A 33 line heat transfer finite element code

Pierre Beckers[a] & Benoit Beckers[b]

[a]University of Liège, Belgium
[b]Roberval Laboratory – Urban Systems Engineering Department, Compiègne University of Technology, Compiègne, France

To clarify and to develop some concepts presented in [Beckers & Beckers 2015][1], we present here some easy to manage additional procedures.

Table 1: Procedure for solving the first part of the first application of reference[1]

This procedure allows computing a standard cinematically admissible solution based on a temperature model. The procedure itself is completed in 22 lines. Eleven additional lines are dedicated to the sketch of the results.

```
clear all;tstart=tic;k=100;t=0.01 ;qin=1000;pa=.02;             % Input data
Kca=k*t/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4];% Element conduct.
ny=16;if ny>80;ny=80;end;if ny<2;ny=2;end;nx=ny*2;% Definition of the mesh
nel=nx*ny;Ntca=(ny+1)*(nx-1);% Numbers of elements & DOF of the C.A. model
loK=zeros(nel,4);my=ny+1;                            % localization vectors
for i=1:nx;  % nodes & elements are numbered: bottom -> top, left -> right
    for j=1:ny;
        loK((i-1)*ny+j,1)  = my*(i-1)+j;
        loK((i-1)*ny+j,2)  = loK((i-1)*ny+j,1)+my;
        loK((i-1)*ny+j,3)  = loK((i-1)*ny+j,2)+1;
        loK((i-1)*ny+j,4)  = loK((i-1)*ny+j,1)+1;
    end
end                                                  % End localization vectors
K=zeros((nx+1)*(ny+1),(nx+1)*(ny+1));       % Full K matrix, initialization
for n=1:nel; for i=1:4; for j=1:4;             % Assembling structural K
K(loK(n,i),loK(n,j))=K(loK(n,i),loK(n,j))+Kca(i,j);end;end;end
Ksca=K(ny+2:(nx+1)*(ny+1)-ny-1,ny+2:(nx+1)*(ny+1)-ny-1);%sub mat to invert
gca=zeros(Ntca,1);                                        % Second member
for i = 1:my:(ny-2)*my+1; gca(i)=2*qin/nx;gca(i+ny)=2*qin/nx;end
gca((ny-1)*my+1)=qin/nx;gca((ny-1)*my+1+ny)=qin/nx;         % End 2d member
tca    = Ksca\gca;                                              % Solution
disca = gca'*tca/2;disp([' D  = ',num2str(disca,'%0.6g')])    % dissipation
figure('Position',[1 1 1024 512]);                   % Drawing the Isotherms
ii=0;B=zeros(my,nx+1);x=zeros(my,nx+1);y=zeros(my,nx+1); % Initializations
for j=2:nx+1;for i=1:my;ii=ii+1;if ii < my*(nx-1)+1;B(i,j)=tca(ii);end
x(i,j)=(j-1)*2/nx;y(i,j)=(i-1)*2/nx;end;end;telapsed=toc(tstart);
B(:,1)=0;B(:,nx+1)=0;y(:,1)=y(:,2);gap=pa*qin/(k*t);
% br56;colormap(br56);
[CS,H]=contour(x,y,B,(0.:gap:max(tca)),'b');hold on;
clabel(CS,H,[100 200 300 400 500 600 700]); % End of isotherms drawing
title(['ny = ',num2str(ny),', D = ',num2str(disca,'%0.6g'),', cpu = ',...
num2str(telapsed,'%0.2g'),', Iso-T, from 0., steps = ', num2str(gap),...
', to Tmax = ',num2str(max(tca),'%0.3g')],'fontsize', 15); % End =========
```

[1] Pierre Beckers, Benoit Beckers, “A 66 line heat transfer finite element code to highlight the dual approach”, *Computers & Mathematics with Applications*, Volume 70 Issue 10, November 2015, Pages 2401-2413.

The procedure listed in *Table 1* corresponds to the first part of the one presented in [Beckers & Beckers 2015] and listed in the *Table 7* of this document. Running it, we obtain the result of *Figure 1*. The results of the tests presented in this report are fully consistent with those of the reference[1].

