RAPPORT ÉCRIT

Projet de méthode numérique en C

Diffusion de la chaleur

NEHMAR Anis
SADDI Ahmed
MYLOT Jason
DUMONT LE BRAZIDEC Solenn

L3 EEA 2019 2020
Unité d'Enseignement LU3EE103

Le 20 décembre 2019

Table des matières

Introduction

Dans le cadre de notre projet de méthodes numériques en C, nous devions modéliser la diffusion de chaleur dans les composants électroniques. Le problème était divisé en deux principaux axes. Premièrement, nous devions le résoudre en monodimensionnel en considérant notre espace comme une ligne. Ensuite, nous l'avons étudié et codé en bidimensionnel.

Pour faciliter le travail en groupe et la clarté du code, nous avons structuré notre code en différentes parties : d'abord un code *initialisation.c* qui contient avec son fichier header *initialisation.h* où nous avons placé toutes les fonctions qui permettent d'initialiser toutes nos structures et de les libérer. Puis, un code *calcul.c* avec un autre fichier header *calcul.h* qui contient la description de nos structures et l'algorithme qui permet de calculer l'équation différentielle et ainsi modéliser la diffusion de chaleur dans les dispositifs électroniques.

Description du problème et méthode de résolution

Dans cette partie, nous allons décrire le problème physique de la diffusion de chaleur et la résolution physique d'après les équations différentielles.

La plupart des équations ici, données dans l'énoncé, ont été résolues à partir d'une méthode de résolution d'équation vue en cours [1].

Ici, on considère Ω, l'espace contenant notre dispositif électronique.

<u>Equation principale :</u>

$\partial T \; \partial t \; (r, t) - \alpha(r)\Delta T(r, t) = f(r, t) \qquad r \in \Omega, t > t0$

$T(r, t) = fc(r) \qquad r \in \partial\Omega, t > t0$

$T(r, t0) = T0(r) \qquad r \in \Omega$

Pour commencer, le système que nous étudions est à une dimension, comme un segment allant de a à b, f(x,t) correspond à notre fonction de diffusion, qui dépend donc de t, le temps, et x, la coordonnées spatiales qui traverse le segment de a à b.

<u>Equation monodimensionnelle :</u>

$\frac{\partial T}{\partial t}(x,t) - \alpha \frac{\partial^2 T}{\partial^2 x}(x,t) = f(x,t) \qquad x \in]a,b[, t > t_0$

$T(a,t) = T_{\text{a}} \; T(b,t) = T_b \qquad t > t_0$

$T(x, t_0) = T_0(x) \qquad x \in]a,b[$

Résolution de l'équation monodimensionnelle :

On a $\frac{\partial T}{\partial t}(x,t) = \frac{T(x,t\ dt)}{dt}$ (1)

Et $\frac{\partial^2 T}{\partial x^2}(x,t) = \frac{T(x-dx,t)-2*T(x,t)+T(x-dx,t)}{dx^2}$ (2)

On obtient : $(1) - \alpha\ (2) = 0$

Pour simplifier la lecture on prend: $n = x\ et\ j = t$

On a donc : $\frac{T_{j+1}^{n}}{dt} - \alpha \frac{T_j^{n-1} - 2\,T_j^{n} + T_j^{n+1}}{dx^2} = 0$ <=> $dt\ \alpha \frac{T_j^{n-1} - 2\,T_j^{n} + T_j^{n+1}}{dx^2} = T_{j+1}^{n}$

Donc, la température au point n, au temps j+1, dépend des températures au même point n, au point connexe (n-1 et n+1) au temps j.

Puis, nous avons résolus les équations pour pouvoir calculer la diffusion de chaleur dans un espace en deux dimensions avec deux coordonnées spatiales x et y.

Équation bidimensionnelle :

$\frac{\partial T}{\partial t}(x,y,t) - \alpha\ (\frac{\partial^2 T}{\partial^2 x}(x,y,t) + \frac{\partial^2 T}{\partial y^2}(x,y,t)) = f(x,y,t) \qquad x \in]0,a[, y \in]0,b[, t > t_0$

$T(0,y,t) = T_{x0}, T(a,y,t) = Ta; \qquad y \in]0,b[,\ t > t_0$

$T(x,0,t) = T_{y0}, T(x,b,t) = Tb; \qquad x \in]0,a[;\ t > t_0$

$T(x,y,t_0) = T_0(x,y) \qquad x \in [0,a],\ y \in [0,b]$

Résolution de l'équation bidimensionnelle :

$\frac{\partial T}{\partial t}(x,y,t) = \frac{T(x,t\ dt)}{dt}$ (1)

$\frac{\partial^2 T}{\partial x^2}(x,y,t) = \frac{T(x-dx,y,t)-2T(x,y,t)+T(x-dx,y,t)}{dx^2}$ (2)

