielle d'ordre 1 dans (2.3) :

$$\begin{aligned} T(x_n + h, t_p - \tau) + T(x_n - h, t_p - \tau) &= 2T(x_n, t_p) - 2\tau\frac{T(x_n, t_p) - T(x_n, t_p - \tau)}{\tau} + h^2\frac{\partial^2 T}{\partial x_n^2}(x_n, t_p) \\ \iff \frac{\partial^2 T}{\partial x_n^2}(x_n, t_p) &= \frac{T(x_n + h, t_p - \tau) + T(x_n - h, t_p - \tau) - 2T(x_n, t_p - \tau)}{h^2} \end{aligned}$$

Il ne nous reste plus qu'à remplacer ces approximations dans l'équation de la chaleur qui est de la forme  $\frac{\partial T}{\partial t_p} - a\frac{\partial^2 T}{\partial x_n^2} = f(x_n, t_p)$  avec  $a$  le coefficient de diffusion, on trouve alors :

$$\begin{aligned} \frac{T(x_n, t_p) - T(x_n, t_p - \tau)}{\tau} - a\frac{T(x_n + h, t_p - \tau) + T(x_n - h, t_p - \tau) - 2T(x_n, t_p - \tau)}{h^2} &= f(x_n, t_p) \\ \iff \frac{T_n^p - T_n^{p-1}}{\tau} - a\frac{T_{n+1}^{p-1} + T_{n-1}^{p-1} - 2T_n^{p-1}}{h^2} &= f(x_n, t_p) \end{aligned}$$

Or nous considérons  $f(x_n, t_p) = 0$  (ce terme ne devrait pas être nul car il y a une source de chaleur, mais pour simplifier le résultat, ce terme a été remplacé par un flux constant), ce qui nous amène à :

$$T_n^p = \frac{\tau a}{h^2}T_{n-1}^{p-1} + (1 - 2\frac{\tau a}{h^2})T_n^{p-1} + \frac{\tau a}{h^2}T_{n+1}^{p-1}$$

Application informatique :

Résoudre cette équation pour des points tout le long de la barre prendrait énormément de temps et serait source d'erreurs. C'est pour cela qu'il est plus intéressant de développer des programmes capables de faire ces calculs à notre place. C'est dans ce but que nous avons essentiellement travaillé à l'amélioration des programmes déjà existants des années précédentes.

La méthode explicite est une méthode rapide mais instable si le pas de temps n'est pas suffisamment petit.

### 2.2.2 Méthode implicite

La méthode de résolution implicite repose sur le même principe que l'explicite mais cette fois ci elle s'effectue au temps  $t_p + \tau$ . En effet la méthode implicite permet de déterminer la chaleur en un point de la barre à un temps t donné à partir des températures au même point mais à l'instant d'après, soit à  $t+1$ .

On trouve : 
$$\frac{\partial T}{\partial t_p}(x_n, t_p) = \frac{T(x_n, t_p + \tau) - T(x_n, t_p)}{\tau}$$

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

en remplaçant les dérivées partielles dans l'équation de la chaleur par ces approximations, on obtient :

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

$$\iff \tau f_n^{p+1} + T_n^p = \frac{-a\tau(T_{n+1}^{p+1} + T_{n-1}^{p+1})}{h^2} + T_n^{p+1}(1 + \frac{2a\tau}{h^2})$$

On peut alors calculer cette formule pour n allant de 1 à N-1, dans les conditions de l'expérience :

Pour n=1 : 
$$\tau f_1^{p+1} + T_1^p = \frac{-a\tau(T_2^{p+1} + T_0^{p+1})}{h^2} + T_1^{p+1}(1 + \frac{2a\tau}{h^2})$$

$$\iff \tau f_1^{p+1} + T_1^p = \frac{-a\tau(T_2^{p+1})}{h^2} - 27.1 \frac{a\tau}{h^2} + T_1^{p+1}(1 + \frac{2a\tau}{h^2}) \text{ car } T_0^{p+1} = 27.1$$

Pour n=2 : 
$$\tau f_2^{p+1} + T_2^p = \frac{-a\tau T_3^{p+1}}{h^2} + T_2^{p+1}(1 + \frac{2a\tau}{h^2}) - T_1^{p+1} \frac{a\tau}{h^2}$$

...

Pour n=N-1 : 
$$\tau f_{N-1}^{p+1} + T_{N-1}^p = -1.8 \frac{a\tau}{h^2} - T_{N-1}^{p+1}(1 + \frac{a\tau}{h^2}) - \frac{a\tau}{h^2} T_{N-2}^p$$

On obtient alors la matrice suivante :

$$\begin{pmatrix} \tau f_1^{p+1} + T_1^p \\ \tau f_2^{p+1} + T_2^p \\ \vdots \\ \vdots \\ \tau f_{N-1}^{p+1} + T_{N-1}^p \end{pmatrix} = \begin{pmatrix} 1 + \frac{2a\tau}{h^2} & -\frac{a\tau}{h^2} & 0 & \dots & \dots & 0 \\ -\frac{k\tau}{h^2} & 1 + \frac{2k\tau}{h^2} & -\frac{a\tau}{h^2} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \dots & \vdots \\ 0 & \dots & \dots & 0 & 1 + \frac{a\tau}{h^2} & -\frac{a\tau}{h^2} \end{pmatrix} \begin{pmatrix} T_1^{p+1} \\ T_2^{p+1} \\ \vdots \\ \vdots \\ T_{N-1}^{p+1} \end{pmatrix}$$

Contrairement à la méthode explicite, la méthode implicite est lente mais stable.

### 2.2.3 Généralisation : le $\theta$ schéma

Il existe un moyen de généraliser les formules de résolution des méthodes explicite, implicite et semi-implicite, c'est le  $\theta$  schéma.

On considère l'équation :

$$\frac{u_j^{n+1} - u_j^n}{\tau} - \theta a \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{h^2} - (1 - \theta)a \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2} = \theta f_j^{n+1} + (1 - \theta)f_j^n$$

Cette équation permet de retrouver les formules vues précédemment pour les résolutions des différentes méthodes.

Si l'on remplace  $\theta = 0$ , on retrouve la formule de la méthode explicite soit :

$$\frac{u_j^{n+1} - u_j^n}{\tau} - a \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2} = f_j^n$$

Si l'on remplace  $\theta = 1$ , on retrouve la formule de la méthode implicite soit :

$$\frac{u_j^{n+1} - u_j^n}{\tau} - a \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{h^2} = f_j^{n+1}$$

De même, si l'on remplace  $\theta = \frac{1}{2}$ , on obtient la formule de la résolution de la méthode de Crank-Nicolson, soit :

$$\frac{u_j^{n+1} - u_j^n}{\tau} - \frac{1}{2}a \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2} - \frac{1}{2}a \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{h^2} = \frac{1}{2}f_j^n + \frac{1}{2}f_j^{n+1}$$

La méthode de Crank- Nicolson, aussi appelée semi-implicite conjugue la méthode explicite et la méthode implicite et est inconditionnellement stable.

Toutes ces méthodes peuvent en fait s'exprimer de façon unique en fonction d'un paramètre  $\theta$  (c'est le paramètre d'implicité) qui désigne une méthode explicite ( $\theta = 0$ ), implicite ( $\theta = 1$ ) ou Cranck-Nicholson ( $\theta = \frac{1}{2}$ ).

