ENSEEIHT – 2ième Année Informatique et Mathématiques Appliquées 2007-2008
CALCUL SCIENTIFIQUE – MAJEURE INFORMATIQUE

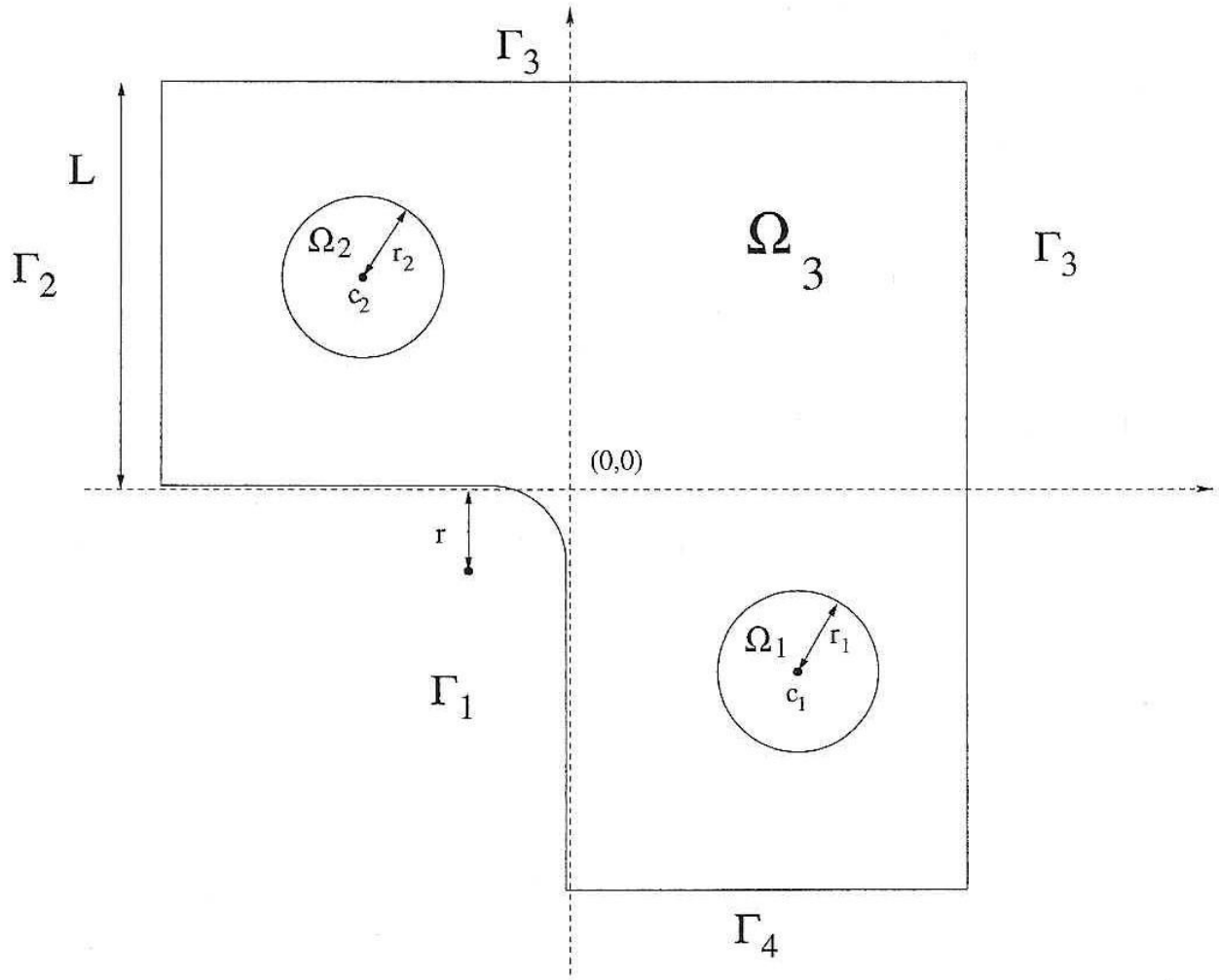
Bureau d'Etude Calcul Scientifique

Franville-Lafargue Thibault

Groupe G

1/ Sujet

Le but de ce bureau d'étude est d'étudier le transfert de chaleur dans une plaque mince. Cette plaque est en forme de L, centrée à l'origine.



Le domaine d'étude est appelé Ω et se décompose en 3 sous-domaines Ω_1, Ω_2 et Ω_3 où les conditions physiques sont différentes.

Γ est la frontière, subdivisée en 4 parties Γ_1, Γ_2, Γ_3 et Γ_4 sur lesquelles les conditions aux limites ne sont pas les mêmes.
L'équation du problème d'évolution est :

$$\rho C \frac{(\partial T)}{(\partial t)} - \lambda \, div(\nabla T) = F, (x, y) \in \Omega, t > 0$$

avec la condition initiale :

$$T(0, x, y) = 0, (x, y) \in \Omega_1 \, U \, \Omega_2 \, U \, \Omega_3$$

et comme terme source :

$$F(t,x,y)=\begin{cases} -2000\, si\,(x,y)\in\Omega_1 \\ 600\, si\,(x,y)\in\Omega_2 \\ 0\, si\,(x,y)\in\Omega_3 \end{cases}$$

et les conditions aux limites sur la frontière Γ :
- des conditions de Neumann homogènes sur Γ_1 et Γ_3,
- des conditions de radiations : $C\frac{(\partial T)}{(\partial n)}+kT=-20\, sur\, \Gamma_2$
- des conditions de Dirichlet : $T=-100\, sur\, \Gamma_4$

Caractéristiques physiques :
$\rho C = 1$ J/m^3/°C , Chaleur volumique
$\lambda = 1$ W/m/°C , Conductivité thermique de Ω
$k = 0.2$ W/m^2/°C , Coefficient de radiation
L = 1 m, Demi-côté de la plaque

Les coordonnées du centre c_1 (respectivement c_2) du sous-domaine Ω_1 (respectivement Ω_2) est (L/2, -L/2) (respectivement (-L/2, L/2)) et son rayon $r_1 = 0.2$ m (respectivement $r_2 = 0.2$ m).

L'arc de cercle du coin arrondi entrant dans Ω a pour rayon r = 0.1 m.

2/ Solution réalisé avec la boite à outils PDE de Matlab

Pour réaliser cette étude, nous allons écrire 5 programmes Matlab permettant de réaliser une simulation numérique de ce problème de transfert de chaleur à l'aide de la boite à outils PDE de Matlab.

