MÉMOIRE

présenté

à

L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI

Comme exigence partielle pour l'obtention du grade de

Maître ès Sciences Appliquées

(M.Sc.A.)

par

Réjean Ouellet, B.Sc.

ÉTUDE DU PROBLÈME D'ABLATION À DEUX DIMENSIONS PAR LA MÉTHODE DES ÉLÉMENTS FINIS DE FRONTIÈRE

Mars 1987

Bibliothèque
Paul-Émile-Boulet

UQAC
Université du Québec
à Chicoutimi

Mise en garde/Advice

Afin de rendre accessible au plus grand nombre le résultat des travaux de recherche menés par ses étudiants gradués et dans l'esprit des règles qui régissent le dépôt et la diffusion des mémoires et thèses produits dans cette Institution, **l'Université du Québec à Chicoutimi (UQAC)** est fière de rendre accessible une version complète et gratuite de cette œuvre.

Motivated by a desire to make the results of its graduate students' research accessible to all, and in accordance with the rules governing the acceptation and diffusion of dissertations and theses in this Institution, the **Université du Québec à Chicoutimi (UQAC)** is proud to make a complete version of this work available at no cost to the reader.

L'auteur conserve néanmoins la propriété du droit d'auteur qui protège ce mémoire ou cette thèse. Ni le mémoire ou la thèse ni des extraits substantiels de ceux-ci ne peuvent être imprimés ou autrement reproduits sans son autorisation.

The author retains ownership of the copyright of this dissertation or thesis. Neither the dissertation or thesis, nor substantial extracts from it, may be printed or otherwise reproduced without the author's permission.

RÉSUMÉ

L'ablation de solides d'aluminium est simulée en utilisant la méthode des éléments finis de frontière qui est basée sur la combinaison des équations intégrales classiques et des concepts d'éléments finis.

La frontière est approximée par des éléments linéaires isoparamétriques et le domaine par des éléments triangulaires constants. Dû au déplacement de la frontière et dû à la nécessité d'intégrer sur le domaine, une technique de maillage automatique est utilisée. Différents pas de temps, points d'intégration et configurations triangulaires sont utilisés pour montrer la sensibilité de la méthode face à ces trois facteurs. Pour réduire le temps de calcul, le nombre d'éléments triangulaires discrétisant le domaine est diminué au fur et à mesure que le gradient de température, à l'intérieur du domaine, s'approche de zéro.

La qualité des résultats obtenus pour l'ablation d'un cylindre circulaire et d'un demi-cylindre elliptique d'aluminium montre que la méthode des éléments finis de frontière donne d'excellents résultats.

Réjean Guillet
Etudiant

R. T. Lee
Directeur de recherche

REMERCIEMENTS

L'auteur tient spécialement à souligner sa reconnaissance pour son directeur, le professeur Rung T. Bui, Ph.D. et son codirecteur, le professeur André Charette, D.Sc. pour la suggestion de ce mémoire et pour les nombreuses discussions enrichissantes. Leurs précieux conseils et encouragements se sont avérés être une source de motivation essentielle à la réalisation de ce travail.

L'auteur tient à remercier le professeur Augustin Gakwaya, Ph.D. de l'Université Laval de même que M. Thierry Bourgeois, M. Guy Simard, M. Jocelyn Tremblay et M. Jean Perron pour l'aide qu'ils ont apportée lors du développement du modèle mathématique.

L'auteur tient à remercier Madame Josée Hudon pour sa grande contribution à la mise en page de ce document.

Finalement, l'auteur désire remercier les Fonds FCAR pour leur support financier.

TABLE DES MATIÈRES

LISTE DES TABLES

LISTE DES FIGURES

INTRODUCTION

Le four de fusion est un des équipements importants dans les industries métallurgiques. Le procédé de fusion est difficile à simuler et une bonne compréhension de ce dernier est essentielle à toute tentative d'améliorer la performance ou la conception des fours.

L'étude théorique que représente ce mémoire entre dans le cadre de travaux préparatoires précédant un projet industriel sur la modélisation du four de fusion de l'aluminium. Le travail consiste à développer une méthode pouvant simuler la fonte d'un bloc d'aluminium solide en utilisant une distribution de flux de chaleur et dont la partie fondue est supposée enlevée immédiatement laissant seulement la partie solide. Ce genre de phénomène, appelé ablation, est aussi connu sous le nom de problème de Landau (1950).

La non-linéarité inhérente à ce problème pose des difficultés mathématiques assez importantes limitant ainsi les solutions analytiques à des situations relativement simples (Rubinstein, 1971 et Lunardini, 1981). Cependant, dû à leurs applications pratiques à de nombreuses situations physiques (Ockendon et al., 1975 ou Wilson et al., 1978), des méthodes de résolution approximative ou numérique s'imposent.

Les techniques numériques les plus souvent utilisées dans la littérature sont les différences finies et les éléments finis. Dans ces deux

méthodes, on divise le domaine en utilisant un maillage ou une grille et en définissant une série de points nodaux pouvant se situer à l'intérieur ou sur la frontière du domaine. Dans ce travail, nous utilisons la méthode des éléments finis de frontière, technique basée sur la combinaison des équations intégrales classiques et des concepts d'éléments finis. Dans cette méthode, les noeuds sont seulement définis sur la surface externe et les inconnues se trouvant à l'intérieur du domaine ne sont pas requises pour solutionner le problème puisque ce dernier est réduit mathématiquement à celui de trouver une solution sur la frontière. Cette réduction est rendue possible par l'application de la seconde identité de Green et en utilisant la solution fondamentale (fonction de Green pour un domaine infini) qui satisfait l'équation de diffusion. Même si l'inclusion d'une condition initiale produit une intégrale sur le domaine, les inconnues nodales sont toujours situées sur la frontière et le problème continue d'être un problème à la frontière. En effet, la satisfaction de la condition frontière produit un système d'équations linéaires servant à déterminer les inconnues sur la frontière. Une fois que toutes les valeurs sur la frontière sont connues, nous pouvons calculer n'importe quelle variable interne puisqu'elle est fonction des valeurs à la frontière.

Les principaux avantages de cette technique sont (Wrobel et al., 1981a) la réduction du nombre d'inconnues gouvernant le problème et aussi la simplicité dans les données d'entrée nécessaires au bon fonctionnement du programme. De plus, la méthode donne généralement de meilleurs résultats que les méthodes des différences finies ou des éléments finis, spécialement pour les régions où le gradient de température est élevé, et

cela en utilisant un pas de temps beaucoup plus élevé que les autres méthodes mentionnées.

Jusqu'à maintenant, quelques travaux ont été faits pour des problèmes d'ablation à une dimension utilisant la méthode des éléments finis de frontière (Chuang et al. (1971), Banerjee et Shaw (1982), Shaw (1982)). Nous extentionnons la méthode utilisée par ces auteurs à deux dimensions, tel que proposé par Banerjee et Shaw (1982) et réalisé par O'Neil (1983). Pour ce dernier cependant, il n'y avait pas de changement soudain de température à la frontière. Comme conséquence, O'Neil pouvait se permettre de négliger les variations de température avec le temps et par le fait même aucune intégration sur le domaine n'était nécessaire. Les problèmes que nous considérons dans ce travail diffèrent en ce sens qu'un grand gradient de température existe à la frontière.

Le but de ce travail est donc de simuler un problème d'ablation à deux dimensions en utilisant la méthode des éléments finis de frontière. La rédaction du mémoire suit le plan suivant:

I) - Développement de l'équation de diffusion avec propriétés thermiques constantes sous une forme intégrale équivalente pour pouvoir solutionner par la méthode des éléments finis de frontière.

II) - Discrétisation de l'équation intégrale dans le temps et l'espace avec traitement des singularités apparaissant dans les intégrales de contour et intégration en coordonnées polaires sur les triangles discrétisant le domaine.

- Développement de la technique de résolution pour suivre l'évolution de la température du solide en fonction du temps ainsi que le déplacement de la frontière.

III) - Simulation de l'ablation d'un cylindre circulaire d'aluminium pour l'optimisation de la méthode en ce qui a trait au choix du pas de temps, au choix du nombre de points d'intégration et au choix de la configuration triangulaire discrétisant le domaine.

- Développement de techniques pouvant diminuer le temps de calcul.

- Étude de l'ablation d'un demi-cylindre elliptique d'aluminium soumis à trois différentes distributions de flux de chaleur.

- Étude des limites éventuelles dans le choix du problème d'ablation susceptible d'être simulé en utilisant la configuration triangulaire proposée.

CHAPITRE I

FORMULATION MATHÉMATIQUE

1.1 Définition du problème

Considérons le problème d'ablation d'un solide tridimensionnel homogène et isotrope dont une des dimensions est supposée infinie de sorte que l'échange thermique est bidimensionnel.

L'équation différentielle gouvernant ce problème est donnée par:

$$\nabla \cdot k\nabla T(\vec{r},t) = \rho c \frac{\partial T(\vec{r},t)}{\partial t} \; ; \; \vec{r} \in \Omega(t) \; , \; t > 0 \tag{1.1}$$

avec la condition initiale

$$T(\vec{r},0) = T_I(\vec{r}) \; ; \; \vec{r} \in \Omega(0) \tag{1.2}$$

et la condition frontière

$$\left(k \frac{\partial T(\vec{r},t)}{\partial n} + q(\vec{r},t) \right) \vec{n} = \rho L \frac{d\vec{S}(\vec{r})}{dt} \; ; \; \vec{r} \in \Gamma(t) \; , \; t > 0 \tag{1.3}$$

où k est la conductivité thermique, c la chaleur massique, ρ la masse volumique et L la chaleur latente de fusion. De plus, $d\vec{S}(\vec{r})/dt$ représente la vitesse de déplacement de la frontière, et est non nul seulement lorsque le changement de phase survient i.e. lorsque le point de fusion T_f (qui est supposé constant) est atteint au point $\vec{r} \varepsilon \Gamma(t)$. $\vec{n}$ est un vecteur unitaire pointant normalement vers l'extérieur du domaine (voir fig. 1).

Puisque nous choisissons $\vec{n}$ comme étant la direction positive, la valeur du flux q est par conséquent négative car pour un problème d'ablation, le flux q est dirigé vers l'intérieur du domaine.

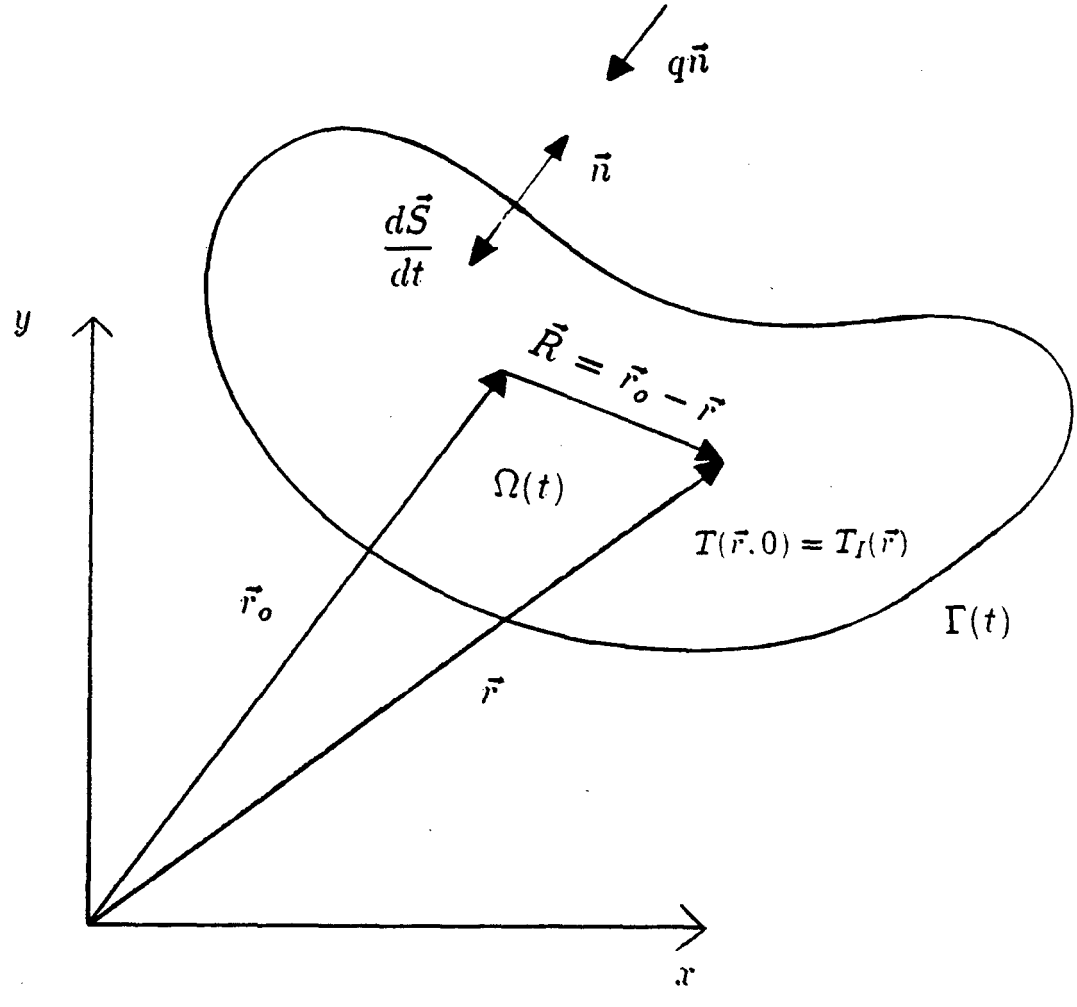


Figure 1 : Définition du problème

Maintenant, pour solutionner l'équation de conduction (1.1) soumise à la condition frontière (1.3) par la méthode des éléments finis de frontière, il est nécessaire d'avoir une diffusivité thermique ($\alpha = k/\rho c$) constante (due à l'introduction de la fonction de Green définie à la section

(1.2) (Morse et al., 1953)). Il y a alors deux alternatives qui se présentent. La première est de considérer les valeurs de k et de ρc comme étant constantes et la deuxième de considérer, s'il y a lieu, une variation de ces valeurs en fonction de la température. Cependant, leur variation doit être proportionnelle pour que la diffusivité demeure constante. Toutefois, pour tenir compte de la variabilité de k et de ρc, tout en rendant possible l'utilisation de la méthode des éléments finis de frontière, nous devons réécrire les équations (1.1) à (1.3) différemment en utilisant, soit la transformation de Kirchhoff, soit l'enthalpie sensible ce qui est présenté en détails à l'appendice B.

Il est à noter que si la diffusivité du solide est constante, alors les résultats obtenus, en utilisant l'une ou l'autre des deux formulations, seront évidemment les mêmes. Cependant, lorsque la diffusivité est fonction de la température du solide, comme c'est le cas pour l'aluminium (Touloukian, 1973), nous devons, dans cette situation, prendre une diffusivité moyenne constante. En procédant ainsi, il serait dangereux de garder k et ρc variables puisque les résultats diffèrent de façon significative suivant que l'on utilise l'une ou l'autre des deux formulations tel qu'expliqué et illustré à l'appendice B. Par conséquent, nous avons choisi de prendre k et ρc comme étant constantes.

En procédant ainsi, nous pouvons réécrire l'équation (1.1) de la façon suivante:

$$\alpha \nabla^2 T(\vec{r},t) = \frac{\partial T(\vec{r},t)}{\partial t} \; ; \; \vec{r} \in \Omega(t) \; , \; t > 0 \qquad (1.4)$$

1.2 Formulation intégrale

Tout comme pour la méthode des éléments finis conventionnels, nous devons réécrire l'équation (1.4) sous une forme intégrale équivalente pour pouvoir solutionner par la méthode des éléments finis de frontière.

Pour ce faire, introduisons la fonction de Green (Morse et al., 1953 ou Beck, 1984) qui est donnée par:

$$G^* = G^*(\vec{r}, t|\vec{r}_o, \tau) = \frac{1}{4\pi\alpha(t-\tau)} \exp\left(\frac{-\|\vec{r}-\vec{r}_o\|}{4\alpha(t-\tau)}\right) u(t-\tau) \qquad (1.5)$$

et est solution, pour un domaine infini, de l'équation différentielle suivante:

$$\alpha\nabla^2 G^* - \frac{\partial G^*}{\partial t} = -\delta(\vec{r}, \vec{r}_o)\delta(t, \tau)$$

où l'on considère $\vec{r}_0$ et τ fixes et $\vec{r}$ et t variables. Si, par contre, $\vec{r}$ et t sont gardés fixes et que l'on fait varier $\vec{r}_0$ et τ, alors G^* est solution de l'équation différentielle ci-dessous.

$$\alpha\nabla_o^2 G^* + \frac{\partial G^*}{\partial \tau} = 0 \ ; \ \tau < t \qquad (1.6)$$

Le terme $u(t-\tau)$ représente la fonction échelon-unité, δ la pseudo fonction delta de Dirac, $\vec{r}$ le vecteur position du point (x,y) et $\vec{r}_0$ celui du point (x_0, y_0) appelé respectivement observateur et source.

Pour fins de concrétisation, une interprétation physique possible de la fonction de Green G* est de la prendre comme étant la température à un point (x, y) au temps t générée par une source ponctuelle instantanée, introduite à un point (x_0, y_0) au temps τ avec une intensité de grandeur unitaire (Carslaw et al., 1959).

La fonction de Green étant maintenant introduite, il ne reste plus qu'à trouver l'équation intégrale équivalente à (1.4). En effet, on a

$$\frac{\partial(G^*T(\vec{r}_o,\tau))}{\partial\tau} = G^*\frac{\partial T(\vec{r}_o,\tau)}{\partial\tau} + T(\vec{r}_o,\tau)\frac{\partial G^*}{\partial\tau} \qquad (1.7)$$

En utilisant les équations (1.4) et (1.6), l'équation (1.7) devient:

$$\frac{\partial(G^*T(\vec{r}_o,\tau))}{\partial\tau} = \alpha\left(G^*\nabla_o^2 T(\vec{r}_o,\tau) - T(\vec{r}_o,\tau)\nabla_o^2 G^*\right) \quad ; \tau < t$$

De plus, en intégrant cette dernière équation sur le domaine $\Omega(\tau)$ et ensuite par rapport au temps de 0 à t-ε (où ε est un nombre positif plus petit que t et pouvant être aussi petit que l'on veut) on obtient:

$$\int_0^{t-\epsilon}\int_{\Omega(\tau)} \frac{\partial(G^*T(\vec{r}_o,\tau))}{\partial\tau} d\Omega_o\, d\tau = \alpha\int_0^{t-\epsilon}\int_{\Omega(\tau)} \left(G^*\nabla_o^2 T(\vec{r}_o,\tau) - T(\vec{r}_o,\tau)\nabla_o^2 G^*\right) d\Omega_o\, d\tau \qquad (1.8)$$

En appliquant la seconde identité de Green au membre de droite de l'équation (1.8), cette dernière devient:

$$\int_0^{t-\epsilon}\int_{\Omega(\tau)} \frac{\partial(G^*T(\vec{r}_o,\tau))}{\partial\tau} d\Omega_o\, d\tau = \alpha \int_0^{t-\epsilon}\int_{\Gamma(\tau)} \left(G^* \frac{\partial T(\vec{r}_o,\tau)}{\partial n} - T(\vec{r}_o,\tau)\frac{\partial G^*}{\partial n} \right) d\Gamma_o\, d\tau \qquad (1.9)$$

où $\frac{\partial}{\partial \vec{n}}$ représente la dérivation suivant la direction du vecteur normal unitaire $\vec{n}$ dirigé vers l'extérieur de la surface du domaine $\Omega(\tau)$.

Pour pouvoir intégrer le membre de gauche de l'équation (1.9), nous devons lui apporter une attention toute particulière. En effet, il n'est pas possible d'inverser l'ordre d'intégration (i.e. intégrer par rapport au temps et ensuite par rapport à l'espace) puisque le domaine subit une déformation au fur et à mesure que le temps progresse. Cependant, il existe une façon de réécrire ce terme en utilisant le théorème de transport de Reynolds généralisé (O'Neill, 1983) qui dit:

$$\frac{\partial}{\partial\tau}\int_{\Omega(\tau)} g\, d\Omega_o = \int_{\Omega(\tau)} \frac{\partial g}{\partial\tau} d\Omega_o + \int_{\Gamma(\tau)} g \frac{dS(\vec{r}_o)}{dt} d\Omega_o \qquad (1.10)$$

où g est une fonction intégrable et différenciable sur $\Omega(\tau)$ et $d\vec{S}(\vec{r}_0)/dt$ est la vitesse de déplacement de la frontière suivant le vecteur normal $\vec{n}$.

En remplaçant g par $G^*T(\vec{r}_0,\tau)$ dans (1.10), le membre de gauche de l'équation (1.9) devient:

$$\int_0^{t-\epsilon}\int_{\Omega(\tau)} \frac{\partial(G^*T(\vec{r}_o,\tau))}{\partial\tau} d\Omega_o\, d\tau = -\int_0^{t-\epsilon}\int_{\Gamma(\tau)} \left(G^*T(\vec{r}_o,\tau)\frac{dS(\vec{r}_o)}{dt}\right) d\Gamma_o\, d\tau + \int_{\Omega(t-\epsilon)} (G^*T(\vec{r}_o,t-\epsilon))\, d\Omega_o - \int_{\Omega(0)} (G^*T(\vec{r}_o,0))\, d\Omega_o \quad (1.11)$$

Par conséquent, l'équation (1.8) devient:

$$\int_{\Omega(t-\epsilon)} (G^*T(\vec{r}_o,t-\epsilon))\, d\Omega_o = \alpha\int_0^{t-\epsilon}\int_{\Gamma(\tau)} \left(G^*\frac{\partial T(\vec{r}_o,\tau)}{\partial n} - T(\vec{r}_o,\tau)\frac{\partial G^*}{\partial n}\right) d\Gamma_o\, d\tau + \int_0^{t-\epsilon}\int_{\Gamma(\tau)} \left(G^*T(\vec{r}_o,\tau)\frac{dS(\vec{r}_o)}{dt}\right) d\Gamma_o\, d\tau + \int_{\Omega(0)} (G^*T(\vec{r}_o,0))\, d\Omega_o \quad (1.12)$$

La raison pour laquelle nous avons intégré par rapport au temps jusqu'à t-ε était d'éviter la singularité qui apparaît dans la fonction de Green lorsque $\tau \to t$ et $(x_0, y_0) \to (x, y)$. Toutefois, en prenant la limite lorsque $\varepsilon \to 0$ et en tenant compte des différentes positions que peut prendre le vecteur $\vec{r}$ (Brebbia et al., 1984) l'équation (1.12) devient:

$$\varphi(\vec{r})T(\vec{r},t) = \alpha\int_0^t\int_{\Gamma(\tau)} \left(G^*\frac{\partial T(\vec{r}_o,\tau)}{\partial n} - T(\vec{r}_o,\tau)\frac{\partial G^*}{\partial n}\right) d\Gamma_o\, d\tau + \int_0^t\int_{\Gamma(\tau)} \left(G^*T(\vec{r}_o,\tau)\frac{dS(\vec{r}_o)}{dt}\right) d\Gamma_o\, d\tau + \int_{\Omega(0)} (G^*T(\vec{r}_o,0))d\Omega_o \quad (1.13)$$

où $\varphi(\vec{r})$ est une constante qui dépend de l'emplacement du vecteur position $\vec{r}$, i.e.

$$\varphi(\vec{r}) = \begin{cases} 1. & \text{si } \vec{r} \in \Omega(\tau): \\ 1/2. & \text{si } \vec{r} \in \Gamma(\tau).\text{où } \Gamma(\tau) \text{ est un contour régulier en } \vec{r}: \\ \phi(\vec{r})/2\pi. & \text{si } \vec{r} \in \Gamma(\tau).\text{où } \Gamma(\tau) \text{ n'est pas un contour régulier} \\ & \text{en } \vec{r}.\text{et } \phi(\vec{r}) \text{ est l'angle intérieur du contour au} \\ & \text{point correspondant à } \vec{r}: \\ 0. & \text{si } \vec{r} \text{ est à l'extérieur de } \Omega(\tau). \end{cases} \tag{1.14}$$

En prenant la température de fusion T_f comme étant constante et aussi comme température de référence, i.e. $T_f = 0$ (Shaw, 1982), il s'en suit que le troisième terme (du membre de droite de l'équation (1.13)) est toujours nul. En effet, $d\vec{S}(\vec{r}_0)/dt$ est non nul seulement lorsque la température de fusion est atteinte à la frontière du point $\vec{r}_0$. Par conséquent, l'équation intégrale (1.13) se réécrit de la façon suivante:

$$\varphi(\vec{r})T(\vec{r},t) = \alpha \int_0^t \int_{\Gamma(\tau)} \left(G^* \frac{\partial T(\vec{r}_o,\tau)}{\partial n} - T(\vec{r}_o,\tau)\frac{\partial G^*}{\partial n} \right) d\Gamma_o \, d\tau + \int_{\Omega(0)} (G^* T(\vec{r}_o,0)) d\Omega_o \tag{1.15}$$

À partir de cette dernière équation intégrale et aussi de la condition initiale (1.2) et de la condition frontière (1.3), nous sommes maintenant en mesure de solutionner l'équation (1.4). Cependant, pour cela, nous devons discrétiser l'équation (1.15) dans l'espace et dans le temps, ce qui est fait au chapitre II.

CHAPITRE II

FORMULATION NUMÉRIQUE

2.1 Discrétisation de l'équation intégrale

Plutôt que de tenter de trouver des solutions analytiques à l'équation (1.15) pour des géométries et des conditions frontières particulières, nous réduisons cette équation à une forme algébrique pouvant être solutionnée par une approche numérique dans le but de considérer différentes géométries et conditions frontières.

2.1.1 Discrétisation dans le temps

Puisque les variations en fonction du temps de $T(\vec{r}_0,\tau)$, $\partial T(\vec{r}_0,\tau)/\partial n$ et de $\Gamma(\tau)$ ne sont pas connues a priori, une technique de discrétisation dans le temps par pas de temps (ne pas confondre avec la méthode des différences finies) doit être introduite pour pouvoir solutionner numériquement l'équation (1.15).

Il existe deux procédés différents de progression dans le temps (Brebbia et al., 1984) utilisant cette technique du pas de temps. Le premier traite chaque pas de temps comme un nouveau problème, et par le fait même, nous devons à la fin de chaque pas de temps, calculer la distribution de température pour un pas de temps donné à l'intérieur du

domaine pour pouvoir l'utiliser comme valeur initiale pour le prochain pas de temps. Pour le second procédé, le processus d'intégration dans le temps repart toujours au temps t = 0; par conséquent, lorsqu'un nouvel incrément dans le temps est ajouté, une intégrale de type convolution doit être évaluée sur un contour variable en revenant dans le temps jusqu'à la condition initiale. De plus, en utilisant ce dernier procédé, il n'est pas nécessaire de calculer la distribution de température à l'intérieur du domaine lors de la simulation. Néanmoins, il y a une intégrale sur le domaine à évaluer qui peut être cependant transformée en une intégrale de contour équivalente si $T(\vec{r}_0,0)$ satisfait l'équation de Laplace, i.e.

$$\nabla^2 T(\vec{r}_o,0) = 0$$

Malgré le fait que nous devons intégrer sur le domaine à chaque pas de temps lors de l'utilisation du premier procédé, nous avons choisi de le prendre puisque dans le second procédé, le nombre d'intégrales de contour à évaluer augmente considérablement lorsque la simulation progresse dans le temps. De plus, nous allons voir ultérieurement (voir sections 3.1.2.2 et 3.1.3) qu'il existe deux considérations particulières permettant de minimiser l'évaluation des intégrales sur le domaine lorsque la simulation avance dans le temps.

Par conséquent, en utilisant le premier procédé de progression dans le temps, l'équation (1.15) s'écrit, pour un pas de temps donné, sous la forme suivante:

$$\varphi(\vec{r})T(\vec{r},t) = \alpha \int_{t_{s-1}}^{t_s} \int_{\Gamma(\tau)} \left(G^* \frac{\partial T(\vec{r}_o,\tau)}{\partial n} - T(\vec{r}_o,\tau)\frac{\partial G^*}{\partial n} \right) d\Gamma_o \, d\tau + \int_{\Omega(t_{s-1})} (G^* T(\vec{r}_o, t_{s-1}))d\Omega_o \quad (2.1)$$

pour s = 1, 2, ..., F où t_F représente le temps final de la simulation et t_0 = 0 s.

Puisque nous ne connaissons pas la variation du déplacement de la frontière en fonction du temps et que nous voulons inverser l'ordre d'intégration pour faciliter l'évaluation de l'équation intégrale (2.1), nous supposons qu'il n'y a pas de déplacement à l'intérieur d'un même pas de temps ($\Delta t_s = t_s - t_{s-1}$) i.e.

$$\Gamma(\tau) = \Gamma(t_s) \quad \text{pour} \quad t_{s-1} < \tau \leq t_s.$$

De plus, ne connaissant pas non plus les variations de $T(\vec{r}_0,\tau)$ et de $\frac{\partial T(\vec{r}_0,\tau)}{\partial n}$ en fonction du temps, nous supposons également qu'ils demeurent constants sur chaque pas de temps, i.e.

$$T(\vec{r}_o,\tau) = T(\vec{r}_o,t_s) \quad et \quad \frac{\partial T(\vec{r}_o,\tau)}{\partial n} = \frac{\partial T(\vec{r}_o,t_s)}{\partial n}$$
$$\text{pour} \quad t_{s-1} < \tau \leq t_s.$$

Il est évidemment possible de supposer des variations linéaires ou même quadratiques de $T(\vec{r}_0,\tau)$ et de $\partial T(\vec{r}_0,\tau)/\partial n$ en fonction du temps (Brebbia et al., 1984) mais pour des raisons de temps de calcul, nous avons décidé de ne pas considérer ces deux types de variations.

En prenant en considération les hypothèses qui viennent d'être énoncées, l'équation (2.1) devient:

$$\varphi(\vec{r},t_s)T(\vec{r},t_s) = \alpha \int_{\Gamma(t_s)} \left(\frac{\partial T(\vec{r}_o,t_s)}{\partial n} \int_{t_{s-1}}^{t_s} G^* d\tau \right) d\Gamma_o$$
$$- \alpha \int_{\Gamma(t_s)} \left(T(\vec{r}_o,t_s) \int_{t_{s-1}}^{t_s} \frac{\partial G^*}{\partial n} d\tau \right) d\Gamma_o \qquad (2.2)$$
$$+ \int_{\Omega(t_{s-1})} (G^* T(\vec{r}_o,t_{s-1}))\, d\Omega_o$$

Comme on le verra à la section (2.1.2), il est possible d'intégrer analytiquement les intégrales suivantes:

$$\int_{t_{s-1}}^{t_s} G^* d\tau \quad et \quad \int_{t_{s-1}}^{t_s} \frac{\partial G^*}{\partial n} d\tau. \qquad (2.3)$$

2.1.2 Discrétisation dans l'espace

Pour la discrétisation dans l'espace, nous commençons par discrétiser la frontière pour pouvoir évaluer les intégrales de contour apparaissant dans l'équation (2.2).

Pour ce faire, nous approximons la frontière $\Gamma(\tau)$ à l'aide de N segments de droite que l'on appelle éléments linéaires (voir fig. 2.1) avec une interpolation linéaire de la température $T(\vec{r}_0,t_s)$ et du gradient $\partial T(\vec{r}_0,t_s)/\partial n$. Ce type d'élément est appelé élément isoparamétrique, i.e. les fonctions de transformation géométrique sont identiques aux fonctions d'interpolation. Ces fonctions sont données par:

$$\psi_{(1)} = \frac{1-\xi}{2} \quad \text{et} \quad \psi_{(2)} = \frac{1+\xi}{2} \tag{2.4}$$

$$\text{où} \quad -1 \leq \xi \leq 1 .$$

Par conséquent, les distributions de température et de gradient de température sur l'élément Γ_j sont données respectivement par:

$$T(\vec{r}_{oj}(\xi).t_s) = \left[\psi_{(1)}\ \psi_{(2)}\right] \begin{bmatrix} T(\vec{r}_o^{\,j}.t_s) \\ T(\vec{r}_o^{\,j+1}.t_s) \end{bmatrix} \tag{2.5}$$

et

$$\frac{\partial T(\vec{r}_{oj}(\xi).t_s)}{\partial n} = \left[\psi_{(1)}\ \psi_{(2)}\right] \begin{bmatrix} \frac{\partial T(\vec{r}_o^{\,j}.t_s)}{\partial n} \\ \frac{\partial T(\vec{r}_o^{\,j+1}.t_s)}{\partial n} \end{bmatrix} \tag{2.6}$$

où $\vec{r}_0^{\,j}$ et $\vec{r}_0^{\,j+1}$ représentent respectivement les vecteurs positions des noeuds (j) et (j + 1) tels qu'illustrés à la figure (2.1). De même $\vec{r}_{oj}(\xi)$ représente l'ensemble des vecteurs positions décrivant l'élément Γ_j, i.e.

$$\vec{r}_{oj}(\xi) = \left[\psi_{(1)}\ \psi_{(2)}\right] \begin{bmatrix} \vec{r}_o^{\,j} \\ \vec{r}_o^{\,j+1} \end{bmatrix} \tag{2.7}$$

$$\text{pour} \quad -1 \leq \xi \leq 1 .$$

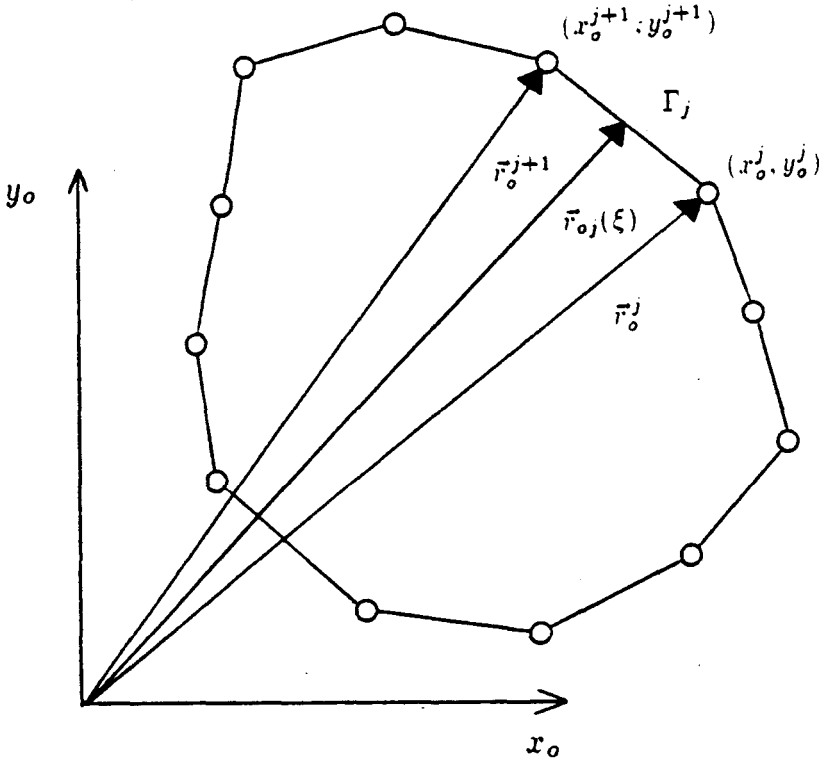


Figure 2.1: Discrétisation d'une frontière en 12 éléments linéaires avec numérotation des noeuds suivant le sens anti-horaire.

En considérant cette discrétisation de la frontière, l'équation (2.2) se réécrit de la façon suivante:

$$\varphi(\vec{r},t_s)T(\vec{r},t_s) = \sum_{j=1}^{N}\left(\alpha\int_{\Gamma_j(t_s)}\left(\left[\psi_{(1)}\ \psi_{(2)}\right]\int_{t_{s-1}}^{t_s} G^*\,d\tau\right)d\Gamma_o\begin{bmatrix}\frac{\partial T(\vec{r}_o^{\,j},t_s)}{\partial n}\\ \frac{\partial T(\vec{r}_o^{\,j+1},t_s)}{\partial n}\end{bmatrix}\right)$$
$$-\sum_{j=1}^{N}\left(\alpha\int_{\Gamma_j(t_s)}\left(\left[\psi_{(1)}\ \psi_{(2)}\right]\int_{t_{s-1}}^{t_s}\frac{\partial G^*}{\partial n}d\tau\right)d\Gamma_o\begin{bmatrix}T(\vec{r}_o^{\,j},t_s)\\ T(\vec{r}_o^{\,j+1},t_s)\end{bmatrix}\right)$$
$$+\int_{\Omega(t_{s-1})}(G^*T(\vec{r}_o,t_{s-1}))\,d\Omega_o \qquad (2.8)$$

ou sous une forme abrégée:

$$\varphi(\vec{r},t_s)T(\vec{r},t_s) = \sum_{j=1}^{N}\left(\alpha\left[g_j^{(1)}(\vec{r},t_s)\;\; g_j^{(2)}(\vec{r},t_s)\right]\begin{bmatrix}\frac{\partial T(\vec{r}_o^{\,j},t_s)}{\partial n}\\ \frac{\partial T(\vec{r}_o^{\,j+1},t_s)}{\partial n}\end{bmatrix}\right)$$
$$-\sum_{j=1}^{N}\left(\alpha\left[h_j^{(1)}(\vec{r},t_s)\;\; h_j^{(2)}(\vec{r},t_s)\right]\begin{bmatrix}T(\vec{r}_o^{\,j},t_s)\\ T(\vec{r}_o^{\,j+1},t_s)\end{bmatrix}\right)$$
$$+\int_{\Omega(t_{s-1})}(G^*T(\vec{r}_o,t_{s-1}))\,d\Omega_o$$

où

$$g_j^{(m)}(\vec{r},t_s) = \int_{\Gamma_j(t_s)}\left(\psi_{(m)}\int_{t_{s-1}}^{t_s}G^*d\tau\right)d\Gamma_o \qquad (2.9)$$

et

$$h_j^{(m)}(\vec{r},t_s) = \int_{\Gamma_j(t_s)}\left(\psi_{(m)}\int_{t_{s-1}}^{t_s}\frac{\partial G^*}{\partial n}d\tau\right)d\Gamma_o\;; \qquad (2.10)$$

avec

$$\vec{r}_o^{\,N+1} = \vec{r}_o^{\,1} \quad et \quad m = 1,2\,.$$

En regroupant les termes en $T(\vec{r}_0^{\,j},t_s)$ et en $\partial T(\vec{r}_0^{\,j+1},t_s)/\partial n$ on obtient:

$$\varphi(\vec{r},t_s)T(\vec{r},t_s) = \sum_{j=1}^{N}\left(\alpha\left(g_j^{(1)}(\vec{r},t_s)+g_{j-1}^{(2)}(\vec{r},t_s)\right)\frac{\partial T(\vec{r}_o^{\,j},t_s)}{\partial n}\right)$$
$$-\sum_{j=1}^{N}\left(\alpha\left(h_j^{(1)}(\vec{r},t_s)+h_{j-1}^{(2)}(\vec{r},t_s)\right)T(\vec{r}_o^{\,j},t_s)\right) \qquad (2.11)$$
$$+\int_{\Omega(t_{s-1})}(G^*T(\vec{r}_o,t_{s-1}))\,d\Omega_o$$

où

$$g_0^{(2)}(\vec{r},t_s) = g_N^{(2)}(\vec{r},t_s) \quad \text{et} \quad h_0^{(2)}(\vec{r},t_s) = h_N^{(2)}(\vec{r},t_s) \,.$$

Pour l'évaluation de l'intégrale sur le domaine Ω nous approximons ce dernier en le subdivisant en M éléments triangulaires similaires à ceux utilisés dans la méthode des éléments finis. Cependant, la technique est conceptuellement différente puisque la distribution de température à l'intérieur du domaine est déjà connue a priori.

Pour des raisons de temps de calcul, nous supposons que la température est uniforme sur chaque triangle. Il n'est évidemment pas nécessaire, pour cette hypothèse, d'avoir une continuité de la température entre les triangles. Ainsi, pour un triangle donné, on obtient:

$$\int_{\Omega_L(t_{s-1})} \left(G^* T(\vec{r}_o^{(L)},t_{s-1})\right) d\Omega_o = T(\vec{r}_o^{(L)},t_{s-1}) \int_{\Omega_L(t_{s-1})} G^* d\Omega_o$$

où $\vec{r}_0^{(L)}$ représente le vecteur position du centre de gravité du $L^{\text{ème}}$ triangle.

En tenant compte de cette dernière discrétisation, l'équation (2.10) prend la forme suivante:

$$\begin{aligned} \varphi(\vec{r},t_s)T(\vec{r},t_s) = &\sum_{j=1}^{N} \left(\alpha \left(g_j^{(1)}(\vec{r},t_s) + g_{j-1}^{(2)}(\vec{r},t_s)\right) \frac{\partial T(\vec{r}_o^{\,j},t_s)}{\partial n}\right) \\ &- \sum_{j=1}^{N} \left(\alpha \left(h_j^{(1)}(\vec{r},t_s) + h_{j-1}^{(2)}(\vec{r},t_s)\right) T(\vec{r}_o^{\,j},t_s)\right) \qquad (2.12) \\ &+ \sum_{L=1}^{M} \left(T(\vec{r}_o^{(L)},t_{s-1}) b_L(\vec{r},t_{s-1})\right) \end{aligned}$$

où

$$b_L(\vec{r}, t_{s-1}) = \int_{\Omega_L(t_{s-1})} G^* d\Omega_o$$
$$= \int_{\Omega_L(t_{s-1})} \left(\frac{1}{4\pi a \Delta t_s} \exp\left(\frac{-R^2}{4a\Delta t_s} \right) \right) d\Omega_o \qquad (2.13)$$

et ce, en vertu de (1.5).

L'évaluation des intégrales dans le temps et l'espace sera vue en détail à la section suivante.

2.1.3 Technique de résolution

À partir de l'équation (2.12), nous sommes en mesure d'évaluer la valeur désirée (i.e. la température ou le gradient de température) à chaque noeud de la frontière. Pour ce faire, nous réécrivons cette équation pour chacun de ces noeuds pour ainsi obtenir le système d'équations linéaires de N équations et N inconnues suivant:

$$[H]\{T_s\} = [G]\{Q_s\} + [B]\{T_{s-1}\} \qquad (2.14)$$

pour $s = 1, 2, \ldots, F$ (t_F étant le temps final);

où les composantes des matrices sont données par:

$$H_{i,j} = \alpha \left(h_j^{(1)}(\vec{r}^i, t_s) + h_{j-1}^{(2)}(\vec{r}^i, t_s) \right) + \varphi(\vec{r}^i)\delta_{i,j} \; ;$$

($\delta_{i,j}$: symbole de Kronecker)

$$G_{i,j} = \alpha \left(g_j^{(1)}(\vec{r}^i, t_s) + g_{j-1}^{(2)}(\vec{r}^i, t_s) \right) \; ;$$

$$B_{i,L} = b_L(\vec{r}^i, t_{s-1}) \; ; \; (i, j = 1, 2, \ldots, N \; . \; L = 1, 2, \ldots, M)$$

avec $\vec{r}^{\,i}$ représentant le vecteur position du noeud (i) de la frontière où l'on désire connaître la température ou le gradient de température, communément appelé le vecteur position de l'observateur.

De même, les composantes des vecteurs colonnes sont données par:

$$\{T_s\}^T = \left[T(\vec{r}_o^{\,1}, t_s)\ \ T(\vec{r}_o^{\,2}, t_s)\ \ \ldots\ \ T(\vec{r}_o^{\,N}, t_s)\right] ;$$

$$\{Q_s\}^T = \left[\frac{\partial T(\vec{r}_o^{\,1}, t_s)}{\partial n}\ \ \frac{\partial T(\vec{r}_o^{\,2}, t_s)}{\partial n}\ \ \ldots\ \ \frac{\partial T(\vec{r}_o^{\,N}, t_s)}{\partial n}\right] ;$$

$$\{T_{s-1}\}^T = \left[T(\vec{r}_o^{\,(1)}, t_{s-1})\ \ T(\vec{r}_o^{\,(2)}, t_{s-1})\ \ \ldots\ \ T(\vec{r}_o^{\,(M)}, t_{s-1})\right] .$$

À noter que $T(\vec{r}_0^{\,i}, t_s)$ représente la température au noeud (i) de la frontière au temps t_s, tandis que $T(\vec{r}_0^{\,(L)}, t_{s-1})$ est la température au noeud (L) du domaine (i.e. au centre de gravité du $L^{ème}$ triangle constituant le domaine Ω) au temps t_{s-1}.

Pour solutionner le système d'équations linéaires (2.14), il suffit de regrouper les inconnues à gauche et les valeurs connues à droite et d'utiliser la méthode la plus appropriée à la résolution. Toutefois, il faut remarquer que, pour le premier pas de temps (i.e. à s = 1), les températures correspondant aux centres de gravité de chacun des triangles sont déjà connues par la condition initiale (1.2). Par contre, pour s = 2, 3, ..., F, nous devons au temps t_s, une fois le système d'équations (2.14) solutionné, calculer une nouvelle distribution de température

correspondant aux centres de gravité des triangles en utilisant l'équation (2.12), et prendre cette nouvelle distribution comme condition initiale pour le prochain pas de temps ($\Delta t_{s+1} = t_{s+1} - t_s$).

En utilisant la technique du pas de temps où chaque pas de temps est traité comme un nouveau problème, il est possible de séparer la simulation du problème d'ablation en deux parties, i.e.

1. La phase de préfusion
2. la phase de fusion

La phase de préfusion est la phase où il n'y a que de la conduction, elle se termine lorsqu'au moins un des points situé à la frontière a atteint la température de fusion. Par contre, la phase de fusion représente celle qui englobe les processus simultanés de conduction et d'ablation du solide; elle suit évidemment la phase de préfusion.

Que l'on soit dans la phase de préfusion ou celle de fusion, le système d'équations (2.14) doit être solutionné. Cependant, la façon de résoudre ce système diffère suivant que l'on se trouve dans l'une ou l'autre de ces phases. En effet, il est facile de constater que les matrices [H], [G] et [B] dépendent de la géométrie du domaine, de la diffusivité thermique et aussi du pas de temps. Par conséquent, puisque la diffusivité est constante, et que pour la phase de préfusion, la géométrie du domaine demeure fixe, il s'en suit qu'en prenant un pas de temps constant pour cette phase, les matrices [H], [G] et [B] n'ont besoin

d'être calculées qu'une seule fois au début. Il suffit donc de les conserver en mémoire si elles sont de petites tailles ou de les emmagasiner sur disque si elles sont de grandes tailles. Par conséquent, pour solutionner le système d'équations (2.14) lors de la phase de préfusion, nous n'avons qu'à inverser la matrice [H] une seule fois au début (les valeurs inconnues étant les températures prises aux noeuds de la frontière) et effectuer des produits matriciels à tous les pas de temps jusqu'à ce qu'au moins un des noeuds de la frontière ait atteint la température de fusion.

Pour la phase de fusion, puisqu'il y a déformation du domaine, les matrices [H], [G] et [B] doivent être recalculées à tous les pas de temps jusqu'à ce que le solide soit complètement fondu. Nous devons donc, à chaque pas de temps, solutionner le système d'équations (2.14) en utilisant la méthode d'élimination de Gauss.

2.1.4 Calcul du déplacement de la frontière

Pour le calcul du déplacement de la frontière lors de la phase de fusion, nous devons utiliser une technique toute particulière. Premièrement, pour le premier pas de temps suivant celui où la température de fusion a été atteint au noeud (i) de la frontière, nous supposons qu'il n'y a pas encore eu de déplacement de ce noeud. Par la suite, afin de trouver la nouvelle position du noeud (i) pour le prochain pas de temps (Δt_{s+1}) (il n'y a pas de déplacement à l'intérieur d'un même pas de temps

par hypothèse) nous commençons par calculer le gradient de température ($\partial T(\vec{r}_0^{\,i}, t_s)/\partial n$) au noeud (i) et au temps t_s (la température étant déjà connue et égale à la température de fusion à ce noeud) en solutionnant le système d'équations (2.14). Ensuite, nous substituons la valeur du gradient ainsi trouvée dans la condition frontière de Stéfan donnée par (1.3) mais écrite entièrement en termes de quantités nodales, tel que développé par O'Neill (1983), i.e.:

$$\Delta \vec{S}(\vec{r}_o^{\,i}) = \frac{\Delta t_s}{\rho L}\left(k\frac{\partial T(\vec{r}_o^{\,i}, t_s)}{\partial n} + q(\vec{r}_o^{\,i}, t_s)\right)\vec{m}^i \qquad (2.15)$$

où

$$\vec{m}^i = \frac{(\vec{n}_{i-1}l_{i-1} + \vec{n}_i l_i)}{\|\vec{n}_{i-1}l_{i-1} + \vec{n}_i l_i\|}.$$

Les termes ℓ_{i-1} et ℓ_i représentent respectivement les longueurs des éléments Γ_{i-1} et Γ_i (voir fig. 2.2).

Il ne reste plus qu'à déplacer le noeud (i) d'une distance égale au module de $\Delta\vec{S}(\vec{r}_0^{\,i})$ et suivant la direction du vecteur $\vec{m}^i$ mais de sens inverse (la valeur de $q(\vec{r}_0^{\,i}, t_s)$ étant par définition négative et supérieure en valeur absolue à la valeur de $k\partial T(\vec{r}_0^{\,i}, t_s)/\partial n$). La partie ainsi fondue est, par hypothèse, enlevée immédiatement.

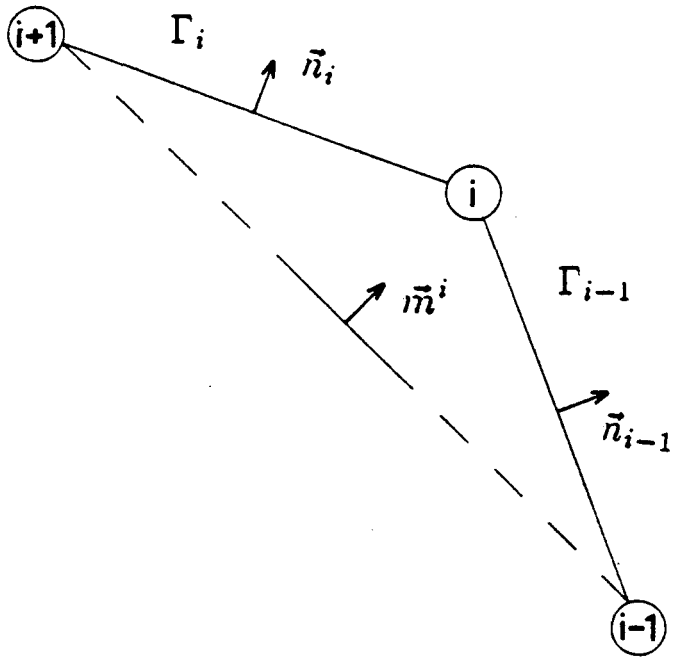


Figure 2.2: Portion de la frontière discrétisée montrant les éléments Γ_{i-1} et Γ_i avec leurs vecteurs normaux unitaires respectifs et aussi le vecteur nodal unitaire $\vec{m}^i$ donnant la direction du déplacement du noeud (i).

Puisque le domaine se déforme en fonction du temps, il est nécessaire de développer un algorithme pouvant discrétiser automatiquement le domaine en éléments triangulaires. Le développement de cet algorithme sera expliqué en détails au chapitre III.

2.2 Évaluation des intégrales

L'évaluation des intégrales de contour, de même que celle de l'intégrale sur le domaine Ω requièrent une attention toute particulière. En effet, dans les intégrales de contour apparaissent certaines singularités. Nous verrons, dans la première partie de cette section, comment traiter ces singularités. Pour l'intégration sur le domaine exposée à la seconde partie, nous montrerons, en utilisant la transformation en coordonnées polaires, le moyen d'éviter les problèmes d'intégration numérique occasionnés par l'utilisation de la méthode directe dite de Hammer. Cette méthode distribue symétriquement les points d'intégration à l'intérieur de chaque triangle et son utilisation peut causer des problèmes numériques si le terme $\alpha\Delta t_s$ est très petit comparativement aux dimensions géométriques du problème (Wrobel et al., 1981a).

2.2.1 Intégration sur la frontière

Avant d'évaluer les intégrales de contour, nous devons, en tout premier lieu, intégrer par rapport au temps les intégrales définies en (2.3). Comme il a été mentionné à la section précédente, il est possible d'évaluer analytiquement ces intégrales. En effet, en posant

$$u = \frac{R^2}{4\alpha(t_s - \tau)} \quad , où\ R = \|\vec{R}\| = \|\vec{r}_o - \vec{r}\| \ ,$$

nous obtenons premièrement

$$\begin{aligned}\int_{t_{s-1}}^{t_s} G^* d\tau &= \int_{t_{s-1}}^{t_s} \left(\frac{1}{4\pi a(t_s - \tau)} \exp\left(\frac{-R^2}{4a(t_s - \tau)} \right) \right) d\tau \\ &= \frac{1}{4\pi a} \int_A^\infty \frac{\exp(-u)}{u} du \\ &= \frac{1}{4\pi a} E_1(A) \end{aligned} \tag{2.16}$$

où

$$A = \frac{R^2}{4a\Delta t_s} \quad \text{avec} \quad \Delta t_s = t_s - t_{s-1} .$$

$E_1(A)$ est appelée intégrale exponentielle et se développe en série absolument convergente de la façon suivante (Abramowitz et al., 1965):

$$E_1(A) = -\gamma - \ln(A) - \sum_{k=1}^{\infty} \left(\frac{(-A)^k}{k \cdot k!} \right) \tag{2.17}$$

avec γ représentant la constante d'Euler.

Ensuite,

$$\begin{aligned}\int_{t_{s-1}}^{t_s} \frac{\partial G^*}{\partial n} d\tau &= \int_{t_{s-1}}^{t_s} (\nabla G^* \cdot \vec{n})\, d\tau \\ &= \int_{t_{s-1}}^{t_s} \left(\frac{\partial G^*}{\partial R} \frac{\vec{R} \cdot \vec{n}}{R} \right) d\tau \\ &= \int_{t_{s-1}}^{t_s} \left(\frac{-R}{8\pi a^2 (t_s - \tau)^2} \exp\left(\frac{-R^2}{4a(t_s - \tau)} \right) \frac{\partial R}{\partial n} \right) d\tau \\ &= \frac{-1}{2\pi a R} \frac{\partial R}{\partial n} \int_A^\infty \exp(-u) du \\ &= \frac{-1}{2\pi a R} \frac{\partial R}{\partial n} \exp(-A) . \end{aligned} \tag{2.18}$$

Comme étape suivante, nous devons évaluer les intégrales de contour. Cependant, ces intégrales, définies en (2.9) et (2.10), contiennent des singularités lorsque R devient nul (voir les équations (2.16) et (2.18)). Nous allons tout d'abord évaluer ces intégrales en considérant les singularités pour ensuite les évaluer lorsqu'il y a absence de celles-ci.