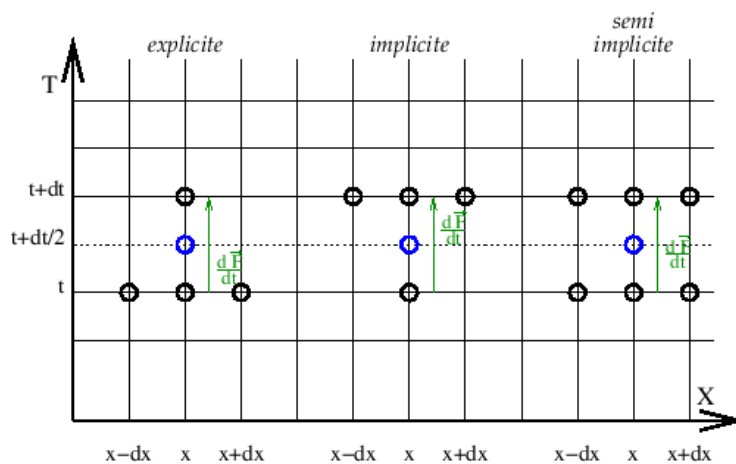


FIGURE 2.1 – Méthodes explicites, implicites, semi-implicites

## 2.3 Calcul de la solution exacte

Nous avons traité la résolution de la solution exacte dans le cas général avec une source de chaleur continue. Voici l'équation de la chaleur à partir de laquelle nous avons basé nos calculs.

$$f(x) = -a \frac{\partial^2 U}{\partial x^2} + \frac{\partial U}{\partial t} \quad (1)$$

On pose :  $U(x, t) = V(x, t) + \epsilon(x, t)$ ;

$$U(0, t) = \phi(t) = U_0 \text{ soit } \epsilon(0, t) = \phi(t)$$

$$\text{Et } U_x(L, t) = \frac{\partial U}{\partial x}(L, t) = \psi(t) = U'_L \text{ soit } \epsilon_x(L, t) = \phi(t)$$

$$\text{Donc on a } V(0, t) = 0 \text{ et } V_x(L, t) = \frac{\partial V}{\partial x}(L, t) = 0.$$

Par ailleurs,  $\epsilon(x, t)$  est de la forme  $a(t)x + b(t)$

$$\text{avec } \epsilon(0, t) = b(t) = \phi(t) = U_0 = 27, 1 \quad \text{et } \epsilon_x(L, t) = a(t) = \psi(t) = U'_L = 1, 55$$

Ainsi on obtient :  $\epsilon(x, t) = 1, 55x + 27, 1$

Ensuite, on calcule avec  $V(x, t)$

Les conditions sont :

$$V(0, t) = 0; V_x(L, t) = 0; V(x, 0) = g(x)$$

D'une part,  $V(x, 0) = g(x) = U(x, 0) - \epsilon(x, 0)$ . Ici on pose  $U(x, 0)$  qui correspond à la température ambiante et  $U(x, 0) = 25, 6$

$$\text{avec } \epsilon(x, 0) = 1, 55x + 27, 1;$$

$$\text{et on a bien } V(x, 0) = g(x) = -1, 55x - 1, 5.$$

Maintenant, recherchons la fonction propre.

$$\text{L'équation dérivée partielle homogène } -a \frac{\partial^2 V}{\partial x^2} + \frac{\partial V}{\partial t} = 0 \quad (2)$$

$$\text{Donc la solution de (2) : } X_n(x)T_n(t) = \sin(k_n x)e^{-(k_n^2 at)} \text{ avec } k_n = \frac{(2n+1)\pi}{2L}$$

La fonction  $X_n(x) = \sin(k_n x)$  vérifie les conditions limites de (1)

Supposons que  $f(x, t)$  puisse s'écrire sous la forme :

$$f(x, t) = \sum_{n=1}^{\infty} C_n(t)X_n(x) = \sum_{n=1}^{\infty} C_n(t)\sin(k_n x)$$

Ce développement est une série de Fourier d'où

$$C_n(t) = \frac{2}{L} \int_0^L f(x, t)\sin(k_n x)dx$$

Maintenant considérons l'équation aux dérivées partielles homogène, du problème (1)

$$-a \frac{\partial^2 V}{\partial x^2} + \frac{\partial V}{\partial t} = C_n(t)X_n(t) \quad (3)$$

On cherche une solution de la forme  $u_n(t)X_n(x)$

L'équation dérivée partielle (3) devient :

$$u'(t)X_n(x) = au(t)X_n''(x) + C_n(t)X_n(t)$$

$$\text{Or, } X_n''(x) = -k_n^2 X_n(x) \text{ avec } X_n(x) = \sin(k_n x)$$

L'équation différentielle vérifiée pour  $u$  est :

$$u'(t) = -k_n^2 a u(t) + C_n(t)$$

Donc, la solution est :

$$u_n(t) = e^{-k_n^2 a t} (A_n + \int_0^t e^{k_n^2 a t} C_n(t) dt) \text{ avec } A_n \text{ constant}$$

On obtient la solution de l'équation dérivée partielle du (1) avec les conditions limites :

$$V(x, t) = \sum_{n=1}^{\infty} u_n(t) X_n(x)$$

$$\iff V(x, t) = \sum_{n=1}^{\infty} (e^{-k_n^2 a t} (A_n + \int_0^t e^{k_n^2 a t} C_n(t) dt)) \sin(k_n x) \quad (4)$$

Considérons la condition limite  $V(x, 0) = g(x) = U(x, 0) - \epsilon(x, 0)$

Et on prend  $t=0$  pour (4)  $V(x, 0) = \sum_{n=1}^{\infty} A_n \sin(k_n x)$

Ce développement est une série de Fourier d'où

$$A_n = \frac{2}{L} \int_0^L V(x, 0) \sin(k_n x) dx$$

Ainsi, en prenant  $f(x,t)=1$

On obtient  $C_n(t) = \frac{4}{(2n+1)\pi}$

Donc  $f(x, t) = \sum_{n=1}^{\infty} C_n(t) X_n(x) = \sum_{n=1}^{\infty} (\frac{4}{(2n+1)\pi}) \sin(k_n x)$

Et puis,  $A_n = \frac{2}{L} \int_0^L (-1.55x - 1.5) \sin(k_n x) dx$

$$\iff A_n = \frac{2}{L} (\frac{1.55}{k_n^2} (-1)^{n+1} - \frac{1.5}{k_n})$$

$u_n(t) = e^{-k_n^2 a t} (A_n + \int_0^t e^{k_n^2 a t} C_n(t) dt)$

$$\iff u_n(t) = e^{-k_n^2 a t} (\frac{2}{L} (\frac{1.55}{k_n^2} (-1)^{n+1} - \frac{1.5}{k_n}) + \int_0^t e^{k_n^2 a t} (\frac{4}{(2n+1)\pi}) dt)$$

$$\iff u_n(t) = \frac{e^{-\frac{(2n+1)\pi}{2L} a t}}{(2n+1)\pi} (\frac{12.4L(-1)^{n+1}}{(2n+1)\pi} - 6 - \frac{16L^2}{((2n+1)\pi)^2 a}) + \frac{16L^2}{((2n+1)\pi)^3 a}$$

Donc  $V(x, t) = \sum_{n=1}^{\infty} u_n(t) X_n(x)$

$$V(x, t) = \sum_{n=1}^{\infty} (\frac{e^{-\frac{(2n+1)\pi}{2L} a t}}{(2n+1)\pi} (\frac{12.4L(-1)^{n+1}}{(2n+1)\pi} - 6 - \frac{16L^2}{((2n+1)\pi)^2 a}) + \frac{16L^2}{((2n+1)\pi)^3 a}) \sin(\frac{(2n+1)\pi}{2L} x)$$

Enfin, on a  $U(x, t) = V(x, t) + \epsilon(x, t)$

$$\iff U(x, t) = \sum_{n=1}^{\infty} \left( \frac{e^{-\left(\frac{(2n+1)\pi}{2L}\right)^2 at}}{(2n+1)\pi} \left( \frac{12.4L(-1)^{n+1}}{(2n+1)\pi} - 6 - \frac{16L^2}{((2n+1)\pi)^2 a} \right) + \frac{16L^2}{((2n+1)\pi)^3 a} \right) \sin\left(\frac{(2n+1)\pi}{2L} x\right) + 1.55x + 27.1$$

Ainsi en prenant L=15.4 et a=1 pour simplifier, cela nous donne pour n allant de 1 à 10 :