Les différents programmes sont :

- tfranvil.m, le programme principal qui exécute la simulation à l'aide des 4 autres programmes et affichent les résultats sous forme de données et de représentations graphiques animées.
- tfranvil_g.m, le programme définissant la géométrie du problème, à savoir la plaque en forme de L, et les différents domaines de l'étude, le milieu extérieur et le domaine Ω (la plaque) constitué de 3 sous-domaines (Ω_1, Ω_2 et Ω_3).
- tfranvil_b.m, le programme qui définit les conditions aux limites, c'est-à-dire les conditions sur les frontières entre le domaine extérieur et la plaque (le domaine Ω).
- tfranvil_f.m, le programme spécifiant les termes sources indépendant du temps des 3 sous-domaines de la plaque (Ω_1, Ω_2 et Ω_3).
- Playsol.m, le programme qui permet d'animer la représentation graphique de notre simulation numérique pour voir en temps réel l'évolution de la chaleur dans la plaque.

2.1/ tfranvil.m

C'est le programme principal qui exécute notre simulation numérique.

On va tout d'abord tracer dans la fenêtre n°1 la géométrie de la plaque en utilisant le programme tfranvil_g.m et la fonction pdegplot de matlab.

```
figure(1);

pdegplot('tfranvil_G'),axis equal % Trace la géométrie
```

On va ensuite créer le maillage à l'aide de la fonction matlab initmesh qui crée
trois matrices permettant de définir le maillage de notre géométrie.
A l'aide de la fonction pdemesh de matlab, on va tracer la géométrie et le maillage dans une seconde fenêtre.

```
[p,e,u] = initmesh('tfranvil_G'); % Intialise le maillage du problème
                                  % p, matrice des points du maillage
                                  % e, segments aux bords
                                  % u, numéro des sommets d'un triangle +
                                  %    le domaine

figure(2);

pdemesh(p,e,u),axis equal % Trace le maillage du problème + la géométrie
```

On peut utiliser la fonction refinemesh de matlab pour raffiner plus notre maillage. Ici, on raffinera 2 fois notre maillage pour augmenter le nombre de triangles dans le maillage et donc la précision de la simulation.

```
figure(3);

[p,e,u] = refinemesh('tfranvil_G',p,e,u); % Raffine le maillage existant
[p,e,u] = refinemesh('tfranvil_G',p,e,u); % Re-raffine le maillage
existant

pdemesh(p,e,u),axis equal
```

On va maintenant afficher le régime permanent obtenu. Pour se faire, on va considérer le régime comme permanent, ie la variation de la température au cours du temps est nulle.
On obtient donc l'équation simplifiée :

$$-\lambda\, div(\nabla T) = F$$

que l'on résoud en faisant appel à la fonction assempde de matlab (fonction résolvant les problèmes de la forme -div(c*grad(u))+a*u=f , voir l'aide MATLAB pour plus de détails).

```
%Régime permanent
%On reprend l'équation du problème en supposant le régime permanent, ie
%   la variation de la température au cours du temps est nulle.

c = 1;
a = 0;

tperm = assempde('tfranvil_B',p,e,u,c,a,'tfranvil_F');
```

On va ensuite calculer la solution du problème en régime transitoire en appelant la fonction parabolic de matlab (fonction résolvant les problèmes de la forme d*du/dt-div(c*grad(u))+a*u=f , voir l'aide MATLAB pour plus de détails).
linspace est une fonction générant une liste de nbiter points équitablement répartis sur le segment [0, tempsfinal].

```
%Régime transitoire

nbiter = 1001;
tempsfinal = 20;

t0 = 0;
ulist = linspace(0,tempsfinal,nbiter);
d = 1;
tempscpu = cputime;
t  = parabolic(t0,ulist,'tfranvil_B',p,e,u,c,a,'tfranvil_F',d);
display('Tempscpu');
cputime - tempscpu
```

Il ne reste alors plus qu'à déterminer le temps t à partir duquel le regime est considéré comme permanent et jouer l'animation de la simulation numérique.

Pour se faire, on compare la température à chaque temps t de notre ulist avec la température de la plaque obtenue pour le calcul du régime permanent.

```
%Calcul de la valeur ti à partir de laquelle le régime est considéré
%   permanent

i = 1;
epsilon = 0.001;

while ((i<(nbiter+1))&&((norm(t(:,i)-tperm)/norm(tperm)) > epsilon))
    i = i+1;
end

display('Température t à partir de laquelle le régime est considéré
permanent');

if i == nbiter+1
    display ('Le régime permanent n est pas atteint avec notre échelle de
temps')
else
    ulist(i)
end
```

Enfin, on appelle la fonction Playsol.m pour jouer la simulation dans une cinquième fenêtre

```
%Représentation du régime transitoire

figure(5);

PlaySol(t,ulist,p,e,u,tmin,tmax);
```

2.2/ tfranvil_g.m

Ce programme doit définir la géométrie du problème et préciser ses différents domaines.

Tout d'abord, on crée une matrice 4*n avec n le nombre de segments nécessaires à la représentation du problème. Ici, notre plaque en forme de L va nécessiter 6 segments et un arc de cercle pour être représenté, auquel il faut encore ajouter 4 demi-cercles pour représenter les 2 cercles définissant 2 sous-domaines dans la plaque (on ne peut définir un cercle avec un unique segment avec matlab).

Ainsi n = 11, notre matrice comprendra 11 colonnes pour les 11 segements nécessaires à la réprésentation graphique du problème.

Les 2 première lignes permettent de préciser l'abscisse curviligne de départ et de fin des segments et les 2 dernières indiquent quels domaines se situent à gauche et à droite du segment.

```
% L'orientation des segments au bord est obtenue, dans le cas présent,
% en les parcourant dans le sens des aiguilles d'une montre et
% inersement pour les deux cercles internes.
d=[
  0 0 0 0 0 0    0 0  pi   0  pi   % abscisse curviligne de départ
  1 2 2 1 1 pi/2 1 pi 2*pi pi 2*pi % abscisse curviligne de fin
  0 0 0 0 0 0    0 2  2    1  1    % label du sous-domaine à gauche
  3 3 3 3 3 3    3 3  3    3  3    % label du sous-domaine à droite
];
```

Cette matrice permet donc de parcourir les segments et de localiser spatialement les différents domaines du problèmes. Il ne reste plus qu'à coder les segments comme pour les 2 exemples :

```
% segment 1:  Bord gauche partie supérieure.
  js=1;  ii=find(bs==js);
 if length(ii)
     x(ii)=interp1([d(1,js),d(2,js)],[-1 -1],s(ii));
     y(ii)=interp1([d(1,js),d(2,js)],[ 0  1],s(ii));
  end
```

```
% segment 8:  Demi-cercle droit du Cercle intérieur partie supérieure
gauche.
  js=8;  ii=find(bs==js);
  if length(ii)
     x(ii)=-0.5 + 0.2 * cos(s(ii));
     y(ii)= 0.5 + 0.2 * sin(s(ii));
  end
```

2.3/ tfranvil_b.m

Ce programme définit les conditions aux limites du problème, c'est-à-dire les conditions sur la frontière entre la plaque et le milieu extérieur.

