FEJavaDemo

1. Problèmes considérés

Grâce à la méthode des éléments finis, FEJavaDemo peut traiter des problèmes de conductivité thermique.
On cherche le champ de température stationnaire *u(x)* dans un domaine bidimensionnel *Ω*, qui peut être composé de plusieurs matériaux ayant des coefficients de conductivité *Λ(x)* différents. Le domaine peut contenir des sources de chaleur qui sont décrites par une fonction *f*.

Trois types de conditions aux limites du domaine sont pris en compte :

- Les conditions de Dirichlet (frontière Γ_1) : La température $g_1(x)$ est fixée.
- Les conditions de Neumann (frontière Γ_2): Le flux de chaleur $g_2(x)$ à travers la frontière est fixé ($g_2(x)=0$ permet de simuler une isolation thermique)
- Les conditions de Fourier / Robin (frontière Γ_3): Modélisation d'un échange libre de chaleur avec l'extérieur (le flux de chaleur est proportionnel (coefficient $\alpha(x)$) à la différence entre la température sur la frontière $u(x)$ et la température extérieure $u_A(x)$)

Formulation classique:

On cherche $u \in C^2(\Omega) \cap C^1(\Omega \cup \Gamma_2 \cup \Gamma_3) \cap C(\bar{\Omega})$, tel que :

$$
\begin{aligned}
-\text{div}(\Lambda(x)\ \text{grad}\ u(x)) &= f(x) && \forall x \in \Omega\ , \\
u(x) &= g_1(x) && \forall x \in \Gamma_1\ , \\
\frac{\partial u}{\partial N} &= g_2(x) && \forall x \in \Gamma_2\ , \\
\frac{\partial u}{\partial N} &= \alpha(x)\ \left[u_A(x) - u(x)\right] && \forall x \in \Gamma_3\ ,
\end{aligned}
$$

où $\Gamma = \partial\Omega = \bar{\Gamma}_1 \cup \bar{\Gamma}_2 \cup \bar{\Gamma}_3$ et $\Gamma_i \cap \Gamma_j = \emptyset$ pour $i \neq j$

et $\Lambda(x) = \begin{pmatrix} \lambda_{xx}(x) & 0 \\ 0 & \lambda_{yy}(x) \end{pmatrix}$ avec $\frac{\partial u}{\partial N} = \lambda_1(x)\frac{\partial u}{\partial x_1} n_1 + \lambda_2(x)\frac{\partial u}{\partial x_2} n_2$

et $\vec{n}(x) = \begin{pmatrix} n_1(x) \\ n_2(x) \end{pmatrix}$: vecteur unitaire sortant au point x $\in \Gamma$.

Formulation variationnelle du problème :

On cherche $u \in V_{g_1}$ tel que

$$a(u,v) = \langle F, v \rangle \qquad \forall v \in V_0$$

où :

$$a(u,v) = \int_\Omega (\text{grad } v(x))^T \Lambda(x) \text{ grad } v(x)\ dx + \int_{\Gamma_3} \alpha(x) u(x) v(x) ds ,$$

$$\langle F, v \rangle = \int_\Omega f(x) v(x) dx + \int_{\Gamma_2} g_2(x) v(x) ds + \int_{\Gamma_3} \alpha(x) u_A(x) v(x) ds ,$$

$$V_{g_1} = \left\{ u \in H^1(\Omega) : u(x) = g_1(x) \text{ auf } \Gamma_1 \right\},$$

$$V_0 = \left\{ v \in H^1(\Omega) : v(x) = 0 \text{ auf } \Gamma_1 \right\}.$$

Le programme ne peut naturellement pas travailler avec n'importe quelles fonctions.

- En ce qui concerne la fonction *f*, on est quand même relativement libre : La fonction peut avoir une définition par domaine. De plus, elle peut être définies grâce aux fonctions tan(), sin(), cos(), exp(), ln(), sqrt() et à la constante π. A titre d'exemple, la fonction f(x,y) = 2*cos(Pi*x*y) est reconnue par le programme.
- Λ est défini par λ_1 et λ_2. Ces coefficients sont des constantes qui peuvent être différentes pour chaque domaine.
- Concernant les conditions de Dirichlet, la température $g_1(x)$ n'est donnée qu'en un nombre fini de points. Ces points sont appelés les « points de Dirichlet ».

Les autres fonctions $g_2(x)$, $\alpha(x)$ et $u_A(x)$ qui modélisent les conditions aux limites sont constantes sur chaque arête frontière. Elles sont ainsi constantes par morceaux.

Méthode de Galerkin

Le principe de la méthode de Galerkin est de remplacer l'espace de dimension infinie dans lequel on cherche la fonction solution par un espace de dimension finie V_h :

$$V_h = \left\{ v_h \ : \ v_h = \sum_{i \in \bar{\omega}_h} v_i p_i(x) \right\} = span\left\{ p_i : i \in \bar{\omega}_h \right\} \quad \subset \quad V = H^1(\Omega)$$

où $\bar{\omega}_h$ est l'ensemble contenant les numéros de tous les nœuds du maillage.

Les fonctions p_i sont appelées « les fonctions de forme ». Elles sont linéairement indépendantes.

Comme la valeur de la solution aux points de Dirichlet est connue, on cherche une solution approchée du problème sous la forme :

$$u_h(x) = \sum_{j \in \omega_h} u_j p_j(x) + \sum_{j \in \gamma_h} u_{*,j} p_j(x)$$

où seuls les u_j sont inconnus.

ω_h est l'ensemble des numéros des nœuds de $\Omega \cup \Gamma_2 \cup \Gamma_3$ et γ_h l'ensemble des nœuds de $\overline{\Gamma}_1$.

($\overline{\omega}_h = \omega_h \cup \gamma_h$)

On obtient finalement le système suivant :

On cherche $\underline{u}_h = [u_j]_{j \in \omega_h} \in \mathbb{R}^{N_h}$ tel que :

$$\sum_{j \in \omega_h} u_j a(p_j, p_i) = \langle F, p_i \rangle - \sum_{j \in \gamma_h} u_{*,j} a(p_j, p_i) \qquad \forall i \in \omega_h$$

c'est-à-dire $\underline{u}_h = [u_j]_{j \in \omega_h} \in \mathbb{R}^{N_h}$ tel que :

$$K_h \underline{u}_h = \underline{f}_h$$

où

$$K_h = [a(p_j, p_i)]_{i,j \in \omega_h}$$

$$\underline{f}_h = \left[\langle F, p_i \rangle - \sum_{j \in \gamma_h} u_{*,j} a(p_j, p_i) \right]_{i \in \omega_h} \in \mathbb{R}^{N_h}$$

N_h est le nombre de nœuds situés dans $\Omega \cup \Gamma_2 \cup \Gamma_3$.

2. Discrétisation relative aux éléments finis

2.1. Type de maillage et raffinage

Le domaine Ω sur lequel on cherche la solution du problème est découpé en éléments finis $T^{(r)}$. Avec FEJavaDemo, ce peut être des triangles ou des quadrilatères.

Lorsque le programme travaille avec des triangles et des fonctions de formes p_i linéaires par morceaux, il existe une possibilité de raffiner le maillage. Après un tel raffinage, une nouvelle procédure de résolution est exécutée avec le maillage raffiné. Lorsque l'utilisateur demande un raffinage, il doit entrer un écart de température maximal. Le programme raffine alors le maillage jusqu'à ce que l'écart de température entre les 3 sommets de chaque triangle soit inférieur à la valeur seuil souhaitée. Si l'utilisateur a entré 0, le maillage est raffiné entièrement une fois (chaque triangle est divisé en quatre).

Algorithme de raffinage du maillage :

(1) Partage des triangles pour lesquels l'écart de température entre deux sommets est trop important. Pour effectuer cette division, on crée un triangle dont les sommets sont les milieux des 3 côtés du triangle initial.