2.2.1.1 Traitement des singularités

Puisque la frontière est discrétisée en éléments linéaires, les singularités (i.e. R = 0) surviennent seulement lorsque nous intégrons sur l'élément précédant ou suivant le noeud (i) de la frontière où se trouve l'observateur (voir fig. 2.3).

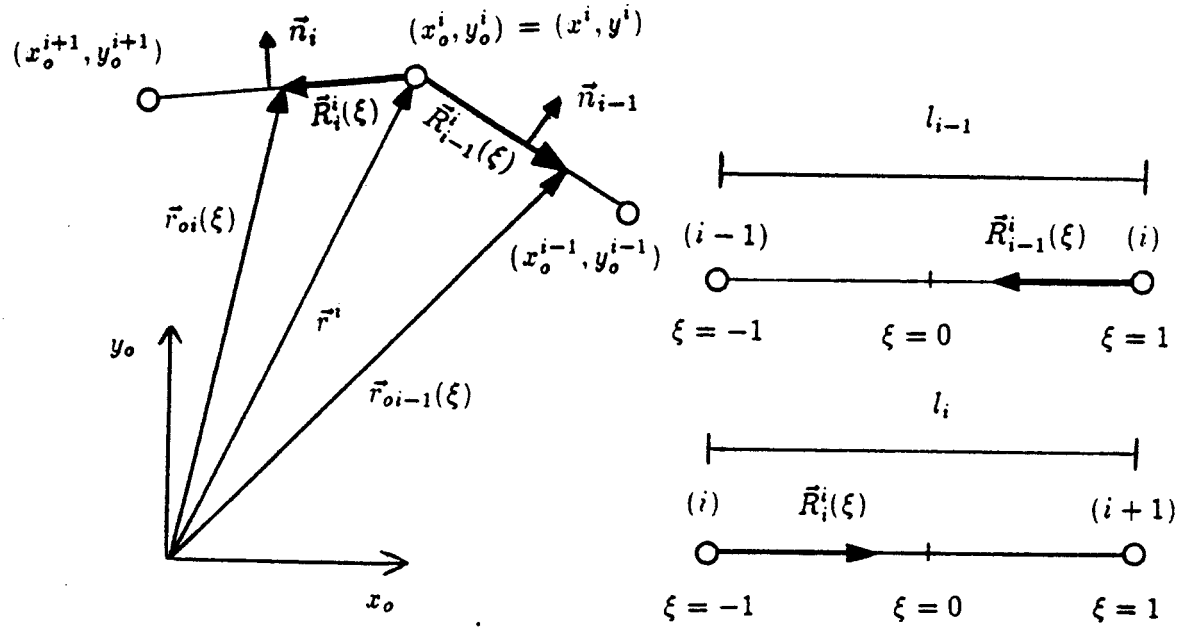


Figure 2.3: Définitions utiles à l'intégration analytique des intégrales de contour lorsqu'il y a présence de singularités.

Pour l'intégrale définie en (2.9), la singularité est de type logarithmique, laquelle est directement intégrable. En effet, pour j = i-1:

$$\begin{aligned} g_{i-1}^{(m)}(\vec{r}^{\,i}, t_s) &= \int_{\Gamma_{i-1}} \psi_{(m)} \int_{t_{s-1}}^{t_s} G^* d\tau \, d\Gamma_o \\ &= \int_{\Gamma_{i-1}} \left(\frac{\psi_{(m)}}{4\pi\alpha} E_1 \left(\frac{\left(R_{i-1}^{i}(\xi)\right)^2}{4\alpha\Delta t_s} \right) \right) d\Gamma_o \end{aligned} \tag{2.19}$$

où

$$R_{i-1}^{i}(\xi) = \|\vec{r}_{oi-1}(\xi) - \vec{r}^{\,i}\| = \left(\frac{1-\xi}{2}\right) l_{i-1}$$

et

$$l_j = \|\vec{r}_o^{\,j+1} - \vec{r}_o^{\,j}\| = \left((x_o^{j+1} - x_o^j)^2 + (y_o^{j+1} - y_o^j)^2\right)^{1/2} .$$

Puisque $\vec{r}_{oj}(\xi)$ est une fonction vectorielle qui paramétrise l'élément linéaire Γ_j (pour j = 1, 2, ...N), telle que définie en (2.7), l'intégrale (2.19) prend la forme suivante:

$$g_{i-1}^{(m)}(\vec{r}^{\,i}, t_s) = \int_{-1}^{1} \left(\frac{\psi_{(m)}}{4\pi\alpha} E_1 \left(\frac{\left(\left(\frac{1-\xi}{2}\right) l_{i-1}\right)^2}{4\alpha\Delta t_s} \right) \frac{l_{i-1}}{2} \right) d\xi \tag{2.20}$$

En posant $v = \left(\frac{1-\xi}{2}\right)\ell_{i-1}$ dans cette dernière intégrale, nous obtenons pour m = 1:

$$\begin{aligned} g_{i-1}^{(1)}(\vec{r}^{\,i}, t_s) &= \frac{l_{i-1}}{8\pi\alpha} \left(\frac{2}{l_{i-1}^2} \int_0^{l_{i-1}} \left(v E_1 \left(\frac{v^2}{4\alpha\Delta t_s} \right) \right) dv \right) \\ &= \frac{l_{i-1}}{8\pi\alpha} \left(1 - \gamma - \ln(a_{i-1}) - \sum_{k=1}^{\infty} \left(\frac{(-a_{i-1})^k}{k(k+1)!} \right) \right) \end{aligned} \tag{2.21}$$

où

$$a_{i-1} = \frac{l_{i-1}^2}{4\alpha\Delta t_s} .$$

En adoptant le même changement de variable pour m = 2, l'intégrale (2.20) devient:

$$\begin{aligned} g_{i-1}^{(2)}(\vec{r}^i, t_s) &= \frac{l_{i-1}}{8\pi\alpha}\left(\frac{2}{l_{i-1}^2}\int_0^{l_{i-1}}\left((-v+l_{i-1})E_1\left(\frac{v^2}{4\alpha\Delta t_s}\right)\right)dv\right) \\ &= \frac{l_{i-1}}{8\pi\alpha}\left(3-\gamma-\ln(a_{i-1})-\sum_{k=1}^{\infty}\left(\frac{(-a_{i-1})^k}{k(2k+1)(k+1)!}\right)\right) \end{aligned} \tag{2.22}$$

De façon similaire pour j = i, on obtient:

$$\begin{aligned} g_i^{(m)}(\vec{r}^i, t_s) &= \int_{\Gamma_i}\left(\frac{\psi_{(m)}}{4\pi\alpha}E_1\left(\frac{\left(R_i^i(\xi)\right)^2}{4\alpha\Delta t_s}\right)\right)d\Gamma_o \\ &= \int_{-1}^{1}\left(\frac{\psi_{(m)}}{4\pi\alpha}E_1\left(\frac{\left(\left(\frac{1+\xi}{2}\right)l_i\right)^2}{4\alpha\Delta t_s}\right)\frac{l_i}{2}\right)d\xi \end{aligned} \tag{2.23}$$

Pour cette dernière intégrale, le changement de variable choisi est $v = \left(\frac{1+\xi}{2}\right)\ell_i$. Ainsi, pour m = 1:

$$\begin{aligned} g_i^{(1)}(\vec{r}^i, t_s) &= \frac{l_i}{8\pi\alpha}\left(\frac{2}{l_i^2}\int_0^{l_i}\left((-v+l_i)E_1\left(\frac{v^2}{4\alpha\Delta t_s}\right)\right)dv\right) \\ &= \frac{l_i}{8\pi\alpha}\left(3-\gamma-\ln(a_i)-\sum_{k=1}^{\infty}\left(\frac{(-a_i)^k}{k(2k+1)(k+1)!}\right)\right) \end{aligned} \tag{2.24}$$

où

$$a_i = \frac{l_i^2}{4\alpha\Delta t_s} .$$

De même, pour m = 2:

$$g_i^{(2)}(\vec{r}', t_s) = \frac{l_i}{8\pi a}\left(\frac{2}{l_i^2}\int_0^{l_i}\left(vE_1\left(\frac{v^2}{4a\Delta t_s}\right)\right)dv\right) \qquad (2.25)$$
$$= \frac{l_i}{8\pi a}\left(1-\gamma-\ln(a_{i-1})-\sum_{k=1}^{\infty}\left(\frac{(-a_{i-1})^k}{k(k+1)!}\right)\right)$$

Lorsque les valeurs de a_{i-1} et a_i sont inférieures ou égales à l'unité, la convergence des séries (2.21), (2.22), (2.24) et (2.25) est garantie avec un maximum de six termes (le degré de précision étant de l'ordre de 10^{-6}). Si par contre a_{i-1} et a_i sont supérieures à l'unité, la convergence devient plus lente et peut même ne jamais être atteinte dépendamment de la capacité numérique de l'ordinateur. Pour remédier à cela, il suffit, premièrement, d'intégrer analytiquement sur des petits segments $d_{i-1}^{(1)}$ et $d_i^{(1)}$ qui appartiennent respectivement aux éléments Γ_{i-1} et Γ_i et qui possèdent les deux propriétés suivantes (voir fig. 2.4):

1) l'une des extrémités de chacun de ces segments correspond au noeud (i)

2) $$\frac{\left(d_{i-1}^{(1)}\right)^2}{4a\Delta t_s} = \frac{\left(d_i^{(1)}\right)^2}{4a\Delta t_s} = 1\ ;$$

ensuite d'intégrer numériquement sur le reste des éléments Γ_{i-1} et Γ_i, c'est-à-dire sur les segments $d_{i-1}^{(2)}$ et $d_i^{(2)}$.

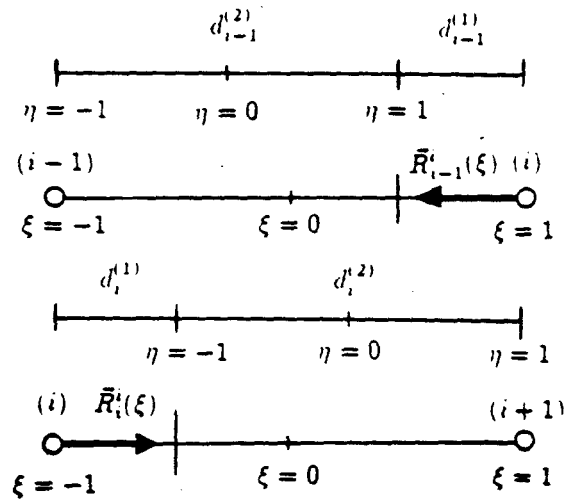


Figure 2.4: Division des éléments Γ_{i-1} et Γ_i en deux parties de manière à ce que $\dfrac{(d_{i-1}^{(1)})^2}{4\alpha\Delta t_s} = \dfrac{(d_i^{(1)})^2}{4\alpha\Delta t_s} = 1$.

En procédant ainsi, nous devons évidemment changer la façon d'évaluer les intégrales définies en (2.20) et (2.23). Nous allons tout d'abord évaluer l'intégrale (2.20).

La valeur de ξ pour laquelle $R_{i-1}^{i}(\xi) = d_{i-1}^{(1)}$ est donnée par:

$$\xi_{i-1} = 1 - \frac{2d_{i-1}^{(1)}}{l_{i-1}} = \frac{d_{i-1}^{(2)} - d_{i-1}^{(1)}}{l_{i-1}}$$

puisque

$$R_{i-1}^{i}(\xi) = \left(\frac{1-\xi}{2}\right) l_{i-1} \quad \text{et} \quad d_{i-1}^{(1)} + d_{i-1}^{(2)} = l_{i-1} \,.$$

Par conséquent, il suffit de séparer l'intégrale (2.20) en deux parties de la façon suivante:

$$g_{i-1}^{(m)}(\vec{r}^{\,i}, t_s) = g_{i-1}^{(m)}(\vec{r}^{\,i}, t_s)_{(1)} + g_{i-1}^{(m)}(\vec{r}^{\,i}, t_s)_{(2)}$$

où

$$g_{i-1}^{(m)}(\vec{r}^{\,i},t_s)_{(1)} = \int_{-1}^{\xi_{i-1}} \left(\frac{\psi_{(m)}}{4\pi\alpha} E_1 \left(\frac{\left(\left(\frac{1-\xi}{2}\right) l_{i-1}\right)^2}{4\alpha\Delta t_s} \right) \frac{l_{i-1}}{2} \right) d\xi \qquad (2.26)$$

et

$$g_{i-1}^{(m)}(\vec{r}^{\,i},t_s)_{(2)} = \int_{\xi_{i-1}}^{1} \left(\frac{\psi_{(m)}}{4\pi\alpha} E_1 \left(\frac{\left(\left(\frac{1-\xi}{2}\right) l_{i-1}\right)^2}{4\alpha\Delta t_s} \right) \frac{l_{i-1}}{2} \right) d\xi \,. \qquad (2.27)$$

Puisque la singularité survient lorsque $\xi = 1$, c'est l'intégrale (2.27) qui doit être évaluée analytiquement. Par conséquent, en posant $v = \left(\frac{1-\xi}{2}\right)\ell_{i-1}$, cette dernière intégrale devient, pour m = 1

$$\begin{aligned} g_{i-1}^{(1)}(\vec{r}^{\,i},t_s)_{(2)} &= \frac{l_{i-1}}{8\pi\alpha} \left(\frac{2}{l_{i-1}^2} \int_0^{d_{i-1}^{(1)}} \left(v E_1 \left(\frac{v^2}{4\alpha\Delta t_s} \right) \right) dv \right) \\ &= \frac{\left(d_{i-1}^{(1)}\right)^2}{8\pi\alpha l_{i-1}} \left(1 - \gamma - \ln\left(\frac{\left(d_{i-1}^{(1)}\right)^2}{4\alpha\Delta t_s} \right) \right) \\ &\quad - \frac{\left(d_{i-1}^{(1)}\right)^2}{8\pi\alpha l_{i-1}} \left(\sum_{k=1}^{\infty} \left(\left(\frac{-\left(d_{i-1}^{(1)}\right)^2}{4\alpha\Delta t_s} \right)^k \frac{1}{k(k+1)!} \right) \right) . \end{aligned} \qquad (2.28)$$

Puisque nous avons choisi $d_{i-1}^{(1)}$ pour que

$$\frac{\left(d_{i-1}^{(1)}\right)^2}{4\alpha\Delta t_s} = 1$$

l'équation (2.28) se réécrit de la façon suivante:

$$g_{i-1}^{(1)}(\vec{r}^i, t_s)_{(2)} = \frac{\left(d_{i-1}^{(1)}\right)^2}{8\pi\alpha l_{i-1}} \left(1 - \gamma - \sum_{k=1}^{\infty} \left(\frac{(-1)^k}{k(k+1)!}\right)\right) . \qquad (2.29)$$

De même pour m = 2

$$\begin{aligned} g_{i-1}^{(2)}(\vec{r}^i, t_s)_{(2)} &= \frac{l_{i-1}}{8\pi\alpha} \left(\frac{2}{l_{i-1}^2} \int_0^{d_{i-1}^{(1)}} \left((-v + l_{i-1}) E_1 \left(\frac{v^2}{4\alpha\Delta t_s}\right)\right) dv\right) \\ &= \frac{d_{i-1}^{(1)}}{8\pi\alpha l_{i-1}} \left(-\gamma \left(l_{i-1} + d_{i-1}^{(2)}\right) - d_{i-1}^{(1)} + 4l_{i-1}\right) \\ &\quad - \frac{d_{i-1}^{(1)}}{8\pi\alpha l_{i-1}} \left(\sum_{k=1}^{\infty} \left(\frac{(-1)^k \left((2k+1) d_{i-1}^{(2)} + l_{i-1}\right)}{k(2k+1)(k+1)!}\right)\right) . \end{aligned} \qquad (2.30)$$

Nous devons maintenant intégrer numériquement l'intégrale définie en (2.26) en ramenant ses bornes d'intégration de -1 à 1. Pour ce faire, nous posons:

$$\xi(\eta) = \frac{\eta d_{i-1}^{(2)} - d_{i-1}^{(1)}}{l_{i-1}} .$$

Par conséquent, l'intégrale (2.26) devient:

$$g_{i-1}^{(m)}(\vec{r}^{\,i}, t_s)_{(1)} = \int_{-1}^{1} \left(\frac{\psi_{(m)}}{4\pi a} E_1(A_{i-1}) \frac{d_{i-1}^{(2)}}{2} \right) d\eta$$
$$= \frac{d_{i-1}^{(2)}}{8\pi a} \int_{-1}^{1} \left(\psi_{(m)} \int_{A_{i-1}}^{\infty} \frac{\exp(-u)}{u} du \right) d\eta \qquad (2.31)$$

où

$$A_{i-1} = \frac{\left(\left(\frac{1-\xi(\eta)}{2} \right) l_{i-1} \right)^2}{4\pi a \Delta t_s} .$$

Cependant, pour évaluer l'intégrale exponentielle apparaissant dans l'équation (2.31), nous ne pouvons pas utiliser son développement en série. En effet, puisque $A_{i-1} > 1$, la convergence de cette série peut être très lente comme il a été déjà mentionné plus tôt dans cette section. Pour y remédier, il suffit de poser $z = u - A_{i-1}$ dans l'intégrale exponentielle pour ramener les bornes d'intégration de 0 à l'infini. En procédant ainsi, l'équation (2.31) se réécrit:

$$g_{i-1}^{(m)}(\vec{r}^{\,i}, t_s)_{(1)} = \frac{d_{i-1}^{(2)}}{8\pi a} \int_{-1}^{1} \left(\psi_{(m)} \exp(-A_{i-1}) \int_{0}^{\infty} \frac{\exp(-z)}{z + A_{i-1}} dz \right) d\eta . \qquad (2.32)$$

Pour évaluer cette dernière intégrale, nous devons intégrer numériquement, en premier lieu, par rapport à z en utilisant la quadrature de Gauss-Laguerre ensuite par rapport à η en utilisant cette fois-ci la quadrature de Gauss-Legendre. Les nombres de points d'intégration

nécessaires à l'évaluation numérique de ces intégrales seront donnés au chapitre III.

De façon similaire, l'intégrale (2.23) doit être également séparée en deux parties pour que

$$\frac{\left(d_i^{(1)}\right)^2}{4\alpha\Delta t_s} = 1 .$$

La valeur de ξ pour avoir $R_i^i(\xi) = d_i^{(1)}$ est donnée par:

$$\xi_i = \frac{2d_i^{(1)}}{l_i} - 1 = \frac{d_i^{(1)} - d_i^{(2)}}{l_i}$$

puisque

$$R_i^i(\xi) = \left(\frac{1+\xi}{2}\right) l_i .$$

Par conséquent, l'intégrale (2.23) devient:

$$g_i^{(m)}(\vec{r}^i, t_s) = g_i^{(m)}(\vec{r}^i, t_s)_{(1)} + g_i^{(m)}(\vec{r}^i, t_s)_{(2)}$$

où

$$g_i^{(m)}(\vec{r}^i, t_s)_{(1)} = \int_{-1}^{\xi_i} \left(\frac{\upsilon^{(m)}}{4\pi\alpha} E_1 \left(\frac{\left(\left(\frac{1+\xi}{2}\right) l_i\right)^2}{4\alpha\Delta t_s} \right) \frac{l_i}{2} \right) d\xi \qquad (2.33)$$

et

$$g_i^{(m)}(\vec{r}^i, t_s)_{(2)} = \int_{\xi_i}^{1} \left(\frac{\psi_{(m)}}{4\pi\alpha} E_1 \left(\frac{\left(\left(\frac{1+\xi}{2} \right) l_i \right)^2}{4\alpha \Delta t_s} \right) \frac{l_i}{2} \right) d\xi \,. \tag{2.34}$$

Contrairement à l'intégrale (2.20), la singularité survient dans l'intégrale (2.23) lorsque $\xi = -1$. Par conséquent, l'intégrale (2.33) doit être évaluée analytiquement en posant $v = \left(\frac{1 + \xi}{2} \right) \ell_i$. Ainsi, nous obtenons pour m = 1,

$$\begin{aligned} g_i^{(1)}(\vec{r}^i, t_s)_{(1)} &= \frac{l_i}{8\pi\alpha} \left(\frac{2}{l_i^2} \int_0^{d_i^{(1)}} \left((-v + l_i) E_1 \left(\frac{v^2}{4\alpha \Delta t_s} \right) \right) dv \right) \\ &= \frac{d_i^{(1)}}{8\pi\alpha l_i} \left(-\gamma \left(l_i + d_i^{(2)} \right) - d_i^{(1)} + 4 l_i \right) \\ &\quad - \frac{d_i^{(1)}}{8\pi\alpha l_i} \left(\sum_{k=1}^{\infty} \left(\frac{(-1)^k \left((2k+1) d_i^{(2)} + l_i \right)}{k(2k+1)(k+1)!} \right) \right) . \end{aligned} \tag{2.35}$$

Dans le cas où m = 2,

$$\begin{aligned} g_i^{(2)}(\vec{r}^i, t_s)_{(1)} &= \frac{l_i}{8\pi\alpha} \left(\frac{2}{l_i^2} \int_0^{d_i^{(1)}} \left(v E_1 \left(\frac{v^2}{4\alpha \Delta t_s} \right) \right) dv \right) \\ &= \frac{\left(d_i^{(1)} \right)^2}{8\pi\alpha l_{i-1}} \left(1 - \gamma - \sum_{k=1}^{\infty} \left(\frac{(-1)^k}{k(k+1)!} \right) \right) . \end{aligned} \tag{2.36}$$

Pour l'évaluation de l'intégrale (2.34), nous procédons de la même façon que celle définie en (2.26) sauf que les changements de variable sont dans l'ordre,

$$z = u - A_i \ . \quad \text{où} \quad A_i = \frac{\left(\left(\frac{1+\xi(\nu)}{2}\right) l_i\right)^2}{4\pi\alpha\Delta t_s} \ .$$

pour pouvoir intégrer numériquement l'intégrale exponentielle $E_1(A_i)$ par la méthode de Gauss-Laguerre, et

$$\xi(\nu) = \frac{\nu d_i^{(2)} + d_i^{(1)}}{l_i}$$

pour permettre l'intégration par rapport à ν par la méthode de Gauss-Legendre. En effectuant ces deux changements de variable, l'intégrale (2.34) prend la forme suivante:

$$g_i^{(m)}(\vec{r}^{\,i}, t_s)_{(2)} = \frac{d_i^{(2)}}{8\pi\alpha} \int_{-1}^{1} \left(\psi_{(m)} \exp(-A_i) \int_0^{\infty} \frac{\exp(-z)}{z + A_i} dz \right) d\nu \qquad (2.37)$$

Maintenant que la singularité qui apparaît dans l'intégrale (2.9) a été traitée, il nous reste celle apparaissant dans l'intégrale (2.10). Cette dernière est de type 1/R (voir l'équation 2.18). Cependant, puisque

$$h_j^{(m)}(\vec{r}^{\,i}, t_s) = \frac{-1}{2\pi\alpha} \int_{\Gamma_j} \left(\frac{\psi_{(m)}}{R_j^i(\xi)} \frac{\partial R_j^i(\xi)}{\partial n_j} \exp\left(\frac{-\left(R_j^i(\xi)\right)^2}{4\alpha\Delta t_s} \right) \right) d\Gamma_o$$

où

$$R_j^i(\xi) = \left\| \vec{r}_{oj}(\xi) - \vec{r}^{\,i} \right\| \ , \quad \vec{n}_j = \frac{(y_o^{j+1} - y_o^j \ , \ x_o^j - x_o^{j+1})}{l_j}$$

pour

$$i,j = 1,2,\ldots,N \quad \text{avec} \quad x_o^{N+1} = x_o^1 \text{ et } y_o^{N+1} = y_o^1 \ )$$

et que, pour $\xi \in [-1, 1]$ (voir fig. 2.3),

$$\frac{\partial R_{i-1}^i}{\partial n_{i-1}} = \nabla \left(R_{i-1}^i(\xi) \right) \cdot \vec{n}_{i-1}$$

$$= \frac{\vec{R}_{i-1}^i(\xi)}{R_{i-1}^i(\xi)} \cdot \vec{n}_{i-1} = 0$$

de même que

$$\frac{\partial R_i^i}{\partial n_i} = \frac{\vec{R}_i^i}{R_i^i(\xi)} \cdot \vec{n}_i = 0$$

il s'ensuit que

$$h_{i-1}^{(m)}(\vec{r}^i, t_s) = h_i^{(m)}(\vec{r}^i, t_s) = 0 \ .$$

2.2.1.2 Évaluation des intégrales ne contenant pas de singularité

Pour terminer l'évaluation des intégrales de contour, il nous reste à réévaluer les intégrales (2.9) et (2.10) mais cette fois-ci, sans la présence de singularité, c'est-à-dire lorsque $j \neq i-1$ et $j \neq i$.

D'abord, pour l'évaluation de l'intégrale (2.9), nous devons faire le changement de variable

$$z = u - A_j \quad \text{avec} \quad A_j = \frac{(R_j^i(\xi))^2}{4\alpha\Delta t_s}$$

pour permettre d'intégrer numériquement l'intégrale exponentielle $E_1(A_j)$ par la méthode de Gauss-Laguerre puisque la série définie en (2.17) converge lentement lorsque A_j est supérieur à l'unité. Par conséquent, l'intégrale (2.9) devient:

$$g_j^{(m)}(\vec{r}^i, t_s) = \frac{l_j}{8\pi a} \int_{-1}^{1} \left(v_{(m)} \exp(-A_j) \int_0^{\infty} \frac{\exp(-z)}{z + A_j} dz \right) d\xi \qquad (2.38)$$

où

$$l_j = \left((x_o^{j+1} - x_o^j)^2 + (y_o^{j+1} - y_o^j)^2\right)^{1/2} ;$$

et

$$R_j^i(\xi) = \|\vec{R}_j^i(\xi)\| = \|\vec{r}_{oj}(\xi) - \vec{r}^i\| .$$

Pour évaluer numériquement l'intégrale (2.38), nous procédons exactement de la même façon que celle définie en (2.32) et (2.37).

Pour l'évaluation de l'intégrale définie en (2.10), avec évidemment $j \neq i-1$ et $j \neq i$, nous utilisons uniquement la méthode de Gauss-Legendre. En effet,

$$h_j^{(m)}(\vec{r}^i, t_s) = \frac{-1}{2\pi a} \int_{\Gamma_j} \left(\frac{\psi_{(m)}}{R_j^i(\xi)} \frac{\partial R_j^i(\xi)}{\partial n_j} \exp\left(\frac{-(R_j^i(\xi))^2}{4a\Delta t_s} \right) \right) d\Gamma_o$$

$$= \frac{-l_j}{4\pi a} \int_{-1}^{1} \left(\frac{\psi_{(m)}}{(R_j^i(\xi))^2} \left(\vec{R}_j^i(\xi) \cdot \vec{n}_j \right) \exp\left(\frac{-(R_j^i(\xi))^2}{4a\Delta t_s} \right) \right) d\xi \qquad (2.39)$$

Il faut toutefois remarquer que pour $\xi \in [-1, 1]$ (voir fig. 2.5)

$$\vec{R}_j^i(\xi) \cdot \vec{n}_j = \vec{R}^{i,j} \cdot \vec{n}_j$$

où

$$\vec{R}^{i,j} = \vec{r}_o^j - \vec{r}^i = (x_o^j - x^i \ , \ y_o^j - y^i)$$

et

$$\vec{n}_j = \frac{(y_o^{j+1} - y_o^j \ , \ x_o^j - x_o^{j+1})}{l_j} \ .$$

Par conséquent, il est possible de sortir le produit scalaire de l'intégrale (2.39) pour obtenir ainsi

$$h_j^{(m)}(\vec{r}^i, t_s) = \frac{-l_j(\vec{R}^{i,j} \cdot \vec{n}_j)}{4\pi a} \int_{-1}^{1} \left(\frac{\psi_{(m)}}{(R_j^i(\xi))^2} \exp\left(\frac{-(R_j^i(\xi))^2}{4a\Delta t_s} \right) \right) d\xi \qquad (2.40)$$

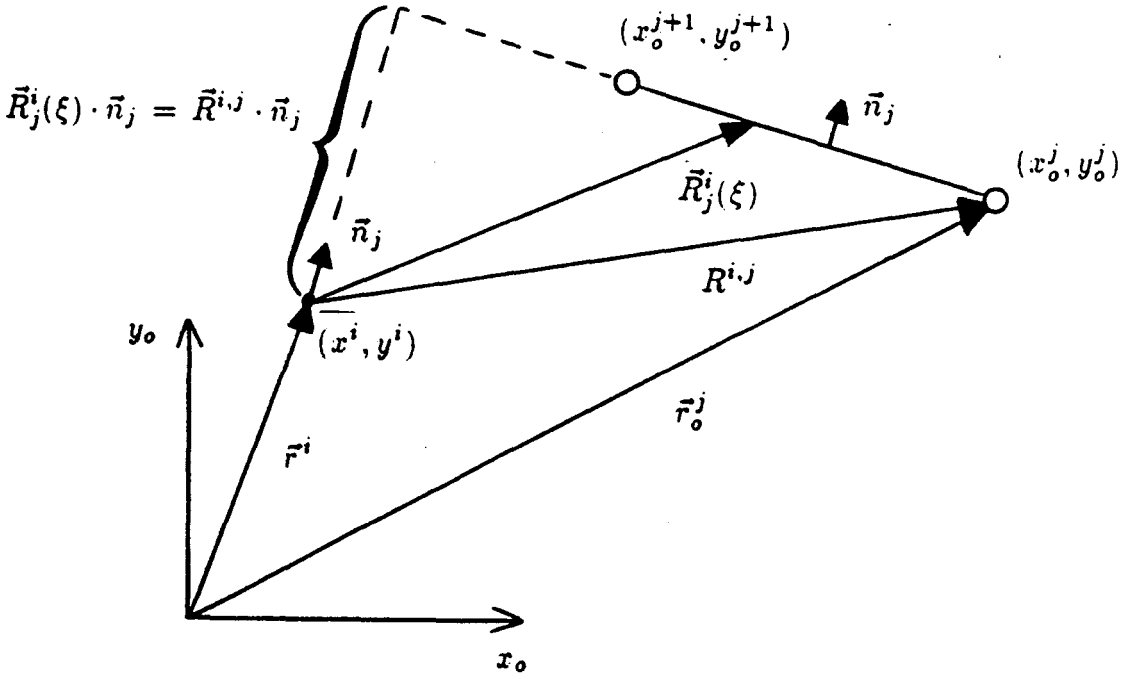


Figure 2.5: Représentation du produit scalaire.

Jusqu'à présent, nous avons évalué les intégrales de contour pour l'observateur situé à chaque noeud de la frontière. Cependant, si nous voulons évaluer la distribution de température à l'intérieur du domaine, nous devons recalculer les intégrales de contour définies en (2.9) et (2.10) mais cette fois-ci pour l'observateur situé à un des centres de gravité des triangles qui constituent le domaine. Puisque dans cette situation il ne peut y avoir de singularité à l'intérieur des intégrales de contour, nous utilisons les intégrales définies en (2.38) et (2.40) pour les évaluer numériquement. Il suffit de remplacer le vecteur position $\vec{r}^i$ par le vecteur position $\vec{r}^{(L)}$ qui est celui du centre de gravité du $L^{ème}$ triangle (L = 1, 2, ...M).

2.2.2 Intégration sur le domaine

Pour intégrer numériquement sur les éléments triangulaires, il existe une méthode directe dite de Hammer (Hammer et al., 1956). Cependant, dû à la petite diffusivité de l'aluminium ($\alpha \cong 10^{-5} m^2/s$), l'utilisation de cette méthode peut occasionner des problèmes numériques (Wrobel et al., 1981a).

La méthode alternative présentée ici, pour l'évaluation de l'intégrale définie en (2.13) est la transformation de cette dernière en coordonnées polaires (R,Θ) avec intégration analytique par rapport à R (Wrobel et al., 1981a). La variable R représente le module du vecteur position $\vec{R} = \vec{r}_0 - \vec{r}$ dans le système de coordonnées (u_0, v_0) et Θ l'angle que fait ce vecteur position avec la direction positive de l'axe u (voir fig. 2.6).

Avant de développer les intégrales en coordonnées polaires, il est primordial de bien déterminer l'emplacement de l'observateur $(\vec{r})$ au temps t_s par rapport au triangle sur lequel s'effectue l'intégration au temps t_{s-1}. En effet, si l'observateur se retrouve

1) à l'extérieur
2) sur un des sommets (2.41)
3) sur un des côtés
4) ou à l'intérieur

du triangle, nous obtenons quatre intégrales différentes à évaluer.

Si le domaine demeurait fixe lors de la simulation, il serait évidemment facile de prédire l'emplacement de l'observateur au temps t_s par rapport aux triangles approximant le domaine au temps t_{s-1}. En effet, l'observateur, de par la technique que nous utilisons, ne peut se situer qu'aux noeuds de la frontière ou aux centres de gravité des triangles et ceci au temps t_s. Cependant, lorsqu'il y a ablation, le domaine subit des déformations et la disposition des triangles et de la frontière au temps t_s n'est évidemment pas la même que celle au temps t_{s-1}. Par conséquent, nous devons développer un algorithme prédisant la position de l'observateur par rapport au triangle où s'effectue l'intégration. Le développement de cet algorithme est expliqué à l'appendice C.

La position de la source par rapport à un triangle étant déterminée, nous sommes prêts à intégrer en considérant les quatres cas présentés en (2.41).

2.2.2.1 L'observateur est situé à l'extérieur du triangle

Tout comme pour la méthode des éléments finis, les sommets, pour un triangle donné, possèdent une numérotation déterminée à l'avance de 1 à 3. Cette numérotation ne tient évidemment pas compte de l'orientation du triangle par rapport à l'observateur. Nous devons donc, avant de passer à l'intégration, refaire une renumérotation des triangles pour bien définir les bornes d'intégration en Θ. Pour commencer, nous devons déterminer le côté (représenté par $R_{L,2}(\Theta)$ à la figure 2.6) qui est balayé entièrement

par Θ seulement lorsque ce dernier a parcouru tout son domaine d'intégration représenté par l'intervalle $[\theta_1, \theta_3]$. Par convention, nous associons le chiffre 2 au sommet qui est opposé à ce côté. L'angle θ_2 est évidemment l'angle que fait le vecteur position du sommet (2), dans le système de coordonnées (u_0, v_0), avec la direction positive de u_0.

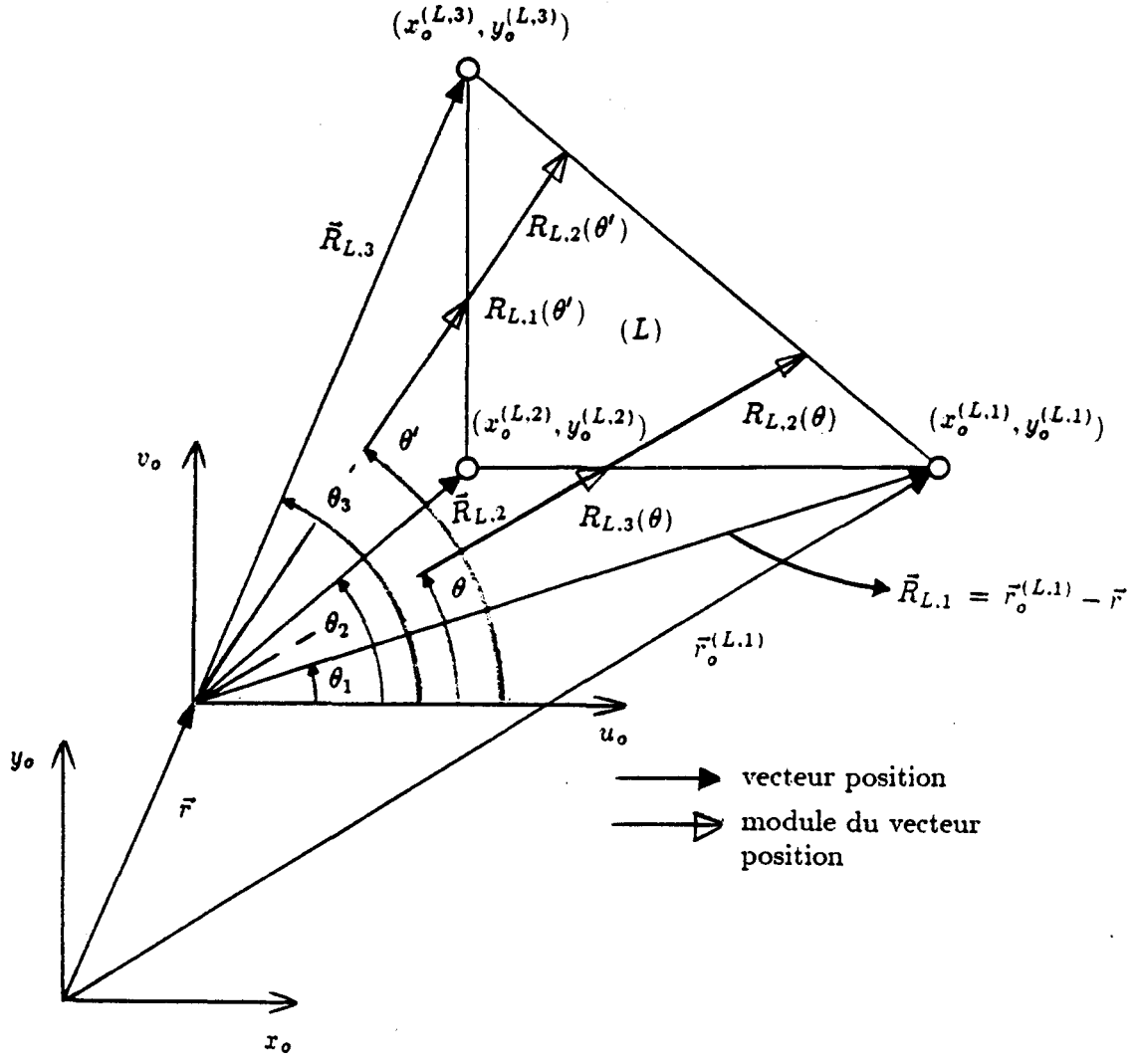
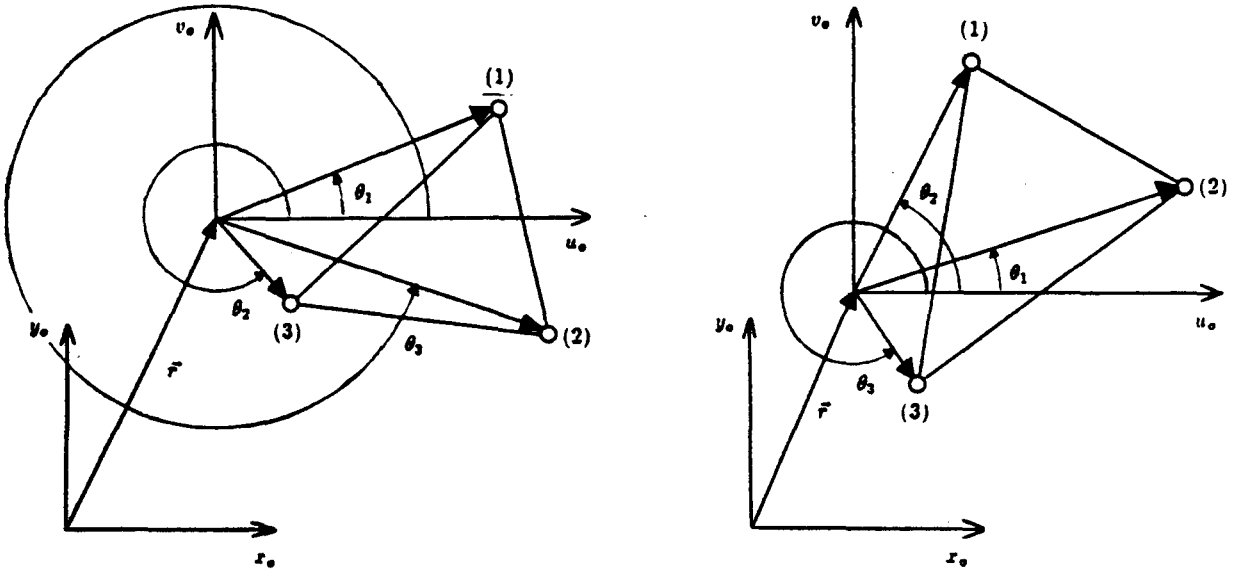


Figure 2.6: Définitions utiles pour l'intégration en coordonnées polaires avec l'observateur situé à l'extérieur du $L^{ème}$ triangle.

Par la suite, une fois l'angle θ_2 déterminé, nous trions en ordre croissant les deux angles correspondant aux deux autres sommets pour que θ puisse varier suivant le sens anti-horaire i.e. $\theta_1 \leq \theta \leq \theta_3$ (fig. 2.6). Cependant, si après avoir trié les deux angles, $\theta_3 - \theta_1 > \Pi$ (fig. 2.7) alors, θ_3 devient θ_1 et l'ancien $\theta_1 + 2\Pi$ devient θ_3. De plus, si $\theta_2 < \theta_1$ (fig. 2.7b), nous devons substituer θ_2 par $\theta_2 + 2\Pi$.



a) Cas où $\theta_2 > \theta_1$

b) Cas où $\theta_2 < \theta_1$, donc θ_2 doit être substitué par $\theta_2 + 2\Pi$.

Figure 2.7: Cas particulier où $\theta_3 - \theta_1 > \Pi$, donc θ_3 doit devenir θ_1 et l'ancien $\theta_1 + 2\Pi$ doit devenir le nouveau θ_3.

Les angles Θ_1, Θ_2 et Θ_3 étant déterminés, l'intégrale définie en (2.13) devient, pour l'observateur ($\vec{r}$ = (x, y)) situé à l'extérieur du $L^{ème}$ triangle.

$$\int_{\Omega_L(t_{s-1})} G^* d\Omega_o = \frac{1}{4\pi a \Delta t_s} \int_{\theta_1}^{\theta_2} \text{abs}\left(\int_{R_{L,3}(\theta)}^{R_{L,2}(\theta)} \left(\exp\left(\frac{-R^2(\theta)}{4a\Delta t_s} \right) R(\theta) \right) dR \right) d\theta$$
$$+ \frac{1}{4\pi a \Delta t_s} \int_{\theta_2}^{\theta_3} \text{abs}\left(\int_{R_{L,1}(\theta)}^{R_{L,2}(\theta)} \left(\exp\left(\frac{-R^2(\theta)}{4a\Delta t_s} \right) R(\theta) \right) dR \right) d\theta \,. \tag{2.42}$$

En intégrant analytiquement par rapport à R, cette dernière équation devient:

$$\int_{\Omega_L(t_{s-1})} G^* d\Omega_o = \frac{1}{2\pi} \int_{\theta_1}^{\theta_2} \text{abs}\left(\exp\left(\frac{-R_{L,3}^2(\theta)}{4a\Delta t_s} \right) - \exp\left(\frac{-R_{L,2}^2(\theta)}{4a\Delta t_s} \right) \right) d\theta$$
$$+ \frac{1}{2\pi} \int_{\theta_2}^{\theta_3} \text{abs}\left(\exp\left(\frac{-R_{L,1}^2(\theta)}{4a\Delta t_s} \right) - \exp\left(\frac{-R_{L,2}^2(\theta)}{4a\Delta t_s} \right) \right) d\theta \tag{2.43}$$

où

$$R_{L,j}(\theta) = -\frac{(x_o^{(L,k)} y_o^{(L,l)} - x_o^{(L,l)} y_o^{(L,k)} + d_{(j)} x + c_{(j)} y)}{d_{(j)} \cos\theta + c_{(j)} \sin\theta} \tag{2.44}$$

avec

$$c_{(j)} = x_o^{(L,l)} - x_o^{(L,k)} \quad . \quad d_{(j)} = y_o^{(L,k)} - x_o^{(L,l)}$$

et

$$k = 2,3,1 \ , \ l = 3,1,2 \quad \text{lorsque} \quad j = 1,2,3 \ .$$

Les points $(x_0^{(L,K)} \ , \ y_0^{(L,K)} \ )$ (K = 1, 2, 3) représentent les coordonnées des sommets du $L^{ème}$ triangle dont l'ordre de numérotation dépend de la façon dont les angles ont été triés. De plus, le point (x,y) représente la position de l'observateur. La démonstration de l'expression $R_{L,j}(\theta)$ (qui est l'équation de la droite passant par le côté opposé au sommet (j) du $L^{ème}$ triangle écrite en coordonnées polaires) est présentée à l'appendice C.

La raison pour laquelle nous prenons la valeur absolue des intégrales en R est que le sommet (2), comme l'illustre la figure 2.7, peut être placé de manière à ce que les intégrales en R s'effectuent en inversant les bornes d'intégration (associées à R) apparaissant dans l'équation (2.42). Mais puisque les résultats des intégrales en R sont toujours positifs (l'intégrant est positif), il n'est donc pas nécessaire de vérifier l'ordre dans lequel les bornes d'intégration sont disposées lorsque nous utilisons la valeur absolue.

Il est à noter que si nous faisons passer une droite par deux sommets d'un triangle et que cette même droite passe aussi par le point où se trouve l'observateur, alors les deux angles correspondant respectivement à ces deux sommets servant à l'intégration en coordonnées polaires sur le triangle, sont égaux (voir fig. C.3 à l'appendice C). Par

conséquent, l'intégrale dont les bornes d'intégration correspondent à ces deux angles, est nulle.

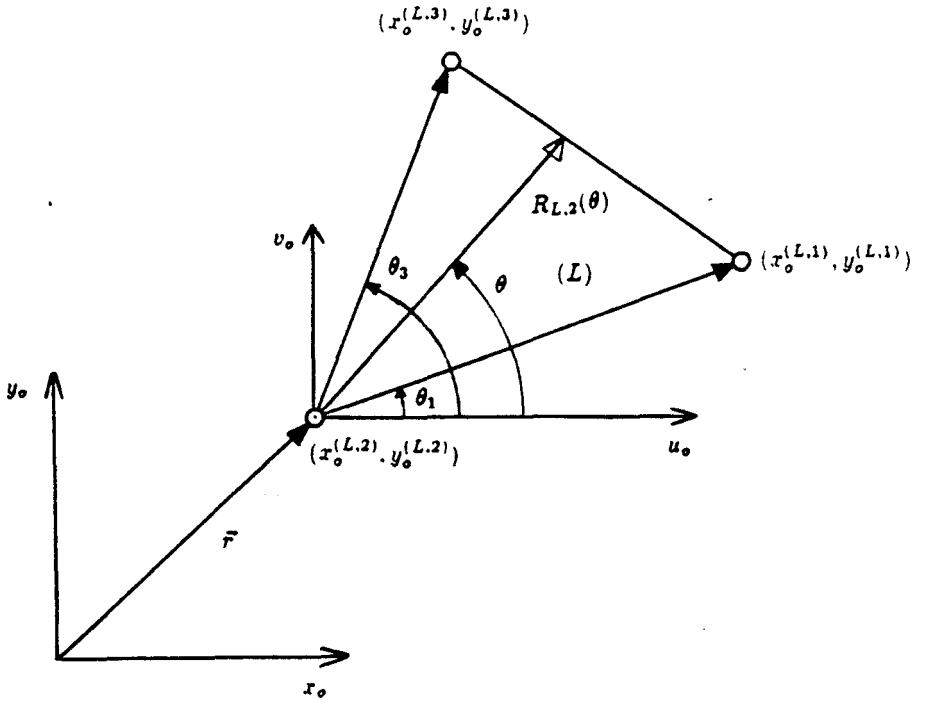
En prenant en considération cette dernière remarque, nous évitons par la même occasion d'obtenir une division par zéro dans l'équation (2.44) puisque cette situation survient seulement lorsque deux sommets sont alignés avec l'observateur (voir appendice C).

2.2.2.2 L'observateur est situé sur un des sommets du triangle

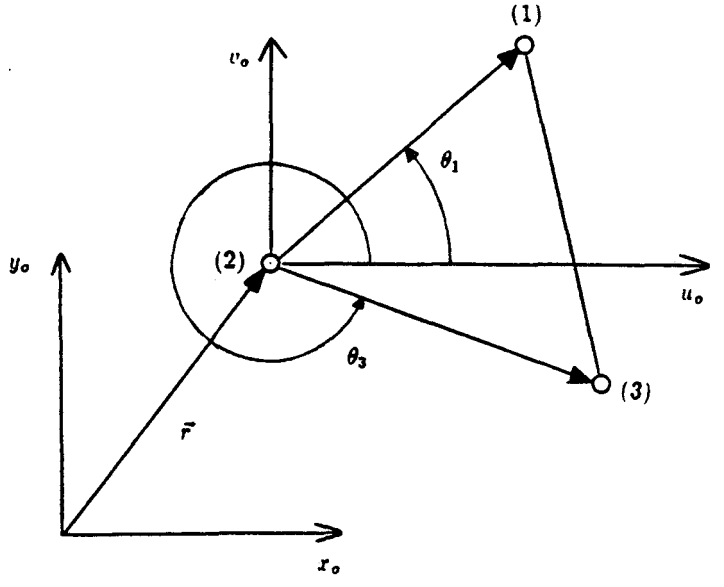
Par convention, nous choisissons de prendre le sommet (2) comme étant le sommet où se trouve l'observateur en lui associant l'angle Θ_2 qui est évidemment nul. Pour les besoins de l'intégration, nous devons, en toute évidence, déterminer l'ordre dans lequel les deux autres angles doivent être disposés. La technique est la même que celle utilisée dans le cas où l'observateur est situé à l'extérieur du triangle sauf que dans cette situation, nous ne considérons pas l'angle Θ_2 (voir fig. 2.8).

En se référant à la figure 2.8, l'intégrale (2.13) devient:

$$\int_{\Omega_L(t_{s-1})} G^* d\Omega_o = \frac{1}{4\pi a \Delta t_s} \int_{\theta_1}^{\theta_3} \left(\int_0^{R_{L,2}(\theta)} \left(\exp\left(\frac{-R^2(\theta)}{4a\Delta t_s} \right) R(\theta) \right) dR \right) d\theta \quad (2.45)$$



a) Cas où $\Theta_3 - \Theta_1 < \Pi$.



b) Cas où $\Theta_3 - \Theta_1 > \Pi$, donc Θ_3 doit devenir Θ_1 et l'ancien $\Theta_1 + 2\Pi$ doit devenir le nouveau Θ_3.

Figure 2.8: Exemples où l'observateur est situé sur un des sommets du $L^{ème}$ triangle.

En intégrant analytiquement par rapport à R, on obtient:

$$\int_{\Omega_L(t_{s-1})} G^* d\Omega_o = \frac{1}{2\pi}\left((\theta_3 - \theta_1) - \int_{\theta_1}^{\theta_3} \exp\left(\frac{-R_{L,2}^2(\theta)}{4\alpha\Delta t_s}\right) d\theta\right) \qquad (2.46)$$

2.2.2.3 L'observateur est situé sur un des côtés du triangle

Comme pour les deux cas précédents, nous définissons une convention pour le sommet (2) qui, dans cette situation, est le sommet opposé au côté où se trouve l'observateur en lui associant évidemment l'angle Θ_2. Une fois l'angle Θ_2 déterminé, nous trions en ordre croissant les angles correspondant aux deux autres sommets (voir fig. 2.9). Cependant, tout comme pour le cas où l'observateur était situé à l'extérieur du triangle, il y a deux cas particuliers à considérer pour bien définir les angles Θ_1 et Θ_3 avant de passer à l'intégration. Toutefois, dans le cas présent, nous devons utiliser une technique différente pour déterminer les angles Θ_1 et Θ_3 car la différence entre ceux-ci est toujours égale à Π (voir fig. 2.9). Pour remédier à cela, il suffit de comparer l'angle Θ_2 avec les deux autres angles. Lorsque l'angle Θ_2 est plus grand que Θ_3 (fig. 2.10a) ou plus petit que Θ_1 (fig. 2.10b) l'angle Θ_3 doit devenir l'angle Θ_1 et l'ancien $\Theta_1 + 2\Pi$ doit devenir le nouveau Θ_3. Pour le deuxième cas particulier (i.e. $\Theta_2 < \Theta_1$), nous devons en plus substituer Θ_2 par $\Theta_2 + 2\Pi$.

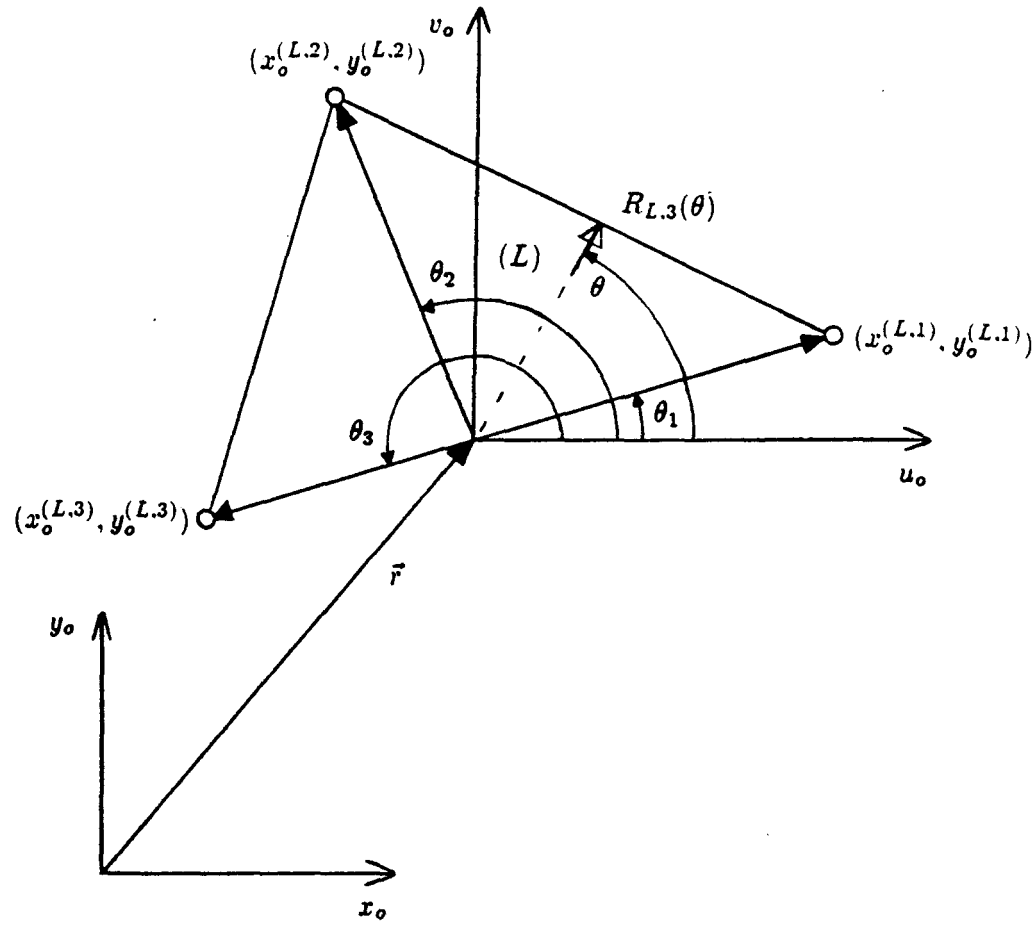


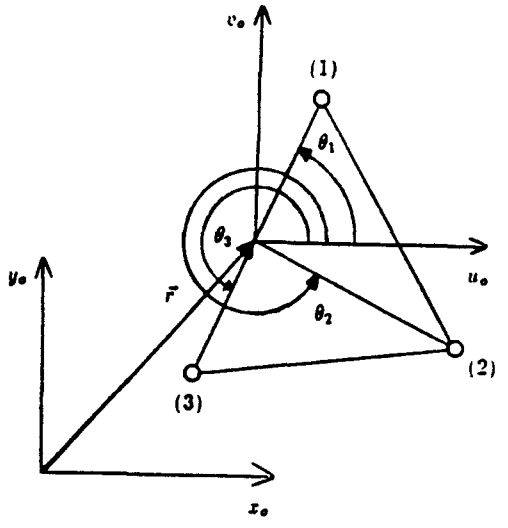
Figure 2.9: Définitions utiles à l'intégration en coordonnées polaires pour l'observateur situé sur un des côtés du $L^{\text{ème}}$ triangle.