On va maintenant associer à chaque segment au bord du domaine (ie les segments dont un des domaines sur sa gauche ou sur sa droite est le le domaine extérieur : c'est à dire les segments numéro 1 à 7) une condition aux limites :

```
cl(1:l2, 1) = b2;  % Bord gauche partie supérieure : C.L. de type "b2"
cl(1:l1, 2) = b1;  % Bord supérieur                : C.L. de type "b1"
cl(1:l1, 3) = b1;  % Bord droit                    : C.L. de type "b1"
cl(1:l3, 4) = b3;  % Bord inférieure partie droite : C.L. de type "b3"
cl(1:l1, 5) = b1;  % Bord gauche partie inférieure : C.L. de type "b1"
cl(1:l1, 6) = b1;  % Quart de cercle extérieur     : C.L. de type "b1"
cl(1:l1, 7) = b1;  % Bord inférieure partie gauche : C.L. de type "b1"
```

Il ne reste plus qu'à définir les 3 types de conditions aux limites.

La condition de type b1 est une condition de Neumann homogènes, il n'y a pas de transfert de chaleur selon les frontières Γ_1 et Γ_3

```
% Descripteur de conditions aux limites de Neumann homogènes : dt/dn = 0
b1 = [   ...
 1       ... % dimension N of the system
 0       ... % number M of Dirichlet boundary conditions
 1       ... % length for the string representing q
 1       ... % length for the string representing g
 '0'     ... % scalar function q
 '0'     ... % scalar function g
]'*1;  l1 = length(b1);
```

La condition de type b2 est une condition de radiation ie de Neumann inhomogènes. De la chaleur s'evacue sur la frontière Γ_2

```
% Descripteur de conditions aux limites de Neumann inhomogènes : dt/dn =
%-0.2*t - 20
b2 = [   ...
 1       ... % dimension N of the system
 0       ... % number M of Dirichlet boundary conditions
 3       ... % length for the string representing q
 3       ... % length for the string representing g
 '0.2'     ... % scalar function q
 '-20'   ... % scalar function g
]'*1;  l2 = length(b2);
```

La condition de type b3 est une condition de Dirichlet.
De la chaleur arrive de la frontière Γ_4.

```
% Descripteur de CL de type Dirichlet inhomogènes: t = 100
b3 = [   ...
 1       ... % dimension N of the system
 1       ... % number M of Dirichlet boundary conditions
 1       ... % length for the string representing q
 1       ... % length for the string representing g
 1       ... % length for the string representing h
 3       ... % length for the string representing r
 '0'     ... % scalar function q
 '0'     ... % scalar function g
 '1'     ... % scalar function h
 '100'   ... % scalar function r
]'*1;  l3 = length(b3);
```

2.4/ tfranvil_f.m

Ce programme précise les termes sources (indépendants du temps) dans les différents sous-domaines du domaine Ω.

On va tout d'abord générer le vecteur nulle de taille le nombre de triangles dans le maillage à l'aide de la fonction pdeintrp.

```
 %Création du vecteur nulle de taille le nombre de triangles dans le
 %   maillage

 f  = pdeintrp(p,u,fp(:));
```

Le domaine Ω_3 a pour terme source 0. Les domaines Ω_1 et Ω_2 ont pour termes sources -2000 et 600.

Il ne reste donc qu'à parcourir ce vecteur pour remplacer les termes sources des triangles appartenant aux domaines Ω_1 et Ω_2 .Pour se faire, on n'a qu'à parcourir la matrice u donnant le domaine des triangles et tester la valeur du domaine :

```
%Correction des valeurs du vecteur f si les traingles appartiennent aux
%   domaines 1 ou 2

for i = 1:nu
    if u(4,i) == 1
        f(i) = -2000;
    elseif u(4,i) == 2
        f(i) = 600;
    end
end
```

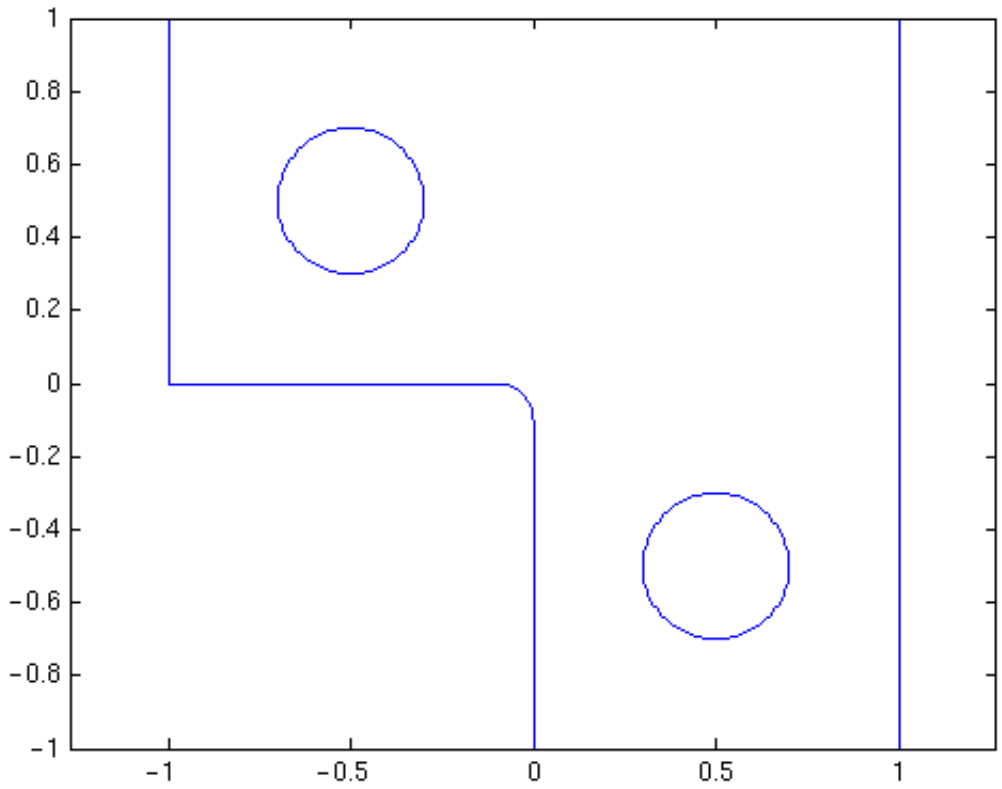
2.5/ Playsol.m

Ce dernier programme permet de jouer l'animation de la simulation numérique pour voir l'évolution de la chaleur dans la plaque au cours du temps.