(2) Tant que des divisions de triangles sont effectuées

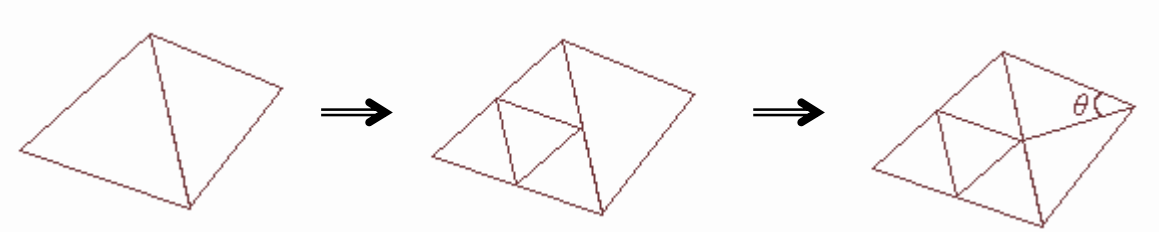
   (i) Dans le maillage raffiné, partage en quatre des triangles qui n'ont pas déjà été divisés en quatre, mais où des nœuds ont été générés sur au moins deux de leur côtés.

   (ii) Pour chaque triangle où exactement un nœud a été généré sur un côté :

      - Si un partage en deux du triangle générerait deux triangles de mauvaise qualité (c'est-à-dire que l'angle θ serait trop petit, $\theta < 25°$), alors partage du triangle en quatre.

      - Sinon, partage du triangle en deux, en reliant ce nœud généré par une précédente division, au sommet opposé.

(3) Résolution du problème avec le maillage raffiné. S'il reste des écarts de température supérieurs à la valeur seuil donnée, retour à l'étape (1).



Raffinage de maillage avec partage en quatre des triangles

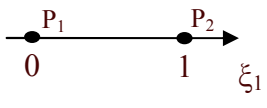
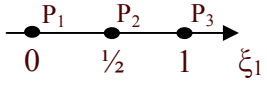
2.2. Choix des fonctions de forme

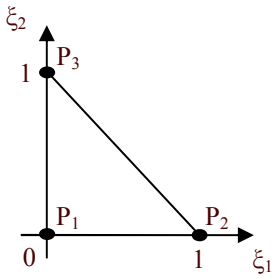
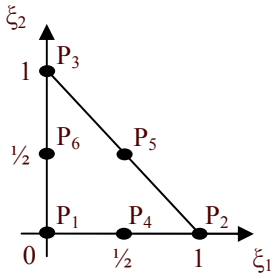
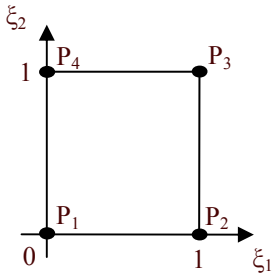
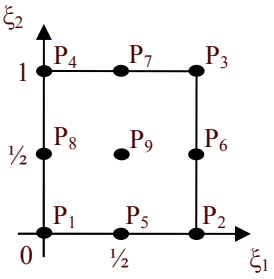
Pour les fonctions de forme définies sur l'élément de référence, nous choisissons des fonctions polynomiales φ_α qui, en chaque nœud P_β de l'élément de référence, satisfont la condition $\varphi_\alpha(\xi_\beta) = \delta_{\alpha\beta}$.

Le programme gère les fonctions de forme linéaires et quadratiques.

Seules les fonctions de forme définies sur l'élément de référence sont nécessaires pour le calcul de la matrice du rigidité et du vecteur second membre, car tous les calculs d'intégrales sont ramenés à des calculs sur l'élément de référence. Le programme utilise pour ceci des transformations pour passer de l'élément de référence à un élément quelconque $T^{(r)}$ du maillage.

Les fonctions de forme suivantes sont utilisées :

	$\varphi_1^l(\xi) = -\xi_1 + 1$ $\varphi_2^l(\xi) = \xi_1$
	$\varphi_1^q(\xi) = 2\xi^2 - 3\xi + 1$ $\varphi_2^q(\xi) = -4\xi^2 + 4\xi$ $\varphi_3^q(\xi) = 2\xi^2 - \xi$

	$\varphi_1^{l2}(\xi_1,\xi_2) = 1\text{-}\xi_1 - \xi_2$ $\varphi_2^{l2}(\xi_1,\xi_2) = \xi_1$ $\varphi_3^{l2}(\xi_1,\xi_2) = \xi_2$
	$\varphi_\alpha^{q2}(\xi_1,\xi_2) = \varphi_\alpha^{l2}\left(2\varphi_\alpha^{l2}-1\right), \quad \alpha = 1,2,3$ $\varphi_4^{q2}(\xi_1,\xi_2) = 4\varphi_1^{l2}\varphi_2^{l2}$ $\varphi_5^{q2}(\xi_1,\xi_2) = 4\varphi_2^{l2}\varphi_3^{l2}$ $\varphi_6^{q2}(\xi_1,\xi_2) = 4\varphi_3^{l2}\varphi_1^{l2}$
	$\varphi_1^{bl}(\xi_1,\xi_2) = \varphi_1^{l}(\xi_1)\varphi_1^{l}(\xi_2)$ $\varphi_2^{bl}(\xi_1,\xi_2) = \varphi_2^{l}(\xi_1)\varphi_1^{l}(\xi_2)$ $\varphi_3^{bl}(\xi_1,\xi_2) = \varphi_2^{l}(\xi_1)\varphi_2^{l}(\xi_2)$ $\varphi_4^{bl}(\xi_1,\xi_2) = \varphi_1^{l}(\xi_1)\varphi_2^{l}(\xi_2)$
	$\varphi_1^{bq}(\xi_1,\xi_2) = \varphi_1^{q}(\xi_1)\varphi_1^{q}(\xi_2)$ $\varphi_2^{bq}(\xi_1,\xi_2) = \varphi_3^{q}(\xi_1)\varphi_1^{q}(\xi_2)$ $\varphi_3^{bq}(\xi_1,\xi_2) = \varphi_3^{q}(\xi_1)\varphi_3^{q}(\xi_2)$ $\varphi_4^{bq}(\xi_1,\xi_2) = \varphi_1^{q}(\xi_1)\varphi_3^{q}(\xi_2)$ $\varphi_5^{bq}(\xi_1,\xi_2) = \varphi_2^{q}(\xi_1)\varphi_1^{q}(\xi_2)$ $\varphi_6^{bq}(\xi_1,\xi_2) = \varphi_3^{q}(\xi_1)\varphi_2^{q}(\xi_2)$ $\varphi_7^{bq}(\xi_1,\xi_2) = \varphi_2^{q}(\xi_1)\varphi_3^{q}(\xi_2)$ $\varphi_8^{bq}(\xi_1,\xi_2) = \varphi_1^{q}(\xi_1)\varphi_2^{q}(\xi_2)$ $\varphi_9^{bq}(\xi_1,\xi_2) = \varphi_2^{q}(\xi_1)\varphi_2^{q}(\xi_2)$

2.3. Méthode de stockage et renumérotation des nœuds

Comme les fonctions de forme ont un support qu'on peut qualifier de “local” (quelques éléments), la matrice de rigidité est creuse. Les éléments $K_{ij} = a(p_i,p_j)$ ne sont en effet différents de zéro que lorsque l'intersection des intérieurs des supports des fonctions p_i et p_j est non vide.

De plus, avec une numérotation des nœuds appropriée, la matrice a une structure en bande. La largeur de la bande, c'est à dire le plus gros écart entre un élément non nul d'une colonne à l'élément de la diagonale correspondant, dépend de la numérotation des nœuds.

Il est très profitable d'utiliser ces deux propriétés pour le stockage de K. Le programme utilise le format dit « Skyline » pour le stockage de la matrice : seuls les éléments situés dans l’ « enveloppe » (ou « profil ») sont stockés, c’est-à-dire entre le 1er élément non nul d’une colonne et l’élément de la diagonale correspondant.

La structure et le degré d’occupation de la matrice de rigidité dépendent de la numérotation des nœuds.

Ainsi, le nombre d’éléments stockés dépend de la numérotation des nœuds. Il est en effet important que les nœuds de chaque élément aient des numéros voisins.

A partir d'une numérotation des nœuds donnée, le programme peut procéder à une renumérotation avec l'algorithme de Cuthill McKee pour réduire la taille du profil de la matrice.

Le programme peut utiliser des fonctions de forme quadratiques (voire cubiques, ...) à partir de fichiers .net écrits pour des fonctions de forme linéaires. Dans ce cas, seuls les 3 sommets des triangles sont numérotés et le programme ajoute les nœuds supplémentaires (3 par triangle dans le cas des fonctions de forme quadratiques) en leur attribuant les numéros qui suivent. Il est possible de procéder automatiquement, dès le chargement des fichiers, à une renumérotation des nœuds pour accélérer le processus de résolution du problème.