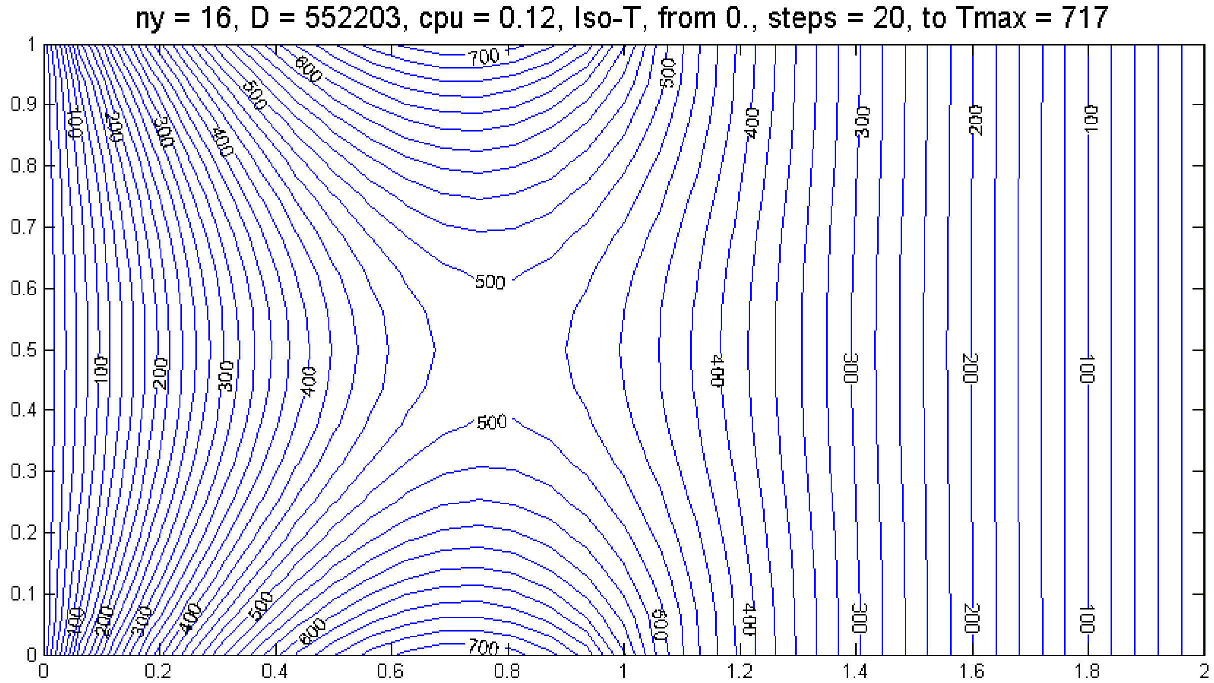


Figure 1: Isothermal curves showing the temperature field

Description of the procedure

Line 01: physical data and loads.
Line 02: definition of the 4 x 4 symmetric conductivity matrix.
Line 03: definition of the mesh according to the number of element in the *y* direction. The maximum number of elements in this direction is equal to 80, so as to give a 80 x 160 = 12800 elements.
Line 04: computation of the number of elements and the number of degrees of freedom (*D.O.F.*), taking into account the fixation of both vertical boundaries where the temperature is assumed to be equal to zero..
Lines 05 – 13: initializations and definition of the localization matrix, 12800 x 4.
Lines 14 – 16: initialization and construction of the global conductivity matrix.
Line 17: definition of the non singular sub matrix corresponding to the free *D.O.F.*
Lines 18 – 20: initialization and construction of the second member: generalized heat fluxes.
Line 21: solving the system of equation.
Line 22: computing and displaying the dissipation energy. For **80 elements** in the vertical direction, we obtain, with the data of the procedure, a dissipation energy equal to **554213 W K** and a maximum temperature of **718°**. The number of *D.O.F.* is equal to **12879** and on a standard laptop; the cpu time is equal to **36 seconds**.
Lines 23 – 33: plot of the level curves: isothermal curves.

Line 24: initializations.

Lines 25 – 26: construction of the 3 matrices containing the two coordinates *x*, *y* and the temperature *B* for each nodal position.

Line 27: adding the values corresponding to the boundaries, computing the elapsed time and the step between two successive levels of the contour lines.

Line 28: modifying the color map of Matlab© to avoid the dark brown of the upper part

Lines 29 – 30: drawing the contour lines.

Lines 31 – 33: displaying a title on the top of the figure.