Par conséquent, lorsque l'observateur est situé sur un des côtés du triangle, l'intégrale (2.13) s'évalue de la façon suivante:

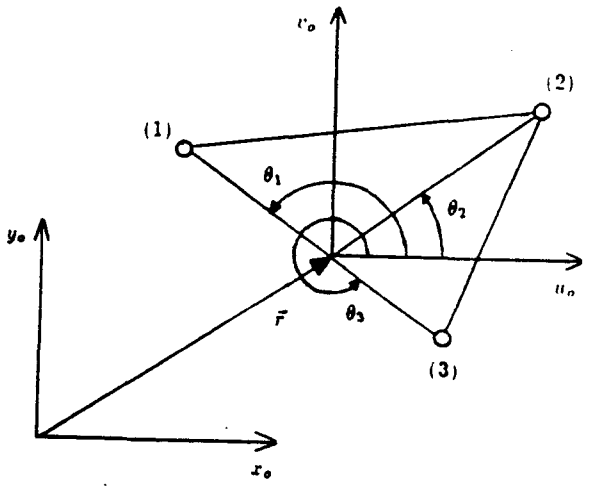
$$\int_{\Omega_L(t_{s-1})} G^* d\Omega_o = \frac{1}{4\pi a \Delta t_s} \int_{\theta_1}^{\theta_2} \left(\int_0^{R_{L,3}(\theta)} \left(\exp\left(\frac{-R^2(\theta)}{4a\Delta t_s} \right) R(\theta) \right) dR \right) d\theta + \frac{1}{4\pi a \Delta t_s} \int_{\theta_2}^{\theta_3} \left(\int_0^{R_{L,1}(\theta)} \left(\exp\left(\frac{-R^2(\theta)}{4a\Delta t_s} \right) R(\theta) \right) dR \right) d\theta . \quad (2.47)$$

En intégrant analytiquement par rapport à R et en tenant compte du fait que $\Theta_3 - \Theta_1 = \Pi$, l'équation (2.47) devient:

$$\int_{\Omega_L(t_{s-1})} G^* d\Omega_o = \frac{1}{2} - \frac{1}{2\pi}\int_{\theta_1}^{\theta_2} \exp\left(\frac{-R^2_{L,3}(\theta)}{4a\Delta t_s}\right) d\theta - \frac{1}{2\pi}\int_{\theta_2}^{\theta_3} \exp\left(\frac{-R^2_{L,1}(\theta)}{4a\Delta t_s}\right) d\theta \,. \tag{2.48}$$



a) Cas où $\Theta_2 > \Theta_3$



b) Cas où $\Theta_2 < \Theta_1$, donc Θ_2 doit être substitué par $\Theta_2 + 2\Pi$.

Figure 2.10: Cas particulier où Θ_3 doit devenir Θ_1 et où l'ancien $\Theta_1 + 2\Pi$ doit devenir le nouveau Θ_3 lorsque a) $\Theta_2 > \Theta_3$ et b) $\Theta_2 < \Theta_1$.

2.2.2.4 L'observateur est situé à l'intérieur du triangle

Lorsque l'observateur est situé à l'intérieur, il suffit, avant de passer à l'intégration, d'ordonner les angles en ordre croissant (voir fig. 2.11) i.e. $0 \leq \Theta_1 < \Theta_2 < \Theta_3 < 2\Pi$.

Une fois les angles triés, l'intégrale définie en (2.13) devient:

$$\int_{\Omega_L(t_s-1)} G^* d\Omega_o = \frac{1}{4\pi a \Delta t_s} \int_{\theta_1}^{\theta_2} \left(\int_0^{R_{L,3}(\theta)} \left(\exp\left(\frac{-R^2(\theta)}{4a\Delta t_s} \right) R(\theta) \right) dR \right) d\theta$$
$$+ \frac{1}{4\pi a \Delta t_s} \int_{\theta_2}^{\theta_3} \left(\int_0^{R_{L,1}(\theta)} \left(\exp\left(\frac{-R^2(\theta)}{4a\Delta t_s} \right) R(\theta) \right) dR \right) d\theta \quad (2.49)$$
$$+ \frac{1}{4\pi a \Delta t_s} \int_{\theta_3}^{\theta_1+2\pi} \left(\int_0^{R_{L,2}(\theta)} \left(\exp\left(\frac{-R^2(\theta)}{4a\Delta t_s} \right) R(\theta) \right) dR \right) d\theta \, .$$

En intégrant analytiquement par rapport à R, on obtient:

$$\int_{\Omega_L(t_s-1)} G^* d\Omega_o = 1 - \frac{1}{2\pi} \int_{\theta_1}^{\theta_2} \exp\left(\frac{-R_{L,3}^2(\theta)}{4a\Delta t_s} \right) d\theta$$
$$- \frac{1}{2\pi} \int_{\theta_2}^{\theta_3} \exp\left(\frac{-R_{L,1}^2(\theta)}{4a\Delta t_s} \right) d\theta \quad (2.50)$$
$$- \frac{1}{2\pi} \int_{\theta_3}^{\theta_1+2\pi} \exp\left(\frac{-R_{L,2}^2(\theta)}{4a\Delta t_s} \right) d\theta \, .$$

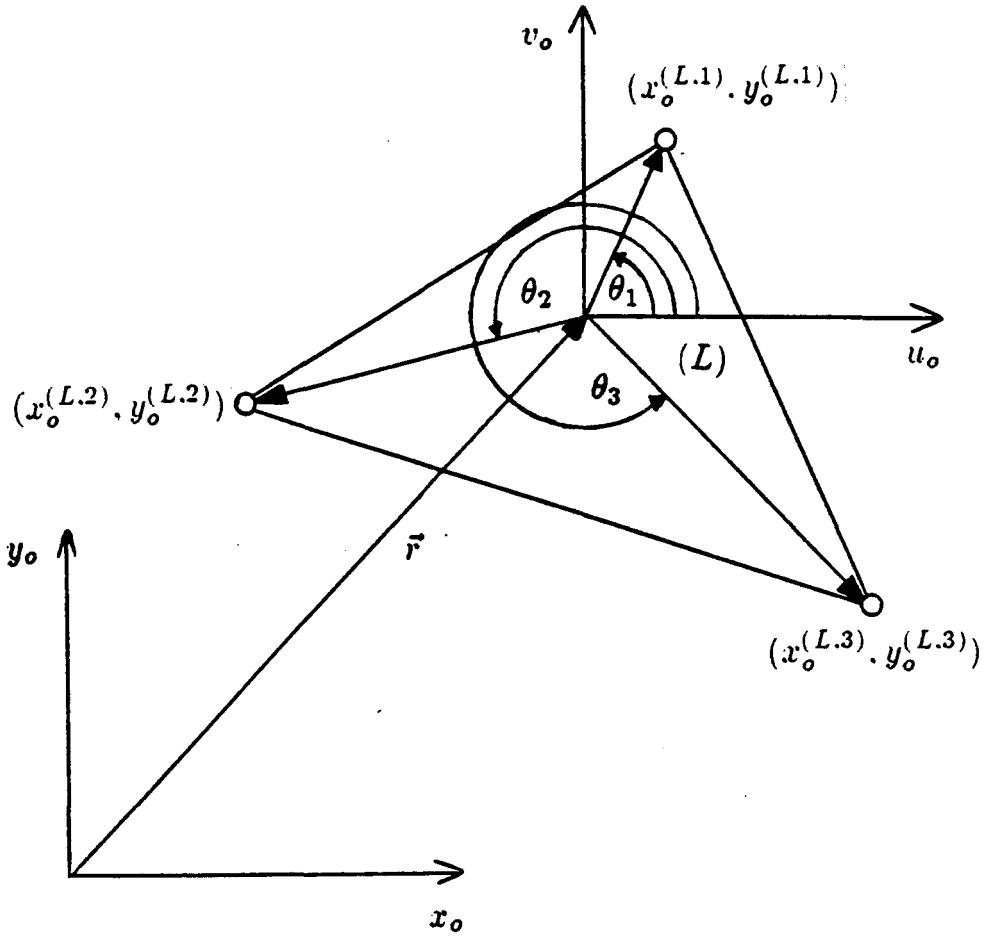


Figure 2.11: Définitions utiles pour l'intégration en coordonnées polaires avec l'observateur situé à l'intérieur du $L^{ème}$ triangle.

2.2.2.5 Remarque

Pour évaluer les intégrales par rapport à Θ, il suffit de ramener les bornes d'intégrations entre -1 et 1 (Carnahan et al., 1969) et d'utiliser la quadrature de Gauss-Legendre. Le nombre de points d'intégration sera déterminé au chapitre suivant.

Comme nous pouvons le constater, le terme $1/\alpha\Delta t_s$ apparaissant dans l'intégrale définie en (2.13) joue un rôle important puisque ce dernier apparaît devant et dans la fonction exponentielle. En effet, la diffusivité étant petite ($\alpha \cong 10^{-5} m^2/s$) nous pouvons éventuellement nous retrouver avec une multiplication d'un grand terme ($1/4\Pi\alpha\Delta t_s$) par un petit terme ($\exp(-R^2/4\alpha\Delta t_s)$) si Δt_s est relativement petit (Δt_s < 100 s), pouvant ainsi occasionner la perte de chiffres significatifs. Le fait de passer en coordonnées polaires, nous permet d'éviter cette situation, dû au fait que le terme $1/4\Pi\alpha\Delta t_s$ est multiplié par 2 $\alpha\Delta t_s$ lorsque le changement de coordonnées est effectué (voir équations (2.43), (2.46), (2.48) et (2.50)).

2.3 Résumé du chapitre

Dans ce chapitre, nous avons d'abord discrétisé l'équation intégrale définie en (1.15) dans le temps et l'espace dans le but d'obtenir le système d'équations linéaires défini en (2.14). Par la suite, nous avons montré comment utiliser les résultats provenant de la résolution de ce système pour calculer l'évolution de la température sur la frontière et à l'intérieur du domaine et aussi pour calculer le déplacement de la frontière en utilisant la condition de Stéfan écrite en termes de quantités nodales (voir éq. (2.15)).

De plus, nous avons vu comment les intégrales de contour doivent être évaluées analytiquement lorsqu'il y a présence d'une singularité et numériquement autrement. Ensuite, pour éviter des problèmes d'intégration

numérique sur le domaine, nous avons montré comment intégrer en coordonnées polaires sur tous les triangles discrétisant le domaine.

CHAPITRE III

EXEMPLES NUMÉRIQUES

3.1 Optimisation de la méthode numérique

Avant de considérer des problèmes de nature complexe, il est important d'étudier le comportement numérique de la méthode des éléments finis de frontière décrite au chapitre II. En effet, la précision des résultats dépend énormément de la discrétisation spatiale, du pas de temps et du nombre de points d'intégration. Pour étudier l'influence qu'ont ces trois facteurs sur les résultats, considérons l'ablation d'un cylindre circulaire d'aluminium de longueur infinie, de rayon 0.5 m, dont la température initiale est uniforme à 300 K (26.8 C) et auquel on impose un flux constant de 100 kW/m^2 sur la face latérale. Les valeurs de la diffusivité (α) et de la conductivité (k) thermiques sont respectivement 8.3×10^{-5} m^2/s et 232 W/ m K (voir section B.2). De même, la valeur de la chaleur latente est 9.53×10^8 J/m^3 (Tremblay, 1986).

Pour solutionner ce problème de Landau à une phase, nous approximons le cercle avec 20 éléments linéaires sur lesquels on impose le flux de chaleur décrit plus tôt (voir fig. 3.1).

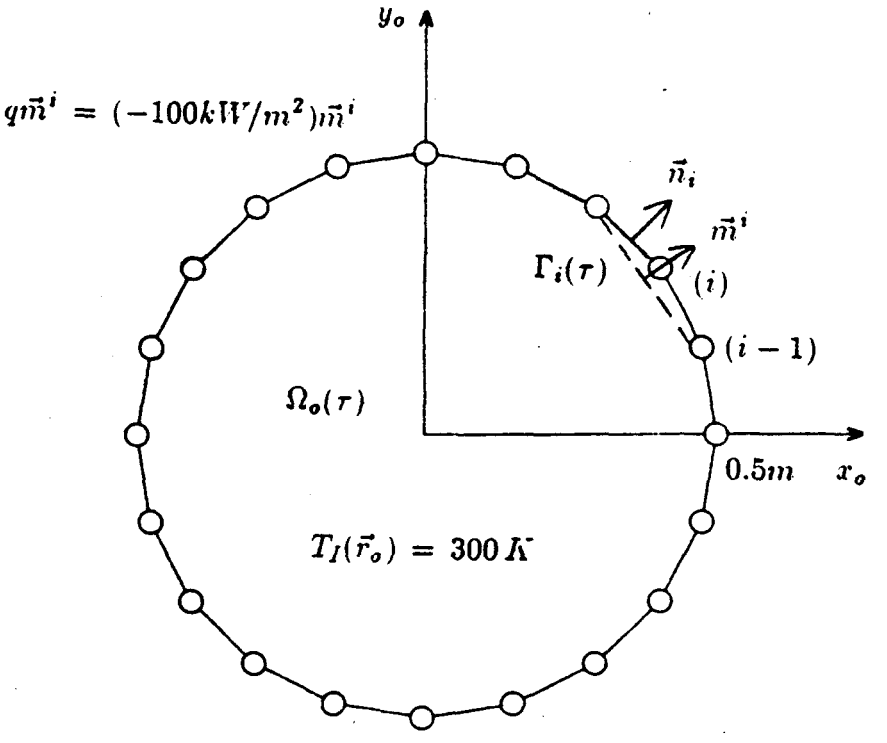


Figure 3.1: Discrétisation du cercle de rayon 0.5 m en 20 éléments linéaires.

Dû à la déformation du domaine lors de la phase de fusion, nous devons intégrer sur le domaine à chaque pas de temps. Pour ce faire, un maillage automatique doit être développé i.e. après chaque déformation du domaine, une reconfiguration triangulaire approximant le domaine doit être effectuée sans aucune aide externe au programme lors de la simulation.

Pour les besoins de ce travail, en termes de maillage automatique, nous utilisons une technique simple qui consiste tout d'abord à trouver le centre de gravité du domaine que l'on définit comme étant l'origine du système d'axes (x_o,y_o). Ensuite, nous joignons avec une ligne droite chaque noeud de la frontière avec le centre de gravité du domaine qui,

dans ce problème, est le centre du cercle. Ces lignes droites sont définies par

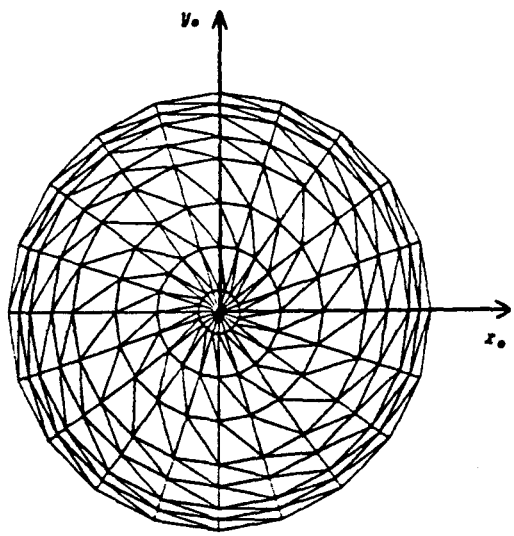
$$(x_o(\lambda), y_o(\lambda))^j = (1-\lambda)\,(x_o^j, y_o^j)$$

où $0 \leq \lambda \leq 1$ et $j = 1,2,\ldots,N = 20$.

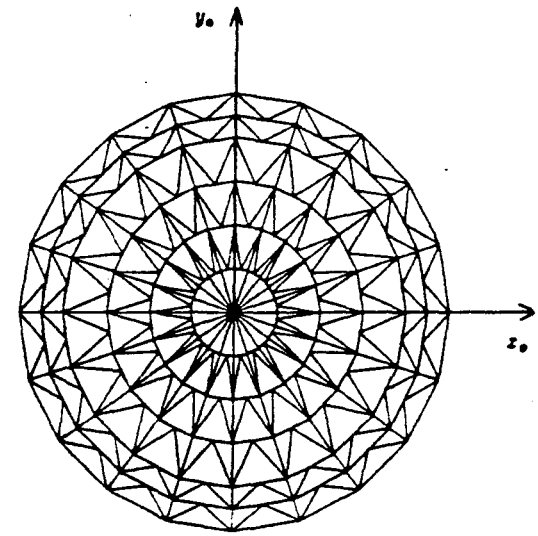
Par la suite, nous subdivisons ces lignes en plusieurs parties en prenant différentes valeurs de λ. Comme étape finale, nous construisons la configuration triangulaire que nous désirons obtenir. Dans ce travail, nous utilisons trois différentes configurations (voir fig. 3.2) et étudions l'influence qu'a chacune de ces configurations sur la précision des résultats. La seule différence qui existe entre la deuxième (fig. 3.2b) et la troisième (fig. 3.2c) configuration est l'orientation des triangles.

La raison pour laquelle nous prenons de petites subdivisions proches de la frontière est de faire absorber le fort gradient de température qui existe à cet endroit au début de la simulation.

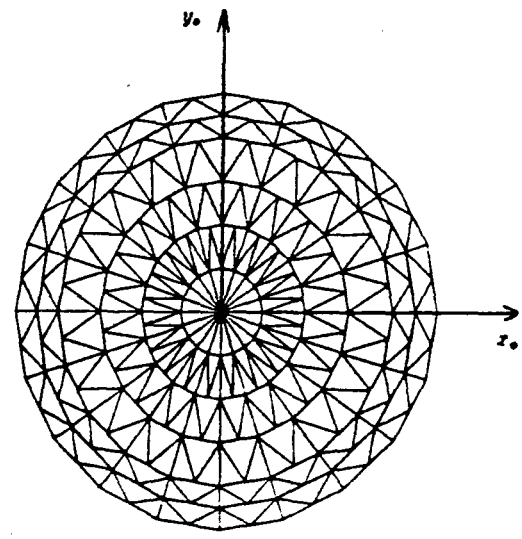
Puisque le gradient de température est très élevé au début de la phase de préfusion, montrons que le choix d'un bon pas de temps et d'une bonne configuration triangulaire est primordial pour obtenir de bons résultats lors de cette phase.



a) Première configuration avec 300 triangles et λ = 0, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7 et 0.9.



b) Deuxième configuration avec 320 triangles et λ = 0, 0.1, 0.2, 0.4, 0.6 et 0.8.



c) Troisième configuration avec 320 triangles et λ = 0, 0.1, 0.2, 0.4, 0.6, et 0.8.

Figure 3.2: Discrétisation du domaine en éléments triangulaires.

3.1.1 Phase de préfusion

Comme il a été mentionné à la section 2.1.3, la phase de préfusion est celle où il n'y a que de la conduction et qui se termine lorsque la température de fusion est atteinte à au moins un des points de la frontière. Vu que la température initiale du cylindre est uniforme et que le flux imposé à sa face latérale est constant, il s'ensuit que l'évolution de la température est la même pour tous les points situés à une distance fixe r de l'axe central du cylindre. Par conséquent, tous les points situés sur la frontière atteignent la température de fusion au même temps t_f. De plus, les propriétés thermiques étant constantes, il existe une solution analytique (voir équation (B.14)) donnant l'évolution de la température à la frontière. Nous allons utiliser cette solution analytique comme référence pour déterminer un pas de temps, un nombre de points d'intégration et une configuration triangulaire.

Pour des problèmes d'ablation de nature plus complexe, il n'existe malheureusement pas toujours de solution analytique pour la phase de préfusion. Toutefois, nous voulons avoir une bonne idée de la tendance des résultats pour de tels problèmes. Pour cela, nous avons décidé de calculer à chaque minute de la phase de préfusion, l'erreur relative commise sur le bilan d'énergie net (i.e. l'énergie absorbée par le solide durant une minute versus l'énergie fournie au cylindre pour la même période de temps) (voir appendice D). La raison de l'utilisation d'un tel procédé est qu'il donne une bonne indication de la précision des résultats. En effet, comme nous allons le voir au cours de cette section, lorsque la solution numérique approche la solution analytique, l'erreur

relative sur le bilan d'énergie net calculée à chaque minute est toujours près de zéro, sauf au début où elle est élevée dû au fort gradient de température.

3.1.1.1 Choix du pas de temps

Si nous voulons que la température à la frontière du cylindre atteigne le point de fusion avec précision, un petit pas de temps doit être adopté. Cependant, dû au fait que l'intégrant (i.e. la solution fondamentale), apparaissant dans l'intégrale sur le domaine, devient une pseudo fonction delta de Dirac lorsque $\Delta t_s \to 0$ (Morse et al, 1953), des problèmes numériques peuvent être introduits à l'intérieur de la solution si le pas de temps est trop petit; et cela même si nous utilisons la transformation en coordonnées polaires (Wrobel et al, 1984).

Par conséquent, le choix d'un petit pas de temps est limité et cette limite dépend de la géométrie du domaine et de la diffusivité thermique (Hong et al., 1983). Par exemple, pour notre problème, un pas de temps de 10 secondes ne donne pas de bons résultats, par contre ceux obtenus avec un pas de temps de 30 secondes sont de beaucoup supérieurs comme le démontrent bien les figures 3.3 et 3.4.

D'autre part, en se référant à ces mêmes figures, nous pouvons constater que les résultats obtenus avec $\Delta t_s = 60$ s ne sont pas supérieurs à ceux obtenus avec $\Delta t_s = 30$ s. La raison est que le pas de temps de 60 s est trop grand pour bien représenter le gradient élevé de température

existant à l'intérieur du cylindre au début de la phase de préfusion. Nous avons donc choisi de prendre un pas de temps de 30 s lors de la phase de préfusion pour le reste des simulations à venir.

La façon dont l'erreur relative sur le bilan d'énergie net est calculée à chaque minute, nous donne des informations intéressantes. En effet, lorsque l'erreur relative sur le bilan d'énergie net calculée à chaque minute est en général positive, cela nous indique que la température augmente plus rapidement que la réalité (il y a plus d'énergie absorbée que d'énergie fournie) (voir fig. 3.4 avec Δt_s = 60 s). Par contre, lorsque l'erreur relative sur le bilan d'énergie net calculée à chaque minute est en général négative, c'est l'inverse qui se produit i.e. la température augmente moins rapidement que la réalité (il y a moins d'énergie absorbée que d'énergie fournie (voir fig. 3.4 avec Δt_s = 10 s).

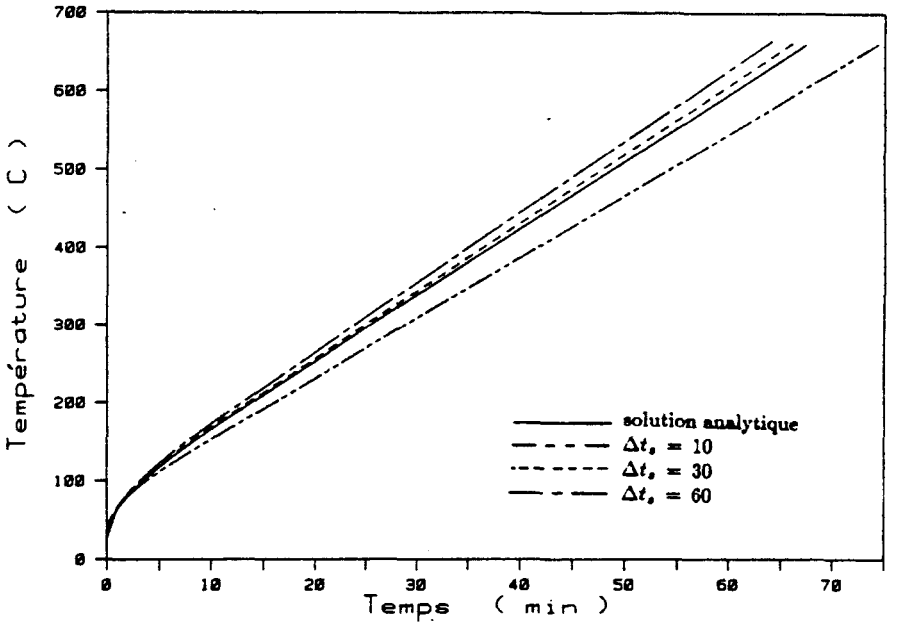


Figure 3.3: Évolution de la température à la frontière en fonction du temps donnée par la solution analytique (B.14) et par le modèle utilisant trois pas de temps différents (Δt_s = 10,30,60 s) avec la configuration triangulaire de la figure 3.2c.

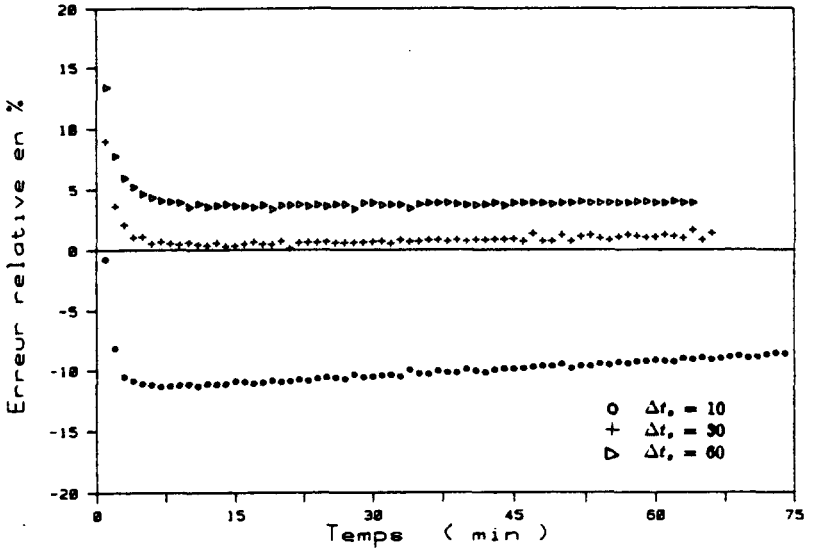


Figure 3.4: Erreur relative sur le bilan d'énergie net calculée à chaque minute jusqu'à ce que le point de fusion soit atteint à la frontière en utilisant différents pas de temps (Δt_s = 10,30,60 s) avec la configuration triangulaire de la figure 3.2c.

3.1.1.2 Choix du nombre de points d'intégration

Lors de la détermination du pas de temps, nous avons utilisé 12 points d'intégration pour intégrer numériquement à partir de la quadrature de Gauss-Legendre au lieu de 6 points tel que recommandé par Wrobel et Brebbia (1981b). Comme l'illustrent les figures 3.5 et 3.6, l'utilisation de 6 points d'intégration ne donne pas des résultats acceptables. La raison de l'utilisation d'un aussi grand nombre de points d'intégration est imputable à la présence de l'expression $\exp\left(\frac{-R^2}{4\alpha\Delta t_s}\right)$ apparaissant dans les intégrales concernées. En effet, dû à la petite valeur de la diffusivité ($\alpha = 8.3 \times 10^{-5}$ m^2/s) et à la valeur du pas de temps ($\Delta t_s = 30$ s), cette expression peut devenir très petite et par conséquent difficilement intégrable numériquement.

Pour l'intégration numérique utilisant la quadrature de Gauss-Laguerre, nous utilisons 6 points d'intégration, tel qu'adopté par Wrobel et Brebbia (1981b).

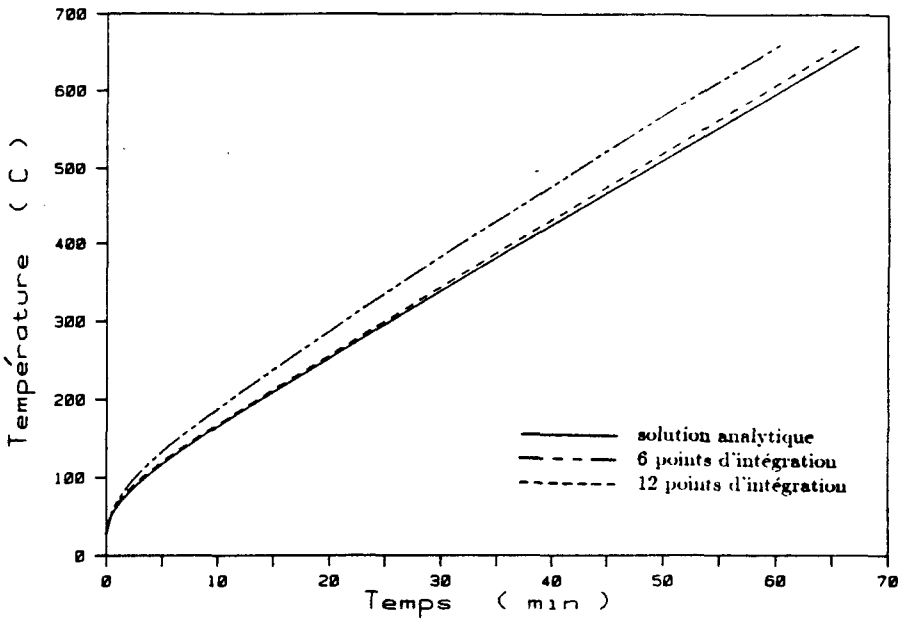


Figure 3.5: Évolution de la température à la frontière en fonction du temps donnée par la solution analytique (B.14) et par le modèle utilisant 6 et 12 points d'intégration (Gauss-Legendre) avec la configuration triangulaire de la figure 3.2c et Δt_s = 30 s.

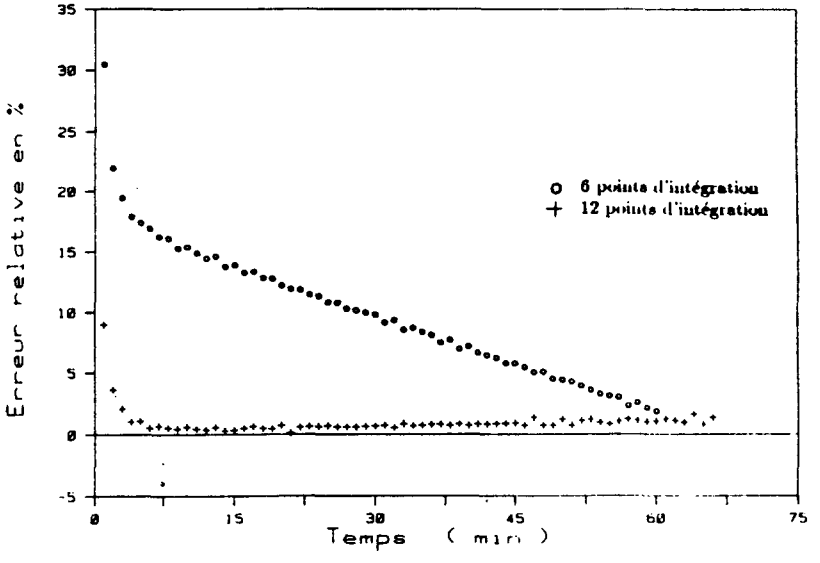


Figure 3.6: Erreur relative sur le bilan d'énergie net calculée à chaque minute jusqu'à ce que le point de fusion soit atteint à la frontière en utilisant 6 et 12 points d'intégration (Gauss-Legendre) avec la configuration triangulaire de la figure 3.2c et Δt_s = 30 s.

3.1.1.3 Choix de la configuration triangulaire

Jusqu'à maintenant, toutes les simulations ont été effectuées en utilisant la configuration triangulaire de la figure 3.2c. En se référant aux figures 3.7 et 3.8, on peut constater que les résultats obtenus avec la troisième configuration sont de beaucoup supérieurs à ceux obtenus en utilisant les deux premières configurations. Si nous analysons la deuxième configuration (fig. 3.2b), nous pouvons constater que les plus gros triangles situés sur la couche en bordure de la frontière absorbent, au début de la phase de préfusion, la majorité du gradient de température, ce qui tend à faire augmenter la température plus rapidement que celle utilisant la troisième configuration. En effet, la configuration de la figure 3.2c est telle que ce sont les plus petits triangles se trouvant en bordure de la frontière qui absorbent la plus grande partie du gradient de température donnant ainsi de meilleurs résultats.

Pour la configuration de la figure 3.2a, la représentation des triangles minces et allongés a pour effet de sous-évaluer l'augmentation de température.

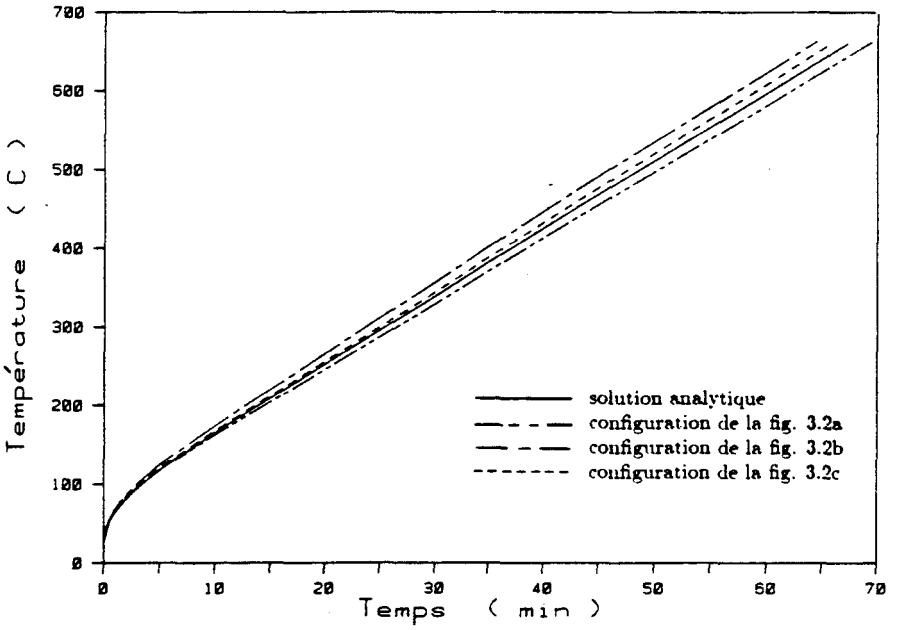


Figure 3.7: Évolution de la température à la frontière en fonction du temps donnée par la solution analytique (B.14) et par le modèle utilisant les trois configurations triangulaires de la figure 3.2 et avec Δt_s = 30 s.

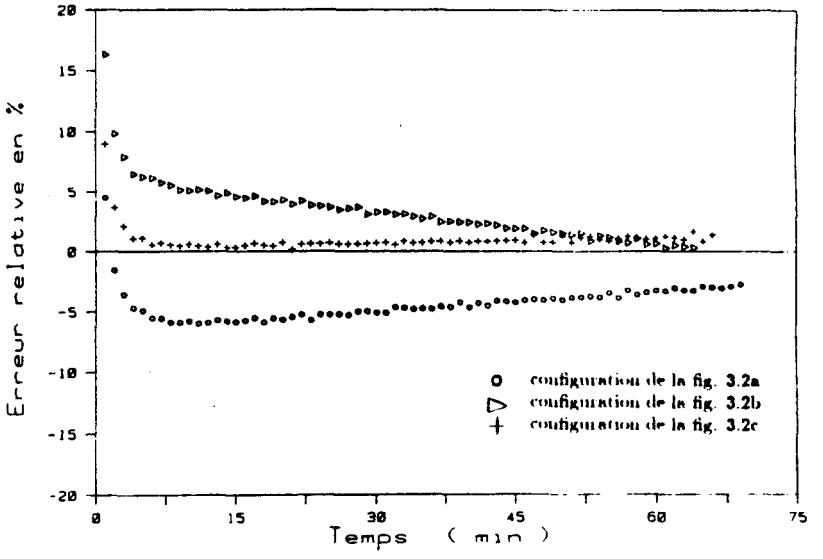


Figure 3.8: Erreur relative sur le bilan d'énergie net calculée à chaque minute jusqu'à ce que le point de fusion soit atteint à la frontière en utilisant les configurations triangulaires de la figure 3.2 et avec Δt_s = 30 s.

3.1.2 Phase de fusion

Puisqu'il n'existe pas de solution analytique pour la phase de fusion et qu'il y a déformation du domaine, donc reconfiguration des triangles, nous évaluons l'erreur relative commise sur le bilan d'énergie net de tout le procédé d'ablation (i.e. les phases de préfusion et de fusion) pour vérifier la validité des résultats (voir appendice D).

3.1.2.1 Choix du pas de temps

Pour la phase de fusion, étant donné que nous recalculons à tous les pas de temps les matrices [H], [G] et [B] (voir équation (2.14)) et que le gradient de température est beaucoup moins important que celui du début de la phase de préfusion, nous jugeons bon d'augmenter le pas de temps pour diminuer le temps de calcul. Cette augmentation ne doit cependant pas être trop importante puisque nous supposons qu'il n'y a pas de déplacement à l'intérieur d'un même pas de temps. Par conséquent, nous choisissons d'augmenter le pas de temps à 60 secondes pour la phase de fusion au lieu de le garder à 30 secondes. Cet accroissement du pas de temps au début de la phase de fusion ne modifie pas significativement les résultats en termes de position de la frontière et de bilan d'énergie puisque le gradient de température à l'intérieur du domaine est beaucoup moins important que celui observé au début de la phase de préfusion (voir table 3.1). En fait le déplacement de la frontière calculé avec un pas de temps de 60 secondes, est le même que celui calculé avec un pas de temps de 30

secondes lors de la phase de fusion. La figure 3.9 montre la position d'un noeud situé sur la frontière en fonction du temps lors de la phase de fusion calculée numériquement à partir de la technique décrite à la section (2.1.4) avec Δt_s = 60 s. Évidemment nous n'avons pas à refaire ce graphique pour les autres noeuds de la frontière puisque, par symétrie, les déplacements sont les mêmes.

TABLE 3.1: Comparaison des résultats entre deux différents pas de temps pour la phase de fusion avec Δt_s = 30 s pour la phase de préfusion en utilisant la configuration de la figure 3.2c.

Pas de temps pour la phase de fusion (s)	Erreur relative en % sur le bilan d'énergie net de tout le procédé d'ablation	Temps nécessaire pour faire fondre le cylindre (h)	Temps de calcul requis sur un MV8000 de Data General (h)
30	1.07	2.54	7.13
60	1.17	2.54	4.20

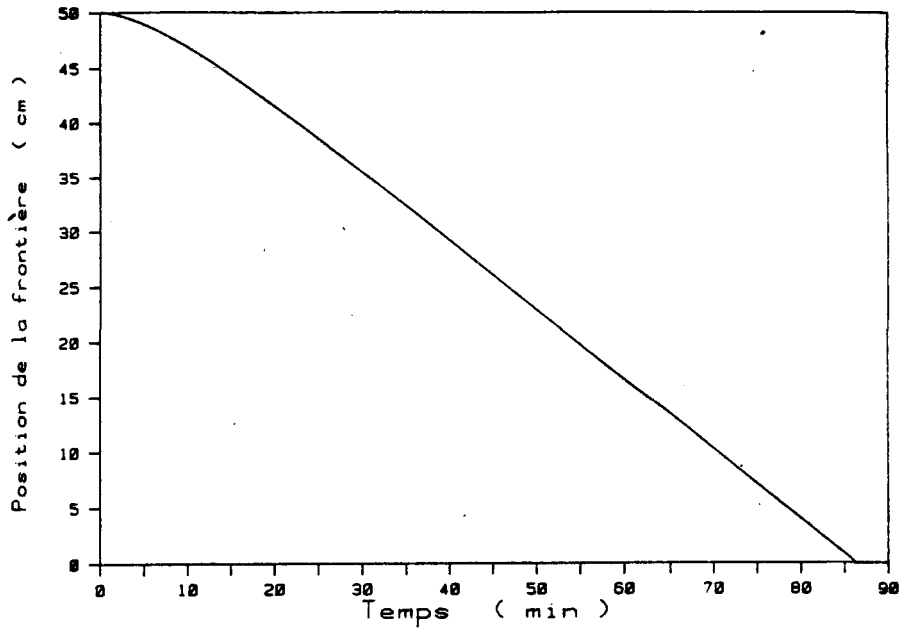


Figure 3.9: Courbe donnant, pour la phase de fusion, la position d'un noeud situé sur la frontière en fonction du temps avec Δt_s = 60 s et utilisant la configuration de la figure 3.2c.

À noter que sur le graphique de la figure 3.9, le temps t = 0 correspond au début de la phase de fusion.

3.1.2.2 Diminution du nombre d'éléments triangulaires lors de la phase de fusion

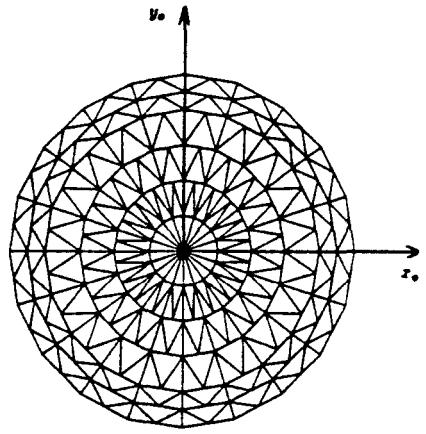
Pour les deux pas de temps utilisés pour la phase de fusion (i.e. Δt_s = 30 s et 60 s), le nombre d'éléments triangulaires est diminué au fur et à mesure que le gradient de température à l'intérieur du domaine diminue. En fait, la diminution du nombre d'éléments se fait suivant le principe qui suit. À chaque pas de temps, avant d'effectuer le remaillage

du domaine lors de la phase de fusion, nous calculons la différence entre la plus grande et la plus petite température du domaine. Ensuite, selon la différence obtenue, le nombre de couches de triangles discrétisant le domaine et leur répartition sont déterminés tel qu'indiqué à la table 3.2.

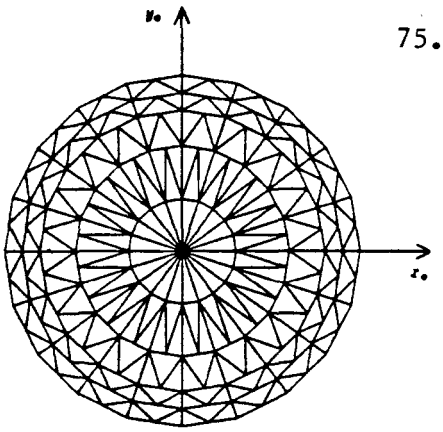
TABLE 3.2: Nombre de couches de triangle discrétisant le domaine avec la répartition de ceux-ci suivant la différence obtenue entre la plus grande et la plus petite température du domaine lors de la phase de fusion.

Différence (notée diff) entre la plus petite et la plus grande température du domaine lors de la phase de fusion (K)	Nombre de couches de triangles discrétisant le domaine	Répartition des couches de triangles suivant le nombre (voir fig. 3.10)
200 < diff < 933.2	6	λ = 0,0.1,0.2,0.4,0.6,0.8
100 < diff < 200	5	λ = 0,0.1,0.2,0.5,0.7
50 < diff < 100	4	λ = 0,0.2,0.4,0.7
10 < diff < 50	3	λ = 0,0.3,0.5
2 < diff < 10	2	λ = 0,0.4
0 < diff < 2	1	λ = 0

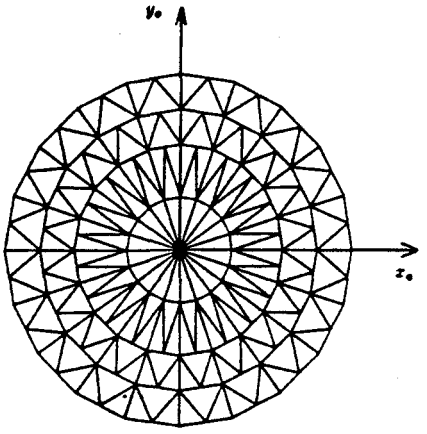
Les choix des valeurs de λ ont été faits pour que l'épaisseur des différentes couches de triangles soit semblable, excepté pour celles situées proche de la frontière où elles doivent être plus minces que les autres couches afin de mieux absorber le gradient de température plus élevé à la frontière qu'au centre au début de la simulation.



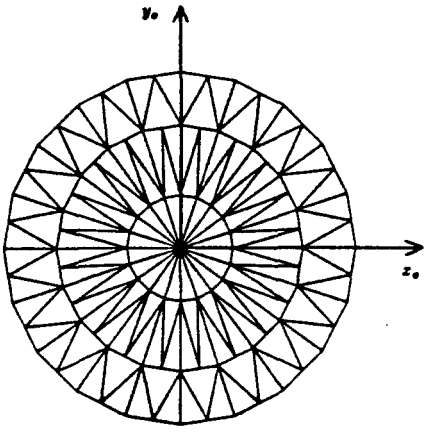
$\lambda = 0, 0.1, 0.2, 0.4, 0.6$ et 0.8



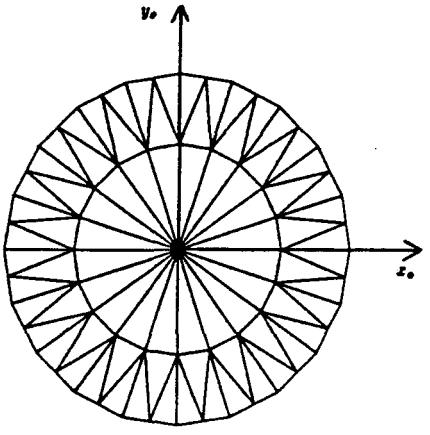
$\lambda = 0, 0.1, 0.2, 0.5$ et 0.7



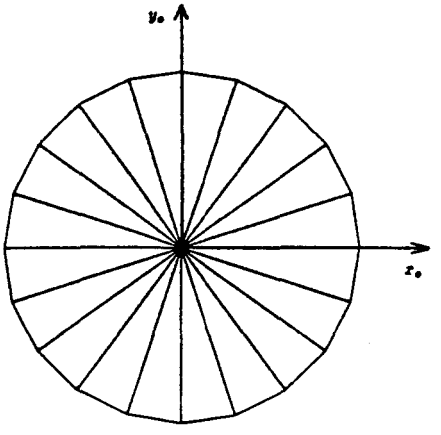
$\lambda = 0, 0.2, 0.4$ et 0.7



$\lambda = 0, 0.3$ et 0.5



$\lambda = 0$ et 0.4



$\lambda = 0$

Figure 3.10: Diminution du nombre d'éléments triangulaires lors de la phase de fusion au fur et à mesure que le gradient de température à l'intérieur du domaine diminue en utilisant la configuration de la figure 3.2c.

Cette technique de diminution du nombre d'éléments triangulaires mène à de très bons résultats et a le grand avantage de diviser par huit le temps de calcul nécessaire lorsque le nombre d'éléments est maintenu fixe. Pour les deux cas, la position de la frontière est la même (celle illustrée à la fig. 3.9) et la différence entre les résultats se situe au niveau de l'erreur relative sur le bilan d'énergie net de tout le procédé d'ablation qui est toutefois négligeable (voir table 3.3).

TABLE 3.3: Comparaison des résultats entre un nombre fixe et un nombre décroissant d'éléments triangulaires durant la phase de fusion avec Δt_s = 30 s pour la phase de préfusion et Δt_s = 60 s pour la phase de fusion et en utilisant la configuration de la figure 3.2c.

Nombre d'éléments triangulaires	Erreur relative en % sur le bilan d'énergie net de tout le procédé d'ablation	Temps nécessaire pour faire fondre le cylindre (h)	Temps de calcul requis sur un MV8000 de Data General (h)
fixe	1.15	2.54	33.32
décroissant	1.17	2.54	4.20

3.1.2.3 Vitesse d'ablation lorsque tout le domaine atteint la température de fusion

Un certain temps après le commencement de la phase de fusion, la température interne du domaine qui n'a pas encore subi l'ablation, atteint le point de fusion sans pour autant provoquer la fonte de tout le domaine dans l'écart de temps considéré, la quantité de chaleur latente emmagasinée n'étant pas suffisante. Par conséquent, nous

n'avons plus à solutionner le système d'équations linéaires (2.14) pour pouvoir utiliser la condition de Stefan qui calcule le déplacement des noeuds (i.e. l'équation (2.15)) puisque le gradient de température ($\partial T/\partial n$) est nul à l'intérieur du domaine. En fait, l'équation (2.15) peut être reformulée de la façon suivante:

$$\Delta \vec{S}(\vec{r}^{\,i}) = \frac{\Delta t_s}{\rho L} q(\vec{r}^{\,i}, t_s)\vec{m}^{\,i}$$

Puisque le flux est constant, le déplacement de la frontière atteint ce qu'on appelle la vitesse d'ablation en régime établi (Warren et al., 1961) donnée par $\frac{q}{\rho L}\vec{m}^{\,i} = -0.3\frac{m}{h}\vec{m}^{\,i}$ pour la valeur de flux utilisée de -100 kW/m^2 (le signe - étant utilisé pour dire que le flux entre dans le domaine).

3.1.3 Plans de symétrie

Toutes les simulations effectuées jusqu'à maintenant l'ont été en solutionnant pour tous les noeuds situés sur la frontière du cercle. Cependant, par symétrie, il est possible de solutionner seulement pour les noeuds situés sur le quart de cercle. En effet, si nous trouvons la température ou le gradient de température à chacun des noeuds du quart de cercle, nous l'avons fait évidemment pour tout le cercle complet par symétrie.

Pour solutionner sur le quart de cercle, nous utilisons la même technique que le cercle complet. En effet, nous utilisons l'équation

(2.12) définie au chapitre II avec comme frontière discrétisée le cercle complet et comme domaine discrétisé l'intérieur de ce cercle complet, sauf que cette équation n'est évaluée que pour l'observateur situé sur un des noeuds du quart de cercle i.e. pour $\vec{r} = \vec{r}^{\,i}$ où $\vec{r}^{\,i}$ (i = 1,2,...,6) est le vecteur position du noeud (i) du quart de cercle. Par la suite, nous mettons en évidence les températures ou les gradients de température qui par symétrie, sont identiques. Par exemple, en évaluant l'intégrale de contour définie à la deuxième ligne de l'équation (2.12) avec l'observateur situé sur le noeud (i) ($\vec{r}^{\,i}$) (i.e. nous multiplions la $i^{ème}$ ligne de la matrice [H] par le vecteur colonne des températures $\{T_s\}$ prises aux noeuds du cercle complet) nous avons:

$$\begin{aligned}\sum_{j=1}^{N=20} H_{i,j}T(\vec{r}_o^{\,j},t_s) &= (H_{i,1}+H_{i,11})T(\vec{r}_o^{\,1}) + (H_{i,2}+H_{i,10}+H_{i,12}+H_{i,20})T(\vec{r}_o^{\,2},t_s)\\ &+ (H_{i,3}+H_{i,9}+H_{i,13}+H_{i,19})T(\vec{r}_o^{\,3},t_s)\\ &+ (H_{i,4}+H_{i,8}+H_{i,14}+H_{i,18})T(\vec{r}_o^{\,4},t_s)\\ &+ (H_{i,5}+H_{i,7}+H_{i,15}+H_{i,17})T(\vec{r}_o^{\,5},t_s)\\ &+ (H_{i,6}+H_{i,16})T(\vec{r}_o^{\,6},t_s) \qquad (3.1)\end{aligned}$$

puisque par symétrie (voir fig. 3.11):

$$\begin{aligned}&T(\vec{r}_o^{\,1},t_s) = T(\vec{r}_o^{\,11},t_s),\\ &T(\vec{r}_o^{\,2},t_s) = T(\vec{r}_o^{\,10},t_s) = T(\vec{r}_o^{\,12},t_s) = T(\vec{r}_o^{\,20},t_s),\\ &T(\vec{r}_o^{\,3},t_s) = T(\vec{r}_o^{\,9},t_s) = T(\vec{r}_o^{\,13},t_s) = T(\vec{r}_o^{\,19},t_s), \qquad (3.2)\\ &T(\vec{r}_o^{\,4},t_s) = T(\vec{r}_o^{\,8},t_s) = T(\vec{r}_o^{\,14},t_s) = T(\vec{r}_o^{\,18},t_s),\\ &T(\vec{r}_o^{\,5},t_s) = T(\vec{r}_o^{\,7},t_s) = T(\vec{r}_o^{\,15},t_s) = T(\vec{r}_o^{\,17},t_s),\\ &T(\vec{r}_o^{\,6},t_s) = T(\vec{r}_o^{\,16},t_s).\end{aligned}$$

Par conséquent, la matrice [H] qui, initialement était une matrice de 20 x 20, est maintenant devenue une matrice (6 x 6) par symétrie; les

lignes 7 à 20 étant automatiquement éliminées dû à leur égalité (tout comme les températures prises aux noeuds du cercle complet (voir les équations en (3.2)) avec les lignes de 1 à 6 i.e. pour j = 1,2,...,6

$$H^S_{1,j} = H^S_{11,j},$$
$$H^S_{2,j} = H^S_{10,j} = H^S_{12,j} = H^S_{20,j},$$
$$H^S_{3,j} = H^S_{9,j} = H^S_{13,j} = H^S_{19,j},$$
$$H^S_{4,j} = H^S_{8,j} = H^S_{14,j} = H^S_{18,j},$$
$$H^S_{5,j} = H^S_{7,j} = H^S_{15,j} = H^S_{17,j},$$
$$H^S_{6,j} = H^S_{16,j}.$$

où $H^s_{i,j}$ est le coefficient multipliant la température $T(\vec{r}^j_o, t_s)$ dans l'équation (3.1).