Algorithme de Cuthill McKee :

L’idée qui est à la base de l’algorithme est qu’il faut prendre en compte les nœuds voisins le plus conjointement possible dans le processus de renumérotation, afin d’éviter des écarts trop importants dans les numéros des nœuds d’un même élément, qui conduisent à de mauvais profils.

Ainsi par exemple, le fait de numéroter les nœuds d'un même niveau selon leur degré croissant (le degré *d(z)* d'un nœud *z* correspond au nombre de ses voisins) repose sur cette constatation triviale que les nœuds avec beaucoup de voisins doivent avoir des numéros globaux les plus grands possibles afin de conserver des écarts d'index faibles dans le niveau suivant.

1. Détermination des voisins de la racine *r*. Ces nœuds doivent être numérotés selon leur degré croissant. Lorsque des nœuds ont le même degré, un choix arbitraire est évidemment nécessaire. Les nœuds numérotés dans cette étape sont à la distance 1 de la racine *r*. Ils forment le 1er niveau.

2. Les nœuds du 1er niveau sont considérés successivement selon leur nouveau numéro. Pour chacun d'eux, numérotation de leur voisins non renumérotés par degré croissant. Les nœuds qui sont numérotés lors de cette étape sont à la distance 2 de la racine et forment le 2ème niveau du processus de renumérotation.
   On répète alors ce processus jusqu'à ce que tous les nœuds aient été renumérotés..

3. Il a été prouvé que le profil pouvait encore être relativement réduit en inversant l'ordre de numérotation des nœuds qui est généré par cet algorithme.

Implémentation de l'algorithme :

(1) Recherche du nœud initial ou « racine » *r*. Il reçoit le numéro 1.
$S = \{r\}$

(2) Tant que tous les nœuds n'ont pas été renumérotés :
- $S_{nou} = \{\}$
- Pour chaque nœud *x* de *S*,
  - Détermination de tous les voisins du nœud *x* qui n'ont pas déjà été renumérotés : $\{k_1, k_2, ..., k_r\}$, et numérotation selon leur degré croissant.
  - $S_{nou} = S_{nou} + \{k_1, k_2, ..., k_r\}$
- $S = S_{nou}$

(3) Inversion de la numérotation grâce à la substitution : $k \rightarrow n+1-k$

Algorithme pour la détermination de la racine :

Pour cette étape, on divise le maillage en niveau.

$R(g) = (L_1, L_2, ..., L_r)$ s'appelle « la structure en niveaux avec racine g », lorsque :

1. $\bigcup_{i=1}^{r} L_i$ = Ensemble des nœuds du maillage
2. L_r est non vide
3. $L_1 = \{g\}$
4. Le plus court chemin entre un noeud $h \in L_i$ et g comporte i-1 arêtes

r s'appelle « la profondeur » de la structure en niveaux. Sa largeur k est le nombre de nœuds du niveau qui en contient le plus.

Gibbs, Poole et Stockmeyer ont proposé d'utiliser des structures en niveaux avec la largeur la plus faible. Celle-ci correspond en général à la structure en niveau qui a la plus grande profondeur. En effet, logiquement, plus il y a de niveaux, moins il y a de nœuds par niveaux.

Idéalement, il faudrait commencer par déterminer ce que l'on appelle « le diamètre » du maillage, qui correspond au chemin le plus court entre deux points du maillage les plus éloignés possible. Le problème est bien sûr résoluble mais l'on se contente la plupart du temps d'un algorithme "approximatif".

Avec un tel algorithme, g n'est pas toujours l'extrémité d'un diamètre du maillage mais il s'en approche. On parle souvent dans ce cas d'un « pseudo-diamètre ».

L'algorithme suivant est très approprié à la recherche d'un bon noeud de départ, car les dépenses en terme de temps de calcul et de place mémoire sont relativement restreintes, étant donné qu'en règle générale, peu de nœuds doivent être testés pour trouver un pseudo-diamètre.

(1) Choix d'un noeud g de degré minimum

(2) Construction de la structure en niveaux $R(g) = (L_1, L_2, ..., L_r)$. Tri des éléments f_j du derniers niveaux L_r selon leur degré croissant, c'est-à-dire $d(f_j) \leq d(f_{j+1})$

(3) Poser $j = 1$

(4) Calcul de la profondeur de la structure en niveaux $R(f_j)$: s. Si $s > r$, poser $g = f_j$ et retour à l'étape (2)

(5) Si $j < l$, incrémentation de j et retour à l'étape (4)

2.4. Mise en œuvre de la méthode des éléments finis

2.4.1. Calcul et assemblage de la matrice de rigidité et du vecteur second membre

Le système à résoudre est établi au niveau des éléments. Le programme assemble dans un premier temps la matrice de rigidité et le vecteur second membre sans prendre en compte les conditions aux limites.

Lors du calcul des matrices de rigidité élémentaires $K^{(r)}$, on procède à un changement de variables dans les intégrales afin de n'avoir à les évaluer que sur l'élément de référence.

Matrice de rigidité élémentaire correspondant à l'élément $T^{(r)}$:

$$K^{(r)} = \left[\int_{\hat{T}} \left[\left(grad_{\xi} \varphi_i(\xi) \right)^T \left(J^{(r)} \right)^{-1} \Lambda \left(J^{(r)} \right)^{-T} grad_{\xi} \varphi_j(\xi) \right] \left| \det J^{(r)} \right| d\xi \right]_{i,j=1}^{\hat{N}}$$

où :

- ξ : coordonnées sur l'élément de référence
- φ_i et φ_j sont les fonctions de forme définies sur l'élément de référence
- $\Lambda = \begin{pmatrix} \lambda_{xx} & 0 \\ 0 & \lambda_{yy} \end{pmatrix}$: coefficients de conductivité du matériau
- $\hat{N}$: nombre de nœuds par élément
- $J^{(r)}$: matrice jacobienne de la transformation qui permet de passer de l'élément de référence $\hat{T}$ à un élément quelconque $T^{(r)}$ du maillage

Seuls les matrices $J^{(r)}$, $(J^{(r)})^{-1}$ et les dérivées des fonctions de forme définies sur l'élément de référence sont nécessaires pour le calcul des matrices de rigidité élémentaires. Les formules explicites des fonctions de forme de chaque élément $T^{(r)}$ ne sont pas requises.

Les vecteurs second membre élémentaires $f^{(r)}$ sont calculés de la même façon :

$$\underline{f}^{(r)} = \left[\int_{\hat{T}} f\left(x_{T^{(r)}}(\xi) \right) \varphi_i(\xi) \left| \det J^{(r)} \right| d\xi \right]_{i=1}^{\hat{N}}$$

où : $x_{T^{(r)}}(\xi) = J^{(r)} \xi + x_1^{(r)}$ représente les coordonnées sur l'élément $T^{(r)}$

Lors de ce premier processus, la matrice de rigidité et le vecteur second membre sont aussi modifiés pour tenir compte des conditions aux limites de Dirichlet.

La matrice de rigidité est corrigée de la sorte :

$$K_{ij}^{(r)} = K_{ji}^{(r)} = \delta_{ij} \qquad \forall i \in \gamma_h^{(r)}, j \in \bar{\omega}_h^{(r)}$$

où $\bar{\omega}_h^{(r)}$: ensemble contenant les numéros de tous les nœuds de l'élément $T^{(r)}$

On corrige aussi le second membre :

$$f_i^{(r)} := f_i^{(r)} - \sum_{j \in \gamma_h^{(r)}} K_{ij} g_1(x_j) \qquad \forall i \in \omega_h^{(r)}$$

où :

- $g_1(x_j)$: valeur des conditions de Dirichlet au point x_j
- $\omega_h^{(r)}$: ensemble contenant les numéros des nœuds qui ne sont pas de Dirichlet dans l'élément $T^{(r)}$
- $\gamma_h^{(r)}$: ensemble contenant les numéros des nœuds de Dirichlet de l'élément $T^{(r)}$

Il aurait aussi été possible d'effectuer ces corrections à la fin du processus d'assemblage mais c'est plus facile de le faire au niveau élémentaire où les matrices et les vecteurs sont naturellement plus petits. De plus, avec le format "*Skyline*" il est difficile, une fois la matrice totalement assemblée, de trouver les lignes qui doivent être mises à zero.