3/ Résultats

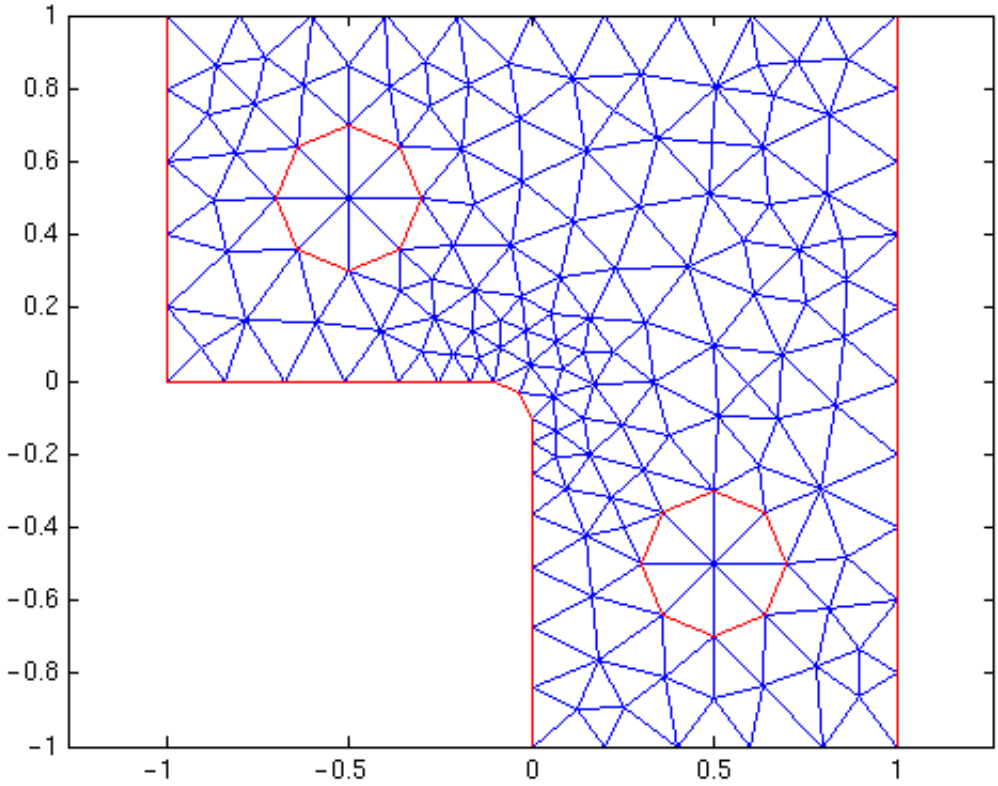
3.1/ Images

3.1.1/ Géométrie

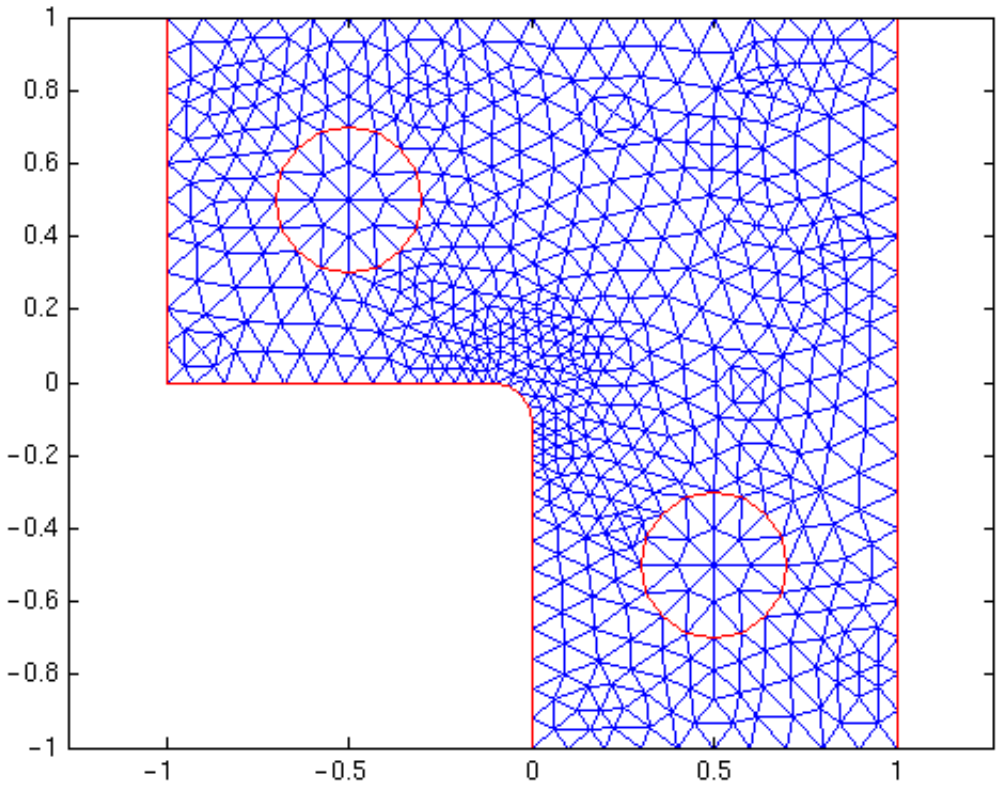


3.1.2/ Maillages