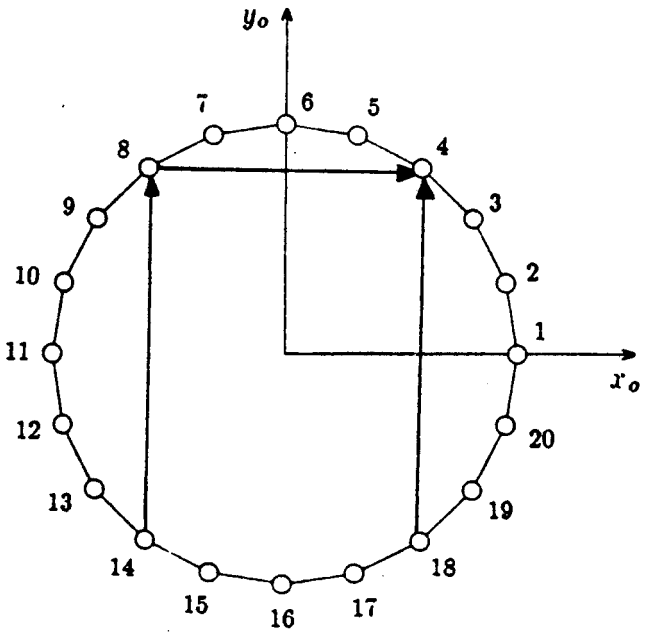


Figure 3.11: Numérotation des noeuds de la frontière pour montrer la symétrie selon les axes des x_o et y_o.

Cette technique, appelée processus de condensation directe avec intégration sur les éléments réfléchis (Brebbia et al., 1984), s'applique

également aux matrices [G] et [B]. Ces matrices deviennent respectivement, après condensation, des matrices de (6 x 6) et (M/4) x 6. M représente le nombre d'éléments triangulaires discrétisant le cercle complet dont la configuration est telle qu'il y a symétrie par rapport aux axes x_o et y_o, ce qui est le cas pour les configurations des figures 3.2b et 3.2c mais non pour celle de la figure 3.2a.

Cette façon de solutionner par symétrie sur le quart de cercle a l'avantage de donner exactement les mêmes résultats que ceux obtenus avec le cercle complet mais avec environ quatre fois moins de temps de calcul c'est-à-dire 1.12h au lieu de 4.2h sur un MV8000 de Data General.

3.1.4 Effet de l'augmentation du nombre d'éléments linéaires à la frontière sur les résultats

Le temps de calcul étant beaucoup moindre en solutionnant sur le quart de cercle que sur le cercle complet, étudions l'influence d'une augmentation du nombre d'éléments linéaires à la frontière sur les résultats avec toutefois le même nombre de couches triangulaires, c'est-à-dire 6. En prenant, par exemple, 32 éléments au lieu de 20 (i.e. 8 éléments sur le quart de cercle au lieu de 5), l'évolution de la température à la frontière est presque identique à celle calculée avec la solution analytique (voir fig. 3.12). De plus, l'erreur relative sur le bilan d'énergie net calculée à chaque minute durant la phase de préfusion est, en général, presque nulle (voir fig. 3.13). Pour la phase de fusion,

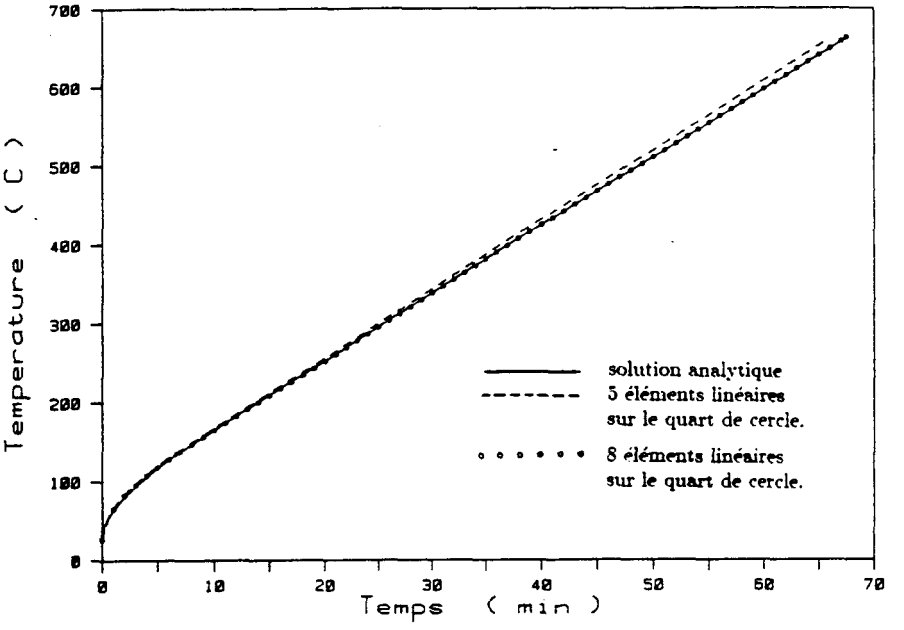


Figure 3.12: Évolution de la température à la frontière donnée par la solution analytique (B.14) et par le modèle utilisant deux nombres différents d'éléments linéaires sur le quart de cercle avec la configuration triangulaire de la figure 3.2c et Δt_s = 30 s.

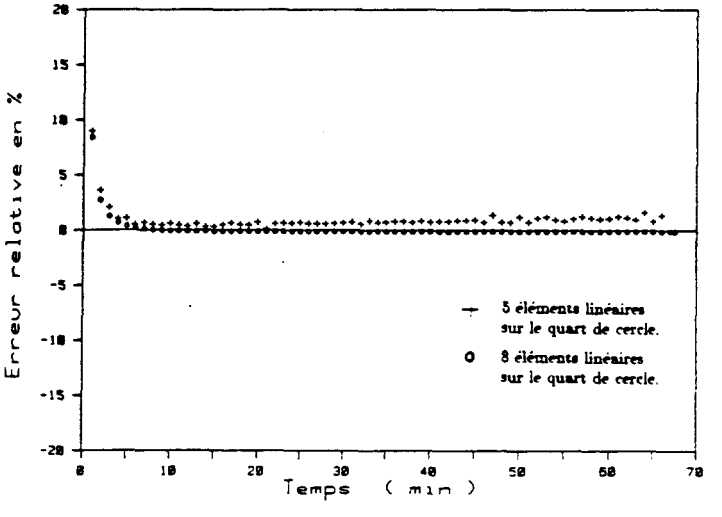


Figure 3.13: Erreur relative sur le bilan d'énergie net calculée à chaque minute jusquà ce que le point de fusion soit atteint à la frontière en utilisant 5 et 8 éléments linéaires sur le quart de cercle avec la configuration triangulaire de la figure 3.2c et Δt_s = 30 s.

le déplacement de la frontière est le même que celui calculé avec 20 éléments (voir fig. 3.9).

Même si les résultats obtenus avec 32 éléments sont supérieurs à ceux obtenus avec 20 éléments, sur le bilan global de la simulation, la différence n'est pas assez importante pour justifier un temps de calcul qui est plus de trois fois supérieur (voir table 3.4).

TABLE 3.4: Comparaison des résultats obtenus en utilisant deux nombres différents d'éléments linéaires à la frontière, en solutionnant sur le quart de cercle avec la configuration de la figure 3.2c.

Nombre d'éléments linéaires sur le quart de cercle	Erreur relative en % sur le bilan d'énergie net calculée sur tout le procédé d'ablation	Temps nécessaire pour faire fondre le cylindre (h)	Temps de calcul requis sur un MV8000 de Data General (h)
5	1.17	2.53	1.12
8	-0.33	2.56	4.13

3.2 Exemples de flux variables sur différentes géométries

Pour effectuer l'ablation de solides d'aluminium, nous avons supposé trois distributions de flux de chaleur différentes.

Dû à l'algorithme que nous avons développé pour mailler le domaine d'intégration, le choix du problème d'ablation susceptible d'être simulé est limité. En effet, du début de la simulation jusqu'à ce que tout le solide ait atteint la température de fusion, la frontière du solide doit être telle qu'il n'y ait pas de concavité, en ce sens qu'il n'y ait pas de droite, partant du centre de gravité du domaine d'intégration et allant jusqu'à un des noeuds de la frontière, qui sorte à l'extérieur du domaine.

3.2.1 Ablation d'un demi-cylindre elliptique d'aluminium

Avant de donner un exemple de solide où une concavité importante survient lors de la simulation, considérons un demi-cylindre elliptique d'aluminium de longueur supposée infinie dont l'équation est donnée par:

$$\frac{x_o^2}{(0.6)^2} + \frac{y_o^2}{(0.4)^2} = 1 \qquad \text{avec } y_o \geq 0 \tag{3.3}$$

Cette géométrie ne possède pas de problème de concavité lors de la simulation pour les distributions de flux de chaleur que nous avons choisies. Il aurait été possible de considérer un demi-cylindre circulaire, mais nous voulions voir la fiabilité du modèle avec une forme géométrique différente de celle utilisée à la section (3.1).

La température initiale du demi-cylindre elliptique est uniforme à 300 K (26.8 C) avec une frontière adiabatique située sur la face latérale plane du demi-cylindre i.e. à y = 0, $k \frac{\partial T(\vec{r},t)}{\partial n} = 0$ (voir fig. 3.14).

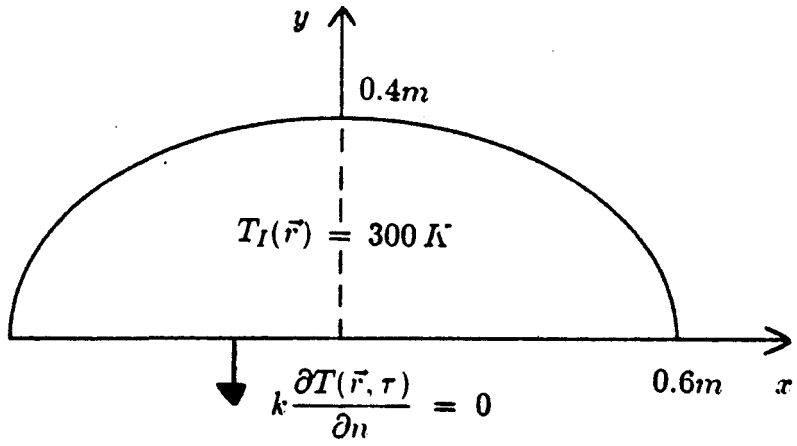


Figure 3.14: Définition du problème utilisant un demi-cylindre elliptique d'aluminium.

Pour les trois distributions de flux utilisées dans cette section, nous supposons que la source de chaleur provient du toit d'un four de grandes dimensions de sorte que le flux de chaleur se dirige vers le bas (l'influence des murs est négligeable par rapport à celle du toit). Puisque le solide est un demi-cylindre elliptique à température uniforme,

l'axe y_o est un axe de symétrie. De même la face latérale plane du demi-cylindre étant une frontière adiabatique, l'axe x_o agit également comme un axe de symétrie. La situation est identique à celle où nous imposerions par symétrie le même flux de chaleur sur l'autre demi-cylindre elliptique imaginaire et situé en dessous de l'axe x_o. Par conséquent, tout comme le cas du cylindre circulaire de la section 3.1, nous solutionnons sur le quart de l'ellipse, en utilisant le processus de condensation directe avec intégration sur les éléments réfléchis, défini à la section 3.1.3.

Pour éviter des problèmes de déplacement de noeuds situés sur la frontière lorsque le solide devient petit, nous arrêtons la simulation lorsque le volume du solide final ne représente plus que 1% du volume initial.

Pour solutionner numériquement, le quart de l'ellipse est discrétisé en 6 éléments linéaires et le domaine en 384 éléments triangulaires utilisant la configuration triangulaire de la figure 3.2c avec 6 couches de triangles. Le pas de temps utilisé pour la phase de préfusion est 30 s et celui utilisé pour la phase de fusion est 60 s. Les propriétés thermiques sont les mêmes que celles utilisées à la section 3.1.

À noter que toutes les simulations apparaissant dans cette section sont effectuées en utilisant un VAX-II/785 de Digital qui est envrion 1.6 fois plus rapide qu'un MV8000 de Data General.

Comme première distribution de flux de chaleur provenant du toit, nous supposons un flux uniforme de 100 kW/m^2. De plus, nous supposons que la distance existant entre le toit et le solide n'a pas d'influence sur le flux. Par conséquent, le flux de chaleur entrant à chaque noeud du quart de l'ellipse est donné par:

$$q\vec{m}^i = \left(-1.0 \times 10^5 \vec{j} \cdot \vec{m}^i\right) \vec{m}^i \qquad (3.4)$$

où $\vec{m}^i$ est tel que défini à l'équation (2.15) et $\vec{j}$ est le vecteur unitaire suivant la direction positive de l'axe y_o.

La figure 3.15 montre que, lors de la phase de préfusion, les résultats obtenus pour la conduction à l'intérieur du demi-cylindre elliptique avec la distribution de flux définie en (3.4), sont excellents.

La figure 3.16 montre à toutes les 5 minutes la position de la frontière du solide. On y voit le temps qu'a pris le domaine tout entier pour atteindre la température de fusion, et aussi la position de la frontière du solide dont le volume ne représente plus que 1% du volume initial. En se référant à cette même figure, on voit que le demi-cylindre elliptique commence à fondre en premier sur le dessus, ce qui est normal puisque le flux de chaleur entrant dans le domaine est maximal (100 kW/m^2) au noeud (7). Au noeud (1) le solide ne peut fondre puisque $q\,\vec{m}^1 = 0\,\vec{m}^1$ W/m^2. Cependant, lorsque le noeud (2) atteint l'axe des x_o, il n'y a plus de solide entre le noeud (2) et le noeud (1), nous devons donc enlever l'élément (1) et renuméroter les noeuds de la frontière de 1 à 6. Ce processus est évidemment répété lorsque le nouveau noeud (2)

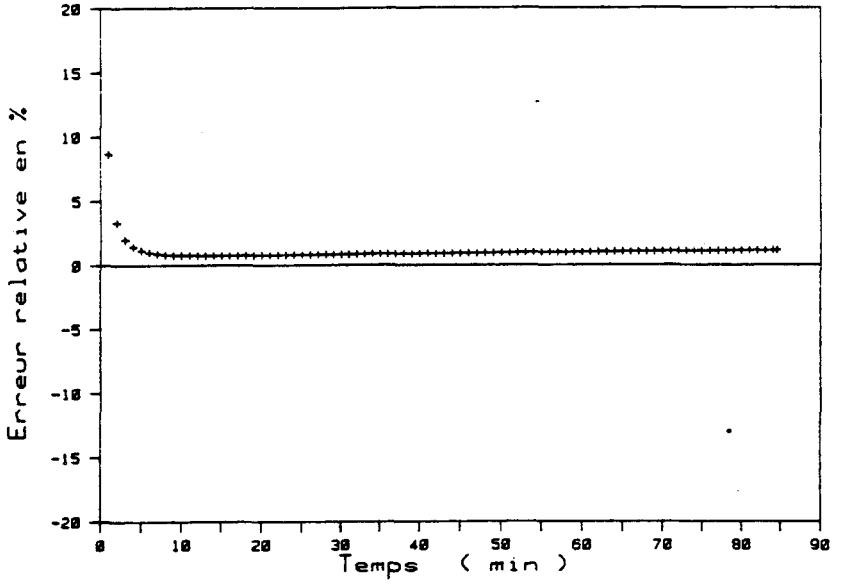


Figure 3.15: Erreur relative sur le bilan d'énergie net calculée à chaque minute jusqu'à ce que le point de fusion soit atteint à la frontière du demi-cylindre elliptique, en utilisant la distribution de flux de chaleur définie en (3.4).

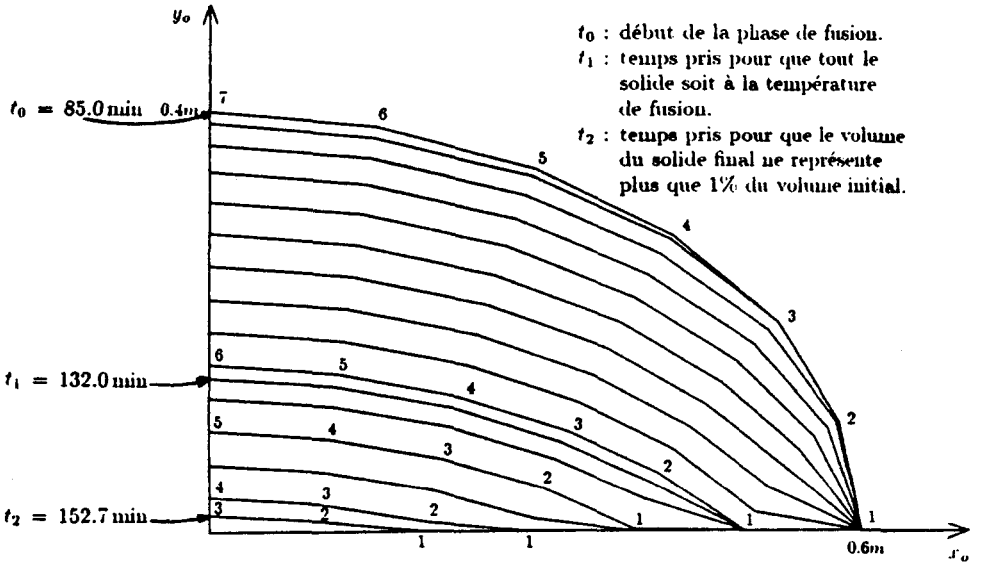


Figure 3.16: Position de la frontière du solide donnée à toutes les 5 minutes; celle où tout le solide est à la température de fusion et celle où le volume du solide final ne représente plus que 1% du volume initial. Le flux utilisé est celui défini en (3.4).

arrive sur l'axe x_o. En fait, à la fin de la simulation, le nombre d'éléments discrétisant la frontière du quart de l'ellipse est tombé à deux.

Même si les éléments linéaires ne représentent que de façon approximative la forme réelle de la frontière du solide, ils nous donnent une bonne indication de la façon dont le solide se déforme et aussi de l'emplacement de la frontière en fonction du temps. En effet, l'erreur relative sur le bilan d'énergie net de tout le procédé d'ablation est seulement de 0.29% (voir la sortie des résultats donnés par le modèle à la table 3.5).

Comme deuxième distribution de flux, nous considérons un flux variant spacialement. Pour cela, nous supposons que le toit est situé à une distance de 2 mètres par rapport au plancher. De plus, nous supposons un flux variant de façon inversement proportionnelle au carré de la distance séparant le toit du solide. Ainsi, le flux entrant à chaque noeud du quart de l'ellipse est donné par:

$$q\vec{m}^i = \left(\left(\frac{-2.0 \times 10^5}{(2.0 - y_o^i)^2} \right) \vec{j} \cdot \vec{m}^i \right) \vec{m}^i \qquad (3.5)$$

où y_o^i est la composante en y_o des coordonnées du noeud (i).

TABLE 3.5: Données du problème pour simuler l'ablation d'un demi-cylindre elliptique utilisant la distribution de flux définie en (3.4) et résultats de la simulation donnés par le modèle.

<<<<< DONNEES DU PROBLEME >>>>>

PAS DE TEMPS POUR LA PREFUSION (s)	:	30
PAS DE TEMPS POUR LA FUSION (s)	:	60
VALEUR DE LA CONDUCTION THERMIQUE (W/(m K))	:	232.0
VALEUR DE LA DENSITE (kg/m**3)	:	2645.0
VALEUR DE LA CHALEUR SPECIFIQUE (J/(kg K)	:	1054.0
VALEUR DE LA DIFFUSIVITE (m**2/SEC.)	:	8.3E-05
CHALEUR LATENTE DE FUSION i.e. RHO_L (J/m**3)	:	9.53E+08
TEMPERATURE DE FUSION (C)	:	660.0
TEMPERATURE INITIALE DU DOMAINE (C)	:	26.8

<<<<< RESULTATS DE LA SIMULATION >>>>>

DUREE DE LA PHASE DE PREFUSION	:	84.50 min.
ERREUR RELATIVE COMMISE SUR BILAN D'ENERGIE NET DE LA PHASE DE PREFUSION	:	0.80 %
ERREUR RELATIVE COMMISE SUR LE BILAN D'ENERGIE NET DE TOUT LE PROCEDE D'ABLATION JUSQU'A CE QU'IL NE RESTE PLUS QUE 0.96 % DU SOLIDE INITIAL	:	0.29 %
TEMPS REEL DE SIMULATION	:	152.67 min.
		(h:min: s)
TEMPS DE CALCUL SUR UN VAX-II/785	:	0: 47:31.26

Malgré le fait que la valeur nominale du flux de chaleur provenant du toit est deux fois plus élevée que celui du premier cas, la distribution du flux à la surface du solide est moins importante. En effet, elle part de zéro au noeud (1) et va jusqu'à 78 kW/m^2 au noeud (7). Par conséquent, il est tout à fait naturel de s'attendre à ce que le temps de simulation soit plus élevé que celui utilisant le flux défini en (3.4) (voir fig. 3.17 et 3.18 et table 3.6).

En examinant le temps de calcul de la simulation utilisant le flux défini en (3.4), nous constatons (table 3.5) qu'il est plus grand que celui de la simulation utilisant le flux défini en (3.5) (table 3.6) et cela même si le temps réel à simuler est plus petit. La raison de ce phénomène est dû à la technique que nous utilisons pour diminuer le nombre d'éléments triangulaires lors de la phase de fusion. En effet, le gradient de température à l'intérieur du domaine étant plus élevé avec l'utilisation du flux défini en (3.4) que celui utilisant le flux défini en (3.5), la diminution du nombre d'éléments triangulaires dans le premier cas est plus lente que dans le second cas, nécessitant par le fait même un temps de calcul plus élevé.

Comme troisième distribution de flux, nous considérons la même situation que le cas précédent sauf que le flux de chaleur provenant du toit est inversement proportionnel à l'exponentiel affecté du carré de la distance séparant le toit du solide. Cependant, cette distance est

divisée par la hauteur du toit pour éviter une diminution trop rapide du flux de chaleur avant qu'il n'atteigne le solide. Ainsi, le flux de chaleur entrant à chaque noeud du quart de l'ellipse est donné par:

$$q\vec{m}^i = \left(-2.0 \times 10^5 \exp\left(-\left(\frac{2.0 - y_o^i}{2.0}\right)^2\right) \vec{j} \cdot \vec{m}^i\right) \vec{m}^i \qquad (3.6)$$

En se référant aux figures 3.19 et 3.20 et à la table 3.7, nous pouvons voir que les résultats obtenus en utilisant cette distribution de flux sont similaires à ceux obtenus avec la première distribution de flux définie en (3.4). En effet, la distribution part de zéro au noeud (1) et va jusqu'à 105 kW/m^2 au noeud (7). Cependant, cette distribution diminue au fur et à mesure que le solide subit l'ablation, ce qui explique un temps de simulation plus long que celui utilisant la distribution définie en (3.4) (voir tables 3.5 et 3.7).

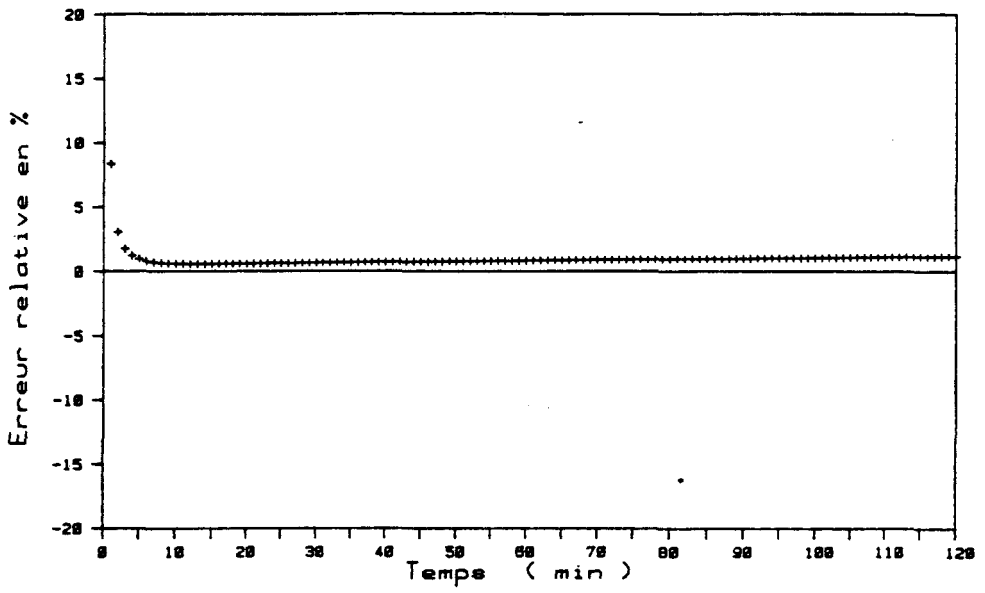


Figure 3.17: Erreur relative sur le bilan d'énergie net calculée à chaque minute jusqu'à ce que le point de fusion soit atteint à la frontière du demi-cylindre elliptique utilisant la distribution de flux de chaleur définie en (3.5).

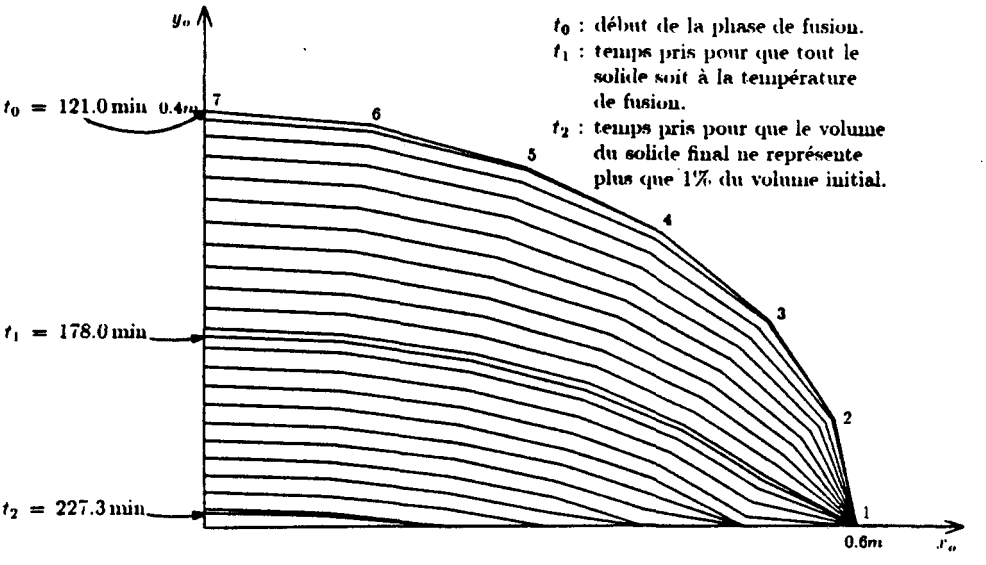


Figure 3.18: Position de la frontière du solide donnée à toutes les 5 minutes, celle où tout le solide est à la température de fusion et celle où le volume du solide final ne représente plus que 1% du volume initial. Le flux utilisé est celui défini en (3.5).

TABLE 3.6: Données du problème pour simuler l'ablation d'un demi-cylindre elliptique utilisant la distribution de flux définie en (3.5) et résultats de la simulation donnés par le modèle.

<<<<< DONNEES DU PROBLEME >>>>>

PAS DE TEMPS POUR LA PREFUSION (s)	:	30
PAS DE TEMPS POUR LA FUSION (s)	:	60
VALEUR DE LA CONDUCTION THERMIQUE (W/(m K))	:	232.0
VALEUR DE LA DENSITE (kg/m**3)	:	2645.0
VALEUR DE LA CHALEUR SPECIFIQUE (J/(kg K)	:	1054.0
VALEUR DE LA DIFFUSIVITE (m**2/SEC.)	:	8.3E-05
CHALEUR LATENTE DE FUSION i.e. RHO_L (J/m**3)	:	9.53E+08
TEMPERATURE DE FUSION (C)	:	660.0
TEMPERATURE INITIALE DU DOMAINE (C)	:	26.8

<<<<< RESULTATS DE LA SIMULATION >>>>>

DUREE DE LA PHASE DE PREFUSION	:	120.50 min.
ERREUR RELATIVE COMMISE SUR BILAN D'ENERGIE NET DE LA PHASE DE PREFUSION	:	0.67 %
ERREUR RELATIVE COMMISE SUR LE BILAN D'ENERGIE NET DE TOUT LE PROCEDE D'ABLATION JUSQU'A CE QU'IL NE RESTE PLUS QUE 0.99 % DU SOLIDE INITIAL	:	0.30 %
TEMPS REEL DE SIMULATION	:	227.25 min.
		(h:min: s)
TEMPS DE CALCUL SUR UN VAX-II/785	:	0: 44:17.44

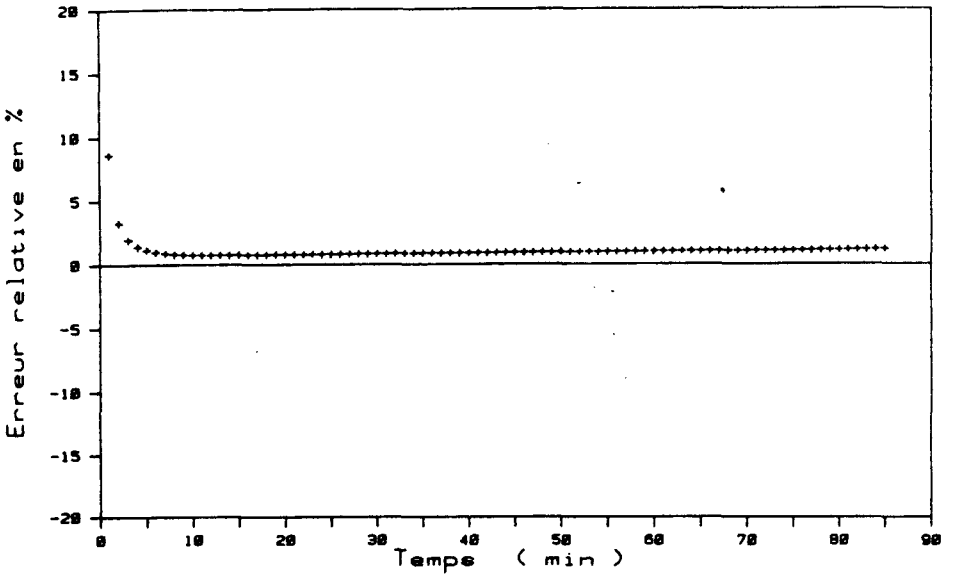


Figure 3.19: Erreur relative sur le bilan d'énergie net calculée à chaque minute jusqu'à ce que le point de fusion soit atteint à la frontière du demi-cylindre elliptique utilisant la distribution de flux de chaleur définie en (3.6).

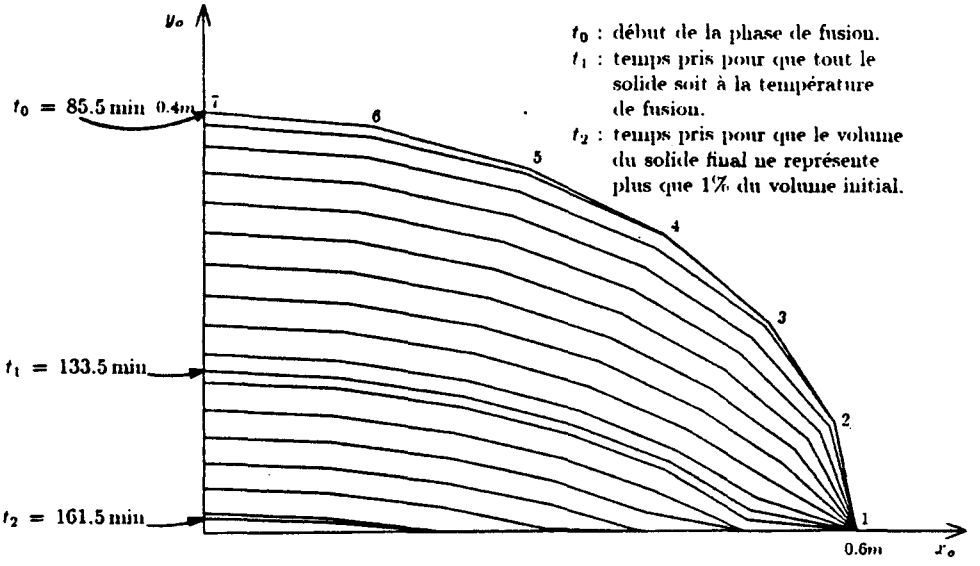


Figure 3.20: Position de la frontière du solide donnée à toutes les 5 minutes; celle où tout le solide est à la température de fusion et celle où le volume du solide final ne représente plus que 1% du volume initial. Le flux utilisé est celui défini en (3.6).

TABLE 3.7: Données du problème pous simuler l'ablation d'un demi-cylindre elliptique utilisant la distribution de flux définie en (3.6) et résultats de la simulation donnés par le modèle.

<<<<< DONNEES DU PROBLEME >>>>>

PAS DE TEMPS POUR LA PREFUSION (s)	:	30
PAS DE TEMPS POUR LA FUSION (s)	:	60
VALEUR DE LA CONDUCTION THERMIQUE (W/(m K))	:	232.0
VALEUR DE LA DENSITE (kg/m**3)	:	2645.0
VALEUR DE LA CHALEUR SPECIFIQUE (J/(kg K)	:	1054.0
VALEUR DE LA DIFFUSIVITE (m**2/SEC.)	:	8.3E-05
CHALEUR LATENTE DE FUSION i.e. RHO_L (J/m**3)	:	9.53E+08
TEMPERATURE DE FUSION (C)	:	660.0
TEMPERATURE INITIALE DU DOMAINE (C)	:	26.8

<<<<< RESULTATS DE LA SIMULATION >>>>>

DUREE DE LA PHASE DE PREFUSION	:	85.00 min.
ERREUR RELATIVE COMMISE SUR BILAN D'ENERGIE NET DE LA PHASE DE PREFUSION	:	0.81 %
ERREUR RELATIVE COMMISE SUR LE BILAN D'ENERGIE NET DE TOUT LE PROCEDE D'ABLATION JUSQU'A CE QU'IL NE RESTE PLUS QUE 0.98 % DU SOLIDE INITIAL	:	0.42 %
TEMPS REEL DE SIMULATION	:	161.50 min.
		(h:min: s)
TEMPS DE CALCUL SUR UN VAX-II/785	:	0: 48:10.31

3.2.2 Exemple d'un problème d'ablation ne pouvant être solutionné en utilisant la configuration triangulaire proposée

Comme exemple d'un problème d'ablation ne pouvant être solutionné en utilisant la configuration triangulaire de la figure 3.2c (i.e. il y a une droite partant du centre de gravité du domaine d'intégration et allant à un des noeuds de la frontière qui sort du domaine lors de la simulation), nous prenons un parallélépipède rectangulaire d'aluminium de longueur infinie. La section transversale de ce solide a pour dimensions 1.2 m de large et 0.6 m de haut. La température initiale du parallélépipède est uniforme à 300 K avec une frontière adiabatique à la base. Le flux de chaleur imposé aux autres faces est celui défini en (3.4).

Tout comme le cas du demi-cylindre elliptique, nous solutionnons sur la moitié du rectangle (i.e. sur le quart du carré de côté 1.2 m) avec les mêmes paramètres de résolution numérique que le demi-cylindre elliptique (voir fig. 3.21) et les mêmes propriétés thermiques.

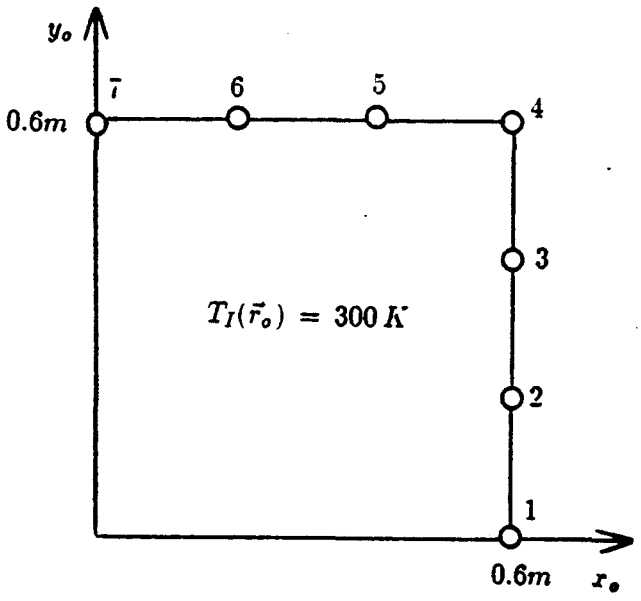


Figure 3.21: Discrétisation de la moitié du rectangle en 6 éléments linéaires.

En se référant à la figure 3.22, on voit que les résultats obtenus pour la phase de préfusion sont satisfaisants. Cependant, la figure 3.23 montre qu'après un certain temps lors de la phase de fusion, la droite partant de l'origine et allant jusqu'au noeud (3) de la dernière frontière, sort du domaine. Par conséquence, le maillage automatique que nous avons développé ne s'applique plus. Pour simuler l'ablation d'un tel solide, un maillage automatique tenant compte de ce genre de concavité serait de mise. Cependant, le développement d'un tel maillage automatique est très complexe et sort du cadre de ce travail.

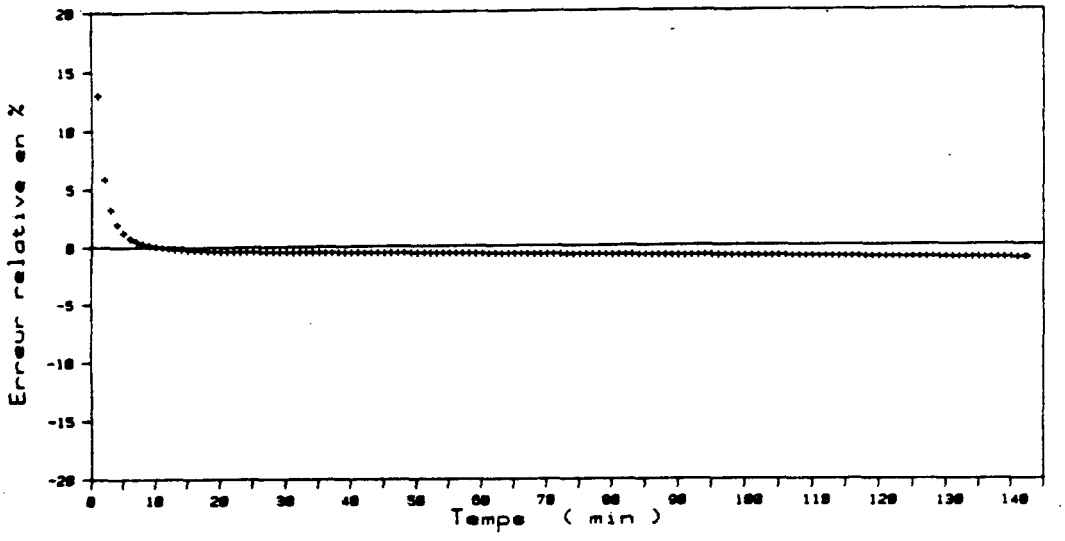


Figure 3.22: Erreur relative sur le bilan d'énergie net calculée à chaque minute jusqu'à ce que le point de fusion soit atteint à la frontière du parallélépipède rectangulaire utilisant la distribution de flux de chaleur définie en (3.4).

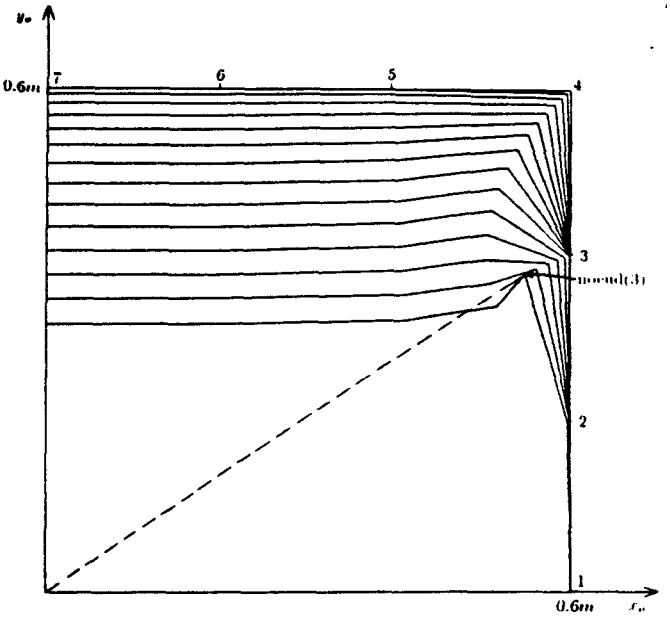


Figure 3.23: Exemple d'un problème d'ablation ne pouvant être simulé en utilisant la configuration triangulaire de la figure 3.2c.

3.3 Résumé du chapitre

Dans ce chapitre, nous avons montré l'importance de choisir une configuration triangulaire et un pas de temps appropriés, spécialement pour la phase de préfusion, pour simuler un problème d'ablation utilisant la méthode des éléments finis de frontière. Il a été démontré qu'un léger changement dans la configuration triangulaire (fig. 3.2c versus 3.2b) peut avoir un effet significatif sur la précision des résultats, et qu'un pas de temps approprié doit être choisi, ni trop petit (pour éviter des problèmes lors de l'intégration numérique) ni trop grand (pour bien accommoder le haut gradient de température existant au début de la simulation).

Nous avons aussi montré, lors de la phase de fusion, qu'une diminution du nombre d'éléments triangulaires, lorsque le gradient de température à l'intérieur du domaine diminue, laisse la vitesse de déplacement de la frontière pratiquement inchangée, tout en réduisant le temps de calcul de façon considérable. De plus, nous avons vu que le processus de condensation directe avec intégration sur les éléments réfléchis pour tenir compte des plans de symétrie, réduit le temps de calcul de façon appréciable tout en donnant exactement les mêmes résultats que ceux obtenus sans l'utilisation des plans de symétrie.

Une fois la méthode des éléments finis de frontière optimisée, nous avons simulé l'ablation d'un demi-cylindre elliptique d'aluminium en le soumettant à trois différents modèles de flux de chaleur. De par la qualité des résultats obtenus, nous avons montré que la méthode des

éléments finis de frontière peut s'adapter à différentes situations. Une seule limite cependant: dû à la technique utilisée pour le maillage automatique, le solide ne doit pas présenter de concavité importante au cours de l'ablation, la concavité étant définie, pour les besoins de ces travaux, à la section 3-2.

CONCLUSION

Pour étudier le problème d'ablation à deux dimensions, nous avons utilisé la méthode des éléments finis de frontière en supposant les propriétés thermiques constantes. Même si cette méthode nécessite l'évaluation d'une intégrale sur le domaine, elle a le principal avantage de réduire de un la dimension du problème, résultant en un plus petit système d'équations linéaires à solutionner. Des simulations faites avec différents pas de temps, points d'intégration et configurations triangulaires ont montré que la méthode est très sensible à ces trois facteurs et qu'un bon choix doit être fait dans chaque cas.

La qualité des résultats obtenus pour l'ablation d'un cylindre circulaire d'aluminium en utilisant un flux de chaleur uniforme et aussi pour l'ablation d'un demi-cylindre elliptique d'aluminium en utilisant trois différents flux de chaleur, montre que la méthode des éléments finis de frontière donne d'excellents résultats. De plus, le temps de calcul peut être réduit considérablement lorsque des techniques telles que la condensation directe (pour tenir compte des plans de symétrie) et la réduction, lors de la phase de fusion, du nombre d'éléments triangulaires discrétisant le domaine lorsque le gradient de température diminue, sont utilisées, et cela sans pour autant changer les résultats. Toutefois, dû à la reconfiguration automatique qui a été développée dans ce travail, le solide dont on veut simuler l'ablation, ne doit pas présenter de concavité importante tout au long de la simulation.

Le modèle développé offre la possibilité de simuler l'ablation de solides autres que ceux d'aluminium. De plus, il pourrait facilement être couplé à un modèle à deux dimensions donnant une distribution de flux de chaleur à l'intérieur d'un four de fusion. L'algorithme de fonctionnement du programme simulant l'ablation d'un solide apparaît à l'appendice E et le listage en annexe.

BIBLIOGRAPHIE

Abramowitz, M. et Stegun, I.A., Handbook of Mathematical Functions, Dover Publications, New York, 1965.

Banerjee, P.K. et Shaw, R.P., Boundary element formulation for melting and solidification problems, dans Developpements in Boundary Element Methods, Vol. 2, édités par P.K. Banerjee et R.P. Shaw, Applied Science Publ., New Jersey, 1982, pp. 1-18.

Beck, J.V., Green's function solution for transient heat conduction problems, International Journal of Heat and Mass transfer, Vol. 27, numéro 8, 1984, pp. 1235-1244.

Brebbia, C.A., Telles, J.C.F. et Wrobel, L.C., Boundary Element Techniques, Springer-Verlag, Berlin, 1984, pp. 141-176.

Carnahan, B., Luther, H.A. et Wilkes, J.O., Applied Numerical Methods, John Wiley, 1969, pp. 100-127.

Carslaw, H.S. et Jaeger, J.C., Conduction of Heat in Solids, seconde édition, Clarendon Press, Oxford, 1959.

Chuans, Y.K. et Szekely, J., On the use of Green's functions for solving melting or solidification problems, International Journal of Heat and Mass Transfer, Vol. 14, 1971, pp. 1285-1294.

Elliot, C.M. et Ockendon, J.R., Weak and variational methods for moving boundary problems, Research notes in Mathematics, Pitman Advanced Publishing Program, 1982.

Hammer, P.C., Marlowe, O.J. et Stroud, A.H., Numerical integration over simplexes and cones, Mathematical Computation, Vol. 10, 1956, pp. 130-137.

Hong, C.P. Umeda, T. et Kimura, Y., Application of the Boundary Element Method in two and three dimensional unsteady Heat Transfer Problems involving Phase Change: Solidification Problems, Boundary Elements, Proceedings of the Fifth International Conference, édités par C.A. Brebbia, T. Futagami et M. Tanaka, Springer-Verlag, 1983, pp. 153-182.

Landau, H.G., Heat conduction in a melting solid, Quart. Appl. Math., Vol. 8, 1950, pp. 81-94.

Lunardini, V.J., Heat Transfer in Cold Climates, Van Nostrand Reinhold, New York, 1981.

Morse, P.M. et Feshbach, H., Methods of Theoretical Physics, Vol. 1, McGraw-Hill, New York, 1953, pp. 857-869.

Ockendon, J.R., Hodgkins, W.R., Moving boundary problems in heat flow and diffusion, résumés des communications présentées à l'Université d'Oxford du 25 au 27 mars 1974, Les presses de l'Université d'Oxford, 1975.

O'Neill, K., Boundary integral solution of moving boundary phase change problems, International Journal for Numerical Methods in Engineering, Vol. 19, 1983, pp. 1825-1850.

Rohsenow, W.M. et Choi, H.Y., Heat, Mass, and Momentum Transfer, Series in engineering of the physical Sciences, Prentice-Hall, New Jersey, 1961, pp. 122-125.

Rubinstein, L.I., The Stefan problem, American mathematical Society, Providence, 1971.

Shaw, R.P., A boundary integral approach to the one dimensional ablation problem, Boundary Element Methods in Engineering, Proceedings of the Fouth International Seminar, édité par C.A. Brebbia, Springer-Verlag, 1982, pp. 127-140.

Touloukian, Y.S., Thermal Diffusivity, IFI/Plenum, 1973, p. 2.

Tremblay, J., Propriétés thermiques de l'aluminium, Rapport interne pour l'équipe d'ingénierie des systèmes, Université du Québec à Chicoutimi, 1986.

Wilson, D.G., Solomon, A.D. et Boggs, P.T., Moving Boundary Problems, Academic Press, New York, 1978.

Wrobel, L.C. et Brebbia, C.A., Boundary elements in thermal problems, dans Numerical Methods in Heat Transfer, édités par R. Lewis, K. Morgan et O.C. Zienkiewicz, John Wiley, 1981a, pp. 91-113.

Wrobel, L.C. et Brebbia, C.A., Time-dependant potential problem, dans Progress in Boundary Element Methods, Vol. 1, édité par Brebbia, John Wiley, 1981b, pp. 192-212.

APPENDICE A

NOMENCLATURE

c	chaleur massique (J/ kg K);
G^*	fonction de Green de l'équation de diffusion pour un domaine infini;
k	conductivité thermique (W/ m K);
L	chaleur latente (J/kg);
q	flux de chaleur (valeur négative par convention) (W/m^2);
$\frac{d\vec{S}(\vec{r})}{dt}$	vitesse de déplacement de la frontière (m/s);
$\vec{r}$	vecteur position du point d'intérêt (x,y) appelé observateur;
$\vec{r}_o$	vecteur position du point (x_o, y_o) appelé source;
$\vec{R}$	vecteur partant de l'observateur et allant à la source (i.e. $\vec{r}_o - \vec{r}$);
R	module du vecteur $\vec{R}$ (m);
$T(\vec{r},\tau)$	température au point correspondant à $\vec{r}$ au temps τ;
T_f	température de fusion du solide supposée constante (K);
Δt_s	pas de temps correspondant à $t_s - t_{s-1}$ (pas nécessairement constant);
t_s	temps après la somme de s pas de temps;
F	nombre de pas de temps nécessaire à la simulation;
$\vec{n}$	vecteur unitaire normal à la frontière $\Gamma(\tau)$ et pointant vers l'extérieur du domaine $\Omega(\tau)$;
N	nombre d'éléments linéaires isoparamétriques discrétisant la frontière $\Gamma(\tau)$;
M	nombre d'éléments triangulaires constants discrétisant le domaine $\Omega(\tau)$;
l_j	longueur de l'élément $\Gamma_j(\tau)$ (m);

$\vec{m}^i$ vecteur unitaire indiquant la direction en sens inverse du déplacement du $i^{ème}$ noeud de la frontière;

$\Delta S(\vec{r}_o^i)$ longueur du déplacement du $i^{ème}$ noeud de la frontière calculée sur la période de temps Δt_s;

$\vec{r}_{oj}(\xi)$ ensemble des vecteurs positions décrivant l'élément Γ_j $(-1 \leq \xi \leq 1)$;

$R_j^i(\xi)$ module du vecteur partant du $i^{ème}$ noeud de la frontière et allant sur un point situé sur l'élément de la frontière Γ_j (i.e. $||\vec{r}_{oj}(\xi) - \vec{r}^i||$) (m);

$\left(x_o^{(L,k)}, y_o^{(L,k)}\right)$ coordonnées cartésiennes des sommets du $L^{ème}$ triangle (k = 1, 2, 3);

$(x_o(\lambda), y_o(\lambda))^j$ ensemble des points situés sur la droite partant de l'origine et allant au noeud (j) de la frontière;

$R_{L,j}(\theta)$ équation de la droite passant par le côté opposé au sommet (j) du $L^{ème}$ triangle écrite en coordonnées polaires;

$\vec{r}_o^{(L)}$ vecteur position du centre de gravité du $L^{ème}$ triangle;

(x, y) coordonnées cartésiennes;

(x_o, y_o) coordonnées cartésiennes;

(u_o, v_o) coordonnées cartésiennes où l'origine est l'observateur $\vec{r}$ = (x, y).

SYMBOLES

$\frac{\partial}{\partial n}$ dérivée directionnelle suivant la direction du vecteur unitaire $\vec{n}$;

∇ gradient;

∇^2 laplacien;

$\frac{d}{dt}$ variation par rapport au temps;

$\| \ \|$ norme d'un vecteur.

ALPHABET GREC

ρ	densité (kg/m^3);
α	diffusivité thermique (m/s^2)
$\varphi(\vec{r})$	coefficient dépendant de la location de l'observateur $\vec{r} = (x, y)$;
$\delta_{i,j}$	symbole de Kronecker;
$\delta(\ )$	pseudo fonction delta de Dirac;
$\Gamma_j(\tau)$	position du $j^{ème}$ élément linéaire de la frontière au temps τ;
$\Omega_L(\tau)$	position du $L^{ème}$ élément triangulaire du domaine au temps τ;
$\psi_{(m)}$	fonctions d'interpolation linéaire (m = 1, 2);
γ	constante d'Euler.

INDICE SUPÉRIEUR

i, j	correspond au $i^{ème}$ ou $j^{ème}$ noeud de la frontière;
(L)	correspond au $L^{ème}$ triangle du domaine;

INDICE INFÉRIEUR

i, j	correspond au $i^{ème}$ ou $j^{ème}$ élément de la frontière;
o	coordonnées relatives à $\vec{r}_o = (x_o, y_o)$.

APPENDICE B

FORMULATION DE L'ÉQUATION D'ÉNERGIE UTILISANT LA TRANSFORMATION DE KIRCHHOFF ET L'ENTHALPIE SENSIBLE

Dans ce qui va suivre, nous allons montrer comment les équations (1.1) à (1.3) du chapitre I peuvent être réécrites de façon équivalente pour pouvoir considérer la variation de la conductivité k et de la chaleur volumique ρc tout en supposant que la diffusivité α demeure constante. Pour ce faire, il existe deux techniques différentes, soit l'utilisation de la transformation de Kirchhoff et celle de l'enthalpie sensible telles que définies plus bas. Ensuite, nous allons, pour un problème de conduction donné, comparer les différents résultats obtenus en utilisant, soit la transformation de Kirchhoff avec une diffusivité constante, soit la formulation enthalpique avec également une diffusivité constante, soit encore toutes les propriétés thermiques constantes.

B.1 Réécriture de l'équation différentielle (1.1)

Tout d'abord, considérons la transformation de Kirchhoff qui est définie par (Elliot et al, 1982):

$$\theta(T) = \int_{T_f}^{T} k\, dT \; ; \; T \leq T_f \qquad \text{(B.1)}$$

où T_f est la température de fusion du solide. Noter que cette définition s'applique à toute valeur de T, plus petite ou plus grande que T_f, mais dans le contexte du problème d'ablation on est seulement intéressé à $T \leq T_f$.

En dérivant par rapport au temps l'équation (B.1), on obtient:

$$\frac{\partial \theta(T)}{\partial t} = \frac{\partial \theta(T)}{\partial T}\frac{\partial T}{\partial t} = k\frac{\partial T}{\partial t} \tag{B.2}$$

De même, en dérivant deux fois par rapport à l'espace, l'équation (B.1) devient:

$$\begin{aligned}\frac{\partial^2 \theta(T)}{\partial x_i^2} &= \frac{\partial}{\partial x_i}\left(\frac{\partial \theta(T)}{\partial x_i}\right) \\ &= \frac{\partial}{\partial x_i}\left(\frac{\partial \theta(T)}{\partial T}\frac{\partial T}{\partial x_i}\right) \\ &= \frac{\partial}{\partial x_i}\left(k\frac{\partial T}{\partial x_i}\right) \quad (i = 1,2)\end{aligned}$$

ou bien

$$\nabla^2 \theta(T) = \nabla \cdot k \nabla T \tag{B.3}$$

En substituant (B.2) et (B.3) dans l'équation différentielle (1.1), on obtient:

$$\nabla^2 \theta(T) = \frac{1}{\alpha}\frac{\partial \theta(T)}{\partial t} \; : \; \vec{r} \in \Omega(t) \; , \; t > 0 \tag{B.4}$$

avec la condition initiale

$$\theta_I(T) = \theta(T_I(\vec{r})) \; : \; \vec{r} \in \Omega(0) \tag{B.5}$$

et la condition frontière

$$\left(\frac{\partial\theta(T)}{\partial n} + q(\vec{r},t)\right)\vec{n} = \rho L\frac{d\vec{S}(\vec{r})}{dt} \; ; \; \vec{r} \in \Gamma(t) \; , \; t > 0 \qquad \text{(B.6)}$$

L'équation (B.4) se solutionne de la même façon que l'équation (1.4) sauf qu'à la place de T, on obtient un potentiel qui est $\theta(T)$ (Carlsaw et al., 1959). Pour obtenir la température à partir de $\theta(T)$, il suffit de trouver sa fonction inverse ou une approximation de celle-ci. Cette fonction inverse existe toujours puisque $\theta(T)$ est une fonction monotone croissante. En effet, par définition, $\theta(T)$ est une intégrale et de plus k est toujours positif.