Les matrices de rigidité élémentaires et les vecteurs second membre élémentaires sont alors assemblés par l'algorithme suivant qui utilise une boucle sur chaque domaine car ceux-ci peuvent avoir des coefficients de conductivité différents.

Pour chaque domaine

  Pour chaque élément $T^{(r)}$ du domaine

    Calcul de $K^{(r)}$ et de $\underline{f}^{(r)}$

    Prise en compte des conditions aux limites de Dirichlet : Correction de $\underline{f}^{(r)}$ et de $K^{(r)}$

    Pour chaque nœud de l'élément de numéro global i et de numéro local k .

      $f_i := f_i + f_k^{(r)}$

      Pour chaque noeud de l'élément de numéro global j et de numéro local l

      $K_{ij} := K_{ij} + K_{kl}^{(r)}$

L'étape suivante consiste à prendre en compte les conditions aux limites :

Conditions aux limites de type 2 :

Vecteur qui doit être assemblé pour les conditions aux limites de Neumann :

$$\underline{f}^{(e_2)} = \left[\left(\int_0^1 \varphi_i(\xi) d\xi \right) g_2 \sqrt{\left(x_{n_2} - x_{n_1}\right)^2 + \left(y_{n_2} - y_{n_1}\right)^2} \right]_{i=1}^{\bar{N}_E}$$

où :

- φ_i : fonctions de forme définies sur l'intervalle de référence
- $\bar{N}_E$: nombre de nœuds par arête
- n_1 et n_2 : extrémités de l'arête
- g_2 : valeur de la condition aux limites sur l'arête

Conditions aux limites de type 3 :

Vecteur qui doit être assemblé pour les conditions aux limites de Robin :

$$\underline{f}^{(e_3)} = \left[\left(\int_0^1 \varphi_i(\xi) d\xi \right) \alpha\, u_A \sqrt{\left(x_{n_2} - x_{n_1}\right)^2 + \left(y_{n_2} - y_{n_1}\right)^2} \right]_{i=1}^{\bar{N}_E}$$

où α et u_A sont les valeurs de la condition aux limites sur l'arête

Matrice qui doit être assemblée pour les conditions aux limites de Robin :

$$K^{(e_3)} = \left[\left(\int_0^1 \varphi_i(\xi) \varphi_j(\xi) d\xi \right) \alpha \sqrt{\left(x_{n_2} - x_{n_1}\right)^2 + \left(y_{n_2} - y_{n_1}\right)^2} \right]_{i=1}^{\bar{N}_E}$$

Le programme corrige aussi bien sûr ce vecteur et cette matrice pour prendre en compte les conditions aux limites de type 1. Ils sont corrigés de façon analogue à ce qui a déjà été vu pour les matrices de rigidité et les vecteurs second membre élémentaires.

Ces matrices et vecteurs sont assemblés par le même algorithme que celui vu précédemment.

Conditions aux limites de type 1 :

Comme les conditions aux limites de Dirichlet ont déjà été en partie traitées lors du calcul de la matrice de rigidité et du vecteur second membre sans prise en compte des conditions aux limites et lors de la prise en compte des conditions aux limites de Robin, il ne reste plus désormais qu'à placer des "1" sur la diagonale de la matrice de rigidité globale et les valeurs des nœuds de Dirichlet sur le vecteur second membre :

Pour chaque nœud de Dirichlet de numéro global i :

$$K_{ii} = 1$$

$$f_i = g_1(x_i)$$

où $g_1(x_i)$ est la valeur de la condition de Dirichlet au point x_i.

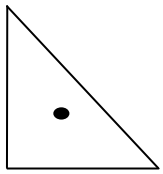
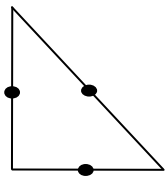
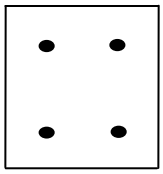
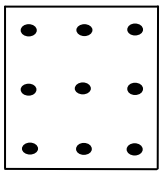
2.4.2. Intégration numérique

Lors du calcul des éléments des matrices et des vecteurs, il n'est pas toujours possible de calculer les intégrales de façon analytique. C'est pourquoi le programme utilise des formules de quadratures dont le principe est d'approcher l'intégrale par une somme. FEJavaDemo utilise des quadratures de la forme :

$$\int_{\bar{T}} w(\xi)d\xi \approx \sum_{i=1}^{l} \alpha_i w(\xi_i)$$

Comme les polynômes que le programme doit intégrer sont toujours d'un degré relativement petit, FEJavaDemo utilise assez de points de quadrature pour calculer en fait les intégrales de façon exacte :

Position des points de quadrature	Coordonnées ξ_i ou $(\xi_{i,1}, \xi_{i,2})$ des points de quadrature	Poids α_i	exact pour polynômes de degré k
	$\frac{3-\sqrt{3}}{6}, \frac{3+\sqrt{3}}{6}$	$\frac{1}{2}, \frac{1}{2}$	k = 3
	$\frac{5-\sqrt{15}}{10}, \frac{1}{2}, \frac{5+\sqrt{15}}{10}$	$\frac{5}{18}, \frac{8}{18}, \frac{5}{18}$	k = 5

	$\left(\frac{1}{3},\frac{1}{3}\right)$	$\frac{1}{2}$	k = 1
	$\left(\frac{1}{2},0\right),\left(\frac{1}{2},\frac{1}{2}\right),\left(0,\frac{1}{2}\right)$	$\frac{1}{6},\frac{1}{6},\frac{1}{6}$	k = 2
	$\left(\frac{3-\sqrt{3}}{6},\frac{3+\sqrt{3}}{6}\right),\left(\frac{3+\sqrt{3}}{6},\frac{3+\sqrt{3}}{6}\right)$ $\left(\frac{3-\sqrt{3}}{6},\frac{3-\sqrt{3}}{6}\right),\left(\frac{3+\sqrt{3}}{6},\frac{3-\sqrt{3}}{6}\right)$	$\frac{1}{4},\frac{1}{4}$ $\frac{1}{4},\frac{1}{4}$	k = 3
	$\left(\frac{5-\sqrt{15}}{10},\frac{5+\sqrt{15}}{10}\right)\left(\frac{1}{2},\frac{5+\sqrt{15}}{10}\right)\left(\frac{5+\sqrt{15}}{10},\frac{5+\sqrt{15}}{10}\right)$ $\left(\frac{5-\sqrt{15}}{10},\frac{1}{2}\right)\left(\frac{1}{2},\frac{1}{2}\right)\left(\frac{5+\sqrt{15}}{10},\frac{1}{2}\right)$ $\left(\frac{5-\sqrt{15}}{10},\frac{5-\sqrt{15}}{10}\right)\left(\frac{1}{2},\frac{5-\sqrt{15}}{10}\right)\left(\frac{5+\sqrt{15}}{10},\frac{5-\sqrt{15}}{10}\right)$	$\frac{25}{324},\frac{10}{81},\frac{25}{324}$ $\frac{10}{81},\frac{16}{81},\frac{10}{81}$ $\frac{25}{324},\frac{10}{81},\frac{25}{324}$	k = 5

Formules de quadrature sur l'intervalle de référence I = [0,1] et sur l'élément de référence

Sur les quadrilatères, les formules sont exactes pour des fonctions qui sont des polynômes de degré *k* dans chaque direction.

L'intégration numérique est utilisée plusieurs fois :

- Lors du calcul des matrices de rigidité et des vecteurs second membre élémentaires. Le programme procède à un changement de variables afin de n'avoir à calculer des intégrales que sur l'élément de référence (triangle, carré, etc.)
- Lors de la prise en compte des conditions aux limites. De même, FEJavaDemo ne calcule des intégrales que sur l'intervalle de référence I = [0,1].

2.5. Résolution du système

Les différents processus d'assemblage conduisent à l'équation linéaire suivante : $K\,u = f$.
K est la matrice de rigidité, f le vecteur second membre et u le vecteur solution cherché.

Pour résoudre une telle équation issue de la méthode des éléments finis, on peut utiliser des méthodes directes ou des méthodes itératives. Les méthodes de résolution directe consistent en des algorithmes qui renvoient le vecteur solution u après un certain nombre d'étapes. Les méthodes itératives construisent une suite (u_k), qui, à partir d'une première approximation u_0 converge vers le vecteur solution u.