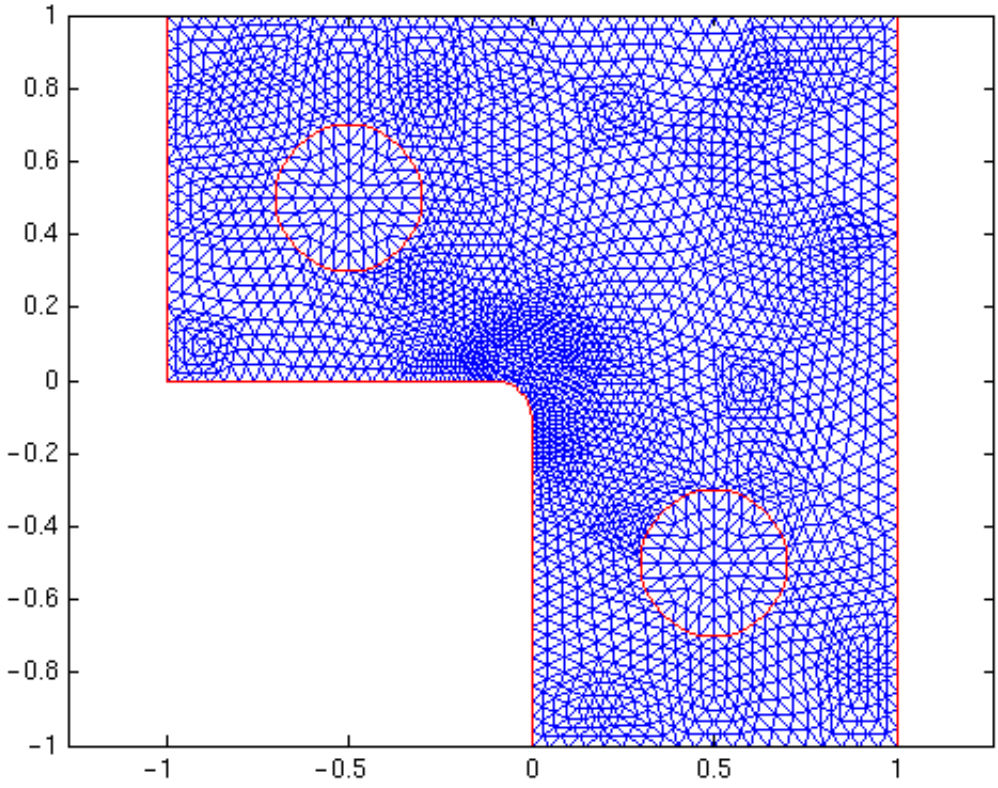
Maillage pour 0 raffinage :



Maillage pour 1 raffinage :

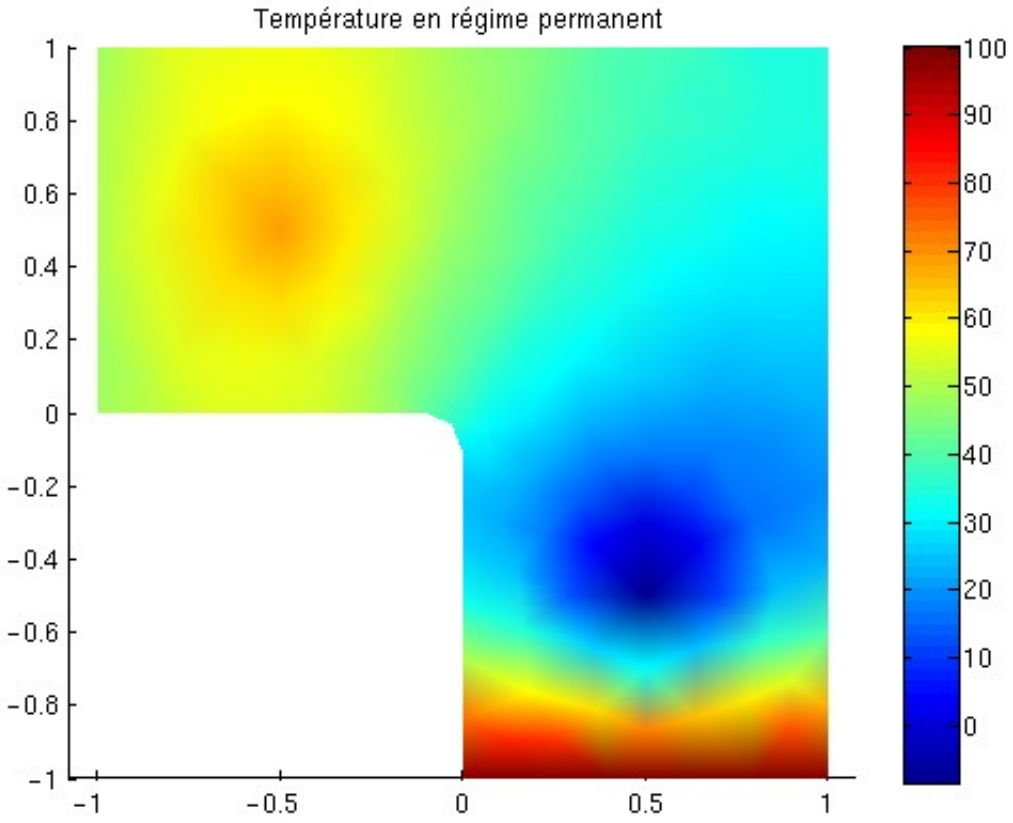


Maillage pour 2 raffinages :