$\frac{\partial^2 T}{\partial y^2}(x,y,t) = \frac{T(x,y-dy,t)-2T(x,y,t)+T(x,y-dy,t)}{dy^2}$ (3)

On obtient : $(1) - \alpha\ [(2) + (3)] = 0$

Pour simplifier la lecture on prend : $n = x, m = y\ et\ j = t$

On a donc : $\frac{T_{j+1}^{\frac{n}{m}}}{dt} - \alpha \left(\frac{T_j^{\frac{n-1}{m}} - 2T_j^{\frac{n}{m}} + T_j^{\frac{n+1}{m}}}{dx^2} + \frac{T_j^{\frac{n}{m-1}} - 2T_j^{\frac{n}{m}} + T_j^{\frac{n}{m+1}}}{dy^2} \right) = 0$

<=> $dt\ \alpha \left(\frac{T_j^{\frac{n-1}{m}} - 2T_j^{\frac{n}{m}} + T_j^{\frac{n+1}{m}}}{dx^2} + \frac{T_j^{\frac{n}{m-1}} - 2T_j^{\frac{n}{m}} + T_j^{\frac{n}{m+1}}}{dy^2} \right) = T_{j+1}^{\frac{n}{m}}$

Problème monodimensionnel

a) Structuration des données

Description des structures dans *calculs.h* :

Def_Domaine

Cette structure rassemble les dimensions spatiales et temporelles du problème données par l'utilisateur :
-*x* et *y*, les coordonnées spatiales et *t*, coordonnées temporelles
-*dx, dy* et *dt*, la distance entre deux valeurs de x, y et t
-*Nx, Ny* et *Nt*, le nombre d'échantillons
- *SourceDeChaleur* définit les positions des sources de chaleur

x	y	t
dx	dy	dt
Nx	Ny	Nt
SourceDeChaleur		

Condition_Probleme

Temps_Init
Def_Domain

Comme son nom l'indique, cette structure permet de contenir les conditions du système, *Temps_Init* est une matrice qui contient les températures initiales du problème et *Domaines_Init* est de type *Def_Domaine* et donc contient les dimensions.

Materiau

K	C	rho
alpha	Nom [30]	

Materiau permet de contenir toutes les informations à propos d'un matériau.

K, *C* et *rho* sont trois caractéristiques données qui permettent de décrire un matériau, ils vont permettre de calculer *alpha*, coefficient important pour calculer l'équation différentielle de diffusion de chaleur.

ListeMateriaux contient seulement la liste chaînée de tous les matériaux disponibles pour notre dispositif électronique.

b) Algorithme proposé

Dans calcul, *Milieuxhomogene* permet de résoudre, en fonction des dimensions d'un domaine homogène, un problème de diffusion de chaleur monodimensionnel ou bidimensionnel. De plus, il permet de vérifier que les dimensions *Nx et Ny* soient correctes pour effectuer les calculs et de choisir la fonction à appeler.

Comme nous avons plusieurs cas, nous utilisons un switch s*witch(init.Domaine_Init.Ny)* pour faciliter la compréhension du code et éviter d'utiliser beaucoup de test if/else. Nous avons trois cas, d'abord si *Ny* = 1, on effectuera la fonction de résolution d'équation différentielle en une dimension (*Calculs1D*). Dans le deuxième cas, un message d'erreur s'affiche car le nombre d'échantillon en y doit être plus grand que 2. Pour finir, si *Ny* > 2, on utilisera la fonction *Calculs2D* pour les problèmes à deux dimensions.

Nous allons maintenant décrire ce qui se passe à l'étape 3 du premier cas dans le cadre d'un problème à une dimension :

<u>Entrée de Calcul1D </u>: *Condition_Probleme init, char* adresse*

1) Ouverture du fichier texte, d'adresse donnée en paramètre.

2) Création et écriture d'une matrice contenant les valeurs des sources de chaleur (0 pour une chaleur fixe, 1 pour une chaleur variable)

3) Calcul du coefficient (*alpha* × *dt)*

4) Initialisation d'un vecteur *calcule* qui contiendra les valeurs de la simulation

5) Pour chaque échantillon du temps, faire :

   a) Remplir la première colonne du fichier de la température *Ta* au bord.

   b) Réinitialisation des points de température droit et gauche

   c) Pour chaque échantillon sur x, faire :

      i) Calcul de *calcule* au point x et écriture dans calcule[n]