The replacement of function “*contour*” of line 29 by function “*contourf*” in *Table 1* leads to the result of *Figure 2*. It is the same as result of *Figure 1*, but the level or height of the temperature field is superposed to make it more sensitive.

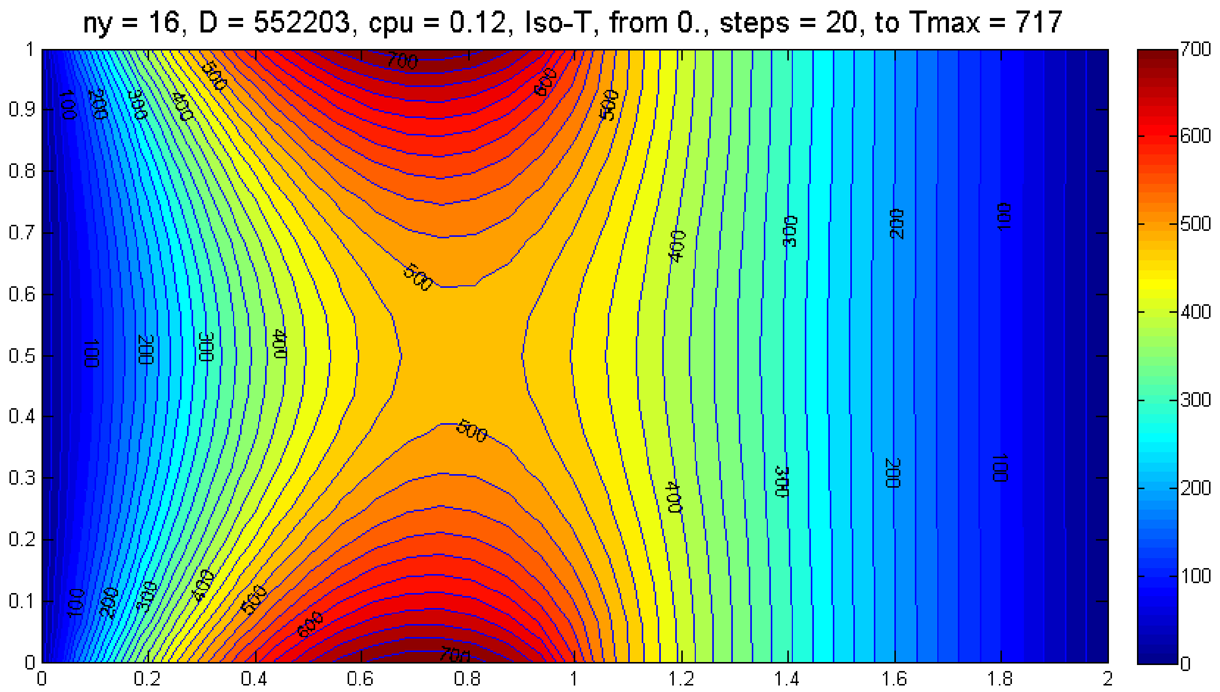


Figure 2: Isothermal curves filled with the Matlab© default color map

The dark brown part of the *Figure 1* color map is removed by activating the line 28 of *Table 1* and introducing in the directory of the Matlab© procedure the function “br56” (see *Table 2*). We then obtain the result of *Figure 3*.