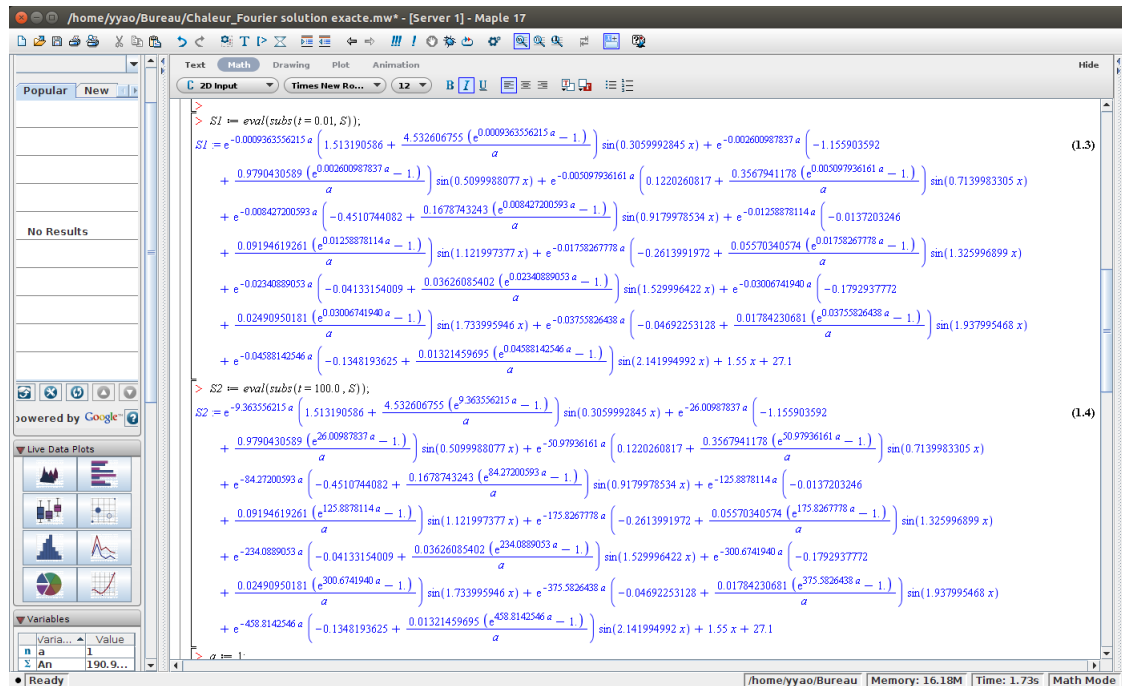


FIGURE 2.2 – Solution exacte sur maple

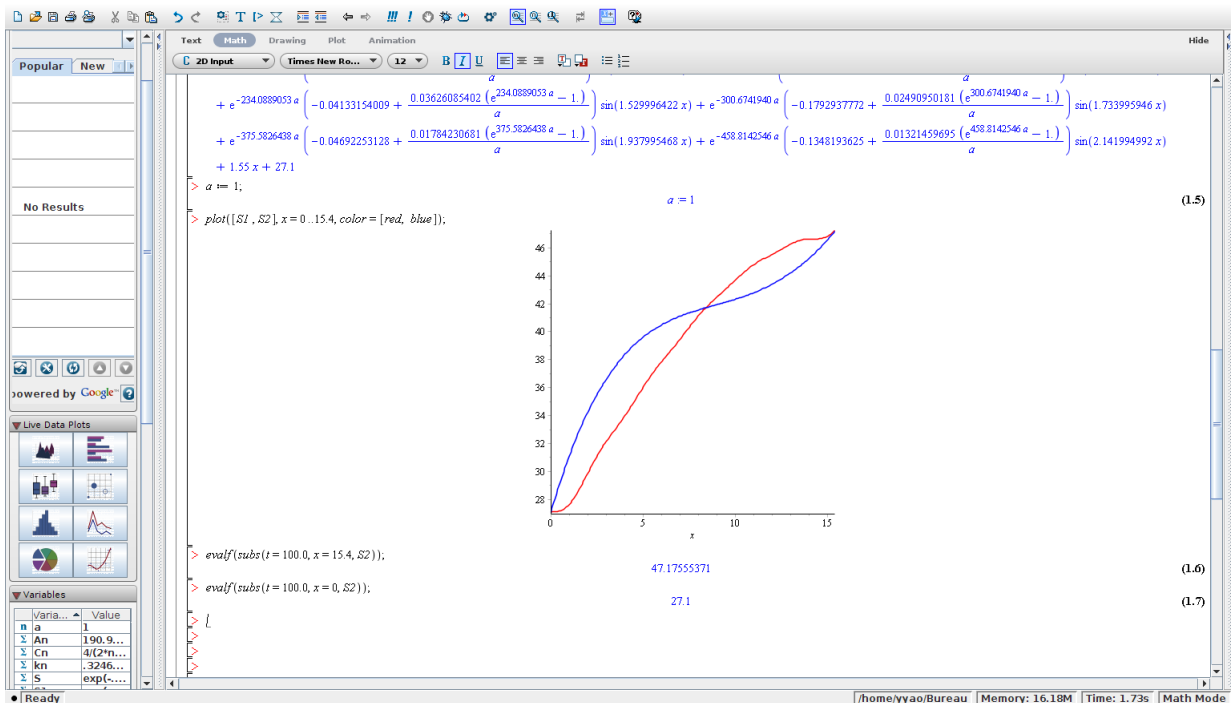


FIGURE 2.3 – Courbe de la solution (exacte) sur maple

# Chapitre 3

## Modélisations

### 3.1 Travail sur le coefficient de diffusion

Lors de ce projet, nous avons travaillé en grande partie sur le coefficient de diffusion. Il nous est apparu intéressant de rajouter ce coefficient dans les programmes. En effet, les années précédentes, pour faire leurs calculs ou leurs expériences, les groupes avaient admis que le coefficient de diffusion était égal à 1. Cependant, dans les conditions de l'expérience, il n'était pas correct d'admettre cette hypothèse puisque la barre utilisée était en métal : en cuivre ou en aluminium, et pour ces matériaux le coefficient de diffusion n'est pas égal à 1.

$$a_{Cu} = 0.000113057m^2/s \text{ et } a_{Al} = 0.00009827m^2/s$$

Le coefficient de diffusion est donné par la formule suivante :  $a = \frac{K}{\rho c}$

avec :

$a$  : Coefficient de diffusion [ $m^2.s^{-1}$ ]

$K$  : Conductivité thermique [ $W.m^{-1}.K^{-1}$ ]

$c$  : Capacité thermique massique [ $J.kg^{-1}.K^{-1}$ ]

$\rho$  : Masse volumique [ $kg.m^{-3}$ ]

Nous avons travaillé sur trois programmes des années précédentes. L'un, exploitant la méthode de résolution explicite dans le cas unidimensionnel (*chaleur\_explicite1D2016.pas*), le second, utilisant les résultats expérimentaux des groupes précédents, toujours pour la méthode explicite (*methode\_explicite2016.pas*) et le dernier combinant les méthodes explicite et implicite dans le cas théorique (*EquationDeLaChaleurFinal2016.pas*).

Nous avons pour cela divisé le travail en trois, puisque chaque programme utilise une méthode de résolution mathématique différente.

**Nous avons commencé par le programme *Chaleur\_explicite1D2016.pas* :**

Dans ce programme, il a fallu modifier le terme source  $f$  :

$$f(x, y) = \frac{x(1-x)^2(1-3xy)}{1+4y^2}$$

Pour se faire, nous sommes repartis de cette équation de base et nous avons appliqué la formule de l'équation aux dérivées partielles en tenant compte du coefficient de diffusion.

$$\frac{\partial u}{\partial t} - \frac{a \partial^2 u}{\partial x^2} = f$$

Nous avons ensuite, effectué les calculs de dérivées en laissant le coefficient de diffusion apparent. Ainsi nous avons obtenu un nouveau terme source  $f$  explicité ci-dessous :

$$f = \frac{(x(1-x)^2(3x(4y^2-1)-8y) - 2a(3x-2-3y(1+x(6x-6))))}{(1+4y^2)}$$

**Maintenant, observons le cas du programme combinant la méthode explicite et implicite :**

Pour la partie de la solution explicite, nous avons rajouté le coefficient de diffusion dans le terme source  $f$  de la même façon que dans le programme précédent.