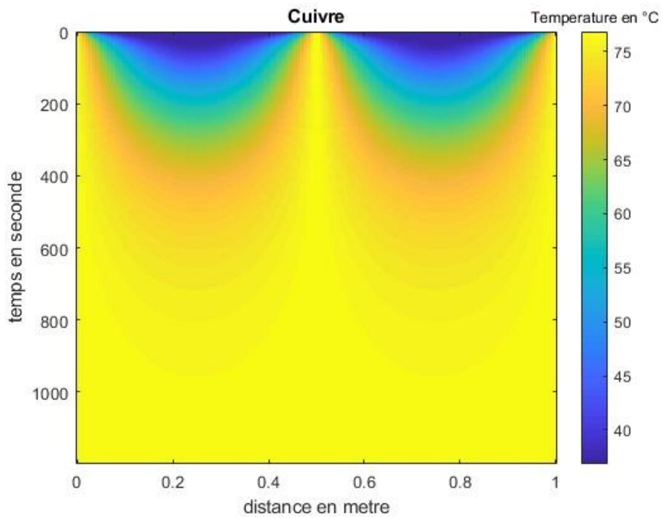
      ii) Changement des points de température important (droite et gauche)

      iii) Enregistrement des données

   d) Remplir la dernière colonne du fichier de la température *Tb* au bord.

6) Fermeture du fichier

7) Libération de la mémoire



Voici un affichage Matlab que nous avons réalisé avec le cuivre comme matériau. On peut visualiser le comportement en fonction du temps grâce à cette image, nous avons le temps (en seconde) en ordonnée et la distance de la barre (en mètre) en abscisse.

Nous pouvons donc valider cette première méthode de modélisation.

Problème bidimensionnel

Dans cette seconde partie, nous allons analyser l'algorithme qui permet de modéliser la diffusion de chaleur en 2D. Celui-ci ressemble à *Calcul1D* en plus complexe. L'algorithme parcourt le temps dans la boucle for principale (j), en créant des fichiers numérotés qui représenteront la température en chaque point à chaque itération de l'échantillonnage de t.

<u>Entrée</u> : *Condition_Probleme init, Materiau Mat, char* adresse*

1) Initialisation des variables

2) Initialisation de *calcule* (matrice réponse) : calcule est à présent une matrice de 0 de taille *Nx, Ny*

3) Initialisation d'adresse : On concatène l'adresse en fonction de la valeur de j.

4) Ecriture dans un fichier principal des dimensions du problème et températures initiales en chaque point de l'espace

5) Pour chaque échantillon de t, faire :

   a) Réinitialisation de l'adresse

   b) Initialisation et écriture d'une matrice contenant les valeurs des sources de chaleur (0 pour une chaleur fixe, 1 pour une chaleur variable)

   c) Création d'un nouveau fichier avec la nouvelle adresse

   d) Test d'ouverture

   e) Ecriture des dimensions du problèmes *Nx Ny* et la valeur du temps (en seconde) de l'échantillon

   f) Sauvegarde des températures initiales dans un vecteur *tmpData*

   g) Pour chaque échantillon de y, faire :

      i) Ecriture de la première colonne *Ta* aux bords

      ii) Réinitialisation des points de températures (gauche, droite, bas, haut) à partir de *tmpData*

      iii) Pour chaque échantillon de *x*, faire :

         a) Calcul avec l'équation différentielle de la température au point *x*, *y* dans calcule

         b) Changement des points de température pour correspondre au prochain point

         c) Sauvegarde de calcule dans *tmpData*

         d) Enregistrement de la température dans le fichier ouvert

iv) Ecriture de la dernière colonne *Tb* aux bords

6) Libération de la mémoire

On réalise deux simulations :

Lors de cette simulation nous avons cherché à montrer la diffusion de la chaleur d'une plaque de cuivre chaude au centre qui se refroidit au cours du temps avec les températures initiales suivante
Température aux bords : 27°C
Température de la plaque : 37°C
Température au centre : 77°C
https://www.youtube.com/watch?v=0PXA12xIz0U&feature=youtu.be

Dans cette seconde simulation on a des bords initialement plus chauds que la plaque et le centre de la plaque qui reste à température constante avec les températures initiales suivante :
Température aux bords : 37°C
Température de la plaque : 17°C
Température au centre : 77°C
https://www.youtube.com/watch?v=DN97321XPxU&feature=youtu.be

Grace à ces deux simulations nous pouvons valider cette méthode.