Une autre façon de réécrire les équations (1.1) à (1.3) est d'utiliser la transformation de l'enthalpie sensible qui est définie par (Elliot et al., 1982):

$$H(T) = \int_{T_f}^{T} \rho c \, dT \; ; \; T \leq T_f \qquad \text{(B.7)}$$

Toutefois, au lieu d'utiliser les équations (1.1) à (1.3) pour trouver leurs formes équivalentes, nous allons passer directement par les équations (B.4) à (B.6). On a, par définition:

$$dH(T) = \rho c dT \qquad \text{(B.8)}$$

et

$$d\theta(T) = k dT \tag{B.9}$$

d'où, en regroupant (B.8) et (B.9),

$$d\theta(T) = \frac{k}{\rho c} dH(T) = \alpha dH(T)$$

De plus, en intégrant cette dernière équation de T à T_f, où $T \leq T_f$ et en considérant α comme étant constant, on obtient:

$$\theta(T) = \alpha H(T) \tag{B.10}$$

Par conséquent, l'équation différentielle (B.4) devient:

$$\alpha \nabla^2 H(T) = \frac{\partial H(T)}{\partial t} \; ; \; \vec{r} \in \Omega(t) \; , \; t > 0 \tag{B.11}$$

avec la condition initiale

$$H_I(T) = H(T_I(\vec{r})) \; ; \; \vec{r} \in \Omega(0) \tag{B.12}$$

et la condition frontière

$$\left(\alpha \frac{\partial H(T)}{\partial n} + q(\vec{r}, t) \right) \vec{n} = \rho L \frac{d\vec{S}(\vec{r})}{dt} \; ; \; \vec{r} \in \Gamma(t) \; , \; t > 0 \tag{B.13}$$

Il est important de remarquer que les équations (B.4) et (B.11) ont exactement la même forme. Ainsi, qu'on utilise l'une ou l'autre des transformations, l'évolution de la température du solide demeure la même. Cependant, il faut distinguer les deux cas. Lorsque l'on solutionne avec la transformation de Kirchhoff et que l'on considère α constant, il est sous-entendu que la chaleur volumique ρc varie proportionnellement avec la conductivité. C'est l'inverse qui se produit lorsque l'on solutionne avec la formulation utilisant l'enthalpie sensible, i.e. k varie proportionnellement à ρc.

Par contre, si la diffusivité du solide est fonction de la température, alors les résultats obtenus diffèrent suivant que l'on utilise l'une ou l'autre des formulations comme le démontre bien l'exemple numérique qui suit.

B.2 Comparaison entre les différents résultats

Soit un cylindre circulaire d'aluminium de longueur infinie de rayon 0.5 m, dont la température initiale est uniforme à 300 K et auquel on impose un flux constant de 100 kW/m^2 sur la face latérale.

Ce problème est le même que celui de la section 1 du chapitre III sauf que l'on ne considère que la phase de préfusion i.e. jusqu'à ce que la température de fusion soit atteinte à la surface du solide. Pour solutionner ce problème, nous procédons de trois façons différentes.

D'abord, nous supposons que les propriétés thermiques de l'aluminium sont toutes constantes, ensuite nous utilisons la transformation de Kirchhoff i.e. nous tenons compte de la variation de la conductivité en fonction de la température de l'aluminium solide tout en conservant la diffusivité constante et pour terminer, nous utilisons la formulation avec l'enthalpie sensible, qui elle, tient compte de la variation de la chaleur volumique ρc, tout en conservant la diffusivité constante.

Il est important de noter que, dans les trois cas mentionnés ci-haut, nous utilisons une diffusivité moyenne i.e.

$$\alpha = \frac{\int_{T_{min}}^{T_{max}} \alpha(T)\, dT}{T_{max} - T_{min}} = 8.3 \times 10^{-5} \ (\mathrm{m^2/s})$$

où T_{min} représente la température minimum du solide qui est 300 K, T_{max} la température maximale qui est celle de fusion et $\alpha(T)$ l'approximation polynômiale (Tremblay, 1986) de la diffusivité en fonction de la température de l'aluminium solide qui est donnée par:

$$\alpha(T) = -3.69 \times 10^{-12} T^2 - 4.26 \times 10^{-8} T + 1.11 \times 10^{-4} \ (\mathrm{m^2/s})$$

$$\text{pour} \quad 250\,\mathrm{K} \leq T \leq 933.15\,\mathrm{K}$$

Pour comparer les trois façons de procéder, nous regardons l'évolution de la température à la frontière, en fonction du temps, jusqu'à ce que la température de fusion de l'aluminium soit atteinte à la surface.

Dans le premier cas où toutes les propriétés thermiques sont gardées constantes, la solution analytique (Carslaw et al., 1959) est:

$$T(t) = \frac{2F_o\alpha t}{kr} + \frac{F_o r}{k}\left\{\frac{1}{4} - 2\sum_{n=1}^{\infty}\frac{\exp(-\alpha a_n^2 t/r^2)}{a_n^2}\right\} + T_I \ \text{(K)} \qquad \text{(B.14)}$$

où k = 232 W/mK représente la conductivité calculée à la température correspondant à la diffusivité moyenne ($\alpha = 8.3 \times 10^{-5} m^2/s$) à partir de l'approximation polynômiale suivante (Tremblay, 1986):

$$k(T) = -8.65 \times 10^{-5}T^2 + 6.26 \times 10^{-2}T + 226 \ \text{(W/(m K))}$$

$$\text{pour} \quad 250\,\text{K} \leq T \leq 933.15\,\text{K}\,.$$

De plus, F_0 représente le flux imposé à la surface, r le rayon du cylindre et a_n, où n = 1,2,..., les racines positives de la fonction de Bessel du premier ordre.

Pour la transformation de Kirchhoff, la solution analytique est donnée par:

$$\theta(t) = \frac{2F_o\alpha t}{r} + F_o r\left\{\frac{1}{4} - 2\sum_{n=1}^{\infty}\frac{\exp(-\alpha a_n^2 t/r^2)}{a_n^2}\right\} + \theta(T_I) \ \text{(W/m)} \qquad \text{(B.15)}$$

où

$$\theta(T) = -2.88 \times 10^{-5}T^3 + 3.13 \times 10^{-2}T^2 + 226T - 2.15 \times 10^5 \ \text{(W/m)} \qquad \text{(B.16)}$$

$$\text{pour} \quad 250\,\text{K} \leq T \leq 933.15\,\text{K}\,.$$

est une approximation polynômiale (Tremblay, 1986) de la transformation de Kirchhoff.

La différence qui existe entre (B.14) et (B.15), à part le fait qu'on a substitué T par θ(T), est la disparition de la conductivité dans (B.15). En effet, cette différence provient essentiellement de la condition frontière (B.6) où la conductivité n'apparaît pas comparativement à (1.3). Il faut remarquer que, le terme de déplacement de la frontière du solide est nul, i.e.

$$\rho L \frac{d\vec{S}(\vec{r})}{dt} = 0 .$$

puisque dans cette section nous ne considérons qu'un problème de conduction pure, le problème se termine lorsque la frontière atteint la température de fusion.

Une fois que l'on connaît la valeur de θ(T) en fonction du temps, nous devons trouver la température correspondante. Pour ce faire, nous prenons une autre approximation polynômiale servant à représenter la fonction inverse de (B.16). Le polynôme est donné par

$$T(\theta) = 1.01 \times 10^{-14}\theta^3 + 3.89 \times 10^{-9}\theta^2 + 4.71 \times 10^{-3}\theta + 933.15 \ (\mathrm{K})$$

$$\text{pour} \quad \theta \leq 0 .$$

Et pour finir, la solution analytique correspondant à la transformation de l'enthalpie sensible est:

$$H(t) = \frac{2F_o t}{r} \pm \frac{F_o r}{\alpha}\left\{\frac{1}{4} - 2\sum_{n=1}^{\infty}\frac{\exp(-\alpha a_n^2 t/r^2)}{a_n^2}\right\} + H(T_I) \ (\mathrm{J/m^3})$$

puisque c'est la diffusivité qui apparaît dans la condition frontière (B.13) à la place de la conductivité dans (1.3).

De façon similaire à la transformation de Kirchhoff, H(T) est donné par (Tremblay, 1986):

$$H(T) = -0.04T^3 + 652T^2 + 2.04 \times 10^6 T - 2.44 \times 10^9 \ (\mathrm{J/m^3})$$

$$\text{pour} \quad 250\,\mathrm{K} \leq T \leq 933.15 \ \mathrm{K}.$$

et

$$T(H) = 2.77 \times 10^{-27} H^3 - 2.01 \times 10^{-17} H^2 + 3.13 \times 10^{-7} H + 933.15 \ (\mathrm{K})$$

$$\text{pour} \quad H \leq 0 \, .$$

Les résultats des trois solutions analytiques sont donnés à la figure (B).

Comme nous pouvons le constater sur le graphique, les résultats obtenus ne sont pas les mêmes pour les trois cas. Ceci est facilement explicable par le fait que la conductivité k ne varie pas proportionnellement à la chaleur volumique ρc. La valeur de ρc augmente d'environ 27% lorsque la température passe de 300 K à 933.15 K, tandis que la valeur de k diminue d'environ 12% (Tremblay, 1986). Pour cette raison, nous avons préféré opter pour les propriétés thermiques constantes dont la courbe se situe entre celle de la formulation enthalpique et celle de la transformation de Kirchhoff, étant évidemment plus près de cette dernière due à la faible variation de k comparativement à celle de ρc.

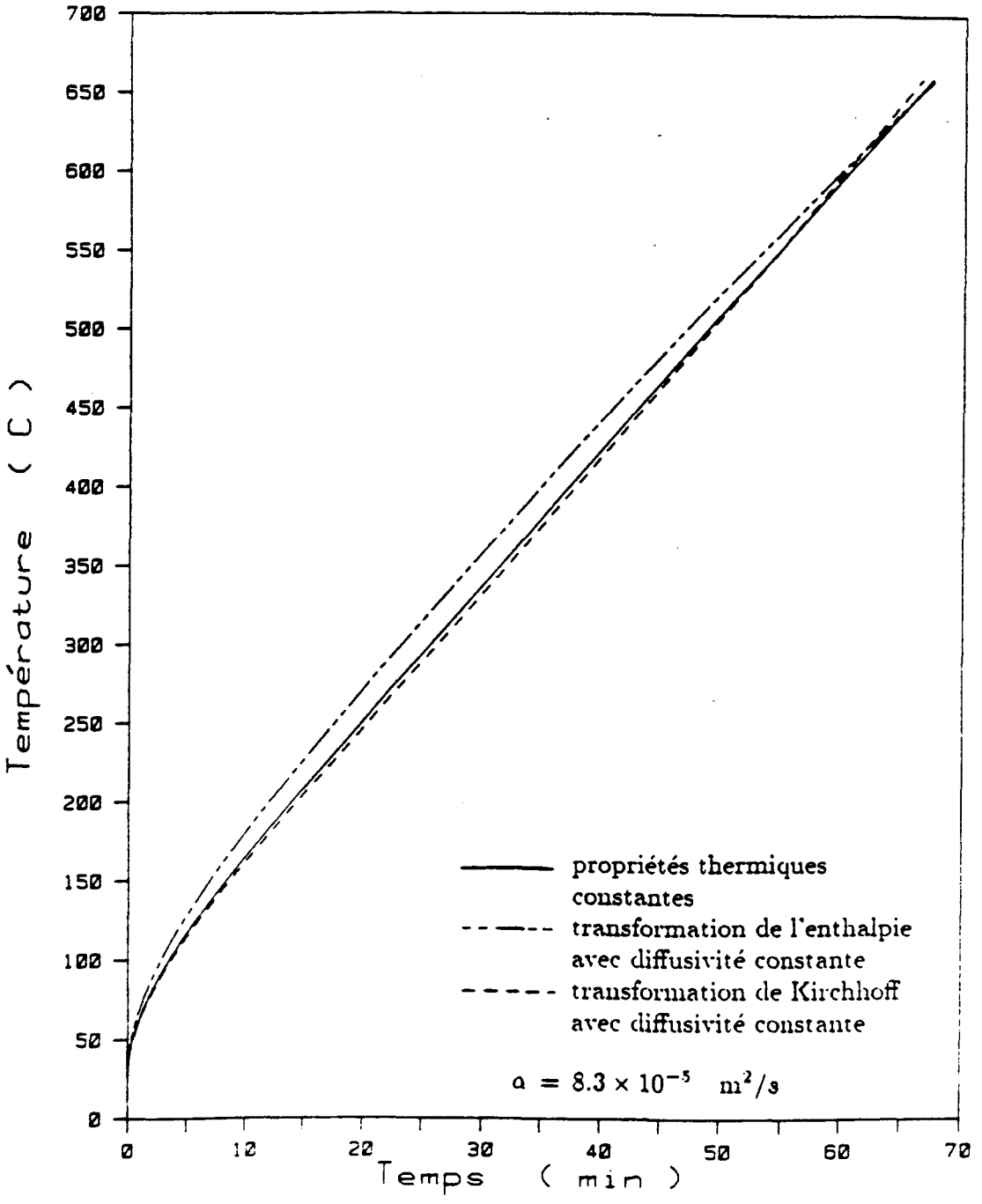


Figure B Comparaison entre les différents résultats obtenus utilisant toutes les propriétés thermiques constantes, la transformation de Kirchhoff avec une diffusivité constante et celle de l'enthalpie sensible avec également une diffusivité constante.

APPENDICE C

COMPLÉMENT À L'INTÉGRATION EN COORDONNÉES POLAIRES DU DOMAINE

C.1 Détermination de l'emplacement de l'observateur par rapport au triangle sur lequel s'effectue l'intégration

En premier lieu, nous vérifions si l'observateur est situé à l'extérieur du triangle sur lequel s'effectue l'intégration que l'on note comme étant le $L^{ème}$ triangle. La technique utilisée pour cela est de déterminer s'il existe au moins une paire de sommets du $L^{ème}$ triangle telle que la droite passant par ces deux sommets se trouve entre l'observateur et le troisième sommet. Par exemple, si l'on définit la paire de sommets comme étant $(x_o^{(L,k)}, y_o^{(L,k)})$ et $(x_o^{(L,\ell)}, y_o^{(L,\ell)})$, alors la façon de déterminer si les coordonnées de l'observateur et celles du troisième sommet $(x_o^{(L,j)} y_o^{(L,j)})$ sont situées de part et d'autre de la droite passant par la paire de sommets, est d'utiliser, en supposant que la droite ne soit pas verticale, l'inégalité suivante (voir fig. C.1a):

$$\left(Y(x_o^{(L,j)}) - y_o^{(L,j)}\right) \times (Y(x) - y) < 0 . \qquad (C.1)$$

Y(X) représente l'équation de la droite passant par $(x_o^{(L,k)}, y_o^{(L,k)})$ et $(x_o^{(L,\ell)}, y_o^{(L,\ell)})$ et est définie par:

$$Y(X) = \left(\frac{y_o^{(L,l)} - y_o^{(L,k)}}{x_o^{(L,l)} - x_o^{(L,k)}} \right) \left(X - x_o^{(L,k)} \right) + y_o^{(L,k)}$$

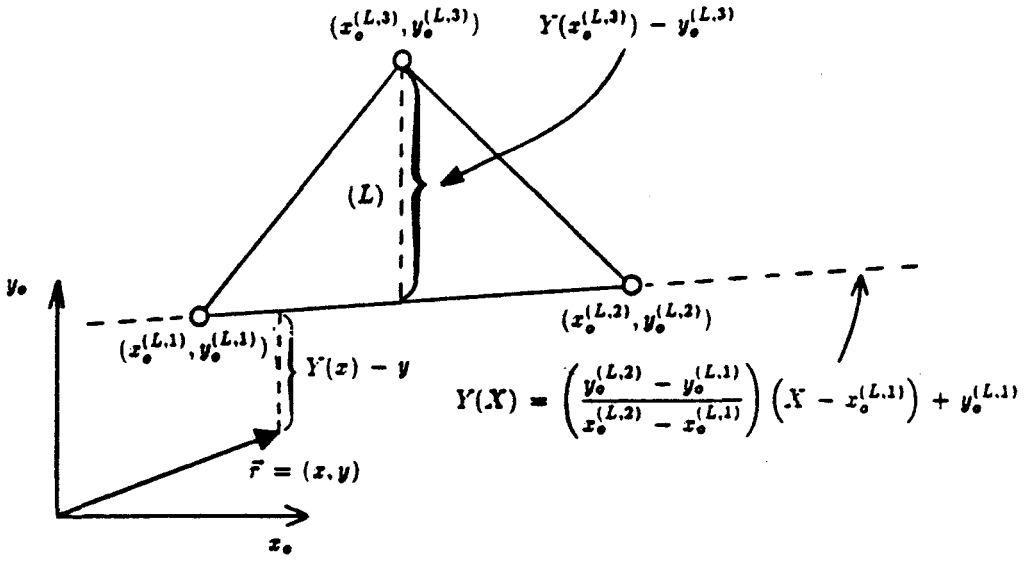
avec $x_o^{(L,l)} - x_o^{(L,k)} \neq 0$ et $j,k,l = 1.2.3$ mais $j \neq k \neq l$.

Dans le cas où la droite est verticale i.e. $x_o^{(L,\ell)} - x_o^{(L,k)} = 0$ (voir fig. C.1b), le test à utiliser est l'inégalité suivante:

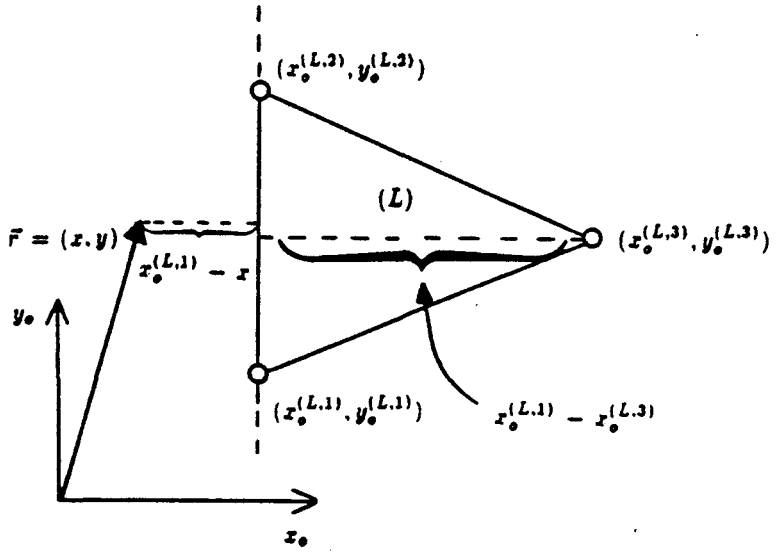
$$\left(x_o^{(L,k)} - x \right) \times \left(x_o^{(L,k)} - x_o^{(L,j)} \right) < 0 . \qquad \text{(C.2)}$$

Si après avoir essayé les trois paires de sommets du $L^{ème}$ triangle, l'inégalité (C.1) ou (C.2) n'est pas vérifiée, alors l'observateur se trouve, soit sur un des sommets, soit sur un des côtés ou encore à l'intérieur du triangle.

Pour déterminer si l'observateur est situé sur un des sommets, il suffit de vérifier si les coordonnées d'un des sommets sont les mêmes que celles de l'observateur. Si ce n'est pas le cas, nous regardons si l'observateur est situé sur un des côtés du triangle. Pour cela, nous vérifions si la somme des distances prises de l'observateur à chacun des sommets correspondant à un côté est égale à la distance entre ces deux sommets qui est, en fait, la longueur du côté (voir fig. C.2). Si, une fois de plus ce cas n'est pas vérifié, alors l'observateur ne peut être qu'à l'intérieur du triangle.



a) $x_o^{(L,2)} - x_o^{(L,1)} \neq 0$



b) $x_o^{(L,2)} - x_o^{(L,1)} = 0$

Figure C.1: Test pour vérifier si l'observateur est situé à l'extérieur du $L^{ème}$ triangle avec j = 3, k = 1, ℓ = 2.

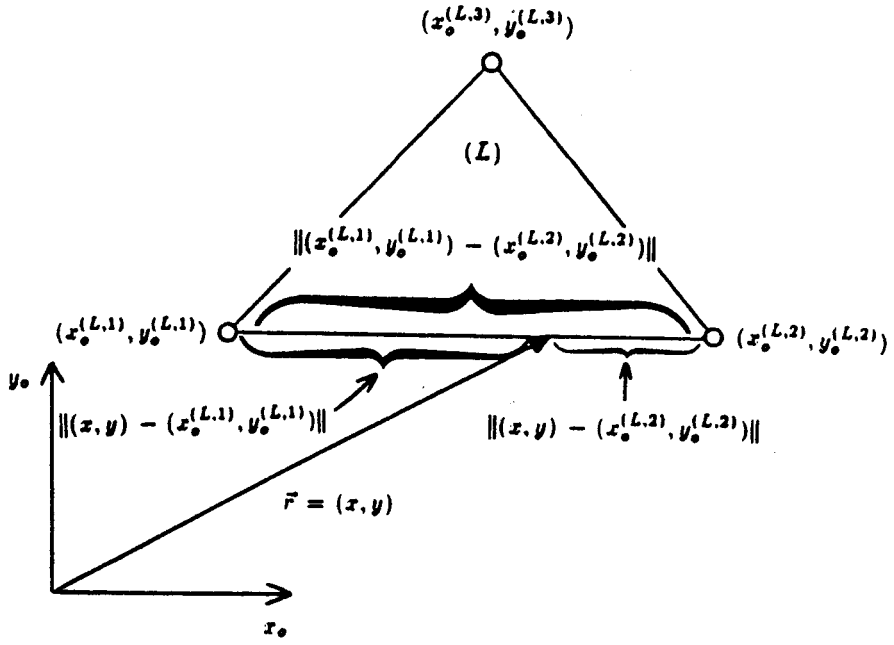


Figure C.2: Test pour vérifier si l'observateur est situé sur un des côtés du $L^{ème}$ triangle.

C.2 Transformation en coordonnées polaires des droites passant par les côtés d'un triangle donné.

La transformation en coordonnées polaires des droites passant par les trois côtés du $L^{ème}$ triangle s'effectue dans le système de coordonnées (u_o, v_o) (voir fig. 2.6). Par conséquent, l'équation de la droite passant par le côté opposé au sommet (j) (j = 1,2,3) est donnée par:

$$\frac{v_o - v_o^{(L,k)}}{u_o - u_o^{(L,k)}} = \frac{v_o^{(L,l)} - v_o^{(L,k)}}{u_o^{(L,l)} - u_o^{(L,k)}}$$

ou bien par

$$\left(v_o - v_o^{(L,k)}\right)\left(u_o^{(L,l)} - u_o^{(L,k)}\right) = \left(u_o - u_o^{(L,k)}\right)\left(v_o^{(L,l)} - v_o^{(L,k)}\right) \qquad (C.3)$$

où k = 2,3,1, ℓ = 3,1,2 lorsque j = 1,2,3

Pour la transformation en coordonnées polaires, nous utilisons l'équation (C.3) puisqu'elle permet de considérer les droites verticales (i.e. $u_o^{(L,\ell)} - u_o^{(L,k)} = 0$). Ainsi, en posant $u_o = R\cos\theta$ et $v_o = R\sin\theta$, l'équation (C.3) devient:

$$\left(R\sin\theta - v_o^{(L,k)}\right)\left(u_o^{(L,l)} - u_o^{(L,k)}\right) = \left(R\cos\theta - u_o^{(L,k)}\right)\left(v_o^{(L,l)} - v_o^{(L,k)}\right) \quad \text{(C.4)}$$

Puisque $u_o = x_o - x$ et que $v_o = y_o - y$, l'équation (C.4) se réécrit (dans le système de coordonnées (x_o,y_o)) sous la forme suivante:

$$R\left(d_{(j)}\cos\theta + c_{(j)}\sin\theta\right) = -\left(x_o^{(L,k)}y_o^{(L,l)} - x_o^{(L,l)}y_o^{(L,k)} + d_{(j)}x + c_{(j)}y\right) \quad \text{(C.5)}$$

où

$$c_{(j)} = x_o^{(L,l)} - x_o^{(L,k)} \quad \text{et} \quad d_{(j)} = y_o^{(L,k)} - y_o^{(L,l)} .$$

Avant de diviser les deux membres de l'équation (C.5) par $d_{(j)}\cos\theta + C_{(j)}\sin\theta$, montrons que lorsque cette expression est nulle, l'équation (C.5) n'a pas besoin d'être utilisée lors de l'intégration en coordonnées polaires du $L^{ème}$ triangle.

En effet, si $d_{(j)}\cos\theta + C_{(j)}\sin\theta = 0$ alors cela implique que

$$\tan\theta = -\frac{d_{(j)}}{c_{(j)}} = \frac{y_o^{(L,l)} - y_o^{(L,k)}}{x_o^{(L,l)} - x_o^{(L,k)}} . \quad \text{(C.6)}$$

Puisque θ est l'angle que fait le vecteur $R_{L,j}$ (voir fig. C.3) avec la direction positive de l'axe u_o, il s'en suit immédiatement que la droite passant par le côté opposé au sommet (j) passe également par l'origine de (u_o,v_o) lorsque l'équation (C.6) est satisfaite. Par conséquent, pour tout point situé sur un côté qui est aligné avec l'observateur,

l'angle Θ demeure toujours constant, d'où la non nécessité d'utiliser ce côté pour l'intégration en coordonnées polaires du triangle.

Par conséquent, tous les côtés indispensables à l'intégration en coordonnées polaires du $L^{ème}$ triangle peuvent s'exprimer sous la forme suivante:

$$R_{L,j} = \frac{-\left(x_o^{(L,k)} y_o^{(L,l)} - x_o^{(L,l)} y_o^{(L,k)} + d_{(j)}x + c_{(j)}y\right)}{d_{(j)}\cos\theta + c_{(j)}\sin\theta} . \qquad (C.7)$$

où $R_{L,j}(\Theta)$ est l'équation en coordonnées polaires de la doite passant par le côté opposé au sommet (j) du $L^{ème}$ triangle.

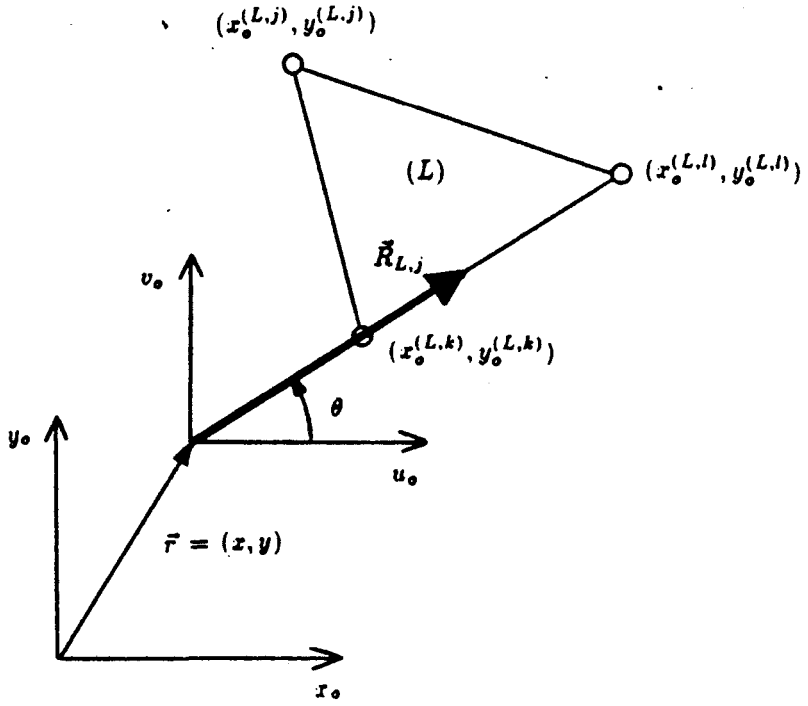


Figure C.3: Exemple d'un triangle dont un des côtés est aligné avec l'observateur.

APPENDICE D

CALCUL DE L'ERREUR RELATIVE SUR LES BILANS D'ÉNERGIE NET

D.1 Phase de préfusion

Pour avoir une bonne idée de la tendance des résultats lors de la phase de préfusion, nous calculons à chaque minute l'erreur relative commise sur le bilan d'énergie net i.e.

$$\text{Erreur relative (pour 60 s)} = \left(\frac{E_a^{60} - E_f^{60}}{E_f^{60}} \right) \times 100\ \% \qquad \text{(D.1)}$$

où

E_a^{60} = l'énergie absorbée par le solide durant une minute;

et

E_f^{60} = l'énergie fournie au solide durant une minute.

La raison pour laquelle nous ne prenons pas la valeur absolue de l'erreur relative est pour nous indiquer que lorsque l'erreur relative calculée à chaque minute est positive, la température augmente plus rapidement que la réalité (il y a plus d'énergie absorbée que d'énergie fournie). Par contre, lorsque l'erreur relative calculée à chaque minute est négative, c'est l'inverse qui se produit (il y a moins d'énergie absorbée que d'énergie fournie).

Puisque nous considérons un solide tridimentionnel dont une des dimensions est supposée infinie de sorte que l'échange thermique est bidimentionnel, nous supposons, pour les besoins du calcul du bilan d'énergie net, que le solide possède une longueur unitaire.

Le domaine étant discrétisé en M éléments triangulaires constants, l'énergie absorbée par le solide pendant une minute est donnée par

$$E_a^{60} = \int_{\Omega(0)} (\rho c(T(\vec{r}_o, t_u) - T(\vec{r}_o, t_u - 60)))\, d\Omega$$
$$= \rho c \sum_{L=1}^{M} \left(A_T(L)(T(\vec{r}_o^{(L)}, t_u) - T(\vec{r}_o^{(L)}, t_u - 60)) \right) \quad (\mathrm{J}) \qquad (\mathrm{D.2})$$

où $\Omega(0)$ est le domaine de la section transversale du solide lors de la phase de préfusion, $A_T(L)$ l'aire du $L^{\text{ème}}$ triangle de la configuration triangulaire discrétisant ce domaine, $\vec{r}_o^{(L)}$ le vecteur position correspondant au centre de gravité du $L^{\text{ème}}$ triangle, t_u un multiple positif de 60s. De plus, ρ et c représentent respectivement la densité et la chaleur massique de l'aluminium et sont calculées à la température correspondant à la valeur moyenne de la diffusivité (voir appendice B) à partir des approximations polynomiales suivantes (Tremblay, 1986):

$$\rho(T) = -2.19 \times 10^{-4} T^2 + 0.104T + 2666 \quad (\mathrm{kg/m^3})$$

et

$$c(T) = 0.459T + 767 \quad (\mathrm{J/(kg\,K)})$$

$$\text{pour} \quad 250\,\mathrm{K} \leq T \leq 933.15\,\mathrm{K}$$

De même la frontière étant discrétisée en N éléments linéaires isoparamétriques et le flux de chaleur imposé à la face latérale du solide

étant négatif par convention, l'énergie fournie pendant une minute est donnée par:

$$
\begin{aligned}
E_f^{60} &= \int_{t_u-60}^{t_u}\int_{\Gamma(0)} (-q(\vec{r}_o,\tau))\, d\Gamma\, d\tau \\
&= \int_{t_u-60}^{t_u} \left(\sum_{j=1}^{N}\int_{\Gamma_j(0)} (-q(\vec{r}_o,\tau))\, d\Gamma\right) d\tau \\
&= -\int_{t_u-60}^{t_u} \left(\sum_{j=1}^{N}\frac{l_j}{2}\int_{-1}^{1}\left(\left(\frac{1-\xi}{2}\right) q(\vec{r}_o^{\,j},\tau) + \left(\frac{1+\xi}{2}\right) q(\vec{r}_o^{\,j+1},\tau)\right) d\xi\right) d\tau \\
&= -\frac{1}{2}\int_{t_u-60}^{t_u} \left(\sum_{j=1}^{N} l_j \left(q(\vec{r}_o^{\,j},\tau) + q(\vec{r}_o^{\,j+1},\tau)\right)\right) d\tau \quad \text{(J)}
\end{aligned}
\tag{D.3}
$$

où ℓ_j représente la longueur du $j^{\text{ème}}$ élément linéaire ($\Gamma_j(0)$) lors de la phase de préfusion, $\vec{r}_o^{\,i}$ ($i = 1,2,..,N$ et $\vec{r}_o^{\,N+1} = \vec{r}_o^{\,1}$) le vecteur position du $i^{\text{ème}}$ noeud de la frontière.

De plus, le flux chaleur étant supposé constant sur chaque pas de temps ($\Delta t_s = t_s - t_{s-1}$), l'équation (D.3) devient:

$$
E_f^{60} = -\frac{1}{2}\sum_{s=u-p+1}^{u}\left(\sum_{j=1}^{N} l_j \left(q(\vec{r}_o^{\,j},t_s) + q(\vec{r}_o^{\,j+1},t_s)\right)\right)(t_s - t_{s-1}) \quad \text{(J)} \tag{D.4}
$$

où u est un multiple positif de p, avec $p = 60/\Delta t_s$, Δt_s étant évidemment choisi pour être un diviseur de 60s.

D.2 Phase de fusion

Pour la phase de fusion, nous ne pouvons pas calculer à chaque minute l'erreur relative sur le bilan d'énergie net. En effet, le solide se déformant et la partie fondue étant enlevée immédiatement, il est très difficile de calculer avec précision l'énergie absorbée par le solide pour une période donnée lors de phase de fusion. De plus, lorsque le solide est à la température de fusion, il est difficile d'estimer la chaleur latente accumulée par le solide. Par conséquent, pour avoir une bonne idée sur la précision des résultats obtenus lors de cette phase, nous calculons l'erreur relative commise sur le bilan d'énergie net de tout le procédé d'ablation jusqu'à ce que le volume du solide final ne représente plus que 1% du volume initial, i.e.

$$\text{Erreur relative globale} = \left(\frac{E_a - E_f}{E_f}\right) \times 100\ \% \qquad (D.5)$$

où

E_a = l'énergie nécessaire pour prendre la température initiale du solide et la monter à celle de fusion (T_f)
+ l'énergie nécessaire pour faire fondre le solide jusqu'à ce qu'il ne représente plus, en volume, que 1% du volume initial,

et

E_f = l'énergie fournie au solide durant tout le procédé d'ablation.

Ainsi, l'énergie absorbée lors de la simulation est donnée par:

$$
\begin{aligned}
E_a &= \int_{\Omega(0)} (\rho c(T_f - T(\vec{r}_o, 0)))\, d\Omega + \rho L \left(\int_{\Omega(0)} d\Omega - \int_{\Omega(t_F)} d\Omega \right) \\
&= \rho c \sum_{L=1}^{M} \left(A_T(L)(T_f - T(\vec{r}_o^{(L)}, 0)) \right) \\
&\quad + \rho L \left(\sum_{L=1}^{M} A_T(L) - \sum_{L=1}^{M_F} A_T^F(L) \right) \quad \text{(J)}
\end{aligned}
\tag{D.6}
$$

où t_F est le temps final de simulation (i.e. le temps nécessaire pour que le volume du solide final ne représente plus que 1% du volume initial), M_F le nombre d'éléments triangulaires constants discrétisant le domaine final et $A_T^F(L)$ l'aire du $L^{\text{ème}}$ triangle de la configuration triangulaire discrétisant ce domaine.

L'équation (D.6), donnant l'énergie absorbée par le solide lors de la simulation, ne considère pas la chaleur latente pouvant être accumulée par le solide restant, cependant, l'erreur maximale pouvant être commise en ne considérant pas cette accumulation de chaleur latente est de l'ordre de 0.4% pour les solides d'aluminium dont la température initiale est uniforme à 300 K.

De même l'énergie fournie lors de la simulation est donnée par:

$$
\begin{aligned}
E_f &= \int_0^{t_F} \int_{\Gamma(\tau)} (-q(\vec{r}_o, \tau))\, d\Gamma\, d\tau \\
&= -\frac{1}{2} \sum_{s=1}^{F} \left(\sum_{j=1}^{N} l_j(t_s) \left(q(\vec{r}_o^{\,j}, t_s) + q(\vec{r}_o^{\,j+1}, t_s) \right) \right) (t_s - t_{s-1}) \quad \text{(J)}
\end{aligned}
\tag{D.7}
$$

où $\ell_j(t_s)$ représente la longueur du $j^{ème}$ élément ($\Gamma_j(t_s)$) discrétisant la frontière ($\Gamma(t_s)$) au temps t_s.

APPENDICE E

ORGANISATION GÉNÉRALE DU PROGRAMME

Le programme appelé ABLATION et servant à simuler un problème d'ablation à deux dimensions est un programme écrit en Fortran-77 et comprend 115 200 octets. La version apparaissant en annexe est une version compatible avec les ordinateurs de la série VAX-II.

E.1 Création des fichiers d'entrée et du sous-programme donnant la distribution de flux de chaleur aux noeuds de la frontière.

Avant d'exécuter le programme ABLATION, deux fichiers appelés respectivement ENTREE.DAT et COORD_FRONT.DAT, doivent être créés. Le premier fichier (i.e. ENTREE.DAT) doit avoir, dans l'ordre, les informations suivantes (voir table E.1):

- endroit où l'on veut que les fichiers, contenant les différents résultats de la simulation, soient imprimés i.e. l'unité de disque avec le répertoire et les sous-répertoires faisant partie de l'arborescence;
- le pas de temps de la phase de préfusion (Δt_p) ainsi que celui de la phase de fusion (Δt_f) (en seconde);
- les intervalles de temps pour imprimer dans les fichiers correspondants, la température à la frontière ainsi qu'à l'intérieur du

domaine de même que la distribution du flux de chaleur (Δt_T), le déplacement de la frontière (Δt_d) et l'erreur relative sur le bilan d'énergie net calculée seulement sur l'intervalle de temps choisi (Δt_E). Ces intervalles de temps doivent être évidemment des multiples des pas de temps utilisés pour la simulation (en seconde);

- température de fusion du matériau solide utilisé (degré C);
- température initiale du matériau solide que l'on suppose uniforme (degré C);
- valeur de la chaleur latente volumique (i.e. ρL) où la valeur de ρ est prise à la température de fusion du matériau (J/m^3);
- valeur de la conductivité thermique (W/(m K));
- valeur de la densité (kg/m^3);
- valeur de la chaleur massique (J/(kg K)).

Table E.1: Contenu du fichier ENTREE.DAT pour la simulation effectuée à la section 3.1.2 avec Δt_p = 30 s et Δt_f = 60 s.

```
DUA0:[R002.REJ.RECH]
30.0  60.0
60.0  300.0 60.0
660.0
26.8
9.53E+8
232.0
2645.0
1054.0
```

Le deuxième fichier (COORD_FRONT.DAT) doit contenir d'abord le nombre de noeuds situés sur la frontière, ensuite la position de ceux-ci, en mètre, par rapport à une origine donnée. Cependant, lorsque la frontière est fermée (voir fig. 3.1), il faut ajouter 1 au nombre de noeuds situés sur cette frontière et mettre comme noeud final la position du premier noeud entré pour que le programme puisse déceler que la frontière est fermée. L'ordre dans lequel la position des noeuds doit être entrée va suivant le sens anti-horaire (voir table E.2). Par contre, lorsqu'il y a un axe de symétrie ou une frontière adiabatique selon x_o, le fichier doit contenir le nombre de noeuds situés sur la frontière non fermée, sans l'axe de symétrie, avec la position de ces derniers. L'origine doit cependant être située à mi-chemin sur l'axe de symétrie et le premier noeud sur l'axe positif des x_o (voir fig. E.1 et table E.3). Si en plus d'avoir un axe de symétrie selon x_o, il y a un axe de symétrie selon y_o le fichier doit contenir le nombre de noeuds situés sur la frontière non fermée sans les axes de symétrie, avec la position de ces noeuds. L'origine est située à l'intersection des deux axes de symétrie (voir fig. E.2 et table E.4).

De plus, avant l'exécution du programme ABLATION, une routine appelée FLUX_DIST et servant à distribuer le flux de chaleur à chacun des noeuds situés sur la frontière doit être écrite, compilée et liée au programme. Pour savoir quelles informations doit contenir la routine FLUX_DIST, voir la figure E.3. La distribution de flux choisie est celle définie à l'équation (3.4).

Table E.2: Contenu du fichier COORD_FRONT.DAT pour la frontière fermée définie à la figure 3.1.

```
      21
 0.5000000   0.0000000
 0.4755283   0.1545085
 0.4045085   0.2938926
 0.2938926   0.4045085
 0.1545085   0.4755283
 0.0000000   0.5000000
-0.1545085   0.4755283
-0.2938926   0.4045085
-0.4045085   0.2938926
-0.4755283   0.1545085
-0.5000000   0.0000000
-0.4755283  -0.1545085
-0.4045085  -0.2938926
-0.2938926  -0.4045085
-0.1545085  -0.4755283
 0.0000000  -0.5000000
 0.1545085  -0.4755283
 0.2938926  -0.4045085
 0.4045085  -0.2938926
 0.4755283  -0.1545085
 0.5000000   0.0000000
```

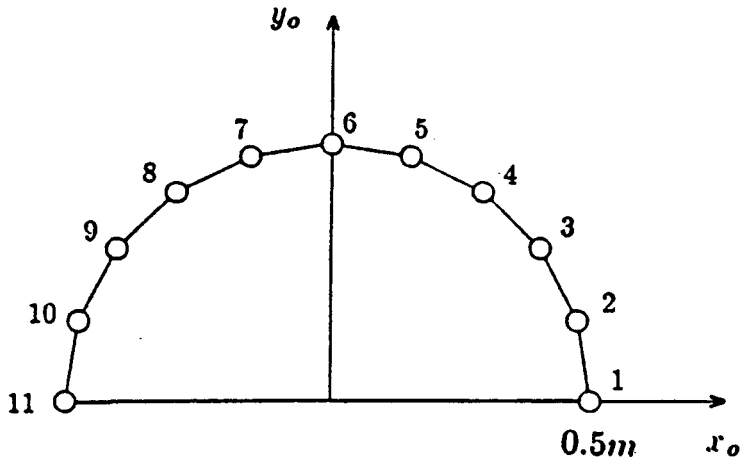


Figure E.1: Discrétisation d'un demi-cercle de rayon 0.5 m en 10 éléments linéaires avec un axe de symétrie selon x_o.

Table E.3: Contenu du fichier COORD_FRONT.DAT pour la frontière non fermée définie à la figure E.1.

```
        11
 0.5000000    0.0000000
 0.4755283    0.1545085
 0.4045085    0.2938926
 0.2938926    0.4045085
 0.1545085    0.4755283
 0.0000000    0.5000000
-0.1545085    0.4755283
-0.2938926    0.4045085
-0.4045085    0.2938926
-0.4755283    0.1545085
-0.5000000    0.0000000
```

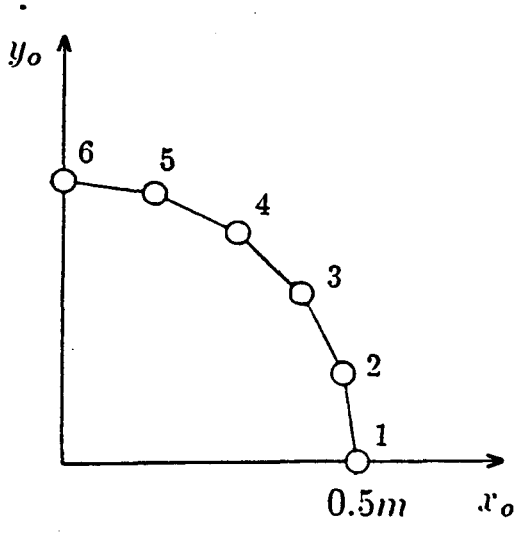


Figure E.2: Discrétisation d'un quart de cercle de rayon 0.5 m en 5 éléments linéaires avec des axes de symétrie selon x_o et y_o.

Table E.4: Contenu du fichier COORD_FRONT.DAT pour la frontière non fermée définie à la figure E.2.

```
         6
0.5000000   0.0000000
0.4755283   0.1545085
0.4045085   0.2938926
0.2938926   0.4045085
0.1545085   0.4755283
0.0000000   0.5000000
```

À la fin de l'exécution du programme ABLATION, les informations suivantes sont disponibles dans les fichiers correspondants qui eux sont situés à l'endroit indiqué dans le fichier ENTREE.DAT:

i) évolution de la température ou du gradient de température, suivant la valeur qui est inconnue aux noeuds de la fontière (VALEUR_INC.DAT);

ii) évolution de la température à l'intérieur du domaine (TEMP_DOMAINE.DAT);

iii) Erreur relative sur le bilan d'énergie net calculée sur l'intervalle de temps indiqué dans le fichier ENTREE.DAT (ERREUR.DAT);

iv) flux de chaleur distribué aux noeuds lors de la simulation (FLUX_DIST.DAT);

v) Configuration triangulaire du domaine initial (DOMAINE.DAT);

vi) position de la frontière en fonction du temps lors de la phase de fusion (POS_FRONT.DAT);

vii) les résultats de la simulation (SORTIE.DAT) comprenant:

1) la durée de la phase de préfusion (t_p),
2) l'erreur relative commise sur le bilan d'énergie net de la phase de préfusion (%),
3) l'erreur relative commise sur le bilan d'énergie net calculée lors de la simulation(%),
4) le temps réel de simulation,
5) le temps de calcul.

```
      SUBROUTINE FLUX_DIST(XF,YF,FLUX_N,TEMP_N,TEMPS)
C
C-------------------------------------------------------------
C
C     ROUTINE SERVANT A DISTRIBUER LE FLUX DE CHALEUR
C     AUX NOEUDS SITUES SUR LA FRONTIERE.
C
C-------------------------------------------------------------
C
      PARAMETER(NB_F = 32)
C
C     NB_F :NOMBRE MAXIMAL DE NOEUDS QUE LE PROGRAMME
C     PEUT CONSIDERER
C
      REAL XF(0:NB_F+1),YF(0:NB_F+1),FLUX_N(NB_F),
     + TEMPS,FLUX_MOY,NORME,TEMP_N(NB_F)
      INTEGER NB_N,NB_N_S,I
      COMMON/FRONT/NB_N,NB_N_S
      DATA EPSILON/1.0E-6/
C
C-------------------------------------------------------------
C
C     XF,YF  :VECTEURS CONTENANT LES COORDONNEES DES
C             NOEUDS SITUES SUR LA FRONTIERE FERMEE.
C
C     NB_N   :NOMBRE DE NOEUDS SITUES SUR LA FRONTIE-
C             RE FERMER.
C
C     NB_N_S :NOMBRE DE NOEUDS DE LA FRONTIERE OUVERTE
C             OU FERMEE DEPENDAMMENT S'IL Y A DES AXES
C             DE SYMETRIE OU NON.
C
C     FLUX_N :VECTEUR CONTENANT LA VALEUR DU FLUX
C             ENTRANT A CHAQUE NOEUD SITUE SUR LA
C             FRONTIERE OUVERTE OU FERMEE.
C
C     TEMP_N :VECTEUR CONTENANT LA TEMPERATURE AUX
C             NOEUDS SITUES SUR LA FRONTIERE OUVERTE
C             OU FERMEE AU TEMPS "TEMPS-DELTA_T"
C
C     TEMPS  :TEMPS ACTUEL DE LA SIMULATION.
C
C-------------------------------------------------------------
C
      DO I=1,NB_N_S
        X=YF(I+1)-YF(I-1)
        Y=XF(I-1)-XF(I+1)
        IF (ABS(X).LT.EPSILON) X=0.0
        IF (ABS(Y).LT.EPSILON) Y=0.0
        NORME=SQRT(X*X+Y*Y)
C
C       (X,Y)/NORME :VECTEUR UNITAIRE SORTANT DEFINI
C                    AU NOEUD I DE LA FRONTIERE.
C
C       LE FLUX ENTRE PERPENDICULAIREMENT A LA SURFACE.
C
        FLUX_N(I)=-1.0E5*Y/NORME
      END DO
C
C--------
C
      RETURN
      END
```

Figure E.3: Listage de la routine FLUX_DIST où le flux distribué est celui défini à l'équation (3.4).

E.2 Organigrammes du programme ABLATION

Le programme ABLATION est divisé en deux parties (voir fig. E.4):

- Initialisation des données nécessaires à la simulation par l'entremise du sous-programme INITIALISER.

- Résolution du problème d'ablation par la méthode des éléments finis de frontière, par l'entremise du sous-programme SIMULATION.

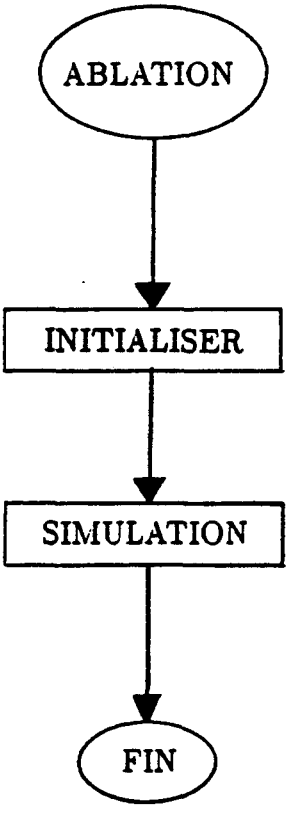


Figure E.4: Organigramme du programme ABLATION.

E.2.1 Description du sous-programme INITIALISER

Le sous-programme INITIALISER (voir Fig. E.5) est exécuté au tout début pour fournir les renseignements nécessaires à la simulation du problème d'ablation. Il commence par lire les fichiers ENTREE.DAT et COORD_FRONT.DAT et vérifie par la suite si la frontière est fermée. Si elle ne l'est pas (i.e. il y a un ou deux axes de symétrie), alors la routine SYMETRIE_FRONT est appelée pour calculer, par symétrie, la position des noeuds manquant pour obtenir une frontière fermée dans le but d'utiliser le processus de condensation directe. La routine MAILLAGE_AUTO est utilisée pour calculer les coordonnées des sommets des triangles discrétisant le domaine et cela à partir des coordonnées des noeuds situés sur la frontière fermée. La configuration triangulaire utilisée est celle de la figure 3.2c.

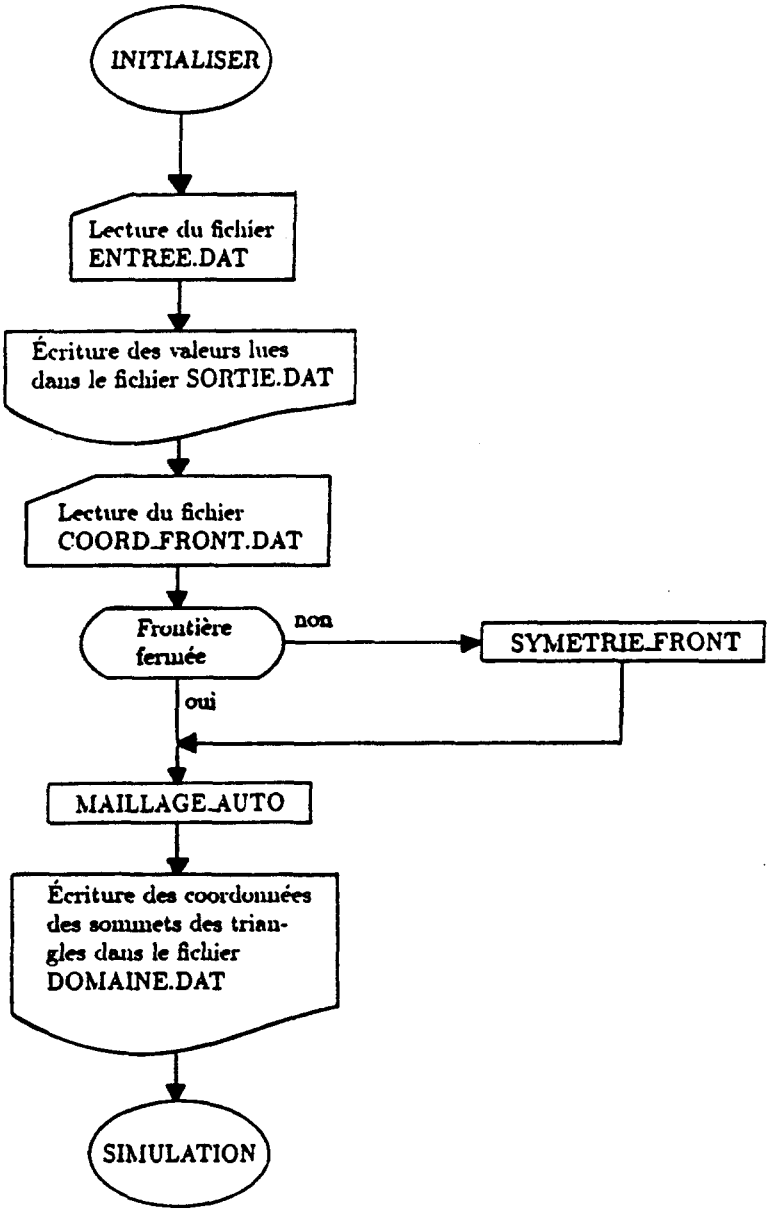


Figure E.5: Organigramme du sous-programme INITIALISER.

E.2.2 Description du sous-programme SIMULATION

Le sous-programme SIMULATION (fig. E.6) sert à simuler le problème d'ablation en commençant par la phase de préfusion, en utilisant le pas de temps choisi pour cette phase (Δt_p), et en finissant par la phase de fusion, en utilisant également le pas de temps correspondant (Δt_f). Cependant, lors de la phase de fusion, lorsque la température du domaine devient saturée, le pas de temps choisi est 5 secondes pour éviter un changement trop brusque de la frontière lorsque le solide devient petit.

Voici la description des routines utilisées dans le sous-programme SIMULATION:

CENTRE_GRA_TRI: calcule le centre de gravité de chacun des triangles pour évaluer la distribution de température à l'intérieur du domaine.

MATRICES_H_G: calcule les éléments de matrices H et G définies en (2.14) et regroupe les valeurs (températures ou gradients de température à la frontière) connues à droite et inconnues à gauche du système d'équations linéaires (2.14).

MATRICES_B: calcule les éléments de la matrice B définie en (2.14).

FLUX_DIST: distribue le flux de chaleur à chaque noeud situé sur la frontière.