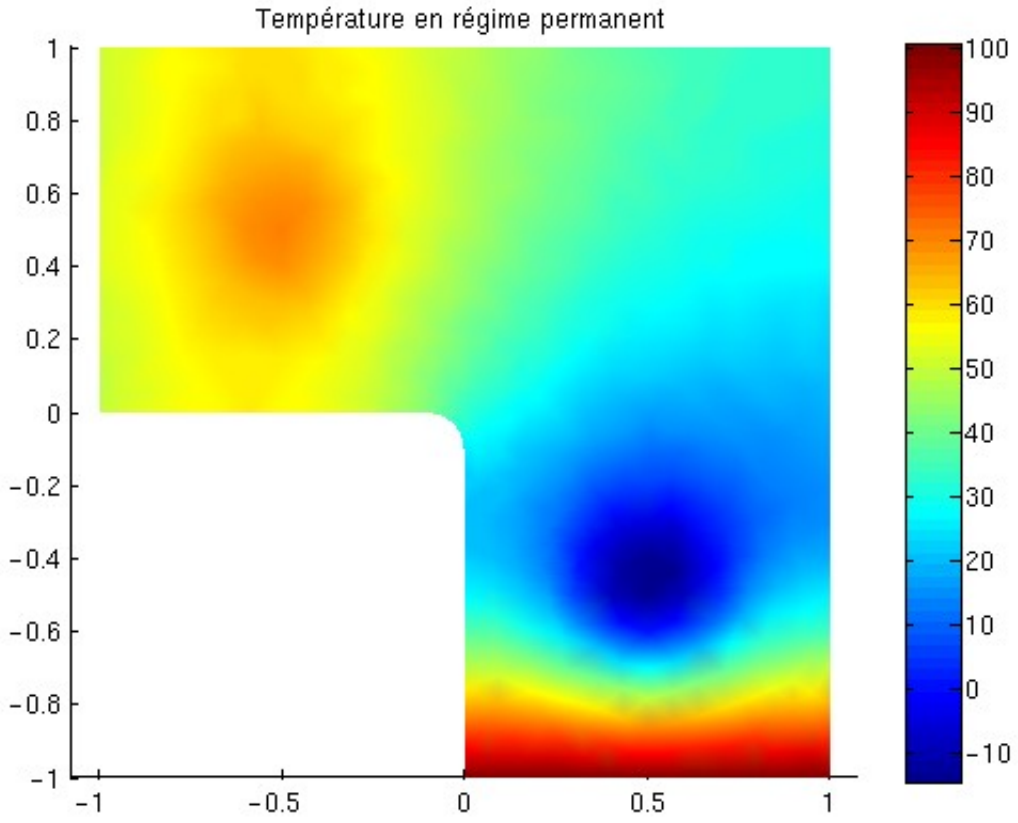


3.1.3/ Régimes permanents

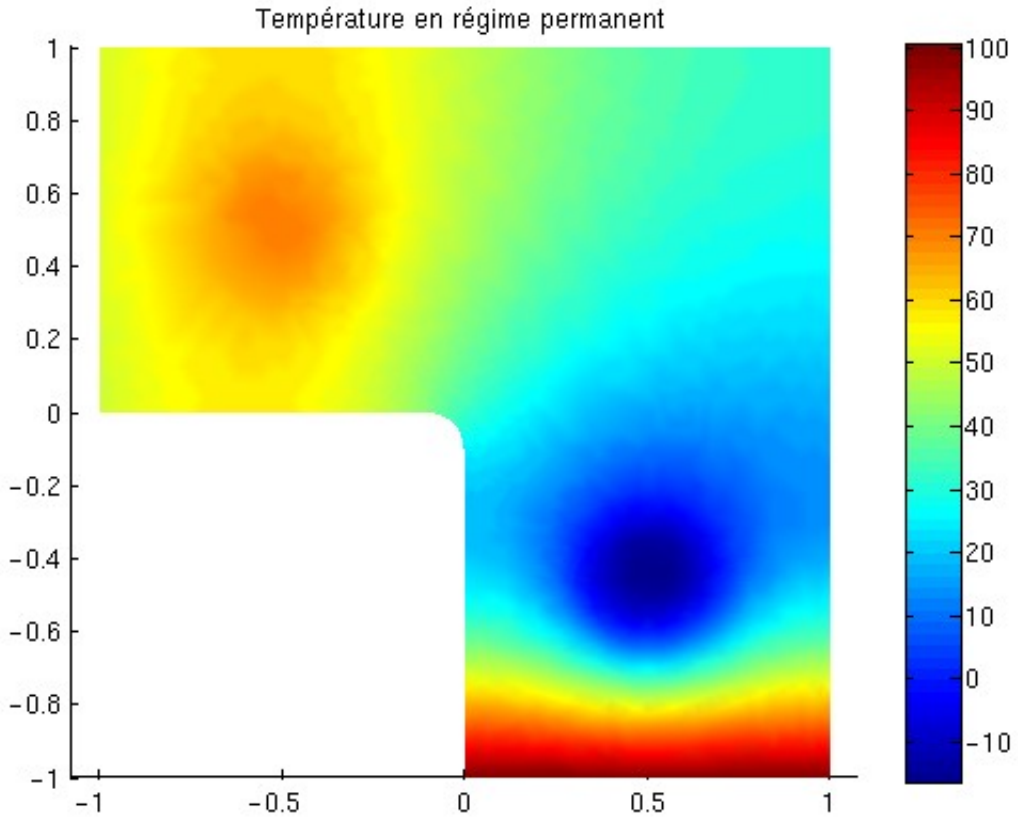
Régime permanent pour 0 raffinage :



Régime permanent pour 1 raffinage :



Régime permanent pour 2 raffinages :



3.2/ Tests

On va tester notre simulation pour différentes tailles de maillage (ie différents nombre de raffinage) et différents pas de discrétisation de notre intervalle de temps du régime transitoire. Les résultats seront affichés sous forme de tableau (avec sur les lignes le nombre de raffinages et sur les colonnes le nombre d'échantillons pris sur le segment [0, tmax]):

Affichage du temps de travail :

Tableau pour tmax = 20

	21 (pas de 1)	101 (pas de 0.2)	1001 (pas de 0.02)
0	temps_cpu = 1.6700	temps_cpu = 1.6600	temps_cpu = 2.1200
1	temps_cpu = 4.6600	temps_cpu = 4.0700	temps_cpu = 5.2500
2	temps_cpu = 17.8200	temps_cpu = 17.9400	temps_cpu = 22.4000

Tableau pour tmax = 100

	21 (pas de 5)	101 (pas de 1)	1001 (pas de 0.1)
0	temps_cpu = 1.8000	temps_cpu = 1.7200	temps_cpu = 2.2700
1	temps_cpu = 4.3400	temps_cpu = 4.3900	temps_cpu = 5.5400
2	temps_cpu = 19.2900	temps_cpu = 19.2700	temps_cpu = 24.3900

La source principale d'augmentation du temps de calcul provient du nombre de raffinages choisis.

Affichage du temps t où on atteint le régime permanent :

Tableau pour tmax = 20

	21 (pas de 1)	101 (pas de 0.2)	1001 (pas de 0.02)
0	Tperm = 16	Tperm = 15.8000	Tperm = 15.6600
1	Tperm = 16	Tperm = 15.8000	Tperm = 15.7600
2	Tperm = 16	Tperm = 15.8000	Tperm = 15.7800

Tableau pour tmax = 100

	21 (pas de 5)	101 (pas de 1)	1001 (pas de 0.1)
0	Tperm = 20	Tperm = 16	Tperm = 15.7000
1	Tperm = 20	Tperm = 16	Tperm = 15.8000
2	Tperm = 20	Tperm = 16	Tperm = 15.8000

Augmenter le nombre de raffinages et/ou diminuer le pas d'échantillonnage améliore la précision du calcul de l'instant t où le régime permanent est atteind.

Affichage du nombre total de résolutions de système linéaires :

Tableau pour tmax = 20

	21 (pas de 1)	101 (pas de 0.2)	1001 (pas de 0.02)
0	385	385	385
1	424	424	424
2	496	496	496

Tableau pour tmax = 100

	21 (pas de 5)	101 (pas de 1)	1001 (pas de 0.1)
0	405	405	405
1	447	447	447
2	527	527	527

Le nombre de résolutions de sytème linéaires augmentent donc avec le nombre de raffinages choisies.