Pour la partie de la solution implicite, il a été plus compliqué de modifier le coefficient de diffusion. En effet, il a déjà fallu identifier le terme qui le contenait. Pour cela, nous nous sommes aidés du rapport de 2015. Nous nous sommes rendus compte que pour cette méthode, l'étude mathématique conduisait à une matrice qui contenait le coefficient de diffusion  $a$  :

$$\begin{pmatrix} \tau f_1^{p+1} + T_1^p \\ \tau f_2^{p+1} + T_2^p \\ \vdots \\ \vdots \\ \tau f_{N-1}^{p+1} + T_{N-1}^p \end{pmatrix} = \begin{pmatrix} 1 + \frac{2a\tau}{h^2} & -\frac{a\tau}{h^2} & 0 & \dots & \dots & 0 \\ -\frac{a\tau}{h^2} & 1 + \frac{2a\tau}{h^2} & -\frac{a\tau}{h^2} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \dots & \vdots \\ 0 & \dots & \dots & 0 & 1 + \frac{a\tau}{h^2} & -\frac{a\tau}{h^2} \end{pmatrix} \begin{pmatrix} T_1^{p+1} \\ T_2^{p+1} \\ \vdots \\ \vdots \\ T_{N-1}^{p+1} \end{pmatrix}$$

**Pour terminer, nous avons modifié le programme *explicite2016.pas***

Il a donc fallu modifier le coefficient de stabilité  $r$ . Ce dernier, pour obtenir des résultats réalistes, doit être supérieur à 0.5 comme expliqué précédemment dans le rapport. Il s'agit donc de faire apparaître notre coefficient de diffusion dans la formule suivante :

$$r = \frac{\tau a}{h^2} > 0.5$$

Implémentation :

Nous avons donc modifié nos trois programmes. Une fois cela effectué, nous nous sommes assuré que nos résultats restaient correctes. Nous avons donc imposé un coefficient de diffusion égale à 1 et testé tous les nouveaux programmes avec les anciens. Nous allons dans le prochain chapitre, vous expliquer ce que nous avons pu trouver.

## 3.2 Réécriture de programme :

Dans le cadre de notre projet physique sur la conduction thermique, l'implémentation de programmes informatiques, ayant pour objectif de calculer la température en fonction du temps et de l'espace dans une barre, supposée en dimension une, est essentielle afin de parvenir à une modélisation du phénomène physique qu'est la conduction de l'énergie thermique.

Au début du semestre, nous avons pu nous appuyer sur les expérimentations et trois programmes en langage Pascal basés sur les méthodes de résolutions implicite et explicite



déjà réalisées les années précédentes.

Toutefois, le Pascal est un langage de programmation créé dans un but pédagogique. C'est pourquoi il est basique et strict. Ces caractéristiques ne sont pas les mieux adaptées à notre projet physique. Nous nous sommes donc permis, avec l'accord de notre enseignant, d'adapter les programmes existants en Pascal dans un langage de programmation, le Python, qui permet une visualisation plus efficace des données obtenues.

Pour faciliter la compréhension du code Python, la décision de rester au plus près du code Pascal fut prise. Mais plusieurs versions du code Pascal ont coexisté, l'une datant de l'année précédente et l'autre ayant été adaptée par notre groupe afin de l'adapter aux conditions réelles, en vue d'effectuer une comparaison avec les valeurs obtenues par expérimentation. Il a donc fallu changer de code source Pascal pendant la programmation en Python.

L'atout du Python est qu'il permet l'affichage de graphiques efficaces et esthétiques grâce au module *matplotlib.pyplot*. Nous avons choisis de représenter nos résultats sur le graphique par la température en fonction du temps. Les différents points de l'espace sont représentés par une couleur et un type de ligne différents.

On utilise la fonction établie les années précédentes :

$$f = \frac{(x(1-x)^2(3x(4y^2-1) - 8y)) - 2a}{(1+4y^2)} - 2a \frac{(3x-2-3y(1+x(6x-6)))}{(1+4y^2)}$$

On répartit uniformément, 10 divisions sur un temps de 0.5 et 4 points sur 1 pour l'espace. Il y a donc en tout 40 valeurs à calculer.

L'objectif du graphique est donc d'afficher l'évolution de la température en 10 temps en 4 points d'une barre de longueur 1 avec un coefficient de diffusion de 1 également, grâce à la méthode de calcul explicite.

### Résultats obtenus par Jean François

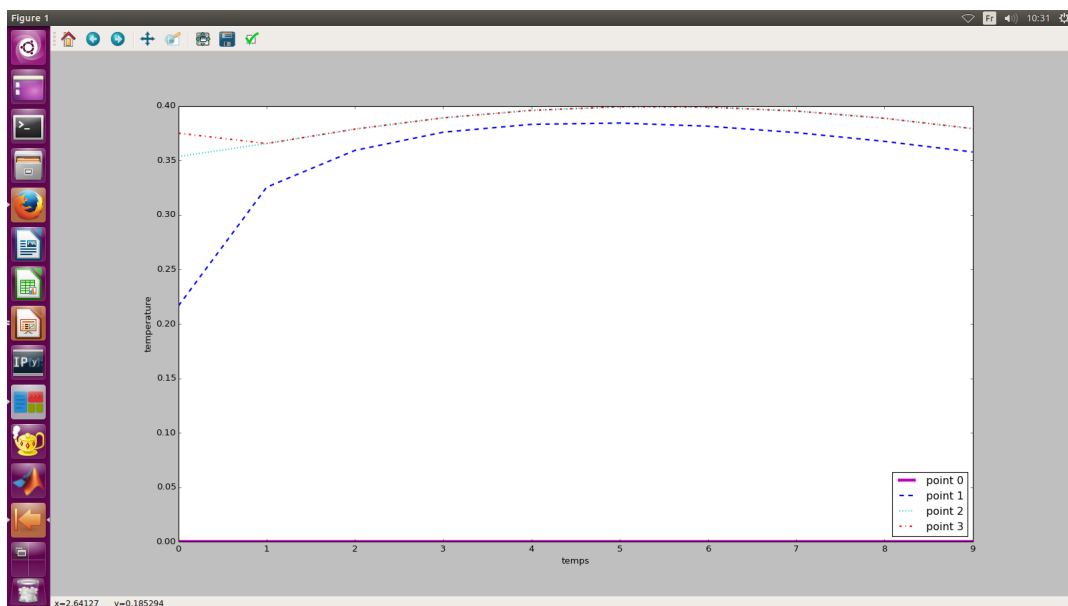


FIGURE 3.1 – Exploitation programme *Chaleurexplicit1D2016.pas* avec  $a = 1$

# Chapitre 4

## Comparaison des différentes méthodes étudiées

Une fois que nous avons ajouté physiquement le coefficient de diffusion dans tous les programmes pascal. Nous avons voulu vérifier que cet ajout n'altérerait pas les résultats des programmes en imposant un coefficient égal à 1. Nous obtenons les mêmes valeurs que les années précédentes, ce qui est donc une réussite.

Ce premier graphique est le fruit du résultat du programme *Chaleurexplicite1D2016.pas* avec un coefficient de diffusion égal à 1.

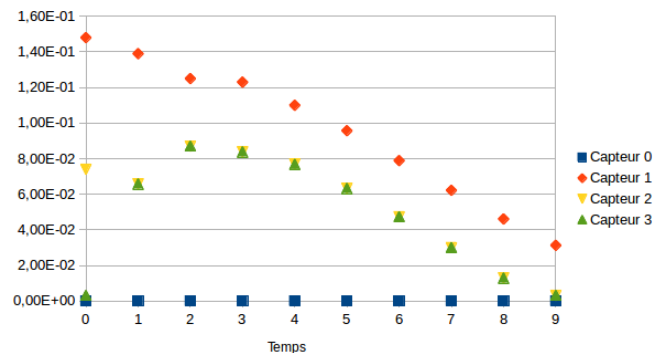


FIGURE 4.1 – Exploitation programme *Chaleurexplicite1D2016.pas* avec  $a = 1$

Nous pensions, dans un premier temps, vous présenter les graphiques obtenus en changeant les coefficients de diffusion au sein du même programme. Cependant, les coefficients de diffusion étant très petits les résultats changés ne sont pas visibles graphiquement. Nous avons donc décidé de vous présenter les résultats des programmes :

```

Terminal
u( 0, 0) = 0.000000000000E+000
u( 1, 0) = 1.48148148148148E-001
u( 2, 0) = 7.40740740740741E-002
u( 3, 0) = 0.000000000000E+000
u( 0, 1) = 0.000000000000E+000
u( 1, 1) = 1.39917695473251E-001
u( 2, 1) = 6.58436213991770E-002
u( 3, 1) = 6.58436213991770E-002
u( 0, 2) = 0.000000000000E+000
u( 1, 2) = 1.25079256664292E-001
u( 2, 2) = 8.70666588328033E-002
u( 3, 2) = 8.70666588328033E-002
u( 0, 3) = 0.000000000000E+000
u( 1, 3) = 1.23902418228397E-001
u( 2, 3) = 8.38814929273251E-002
u( 3, 3) = 8.38814929273251E-002
u( 0, 4) = 0.000000000000E+000
u( 1, 4) = 1.10459264982181E-001
u( 2, 4) = 7.68549185408240E-002
u( 3, 4) = 7.68549185408240E-002
u( 0, 5) = 0.000000000000E+000
u( 1, 5) = 9.57838910555823E-002
u( 2, 5) = 6.33315169572371E-002
u( 3, 5) = 6.33315169572371E-002
u( 0, 6) = 0.000000000000E+000
u( 1, 6) = 7.89839025393757E-002
u( 2, 6) = 4.73101663298937E-002
u( 3, 6) = 4.73101663298937E-002
u( 0, 7) = 0.000000000000E+000
u( 1, 7) = 6.22261986591393E-002
u( 2, 7) = 3.08547705731481E-002
u( 3, 7) = 3.08547705731481E-002
u( 0, 8) = 0.000000000000E+000
u( 1, 8) = 4.61319217757219E-002
u( 2, 8) = 1.29846660757058E-002
u( 3, 8) = 1.29846660757058E-002
u( 0, 9) = 0.000000000000E+000
u( 1, 9) = 3.13040651028549E-002
u( 2, 9) = -3.13294666088941E-003
u( 3, 9) = -3.13294666088941E-003
la difference maximum entre la solution approchee et la solution exacte est 8.70666588328033E-002en 3 2
-1.48148148148148E-001
-----
(program exited with code: 0)
Press return to continue