VERI_TEMP_FUSION: vérifie s'il y a un noeud à la frontière qui a atteint la température de fusion. Si oui, la valeur inconnue à ce noeud est maintenant le gradient de température et la valeur connue est la température de fusion.

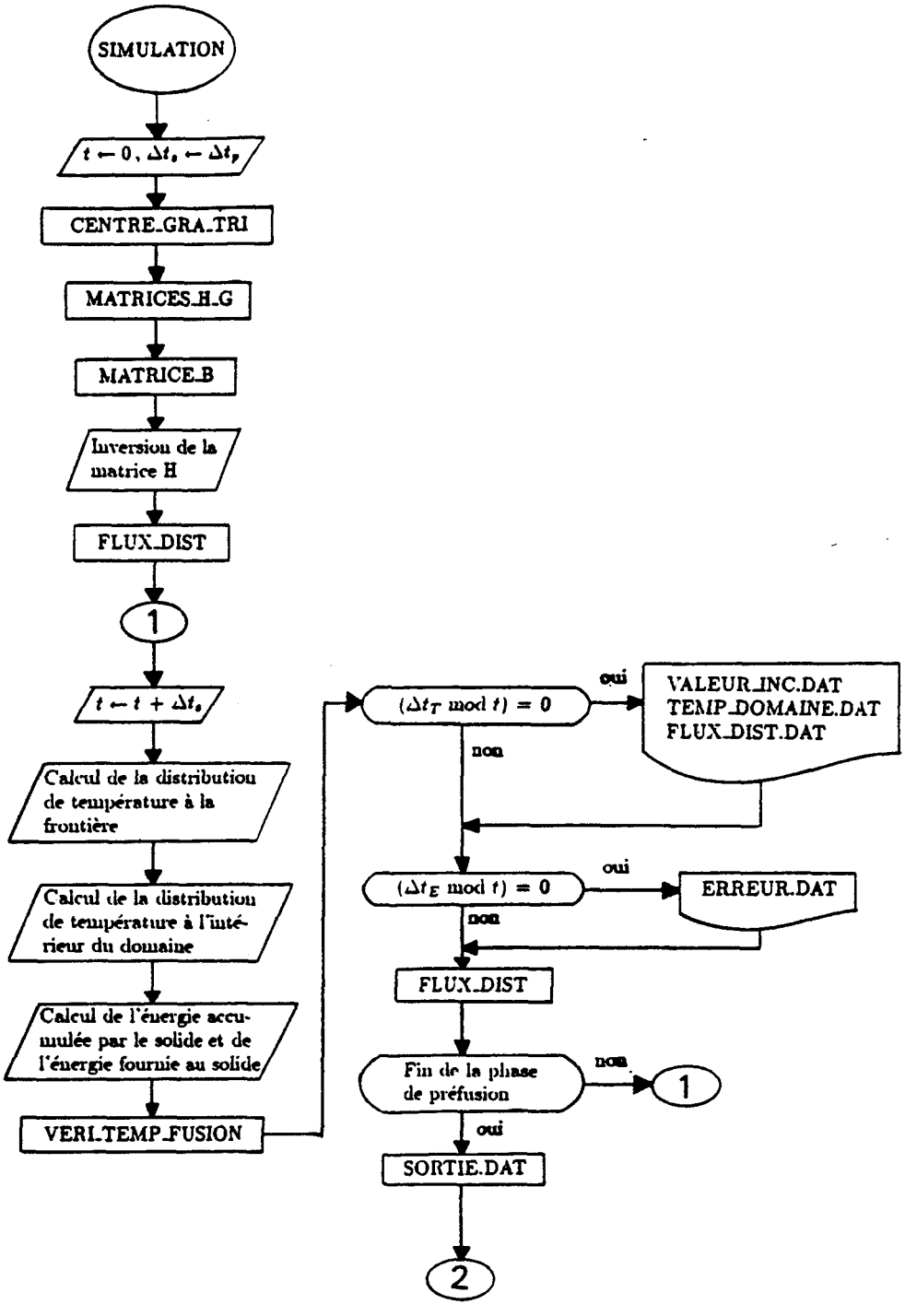
DEP_FRONT: déplace les noeuds de la frontière qui ont atteint la température de fusion en utilisant la condition de Stéfan écrite en termes de quantités nodales (voir équation (2.15)).

DIMIN_NB_COUCHE: diminue le nombre de couches triangulaires discrétisant le domaine et cela en suivant les spécifications de la table 3.2.

AIRE_TRIANGLE: calcule l'aire des triangles discrétisant le domaine pour déterminer le pourcentage en volume du solide restant.

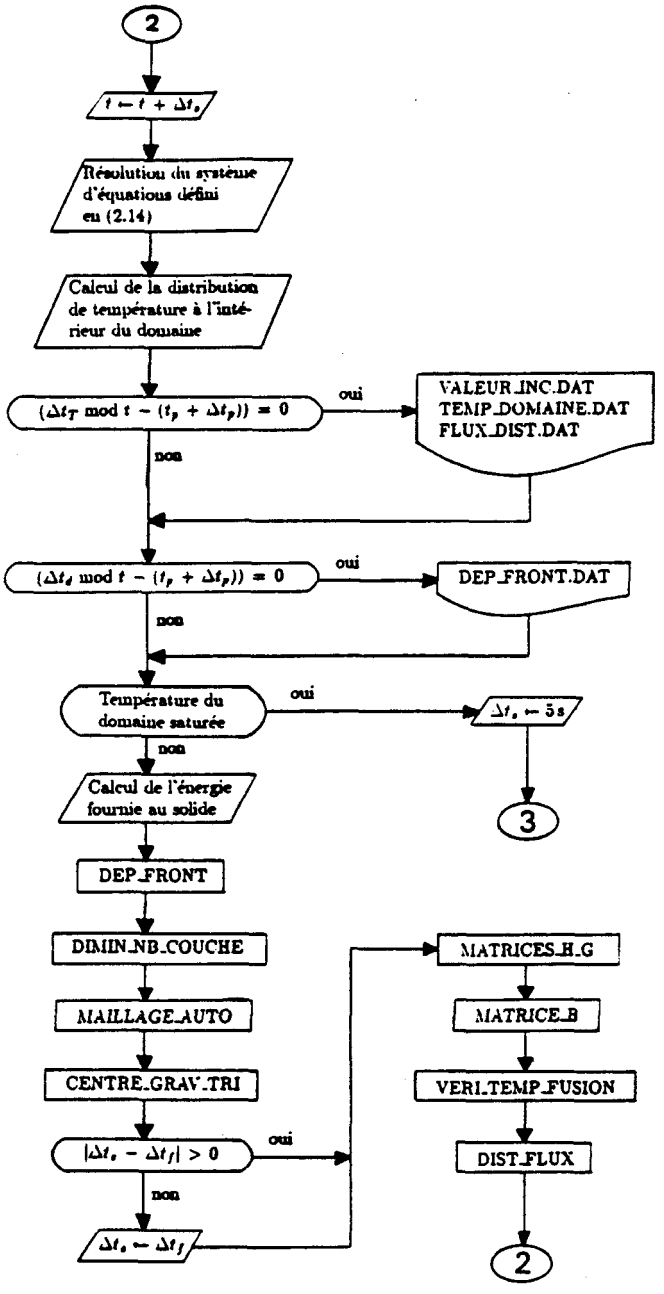
E.3 Remarque

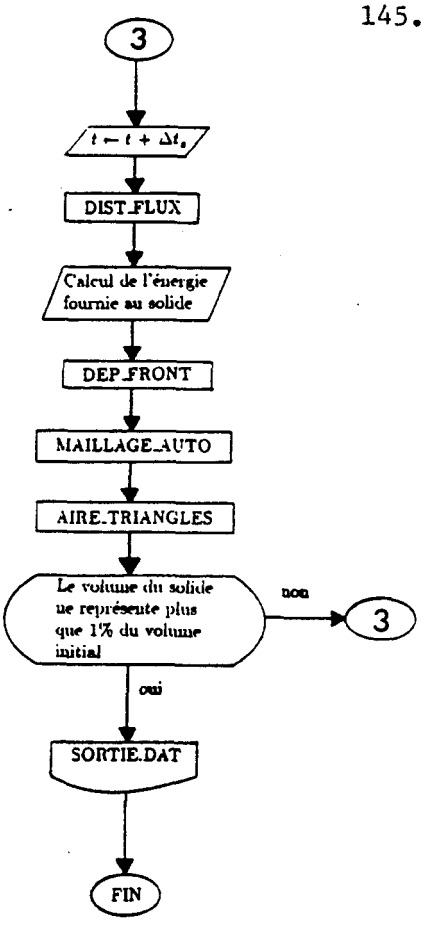
Le choix du solide homogène et isotrope dont l'échange thermique est bidimensionnel n'est pas limitatif. Cependant, pour considérer un solide dont l'aire de la section transversale est plus grande que 1 m^2, il faut choisir un nombre plus élevé de couches subdivisant le domaine. Pour cela, il suffit de changer dans la routine INITIALISER, la valeur de la variable NB_COUCHE de même que les valeurs du vecteur LAMBDA. De plus, il faut modifier la routine DIMIN_NB_COUCHE pour l'adapter à la nouvelle distribution de couches choisies. Il faut toutefois s'assurer que les matrices utilisées dans le programme ABLATION soient de dimensions adéquates.



a) Phase de préfusion

Figure E.6: Organigramme du sous-programme SIMULATION (suite page suivante).





i) non saturée ii) saturée

b) Phase de fusion avec température du domaine i) non saturée et ii) saturée.

Figure E.6: (Suite)