4/ Codes complets

4.1/ tfranvil.m

```
figure(1);

pdegplot('tfranvil_G'),axis equal % Trace la géométrie

[p,e,u] = initmesh('tfranvil_G'); % Intialise le maillage du problème
                                  % p, matrice des points du maillage
                                  % e, segments aux bords
                                  % u, numéro des sommets d'un triangle +
                                  %    le domaine

figure(2);

pdemesh(p,e,u),axis equal % Trace le maillage du problème + la géométrie

figure(3);

[p,e,u] = refinemesh('tfranvil_G',p,e,u); % Raffine le maillage existant
[p,e,u] = refinemesh('tfranvil_G',p,e,u); % Re-raffine le maillage existant

pdemesh(p,e,u),axis equal

%Régime permanent
%On reprend l'équation du problème en supposant le régime permanent, ie
%   la variation de la température au cours du temps est nulle.

c = 1;
a = 0;

tperm = assempde('tfranvil_B',p,e,u,c,a,'tfranvil_F');

tmin = min(tperm);
tmax = max(tperm);
```

```
Titre = 'Température en régime permanent';

figure(4);

pdeplot(p,e,u,'xydata',tperm,'title',Titre,'colormap','jet','mesh','off','contou
r','off','levels',20), axis equal
caxis([tmin tmax]);
colorbar;

%Régime transitoire

nbiter = 1001;
tempsfinal = 20;

t0 = 0;
ulist = linspace(0,tempsfinal,nbiter);
d = 1;
tempscpu = cputime;
t  = parabolic(t0,ulist,'tfranvil_B',p,e,u,c,a,'tfranvil_F',d);
display('Tempscpu');
cputime - tempscpu

%Calcul de la valeur ti à partir de laquelle le régime est considéré
%   permanent

i = 1;
epsilon = 0.001;

while ((i<(nbiter+1))&&((norm(t(:,i)-tperm)/norm(tperm)) > epsilon))
    i = i+1;
end

display('Température t à partir de laquelle le régime est considéré permanent');

if i == nbiter+1
    display ('Le régime permanent n est pas atteint avec notre échelle de
temps')
else
    ulist(i)
end

%Représentation du régime transitoire

figure(5);

PlaySol(t,ulist,p,e,u,tmin,tmax);
```

4.2/ tfranvil_g.m

```
function [x,y]=tfranvil_G(bs,s)
%
% Exécutable MATLAB décrivant la géométrie du problème d'EDP simulant
%   le transfert de chaleur à travers une plaque mince en
%   forme de L inversé et composée de 3 sous-domaines.
%
%
% La description de la géométrie se fait en trois temps, spécifiés par des
% paramètres d'entrée et de sortie en nombre différents:
%
%   NE=tfranvil_G retourne le nombre de segments frontière.
%
%   D=tfranvil_G(BS) retourne une matrice avec un vecteur colonne décrivant
%   chacun des segments frontière indiqué dans le tableau BS en entrée.
%   Les informations contenues dans chacune de ces colonnes sont:
```

```
%     En ligne 1, l'abscisse curviligne de départ sur le segment considéré.
%     En ligne 2, l'abscisse curviligne de fin sur le segment considéré.
%     En ligne 3, le label numérique (0 pour l'extérieur au domaine)
%                 du sous domaine à gauche du segment orienté par les
%                 deux paramètres précédents.
%     En ligne 4, le label numérique (0 pour l'extérieur au domaine)
%                 du sous domaine à droite de ce même segment orienté.
%     Rque: Un segment est au bord du domaine si il a, à sa gauche ou
%           à sa droite le domaine extérieur "0".
%           Il peut aussi y avoir des segments internes au domaine,
%           qui délimitent la frontières entre deux sous-domaines
%           différents dans lesquels les équations diffèrent quelque peu.
%
%   [X,Y]=tfranvil_G(BS,S) retourne les coordonnées d'un ensemble de points
%   sur ces segments frontière. Les points en question sont définis en
%   entrée, d'une part par le tableau BS qui specifie pour chaque point
%   le label numérique du segment auquel il appartient et, d'autre part,
%   par les abscisses curvilignes contenues dans le tableau S et qui
%   positionnent chaque point de manière unique sur chacun des segments
%   en question.

nbs=11;

if nargin==0,
  x=nbs; % nombre de segments frontière au total.
  return
end

% L'orientation des segments au bord est obtenue, dans le cas présent,
% en les parcourant dans le sens des aiguilles d'une montre et
% inersement pour les deux cercles internes.
d=[
  0 0 0 0 0 0    0 0  pi   0  pi   % abscisse curviligne de départ
  1 2 2 1 1 pi/2 1 pi 2*pi pi 2*pi % abscisse curviligne de fin
  0 0 0 0 0 0    0 2  2    1  1    % label du sous-domaine à gauche
  3 3 3 3 3 3    3 3  3    3  3    % label du sous-domaine à droite
];

% La ligne suivante force la mise sous forme vecteur ligne
% du vecteur bs en entrée
bs1=bs(:)';

if find(bs1<1 | bs1>nbs),
  error('Non existent boundary segment number')
end

if nargin==1,
  x=d(:,bs1);
  return
end

x=zeros(size(s));
y=zeros(size(s));
[m,n]=size(bs);
if m==1 && n==1,
  bs=bs*ones(size(s)); % dans le cas ou on n'a qu'un seul label de segment
                       % en entrée, c'est à dire que "bs" est un scalaire,
                       % on duplique ce scalaire autant de fois que de points
                       % attendus en sortie (nombre d'abscisses dans "s").
elseif m~=size(s,1) || n~=size(s,2),
  error('bs doit être un scalaire ou avoir la même taille que s');
end
```

```
% Enfin, calcul des coordonnées des points sur chacun des segments
% en question en fonction des abscisses curvilignes contenues dans
% le tableau "s":
if ~isempty(s),

% segment 1:  Bord gauche partie supérieure.
  js=1;  ii=find(bs==js);
 if length(ii)
     x(ii)=interp1([d(1,js),d(2,js)],[-1 -1],s(ii));
     y(ii)=interp1([d(1,js),d(2,js)],[ 0  1],s(ii));
  end

% segment 2:  Bord supérieur.
  js=2;  ii=find(bs==js);
  if length(ii)
     x(ii)=interp1([d(1,js),d(2,js)],[-1  1],s(ii));
     y(ii)=interp1([d(1,js),d(2,js)],[ 1  1],s(ii));
  end

% segment 3:  Bord droit.
  js=3;  ii=find(bs==js);
  if length(ii)
     x(ii)=interp1([d(1,js),d(2,js)],[ 1  1],s(ii));
     y(ii)=interp1([d(1,js),d(2,js)],[ 1 -1],s(ii));
  end

% segment 4:  Bord inférieur partie droite.
  js=4;  ii=find(bs==js);
  if length(ii)
     x(ii)=interp1([d(1,js),d(2,js)],[ 1  0],s(ii));
     y(ii)=interp1([d(1,js),d(2,js)],[-1 -1],s(ii));
  end

% segment 5:  Bord gauche partie inférieure.
  js=5;  ii=find(bs==js);
  if length(ii)
     x(ii)=interp1([d(1,js),d(2,js)],[ 0  0],s(ii));
     y(ii)=interp1([d(1,js),d(2,js)],[-1 -0.1],s(ii));
  end

% segment 6:  Quart de cercle extérieur.
  js=6;  ii=find(bs==js);
  if length(ii)
     x(ii)=-0.1 + 0.1 * cos(s(ii));
     y(ii)=-0.1 + 0.1 * sin(s(ii));
  end

% segment 7:  Bord inférieur partie gauche.
  js=7;  ii=find(bs==js);
  if length(ii)
     x(ii)=interp1([d(1,js),d(2,js)],[-0.1 -1],s(ii));
     y(ii)=interp1([d(1,js),d(2,js)],[ 0  0],s(ii));
  end

  % segment 8:  Demi-cercle droit du Cercle intérieur partie supérieure gauche.
  js=8;  ii=find(bs==js);
  if length(ii)
     x(ii)=-0.5 + 0.2 * cos(s(ii));
     y(ii)= 0.5 + 0.2 * sin(s(ii));
  end

  % segment 9:  Demi-cercle gauche du Cercle intérieur partie supérieure gauche.
  js=9;  ii=find(bs==js);
  if length(ii)
```

```
    x(ii)=-0.5 + 0.2 * cos(s(ii));
    y(ii)= 0.5 + 0.2 * sin(s(ii));
  end

  % segment 10:  Demi-cercle droit du Cercle intérieur partie inférieure droite.
  js=10;  ii=find(bs==js);
  if length(ii)
    x(ii)= 0.5 + 0.2 * cos(s(ii));
    y(ii)=-0.5 + 0.2 * sin(s(ii));
  end

  % segment 11:  Demi-cercle droit du Cercle intérieur partie inférieure droite.
  js=11;  ii=find(bs==js);
  if length(ii)
    x(ii)= 0.5 + 0.2 * cos(s(ii));
    y(ii)=-0.5 + 0.2 * sin(s(ii));
  end

end
```