Nous savons que la matrice K est symétrique définie positive. Dans ce cas, la *méthode de Cholesky*, qui fait partie des méthodes de résolution directe, est fréquemment utilisée.

La méthode de Cholesky consiste à écrire la matrice K comme étant le produit d'une matrice triangulaire supérieure par sa transposée : $K = S^T S$

Nous savons aussi que la matrice K est creuse. Cette propriété est conservée lors de la factorisation de Cholesky. Ainsi, les matrices K et S peuvent être sauvegardées dans des emplacements mémoire similaires. De plus, le programme peut utiliser un algorithme de décomposition particulier qui ne calcule pas les éléments nuls de S, ce qui est très bénéfique :

Algorithme de décomposition ($K = S^T S$) avec prise en compte du profil de K :

$$S_{11} = \sqrt{K_{11}}$$

Pour $j = 2, 3, \ldots, N$

Si $l_0(j)+1 < j$, alors pour $i = l_0(j)+1, l_0(j)+2, \ldots, j-1$:

$$S_{ij} = \frac{1}{S_{ii}}\left(K_{ij} - \sum_{l=\max\{l_0(i),l_0(j)\}+1}^{i-1} S_{li} S_{lj} \right)$$

$$S_{jj} = \sqrt{K_{jj} - \sum_{l=l_0(j)+1}^{j-1} S_{lj}^2}$$

(où $l_0(j)$ est le numéro de ligne pour lequel, dans la $j^{\text{ème}}$ colonne, $K_{ij} = 0$ pour $0 < i < l_0(j) + 1$ et $K_{l_0(j)+1,\,j} \neq 0$)

L'équation $Ku = f$ est alors équivalente à $S^T S\, u = f$. Le programme résout ce système en deux étapes grâce à une *descente* et à une *remontée*. Cela signifie qu'il résout d'abord $S^T y = f$ puis $S u = y$. Ces deux systèmes sont particulièrement faciles à résoudre car ce sont des systèmes triangulaires.

Algorithme de descente de Cholesky :

$$\begin{pmatrix} S_{11} & 0 & \cdots & 0 \\ S_{12} & S_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ S_{1N} & S_{2N} & \cdots & S_{NN} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{pmatrix}$$

$$y_1 = \frac{f_1}{S_{11}}$$

$$y_i = \frac{1}{S_{ii}} \left(f_i - \sum_{j=1}^{i-1} S_{ji} y_j \right) \qquad i = 2,\ 3,\ \ldots,\ N$$

Algorithme de remontée de Cholesky :

$$\begin{pmatrix} S_{11} & S_{12} & \cdots & S_{1N} \\ 0 & S_{22} & \cdots & S_{2N} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & S_{NN} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

$$u_N = \frac{y_N}{S_{NN}}$$

$$u_i = \frac{1}{S_{ii}} \left(y_i - \sum_{j=i+1}^{N} S_{ij} u_j \right) \qquad i = N-1,\ N-2,\ \ldots,\ 1$$

3. Structure des fichiers de données

Pour pouvoir travailler sur un nouveau maillage, il faut écrire deux fichiers de données : un fichier ".net" et un fichier ".dat". Le premier contient les informations sur le maillage et le second sur les conditions aux limites et la fonction *f*.

3.1. Le fichier .net

Les lignes qui commencent par # sont des commentaires et sont ignorées par le programme.

La 1ère ligne décrit le type des éléments utilisés dans le maillage (1 pour triangles, 2 pour quadrilatères, etc.).

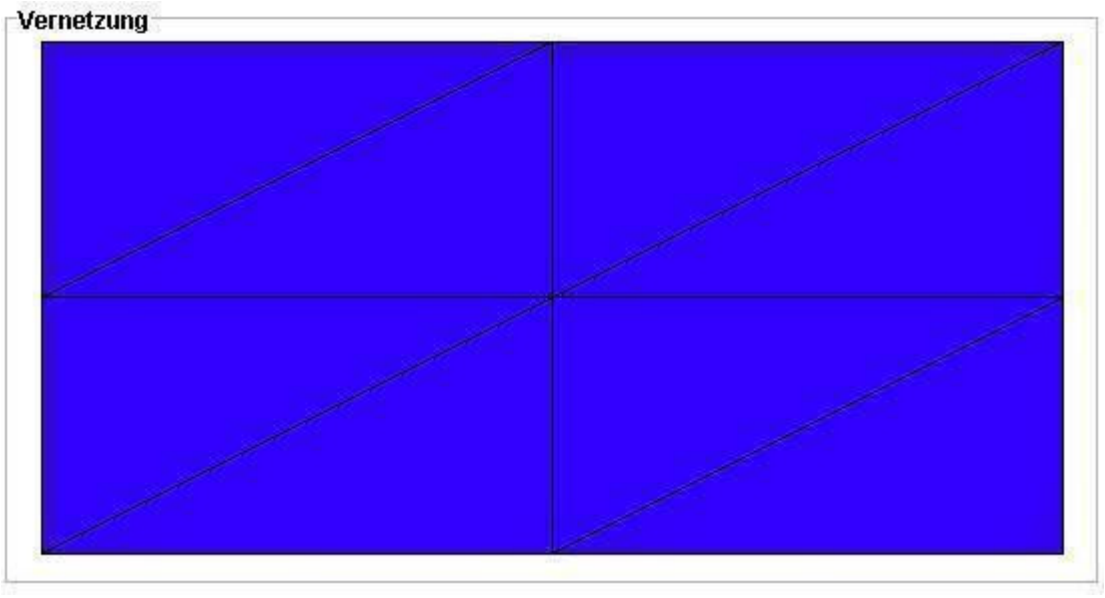
Suivent le nombre de nœuds (c'est-à-dire la dimension de la matrice de rigidité globale) et le nombre d'éléments.

Les nœuds sont alors définis grâce à 3 valeurs : leur numéro global, leur abscisse et leur ordonnée.

Ce sont ensuite les éléments qui sont définis : chacun a un numéro global suivi des numéros globaux de leurs sommets (3 numéros pour les triangles, 4 pour les quadrilatères, …) et de numéro du domaine dans lequel il se situe.

La ligne suivante contient le nombre d'arêtes-frontières. Celles-ci sont définies juste en dessous par leur numéro global ainsi que par les nœuds qui sont à leurs extrémités.

L'exemple qui suit est le fichier .net d'un maillage très simple :



```
# FEJavaDemo – Fichier .net
#
# Type des éléments (1 pour triangles, 2 pour quadrilatères, etc.)
  1
#
# Nombre de noeuds et nombre d'éléments
  9 8
#
# Numéros globaux et coordonnées des noeuds
  1  0.0  0.0
  2  0.0  1.25
  3  0.0  2.5
  4  2.5  0.0
  5  2.5  1.25
  6  2.5  2.5
  7  5.0  0.0
  8  5.0  1.25
  9  5.0  2.5
#
# Numéros globaux des sommets des éléments (et numéro du domaine)
  1  4  5  1  1
  2  2  1  5  1
  3  5  6  2  1
  4  3  2  6  1
  5  7  8  4  1
  6  5  4  8  1
  7  8  9  5  1
  8  6  5  9  1
#
# Nombre d'arêtes-frontières
  8
#
# Numéros globaux et extrémités des arêtes-frontières
  1   1  4
  2   4  7
  3   7  8
  4   8  9
  5   9  6
  6   6  3
  7   3  2
  8   2  1
```

3.2. Le fichier .dat

Le fichier commence par le nombre de domaines différents qui composent le maillage puis suivent les coefficients de conductivité λ_1 et λ_2 de chaque domaine.

Les domaines-frontières sont alors définis:

Le nombre de différents domaines-frontières est d'abord écrit.

Pour chaque domaine, suivent alors deux nombres : le nombres d'arêtes qui le composent puis le type de conditions aux limites (1 pour Dirichlet, 2 pour Neumann et 3 pour Robin).

Dans les lignes qui suivent, on peut lire les numéros globaux des arêtes-frontières qui composent chaque domaine avec, pour chaque arête, la ou les valeur(s) des conditions aux limites correspondantes :

- Deux valeurs doivent être fournies pour les conditions aux limites de Dirichlet : les températures aux deux extrémités du segment.
- En ce qui concerne les conditions aux limites de Neumann, une seule valeur est nécessaire : le flux de chaleur à travers l'arête $\left(\frac{\partial u}{\partial N} = a \right)$.
- Pour les conditions aux limites de Robin, on doit entrer les deux valeurs a et b qui caractérise le transfert $\left(\frac{\partial u}{\partial N} = a \left[b - u(x) \right] \right)$.