ANNEXE

LISTAGE DU PROGRAMME

```
C
C##################################################
C#                                                #
C#               PROGRAMME ABLATION               #
C#              ------------------                #
C#                                                #
C#      MODELE MATHEMATIQUE SERVANT A SIMULER     #
C#      UN PROBLEME D'ABLATION A DEUX DIMEN-      #
C#      SIONS POUR UN DOMAINE HOMOGENE AVEC       #
C#      UNE CONDITION DE NEUMANN A SA FRON-       #
C#      TIERE.                                    #
C#                                                #
C#                           R.O. JANVIER,1987    #
C#                                                #
C##################################################
C
        PARAMETER(NB_F = 32,NB_E = 532,NB_D = 365)
        REAL XF(0:NB_F+1),YF(0:NB_F+1),TEMP_F(NB_F),
     +  T(NB_E),XI(NB_D),YI(NB_D)
        INTEGER E(NB_E,3),TY(NB_F)
C
C       XF,YF  :COORDONNEES DES NOEUDS A LA FRONTIERE.
C       TEMP_F :TEMPERATURE IMPOSEE OU CALCULEE, SELON LE CAS, AUX
C               NOEUDS SITUES SUR LA FRONTIERE DU DOMAINE.
C       TY(I)  :SI TY(I)='0' ON A UNE CONDITION DE TYPE DIRICHLET AU
C               NOEUD I,
C               SI TY(I)='1' ON A UNE CONDITION DE TYPE NEUMANN AU
C               NOEUD I.
C       E      :NUMEROTATION DES NOEUDS POUR LES ELEMENTS TRIANGULAIRES
C               DISCRETISANT LE DOMAINE.
C       T      :TEMPERATURE CORRESPONDANT AU CENTRE DE GRAVITE
C               DE CHACUN DES ELEMENTS TRIANGULAIRES.
C       XI,YI  :COORDONNEES DES SOMMETS DES TRIANGLES DISCRETISANT
C               LE DOMAINE.
C
        CALL INITIALISER(XF,YF,TY,TEMP_F,E,T,XI,YI)
        CALL CPUTIME
        CALL SIMULATION(XF,YF,TY,TEMP_F,E,T,XI,YI)
        CALL CPUTIME
```

```
C
C--------
C
      END


      SUBROUTINE INITIALISER(XF,YF,TY,TEMP_F,E,T,XI,YI)
C
C-----------------------------------------------------------------
C
C     ROUTINE SEVANT A INITIALISER LES VALEURS NECESSAIRES
C     AU DEROULEMENT DU PROGRAMME.
C
C-----------------------------------------------------------------
C
      PARAMETER(NB_F = 32,NB_E = 532,NB_D = 365,NB_DIV = 10)
      REAL XF(0:NB_F+1),YF(0:NB_F+1),TEMP_F(NB_F),LAMBDA(NB_DIV),
     + K,RHO_L,PI,IMP,CGT(NB_E,2),T(NB_E),XI(NB_D),YI(NB_D),
     + DT_DEP,DT_C,DT_A,RHO,C,ALPHA,FUSION,KELVIN,EPSILON
      INTEGER ORD1,E(NB_E,3),ORD2,EL,NB_COUCHE,TY(NB_F),N_ELE,
     +        NOF_M,NOF,DT_IMP_T,DT_IMP_P,DT_ERR
      LOGICAL FERMER,QUART
      CHARACTER *40 NOM,ENDROIT
      CHARACTER *80 NOMFICH
      COMMON/CONTOUR/FERMER/DEGRE_KEL/KELVIN
      COMMON/TEMPS/DT_C,DT_A,DT_DEP/PROP/ALPHA,K,RHO_L
      COMMON/INTEG1/ORD1/FRONT/NOF,NOF_M/MESH/LAMBDA,NB_COUCHE
      COMMON/DOMM/EL,NOI,N_ELE/CONST/DENOM,PI
      COMMON/INTEG1_I/ORD2/TEMP_REF/FUSION/FORM/QUART
      COMMON/ENERGIE/RHO,CP/FICHIER/ENDROIT
      COMMON/IMPRESSION/DT_IMP_T,DT_IMP_P,DT_ERR
      DATA EPSILON/1.0E-6/
C
C     ORD1       :NOMBRE DE POINTS UTILISES POUR L'INTEGRATION NU-
C                 MERIQUE A UNE DIMENSION PAR LA METHODE DE GAUSS-
C                 LEGENDRE.
C     ORD2       :NOMBRE DE POINTS UTILISES POUR L'INTEGRATION NUME-
C                 RIQUE A UNE DIMENSION PAR LA METHODE DE GAUSS-
C                 LAGUERRE.
C     NB_COUCHE  :NOMBRE DE COUCHES POUR DISCRETISER LE DOMAINE.
C     LAMBDA     :DETERMINE LA LARGEUR DES COUCHES SERVANT A
C                 DISCRETISER LE DOMAINE.
C     NOF        :NOMBRE DE NOEUDS ET D'ELEMENTS  SUR LA FRONTIERE
C                 FERMEE.
C     NOF_M      :NOMBRE DE NOEUDS SUR LA FRONTIERE N'INCLUANT PAS
C                 LA PARTIE ADIABETIQUE S'IL Y A LIEU.
C     NOI        :NOMBRE DE NOEUDS SITUES DANS LE DOMAINE.
C     EL         :NOMBRE D'ELEMENTS TRIANGULAIRES SITUES DANS LE
C                 DOMAINE.
C     N_ELE      :NOMBRE D'ELEMENTS TRIANGULAIRES DISCRETISANT LE
C                 DOMAINE OUVERT OU FERMER SELON LE CAS.
```

```
C       ALPHA      :DIFFUSIVITE THERMIQUE.(m**2/s)
C       K          :CONDUCTIVITE THERMIQUE.(W/(m K))
C       RHO        :DENSITE (kg/m**3)
C       CP         :CHALEUR MASSIQUE (J/(kg K))
C       RHO_L      :DIFFERENCE D'ENTHAPIE DE LA FUSION OU 'L' EST
C                   LA CHALEUR LATENTE.(J/m**3)
C       DT_IMP_T   :INCREMENT POUR IMPRIMER LES DIFFERENTES TEMPERATURES.
C       DT_IMP_P   :INCREMENT POUR IMPRIMER LA POSITION DE LA FRONTIERE.
C       DT_ERR     :INCREMENT POUR IMPRIMER L'ERREUR RELATIVE SUR LE
C                   BILAN D'ENERGIE NET CALCULE, LORS DE LA PHASE DE PRE-
C                   FUSION, SEULEMENT POUR LA PERIODE DE CET INCREMENT.
C       FERMER     :DIT SI LA SURFACE EST FERMEE OU NON.ELLE EST FERMEE
C                   SI IL N'Y A AUCUNE SURFACE ADIABETIQUE.
C       QUART      :DIT S'IL IL Y A UN PLAN DE SYMETRIE PAR RAPPORT A Y.
C       FUSION     :TEMPERATURE DE FUSION DU MATERIAU CONSIDERE EN CELSIUS.
C       TEMP       :TEMPERATURE INITIALE DU DOMAINE EN CELSIUS.
C       KELVIN     :DONNE LA CORRESPONDANCE DE 0 DEGRE CELCIUS EN KELVIN.
C       ENDROIT    :NOM DE L'ENDROIT OU L'ON VEUT IMPRIMER LE OU LES
C                   FICHIERS DE SORTIE.
C       CGT        :COORDONNEES DU CENTRE DE GRAVITE DES TRIANGLES
C                   DISCRTISANT LE DOMAINE.
C       T          :TEMPERATURES AU CENTRE DE GRAVITE DES TRIANGLES
C                   DISCRETISANT LE DOMAINE.
C
C
        KELVIN=273.2
        ORD1=12
        ORD2=6
        DO I=1,NB_F
          TY(I)=0
        END DO
        PI=3.1415926535897 93E0
        OPEN(1,FILE='ENTREE.DAT',STATUS='UNKNOWN')
        READ(1,1) ENDROIT
1       FORMAT(A)
        READ(1,*) DT_C,DT_A
        READ(1,*) DT_IMP_T,DT_IMP_P,DT_ERR
        READ(1,*) FUSION
        READ(1,*) TEMP
        READ(1,*) RHO_L
        READ(1,*) K
        READ(1,*) RHO
        READ(1,*) CP
        CLOSE(1)
        ALPHA=K/(RHO*CP)
        DT_DEP=5.0
        NB_COUCHE=6
        LAMBDA(1)=0.0
        LAMBDA(2)=0.1
        LAMBDA(3)=0.2
        LAMBDA(4)=0.4
        LAMBDA(5)=0.6
```

```
      LAMBDA(6)=0.8
      PRINT *
      PRINT *
      PRINT *,'-------------------------------------------------'
      PRINT *
      PRINT *,'         SIMULATION D''UN PROBLEME D''ABLATION'
      PRINT *
      PRINT *,'-------------------------------------------------'
      PRINT *
      PRINT *
      NOM='SORTIE.DAT'
      NOMFICH=ENDROIT//NOM
      OPEN(1,FILE=NOMFICH,STATUS='UNKNOWN')
      WRITE(1,10) JNINT(DT_C),JNINT(DT_A),K,RHO,CP,ALPHA,RHO_L,
     +  FUSION,TEMP
10    FORMAT(X,3(/),15(' '),'<<<<< DONNEES DU PROBLEME >>>>>',2(/),
     +  2X,60('-'),2(/),
     +  2X,'PAS DE TEMPS POUR LA PREFUSION (s)           : ',I4,2(/),
     +  2X,'PAS DE TEMPS POUR LA FUSION (s)              : ',I4,2(/),
     +  2X,'VALEUR DE LA CONDUCTION THERMIQUE (W/(m K))  : ',F7.1,2(/),
     +  2X,'VALEUR DE LA DENSITE (kg/m**3)               : ',F8.1,2(/),
     +  2X,'VALEUR DE LA CHALEUR MASSIQUE (J/(kg K)      : ',F8.1,2(/),
     +  2X,'VALEUR DE LA DIFFUSIVITE (m**2/SEC.)         : ',1PE9.1,
     +     2(/),
     +  2X,'CHALEUR LATENTE DE FUSION i.e. RHO_L (J/m**3) : ',1PE10.2,
     +     2(/),
     +  2X,'TEMPERATURE DE FUSION (C)                    : ',0PF7.1,
     +     2(/),
     +  2X,'TEMPERATURE INITIALE DU DOMAINE (C)          :',F7.1,2(/),
     +        2X,60('-'),3(/))
      CLOSE(1)
      TEMP=TEMP+KELVIN
      FUSION=FUSION+KELVIN
      OPEN(2,FILE='COORD_FRONT.DAT',STATUS='UNKNOWN')
      READ(2,*) NOF_M
      DO I=1,NOF_M
        READ(2,*) XF(I),YF(I)
        TY(I)=1
      END DO
      CLOSE(2)
      QUART=.FALSE.
      IF ((ABS(XF(1)-XF(NOF_M)).LT.EPSILON).AND.
     +    (ABS(YF(1)-YF(NOF_M)).LT.EPSILON)) THEN
        XF(NOF_M)=0.0
        YF(NOF_M)=0.0
        TY(NOF_M)=0
        TEMP_F(NOF_M)=0.0
        FERMER=.TRUE.
        NOF_M=NOF_M-1
        NOF=NOF_M
        N1=NOF
      ELSE
```

```
          FERMER=.FALSE.
          IF (ABS(YF(NOF_M)).GT.EPSILON) QUART=.TRUE.
          N1=NOF_M-1
        END IF
        IF (.NOT.FERMER) CALL SYMETRIE_FRONT(XF,YF)
        CALL MAILLAGE_AUTO(XF,YF,XI,YI,E)
        NB_TRI=NOF*3
        DO I=1,NOF_M
          TEMP_F(I)=TEMP-FUSION
        END DO
        DO I=1,N_ELE
           T(I)=TEMP-FUSION
        END DO
        NOM='DOMAINE.DAT'
        NOMFICH=ENDROIT//NOM
        OPEN(3,FILE=NOMFICH,STATUS='UNKNOWN')
        N2=3*N1
        WRITE(3,*) N_ELE
        DO L=1,NB_COUCHE-1
          M=(L-1)*NB_TRI
          DO I=1+M,N2+M
            WRITE(3,30) XI(E(I,1)),YI(E(I,1)),XI(E(I,2)),YI(E(I,2)),
     +                  XI(E(I,3)),YI(E(I,3))
          END DO
        END DO
        M=(NB_COUCHE-1)*NB_TRI
        DO I=1+M,N1+M
          WRITE(3,30) XI(E(I,1)),YI(E(I,1)),XI(E(I,2)),YI(E(I,2)),
     +                XI(E(I,3)),YI(E(I,3))
        END DO
30      FORMAT(2X,6(F6.3,2X))
        CLOSE(3)
C
C-------------
C
        RETURN
        END


        SUBROUTINE SIMULATION(XF,YF,TY,TEMP_F,E,T,XI,YI)
C
C-----------------------------------------------
C
C       ROUTINE SOLUTIONNANT LE PROBLEME
C       D'ABLATION A DEUX DIMENSIONS PAR LA
C       METHODE DES ELEMENTS FINIS DE FRONTIERE
C       EN CONSIDERANT UNE VARIATION LINEAIRE
C       DE U ET DE DU/DN SUR DES ELEMENTS LINE-
C       AIRES CONSTITUANT LA FRONTIERE.LE DOMAI-
C       EST DISCRETISE EN ELEMENTS TRIANGULAIRES
C       LINEAIRES AVEC UNE TEMPERATURE CONSTANTE
```

```
C       SUR CHAQUE ELEMENT.
C
C-----------------------------------------------
C
        PARAMETER(NB_F = 32,NB_E = 532,NB_D = 365,NB_DIV = 10)
        REAL K,XI(NB_D),YI(NB_D),CGI(NB_E,2),XF(0:NB_F+1),
     +  YF(0:NB_F+1),H(NB_F,NB_F+1),G(NB_F,2*NB_F),
     +  XI_AV(NB_D),YI_AV(NB_D),AIRE(NB_E),TEMP_F(NB_F),
     +  HD(NB_E,NB_F+1),GD(NB_E,2*NB_F),B(NB_F,NB_E),
     +  BD(NB_E,NB_E),PRODG(NB_F),PRODGD(NB_E),PRODHD(NB_E),
     +  PRODBD(NB_E),PRODB(NB_F),VAL(2*NB_F),T(NB_E),VAL_INC(NB_F),
     +  FUSION,VECT_C(NB_F),LAMBDA(NB_DIV),DS(NB_F),FLUX(NB_F,2),
     +  DT_DEP,DT,DT_C,DT_A,H_TEMP(NB_F,NB_F+1),AIRE_TOT,NORME,
     +  ENER_D,ENER_D_TOT,ENER_IN,ENER_IN_TOT,PAS_AV,T_AV(NB_E),
     +  RHO,CP,T_INIT(NB_E),AIRE_TOT_INIT,LXY(0:NB_F+1),AIRE_INIT(NB_E),
     +  TEMPS_COND,EPSILON,KELVIN,UNITE,TEMPS_FIN,FLUX_MOY,LONG_MAX,
     +  FLUX_NOEUD(NB_F)
        INTEGER E_AV(NB_E,3),E(NB_E,3),EL,EL_AV,EL_INIT,NB_COUCHE,
     +          INDK(3),INDL(3),TY(NB_F),DT_ERR,DT_IMP_T,N_ELE,
     +          DT_IMP_P,NOF_M,NOF,NOI,EXPO,NB_COUCHE_AV
        LOGICAL ABLA(NB_F+1),FIN_COND,STEADY_STATE,FERMER,QUART,
     +          TERMINER
        CHARACTER *1 C
        CHARACTER *40 NOM,ENDROIT
        CHARACTER *80 NOMFICH
        COMMON/TEMPS/DT_C,DT_A,DT_DEP/FRONT/NOF,NOF_M/DOMM/EL,NOI,N_ELE
        COMMON/MESH/LAMBDA,NB_COUCHE/COORD_TRI/INDK,INDL
        COMMON/CONST/DENOM,PI/TEMP_REF/FUSION/DEGRE_KEL/KELVIN
        COMMON/CONTOUR/FERMER/LONG/LXY/ENERGIE/RHO,CP
        COMMON/IMPRESSION/DT_IMP_T,DT_IMP_P,DT_ERR/PROP/ALPHA,K,RHO_L
        COMMON/FICHIER/ENDROIT/FORM/QUART/MESH_AV/NB_COUCHE_AV
        DATA EPSILON/1.0E-6/
        INTRINSIC JNINT,JMOD
C
C       VAL_INC     :VALEURS DES TEMPERATURES OU DES GRADIENTS TROUVES
C                    APRES LA RESOLUTION DU SYSTEME D'EQUATIONS.
C       EL_AV       :NOMBRE D'ELEMENTS DANS LE DOMAINE AU PAS TEMPS DE
C                    PRECEDANT.
C       ABLA        :INDIQUE QUEL NOEUD EST EN PHASE DE FUSION.
C       STEADY_STATE:INDIQUE SI LE DOMAINE A ATTEINT LA TEMPERATURE
C                    DE SATURATION.
C       FIN_COND    :INDIQUE SI LA PHASE DE PREFUSION EST TERMINER.
C       TERMINER    :INDIQUE SI LA SIMULATION EST TERMINER.
C       LXY         :LONGUEUR DES ELEMENTS CONSTITUANT LA FRONTIERE.
C
C
C
C       INITIALISATION DES POINTS D'INTEGRATION ET DES POIDS POUR
C       L'INTEGRATION NUMERIQUE,DE MEME QUE DES FACTORIELS UTILE
C       L'EVALUATION DES SERIES.
C
        CALL VAL_INT1
```

```
      CALL VAL_INT1_LAG
      CALL FACTORIEL
C
C
C     INDK ET INDL SONT UTILISER POUR POUVOIR FAIRE L'INTEGRATION
C     EN COORDONNEES POLAIRES SUR LE DOMAINE.
C
      INDK(1)=2
      INDK(2)=3
      INDK(3)=1
      INDL(1)=3
      INDL(2)=1
      INDL(3)=2
C
C     INITIALISATION ET DETERMINATION DU NOMBRE DE COLONNES QUE
C     POSSEDE LA MATRICE G.
C
      N_ELE_INIT=N_ELE
      NB_COUCHE_AV=NB_COUCHE
      NB_TRI=NOF*3
      DO I=1,N_ELE
        T_INIT(I)=T(I)
      END DO
      CALL LONGUEUR(XF,YF)
      CALL FLUX_DIST(XF,YF,FLUX_NOEUD,TEMP_F,DT_C)
      DO I=1,NOF_M
        FLUX(I,1)=FLUX_NOEUD(I)
        FLUX(I,2)=FLUX_NOEUD(I)
      END DO
      IF (FERMER) THEN
        NE_G=2*NOF
        DO I=1,NOF
          IF ((TY(I).EQ.4).OR.(TY(I).EQ.5)) NE_G=NE_G-1
        END DO
        N1=NOF
      ELSE
        NE_G=2*(NOF_M-1)
        DO I=2,NOF_M-1
          IF ((TY(I).EQ.4).OR.(TY(I).EQ.5)) NE_G=NE_G-1
        END DO
        N1=NOF_M-1
      END IF
      N2=3*N1
      DO I=1,NOF_M
        ABLA(I)=.FALSE.
        IF (TY(I).EQ.1) THEN
          VAL_INC(I)=TEMP_F(I)
        ELSE
          VAL_INC(I)=0.0
        END IF
      END DO
      IF (.NOT.FERMER) CALL SYMETRIE_ABLA(ABLA)
```

```
      ABLA(NOF+1)=.FALSE.
C
C     DETERMINATION DES VALEURS CONNUES CONSTITUANT LE
C     VECTEUR COLONNE QUI MULTIPLI LA MATRICE G.
C
      CALL ASSIG_VAL_CONNUE(TY,FLUX,TEMP_F,VAL,NE_G)
C
C
C     DEBUT DE LA CONDUCTION
C
      FIN_COND=.FALSE.
      CALL CENTRE_GRAV_TRI(E,XI,YI,CGT)
      NOM='VALEUR_INC.DAT'
      NOMFICH=ENDROIT//NOM
      OPEN(1,FILE=NOMFICH,STATUS='UNKNOWN')
      NOM='TEMP_DOMAINE.DAT'
      NOMFICH=ENDROIT//NOM
      OPEN(2,FILE=NOMFICH,STATUS='UNKNOWN')
      NOM='ERREUR.DAT'
      NOMFICH=ENDROIT//NOM
      OPEN(3,FILE=NOMFICH,STATUS='UNKNOWN')
      NOM='FLUX_DIST.DAT'
      NOMFICH=ENDROIT//NOM
      OPEN(8,FILE=NOMFICH,STATUS='UNKNOWN')
      NOM='SORTIE.DAT'
      NOMFICH=ENDROIT//NOM
      OPEN(15,FILE=NOMFICH,STATUS='UNKNOWN',ACCESS='APPEND')
      PAS=0.0
      PAS_AV=0.0
      WRITE(1,12)
12    FORMAT(2X,'T = TEMPERATURE (C), G = GRADIENT (C/m)',2(/),
     +       2X,'NOEUD    X       Y    TYPE',/)
      WRITE(2,1)
1     FORMAT(2X,'NOEUD    X       Y    TEMPERATURE (C)',/)
      WRITE(3,2)
2     FORMAT(2X,'TEMPS    ERREUR EN %',/)
      WRITE(8,8)
8     FORMAT(2X,'NOEUD    X       Y     FLUX (W/m**2)',/)
      ENER_D_TOT=0.0
      ENER_IN_TOT=0.0
      WRITE(1,*) PAS
      WRITE(8,*) PAS
      DO I=1,NOF_M
        IF (TY(I).EQ.1) THEN
          C='T'
          VAL_IMP=TEMP_F(I)+FUSION-KELVIN
        ELSE
          C='G'
          VAL_IMP=FLUX(I,1)/K
        END IF
        WRITE(1,35) I,XF(I),YF(I),C,VAL_IMP
        WRITE(8,10) I,XF(I),YF(I),0.0
```

```
          END DO
          WRITE(2,*) PAS
          I=0
          DO L=1,NB_COUCHE-1
            M=(L-1)*NB_TRI
            DO J=M+1,M+N2
              I=I+1
              WRITE(2,5) I,CGT(J,1),CGT(J,2),T(I)+FUSION-KELVIN
            END DO
          END DO
          M=(NB_COUCHE-1)*NB_TRI
          DO J=M+1,M+N1
            I=I+1
            WRITE(2,5) I,CGT(J,1),CGT(J,2),T(I)+FUSION-KELVIN
          END DO
C
C         EVALUATION DES MATRICES POUR SOLUTIONNER LE SYSTEME D'EQUATIONS
C
          DENOM=4.0*ALPHA*DT_C
C
          CALL MATRICES_H_G(XF,YF,H,G,TY,ABLA,NE_G)
          CALL MATRICE_B(XI,YI,E,XF,YF,B,EL)
C
C         EVALUATION DES MATRICES POUR L'EVALUATION DES TEMPERATURES
C         INTERNES.
C
          CALL MATRICES_HD_GD(XF,YF,CGT,HD,GD,TY,ABLA,NE_G)
          CALL MATRICE_BD(XI,YI,E,CGT,BD,EL)
C
C         ROUTINE SERVANT A VERIFIER S'IL N'Y PAS DE NOEUD QUI
C         A ATTEINT LA TEMPERATURE DE FUSION.
C
          CALL VERI_TEMP_FUSION(TY,ABLA,FLUX,VAL,VAL_INC,G,H,
     +                          GD,HD,NE_G)
          I=1
C
C         VERIFIE SI LA PHASE DE PREFUSION N'EST PAS TERMINER
C
          DO WHILE ((I.LE.NOF_M).AND.(.NOT.FIN_COND))
            IF (ABLA(I)) FIN_COND=.TRUE.
            I=I+1
          END DO
          IF (.NOT.FIN_COND) THEN
            DO I=1,NOF_M
              DO J=1,NOF_M
                H_TEMP(I,J)=H(I,J)
              END DO
            END DO
            DETER=SIMUL(NB_F,NOF_M,H_TEMP,VAL_INC,-1)
            IF (DETER.EQ.0.0) THEN
              WRITE(15,50)
50            FORMAT(2X,'LE DET(H)=0 POUR LA PHASE DE LA CONDUCTION.',/)
```

```
            CALL CPUTIME
            STOP
          END IF
          CALL PRODUIT_MAT(G,VAL,PRODG,NOF_M,NE_G,NB_F,2*NB_F)
          CALL PRODUIT_MAT(GD,VAL,PRODGD,N_ELE,NE_G,NB_E,2*NB_F)
        END IF
        CALL AIRE_TRIANGLES(E,XI,YI,AIRE)
        AIRE_TOT_INIT=0.0
        DO I=1,N_ELE
          AIRE_TOT_INIT=AIRE_TOT_INIT+AIRE(I)
          AIRE_INIT(I)=AIRE(I)
        END DO
        ENER_D_IMP=0.0
        ENER_IN_IMP=0.0
        DT=DT_C
        DO WHILE (.NOT.FIN_COND)
          PAS_AV=PAS
          PAS=PAS+DT
          ENER_D=0.0
          ENER_IN=0.0
          DO I=1,N_ELE
            T_AV(I)=T(I)
          END DO
          CALL PRODUIT_MAT(B,T,PRODB,NOF_M,N_ELE,NB_F,NB_E)
          CALL PRODUIT_MAT(BD,T,PRODBD,N_ELE,N_ELE,NB_E,NB_E)
C
C          CALCUL DE LA MATRICE AUGMENTEE.
C
          DO I=1,NOF_M
            VECT_C(I)=PRODG(I) + PRODB(I)
          END DO
          CALL PRODUIT_MAT(H_TEMP,VECT_C,VAL_INC,NOF_M,NOF_M,
     +                     NB_F,NB_F+1)
          CALL PRODUIT_MAT(HD,VAL_INC,PRODHD,N_ELE,NOF_M,NB_E,NB_F+1)
C
C         EVALUATION DES TEMPERATURES INTERNES.
C
          DO I=1,N_ELE
            T(I) = PRODGD(I) + PRODBD(I) - PRODHD(I)
          END DO
C
C         DETERMINATION DE LA TEMPERATURE ET DU FLUX SELON LE CAS
C
          DO I=1,NOF_M
            IF (TY(I).EQ.1) THEN
               TEMP_F(I)=VAL_INC(I)
            ELSE IF (TY(I).EQ.2) THEN
              FLUX(I,2)=-VAL_INC(I)*K
            ELSE IF (TY(I).EQ.3) THEN
              FLUX(I,1)=-VAL_INC(I)*K
            ELSE IF (TY(I).EQ.5) THEN
              FLUX(I,1)=-VAL_INC(I)*K
```

```
            FLUX(I,2)=-VAL_INC(I)*K
          END IF
        END DO
C
C       VERIFICATION DE LA FIN DE LA CONDUCTION
C
        CALL VERI_TEMP_FUSION(TY,ABLA,FLUX,VAL,VAL_INC,G,H,
     +                        GD,HD,NE_G)
        I=1
        DO WHILE ((I.LE.NOF_M).AND.(.NOT.FIN_COND))
          IF (ABLA(I)) FIN_COND=.TRUE.
          I=I+1
        END DO
C
C       CALCUL DE L'ERREUR RELATIVE SUR LE BILAN D'ENERGIE ET
C       DE L'ENERGIE CUMULEE ,ET IMPRRESSION DE RESULTATS
C
        DO I=1,N_ELE
          ENER_IN=ENER_IN+AIRE(I)*RHO*CP*(T(I)-T_AV(I))
        END DO
        ENER_IN_IMP=ENER_IN_IMP+ENER_IN
        ENER_IN_TOT=ENER_IN_TOT+ENER_IN
        DO I=1,NOF_M-1
          ENER_D=ENER_D+LXY(I)*(-FLUX(I,2)-FLUX(I+1,1))*0.5
        END DO
        IF (FERMER) ENER_D=ENER_D+LXY(NOF)*
     +                     (-FLUX(NOF,2)-FLUX(1,1))*0.5
        ENER_D=ENER_D*(PAS-PAS_AV)
        ENER_D_IMP=ENER_D_IMP+ENER_D
        ENER_D_TOT=ENER_D_TOT+ENER_D
        IF ((JMOD(JNINT(PAS),DT_ERR).EQ.0).OR.(FIN_COND)) THEN
          ERREUR=100.0*(ENER_IN_IMP-ENER_D_IMP)/ENER_D_IMP
          WRITE(3,25) PAS/60.0,ERREUR
          ENER_D_IMP=0.0
          ENER_IN_IMP=0.0
        END IF
        IF ((JMOD(JNINT(PAS),DT_IMP_T).EQ.0).OR.(FIN_COND)) THEN
          WRITE(1,*) PAS
          WRITE(8,*) PAS
          DO I=1,NOF_M
            IF ((TY(I).EQ.1).OR.ABLA(I)) THEN
              C='T'
              VAL_IMP=VAL_INC(I)+FUSION-KELVIN
            ELSE
              C='G'
              VAL_IMP=VAL_INC(I)
            END IF
            WRITE(1,35) I,XF(I),YF(I),C,VAL_IMP
            WRITE(8,10) I,XF(I),YF(I),FLUX(I,1)
          END DO
          WRITE(2,*) PAS
          I=0
```

```
            DO L=1,NB_COUCHE-1
              M=(L-1)*NB_TRI
              DO J=M+1,M+N2
                I=I+1
                WRITE(2,5) I,CGT(J,1),CGT(J,2),T(I)+FUSION-KELVIN
              END DO
            END DO
            M=(NB_COUCHE-1)*NB_TRI
            DO J=M+1,M+N1
              I=I+1
              WRITE(2,5) I,CGT(J,1),CGT(J,2),T(I)+FUSION-KELVIN
            END DO
          END IF
          DO I=1,NOF_M
            IF (TY(I).EQ.1) TEMP_F(I)=VAL_INC(I)
            IF (ABLA(I)) TEMP_F(I)=0.0
          END DO
          CALL FLUX_DIST(XF,YF,FLUX_NOEUD,TEMP_F,PAS+DT)
          DO I=1,NOF_M
            FLUX(I,1)=FLUX_NOEUD(I)
            FLUX(I,2)=FLUX_NOEUD(I)
          END DO
          CALL ASSIG_VAL_CONNUE(TY,FLUX,TEMP_F,VAL,NE_G)
        END DO
        CLOSE(3)
        TEMPS_COND=PAS
        WRITE(15,55) PAS/60.0
55      FORMAT(X,/,11(' '),'<<<<< RESULTATS DE LA SIMULATION >>>>>',
     +         2(/),2X,60('-'),2(/),
     +         2X,'DUREE DE LA PHASE DE PREFUSION',T45,': ',F8.2,
     +         ' min.',/)
        IF (JNINT(PAS).GT.0) THEN
          ERREUR=100.0*(ENER_IN_TOT-ENER_D_TOT)/ENER_D_TOT
          WRITE(15,60) ERREUR
60        FORMAT(2X,'ERREUR RELATIVE COMMISE SUR BILAN D''ENERGIE
     + NET DE',/,
     +           2X,'LA PHASE DE PREFUSION',T45,': ',F7.2,' %',/)
        END IF
C
C       DEBUT DE L'ABLATION.
C
        EL_AV=EL
        N_ELE_AV=N_ELE
        NOM='POS_FRONT.DAT'
        NOMFICH=ENDROIT//NOM
        OPEN(3,FILE=NOMFICH,STATUS='UNKNOWN')
C
C         IMPRESSION DES RESULTATS.
C
C
C       CONSTRUCTION DES FICHIERS DE SORTIE.
C
```

```
      WRITE(3,11)
11    FORMAT(2X,'NOEUD    X (m)    Y (m)',/)
C
C
      STEADY_STATE=.FALSE.
      DO WHILE (.NOT.STEADY_STATE)
        PAS_AV=PAS
        PAS=PAS+DT
        CALL PRODUIT_MAT(B,T,PRODB,NOE_M,N_ELE_AV,NB_E,NB_E)
        CALL PRODUIT_MAT(G,VAL,PRODG,NOE_M,NE_G,NB_E,2*NB_E)
        DO I=1,NOE_M
          H(I,NOE_M+1)=PRODG(I)+PRODB(I)
          UNITE=ABS(H(I,I))
          IF (UNITE.LT.EPSILON) THEN
            WRITE(15,80) I,JNINT(PAS)
80          FORMAT(2X,'L''ELEMENT ',I2,' DE LA DIAGONALE EST NUL AU'
     +           ,/,2X,'TEMPS T = ',I7,' s.')
            CALL CPUTIME
            STOP
          END IF
          EXPOS=0
          DO WHILE (UNITE.LT.1.0)
            UNITE=UNITE*10.0
            EXPOS=EXPOS+1
          END DO
          IF (EXPOS.GT.0) THEN
            DO J=1,NOE_M+1
              H(I,J)=H(I,J)*(10.0**EXPOS)
            END DO
          END IF
        END DO
        DETER=SIMUL(NB_E,NOE_M,H,VAL_INC,1)
        IF (DETER.EQ.0.0) THEN
          WRITE(15,65) JNINT(PAS)
65        FORMAT(2X,'LE DET(H) = 0 POUR LA PHASE DE FUSION A T = ',
     +           I7,' s.')
          CALL CPUTIME
          STOP
        END IF
        STEADY_STATE=.TRUE.
        I=1
        DO WHILE ((STEADY_STATE).AND.(I.LE.NOE_M))
          IF ((ABLA(I)).AND.(ABS(VAL_INC(I)).GT.(1.0E-3))) THEN
            STEADY_STATE=.FALSE.
          ELSE
            I=I+1
          END IF
        END DO
        CALL PRODUIT_MAT(GD,VAL,PRODGD,N_ELE,NE_G,NB_E,2*NB_E)
        CALL PRODUIT_MAT(HD,VAL_INC,PRODHD,N_ELE,NOE_M,NB_E,NB_E+1)
        CALL PRODUIT_MAT(BD,T,PRODBD,N_ELE,N_ELE_AV,NB_E,NB_E)
        DO I=1,N_ELE
```

```
            T(I) = PRODGD(I) + PRODBD(I) - PRODHD(I)
          END DO
          DO I=1,EL
            DO J=1,3
              E_AV(I,J)=E(I,J)
            END DO
          END DO
          DO I=1,NOI
            XI_AV(I)=XI(I)
            YI_AV(I)=YI(I)
          END DO
          EL_AV=EL
          NB_COUCHE_AV=NB_COUCHE
          N_ELE_AV=N_ELE
C
C         IMPRESSION DES RESULTATS.
C
C
C         CONSTRUCTION DES FICHIERS DE SORTIE.
C
          IF ((JMOD(JNINT(PAS-TEMPS_COND-DT_C),DT_IMP_P).EQ.0)
     +        .OR.(STEADY_STATE)) THEN
            WRITE(3,*) PAS,NOF_M
            DO I=1,NOF_M
              WRITE(3,20) I,XF(I),YF(I)
            END DO
          END IF
          IF ((JMOD(JNINT(PAS-TEMPS_COND-DT_C),DT_IMP_T).EQ.0)
     +        .OR.(STEADY_STATE)) THEN
            WRITE(1,*) PAS
            WRITE(8,*) PAS
            DO I=1,NOF_M
              IF (TY(I).EQ.1) THEN
                C='T'
                VAL_IMP=VAL_INC(I)+FUSION-KELVIN
              ELSE
                C='G'
                VAL_IMP=VAL_INC(I)
              END IF
              WRITE(1,35) I,XF(I),YF(I),C,VAL_IMP
              WRITE(8,10) I,XF(I),YF(I),FLUX(I,1)
            END DO
            WRITE(2,*) PAS
            IF (FERMER) THEN
              N1=NOF
            ELSE
              N1=NOF_M-1
            END IF
            N2=3*N1
            NB_TRI=NOF*3
            I=0
            DO L=1,NB_COUCHE-1
```

```
            M=(L-1)*NB_TRI
            DO J=M+1,M+N2
              I=I+1
              WRITE(2,5) I,CGT(J,1),CGT(J,2),T(I)+FUSION-KELVIN
            END DO
          END DO
          M=(NB_COUCHE-1)*NB_TRI
          DO J=M+1,M+N1
            I=I+1
            WRITE(2,5) I,CGT(J,1),CGT(J,2),T(I)+FUSION-KELVIN
          END DO
        END IF
C
C       DEPLACEMENT DE LA FRONTIERE.
C
        IF (.NOT.STEADY_STATE) THEN
          ENER_D=0.0
          DO I=1,NOE_M-1
            ENER_D=ENER_D+LXY(I)*(-FLUX(I,2)-FLUX(I+1,1))*0.5
          END DO
          IF (FERMER) ENER_D=ENER_D+LXY(NOE)*
     +                       (-FLUX(NOE,2)-FLUX(1,1))*0.5
          ENER_D=ENER_D*(PAS-PAS_AV)
          ENER_D_TOT=ENER_D_TOT+ENER_D
          CALL DEP_FRONT(XF,YF,FLUX,VAL_INC,DS,DT,TY,ABLA)
          CALL DIMIN_NB_COUCHE(T)
          CALL MAILLAGE_AUTO(XF,YF,XI,YI,E)
          CALL CENTRE_GRAV_TRI(E,XI,YI,CGT)
          IF (ABS(DT-DT_A).GT.EPSILON) THEN
            DT=DT_A
            DENOM=4.0*DT*ALPHA
          END IF
          CALL MATRICE_B(XI_AV,YI_AV,E_AV,XF,YF,B,EL_AV)
          CALL MATRICE_BD(XI_AV,YI_AV,E_AV,CGT,BD,EL_AV)
          CALL MATRICES_H_G(XF,YF,H,G,TY,ABLA,NE_G)
          CALL MATRICES_HD_GD(XF,YF,CGT,HD,GD,TY,ABLA,NE_G)
          CALL VERI_TEMP_FUSION(TY,ABLA,FLUX,VAL,VAL_INC,G,H,
     +                          GD,HD,NE_G)
          DO I=1,NOE_M
            IF (TY(I).EQ.1) TEMP_F(I)=VAL_INC(I)
            IF (ABLA(I)) TEMP_F(I)=0.0
          END DO
          CALL FLUX_DIST(XF,YF,FLUX_NOEUD,TEMP_F,PAS+DT)
          DO I=1,NOE_M
            FLUX(I,1)=FLUX_NOEUD(I)
            FLUX(I,2)=FLUX_NOEUD(I)
          END DO
          CALL ASSIG_VAL_CONNUE(TY,FLUX,TEMP_F,VAL,NE_G)
        END IF
      END DO
      CLOSE(1)
      CLOSE(2)
```

```
      CALL AIRE_TRIANGLES(E,XI,YI,AIRE)
      AIRE_TOT=0.0
      DO I=1,N_ELE
        AIRE_TOT=AIRE_TOT+AIRE(I)
      END DO
      DT=DT_DEP
      DO I=1,NOF
        VAL_INC(I)=0.0
      END DO
      NB_COUCHE=1
      POURC_SOLIDE=(AIRE_TOT/AIRE_TOT_INIT)*100.0
      IF (POURC_SOLIDE.LT.1.0) THEN
        TERMINER=.TRUE.
      ELSE
        TERMINER=.FALSE.
      END IF
      DO WHILE (.NOT.TERMINER)
        CALL FLUX_DIST(XF,YF,FLUX_NOEUD,TEMP_F,PAS)
        DO I=1,NOF_M
          FLUX(I,1)=FLUX_NOEUD(I)
          FLUX(I,2)=FLUX_NOEUD(I)
        END DO
        PAS_AV=PAS
        PAS=PAS+DT
        ENER_D=0.0
        DO I=1,NOF_M-1
          ENER_D=ENER_D+LXY(I)*(-FLUX(I,2)-FLUX(I+1,1))*0.5
        END DO
        IF (FERMER) ENER_D=ENER_D+LXY(NOF)*
     +                    (-FLUX(NOF,2)-FLUX(I,1))*0.5
        ENER_D_TOT=ENER_D_TOT+ENER_D*(PAS-PAS_AV)
        CALL DEP_FRONT(XF,YF,FLUX,VAL_INC,DS,DI,TY,ABLA)
        CALL MAILLAGE_AUTO(XF,YF,XI,YI,E)
        CALL LONGUEUR(XF,YF)
        CALL AIRE_TRIANGLES(E,XI,YI,AIRE)
        AIRE_TOT=0.0
        DO I=1,N_ELE
          AIRE_TOT=AIRE_TOT+AIRE(I)
        END DO
        POURC_SOLIDE=(AIRE_TOT/AIRE_TOT_INIT)*100.0
        IF (POURC_SOLIDE.LT.1.0) TERMINER=.TRUE.
C
C       IMPRESSION DES RESULTATS.
C
C
C       CONSTRUCTION DES FICHIERS DE SORTIE.
C
        IF ((JMOD(JNINT(PAS-TEMPS_COND-DT_C),DT_IMP_P).EQ.0)
     +       .OR.(TERMINER)) THEN
          WRITE(3,*) PAS,NOF_M
          DO I=1,NOF_M
            WRITE(3,20) I,XF(I),YF(I)
```

```
          END DO
        END IF
        IF ((JMOD(JNINT(PAS-TEMPS_COND-DT_C),DT_IMP_T).EQ.0)
     +      .OR.(TERMINER)) THEN
          WRITE(8,*) PAS
          DO I=1,NOE_M
            WRITE(8,10) I,XF(I),YF(I),FLUX(I,1)
          END DO
        END IF
      END DO
      ENER_IN_TOT=(AIRE_TOT_INIT-AIRE_TOT)*RHO_L
      ENER_IN=0.0
      DO I=1,N_ELE_INIT
        ENER_IN=ENER_IN+AIRE_INIT(I)*(-T_INIT(I))
      END DO
      ENER_IN_TOT=ENER_IN_TOT+ENER_IN*RHO*CP
      IF (ABS(ENER_IN_TOT).LT.EPSILON) THEN
        ERREUR=0.0
      ELSE
        ERREUR=100.0*(ENER_IN_TOT-ENER_D_TOT)/ENER_IN_TOT
      END IF
      WRITE(15,70) POURC_SOLIDE,ERREUR
70    FORMAT(2X,'ERREUR RELATIVE COMMISE SUR LE BILAN D''ENERGIE NET
     + DE',/,
     +       2X,'TOUT LE PROCEDE D''ABLATION JUSQU''A CE QU''IL NE',/,
     +       2X,'RESTE PLUS QUE ',F5.2,' % DU SOLIDE INITIAL',T45,
     +       ': ',F8.2,' %',/)
      WRITE(15,85) PAS/60.0
85    FORMAT(2X,'TEMPS REEL DE SIMULATION',T45,': ',F8.2,' min.',/)
      PRINT *
      PRINT *,'##########################################################'
      PRINT *,'##                                                      ##'
      PRINT *,'##                                                      ##'
      PRINT *,'##            FIN DE LA SIMULATION                      ##'
      PRINT *,'##                                                      ##'
      PRINT *,'##                                                      ##'
      PRINT *,'##########################################################'
5     FORMAT(2X,I3,3X,2(F6.3,2X),F9.1)
10    FORMAT(2X,I3,3X,2(F6.3,2X),2(F9.1,2X))
20    FORMAT(2X,I3,3X,2(F6.3,2X))
25    FORMAT(2X,F6.2,3X,F6.2)
30    FORMAT(2X,F6.2,3X,F12.4)
35    FORMAT(2X,I3,3X,2(F6.3,2X),A,3X,F9.1)
100   FORMAT(2X,I3,3X,F9.7)
      CLOSE(3)
      CLOSE(8)
      CLOSE(15)
C
C--------------
C
      RETURN
      END
```

```
      SUBROUTINE VAL_INT1
C
C-----------------------------------------------
C
C     ROUTINE CALCULANT LES VALEURS POUR L'INTE-
C     GRATION A UNE DIMENSION PAR LA METHODE
C     DE GAUSS-LEGENDRE ET CECI POUR DIFFERENTS
C     NOMBRES DE POINTS.
C
C-----------------------------------------------
C
      REAL W1(12,12),PT(12,12)
      INTEGER IORD
      COMMON/INTEG1/IORD,W1,PT
C
      PT(1,1)=0.0E0
      PT(2,1)=+0.577350269189626E0
      PT(2,2)= -PT(2,1)
      PT(3,1)=0.0E0
      PT(3,2)=+0.774596669241483E0
      PT(3,3)=-PT(3,2)
      PT(4,1)=+0.339981043584856E0
      PT(4,2)=-PT(4,1)
      PT(4,3)=+0.861136311594053E0
      PT(4,4)= -PT(4,3)
      PT(5,1)=0.0E0
      PT(5,2)=+0.538469310105683E0
      PT(5,3)= -PT(5,2)
      PT(5,4)=+0.906179845938664E0
      PT(5,5)= -PT(5,4)
      PT(6,1)=+0.238619186083197E0
      PT(6,2)= -PT(6,1)
      PT(6,3)= +0.661209386466265E0
      PT(6,4)= -PT(6,3)
      PT(6,5)= +0.932469514203152E0
      PT(6,6)= -PT(6,5)
      PT(7,1)= 0.0E0
      PT(7,2)= +0.405845151377397E0
      PT(7,3)= -PT(7,2)
      PT(7,4)= +0.741531185599394E0
      PT(7,5)= -PT(7,4)
      PT(7,6)=+0.949107912342759E0
      PT(7,7)= -PT(7,6)
      PT(8,1)=+0.183434642495650E0
      PT(8,2)= -PT(8,1)
      PT(8,3)= 0.525532409916329E0
      PT(8,4)= -PT(8,3)
      PT(8,5)=+0.796666477413627E0
      PT(8,6)= -PT(8,5)
```

```
PT(8,7)=+0.9602898564975 36E0
PT(8,8)= -PT(8,7)
PT(9,1)=+0.0E0
PT(9,2)=+0.3242534234038 09E0
PT(9,3)= -PT(9,2)
PT(9,4)=+0.6133714327005 90E0
PT(9,5)= -PT(9,4)
PT(9,6)=+0.8360311073266 36E0
PT(9,7)= -PT(9,6)
PT(9,8)=+0.9681602395076 26E0
PT(9,9)= -PT(9,8)
PT(10,1)=+0.1488743389816 31E0
PT(10,2)= -PT(10,1)
PT(10,3)= 0.4333953941292 47E0
PT(10,4)= -PT(10,3)
PT(10,5)=+0.6794095682990 24E0
PT(10,6)= -PT(10,5)
PT(10,7)=+0.8650633666889 85E0
PT(10,8)= -PT(10,7)
PT(10,9)=+0.9739065285171 72E0
PT(10,10)= -PT(10,9)
PT(12,1)=0.1252334085114 69E0
PT(12,2)=-PT(12,1)
PT(12,3)=0.3678314989981 80E0
PT(12,4)=-PT(12,3)
PT(12,5)=0.5873179542866 17E0
PT(12,6)=-PT(12,5)
PT(12,7)=0.7699026741943 05E0
PT(12,8)=-PT(12,7)
PT(12,9)=0.9041172563704 75E0
PT(12,10)=-PT(12,9)
PT(12,11)=0.9815606342467 19E0
PT(12,12)=-PT(12,11)
W1(1,1)=2.0E0
W1(2,1)=1.0E0
W1(2,2)=1.0E0
W1(3,1)=0.8888888888888 89E0
W1(3,2)=0.5555555555555 56E0
W1(3,3)=W1(3,2)
W1(4,1)=0.6521451548625 46E0
W1(4,2)=W1(4,1)
W1(4,3)=0.3478548451374 54E0
W1(4,4)=W1(4,3)
W1(5,1)=0.5688888888888 89E0
W1(5,2)=0.4786286704993 66E0
W1(5,3)=W1(5,2)
W1(5,4)=0.2369268850561 89E0
W1(5,5)=W1(5,4)
W1(6,1)=0.4679139345726 91E0
W1(6,2)=W1(6,1)
W1(6,3)=0.3607615730481 39E0
W1(6,4)=W1(6,3)
```

```
      W1(6,5)=0.1713244923791 70E0
      W1(6,6)=W1(6,5)
      W1(7,1)=0.4179591836734 69E0
      W1(7,2)=0.3818300505051 19E0
      W1(7,3)=W1(7,2)
      W1(7,4)=0.2797053914892 77E0
      W1(7,5)=W1(7,4)
      W1(7,6)=0.1294849661688 70E0
      W1(7,7)=W1(7,6)
      W1(8,1)=0.3626837833783 62E0
      W1(8,2)=W1(8,1)
      W1(8,3)=0.3137066458778 87E0
      W1(8,4)=W1(8,3)
      W1(8,5)=0.2223810344533 74E0
      W1(8,6)=W1(8,5)
      W1(8,7)=0.1012285362903 76E0
      W1(8,8)=W1(8,7)
      W1(9,1)=0.3302393550012 60E0
      W1(9,2)=0.3123470770400 03E0
      W1(9,3)=W1(9,2)
      W1(9,4)=0.2606106964029 35E0
      W1(9,5)=W1(9,4)
      W1(9,6)=0.1806481606948 57E0
      W1(9,7)=W1(9,6)
      W1(9,8)=0.0812743883615 74E0
      W1(9,9)=W1(9,6)
      W1(10,1)=0.2955242247147 53E0
      W1(10,2)=W1(10,1)
      W1(10,3)=0.2692667193099 96E0
      W1(10,4)=W1(10,3)
      W1(10,5)=0.2190863625159 82E0
      W1(10,6)=W1(10,5)
      W1(10,7)=0.1494513491505 81E0
      W1(10,8)=W1(10,7)
      W1(10,9)=0.0666713443086 88E0
      W1(10,10)=W1(10,9)
      W1(12,1)=0.2491470458134 03E0
      W1(12,2)=W1(12,1)
      W1(12,3)=0.2334925365383 55E0
      W1(12,4)=W1(12,3)
      W1(12,5)=0.2031674267230 66E0
      W1(12,6)=W1(12,5)
      W1(12,7)=0.1600783285433 46E0
      W1(12,8)=W1(12,7)
      W1(12,9)=0.1069393259953 18E0
      W1(12,10)=W1(12,9)
      W1(12,11)=0.0471753363865 12E0
      W1(12,12)=W1(12,11)
C
C-----------
C
      RETURN
```

```
      END


      SUBROUTINE VAL_INTl_LAG
C
C-----------------------------------------------
C
C     ROUTINE CALCULANT LES VALEURS POUR L'INTE-
C     GRATION NUMERIQUE A UNE DIMENSION PAR LA
C     METHODE DE GAUSS-LAGUERRE ET CECI POUR
C     DIFFERENTS NOMBRES DE POINTS.
C
C-----------------------------------------------
C
      REAL WZ(15,15),PZ(15,15)
      INTEGER IORD
      COMMON/INTEGl_I/IORD,WZ,PZ
C
      PZ(1,1)=0.0E0
      PZ(2,1)=0.585786437627E0
      PZ(2,2)=3.414213562373E0
      PZ(3,1)=0.415774556783E0
      PZ(3,2)=2.294280360279E0
      PZ(3,3)=6.289945082937E0
      PZ(4,1)=0.322547689619E0
      PZ(4,2)=1.745761101158E0
      PZ(4,3)=4.536620296921E0
      PZ(4,4)=9.39507091230lE0
      PZ(5,1)=0.263560319718E0
      PZ(5,2)=1.413403059107E0
      PZ(5,3)=3.596425771041E0
      PZ(5,4)=7.085810005859E0
      PZ(5,5)=12.640800844276E0
      PZ(6,1)=0.222846604179E0
      PZ(6,2)=1.188932101673E0
      PZ(6,3)=2.992736326059E0
      PZ(6,4)=5.775143569105E0
      PZ(6,5)=9.837467418383E0
      PZ(6,6)=15.982873980602E0
      PZ(10,1)=0.137793470540E0
      PZ(10,2)=0.729454549503E0
      PZ(10,3)=1.808342901740E0
      PZ(10,4)=3.401433697855E0
      PZ(10,5)=5.552496140064E0
      PZ(10,6)=8.330152746764E0
      PZ(10,7)=11.843785837900E0
      PZ(10,8)=16.279257831378E0
      PZ(10,9)=21.996585811981E0
      PZ(10,10)=29.920697012274E0
      PZ(15,1)=0.093307812017E0
      PZ(15,2)=0.492691740302E0
```

```
PZ(15,3)=1.215595412071E0
PZ(15,4)=2.269949526204E0
PZ(15,5)=3.6676227217S1E0
PZ(15,6)=5.425336627414E0
PZ(15,7)=7.565916226613E0
PZ(15,8)=10.120228568019E0
PZ(15,9)=13.130282482176E0
PZ(15,10)=16.654407708330E0
PZ(15,11)=20.776478899449E0
PZ(15,12)=25.623894226729E0
PZ(15,13)=31.407519169754E0
PZ(15,14)=38.530683306486E0
PZ(15,15)=48.026085572686E0
WZ(1,1)=0.0E0
WZ(2,1)=0.853553390593E0
WZ(2,2)=0.146446609407E0
WZ(3,1)=0.711093009929E0
WZ(3,2)=0.278517733569E0
WZ(3,3)=0.103892565016E-1
WZ(4,1)=0.603154104342E0
WZ(4,2)=0.357418692438E0
WZ(4,3)=0.388879085150E-1
WZ(4,4)=0.539294705561E-3
WZ(5,1)=0.521755610583E0
WZ(5,2)=0.398666811083E0
WZ(5,3)=0.0759424496817E0
WZ(5,4)=0.00361175867992E0
WZ(5,5)=0.00002336997238S8E0
WZ(6,1)=0.458964673950E0
WZ(6,2)=0.417000830772E0
WZ(6,3)=0.113373382074E0
WZ(6,4)=0.103991974531E-1
WZ(6,5)=0.261017202815E-3
WZ(6,6)=0.898547906430E-6
WZ(10,1)=0.308441115765E0
WZ(10,2)=0.401119929155E0
WZ(10,3)=0.218068287612E0
WZ(10,4)=0.620874560987E-1
WZ(10,5)=0.950151697518E-2
WZ(10,6)=0.753008388588E-3
WZ(10,7)=0.282592334960E-4
WZ(10,8)=0.424931398496E-6
WZ(10,9)=0.183956482398E-8
WZ(10,10)=0.991182721961E-12
WZ(15,1)=0.218234885940E0
WZ(15,2)=0.342210177923E0
WZ(15,3)=0.263027577942E0
WZ(15,4)=0.126425818106E0
WZ(15,5)=0.402068649210E-1
WZ(15,6)=0.856387780361E-2
WZ(15,7)=0.121243614721E-2
WZ(15,8)=0.111674392344E-3
```

```
        WZ(15,9)=0.645992676202E-5
        WZ(15,10)=0.222631690710E-6
        WZ(15,11)=0.422743038498E-8
        WZ(15,12)=0.392189726704E-10
        WZ(15,13)=0.145651526407E-12
        WZ(15,14)=0.148302705111E-15
        WZ(15,15)=0.160059490621E-19
C
C------------
C
        RETURN
        END


        SUBROUTINE FACTORIEL
C
C-------------------------------------------------------
C
C      ROUTINE CALCULANT LES MAX PREMIERS FACTORIELLES
C
C-------------------------------------------------------
C
        PARAMETER (MSOM = 20)
        REAL FACT(MSOM+1)
        INTEGER I
        COMMON/FACTOR/FACT
C
        FACT(1) = 1.0
        DO I=2,MSOM+1
          FACT(I) = FACT(I-1)*I
        END DO
C
C---------
C
        RETURN
        END


        SUBROUTINE MAILLAGE_AUTO(XF,YF,XI,YI,E)
C
C-------------------------------------------------------
C
C        ROUTINE SERVANT A FAIRE LE MAILLAGE AUTO-
C        MATIQUE D'UN DOMAINE CONVEXE.
C
C
C-------------------------------------------------------
C
        PARAMETER(NB_F = 32,NB_E = 532,NB_D = 365,NB_DIV = 10)
        REAL XF(0:NB_F+1),YF(0:NB_F+1),XI(NB_D),YI(NB_D),LAMBDA(NB_DIV)
        INTEGER E(NB_E,3),NB_COUCHE,EL,NOF,NOF_M,M,N,N_ELE
```

```
      LOGICAL FERMER
      COMMON/MESH/LAMBDA,NB_COUCHE/FRONT/NOF,NOF_M/DOMM/EL,NOI,N_ELE
      COMMON/CONTOUR/FERMER
C
      CGDX=0
      CGDY=0
      IF (FERMER) THEN
        DO I=1,NOF
          CGDX = CGDX + XF(I)
          CGDY = CGDY + YF(I)
        END DO
        CGDX = CGDX/NOF
        CGDY = CGDY/NOF
        N1=NOF
      ELSE
        N1=NOF_M-1
      END IF
      N2=3*N1
      I=1
      L=1
      IF (NB_COUCHE.GT.1) THEN
        DO I=1,NB_COUCHE-2
          M=1
          KCONST1=2*(I-1)*NOF
          KCONST2=2*I*NOF
          CONST3=LAMBDA(I)*CGDX
          CONST4=LAMBDA(I)*CGDY
          CONST5=1.0-LAMBDA(I)
          DO J=1,NOF-1
            E(L,1)=KCONST1+M
            E(L,2)=KCONST1+M+1
            E(L,3)=KCONST2+M
            E(L+1,1)=E(L,2)
            E(L+1,2)=KCONST2+M+2
            E(L+1,3)=E(L,3)
            E(L+2,1)=E(L,2)
            E(L+2,2)=KCONST1+M+2
            E(L+2,3)=E(L+1,2)
            XI(E(L,1))=CONST3+CONST5*XF(J)
            YI(E(L,1))=CONST4+CONST5*YF(J)
            XI(E(L,2))=(XI(E(L,1))+(CONST3+CONST5*XF(J+1)))/2.0
            YI(E(L,2))=(YI(E(L,1))+(CONST4+CONST5*YF(J+1)))/2.0
            M=M+2
            L=L+3
          END DO
          E(L,1)=KCONST2-1
          E(L,2)=KCONST2
          E(L,3)=KCONST2+M
          E(L+1,1)=E(L,2)
          E(L+1,2)=KCONST2+1
          E(L+1,3)=E(L,3)
          E(L+2,1)=E(L,2)
```

```
      E(L+2,2)=KCONST1+1
      E(L+2,3)=E(L+1,2)
      XI(E(L,1))=CONST3+CONST5*XF(NOF)
      YI(E(L,1))=CONST4+CONST5*YF(NOF)
      XI(E(L,2))=(XI(E(L,1))+(CONST3+CONST5*XF(1)))/2.0
      YI(E(L,2))=(YI(E(L,1))+(CONST4+CONST5*YF(1)))/2.0
      L=L+3
    END DO
    KCONST1=2*(NB_COUCHE-2)*NOF
    KCONST2=2*(NB_COUCHE-1)*NOF
    CONST3=LAMBDA(NB_COUCHE-1)*CGDX
    CONST4=LAMBDA(NB_COUCHE-1)*CGDY
    CONST5=1.0-LAMBDA(NB_COUCHE-1)
    M=1
    DO J=1,NOF-1
      E(L,1)=KCONST1+M
      E(L,2)=KCONST1+M+1
      E(L,3)=KCONST2+J
      E(L+1,1)=E(L,2)
      E(L+1,2)=KCONST2+J+1
      E(L+1,3)=E(L,3)
      E(L+2,1)=E(L,2)
      E(L+2,2)=KCONST1+M+2
      E(L+2,3)=E(L+1,2)
      XI(E(L,1))=CONST3+CONST5*XF(J)
      YI(E(L,1))=CONST4+CONST5*YF(J)
      XI(E(L,2))=(XI(E(L,1))+(CONST3+CONST5*XF(J+1)))/2.0
      YI(E(L,2))=(YI(E(L,1))+(CONST4+CONST5*YF(J+1)))/2.0
      M=M+2
      L=L+3
    END DO
    E(L,1)=KCONST2-1
    E(L,2)=KCONST2
    E(L,3)=KCONST2+NOF
    E(L+1,1)=E(L,2)
    E(L+1,2)=KCONST2+1
    E(L+1,3)=E(L,3)
    E(L+2,1)=E(L,2)
    E(L+2,2)=KCONST1+1
    E(L+2,3)=E(L+1,2)
    XI(E(L,1))=CONST3+CONST5*XF(NOF)
    YI(E(L,1))=CONST4+CONST5*YF(NOF)
    XI(E(L,2))=(XI(E(L,1))+(CONST3+CONST5*XF(1)))/2.0
    YI(E(L,2))=(YI(E(L,1))+(CONST4+CONST5*YF(1)))/2.0
    L=L+3
  END IF
  KCONST1=2*(NB_COUCHE-1)*NOF
  KCONST2=NOF*(2*NB_COUCHE-1)+1
  CONST3=LAMBDA(NB_COUCHE)*CGDX
  CONST4=LAMBDA(NB_COUCHE)*CGDY
  CONST5=1.0-LAMBDA(NB_COUCHE)
  DO J=1,NOF-1
```

```
          E(L,1)=KCONST1+J
          E(L,2)=KCONST1+J+1
          E(L,3)=KCONST2
          XI(E(L,1))=CONST3+CONST5*XF(J)
          YI(E(L,1))=CONST4+CONST5*YF(J)
          L=L+1
        END DO
        E(L,1)=KCONST2-1
        E(L,2)=KCONST1+1
        E(L,3)=KCONST2
        XI(E(L,1))=CONST3+CONST5*XF(NOF)
        YI(E(L,1))=CONST4+CONST5*YF(NOF)
        XI(E(L,3))=CGDX
        YI(E(L,3))=CGDY
        NOI=KCONST2
        EL=3*(NB_COUCHE-1)*NOF+NOF
        N_ELE=N2*(NB_COUCHE-1)+N1
        XF(0)=XF(NOF)
        YF(0)=YF(NOF)
        XF(NOF+1)=XF(1)
        YF(NOF+1)=YF(1)
C
C------------
C
        RETURN
        END



        SUBROUTINE CENTRE_GRAV_TRI(E,XI,YI,CGT)
C
C-------------------------------------------------
C
C    ROUTINE CALCULANT LE CENTRE DE GRAVITE DES
C    ELEMENTS TRIANGULAIRES
C
C-------------------------------------------------
C
        PARAMETER (NB_E = 532,NB_D = 365)
        REAL XI(NB_D),YI(NB_D),CGT(NB_E,2)
        INTEGER E(NB_E,3),EL
        COMMON/DOMM/EL
C
        DO I=1,EL
          CGT(I,1) = (XI(E(I,1)) + XI(E(I,2)) + XI(E(I,3)))/3
          CGT(I,2) = (YI(E(I,1)) + YI(E(I,2)) + YI(E(I,3)))/3
        END DO
C
C-----------
C
        RETURN
        END
```

```
      SUBROUTINE LONGUEUR(XF,YF)
C
C-----------------------------------------------
C
C        ROUTINE CALCULANT LA LONGUEUR DES
C        ELEMENTS A LA FRONTIERE.(M.)
C
C-----------------------------------------------
C
      PARAMETER (NB_F = 32)
      REAL XF(0:NB_F+1),YF(0:NB_F+1),LXY(0:NB_F+1)
      COMMON/LONG/LXY/FRONT/NOF
C
      DO J=1,NOF
        LXY(J) = SQRT((XF(J+1)-XF(J))**2 + (YF(J+1)-YF(J))**2)
      END DO
      LXY(0)=LXY(NOF)
      LXY(NOF+1)=LXY(1)
C
C----------------
C
      RETURN
      END


      SUBROUTINE ASSIG_VAL_CONNUE(TY,FLUX,TEMP_F,VAL,NE_G)
C
C-------------------------------------------------------------------
C
C     ROUTINE SERVANT A ASSIGNER AU VECTEUR COLONNE 'VAL' LES VALEURS
C     CONNUES POUR SOLUTIONNER LE SYSTEME D'EQUATIONS LINEAIRES.
C
C-------------------------------------------------------------------
C
      PARAMETER(NB_F = 32)
      REAL FLUX(NB_F,2),TEMP_F(NB_F),VAL(NB_F),K,ALPHA,RHO_L
      INTEGER TY(NB_F),NE_G,L,I,NOF,NOF_M
      LOGICAL FERMER
      COMMON/FRONT/NOF,NOF_M/PROP/ALPHA,K,RHO_L/CONTOUR/FERMER
C
C
      L=3
      DO I=2,NOF_M
        IF (TY(I).EQ.1) THEN
          IF (I.EQ.NOF_M) THEN
C
C           DU/DN=-FLUX/K
C
```

```
            VAL(L-1)=-FLUX(I,1)/K
            IF (FERMER) VAL(L)=-FLUX(I,2)/K
          ELSE
            VAL(L-1)=-FLUX(I,1)/K
            VAL(L)=-FLUX(I,1)/K
          END IF
        ELSE IF (TY(I).EQ.2) THEN
          VAL(L-1)=-FLUX(I,1)/K
          VAL(L)=TEMP_F(I)
        ELSE IF (TY(I).EQ.3) THEN
          VAL(L-1)=TEMP_F(I)
          VAL(L)=-FLUX(I,2)/K
        ELSE IF ((TY(I).EQ.4).OR.(TY(I).EQ.5)) THEN
          VAL(L-1)=TEMP_F(I)
          L=L-1
        END IF
        L=L+2
      END DO
      IF (FERMER) THEN
        IF (TY(1).EQ.1) THEN
C
C         DU/DN=-FLUX/K
C
          VAL(NE_G)=-FLUX(1,1)/K
          VAL(1)=-FLUX(1,2)/K
        ELSE IF (TY(1).EQ.2) THEN
          VAL(NE_G)=-FLUX(1,1)/K
          VAL(1)=TEMP_F(1)
        ELSE IF (TY(1).EQ.3) THEN
          VAL(NE_G)=TEMP_F(1)
          VAL(1)=-FLUX(1,2)/K
        ELSE IF ((TY(1).EQ.4).OR.(TY(1).EQ.5)) THEN
          VAL(1)=TEMP_F(1)
        END IF
      ELSE
        IF (TY(1).EQ.1) THEN
C
C         DU/DN=-FLUX/K
C
          VAL(1)=-FLUX(1,2)/K
        ELSE
          VAL(1)=TEMP_F(1)
        END IF
      END IF
C
C-------
C
      RETURN
      END
```

```
      SUBROUTINE MATRICES_H_G(XF,YF,H,G,TY,ABLA,NE_G)
C
C-------------------------------------------------------
C
C     ROUTINE CALCULANT LES MATRICES H ET G AVEC L'OB-
C     SERVATEUR SITUE SUR LA FRONTIERE.
C
C-------------------------------------------------------
C
C
      PARAMETER(NB_F = 32,MSOM = 50)
      REAL XF(0:NB_F+1),YF(0:NB_F+1),G(NB_F,2*NB_F),H(NB_F,NB_F+1)
     +,SINGSOMG1(MSOM),SINGSOMG2(MSOM),LXY(0:NB_F+1),CHANGE,
     +FACT(MSOM+1),KSI,PI,DIST1,DIST2,X3,Y3,LAMBDA,EULER,EPSILON
      INTEGER P,PLIM,NOF,I,J,K,TY(NB_F)
      LOGICAL ABLA(NB_F+1),FERMER
      COMMON/PROP/ALPHA/LONG/LXY/CONTOUR/FERMER
      COMMON/FRONT/NOF,NOF_M/INDICE/I,J/CONST/DENOM,PI/FACTOR/FACT
      DATA EULER/0.577215664901532E0/,EPSILON/1.0E-6/
      EXTERNAL R1F,FH1F,FH2F,FG1F,FG2F
      EXTERNAL R1S,R2S,FG1S,FG2S,FG3S,FG4S
C
      DO I=1,NOF_M
        DO J=1,NOF+1
          H(I,J)=0.0
        END DO
        DO J=1,2*NOF
          G(I,J)=0.0
        END DO
      END DO
C
C     CALCUL DE LA CONSTANTE QUI DEPEND DE L'EMPLACEMENT
C     DE L'OBSERVATEUR SITUE A CHAQUE NOEUD DE LA FRON-
C     TIERE.
C
      DO J=1,NOF_M
        IF (.NOT.ABLA(J)) THEN
          DIFF=YF(J)-YF(J-1)
          IF (ABS(DIFF).LT.EPSILON) DIFF=0.0
          TETA1=ACOS(DIFF/LXY(J-1))
          DIFF=YF(J+1)-YF(J)
          IF (ABS(DIFF).LT.EPSILON) DIFF=0.0
          TETA2=ACOS(DIFF/LXY(J))
          DIFF=XF(J-1)-XF(J)
          IF (ABS(DIFF).LT.EPSILON) DIFF=0.0
          IF (DIFF.LT.0.0) TETA1 = 2.0*PI - TETA1
          DIFF=XF(J)-XF(J+1)
          IF (ABS(DIFF).LT.EPSILON) DIFF=0.0
          IF (DIFF.LT.0.0) TETA2 = 2.0*PI - TETA2
          IF (ABS(TETA1-TETA2).LT.1.0E-4) TETA1=TETA2
          IF (TETA1.GT.TETA2) TETA2 = TETA2 + 2.0*PI
          H(J,J)=(PI + TETA1 -TETA2)/(2.0*PI)
```

```
          END IF
        END DO
C
        PLIM=10
        DO P=1,PLIM
          SINGSOMG1(P)=1.0/(FACT(P+1)*(2.0*P*P+P))
          SINGSOMG2(P)=1.0/(FACT(P+1)*P)
        END DO
        DO I=1,NOF_M
          L=1
          DO J=1,NOF
            K=J
            IF ((I.EQ.1) .AND. (K.EQ.NOF)) K = 0
C
C           TRAITEMENT DES SINGULARITES.
C
            IF (K.EQ.(I-1)) THEN
              TERM= (LXY(J)**2)/DENOM
              IF (TERM.LE.1.0) THEN
                SOM1=0.0
                SOM2=0.0
                DO P=1,PLIM
                  SOM1=SOM1 + SINGSOMG2(P)*((-TERM)**P)
                  SOM2=SOM2 + SINGSOMG1(P)*((-TERM)**P)
                END DO
                G(I,L) = G(I,L) + (1.0-EULER-LOG(TERM)-SOM1)*
     +                   LXY(J)/(8.0*PI)
                G(I,L+1) = G(I,L+1) + (3.0-EULER-LOG(TERM)-SOM2)*
     +                     LXY(J)/(8.0*PI)
              ELSE
                LAMBDA=1.0/SQRT(TERM)
                X3=LAMBDA*XF(J)+(1.0-LAMBDA)*XF(I)
                Y3=LAMBDA*YF(J)+(1.0-LAMBDA)*YF(I)
                DIST1=SQRT((X3-XF(I))*(X3-XF(I))+(Y3-YF(I))*
     +                (Y3-YF(I)))
                DIST2=LXY(J)-DIST1
                IF (DIST2.GT.EPSILON) THEN
                  CALL INTENUM1_TFS(FG1S,R1S,DIST1,DIST2,RES1)
                  CALL INTENUM1_TFS(FG2S,R1S,DIST1,DIST2,RES2)
                  G(I,L)=G(I,L)+RES1*DIST2/(16.0*PI)
                  G(I,L+1)=G(I,L+1)+RES2*DIST2*DIST2/(16.0*PI*LXY(J))
                END IF
                IF (DIST1.GT.EPSILON) THEN
                  SOM1=0.0
                  SOM2=0.0
                  DO P=1,PLIM
                    SOM1=SOM1+SINGSOMG2(P)*((-1.0)**P)
                    SOM2=SOM2+SINGSOMG1(P)*((2.0*P+1.0)*DIST2+
     +                   LXY(J))*((-1.0)**P)
                  END DO
                  G(I,L)=G(I,L)+(1.0-EULER-SOM1)*DIST1*
     +                   DIST1/(8.0*PI*LXY(J))
```

```
            G(I,L+1)=G(I,L+1)+(-EULER*(LXY(J)+DIST2)-DIST1+
     +                  4.0*LXY(J)-SOM2)*DIST1/(8.0*PI*LXY(J))
          END IF
        END IF
      ELSE IF (K.EQ.I) THEN
        TERM = (LXY(J)**2)/DENOM
        IF (TERM.LE.1.0) THEN
          SOM1=0.0
          SOM2=0.0
          DO P=1,PLIM
            SOM1=SOM1 + SINGSOMG1(P)*((-TERM)**P)
            SOM2=SOM2 + SINGSOMG2(P)*((-TERM)**P)
          END DO
          G(I,L) = G(I,L) + (3.0-EULER-LOG(TERM)-SOM1)*
     +             LXY(J)/(8.0*PI)
          G(I,L+1) = G(I,L+1) + (1.0-EULER-LOG(TERM)-SOM2)*
     +               LXY(J)/(8.0*PI)
        ELSE
          LAMBDA=1.0/SQRT(TERM)
          X3=LAMBDA*XF(J+1)+(1.0-LAMBDA)*XF(I)
          Y3=LAMBDA*YF(J+1)+(1.0-LAMBDA)*YF(I)
          DIST1=SQRT((X3-XF(I))*(X3-XF(I))+(Y3-YF(I))*
     +              (Y3-YF(I)))
          DIST2=LXY(J)-DIST1
          IF (DIST1.GT.EPSILON) THEN
            SOM1=0.0
            SOM2=0.0
            DO P=1,PLIM
              SOM1=SOM1+SINGSOMG1(P)*((2.0*P+1)*DIST2+
     +             LXY(J))*((-1.0)**P)
              SOM2=SOM2+SINGSOMG2(P)*((-1)**P)
            END DO
            G(I,L)=G(I,L)+(-EULER*(LXY(J)+DIST2)-DIST1+
     +             4.0*LXY(J)-SOM1)*DIST1/(8.0*PI*LXY(J))
            G(I,L+1)=G(I,L+1)+(1.0-EULER-SOM2)*DIST1*
     +               DIST1/(8.0*PI*LXY(J))
          END IF
          IF (DIST2.GT.EPSILON) THEN
            CALL INTENUM1_IFS(FG3S,R2S,DIST1,DIST2,RES1)
            CALL INTENUM1_IFS(FG4S,R2S,DIST1,DIST2,RES2)
            G(I,L)=G(I,L)+RES1*DIST2*DIST2/(16.0*PI*LXY(J))
            G(I,L+1)=G(I,L+1)+RES2*DIST2/(16.0*PI)
          END IF
        END IF
      ELSE
        IF ((.NOT.ABLA(J)).OR.(.NOT.ABLA(J+1))) THEN
          CALL PROD_SCAL(XF,YF,XF(J)-XF(I),YF(J)-YF(I),PRO)
          IF (ABS(PRO).GE.EPSILON) THEN
            IF (.NOT.ABLA(J)) THEN
              CALL INTENUM1_F(XF,YF,FH1F,RES1)
              H(I,J) = H(I,J) - PRO*RES1/(8.0*PI)
            END IF
```

```
              IF (.NOT.ABLA(J+1)) THEN
                CALL INTENUM1_F(XF,YF,FH2F,RES2)
                H(I,J+1) = H(I,J+1) - PRO*RES2/(8.0*PI)
              END IF
            END IF
          END IF
          CALL INTENUM1_TF(XF,YF,FG1F,R1F,RES1)
          CALL INTENUM1_TF(XF,YF,FG2F,R1F,RES2)
          G(I,L) = G(I,L) + RES1*LXY(J)/(16.0*PI)
          G(I,L+1) = G(I,L+1) + RES2*LXY(J)/(16.0*PI)
        END IF
        L=L+2
      END DO
      IF (.NOT.ABLA(1)) THEN
        H(I,1) = H(I,1) + H(I,NOF+1)
        H(I,NOF+1) = 0.0
      END IF
    END DO
    IF (.NOT.FERMER) CALL SYMETRIE_MAT_FRONT(H,G,NOF_M,NB_F)
C
C     REGROUPEMENT DES COLONNES OU LE NOEUD CORESPONDANT A UNE
C     CONDITION DE DIRICHLET
C
    IF (FERMER) THEN
      NE_G=2*NOF
      IF ((TY(NOF).EQ.4).OR.(TY(NOF).EQ.5)) THEN
        DO I=1,NOF
          G(I,NE_G-2)=G(I,NE_G-2)+G(I,NE_G-1)
          G(I,NE_G-1)=G(I,NE_G)
          G(I,NE_G)=0.0
        END DO
        NE_G=NE_G-1
      END IF
      IF ((TY(1).EQ.4).OR.(TY(1).EQ.5)) THEN
        DO I=1,NOF
          G(I,1)=G(I,NE_G)+G(I,1)
          G(I,NE_G)=0.0
        END DO
        NE_G=NE_G-1
      END IF
    ELSE
      NE_G=2*(NOF_M-1)
    END IF
    L=1
    DO J=2,NOF_M-1
      L=L+2
      IF ((TY(J).EQ.4).OR.(TY(J).EQ.5)) THEN
        N=L-1
        DO I=1,NOF_M
          G(I,N)=G(I,N)+G(I,L)
          DO K=L,NE_G-1
            G(I,K)=G(I,K+1)
```

```
            END DO
            G(I,NE_G)=0.0
          END DO
          NE_G=NE_G-1
          L=L-1
        END IF
      END DO
C
C     REGROUPEMENT DES VALEURS CONNUES A DROITE ET INCONNUES A GAUCHE
C
      L=1
      DO J=1,NOE_M
        IF (TY(J).EQ.2) THEN
          DO I=1,NOE_M
            CHANGE=G(I,L)
            G(I,L)=-H(I,J)
            H(I,J)=-CHANGE
          END DO
        ELSE IF (TY(J).EQ.3) THEN
          IF (J.EQ.1) THEN
            DO I=1,NOE_M
              CHANGE=G(I,NE_G)
              G(I,NE_G)=-H(I,1)
              H(I,1)=-CHANGE
            END DO
          ELSE
            DO I=1,NOE_M
              CHANGE=G(I,L-1)
              G(I,L-1)=-H(I,J)
              H(I,J)=-CHANGE
            END DO
          END IF
        ELSE IF (TY(J).EQ.5) THEN
          IF (J.GT.1) L=L-1
          DO I=1,NOE_M
            CHANGE=G(I,L)
            G(I,L)=-H(I,J)
            H(I,J)=-CHANGE
          END DO
        END IF
        L=L+2
      END DO
C
C----------
C
      RETURN
      END


      SUBROUTINE INTENUM1_IFS(F,RS,DIST1,DIST2,Z)
C
```

```
C-------------------------------------------------------
C
C       ROUTINE SERVANT A CALCULER NUMERIQUEMENT L'INTE-
C       GRALE P/R AU TEMPS ET P/R A L'ESPACE PAR LA ME-
C       THODE DE GAUSS-LAGUERRE ET PAR LA METHODE DE
C       GAUSS-LEGENDRE,ET CECI POUR TRAITER LA SINGULA-
C       RITE LORSQUE L'OBSERVATEUR EST SITUE SUR LA FRON-
C       TIERE.
C
C-------------------------------------------------------
C
        REAL Z,X,DIST1,DIST2,R,W1(12,12),PT(12,12),
     +       PZ(15,15),WZ(15,15),DENOM
        INTEGER K,L,ORD1,ORD2
        COMMON/INTEG1/ORD1,W1,PT/INTEG1_T/ORD2,WZ,PZ
        COMMON/CONST/DENOM
C
        Z=0.0
        DO K=1,ORD1
          X=0.0
          DO L=1,ORD2
            R=RS(PT(ORD1,K),DIST1,DIST2)
            X=X+WZ(ORD2,L)/(PZ(ORD2,L)+(R*R)/DENOM)
          END DO
          Z=Z+X*(W1(ORD1,K)*F(PT(ORD1,K),DIST1,DIST2))
        END DO
C
C------------
C
        RETURN
        END



        FUNCTION R1S(V,DIST1,DIST2)
C
C-------------------------------------------------------
C
C       FONCTION DONNANT LA DISTANCE ENTRE L'OBSER-
C       VATEUR ET LA SOURCE SITUES SUR LA FRONTIERE
C       ET CECI P/R AU DEUXIEME ELEMENT DE REFERENCE
C       UTILISE POUR TRAITER LA SINGULARITE POUR
C       J=I-1.
C
C-------------------------------------------------------
C
        PARAMETER(NB_F = 32)