```

FIGURE 4.2 – Exploitation programme *Chaleurexplicit1D2016.pas* avec  $a = 1$

```

Terminal
u( 0, 0) = 0.000000000000E+000
u( 1, 0) = 1.48148148148148E-001
u( 2, 0) = 7.40740740740741E-002
u( 3, 0) = 0.000000000000E+000
u( 0, 1) = 0.000000000000E+000
u( 1, 1) = 1.39917695473251E-001
u( 2, 1) = 6.58436213991770E-002
u( 3, 1) = 6.58436213991770E-002
u( 0, 2) = 0.000000000000E+000
u( 1, 2) = 1.28416276891382E-001
u( 2, 2) = 5.61302231936886E-002
u( 3, 2) = 5.61302231936886E-002
u( 0, 3) = 0.000000000000E+000
u( 1, 3) = 1.14667106076717E-001
u( 2, 3) = 4.57058599124203E-002
u( 3, 3) = 4.57058599124203E-002
u( 0, 4) = 0.000000000000E+000
u( 1, 4) = 9.98514570899816E-002
u( 2, 4) = 3.53372401195469E-002
u( 3, 4) = 3.53372401195469E-002
u( 0, 5) = 0.000000000000E+000
u( 1, 5) = 8.50419567230585E-002
u( 2, 5) = 2.56313657368336E-002
u( 3, 5) = 2.56313657368336E-002
u( 0, 6) = 0.000000000000E+000
u( 1, 6) = 7.10382743814688E-002
u( 2, 6) = 1.69693351714038E-002
u( 3, 6) = 1.69693351714038E-002
u( 0, 7) = 0.000000000000E+000
u( 1, 7) = 5.83262309378496E-002
u( 2, 7) = 9.51823388467316E-003
u( 3, 7) = 9.51823388467316E-003
u( 0, 8) = 0.000000000000E+000
u( 1, 8) = 4.7121968659399E-002
u( 2, 8) = 3.28524471748214E-003
u( 3, 8) = 3.28524471748214E-003
u( 0, 9) = 0.000000000000E+000
u( 1, 9) = 3.74490614655813E-002
u( 2, 9) = -1.82034595169040E-003
u( 3, 9) = -1.82034595169040E-003
la difference maximum entre la solution approchee et la solution exacte est 6.58436213991770E-002en 3 1
-1.48148148148148E-001
-----
(program exited with code: 0)
Press return to continue

```

FIGURE 4.3 – Exploitation programme *Chaleurexplicit1D2016.pas* avec  $a_{AI}=0.00009827m^2/s$

```

Terminal
u( 0, 0)= 0.00000000000000E+000
u( 1, 0)= 1.48148148148148E-001
u( 2, 0)= 7.40740740740741E-002
u( 3, 0)= 0.00000000000000E+000
u( 0, 1)= 0.00000000000000E+000
u( 1, 1)= 1.39917695473251E-001
u( 2, 1)= 6.58436213991770E-002
u( 3, 1)= 6.58436213991770E-002
u( 0, 2)= 0.00000000000000E+000
u( 1, 2)= 1.28416227542015E-001
u( 2, 2)= 5.6130606957211E-002
u( 3, 2)= 5.6130606957211E-002
u( 0, 3)= 0.00000000000000E+000
u( 1, 3)= 1.14666964583756E-001
u( 2, 3)= 4.57066778680334E-002
u( 3, 3)= 4.57066778680334E-002
u( 0, 4)= 0.00000000000000E+000
u( 1, 4)= 9.98511910125332E-002
u( 2, 4)= 3.53383214358973E-002
u( 3, 4)= 3.53383214358973E-002
u( 0, 5)= 0.00000000000000E+000
u( 1, 5)= 8.50415418428735E-002
u( 2, 5)= 2.56326190762457E-002
u( 3, 5)= 2.56326190762457E-002
u( 0, 6)= 0.00000000000000E+000
u( 1, 6)= 7.10376983591268E-002
u( 2, 6)= 1.69706789187185E-002
u( 3, 6)= 1.69706789187185E-002
u( 0, 7)= 0.00000000000000E+000
u( 1, 7)= 5.83254883840996E-002
u( 2, 7)= 9.51959812832286E-003
u( 3, 7)= 9.51959812832286E-003
u( 0, 8)= 0.00000000000000E+000
u( 1, 8)= 4.71210597701910E-002
u( 2, 8)= 3.28657159242468E-003
u( 3, 8)= 3.28657159242468E-003
u( 0, 9)= 0.00000000000000E+000
u( 1, 9)= 3.74487901510525E-002
u( 2, 9)= -1.81910301355499E-003
u( 3, 9)= -1.81910301355499E-003
la difference maximum entre la solution approchee et la solution exacte est 6.58436213991770E-002en 3 1
-1.48148148148148E-001
-----
(program exited with code: 0)
Press return to continue

```

FIGURE 4.4 – Exploitation programme *Chaleurexplicite1D2016.pas* avec  $a_{Cu}=0.000113057m^2/s$

Nous pouvons donc affirmer que nos résultats sont assez probants. En effet, nos résultats avec un coefficient de diffusion égal à 1 sont identiques aux programmes des années précédentes. De plus, avec les coefficients de diffusion des barres métalliques nous obtenons aussi des résultats probants.

# Conclusion

Pour conclure, nous avons apprécié travailler sur un sujet de recherche qui fut une première pour nous tous, nous pensons avoir atteint nos objectifs décrits en début de rapport.

Notre sujet sur la conduction thermique n'était pas simple à traiter. En effet, ce sujet nous a demandé un grand travail de compréhension sur les rapports des années précédentes ainsi que de recherche, pour assimiler correctement le sujet.

Cette année nous avons pour objectif d'étudier différentes méthodes de résolution de l'équation de la chaleur. En effet, nous avons amélioré les résolutions explicites, implicites prioritairement et nous avons aussi commencé l'étude de la solution semi-implicite.

Ce sujet nous a permis de travailler en groupe et d'effectuer un travail de recherche qui n'est pas habituel dans notre enseignement à l'INSA. Nous avons dû nous répartir les tâches et apprendre à communiquer entre nous. En effet, avant ce projet, aucun de nous ne se connaissait. De plus, nous évoluons dans quatre spécialités différentes à savoir GM, ASI, EP et Méca. Il a donc fallu savoir utiliser au mieux les compétences de chacun pour réaliser ce projet. Certains étaient plus compétents pour la partie mathématique et d'autres pour la partie physique. Nous pouvons donc dire que nous avons été assez complémentaires.

Pour les années à venir, nous pouvons conseiller aux futurs groupes de se pencher sur la méthode de Crank-Nicolson que nous n'avons pas eu le temps d'aborder et de refaire une expérience pour valider les résultats trouvés cette année.

# Bibliographie

Voici les sites qui nous ont aidés à la rédaction de notre rapport :

- Rapports P6 2013-4-5

- <http://www.formules-physique.com>. Site valide au 10/06/2016.

- <http://www.utc.fr/houde/TF06/CoursTransfertdechaleur.pdf>. Site valide au 10/06/2016.