Les dernières informations du fichier concernent la fonction *f*. Elle peut être différente pour chaque domaine et doit être écrite entièrement (par exemple : 2*x+tan(y)+7). Les fonctions qui peuvent être utilisées pour les définir sont les suivantes : tan(), sin(), cos(), exp(), ln(), sqrt() et la constane π.

L'exemple suivant est le fichier .dat qui correspond au fichier .net précédent :

```
# FEDemoJava – fichier .dat
#
# Nombre de domaines
  2
# Lambda1 et Lambda 2 dans le domaine 1
  200.0  200.0
# Lambda1 et Lambda 2 dans le domaine 2
  100.0  100.0
#
# Nombre de Domaines-frontières
  3
# Nb d'arêtes et type de conditions aux limites dans le domaine-frontière 1
  4  1
# Nb d'arêtes et type de conditions aux limites dans le domaine-frontière 2
  2  2
# Nb d'arêtes et type de conditions aux limites dans le domaine-frontière 3
  2  3
#
# N° globaux et données (coeff a (et b)) des arêtes du domaine-frontière 1
  1   0.   10.
  2  10.   20.
  3  20.   30.
  4  30.   40.
# N° globaux et données (coeff a (et b)) des arêtesdu domaine-frontière 2
  5   5.
  6   7.
# N° globaux et données (coeff a (et b)) des arêtes du domaine-frontière 3
  7  10.   50.
  8  10.   50.
#
# Fonction F(x,y) dans le domaine 1
  3*cos(x*y*Pi)
# Fonction F(x,y) dans le domaine 2
  0
```

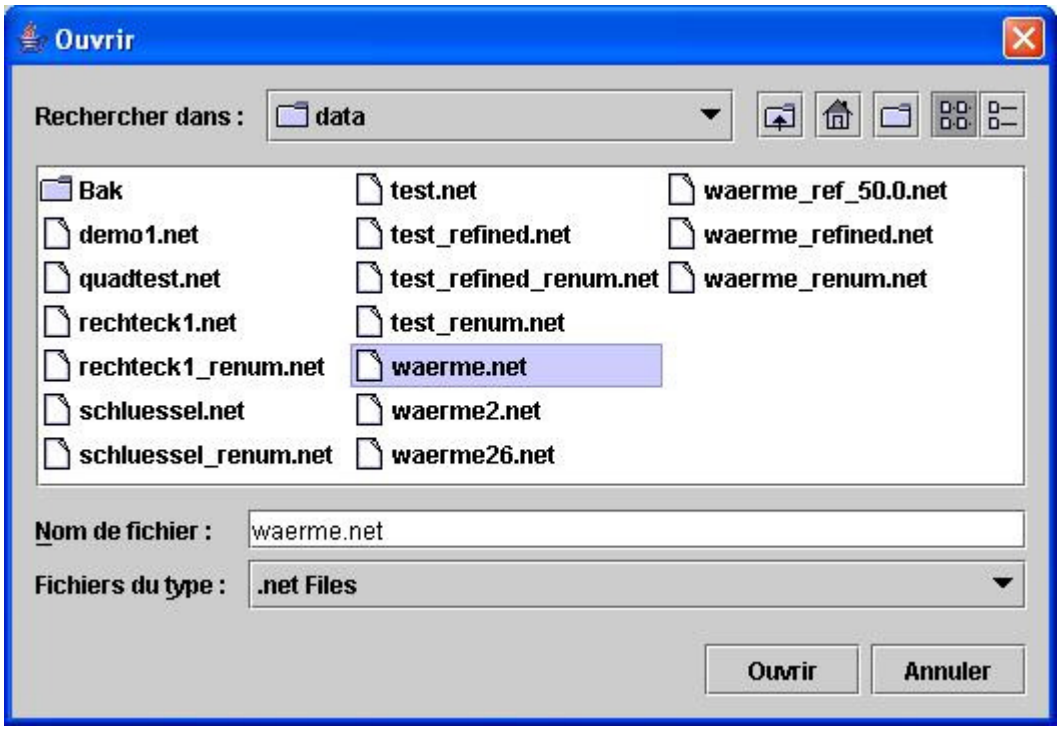
4. Déroulement du programme

Choix du type de fonctions de forme :

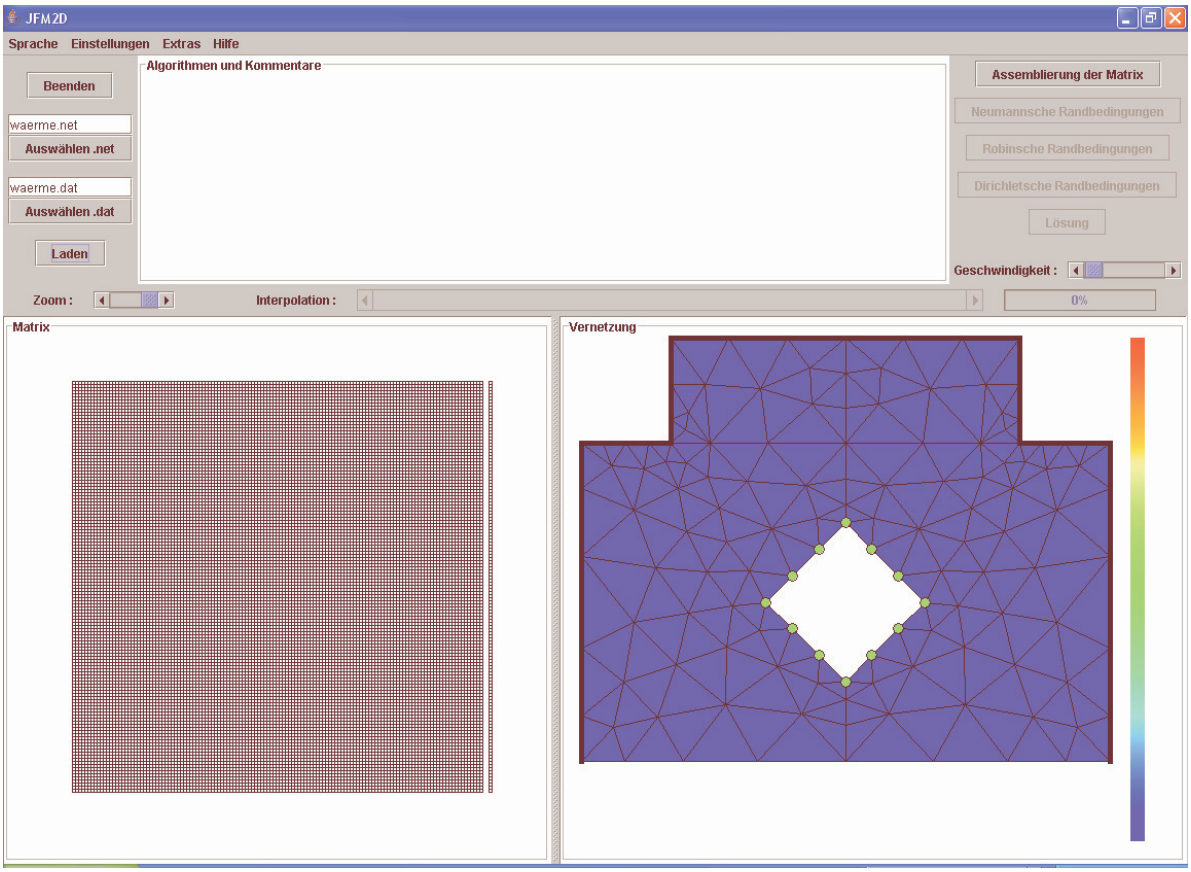
Le programme travaille par défaut avec des fonctions de forme linéaires. Grâce au menu « Propriétés → Type des fonctions de forme », on peut aussi choisir des fonctions de forme quadratiques.

Chargement des fichiers de données à utiliser :

En cliquant sur le bouton « Parcourir .net » (respectivement « Parcourir .dat »), les fichiers de maillages (respectivement les fichiers .dat) à disposition sont affichés.