4.3/ tfranvil_b.m

```
function [q,g,h,r]=tfranvil_B(p,e,t,time)
%
% Exécutable MATLAB spécifiant les conditions aux limites pour le problème
%   d'EDP simulant le transfert de chaleur à travers une plaque mince en
%   forme de L inversé et composée de 3 sous-domaines.
%

% Nombre de segments au bord du domaine:
nbs = 7;             %

% Descripteur de conditions aux limites de Neumann homogènes : dt/dn = 0
b1 = [   ...
 1       ... % dimension N of the system
 0       ... % number M of Dirichlet boundary conditions
 1       ... % length for the string representing q
 1       ... % length for the string representing g
 '0'     ... % scalar function q
 '0'     ... % scalar function g
]'*1;  l1 = length(b1);

% Descripteur de conditions aux limites de Neumann inhomogènes : dt/dn =
% -0.2*t - 20
b2 = [   ...
 1       ... % dimension N of the system
 0       ... % number M of Dirichlet boundary conditions
 3       ... % length for the string representing q
 3       ... % length for the string representing g
 '0.2'     ... % scalar function q
 '-20'   ... % scalar function g
]'*1;  l2 = length(b2);

% Descripteur de CL de type Dirichlet inhomogènes: t = 100
b3 = [   ...
 1       ... % dimension N of the system
 1       ... % number M of Dirichlet boundary conditions
 1       ... % length for the string representing q
 1       ... % length for the string representing g
 1       ... % length for the string representing h
 3       ... % length for the string representing r
 '0'     ... % scalar function q
 '0'     ... % scalar function g
 '1'     ... % scalar function h
```

```
 '100'   ... % scalar function r
]'*1;  l3 = length(b3);

lb = max([ l1 l2 l3 ]);  cl = blanks(lb)'*ones(1,nbs);
cl(1:l2, 1) = b2;  % Bord gauche partie supérieure : C.L. de type "b2"
cl(1:l1, 2) = b1;  % Bord supérieur               : C.L. de type "b1"
cl(1:l1, 3) = b1;  % Bord droit                  : C.L. de type "b1"
cl(1:l3, 4) = b3;  % Bord inférieure partie droite : C.L. de type "b3"
cl(1:l1, 5) = b1;  % Bord gauche partie inférieure : C.L. de type "b1"
cl(1:l1, 6) = b1;  % Quart de cercle extérieur     : C.L. de type "b1"
cl(1:l1, 7) = b1;  % Bord inférieure partie gauche : C.L. de type "b1"

if any(size(t))
  [q,g,h,r]=pdeexpd(p,e,t,time,cl);
else
  [q,g,h,r]=pdeexpd(p,e,time,cl);
end
```

4.4/ tfranvil_f.m

```
function f=tfranvil_F(p,u,t,time)

%
% Exécutable MATLAB spécifiant le terme source (indépendant du temps)
%   pour le problème d'EDP simulant le transfert de chaleur à travers
%   une plaque mince en forme de L inversé et composée de 3 sous-domaines.
%

 np = size(p,2);
 fp = zeros(np,1);

 %Création du vecteur nulle de taille le nombre de triangles dans le
 %   maillage

 f  = pdeintrp(p,u,fp(:));

 nu = size(u,2);

 %Correction des valeurs du vecteur f si les traingles appartiennent aux
 %   domaines 1 ou 2

 for i = 1:nu
     if u(4,i) == 1
         f(i) = -2000;
     elseif u(4,i) == 2
         f(i) = 600;
     end
 end
```

4.5/ Playsol.m

```
function [F,fps] = PlaySol(U,tlist,p,e,t,tmin,tmax)

% Plot solutions :
fig = figure(5); clf;
set(fig, 'DoubleBuffer', 'on');

ndt = length(tlist);
for j=1:ndt
   Titre = [ 'Température au temps ' num2str(tlist(j)) ];
   pdeplot(p,e,t,'xydata',U(:,j), 'title', Titre, 'colormap','jet', ...
           'mesh','off','contour','off','levels',20), axis equal
   caxis([tmin tmax]); colorbar;
   F(j) = getframe(gcf);
```

```
end

fps = ndt/tlist(ndt);
movie(F,1,fps,[-70 -45 0 0])
```