        REAL LXY(0:NB_F+1),DIST1,DIST2,V
        INTEGER I
        COMMON/LONG/LXY/INDICE/I
C
        R1S=(LXY(I-1)+DIST1-V*DIST2)/2.0
```

```
C
C--------
C
        RETURN
        END


        FUNCTION R2S(V,DIST1,DIST2)
C
C-------------------------------------------------
C
C       FONCTION DONNANT LA DISTANCE ENTRE L'OBSER-
C       VATEUR ET LA SOURCE SITUES SUR LA FRONTIERE
C       ET CECI P/R AU DEUXIEME ELEMENT DE REFERENCE
C       UTILISE POUR TRAITER LA SINGULARITE POUR
C       J=I.
C
C-------------------------------------------------
C
        PARAMETER(NB_F = 32)
        REAL LXY(0:NB_F+1),DIST1,DIST2,V
        INTEGER I
        COMMON/LONG/LXY/INDICE/I
C
        R2S=(LXY(I)+DIST1+V*DIST2)/2.0
C
C--------
C
        RETURN
        END


        FUNCTION FG1S(V,DIST1,DIST2)
C
C-----------------------------------------------
C
C       FONCTION SERVANT A CALCULER LA
C       MATRICE G DANS LE CAS OU J=I-1.
C
C-----------------------------------------------
C
        PARAMETER(NB_F = 32)
        REAL V,LXY(0:NB_F+1),DENOM,DIST1,DIST2
        INTEGER I
        COMMON/CONST/DENOM/LONG/LXY/INDICE/I
C
        R=(LXY(I-1)+DIST1-V*DIST2)/2.0
        FG1S=(1.0-(V*DIST2-DIST1)/LXY(I-1))*EXP(-(R*R)/DENOM)
C
C---------
```

```
C
      RETURN
      END


      FUNCTION FG2S(V,DIST1,DIST2)
C
C-------------------------------------------------
C
C     FONCTION SERVANT A CALCULER LA
C     MATRICE G DANS LE CAS OU J=I-1.
C
C-------------------------------------------------
C
      PARAMETER(NB_F = 32)
      REAL V,LXY(0:NB_F+1),DENOM,DIST1,DIST2
      INTEGER I
      COMMON/CONST/DENOM/LONG/LXY/INDICE/I
C
      R=(LXY(I-1)+DIST1-V*DIST2)/2.0
      FG2S=(1.0+V)*EXP(-(R*R)/DENOM)
C
C---------
C
      RETURN
      END


      FUNCTION FG3S(V,DIST1,DIST2)
C
C-------------------------------------------------
C
C     FONCTION SERVANT A CALCULER LA
C     MATRICE G DANS LE CAS OU J=I.
C
C-------------------------------------------------
C
      PARAMETER(NB_F = 32)
      REAL V,LXY(0:NB_F+1),DENOM,DIST1,DIST2
      INTEGER I
      COMMON/CONST/DENOM/LONG/LXY/INDICE/I
C
      R=(LXY(I)+DIST1+V*DIST2)/2.0
      FG3S=(1.0-V)*EXP(-(R*R)/DENOM)
C
C---------
C
      RETURN
      END
```

```
      FUNCTION FG4S(V,DIST1,DIST2)
C
C-----------------------------------------------
C
C     FONCTION SERVANT A CALCULER LA
C     MATRICE G DANS LE CAS OU J=I.
C
C-----------------------------------------------
C
      PARAMETER(NB_F = 32)
      REAL V,LXY(0:NB_F+1),DENOM,DIST1,DIST2
      INTEGER I
      COMMON/CONST/DENOM/LONG/LXY/INDICE/I
C
      R=(LXY(I)+DIST1+V*DIST2)/2.0
      FG4S=(1.0+(DIST1+V*DIST2)/LXY(I))*EXP(-(R*R)/DENOM)
C
C---------
C
      RETURN
      END


      SUBROUTINE INTENUM1_F(XF,YF,F,Z)
C
C-------------------------------------------------------------
C
C     ROUTINE CALCULANT NUMERIQUEMENT PAR LA METHODE DE
C     GAUSS-LEGENDRE L'INTEGRALE A UNE DIMENSION,ET
C     CECI POUR L'OBSERVATEUR SITUE SUR LA FRONTIERE.
C
C-------------------------------------------------------------
C
      PARAMETER(NB_F = 32)
      REAL XF(0:NB_F+1),YF(0:NB_F+1)
      REAL W1(12,12),PT(12,12)
      COMMON/INTEG1/IORD,W1,PT/INDICE/I,J/CONST/DENOM
      EXTERNAL R1F
C
      Z=0.0
      DO K=1,IORD
        Z = Z + W1(IORD,K)*F(PT(IORD,K),R1F,XF,YF)
      END DO
C
C---------------
C
      RETURN
      END
```

```
      SUBROUTINE INTENUM1_TF(XF,YF,F,R1F,Z)
C
C-----------------------------------------------------------------
C
C       ROUTINE CALCULANT NUMERIQUEMENT L'INTEGRALE P/R AU TEMPS
C       ET P/R A L'ESPACE PAR LA METHODE DE GAUSS-LAGUERRE ET
C       PAR LA METHODE DE GAUSS-LEGENDRE,ET CECI POUR L'OBSER-
C       VATEUR SITUE SUR LA FRONTIERE.
C
C-----------------------------------------------------------------
C
      PARAMETER(NB_F = 32)
      REAL XF(0:NB_F+1),YF(0:NB_F+1)
      REAL W1(12,12),PT(12,12),PZ(15,15),WZ(15,15)
      INTEGER ORD1,ORD2
      COMMON/INTEG1/ORD1,W1,PT/INTEG1_T/ORD2,WZ,PZ
      COMMON/INDICE/I,J/CONST/DENOM
      EXTERNAL R1F
C
      Z=0.0
      DO K=1,ORD1
        X=0.0
        DO L=1,ORD2
          X = X + WZ(ORD2,L)/(PZ(ORD2,L) +
     +          R1F(PT(ORD1,K),XF,YF)/DENOM)
        END DO
        Z = Z + X*(W1(ORD1,K)*F(PT(ORD1,K),R1F,XF,YF))
      END DO
C
C-------------
C
      RETURN
      END



      FUNCTION R1F(KSI,XF,YF)
C
C-----------------------------------------------
C
C       FONCTION SERVANT A CALCULER LA DISTANCE
C       ENTRE L'OBSERVATEUR ET LA SOURCE SITUES
C       SUR LA FRONTIERE.
C
C-----------------------------------------------
C
      PARAMETER(NB_F = 32)
      REAL KSI,XF(0:NB_F+1),YF(0:NB_F+1)
      COMMON/INDICE/I,J
C
```

```
      R1F=((XF(J)*(1.0-KSI)/2.0+XF(J+1)*(1.0+KSI)/2.0)-XF(I))**2 +
     +     ((YF(J)*(1.0-KSI)/2.0+YF(J+1)*(1.0+KSI)/2.0)-YF(I))**2
C
C--------
C
      RETURN
      END


      FUNCTION FH1F(KSI,R1F,XF,YF)
C
C-----------------------------------------------------
C
C     FONCTION UTILISEE POUR CALCULER LA MATRICE H.
C
C-----------------------------------------------------
C
      PARAMETER(NB_F = 32)
      REAL KSI,DENOM,XF(0:NB_F+1),YF(0:NB_F+1)
      COMMON/CONST/DENOM
C
      FH1F=(1.0-KSI)*EXP(-R1F(KSI,XF,YF)/DENOM)/R1F(KSI,XF,YF)
C
C----------
C
      RETURN
      END


      FUNCTION FH2F(KSI,R1F,XF,YF)
C
C-----------------------------------------------------
C
C     FONCTION UTILISEE POUR CALCULER LA MATRICE H.
C
C-----------------------------------------------------
C
      PARAMETER(NB_F = 32)
      REAL KSI,DENOM,XF(0:NB_F+1),YF(0:NB_F+1)
      COMMON/CONST/DENOM
C
      FH2F=(1.0+KSI)*EXP(-R1F(KSI,XF,YF)/DENOM)/R1F(KSI,XF,YF)
C
C----------
C
      RETURN
      END
```

```
      FUNCTION FG1F(KSI,R1F,XF,YF)
C
C-----------------------------------------------------
C
C     FONCTION UTILISEE POUR CALCULER LA MATRICE G.
C
C-----------------------------------------------------
C
      PARAMETER(NB_F = 32)
      REAL KSI,DENOM,XF(0:NB_F+1),YF(0:NB_F+1)
      COMMON/CONST/DENOM
C
      FG1F=(1.0-KSI)*EXP(-R1F(KSI,XF,YF)/DENOM)
C
C----------
C
      RETURN
      END


      FUNCTION FG2F(KSI,R1F,XF,YF)
C
C-----------------------------------------------------
C
C     FONCTION UTILISEE POUR CALCULER LA MATRICE G.
C
C-----------------------------------------------------
C
      PARAMETER(NB_F = 32)
      REAL KSI,DENOM,XF(0:NB_F+1),YF(0:NB_F+1)
      COMMON/CONST/DENOM
C
      FG2F=(1.0+KSI)*EXP(-R1F(KSI,XF,YF)/DENOM)
C
C----------
C
      RETURN
      END


      SUBROUTINE MATRICES_HD_GD(XF,YF,CGT,HD,GD,TY,ABLA,NE_G)
C
C-----------------------------------------------------------
C
C     ROUTINE CALCULANT LES MATRICES HD ET GD
C     POUR L'OBSERVATEUR SITUE A L'INTERIEUR DU
C     DOMAINE
C
C-----------------------------------------------------------
C
```

```
      PARAMETER(NB_F = 32,MSOM = 50,NB_E = 532,NB_DIV = 10)
      REAL XF(0:NB_F+1),YF(0:NB_F+1),GD(NB_E,2*NB_F),HD(NB_E,NB_F+1),
     +  LXY(0:NB_F+1),CGI(NB_E,2),FACT(MSOM+1),KSI,PI,EULER,EPSILON,
     +  CHANGE,LAMBDA(NB_DIV)
      INTEGER P,PLIM,EL,TY(NB_F),NE_G,NB_COUCHE,N_ELE
      LOGICAL ABLA(NB_F+1),FERMER
      COMMON/PROP/ALPHA/DOMM/EL,NOI,N_ELE
      COMMON/FRONT/NOF,NOF_M/CONST/DENOM,PI/FACTOR/FACT/LONG/LXY
      COMMON/INDICE/I,J/MESH/LAMBDA,NB_COUCHE/CONTOUR/FERMER
      DATA EULER/0.577215664901532E0/,EPSILON/1.0E-6/
      EXTERNAL R1D,FH1D,FH2D,FG1D,FG2D
C
      IF (FERMER) THEN
        N1=NOF
      ELSE
        N1=NOF_M-1
      END IF
      N2=3*N1
      NB_TRI=NOF*3
C
      DO I=1,N_ELE
        DO J=1,NOF+1
          HD(I,J)=0.0
        END DO
        DO J=1,2*NOF
          GD(I,J)=0.0
        END DO
      END DO
C
      I1=0
      DO K=1,NB_COUCHE-1
        M=(K-1)*NB_TRI
        DO I=M+1,M+N2
          I1=I1+1
          L=1
          DO J=1,NOF
            IF ((.NOT.ABLA(J)).OR.(.NOT.ABLA(J+1))) THEN
              CALL PROD_SCAL(XF,YF,XF(J)-CGI(I,1),YF(J)-CGI(I,2),PRO)
              IF (ABS(PRO).GE.EPSILON) THEN
                IF (.NOT.ABLA(J)) THEN
                  CALL INTENUM1_D(XF,YF,CGI,FH1D,RES1)
                  HD(I1,J) = HD(I1,J) - PRO*RES1/(8.0*PI)
                END IF
                IF (.NOT.ABLA(J+1)) THEN
                  CALL INTENUM1_D(XF,YF,CGI,FH2D,RES2)
                  HD(I1,J+1) = HD(I1,J+1) - PRO*RES2/(8.0*PI)
                END IF
              END IF
            END IF
            CALL INTENUM1_TD(XF,YF,CGI,FG1D,R1D,RES1)
            CALL INTENUM1_TD(XF,YF,CGI,FG2D,R1D,RES2)
            GD(I1,L) = GD(I1,L) + RES1*LXY(J)/(16.0*PI)
```

```
            GD(I1,L+1) = GD(I1,L+1) + RES2*LXY(J)/(16.0*PI)
            L=L+2
          END DO
          IF (.NOT.ABLA(1)) THEN
            HD(I1,1) = HD(I1,1) + HD(I1,NOF+1)
            HD(I1,NOF+1) = 0.0
          END IF
        END DO
      END DO
      M=(NB_COUCHE-1)*NB_TRI
      DO I=M+1,M+N1
        I1=I1+1
        L=1
        DO J=1,NOF
          IF ((.NOT.ABLA(J)).OR.(.NOT.ABLA(J+1))) THEN
            CALL PROD_SCAL(XF,YF,XF(J)-CGT(I,1),YF(J)-CGT(I,2),PRO)
            IF (ABS(PRO).GE.EPSILON) THEN
              IF (.NOT.ABLA(J)) THEN
                CALL INTENUM1_D(XF,YF,CGT,FH1D,RES1)
                HD(I1,J) = HD(I1,J) - PRO*RES1/(8.0*PI)
              END IF
              IF (.NOT.ABLA(J+1)) THEN
                CALL INTENUM1_D(XF,YF,CGT,FH2D,RES2)
                HD(I1,J+1) = HD(I1,J+1) - PRO*RES2/(8.0*PI)
              END IF
            END IF
          END IF
          CALL INTENUM1_ID(XF,YF,CGT,FG1D,R1D,RES1)
          CALL INTENUM1_ID(XF,YF,CGT,FG2D,R1D,RES2)
          GD(I1,L) = GD(I1,L) + RES1*LXY(J)/(16.0*PI)
          GD(I1,L+1) = GD(I1,L+1) + RES2*LXY(J)/(16.0*PI)
          L=L+2
        END DO
        IF (.NOT.ABLA(1)) THEN
          HD(I1,1) = HD(I1,1) + HD(I1,NOF+1)
          HD(I1,NOF+1) = 0.0
        END IF
      END DO
      IF (.NOT.FERMER) CALL SYMETRIE_MAT_FRONT(HD,GD,N_ELE,NB_E)
C
C     REGROUPEMENT DES COLONNES OU LE NOEUD CORESPONDANT A UNE
C     CONDITION DE DIRICHLET
C
      IF (FERMER) THEN
        NE_G=2*NOF
        IF ((TY(NOF).EQ.4).OR.(TY(NOF).EQ.5)) THEN
          DO I=1,N_ELE
            GD(I,NE_G-2)=GD(I,NE_G-2)+GD(I,NE_G-1)
            GD(I,NE_G-1)=GD(I,NE_G)
            GD(I,NE_G)=0.0
          END DO
          NE_G=NE_G-1
```

```
          END IF
          IF ((TY(1).EQ.4).OR.(TY(1).EQ.5)) THEN
            DO I=1,N_ELE
              GD(I,1)=GD(I,NE_G)+GD(I,1)
              GD(I,NE_G)=0.0
            END DO
            NE_G=NE_G-1
          END IF
        ELSE
          NE_G=2*(NOE_M-1)
        END IF
        L=1
        DO J=2,NOE_M-1
          L=L+2
          IF ((TY(J).EQ.4).OR.(TY(J).EQ.5)) THEN
            N=L-1
            DO I=1,N_ELE
              GD(I,N)=GD(I,N)+GD(I,L)
              DO K=L,NE_G-1
                GD(I,K)=GD(I,K+1)
              END DO
              GD(I,NE_G)=0.0
            END DO
            NE_G=NE_G-1
            L=L-1
          END IF
        END DO
C
C       REGOUPEMENT DES VALEURS CONNUES A DROITE ET INCONNUES A GAUCHE
C
        L=1
        DO J=1,NOE_M
          IF (TY(J).EQ.2) THEN
            DO I=1,N_ELE
              CHANGE=GD(I,L)
              GD(I,L)=-HD(I,J)
              HD(I,J)=-CHANGE
            END DO
          ELSE IF (TY(J).EQ.3) THEN
            IF (J.EQ.1) THEN
              DO I=1,N_ELE
                CHANGE=GD(I,NE_G)
                GD(I,NE_G)=-HD(I,1)
                HD(I,1)=-CHANGE
              END DO
            ELSE
              DO I=1,N_ELE
                CHANGE=GD(I,L-1)
                GD(I,L-1)=-HD(I,J)
                HD(I,J)=-CHANGE
              END DO
            END IF
```

```
          ELSE IF (TY(J).EQ.5) THEN
            IF (J.GT.1) L=L-1
            DO I=1,N_ELE
              CHANGE=GD(I,L)
              GD(I,L)=-HD(I,J)
              HD(I,J)=-CHANGE
            END DO
          END IF
          L=L+2
        END DO
C
C----------
C
        RETURN
        END


        SUBROUTINE INTENUM1_D(XF,YF,CGT,F,Z)
C
C-----------------------------------------------------------------
C
C       ROUTINE CALCULANT NUMERIQUEMENT PAR LA METHODE DE
C       GAUSS-LEGENDRE L'INTEGRALE A UNE DIMENSION,ET
C       CECI POUR L'OBSERVATEUR SITUE DANS LE DOMAINE.
C
C-----------------------------------------------------------------
C
        PARAMETER(NB_F = 32,NB_E = 532)
        REAL XF(0:NB_F+1),YF(0:NB_F+1),CGT(NB_E,2)
        REAL W1(12,12),PT(12,12)
        COMMON/INTEG1/IORD,W1,PT/INDICE/I,J/CONST/DENOM
        EXTERNAL R1D
C
        Z=0.0
        DO K=1,IORD
          Z = Z + W1(IORD,K)*F(PT(IORD,K),R1D,XF,YF,CGT)
        END DO
C
C---------------
C
        RETURN
        END


        SUBROUTINE INTENUM1_TD(XF,YF,CGT,F,R1D,Z)
C
C-----------------------------------------------------------
C
C       ROUTINE CALCULANT L'INTEGRALE P/R AU TEMPS PAR
C       LA METHODE DE GAUSS-LAGUERRE ET P/R A L'ESPACE
```

```
C       PAR LA METHODE DE GAUSS-LEGENDRE ,ET CECI POUR L'OBSERVA-
C       TEUR SITUE DANS LE DOMAINE
C
C-------------------------------------------------------------
C
        PARAMETER(NB_F = 32,NB_E = 532)
        REAL XF(0:NB_F+1),YF(0:NB_F+1),CGT(NB_E,2)
        REAL W1(12,12),PT(12,12),WZ(15,15),PZ(15,15)
        INTEGER ORD1,ORD2
        COMMON/INTEG1/ORD1,W1,PT/INTEG1_T/ORD2,WZ,PZ
        COMMON/INDICE/I,J/CONST/DENOM
        EXTERNAL R1D
C
        Z=0.0
        DO K=1,ORD1
          X=0.0
          DO L=1,ORD2
            X = X + WZ(ORD2,L)/(PZ(ORD2,L) +
     +          R1D(PT(ORD1,K),XF,YF,CGT)/DENOM)
          END DO
          Z = Z + X*(W1(ORD1,K)*F(PT(ORD1,K),R1D,XF,YF,CGT))
        END DO
C
C---------
C
        RETURN
        END



        FUNCTION R1D(KSI,XF,YF,CGT)
C
C-----------------------------------------------
C
C       FONCTION SERVANT A CALCULER LA DISTANCE
C       ENTRE LA SOURCE ET L'OBSERVATEUR SITUE
C       DANS LE DOMAINE.
C
C-----------------------------------------------
C
        PARAMETER(NB_F = 32,NB_E = 532)
        REAL KSI,CGT(NB_E,2),XF(0:NB_F+1),YF(0:NB_F+1)
        COMMON/INDICE/I,J
C
        R1D=((XF(J)*(1.0-KSI)/2.0+XF(J+1)*(1.0+KSI)/2.0)-CGT(I,1))**2 +
     +      ((YF(J)*(1.0-KSI)/2.0+YF(J+1)*(1.0+KSI)/2.0)-CGT(I,2))**2
C
C--------
C
        RETURN
        END
```

```
      FUNCTION FH1D(KSI,R1D,XF,YF,CGT)
C
C-----------------------------------------------------
C
C     FONCTION UTILISEE POUR CALCULER LA MATRICE HD.
C
C-----------------------------------------------------
C
      PARAMETER(NB_F = 32,NB_E = 532)
      REAL KSI,CGT(NB_E,2),DENOM,XF(0:NB_F+1),YF(0:NB_F+1)
      COMMON/CONST/DENOM
C
      FH1D=(1.0-KSI)*EXP(-R1D(KSI,XF,YF,CGT)/DENOM)/R1D(KSI,XF,YF,CGT)
C
C----------
C
      RETURN
      END


      FUNCTION FH2D(KSI,R1D,XF,YF,CGT)
C
C-----------------------------------------------------
C
C     FONCTION UTILISEE POUR CALCULER LA MATRICE HD.
C
C-----------------------------------------------------
C
      PARAMETER(NB_F = 32,NB_E = 532)
      REAL KSI,CGT(NB_E,2),DENOM,XF(0:NB_F+1),YF(0:NB_F+1)
      COMMON/CONST/DENOM
C
      FH2D=(1.0+KSI)*EXP(-R1D(KSI,XF,YF,CGT)/DENOM)/R1D(KSI,XF,YF,CGT)
C
C----------
C
      RETURN
      END


      FUNCTION FG1D(KSI,R1D,XF,YF,CGT)
C
C-----------------------------------------------------
C
C     FONCTION UTILISEE POUR CALCULER LA MATRICE GD.
C
C-----------------------------------------------------
C
```

```
      PARAMETER(NB_F = 32,NB_E = 532)
      REAL KSI,CGT(NB_E,2),DENOM,XF(0:NB_F+1),YF(0:NB_F+1)
      COMMON/CONST/DENOM
C
      FG1D=(1.0-KSI)*EXP(-R1D(KSI,XF,YF,CGT)/DENOM)
C
C----------
C
      RETURN
      END


      FUNCTION FG2D(KSI,R1D,XF,YF,CGT)
C
C-------------------------------------------------
C
C     FONCTION UTILISEE POUR CALCULER LA MATRICE GD.
C
C-------------------------------------------------
C
      PARAMETER(NB_F = 32,NB_E = 532)
      REAL KSI,CGT(NB_E,2),DENOM,XF(0:NB_F+1),YF(0:NB_F+1)
      COMMON/CONST/DENOM
C
      FG2D=(1.0+KSI)*EXP(-R1D(KSI,XF,YF,CGT)/DENOM)
C
C----------
C
      RETURN
      END


      SUBROUTINE SYMETRIE_MAT_FRONT(H,G,NB_L,NB_LIG)
C
C-------------------------------------------------------------
C
C     ROUTINE SERVANT A COMPLETER LA MATRICE H, HD, G ET GD
C     POUR LES COTES DU DOMAINE QUI SONT SYMETRIQUES.
C
C-------------------------------------------------------------
C
      PARAMETER(NB_F = 32)
      REAL H(NB_LIG,NB_F+1),G(NB_LIG,2*NB_F)
      INTEGER I,I1,I2,J,L1,L2,K1,K2,K3,M,N,NOF,NOF_M
      LOGICAL QUART
      COMMON/FRONT/NOF,NOF_M/FORM/QUART
C
      M=2*NOF
      N=NOF/2+1
      DO I=1,NB_L
```

```
            G(I,1)=G(I,1)+G(I,M)
            G(I,2)=G(I,2)+G(I,M-1)
            J1=0
            L1=3
            L2=4
            K1=2
            K2=3
            DO J=2,N-1
              H(I,J)=H(I,J)+H(I,NOF-J1)
              G(I,L1)=G(I,L1)+G(I,M-K1)
              G(I,L2)=G(I,L2)+G(I,M-K2)
              H(I,NOF-J1)=0.0
              G(I,M-K1)=0.0
              G(I,M-K2)=0.0
              L1=L1+2
              L2=L2+2
              K1=K1+2
              K2=K2+2
              J1=J1+1
            END DO
          END DO
          IF (QUART) THEN
            DO I=1,NB_L
              J1=0
              K1=0
              K2=1
              L1=1
              L2=2
              DO J=1,NOF_M-1
                H(I,J)=H(I,J)+H(I,N-J1)
                G(I,L1)=G(I,L1)+G(I,NOF-K1)
                G(I,L2)=G(I,L2)+G(I,NOF-K2)
                H(I,N-J1)=0.0
                G(I,NOF-K1)=0.0
                G(I,NOF-K2)=0.0
                L1=L1+2
                L2=L2+2
                K1=K1+2
                K2=K2+2
                J1=J1+1
              END DO
            END DO
          END IF
C
C----------
C
          RETURN
          END


          SUBROUTINE PROD_SCAL(XF,YF,DX,DY,SORTIE)
```

```
C
C-----------------------------------------------
C
C       ROUTINE CALCULANT LE PRODUIT SCALAIRE
C       ENTRE LE VECTEUR ALLANT DE L'OBSERVATEUR
C       A LA SOURCE ET LE VECTEUR NORMAL DIRIGE
C       VERS L'EXTERIEUR DE LA SURFACE.
C
C-----------------------------------------------
C
        PARAMETER(NB_F = 32)
        REAL XF(0:NB_F+1),YF(0:NB_F+1)
        COMMON/INDICE/I,J
C
        SORTIE = (YF(J+1)-YF(J))*DX + (XF(J)-XF(J+1))*DY
C
C--------------
C
        RETURN
        END



        SUBROUTINE MATRICE_B(XI,YI,E,XF,YF,B,EL_AV)
C
C-------------------------------------------------------
C
C       ROUTINE CALCULANT LES ELEMENTS DE LA MATRICE B
C       AVEC L'OBSERVATEUR SITUE SUR LA FRONTIERE.
C
C-------------------------------------------------------
C
        PARAMETER(NB_F = 32,NB_E = 532,NB_D = 365)
        REAL PI
        REAL KSI,XI(NB_D),YI(NB_D),XF(0:NB_F+1),YF(0:NB_F+1),
     +       B(NB_F,NB_E)
        INTEGER E(NB_E,3),EL_AV
        LOGICAL FERMER
        COMMON/FRONT/NOF,NOF_M/INDICE/I,J/CONST/DENOM,PI
        COMMON/CONTOUR/FERMER
C
        DO I=1,NOF_M
          DO J=1,EL_AV
            CALL INTENUM2P(XI,YI,XF(I),YF(I),E,B(I,J))
          END DO
        END DO
        IF (.NOT.FERMER) CALL SYMETRIE_MAT_DOMM(B,EL_AV,NOF_M,NB_F)
C
C----------------
C
        RETURN
        END
```

```
      SUBROUTINE MATRICE_BD(XI,YI,E,CGT,BD,EL_AV)
C
C-------------------------------------------------------
C
C     ROUTINE CALCULANT LA MATRICE BD POUR L'OBSERVATEUR
C     SITUE DANS LE DOMAINE.
C
C-------------------------------------------------------
C
      PARAMETER(NB_E = 532,NB_D = 365,NB_DIV = 10)
      REAL KSI,PI,XI(NB_D),YI(NB_D),CGT(NB_E,2),BD(NB_E,NB_E),
     + LAMBDA(NB_DIV)
      INTEGER E(NB_E,3),EL,EL_AV,NOF,NOF_M,N1,I,J,K,N2,M,NB_COUCHE
      LOGICAL FERMER
      COMMON/DOMM/EL,NOI,N_ELE/CONST/DENOM,PI/INDICE/I,J
      COMMON/FRONT/NOF,NOF_M/CONTOUR/FERMER/MESH/LAMBDA,NB_COUCHE
C
      IF (FERMER) THEN
        N1=NOF
      ELSE
        N1=NOF_M-1
      END IF
      N2=N1*3
      NB_TRI=3*NOF
      I1=0
      DO K=1,NB_COUCHE-1
        M=(K-1)*NB_TRI
        DO I=M+1,M+N2
          I1=I1+1
          DO J=1,EL_AV
            CALL INTENUM2P(XI,YI,CGT(I,1),CGT(I,2),E,BD(I1,J))
          END DO
        END DO
      END DO
      M=(NB_COUCHE-1)*NB_TRI
      DO I=M+1,M+N1
        I1=I1+1
        DO J=1,EL_AV
          CALL INTENUM2P(XI,YI,CGT(I,1),CGT(I,2),E,BD(I1,J))
        END DO
      END DO
      IF (.NOT.FERMER) CALL SYMETRIE_MAT_DOMM(BD,EL_AV,N_ELE,NB_E)
C
C--------------
C
      RETURN
      END
```

```
      SUBROUTINE SYMETRIE_MAT_DOMM(B,EL_AV,NB_L,NB_LIG)
C
C-----------------------------------------------------------------
C
C     ROUTINE SERVANT A REGROUPER LES TERMES DE LA MATRICE
C     'B' ET 'BD' QUI CORRESPONDENT A LA MEME TEMPERATURE A
C     UN POINT DONNE DU DOMMAINE PAR SYMETRIE.
C
C-----------------------------------------------------------------
C
      PARAMETER(NB_E = 532)
      REAL B(NB_LIG,NB_E)
      INTEGER EL_AV,NOF_M,NB_COUCHE_AV
      LOGICAL QUART
      COMMON/FORM/QUART/MESH_AV/NB_COUCHE_AV/FRONT/NOF,NOF_M
C
      N=3*(NOF_M-1)
      NB_TRI=3*NOF
      IF (QUART) THEN
        N1=4*N
        N2=2*N
        N3=2*(NOF_M-1)
      ELSE
        N1=2*N
        N2=N
        N3=NOF_M-1
      END IF
      DO I=1,NB_L
        J=0
        DO K=1,NB_COUCHE_AV-1
          M1=(K-1)*NB_TRI
          M2=M1+N1
          K3=0
          DO K2=1+M1,N2+M1
            J=J+1
            B(I,J)=B(I,K2)+B(I,M2-K3)
            K3=K3+1
          END DO
        END DO
        M1=(NB_COUCHE_AV-1)*NB_TRI
        M2=EL_AV
        K3=0
        DO K2=1+M1,N3+M1
          J=J+1
          B(I,J)=B(I,K2)+B(I,M2-K3)
          K3=K3+1
        END DO
      END DO
      IF (QUART) THEN
        DO I=1,NB_L
          J=0
```

```
            DO K=1,NB_COUCHE_AV-1
              M1=(K-1)*NB_TRI/2
              M2=M1+N2
              K3=0
              DO K2=1+M1,N+M1
                J=J+1
                B(I,J)=B(I,K2)+B(I,M2-K3)
                K3=K3+1
              END DO
            END DO
            M1=(NB_COUCHE_AV-1)*NB_TRI/2
            M2=M1+N3
            K3=0
            DO K2=1+M1,(NOE_M-1)+M1
              J=J+1
              B(I,J)=B(I,K2)+B(I,M2-K3)
              K3=K3+1
            END DO
          END DO
        END IF
C
C-------
C
        RETURN
        END



        SUBROUTINE INTENUM2P(XI,YI,PX,PY,E,Z)
C
C-------------------------------------------------------
C
C       ROUTINE CALCULANT EN COORDONNEES POLAIRES L'IN-
C       TEGRALE SUR LE DOMAINE POUR L'OBSERVATEUR
C       SITUE SUR LA FRONTIERE.
C
C-------------------------------------------------------
C
        PARAMETER(NB_E = 532,NB_D = 365)
        REAL XI(NB_D),YI(NB_D),PX,PY,
     +  R1,R2,R3,PHI(3),Z,X,Y,DENOM,W1(12,12),PT(12,12),PI,MAX,DIFF(3)
        INTEGER IORD,CAS,I1,N2,I,J,P(3),E(NB_E,3),
     +          K(3),L(3)
        COMMON/INTEG1/IORD,W1,PT/CONST/DENOM,PI/INDICE/I,J
        COMMON/COORD_TRI/K,L
        DATA EPSILON/1.0E-6/
        EXTERNAL R2P
C
        CALL POSITION_POINT(XI,YI,E,PX,PY,INSIDE)
        CAS=0
        IF (INSIDE.EQ.1) THEN
          DO I1=1,3
```

```
        P(I1)=I1
        X=XI(E(J,I1))-PX
        Y=YI(E(J,I1))-PY
        IF ((ABS(X).LT.EPSILON).AND.(ABS(Y).LT.EPSILON)) THEN
          N2=I1
          PHI(N2)=0.0
          CAS=1
        ELSE
          IF (ABS(X).LT.EPSILON) THEN
            X=0.0
            R1=ABS(Y)
          ELSE IF (ABS(Y).LT.EPSILON) THEN
            Y=0.0
            R1=ABS(X)
          ELSE
            R1=SQRT((X*X)+(Y*Y))
          END IF
          PHI(I1)=ACOS(X/R1)
          IF (Y.LT.0.0) PHI(I1)=2.0*PI-PHI(I1)
        END IF
      END DO
      IF (CAS.NE.1) THEN
        DO I1=1,3
          X=XI(E(J,K(I1)))-PX
          Y=YI(E(J,K(I1)))-PY
          R1=SQRT((X*X)+(Y*Y))
          X=XI(E(J,L(I1)))-PX
          Y=YI(E(J,L(I1)))-PY
          R2=SQRT((X*X)+(Y*Y))
          X=XI(E(J,K(I1)))-XI(E(J,L(I1)))
          Y=YI(E(J,K(I1)))-YI(E(J,L(I1)))
          R3=SQRT((X*X)+(Y*Y))
          IF (ABS((R1+R2)-R3).LT.EPSILON) THEN
            CAS=2
            N2=I1
          END IF
        END DO
        IF (CAS.EQ.2) THEN
          CALL POINT_SUR_COTE(PHI,P,N2,E)
          X=0.0
          IF (ABS(PHI(2)-PHI(1)).GT.PI) PHI(1)=PHI(1)-2.0*PI
          DO I1=1,IORD
            X=X+W1(IORD,I1)*EXP(-(R2P(PT(IORD,I1),XI,YI,P,PX,PY,
     +                                PHI(1),PHI(2),3)**2)/DENOM)
          END DO
          X=X*(PHI(2)-PHI(1))
          Z=0.0
          IF (ABS(PHI(3)-PHI(2)).GT.PI) PHI(2)=PHI(2)-2.0*PI
          DO I1=1,IORD
            Z=Z+W1(IORD,I1)*EXP(-(R2P(PT(IORD,I1),XI,YI,P,PX,PY,
     +                                PHI(2),PHI(3),1)**2)/DENOM)
          END DO
```

```
          END DO
          CALL POINT_OUTSIDE_SOMMET(PHI,P,N2,E)
          X=0.0
          IF (ABS(PHI(2)-PHI(1)).GE.1.0E-4) THEN
            IF (ABS(PHI(2)-PHI(1)).GT.PI) PHI(1)=PHI(1)-2.0*PI
            DO I1=1,IORD
              X=X+W1(IORD,I1)*(EXP(-(R2P(PT(IORD,I1),XI,YI,P,PX,PY,
     +                                 PHI(1),PHI(2),3)**2)/DENOM)-
     +                         EXP(-(R2P(PT(IORD,I1),XI,YI,P,PX,PY,
     +                                 PHI(1),PHI(2),2)**2)/DENOM))
            END DO
          END IF
          X=ABS(X)*(PHI(2)-PHI(1))
          Z=0.0
          IF (ABS(PHI(3)-PHI(2)).GE.1.0E-4) THEN
            IF (ABS(PHI(3)-PHI(2)).GT.PI) PHI(2)=PHI(2)-2.0*PI
            DO I1=1,IORD
              Z=Z+W1(IORD,I1)*(EXP(-(R2P(PT(IORD,I1),XI,YI,P,PX,PY,
     +                                 PHI(2),PHI(3),1)**2)/DENOM)-
     +                         EXP(-(R2P(PT(IORD,I1),XI,YI,P,PX,PY,
     +                                 PHI(2),PHI(3),2)**2)/DENOM))
            END DO
          END IF
          Z=(X+ABS(Z)*(PHI(3)-PHI(2)))*0.25/PI
        END IF
C
C-------------
C
        RETURN
        END



        SUBROUTINE POSITION_POINT(XI,YI,E,PX,PY,INSIDE)
C
C-------------------------------------------------------
C
C       ROUTINE SERVANT A DETERMINER SI L'OBSERVATEUR
C       SITUE SUR LA FRONTIERE EST SITUE A L'INTERIEUR D'UN
C       TRIANGLE CONSTITUANT LE DOMAINE DU PAS DE TEMPS PRECE-
C       DANT.
C
C-------------------------------------------------------
C
        PARAMETER(NB_E = 532,NB_D = 365)
        REAL XI(NB_D),YI(NB_D),PX,PY,Y,
     +       Y1,Y2,X
        INTEGER E(NB_E,3),I,J,INSIDE
        COMMON/INDICE/I,J
        DATA EPSILON/1.0E-6/
C
        X=XI(E(J,2))-XI(E(J,1))
```

```
          Z=0.5-(X+(PHI(3)-PHI(2))*Z)*0.25/PI
        ELSE
          CALL POINT_INSIDE(PHI,P,E)
          X=0.0
          Y=0.0
          Z=0.0
          DO I1=1,IORD
            X=X+W1(IORD,I1)*EXP(-(R2P(PT(IORD,I1),XI,YI,P,PX,PY,
+                                   PHI(1),PHI(2),3)**2)/DENOM)
            Y=Y+W1(IORD,I1)*EXP(-(R2P(PT(IORD,I1),XI,YI,P,PX,PY,
+                                   PHI(2),PHI(3),1)**2)/DENOM)
            Z=Z+W1(IORD,I1)*EXP(-(R2P(PT(IORD,I1),XI,YI,P,PX,PY,
+                                 PHI(3)-2.0*PI,PHI(1),2)**2)/DENOM)
          END DO
          Z = 1.0-((PHI(2)-PHI(1))*X+(PHI(3)-PHI(2))*Y+
+                 (PHI(1)-PHI(3)+2.0*PI)*Z)*0.25/PI
        END IF
      ELSE
        CALL POINT_OUTSIDE_SOMMET(PHI,P,N2,E)
        IF (ABS(PHI(3)-PHI(1)).GT.PI) PHI(1)=PHI(1)-2.0*PI
        Z=0.0
        DO I1=1,IORD
              Z=Z+W1(IORD,I1)*EXP(-(R2P(PT(IORD,I1),XI,YI,P,PX,PY,
+                                     PHI(1),PHI(3),2)**2)/DENOM)
        END DO
        Z=(1.0-Z/2.0)*(PHI(3)-PHI(1))/(2.0*PI)
      END IF
    ELSE
      DO I1=1,3
        P(I1)=I1
        X=XI(E(J,I1))-PX
        Y=YI(E(J,I1))-PY
        IF (ABS(X).LT.EPSILON) THEN
          X=0.0
          R1=ABS(Y)
        ELSE IF (ABS(Y).LT.EPSILON) THEN
          Y=0.0
          R1=ABS(X)
        ELSE
          R1=SQRT((X*X)+(Y*Y))
        END IF
        PHI(I1)=ACOS(X/R1)
        IF (Y.LT.0.0) PHI(I1)=2.0*PI-PHI(I1)
      END DO
      MAX=0.0
      DO I1=1,3
        DIFF(I1)=ABS(PHI(K(I1))-PHI(L(I1)))
        IF (DIFF(I1).GT.PI) DIFF(I1)=2.0*PI-DIFF(I1)
        IF (DIFF(I1).GT.MAX) THEN
          MAX=DIFF(I1)
          N2=I1
        END IF
```

```
        IF (ABS(X).LT.EPSILON) THEN
          CALL POS_POINT(XI(E(J,1)),XI(E(J,3)),XI(E(J,1)),PX,INSIDE)
        ELSE
          Y=(YI(E(J,2))-YI(E(J,1)))/X
          Y1=(PX-XI(E(J,1)))*Y + YI(E(J,1))
          Y2=(XI(E(J,3))-XI(E(J,1)))*Y + YI(E(J,1))
          CALL POS_POINT(Y1,PY,Y2,YI(E(J,3)),INSIDE)
        END IF
        IF (INSIDE.EQ.1) THEN
          X=XI(E(J,3))-XI(E(J,1))
          IF (ABS(X).LT.EPSILON) THEN
            CALL POS_POINT(XI(E(J,1)),XI(E(J,2)),XI(E(J,1)),PX,INSIDE)
          ELSE
            Y=(YI(E(J,3))-YI(E(J,1)))/X
            Y1=(PX-XI(E(J,1)))*Y + YI(E(J,1))
            Y2=(XI(E(J,2))-XI(E(J,1)))*Y + YI(E(J,1))
            CALL POS_POINT(Y1,PY,Y2,YI(E(J,2)),INSIDE)
          END IF
        END IF
        IF (INSIDE.EQ.1) THEN
          X=XI(E(J,3))-XI(E(J,2))
          IF (ABS(X).LT.EPSILON) THEN
            CALL POS_POINT(XI(E(J,2)),XI(E(J,1)),XI(E(J,2)),PX,INSIDE)
          ELSE
            Y=(YI(E(J,3))-YI(E(J,2)))/X
            Y1=(PX-XI(E(J,2)))*Y + YI(E(J,2))
            Y2=(XI(E(J,1))-XI(E(J,2)))*Y + YI(E(J,2))
            CALL POS_POINT(Y1,PY,Y2,YI(E(J,1)),INSIDE)
          END IF
        END IF
C
C---------
C
        RETURN
        END


        SUBROUTINE POS_POINT(X1,X2,X3,X4,INSIDE)
C
C-------------------------------------------------------
C
C       ROUTINE SERVANT A DETERMINER SI DEUX POINTS SE TROUVE
C       DE PART ET D'AUTRE D'UNE DROITE.
C
C-------------------------------------------------------
C
        REAL X1,X2,X3,X4,DIFF1,DIFF2
        INTEGER INSIDE
        DATA EPSILON/1.0E-6/
C
        INSIDE=1
```

```
        DIFF1=X2-X1
        IF (ABS(DIFF1).LT.EPSILON) DIFF1=0
        DIFF2=X4-X3
        IF (ABS(DIFF2).LT.EPSILON) DIFF2=0
        IF ((DIFF1*DIFF2).LT.0.0) INSIDE=0
C
C-------
C
        RETURN
        END


        SUBROUTINE POINT_OUTSIDE_SOMMET(PHI,P,N2,E)
C
C-----------------------------------------------------------------
C
C       ROUTINE SERVANT A METTRE EN ORDRE CROISSANT LES DEUX
C       AUTRES ANGLES CORRESPONDANT AUX SOMMETS AUTRES QUE
C       'N2' AVEC L'OBSERVATEUR SITUE SUR UN SOMMET (N2) OU
C       A L'EXTERIEUR DU TRIANGLE.DANS LE DERNIER CAS N2 REPRE-
C       SENTE LE SOMMET QUI SEPARE L'INTEGRATION EN DEUX PARTIES
C
C-----------------------------------------------------------------
C
        PARAMETER(NB_E = 532)
        REAL PHI1,PHI2,PHI3,PHI(3)
        INTEGER I,K(3),L(3),P(3),N1,N2,N3,E(NB_E,3),J,ITEMP
        COMMON/COORD_TRI/K,L/INDICE/I,J/CONST/DENOM,PI
C
        N1=K(N2)
        N3=L(N2)
        IF (N1.GT.N3) THEN
          ITEMP=N1
          N1=N3
          N3=ITEMP
        END IF
        P(2)=E(J,N2)
        PHI2=PHI(N2)
        IF (ABS(PHI(N1)-PHI(N3)).GT.PI) THEN
          IF (PHI(N1).GT.PI) THEN
            P(1)=E(J,N1)
            P(3)=E(J,N3)
            PHI1=PHI(N1)
            PHI3=PHI(N3)
          ELSE
            P(1)=E(J,N3)
            P(3)=E(J,N1)
            PHI1=PHI(N3)
            PHI3=PHI(N1)
          END IF
        ELSE
```

```
          IF (PHI(N3).LT.PHI(N1)) THEN
            PHI3=PHI(N1)
            PHI1=PHI(N3)
            P(1)=E(J,N3)
            P(3)=E(J,N1)
          ELSE
            PHI3=PHI(N3)
            PHI1=PHI(N1)
            P(1)=E(J,N1)
            P(3)=E(J,N3)
          END IF
        END IF
        PHI(1)=PHI1
        PHI(2)=PHI2
        PHI(3)=PHI3
C
C---------
C
        RETURN
        END



        SUBROUTINE POINT_SUR_COTE(PHI,P,N2,E)
C
C-------------------------------------------------------
C
C       ROUTINE SERVANT A DETERMINER L'ORDRE DES DEUX
C       AUTRES ANGLES POUR POUVOIR BIEN EFFECTUER
C       L'INTEGRATION EN COORDONNEES POLAIRES,ET CECI
C       DANS LE CAS OU L'OBSERVATEUR EST SITUE SUR UN DES
C       COTES DU TRIANGLE A L'EXCLUSION DES SOMMETS.
C
C-------------------------------------------------------
C
        PARAMETER(NB_E = 532)
        REAL PHI(3),PHI1,PHI2,PHI3
        INTEGER E(NB_E,3),I,J,ITEMP,N1,N2,N3,P(3),K(3),L(3)
        COMMON/COORD_TRI/K,L/INDICE/I,J
C
        N1=K(N2)
        N3=L(N2)
        IF (N1.GT.N3) THEN
          ITEMP=N1
          N1=N3
          N3=ITEMP
        END IF
        P(2)=E(J,N2)
        PHI2=PHI(N2)
        IF (PHI(N1).LT.PHI(N3)) THEN
          PHI1=PHI(N1)
          PHI3=PHI(N3)
```

```
          P(1)=E(J,N1)
          P(3)=E(J,N3)
        ELSE
          PHI1=PHI(N3)
          PHI3=PHI(N1)
          P(1)=E(J,N3)
          P(3)=E(J,N1)
        END IF
        IF ((PHI2.LT.PHI1).OR.(PHI2.GT.PHI3)) THEN
          PHI(1)=PHI3
          PHI(3)=PHI1
          ITEMP=P(1)
          P(1)=P(3)
          P(3)=ITEMP
        ELSE
          PHI(1)=PHI1
          PHI(3)=PHI3
        END IF
        PHI(2)=PHI2
C
C--------
C
        RETURN
        END


        SUBROUTINE POINT_INSIDE(PHI,P,E)
C
C-------------------------------------------------------
C
C       ROUTINE SERVANT A METTRE EN ORDRE CROISSANT LES
C       ANGLES CORRESPONDANT AUX SOMMETS DU TRIANGLES
C       AVEC L'OBSERVATEUR SITUE A L'INTERIEUR DU TRI-
C       ANGLE.
C
C-------------------------------------------------------
C
        PARAMETER(NB_E = 532)
        REAL PHI(3),TEMP
        INTEGER E(NB_E,3),I,J,N,I1,I2,I3,P(3)
        COMMON/INDICE/I,J
C
        DO I1=1,2
          I2=I1
          N=P(I1)
          TEMP=PHI(I1)
          DO I3=I1+1,3
            IF (PHI(I3).LT.TEMP) THEN
              I2=I3
              TEMP=PHI(I3)
              N=P(I3)
```

```
            END IF
          END DO
          PHI(I2)=PHI(I1)
          PHI(I1)=TEMP
          P(I2)=P(I1)
          P(I1)=N
        END DO
        DO I1=1,3
          P(I1)=E(J,P(I1))
        END DO
C
C--------
C
        RETURN
        END



        FUNCTION R2P(ZETA,XI,YI,P,PX,PY,PHI1,PHI2,N)
C
C-------------------------------------------------------
C
C       FONCTION SERVANT A CALCULER LA DISTANCE ENTRE
C       UN COTE D'UN TRIANGLE ET L'OBSERVATEUR SITUE
C       SUR LA FRONTIERE.
C
C-------------------------------------------------------------
C
        PARAMETER(NB_E = 532,NB_D = 365)
        REAL ZETA,XI(NB_D),YI(NB_D),PX,PY,A,B,
     +       PHI1,PHI2,TETA
        INTEGER N,P(3),K(3),L(3),I,J,IND1,IND2
        COMMON/COORD_TRI/K,L
        DATA EPSILON/1.0E-6/
C
        IND1=P(K(N))
        IND2=P(L(N))
        A=XI(IND2)-XI(IND1)
        B=YI(IND1)-YI(IND2)
        TETA=((PHI2-PHI1)*ZETA+(PHI2+PHI1))*0.5
        R2P = -(XI(IND1)*YI(IND2)-XI(IND2)*YI(IND1)+B*PX+A*PY)/
     +         (B*COS(TETA)+A*SIN(TETA))
C
C--------
C
        RETURN
        END



        FUNCTION SIMUL(NB, N, A, X, INDIC )
C
```

```
C-----------------------------------------------------------------------
C
C       INDIC  :LORSQUE INDIC EST NEGATIF, SIMUL CALCUL L'INVERSE
C               DE LA MATRICE A QUI EST N PAR N.LORSQUE INDIC EST
C               NUL, SIMUL SOLUTIONNE LE SYSTEME D'EQUATIONS LI-
C               NEAIRES ET INVERSE LA MATRICES A.LE RESULTAT EST
C               DANS LE VECTEUR X.LORSQUE INDIC EST POSITIF, SIMUL
C               SOLUTIONNE LE SYSTEME D'EQUATIONS LINEAIRES SANS
C               INVERSER LA MATRICE.
C
C       SIMUL  :RETOURNE LA VALEUR DU DETERMINANT DE LA MATRICE.
C
C       NB     :DIMENSION MAXIMALE DE LA MATRICE A.
C
C
C       POUR PLUS D'EXPLICATIONS VOIR LE LIVRE 'APPLIED NUMERICAL
C       METHODS' PAR B.CARNAHAN,H.A. LUTHER ET JAMES O.WILKES
C       AUX PAGES 282 A 296.
C
C-----------------------------------------------------------------------
C
        PARAMETER(NB_E = 40)
        REAL A(NB,NB+1),SIMUL,Y(NB_E),X(NB)
        INTEGER IROW(NB_E),JCOL(NB_E),JORD(NB_E)
        DATA EPSILON/1.0E-6/
C
        MAX=N
        IF (INDIC.GE.0) MAX=N+1
C
C       .....BEGIN ELIMINATION PROCEDURE.....
5        DETER=1.
        DO 18 K=1,N
        KM1=K-1
C
C       ......SEARCH FOR THE PIVOT ELEMENT ....
        PIVOT=0.
        DO 11 I=1,N
        DO 11 J=1,N
C       .......SCAN IROW AND JCOL ARRAYS FOR INVALID PIVOT SUBSCRIPTS....
        IF (K.EQ.1) GO TO 9
        DO 8 ISCAN=1,KM1
        DO 8 JSCAN=1,KM1
        IF (I.EQ.IROW(ISCAN)) GO TO 11
        IF (J.EQ.JCOL(JSCAN)) GO TO 11
8        CONTINUE
9        IF (ABS(A(I,J)).LE.ABS(PIVOT)) GO TO 11
        PIVOT=A(I,J)
        IROW(K)=I
        JCOL(K)=J
11        CONTINUE
C
C       ......INSURE THAT SELECTED PIVOT IS LARGER THAN EPSILON ......
```

```
      IF (ABS(PIVOT).GT.EPSILON) GO TO 13
      SIMUL=0.
      RETURN
C
C     .....UPDATE THE DETERMINANT VALUE .....
13      IROWK=IROW(K)
      JCOLK=JCOL(K)
      DETER=DETER*PIVOT
C
C     ......NORMALIZE PIVOT ROW ELEMENTS.....
      DO 14 J=1,MAX
14      A(IROWK,J)=A(IROWK,J)/PIVOT
C
C     ......CARRY OUT ELIMINATION AND DEVELOP INVERSE ....
      A(IROWK,JCOLK)=1./PIVOT
      DO 18 I=1,N
      AIJCK=A(I,JCOLK)
      IF (I.EQ.IROWK) GO TO 18
      A(I,JCOLK)=-AIJCK/PIVOT
      DO 17 J=1,MAX
17      IF (J.NE.JCOLK) A(I,J)=A(I,J)-AIJCK*A(IROWK,J)
18      CONTINUE
C
C     ...ORDER SOLUTION VALUES (IF ANY) AND CREATE JORD ARRAY....
      DO 20 I=1,N
      IROWI=IROW(I)
      JCOLI=JCOL(I)
      JORD(IROWI)=JCOLI
20      IF (INDIC.GE.0) X(JCOLI)=A(IROWI,MAX)
C
C     ....ADJUST SIGN OF DETERMINANT.....
      INTCH=0
      NM1=N-1
      DO 22 I=1,NM1
      IP1=I+1
      DO 22 J=IP1,N
      IF (JORD(J).GE.JORD(I)) GO TO 22
      JTEMP=JORD(J)
      JORD(J)=JORD(I)
      JORD(I)=JTEMP
      INTCH=INTCH+1
22      CONTINUE
      IF (INTCH/2*2.NE.INTCH) DETER=-DETER
C
C     .....IF INDIC IS POSITIVE RETURN WITH RESULTS.....
      IF (INDIC.LE.0) GO TO 26
      SIMUL=DETER
      RETURN
C
C     IF INDIC IS NEGATIVE OR ZERO,UNSCRAMBLE THE INVERSE
C     FIRST BYE ROWS.......
26      DO 28 J=1,N
```

```
        DO 27 I=1,N
        IROWI=IROW(I)
        JCOLI=JCOL(I)
27         Y(JCOLI)=A(IROWI,J)
        DO 28 I=1,N
28        A(I,J)=Y(I)
C       .....THEN BYE COLUMNS.....
        DO 30 I=1,N
        DO 29 J=1,N
        IROWJ=IROW(J)
        JCOLJ=JCOL(J)
29        Y(IROWJ)=A(I,JCOLJ)
        DO 30 J=1,N
30        A(I,J)=Y(J)
C
C       ....RETURN FOR INDIC NEGATIVE OR ZERO....
        SIMUL=DETER
C
C-------
C
        RETURN
        END


        SUBROUTINE PRODUIT_MAT(A,B,C,NB_L,NB_C,NB_LIG,NB_COL)
C
C-------------------------------------------------------
C
C       ROUTINE CALCULANT LE PRODUIT MATRICIEL D'UNE
C       MATRICE NB_L X NB_C AVEC UNE MATRICE NB_C X 1
C
C-------------------------------------------------------
C
        REAL A(NB_LIG,NB_COL),B(NB_COL),C(NB_COL)
        COMMON/FRONT/NOE,NOE_M
C
        DO I=1,NB_L
          C(I)=0.0
          DO J=1,NB_C
            C(I)=C(I) + A(I,J)*B(J)
          END DO
        END DO
C
C-------------
C
        RETURN
        END


        SUBROUTINE VERI_TEMP_FUSION(TY,ABLA,FLUX,VAL,VAL_INC,G,H,GD,
```

```
     +                                HD,NE_G)
C
C-----------------------------------------------------------------
C
C       ROUTINE SERVANT A VERIFIER SI UN OU PLUSIEURS NOEUDS ONT
C       ATTEINT LA TEMPERATURE DE FUSION ET A FAIRE LE REARRANGE-
C       MENT DES MATRICES.
C
C-----------------------------------------------------------------
C
        PARAMETER(NB_F = 32,NB_E = 532)
        REAL FLUX(NB_F,2),VAL(2*NB_F),G(NB_E,2*NB_F),H(NB_F,NB_F+1),
     +       GD(NB_E,2*NB_F),HD(NB_E,NB_F+1),VAL_INC(NB_F)
        INTEGER EL,NE_G,TY(NB_F),DEBUT,FIN
        LOGICAL ABLA(NB_F+1),FERMER,QUART
        COMMON/FRONT/NOF,NOF_M/DONN/EL,NOI,N_ELE/CONTOUR/FERMER
        COMMON/FORM/QUART
        DATA EPSILON/1.0E-6/
C
        IF (.NOT.FERMER) THEN
          DEBUT=2
          FIN=NOF_M-1
          L=1
          IF ((TY(1).EQ.1).AND.(.NOT.ABLA(1)).AND.
     +       (VAL_INC(1).GE.0.0)) THEN
            IF (ABS(FLUX(1,2)).GE.EPSILON) THEN
              TY(1)=5
              DO I=1,NOF_M
                H(I,1)=-G(I,1)
                G(I,1)=0.0
              END DO
              DO I=1,N_ELE
                HD(I,1)=-GD(I,1)
                GD(I,1)=0.0
              END DO
              VAL(1)=0.0
              ABLA(1)=.TRUE.
            END IF
          END IF
          IF ((TY(NOF_M).EQ.1).AND.(.NOT.ABLA(NOF_M)).AND.
     +       (VAL_INC(NOF_M).GE.0.0)) THEN
            IF (ABS(FLUX(NOF_M,1)).GE.EPSILON) THEN
              TY(NOF_M)=5
              DO I=1,NOF_M
                H(I,NOF_M)=-G(I,NE_G)
                G(I,NE_G)=0.0
              END DO
              DO I=1,N_ELE
                HD(I,NOF_M)=-GD(I,NE_G)
                GD(I,NE_G)=0.0
              END DO
              VAL(NE_G)=0.0
```

```
          ABLA(NOF_M)=.TRUE.
        END IF
      END IF
    ELSE
      L=0
      DEBUT=1
      FIN=NOF
    END IF
    DO J=DEBUT,FIN
      IF ((TY(J).EQ.4).OR.(TY(J).EQ.5).OR.(J.EQ.1)) THEN
        L=L+1
      ELSE
        L=L+2
      END IF
      IF ((TY(J).EQ.1).AND.(.NOT.ABLA(J)).AND.
     +    (VAL_INC(J).GE.0.0)) THEN
        IF ((ABS(FLUX(J,1)).LT.EPSILON).AND.
     +      (ABS(FLUX(J,2)).GE.EPSILON)) THEN
          TY(J)=2
          DO I=1,NOF_M
            H(I,J)=-G(I,L)
            G(I,L)=0.0
          END DO
          DO I=1,N_ELE
            HD(I,J)=-GD(I,L)
            GD(I,J)=0.0
          END DO
          VAL(L)=0.0
          ABLA(J)=.TRUE.
        ELSE IF ((ABS(FLUX(J,1)).GE.EPSILON).AND.
     +           (ABS(FLUX(J,2)).LT.EPSILON)) THEN
          TY(J)=3
          IF (J.EQ.1) THEN
            DO I=1,NOF_M
              H(I,1)=-G(I,NE_G)
              G(I,NE_G)=0.0
            END DO
            DO I=1,N_ELE
              HD(I,1)=-GD(I,NE_G)
              GD(I,NE_G)=0.0
            END DO
            VAL(NE_G)=0.0
          ELSE
            DO I=1,NOF_M
              H(I,J)=-G(I,L-1)
              G(I,L-1)=0.0
            END DO
            DO I=1,N_ELE
              HD(I,J)=-GD(I,L-1)
              GD(I,L-1)=0.0
            END DO
            VAL(L-1)=0.0
```

```
              END IF
              ABLA(J)=.TRUE.
            ELSE IF ((ABS(FLUX(J,1)).GE.EPSILON).AND.
     +               (ABS(FLUX(J,2)).GE.EPSILON)) THEN
              TY(J)=5
              IF (J.EQ.1) THEN
                DO I=1,NOF_M
                  H(I,1)=-(G(I,NE_G) + G(I,1))
                  G(I,1)=0.0
                  G(I,NE_G)=0.0
                END DO
                DO I=1,N_ELE
                  HD(I,1)=-(GD(I,NE_G) + GD(I,1))
                  GD(I,1)=0.0
                  GD(I,NE_G)=0.0
                END DO
                VAL(NE_G)=0.0
                VAL(1)=0.0
                NE_G=NE_G-1
              ELSE
                DO I=1,NOF_M
                  H(I,J)=-(G(I,L-1) + G(I,L))
                  DO M=L,NE_G-1
                    G(I,M)=G(I,M+1)
                  END DO
                  G(I,L-1)=0.0
                  G(I,NE_G)=0.0
                END DO
                DO I=1,N_ELE
                  HD(I,J)=-(GD(I,L-1) + GD(I,L))
                  DO M=L,NE_G-1
                    GD(I,M)=GD(I,M+1)
                  END DO
                  GD(I,L-1)=0.0
                  GD(I,NE_G)=0.0
                END DO
                DO M=L,NE_G-1
                  VAL(M)=VAL(M+1)
                END DO
                VAL(L-1)=0.0
                VAL(NE_G)=0.0
                NE_G=NE_G-1
                L=L-1
              END IF
              ABLA(J)=.TRUE.
            END IF
          END IF
        END DO
        IF (.NOT.FERMER) CALL SYMETRIE_ABLA(ABLA)
        ABLA(NOF+1)=ABLA(1)
C
C----------
```

```
C
      RETURN
      END


      SUBROUTINE DEP_FRONT(XF,YF,FLUX,DU_DN,DS,DT,TY,ABLA)
C
C-------------------------------------------------------------
C
C     ROUTINE SERVANT A DEPLACER LA FRONTIERE DE
C     CHANGEMENT DE PHASE EN UTILISANT LA CONDITION
C     FRONTIERE DE STEPHAN.
C
C-------------------------------------------------------------
C
      PARAMETER (NB_F = 32)
      REAL XF(0:NB_F+1),YF(0:NB_F+1),K,RHO_L,NORME,FLUX(NB_F,2),
     +     DS(NB_F),DU_DN(NB_F),M(NB_F,2),XF_TEMP(0:NB_F+1),
     +     YF_TEMP(0:NB_F+1),DT
      INTEGER TY(NB_F),NOF,NOF_M
      LOGICAL ABLA(NB_F+1),FERMER
      COMMON/FRONT/NOF,NOF_M/PROP/ALPHA,K,RHO_L/CONTOUR/FERMER
C
      DO I=1,NOF_M
        DS(I)=0.0
      END DO
      DO I=0,NOF_M+1
        XF_TEMP(I)=XF(I)
        YF_TEMP(I)=YF(I)
      END DO
      DO I=1,NOF_M
        IF (ABLA(I)) THEN
          X=YF_TEMP(I+1)-YF_TEMP(I-1)
          Y=XF_TEMP(I-1)-XF_TEMP(I+1)
          DS(I)=(FLUX(I,2)+K*DU_DN(I))*DT/RHO_L
          NORME=SQRT((X*X)+(Y*Y))
          M(I,1)=X/NORME
          M(I,2)=Y/NORME
          XF(I)=XF_TEMP(I)+M(I,1)*DS(I)
          YF(I)=YF_TEMP(I)+M(I,2)*DS(I)
        END IF
      END DO
      IF (.NOT.FERMER) CALL SYMETRIE_FRONT(XF,YF)
      CALL LONGUEUR(XF,YF)
      CALL ENLEVE_ELE_FRONT(XF,YF,TY,ABLA,XF_TEMP,YF_TEMP)
C
C-----------
C
      RETURN
      END
```

```
      SUBROUTINE ENLEVE_ELE_FRONT(XF,YF,TY,ABLA,XF_AV,YF_AV)
C
C-----------------------------------------------------------------------
C
C     ROUTINE SERVANT A ENLEVER LE NOEUD J+1 LORSQUE L'ELEMENT J EST
C     DEVENU TROP PETIT.
C
C-----------------------------------------------------------------------
C
      PARAMETER(NB_F = 32)
      REAL XF(0:NB_F+1),YF(0:NB_F+1),LXY(0:NB_F+1),LONG_MIN,
     +     XF_AV(0:NB_F+1),YF_AV(0:NB_F+1)
      INTEGER TY(NB_F),NOF,NOF_M,I,FIN
      LOGICAL FERMER,ABLA(NB_F+1),QUART
      COMMON/FRONT/NOF,NOF_M/CONTOUR/FERMER/LONG/LXY/FORM/QUART
      DATA LONG_MIN/1.0E-3/
C
      IF (FERMER) THEN
        FIN=NOF-1
      ELSE
        FIN=NOF_M-2
        IF (LXY(NOF_M-1).LT.LONG_MIN) THEN
          LXY(NOF_M-2)=SQRT((XF(NOF_M)-XF(NOF_M-2))**2+
     +                      (YF(NOF_M)-YF(NOF_M-2))**2)
          XF(NOF_M-1)=XF(NOF_M)
          YF(NOF_M-1)=YF(NOF_M)
          TY(NOF_M-1)=TY(NOF_M)
          ABLA(NOF_M-1)=ABLA(NOF_M)
          LXY(NOF_M-1)=LXY(NOF_M)
          NOF_M=NOF_M-1
        END IF
      END IF
      I=1
      DO WHILE (I.LE.FIN)
        IF (LXY(I).LT.LONG_MIN) THEN
          LXY(I)=SQRT((XF(I+2)-XF(I))**2+(YF(I+2)-YF(I))**2)
          DO J=I+1,NOF_M-1
            XF(J)=XF(J+1)
            YF(J)=YF(J+1)
            TY(J)=TY(J+1)
            ABLA(J)=ABLA(J+1)
            LXY(J)=LXY(J+1)
          END DO
          NOF_M=NOF_M-1
          IF (FERMER) THEN
            XF(0)=XF(NOF_M)
            YF(0)=YF(NOF_M)
            XF(NOF+1)=XF(1)
            YF(NOF+1)=YF(1)
          END IF
```

```
          END IF
          I=I+1
        END DO
        IF (FERMER) THEN
          IF (LXY(NOF).LT.LONG_MIN) THEN
            LXY(NOF-1)=SQRT((XF(NOF-1)-XF(1))**2+
     +                      (YF(NOF-1)-YF(1))**2)
            NOF=NOF-1
          END IF
        ELSE
          IF (YF(2).LE.0.0) THEN
            YF(1)=0.0
            XF(1)=XF(2)
            LXY(1)=LXY(2)
            DO I=2,NOF_M-1
              XF(I)=XF(I+1)
              YF(I)=YF(I+1)
              TY(I)=TY(I+1)
              ABLA(I)=ABLA(I+1)
              LXY(I)=LXY(I+1)
            END DO
            NOF_M=NOF_M-1
          END IF
          IF ((.NOT.QUART).AND.(YF(NOF_M-1).LT.0.0)) THEN
            XF(NOF_M)=0.0
            YF(NOF_M)=0.0
            YF(NOF_M-1)=0.0
            ABLA(NOF_M-1)=ABLA(NOF_M)
            TY(NOF_M-1)=TY(NOF_M)
            NOF_M=NOF_M-1
          END IF
          CALL SYMETRIE_FRONT(XF,YF)
          CALL SYMETRIE_ABLA(ABLA)
          CALL LONGUEUR(XF,YF)
        END IF
        ABLA(NOF+1)=ABLA(1)
C
C--------
C
        RETURN
        END


        SUBROUTINE DIMIN_NB_COUCHE(T)
C
C-------------------------------------------------------
C
C        ROUTINE SERVANT A DECIDER LE NOMBRE DE DIVISIONS
C        QUE L'ON PEUT PRENDRE POUR L'EVALUATION DE L'IN-
C        TEGRALE SUR LE DOMAINE.
C
```

```
C-------------------------------------------------------
C
      PARAMETER (NB_E = 532,NB_DIV = 10)
      REAL T(NB_E),MAX,MIN,LAMBDA(NB_DIV),DIFF
      INTEGER EL,NB_COUCHE
      COMMON/DONM/EL,NOI,N_ELE/MESH/LAMBDA,NB_COUCHE
C
      MIN=1.0E30
      MAX=-1.0E30
      DO I=1,N_ELE
        IF (T(I).GT.MAX) MAX=T(I)
        IF (T(I).LT.MIN) MIN=T(I)
      END DO
      DIFF=MAX-MIN
      LAMBDA(1)=0.0
      IF (DIFF.GT.200.0) THEN
        NB_COUCHE=6
        LAMBDA(2)=0.1
        LAMBDA(3)=0.2
        LAMBDA(4)=0.4
        LAMBDA(5)=0.6
        LAMBDA(6)=0.8
      ELSE IF ((DIFF.LE.200.0) .AND. (DIFF.GT.100.0)) THEN
        NB_COUCHE=5
        LAMBDA(2)=0.1
        LAMBDA(3)=0.2
        LAMBDA(4)=0.4
        LAMBDA(5)=0.7
      ELSE IF ((DIFF.LE.100.0) .AND. (DIFF.GT.50.0)) THEN
        NB_COUCHE=4
        LAMBDA(2)=0.2
        LAMBDA(3)=0.4
        LAMBDA(4)=0.7
      ELSE IF ((DIFF.LE.50.0) .AND. (DIFF.GT.10.0)) THEN
        NB_COUCHE=3
        LAMBDA(2)=0.3
        LAMBDA(3)=0.6
      ELSE IF ((DIFF.LE.10.0) .AND. (DIFF.GT.2.0)) THEN
        NB_COUCHE=2
        LAMBDA(2)=0.4
      ELSE
        NB_COUCHE=1
      END IF
C
C----------
C
      RETURN
      END


      SUBROUTINE SYMETRIE_FRONT(XF,YF)
```

```
C
C-----------------------------------------------------------------
C
C       ROUTINE SERVANT A COMPLETER LA FRONTIERE PAR SYMETRIE
C
C-----------------------------------------------------------------
C
        PARAMETER(NB_F = 32)
        REAL XF(0:NB_F+1),YF(0:NB_F+1)
        INTEGER NOF,NOF_M,N,M,I
        LOGICAL QUART
        COMMON/FRONT/NOF,NOF_M/FORM/QUART
C
        N=NOF_M-1
        IF (QUART) THEN
          NOF=4*N
          M=2*N+1
          DO I=1,N
            XF(NOF_M+I)=-XF(NOF_M-I)
            YF(NOF_M+I)=YF(NOF_M-I)
          END DO
        ELSE
          M=NOF_M
          NOF=2*N
        END IF
        DO I=1,M-2
          XF(NOF-(I-1))=XF(I+1)
          YF(NOF-(I-1))=-YF(I+1)
        END DO
        XF(0)=XF(NOF)
        YF(0)=YF(NOF)
        XF(NOF+1)=XF(1)
        YF(NOF+1)=YF(1)
C
C------------
C
        RETURN
        END


        SUBROUTINE SYMETRIE_ABLA(ABLA)
C
C-----------------------------------------------------------------
C
C       ROUTINE SERVANT A DIRE QUELS NOEUDS A ATTEINT
C       LA TEMPERATURE DE FUSION PAR SYMETRIE.
C
C-----------------------------------------------------------------
C
        PARAMETER(NB_F = 32)
        LOGICAL ABLA(NB_F+1),QUART
```

```
      INTEGER NOE,NOE_M,N,M,I
      COMMON/FRONT/NOE,NOE_M/FORM/QUART
C
      N=NOE_M-1
      IF (QUART) THEN
        M=2*N+1
        DO I=1,N
          ABLA(NOE_M+I)=ABLA(NOE_M-I)
        END DO
      ELSE
        M=NOE_M
      END IF
      DO I=1,M-2
        ABLA(NOE-(I-1))=ABLA(I+1)
      END DO
C
C------------
C
      RETURN
      END


      SUBROUTINE AIRE_TRIANGLES(E,XI,YI,AIRE)
C
C-------------------------------------------------------
C
C         ROUTINE CALCULANT L'AIRE DES ELEMENTS
C         TRIANGULAIRES.(M**2)
C
C-------------------------------------------------------
C
      PARAMETER (NB_D = 365,NB_E = 532,NB_DIV = 10)
      REAL XI(NB_D),YI(NB_D),AIRE(NB_E),LAMBDA(NB_DIV)
      INTEGER E(NB_E,3),EL,NB_COUCHE
      LOGICAL FERMER
      COMMON/DOMM/EL/MESH/LAMBDA,NB_COUCHE/CONTOUR/FERMER
      COMMON/FRONT/NOE,NOE_M
C
      IF (FERMER) THEN
        N1=NOE
      ELSE
        N1=NOE_M-1
      END IF
      N2=3*N1
      NB_TRI=NOE*3
      I1=0
      DO K=1,NB_COUCHE-1
        M=(K-1)*NB_TRI
        DO I=1+M,M+N2
          I1=I1+1
          AIRE(I1)=ABS((XI(E(I,2))-XI(E(I,1)))*
```

```
     +              (YI(E(I,3))-YI(E(I,1)))
     +             -(XI(E(I,3))-XI(E(I,1)))*
     +              (YI(E(I,2))-YI(E(I,1))))*0.5
          END DO
        END DO
        M=(NB_COUCHE-1)*NB_TRI
        DO I=1+M,N1+M
          I1=I1+1
          AIRE(I1)=ABS((XI(E(I,2))-XI(E(I,1)))*
     +                 (YI(E(I,3))-YI(E(I,1)))
     +                -(XI(E(I,3))-XI(E(I,1)))*
     +                 (YI(E(I,2))-YI(E(I,1))))*0.5
        END DO
C
C---------------
C
        RETURN
        END



       SUBROUTINE CPUTIME
C
C-----------------------------------------------------------------
C
C       ROUTINE SERVANT A CALCULER LE TEMPS 'CPU' ET D'AUTRES
C       STATISTIQUES ET A LES IMPRIMER DANS LE FICHIER 'STATS.DAT'
C
C-----------------------------------------------------------------
C
        INTEGER*4 TIMER_DATA,TIMER_ADDR,TIMER_ROUTINE,STATUS
        EXTERNAL TIMER_ROUTINE
        INTEGER*4 LIB$INIT_TIMER,LIB$SHOW_TIMER,LIB$FREE_TIMER
        EXTERNAL  LIB$INIT_TIMER,LIB$SHOW_TIMER,LIB$FREE_TIMER
        LOGICAL FIRST
        DATA FIRST/.TRUE./
C
        TIMER_DATA=2
        IF(FIRST)THEN
C
        STATUS = LIB$INIT_TIMER (TIMER_ADDR)
        IF(.NOT.STATUS)CALL LIB$SIGNAL (%VAL(STATUS))
        FIRST=.FALSE.
C
        RETURN
C
        ENDIF
C
        STATUS = LIB$SHOW_TIMER (TIMER_ADDR,,TIMER_ROUTINE,TIMER_DATA)
        IF(.NOT.STATUS)CALL LIB$SIGNAL (%VAL(STATUS))
        STATUS = LIB$FREE_TIMER (TIMER_ADDR)
        IF(.NOT.STATUS)CALL LIB$SIGNAL (%VAL(STATUS))
```

```
C
      FIRST=.TRUE.
C
C-------
C
      RETURN
      END


      INTEGER FUNCTION TIMER_ROUTINE (STATS,TIMER_DATA)
C
C
      CHARACTER*(*) STATS
      CHARACTER *40 NOM,ENDROIT
      CHARACTER *80 NOMFICH
      CHARACTER*13 TEMPS_CALCUL
      INTEGER TIMER_DATA
      COMMON/FICHIER/ENDROIT
C
      NOM='SORTIE.DAT'
      NOMFICH=ENDROIT//NOM
C
      OPEN (10,FILE=NOMFICH,STATUS='UNKNOWN',ACCESS='APPEND')
      I=1
      J=INDEX(STATS(I:),'CPU')
      IF (J.NE.0) THEN
        I=I+(J-1)+4
        L=1
        DO K=1,13
          IF (K.EQ.4) THEN
            TEMPS_CALCUL(K:K)=' '
          ELSE
            TEMPS_CALCUL(K:K)=STATS(I+L-1:I+L-1)
            L=L+1
          END IF
        END DO
      ELSE
        TEMPS_CALCUL='INCONNU'
      END IF
      WRITE(10,10) TEMPS_CALCUL
10    FORMAT(2X,T46,'  ( h:min:  s  ) ',/
     +         ,2X,'TEMPS DE CALCUL SUR UN VAX-II/785',T45,':   'A)
```