Problème hétérogène et chaleur variable

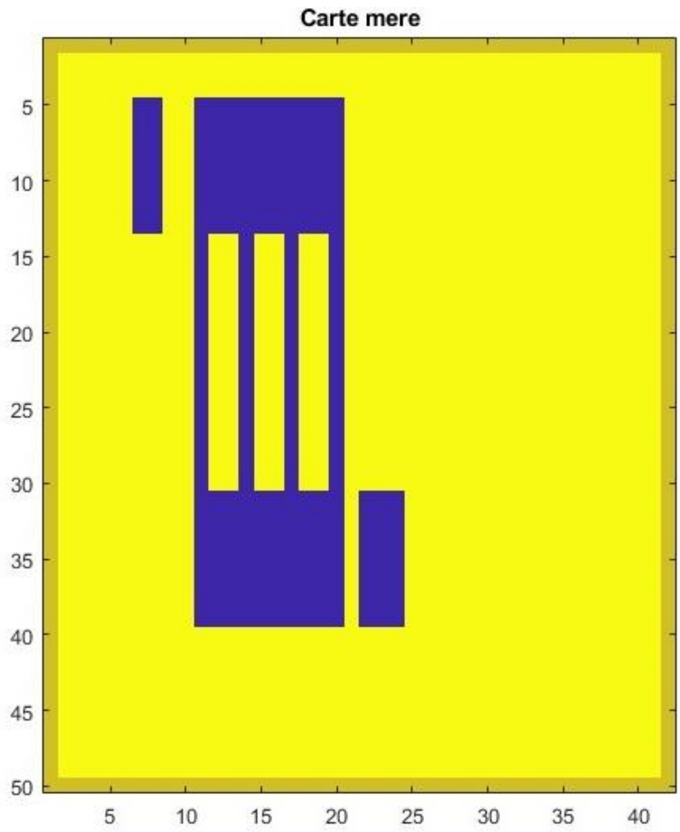
Maintenant, on considère des espaces hétérogènes. On change donc nos fonctions homogènes pour passer sur des conditions hétérogènes. La structure *Def_Domaine* change, on rajoute *AlphaLocal*, une matrice qui contient *alpha* en chaque point de l'espace étudié déterminé à partir d'un fichier d'entrée qui détient les différents types de matériau. L'adresse de réponse est obtenue avec concaténation.

Pour représenter une source de chaleur variable, il a fallu diviser en deux le problème. D'abord dans un cas monodimensionnel, on génère une matrice chaleur dans un fichier texte de taille *Nx Nt* qui contient la température en chaque point de l'espace. Ensuite, en bidimensionnel, grâce à un nombre *Nt* de fichiers textes contenant des matrices de taille *Nx Ny* contenues dans le dossier *ChaleurVariable*.

On réalise une simulation allégée avec trois zones principales en cuivre, une carte mère en époxy [2] et de l'air sur les contours. Les zones en cuivre représentent les circuits électroniques et leur alimentations (CPU et GPU) ainsi que des fils les reliant.

Voici la vidéo de la simulation réalisée :

https://www.youtube.com/watch?v=DkNVT9I0Cf0&feature=youtu.be



Représentation du type de matériau utiliser

Bleu : cuivre
Jaune : époxy [2]
Ocre : air

Grace à cette simulation nous pouvons valider cette méthode.

Complexité

Concernant la complexité, nous avons $O(n^2)$ pour notre problème monodimensionnel et une complexité $O(n^3)$ pour bidimensionnel. La valeur de la puissance correspond au nombre de dimension considérées. On a une dimension temporelle et une ou deux dimension(s) spatiale(s) dans notre cas.

Suggestions d'améliorations et bugs connus

Nous n'avons pas réussi à implémenter une résolution de la dérivée temporelle avec la méthode de Runge Kutta d'ordre 2 dans notre projet, ce qui aurait permis d'augmenter le pas de temps et donc de réduire les temps de calcul.

On pourrait aussi alléger le poids des fichiers en utilisant des fichiers binaires au lieu de fichiers textes qui ont un poids conséquent.

Concernant les bugs nous avons des problèmes d'overflow. Expérimentalement, quand les plages de températures varient entre 290°K et 400°K, nous avons déterminé que quand *dt* est supérieur à 150 ms, nous avons un problème d'overflow. De même, lorsque *dx* et *dy* sont inférieurs à 6 mm.

Conclusion

Nous avons répondu à chacun des problèmes posés lors de ce projet, et, présenté dans ce rapport une description détaillée de notre travail. Notre programme en C modélise donc la diffusion de chaleur en une et deux dimension(s). Il permet aussi de simuler cette diffusion dans un espace hétérogène avec différentes sources de chaleur. De plus, nous avons conçu un petit code Matlab permettant visualiser nos simulations de diffusion de chaleur.

Bibliographie

[1] Programmation et Méthodes Numériques-Résolution numérique des équations différentielles_CM4/M. Casaletti. - Laboratoire d'Electronique et Electromagnétisme (L2E)-Université Pierre et Marie Curie. – 39 ; PDF.

[2] TISSU DE VERRE ÉPOXY FR4/plastiservice. - http://www.plastiservice.com/uploads/produits/59/6_tissu_de_verre_epoxy_fr4_XY%20FR4%20OK.pdf. – 2 ; PDF.