Après avoir choisi les fichiers souhaités, les données seront chargées en cliquant sur le bouton « Chargement ».



Calcul de la matrice de rigidité et du vecteur second membre sans prise en compte des conditions aux limites :

Dès que le chargement des fichiers de données est terminé, le bouton « Assemblage de la matrice » devient actif. Il permet de démarrer le calcul et l'assemblage de la matrice de rigidité et du vecteur second membre sans prise en compte des conditions aux limites.

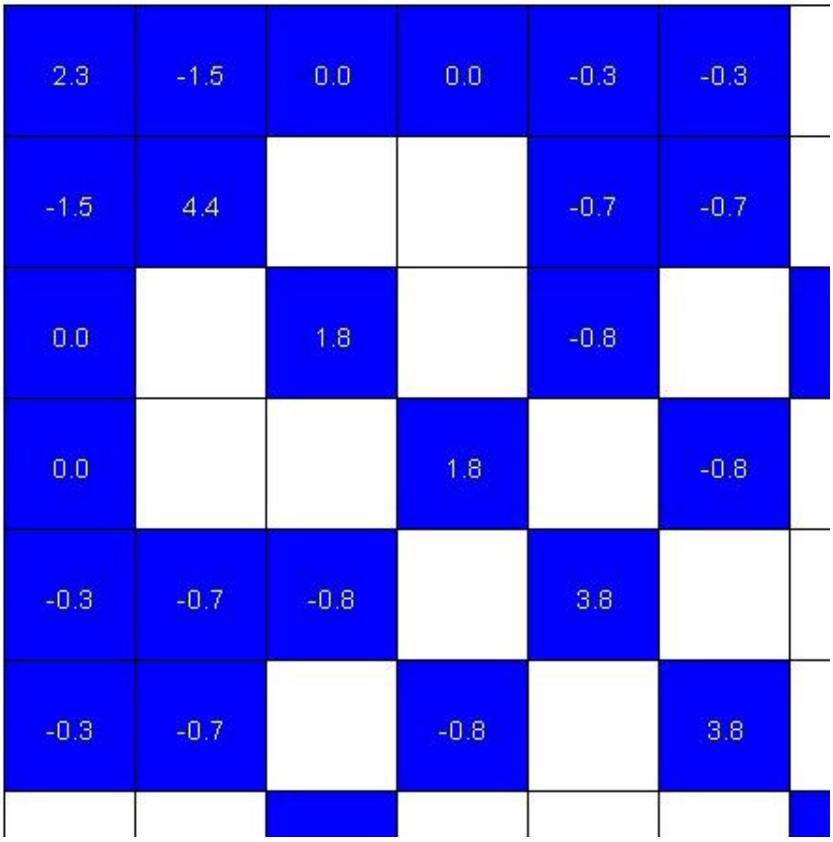
Pendant le déroulement du processus, un ascenseur permet de modifier la vitesse d'affichage. L'utilisateur préfère souvent que le programme livre la solution le plus rapidement possible. C'est pourquoi les étapes de l'assemblage ne sont plus affichées lorsque le curseur est placé à l'extrême gauche.

Prise en compte des conditions aux limites :

Les trois boutons suivant permettent la prise en compte des conditions aux limites.

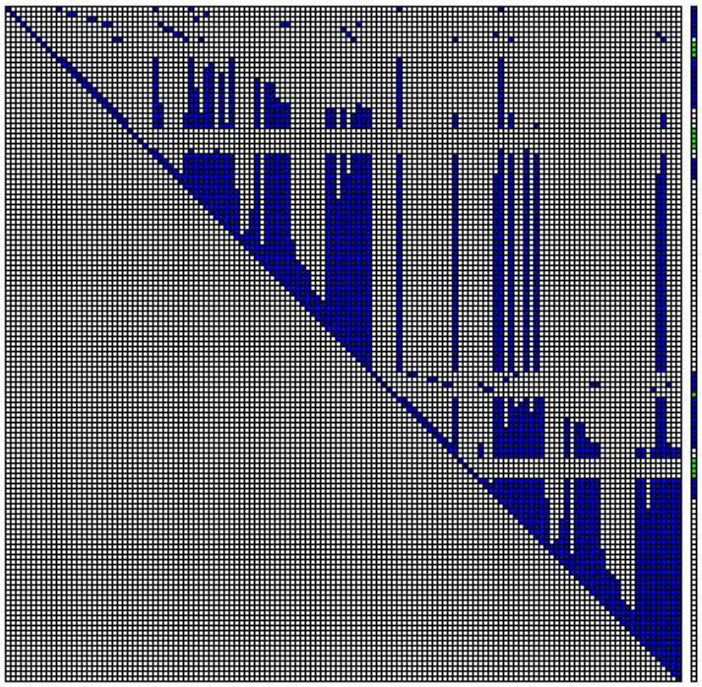
Lors des différents processus d'assemblage, les éléments du maillage sur lesquels travaillent le programme, ainsi que les cases correspondantes dans la matrice sont coloriés en rouge. Les éléments non nuls de la matrice sont en bleu afin de montrer que cette dernière est très creuse. Les cases vertes correspondent aux nœuds de Dirichlet.

Un ascenseur permet de modifier le zoom sur la matrice. Lorsque le curseur est à l'extrême gauche, les valeurs des cases de la matrice sont affichées. On peut aussi lire les numéros des lignes et des colonnes.

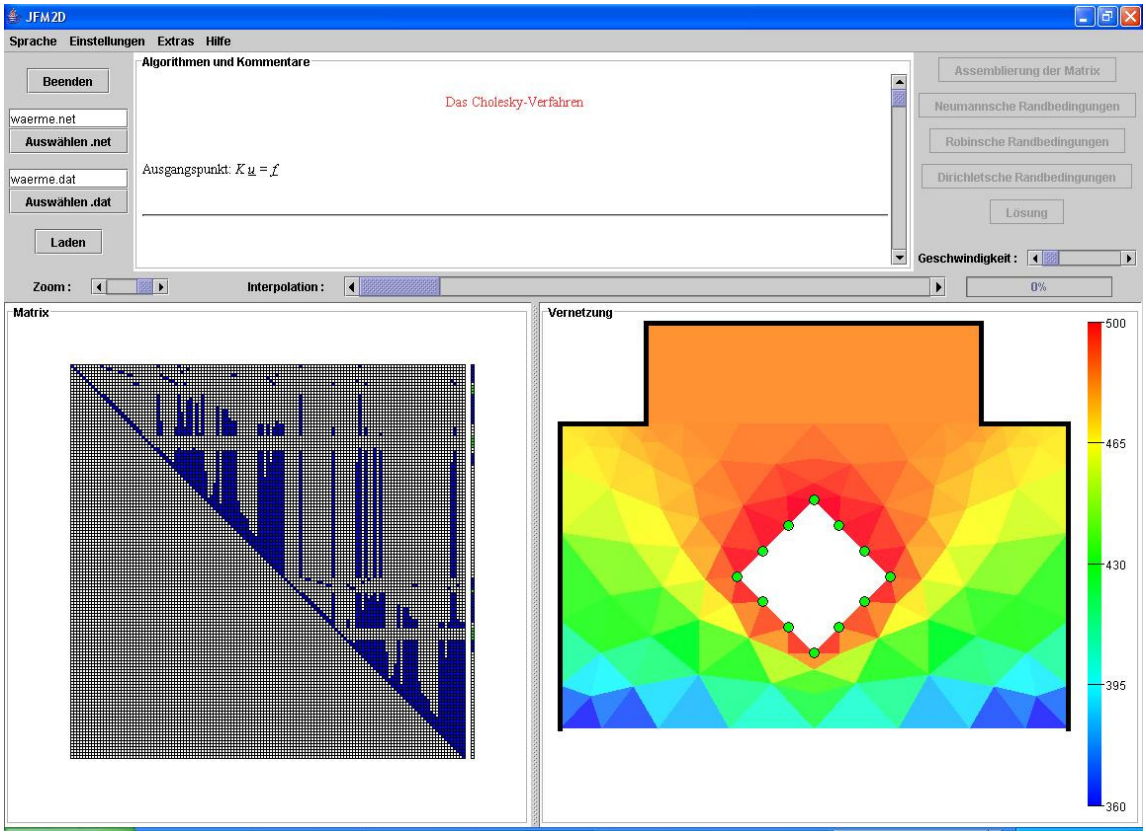


Résolution du système :

Lorsque toutes les conditions aux limites ont été prises en compte, le bouton « Résolution » devient actif. En cochant « Propriétés → Affichage → afficher la décomposition S'S », on peut observer la matrice diagonale supérieure S de la factorisation S^TS. Une petite animation symbolise le processus de descente et de remontée de Cholesky.