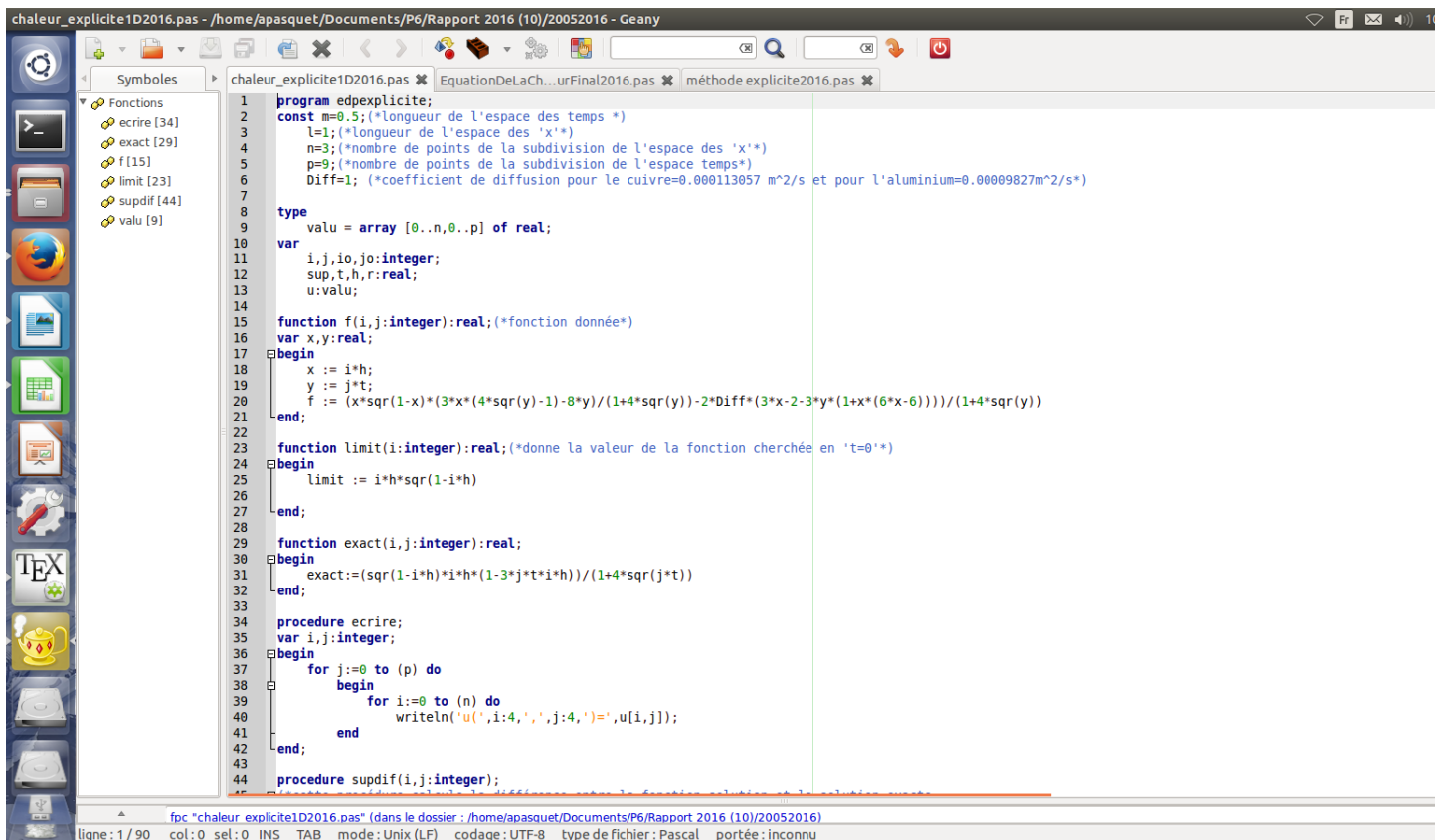
- <http://mathias.bavay.free.fr/these/html/node308.html>. Site valide au 10/06/2016.

- <http://www.sciences.univ-nantes.fr/sites/claudesaintblanquet/conducti/11intro/11intro.htm>.  
Site valide au 11/06/2016.

- <http://www.conductivitethermique.fr>. Site valide au 11/06/2016.

# Annexes

Voici l'ensemble des programmes que nous avons modifié :



```
1 program edexplicit;
2 const m=0.5;(*longueur de l'espace des temps *)
3       l=1;(*longueur de l'espace des 'x'*)
4       n=3;(*nombre de points de la subdivision de l'espace des 'x'*)
5       p=9;(*nombre de points de la subdivision de l'espace temps*)
6       Diff=1; (*coefficient de diffusion pour le cuivre=0.000113057 m^2/s et pour l'aluminium=0.00009827m^2/s*)
7
8 type
9   valu = array [0..n,0..p] of real;
10
11 var
12   i,j,io,jo:integer;
13   sup,t,h,r:real;
14   u:valu;
15
16 function f(i,j:integer):real;(*fonction donnée*)
17 var x,y:real;
18 begin
19   x := i*h;
20   y := j*t;
21   f := (x*sqr(1-x)*(3*x*(4*sqr(y)-1)-8*y)/(1+4*sqr(y))-2*Diff*(3*x-2-3*y*(1+x*(6*x-6)))/(1+4*sqr(y)))
22 end;
23
24 function limit(i:integer):real;(*donne la valeur de la fonction cherchée en 't=0'*)
25 begin
26   limit := i*h*sqr(1-i*h)
27 end;
28
29 function exact(i,j:integer):real;
30 begin
31   exact:=(sqr(1-i*h)*i*h*(1-3*j*t*i*h))/(1+4*sqr(j*t))
32 end;
33
34 procedure ecrire;
35 var i,j:integer;
36 begin
37   for j:=0 to (p) do
38     begin
39       for i:=0 to (n) do
40         writeln('u(',i:4,',',j:4,')=',u[i,j]);
41       end
42     end;
43 end;
44
45 procedure supdif(i,j:integer);
```

FIGURE 4.5 – *Chaleurexplicit1D2016.pas*

```

41      end
42  end;
43
44  procedure supdif(i,j:integer);
45  /*cette procédure calcule la différence entre la fonction solution et la solution exacte,
46  au point (ih,jt), et enregistre le maximum de cette différence sur la grille*)
47  var a:real;
48  begin
49    a:=abs(u[i,j]-exact(i,j));
50    if a >= sup then
51      begin
52        sup := a;
53        io :=i;
54        jo:=j
55      end
56    end;
57
58  (*début du corps de programme*)
59
60  begin
61    io:=0;
62    jo:=0;
63    sup:=0;
64    t:=m/p; {pas de temps}
65    h:=l/n; {pas d'espace}
66    r:=t/sqr(h); {t représente le pas de temps: t=1/18}
67    if r>0.5 then writeln('le probleme ainsi discretise n'est pas stable,', 'changer les constantes pour que la condition r<=0.5 soit verifiee')
68    else
69      begin
70        for i:=0 to (n) do
71          u[i,0]:=limit(i); {j=temps}
72          for j:=0 to (p-1) do
73            begin
74              u[0,j+1]:=0;
75              for i:=1 to (n-1) do {i=espace}
76                begin
77                  u[i,j+1]:=u[i,j]+*(u[i-1,j]-2*u[i,j]+u[i+1,j])+t*f(i,j);{r=coef de stabilité; j=n}
78                  {writeln('u[1,0]=' ,u[i,j]);}
79                  writeln('u[1,1]=' ,u[i,j+1]); read(u[1,1]);}
80                supdif(i,j+1)
81              end;
82            u[n,j+1]:=u[n-1,j+1];{flux constant pour tout t}
83            supdif(n,j+1)
84          end;
85        end;
86      end;
87      writeln('la difference maximum entre la solution approchee', 'et la solution exacte est',sup,'en',io:4, ', ',jo:4);
88      write(f(2,0));
89    end
90  end.

```

FIGURE 4.6 – *Chaleurexplicit1D2016.pas*

```

47  var a:real;
48  begin
49    a:=abs(u[i,j]-exact(i,j));
50    if a >= sup then
51      begin
52        sup := a;
53        io :=i;
54        jo:=j
55      end
56    end;
57
58  (*début du corps de programme*)
59
60  begin
61    io:=0;
62    jo:=0;
63    sup:=0;
64    t:=m/p; {pas de temps}
65    h:=l/n; {pas d'espace}
66    r:=t/sqr(h); {t représente le pas de temps: t=1/18}
67    if r>0.5 then writeln('le probleme ainsi discretise n'est pas stable,', 'changer les constantes pour que la condition r<=0.5 soit verifiee')
68    else
69      begin
70        for i:=0 to (n) do
71          u[i,0]:=limit(i); {j=temps}
72          for j:=0 to (p-1) do
73            begin
74              u[0,j+1]:=0;
75              for i:=1 to (n-1) do {i=espace}
76                begin
77                  u[i,j+1]:=u[i,j]+*(u[i-1,j]-2*u[i,j]+u[i+1,j])+t*f(i,j);{r=coef de stabilité; j=n}
78                  {writeln('u[1,0]=' ,u[i,j]);}
79                  writeln('u[1,1]=' ,u[i,j+1]); read(u[1,1]);}
80                supdif(i,j+1)
81              end;
82            u[n,j+1]:=u[n-1,j+1];{flux constant pour tout t}
83            supdif(n,j+1)
84          end;
85        end;
86      end;
87      writeln('la difference maximum entre la solution approchee', 'et la solution exacte est',sup,'en',io:4, ', ',jo:4);
88      write(f(2,0));
89    end
90  end.

```

FIGURE 4.7 – *Chaleurexplicit1D2016.pas*



```

*EquationDeLaCh...urFinal2016.pas - /home/apasquet/Documents/P6/Rapport 2016 (10)/20052016 - Geany
chaleur_explicite1D2016.pas * EquationDeLaCh...urFinal2016.pas * méthode explicite2016.pas *
1 program EquationDeLaChaleur;
2
3 uses crt;
4
5 const TabMax=50;
6     Nt=9; (*nombre de points de la subdivision de l'espace temps*)
7     N=3; (*nombre de points de la subdivision de l'espace des 'x'*)
8     TMax=0.5; (*longueur de l'espace des temps*)
9     l=1; (*longueur de l'espace des x*)
10    Dt=TMax/Nt; (*Deltat = pas selon l'espace temps*)
11    h=l/N; (*Pas selon l'espace des x*)
12    Diff=1; (*coefficient de diffusion pour le cuivre=0.00113057 m^2/s et pour l'aluminium=0.00009827m^2/s*)
13
14 Type Tabulation=Array[0..TabMax] of real;
15
16 function f(i,j:Real):real;(*fonction donnée qui est utilisée pour calculer Up[i] dans la méthode explicite*)
17 var x,y:real;
18 begin
19     x := i*h;
20     y := j*Dt;
21     f := (x*sqr(1-x)*(3*x*(4*sqr(y)-1)-8*y)/(1+4*sqr(y))-Diff*2*(3*x-2-3*y*(1+x*(6*x-6)))/(1+4*sqr(y)));
22 end;
23
24 function exact(i,j:real):real; (*fonction utilisée dans la procedure supdif pour calculer la solution exacte au point (i,j)*)
25 var x,y: real;
26
27 begin
28     x := i*h;
29     y := j*Dt;
30     exact:=(sqr(1-x)*x*(1-3*y*x))/(1+4*sqr(y));
31 end;
32
33 procedure supdif(i:integer;j:real;U:Tabulation;var sup,jo:real; var io:integer);
34 (*cette procédure calcule la différence entre la fonction solution et la solution exacte,
35 au point (i,j), et enregistre le maximum de cette différence sur la grille*)
36 var a:real;
37 begin
38     a:=abs(U[i]-exact(i,j));
39     if a >= sup then
40         begin
41             sup := a;
42             io :=i;
43             jo:=j;
44         end
45 end;

```

FIGURE 4.8 – Equationdelachaleurfinale2016.pas

```

*EquationDeLaCh...urFinal2016.pas - /home/apasquet/Documents/P6/Rapport 2016 (10)/20052016 - Geany
chaleur_explicite1D2016.pas * EquationDeLaCh...urFinal2016.pas * méthode explicite2016.pas *
45 end;
46
47 procedure Recurrence1(U:tabulation;T,Tp:Real; var Up:tabulation; var sup,jo:real; var io:integer); (*Methode explicite*)
48 var i:integer;
49     L:Real;
50
51 begin
52     L:=Dt*SQR(N); (*r=t/h^2 coefficient de stabilité; L=r*a où a est le coefficient de diffusion, a=1: r=L. Dt=pas de temps*)
53     for i:=1 to N-1 do
54         begin
55             Up[i]:=U[i]+L*(U[i-1]-2*U[i]+U[i+1])+(Tp-T)*f(i,(Nt*T)/TMax); (*calcul de récurrence*)
56             supdif(i,(Nt*T)/TMax,U,sup,jo,io); (*calcul de la différence*)
57         end;
58     Up[0]:=0; (*condition aux limites en 0: température nulle*)
59     Up[N]:=Up[N-1]; (*condition aux limites au bout de la barre: gradient de température nul = pas de flux*)
60     supdif(N,(Nt*T)/TMax,U,sup,jo,io);
61 end;
62
63 procedure TriDiag(Dim:Integer; T:real;Sous,Diag,Sur,Donnee:Tabulation; var Result:Tabulation; var sup,jo:real; var io:integer);
64 (*algorithme*)
65 for i:=2 to Dim do Diag[i]:=Diag[i]-Sous[i]/Diag[i-1]*Sur[i-1];
66 for i:=2 to Dim do Donnee[i]:=Donnee[i]-Sous[i]/Diag[i-1]*Donnee[i-1];
67 Result[Dim]:=Donnee[Dim]/Diag[Dim];
68 for i:=Dim-1 downto 1 do
69     begin
70         Result[i]:=(Donnee[i]-Sur[i]*Result[i+1])/Diag[i];
71         supdif(i,(Nt*T)/TMax,Donnee,sup,jo,io); (*calcul de la différence*)
72     end;
73 end;
74
75 procedure Recurrence2(U:tabulation;T,Tp:Real; var Up:tabulation; var sup,jo:real; var io:integer); (*méthode implicite*)
76 var i: integer;
77     Sous,D,Sur:Tabulation;
78     L:REAL;
79
80 begin
81     L:=Dt*sqr(N)*Diff;
82     for i:=1 to N-1 do (*On initialise les 3 diagonales de la matrice*)
83         begin
84             Sous[i]:=-L;
85             D[i]:=1+2*L;
86             Sur[i]:=-L;
87         end;
88     D[N]:=1+L;
89     TriDiag(N,T,Sous,D,Sur,U,Up,sup,jo,io); (*on détermine Up grâce à l'algorithme*)
90 end;

```

FIGURE 4.9 – Equationdelachaleurfinale2016.pas

```

EquationDeLaCh...urFinal2016.pas - /home/apasquet/Documents/P6/Rapport 2016 (10)/20052016 - Geany
chaleur_expliciteID2016.pas EquationDeLaCh...urFinal2016.pas méthode explicite2016.pas
Fonctions
  Init [99]
  Recurrence1
  Recurrence2
  Tabulation [1]
  TriDiag [62]
  U0 [94]
  exact [24]
  f [16]
  supdif [33]
86   end;
87   D[N]:=1+l;
88   TriDiag(N,T,Sous,D,Sur,U,Up,sup,jo,io); (*on détermine Up grâce à l'algorithme*)
89   Up[0]:=0; (*condition aux limites en 0: température nulle*)
90   Up[N]:=Up[N-1]; (*condition aux limites au bout de la barre: gradient de température nul = pas de flux*)
91   supdif(N,(Nt*T)/TMax,U,sup,jo,io);
92 -end;
93
94 function U0(x:real):real; (*fonction utilisée dans la procedure Init*)
95 begin
96   U0:=x*sqr(1-x);
97 -end;
98
99 procedure Init(var U:Tabulation;N:Integer); (*La procedure Init calcule les températures initiales dans la barre, i.e. à t=0*)
100 var i:integer;
101 begin
102   for i:=0 to N do U[i]:=U0(I/N);
103 -end;
104
105 (*Début du programme principal*)
106 var U,Up:tabulation;
107     i,io:integer;
108     jo,sup,T,Tp,Lambda:Real;
109     Car:Char;
110 begin
111   clrscr;
112   writeln('EQUATION DE LA CHALEUR SUR [0,1]');
113   writeln('N=',N,' l'intervalle sur x est ',h:7:5);
114   writeln('Resolution de t=0 à t=',TMax:7:4);
115   repeat
116     writeln('Methode (E)xplícite (I)mplicite (F)in');
117     Car:=UpCase(ReadKey);
118     case Car of
119       'E': writeln('Choix de la methode explicite');
120       'I': writeln('Choix de la methode implicite');
121     end;
122     writeln;
123     If Car<<'F' then
124     begin
125       io:=0;
126       jo:=0;
127       sup:=0; (*les 3 variables io, jo et sup seront utilisées pour le calcul de la différence*)
128       Init(U,N); (*on initialise U à l'instant t=0*)
129       Lambda:=Dt/SQR(h); (*Lambda vérifie si la condition de stabilité est respectée. On doit trouver Lambda <= 0.5*)
130