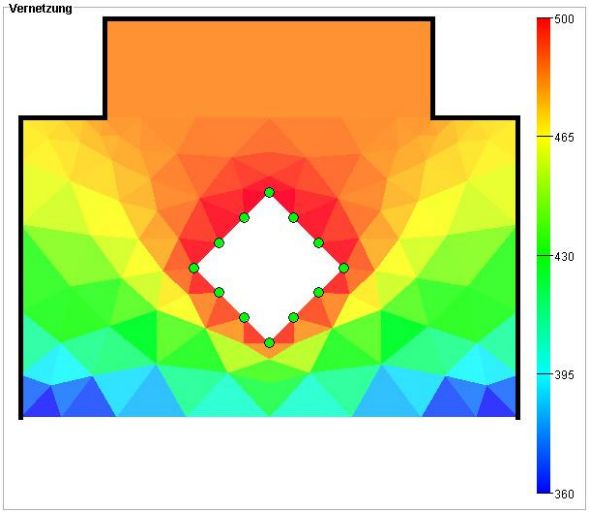


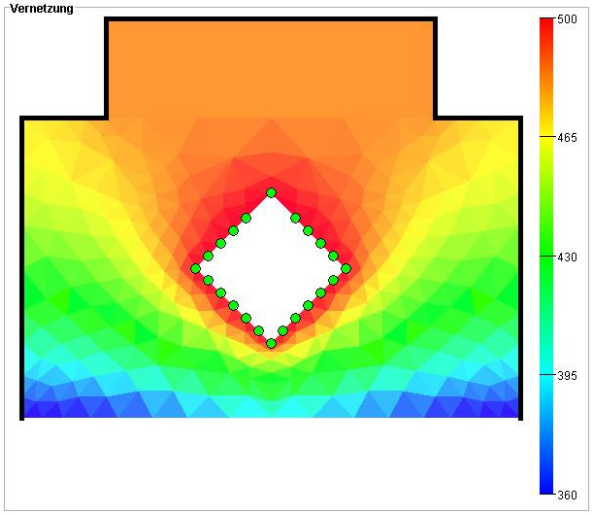
Lorsque la résolution est terminée, le champ de température est représenté, muni d'une échelle de température.



Raffinage du maillage :

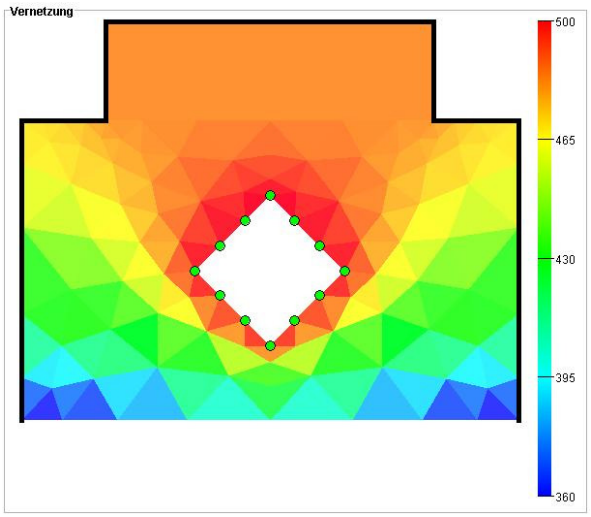
Si le programme travaille avec des fonctions de forme linéaires sur des triangles, on peut demander un raffinage du maillage (« Accessoires → Raffinage du maillage »). Dans ce cas, l'utilisateur doit entrer l'écart maximal de température entre deux nœuds. S'il entre "0", le maillage sera entièrement raffiné une fois. Il est possible de sauvegarder les fichiers du nouveau maillage raffiné (« Accessoires → Création de nouveaux fichiers… → ... lors d'un raffinage »).

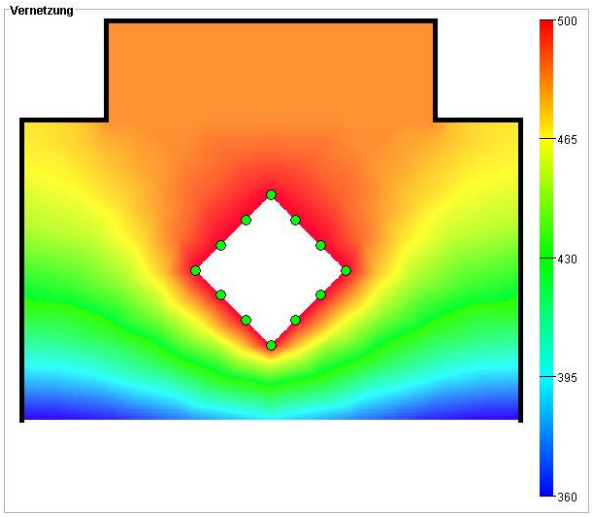




Interpolation de la solution :

Grâce à l'ascenseur « Interpolation », la représentation de la solution peut être améliorée: les éléments peuvent être divisés et la solution est alors interpolée linéairement aux nouveaux nœuds Comme ceci peut être assez long, une barre d'avancement permet de connaître quel pourcentage du processus a déjà été exécuté.





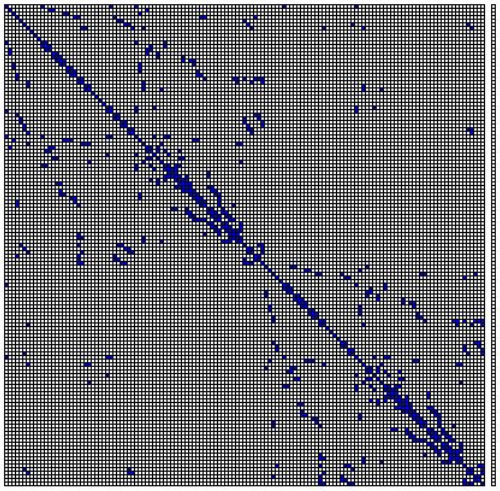
Sauvegarde de la solution :

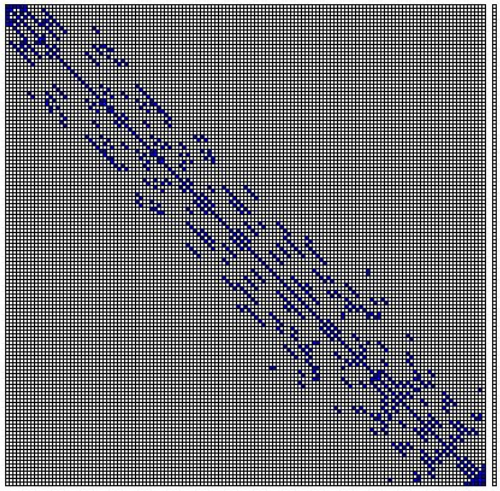
La solution (plus précisément les coordonnées des nœuds et leur température) peut être sauvegardée par le menu « Accessoires → Sauvegarder la solution ».

Renumérotation des nœuds :

Cette possibilité est offerte par le menu « Accessoires → Renumérotation des nœuds ».

La case à cocher « Renumérotation automatique des nœuds avec les fonctions quadratiques » est choisie par défaut. Ceci permet d'accélérer le processus de résolution. Il existe aussi une possibilité de sauvegarder la nouvelle numérotation grâce au menu (« Accessoires → Création de nouveaux fichiers… → ... lors d'une renumérotation »).





Autres possibilités :

Le logiciel peut bien sûr être traduit grâce au menu « Langue ».

Entre chaque étape, il est possible d'enregistrer des images Jpeg de la matrice et du maillage (« Accessoires → Création de Jpeg »).

Grâce au menu « Propriétés → Affichage → Afficher les conditions aux limites », il est possible de colorier les nœuds de Dirichlet en vert et de mettre en gras les bordures qui sont isolées thermiquement de l'extérieur. On peut aussi afficher les numéros globaux des nœuds (« Propriétés → Affichage → Afficher les numéros de nœuds »).