```

FIGURE 4.10 – Equationdelachaleur finale2016.pas

```

EquationDeLaCh...urFinal2016.pas - /home/apasquet/Documents/P6/Rapport 2016 (10)/20052016 - Geany
chaleur_expliciteID2016.pas EquationDeLaCh...urFinal2016.pas méthode explicite2016.pas
111   clrscr;
112   writeln('EQUATION DE LA CHALEUR SUR [0,1]');
113   writeln('N=',N,' l'intervalle sur x est ',h:7:5);
114   writeln('Resolution de t=0 à t=',TMax:7:4);
115   repeat
116     writeln('Methode (E)xplícite (I)mplicite (F)in');
117     Car:=UpCase(ReadKey);
118     case Car of
119       'E': writeln('Choix de la methode explicite');
120       'I': writeln('Choix de la methode implicite');
121     end;
122     writeln;
123     If Car<<'F' then
124     begin
125       io:=0;
126       jo:=0;
127       sup:=0; (*les 3 variables io, jo et sup seront utilisées pour le calcul de la différence*)
128       Init(U,N); (*on initialise U à l'instant t=0*)
129       Lambda:=Dt/SQR(h); (*Lambda vérifie si la condition de stabilité est respectée. On doit trouver Lambda <= 0.5*)
130       writeln(Lambda);
131       if Lambda > 0.5 then
132         writeln('La condition de stabilité n'est pas vérifiée')
133       else
134         begin
135           writeln('Lambda = ',Lambda:7:5,' ',Nt,' pas en t');
136           T:=0;
137           for i:=0 to N do writeln('U[',i,',', (Nt*T)/TMax:1:0,']= ',U[i]);
138           while T<=TMax do
139             begin
140               Tp:=T+Dt; (*On fait varier l'espace temps de Dt*)
141               if Car='E' then Recurrence1(U,T,Tp,Up,sup,jo,io) (*On lance la méthode explicite*)
142               else Recurrence2(U,T,Tp,Up,sup,jo,io); (*On lance la méthode implicite*)
143               U:=Up; (*U contient les données à l'instant t, Up à l'instant t+1. Ici on se décale pour calculer les données à l'instant t+2 en utilisant
144               T:=Tp;
145               if (Nt*T)/TMax < Nt+1 then for i:=0 to N do writeln('U[',i,',', (Nt*T)/TMax:1:0,']= ',U[i]);
146             end;
147             writeln('la difference maximum entre la solution approchée et la solution exacte est ',sup,' en ',io:4,' ',jo:4:0);
148             writeln ('le coefficient de diffusion est ', Diff );
149           end;
150         end;
151     end;
152   until Car='F';
153 -end.
154

```

FIGURE 4.11 – Equationdelachaleur finale2016.pas

```

1  program edexplicitExperimentation;
2  const m=800;(*longueur de l'espace des temps *)
3        l=15.4;(*longueur de l'espace des 'x'*)
4        n=7;(*nombre de points de la subdivision de l'espace des 'x'*)
5        p=800;(*nombre de points de la subdivision de l'espace temps*)
6        Diff=1;(*coefficient de diffusion pour le cuivre=0.000113057 m^2/s et pour l'aluminium=0.0009827m^2/s*)
7
8  type
9    valu = array [0..n,0..p] of real;
10
11  var
12    i,j,io,jo:integer;
13    sup,t,h,r:real;
14    u:valu;
15
16  function limit(i:integer):real;(*donne la valeur de la fonction cherchée en 'x=0' t=0 ??*)
17  begin
18    limit := 25.6; //température de la pièce
19  end;
20
21  function exact(i,j:integer):real;
22  begin
23    exact:=(sqr(1-i*h)*i*h*(1-3*j*t+i*h))/(1+4*sqr(j*t))
24  end;
25
26  procedure écrire;
27  var i,j:integer;
28  begin
29    for j:=0 to (p) do
30    begin
31      for i:=0 to (n) do
32      begin
33        writeln('u(',i:4,',',j:4,')=',u[i,j]);
34      end;
35    end;
36  end;
37
38  procedure ecriturefichier;
39  var i,j : integer;
40      fil : Text;
41  begin
42    Assign(fil,'valeursfinalexplicite.txt');
43    Rewrite(fil);
44    write(fil,'M = [');
45
46  end;
47
48  /usr/bin/ld.bfd: Avertissement: link.res contient des sections de sortie; avez-vous oublié -T?
49  ligne : 22 / 115 col : 5 sel : 0 INS TAB mode : Unix (LF) codage : UTF-8 type de fichier : Pascal portée : inconnu

```

FIGURE 4.12 – *Methodeexplicitite2016.pas*

```

42  Assign(fil,'valeursfinalexplicite.txt');
43  Rewrite(fil);
44  write(fil,'M = [');
45  for j:=0 to (p) do
46  begin
47    for i:=0 to (n) do
48    begin
49      write(fil,u[i,j]:4:2,'');
50      // On evite d'écrire la dernière virgule de chaque ligne
51      if i << n then
52        write(fil,',');
53    end;
54    // On evite d'écrire le dernier ;
55    if j << p then
56    begin
57      write(fil,',');
58      writeln(fil);
59    end;
60  end;
61  write(fil,']');
62  Close (fil);
63  writeln('Ecriture du fichier OK !');
64  end;
65
66  procedure supdif(i,j:integer);
67  (*cette procédure calcule la différence entre la fonction solution et la solution exacte,
68  au point (ih,jt), et enregistre le maximum de cette différence sur la grille*)
69  var a:real;
70  begin
71    a:=abs(u[i,j]-exact(i,j));
72    if a >= sup then
73    begin
74      sup := a;
75      io :=i;
76      jo:=j
77    end
78  end;
79
80  (*début du corps de programme*)
81
82  begin
83    io:=0;
84    jo:=0;
85    sup:=0;
86
87  end;
88
89  /usr/bin/ld.bfd: Avertissement: link.res contient des sections de sortie; avez-vous oublié -T?
90  ligne : 22 / 115 col : 5 sel : 0 INS TAB mode : Unix (LF) codage : UTF-8 type de fichier : Pascal portée : inconnu

```

FIGURE 4.13 – *Methodeexplicitite2016.pas*

```

72   if a >= sup then
73   begin
74     sup := a;
75     io :=1;
76     jo:=j
77   end
78 end;
79
80 (*début du corps de programme*)
81
82 begin
83   io:=0;
84   jo:=0;
85   sup:=0;
86   t:=m/p;
87   h:=l/n;
88   r:=Diff*t/sqr(h);
89   if r>0.5 then writeln('le probleme ainsi discretise n'est pas stable,' changer les constantes pour que la condition r<=0.5 soit verifiee')
90   else
91   begin
92     u[0,0]:=27.1;
93     for i:=1 to (n) do
94       u[i,0]:=limit(i);
95     for j:=0 to(p-1) do
96     begin
97       u[n,j+1]:=50.5;
98       u[0,j+1]:=27.1;
99       u[n-1,j+1]:=u[n,j+1]-1.8;
100      for l:=1 to (n-2) do
101      begin
102        u[i,j+1]:=u[i,j]+r*(u[i-1,j]-2*u[i,j]+u[i+1,j]);
103        supdif(i,j+1)
104      end;
105    end;
106  end;
107  supdif(n,j+1)
108 end;
109 écrire;
110 ecriturefichier;
111 writeln('la difference maximum entre la solution approchee',' et la solution exacte est',sup,'en',io:4,' ',jo:4)
112 end
113
114 end.
115

```

ligne : 22 / 115 col : 5 sel : 0 INS TAB mode : Unix (LF) codage : UTF-8 type de fichier : Pascal portée : inconnu

FIGURE 4.14 – *Methodeexplicitite2016.pas*