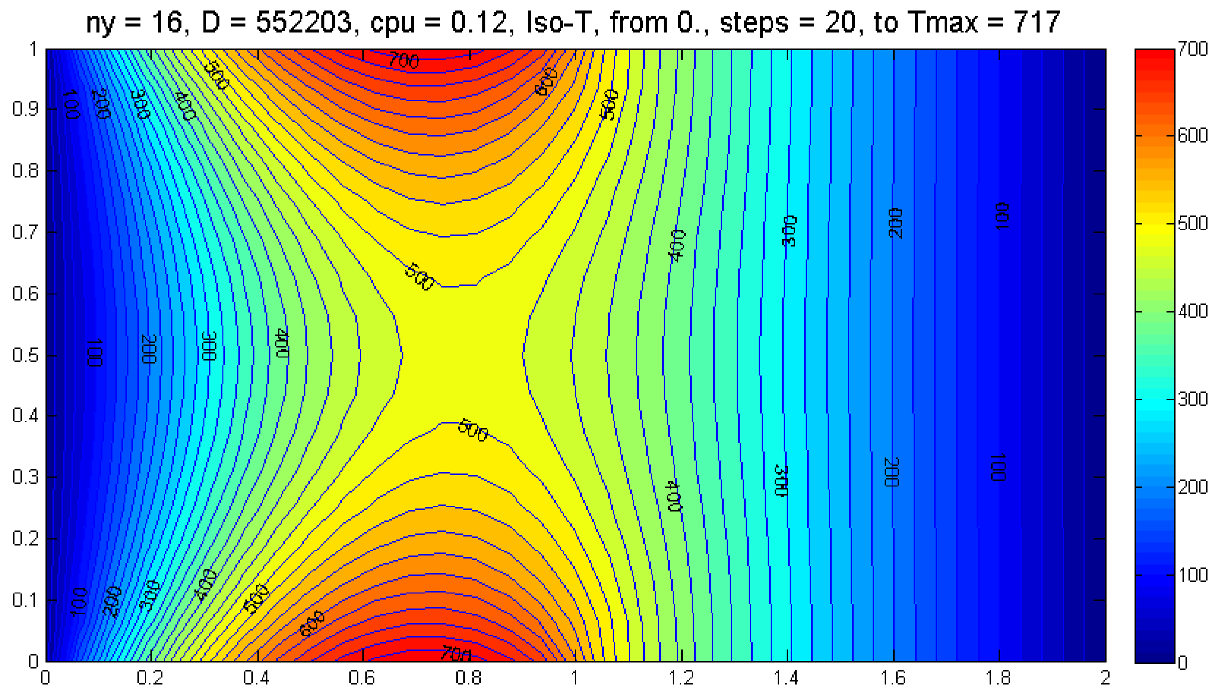


Figure 3: Isothermal curves filled with the modified color map

Table 2: Modified color map

```
function [bbr] = br56
bbr=[     0          0     0.5625
          0          0     0.6250
          0          0     0.6875
          0          0     0.7500
          0          0     0.8125
          0          0     0.8750
          0          0     0.9375
          0          0     1.0000
          0     0.0625     1.0000
          0     0.1250     1.0000
          0     0.1875     1.0000
          0     0.2500     1.0000
          0     0.3125     1.0000
          0     0.3750     1.0000
          0     0.4375     1.0000
          0     0.5000     1.0000
          0     0.5625     1.0000
          0     0.6250     1.0000
          0     0.6875     1.0000
          0     0.7500     1.0000
          0     0.8125     1.0000
          0     0.8750     1.0000
          0     0.9375     1.0000
          0     1.0000     1.0000
     0.0625     1.0000     0.9375
     0.1250     1.0000     0.8750
     0.1875     1.0000     0.8125
     0.2500     1.0000     0.7500
     0.3125     1.0000     0.6875
     0.3750     1.0000     0.6250
     0.4375     1.0000     0.5625
     0.5000     1.0000     0.5000
     0.5625     1.0000     0.4375
     0.6250     1.0000     0.3750
     0.6875     1.0000     0.3125
     0.7500     1.0000     0.2500
     0.8125     1.0000     0.1875
     0.8750     1.0000     0.1250
     0.9375     1.0000     0.0625
     1.0000     1.0000          0
     1.0000     0.9375          0
     1.0000     0.8750          0
     1.0000     0.8125          0
     1.0000     0.7500          0
     1.0000     0.6875          0
     1.0000     0.6250          0
     1.0000     0.5625          0
     1.0000     0.5000          0
     1.0000     0.4375          0
     1.0000     0.3750          0
     1.0000     0.3125          0
     1.0000     0.2500          0
     1.0000     0.1875          0
     1.0000     0.1250          0
     1.0000     0.0625          0
     1.0000          0          0];
end
```

The second application of [Beckers & Beckers 2015] is implemented as shown in *Table 3*. It is basically the same as *Table 1*, but with different boundary conditions.

Table 3: Procedure for solving the second application of reference[1]

```
clear all;tstart=tic;k=100;th=0.01;qin=100;pa=0.1;kt=k*th;f1=qin;f2=-f1/2;
Kca=kt/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4];  % Conduct element
ny=64;if ny>64;ny=64;end;if ny<2;ny=2;end;nx=ny*2;% Definition of the mesh
Ntca=(ny+1)*nx;nel=nx*ny;                           % Numbers of nodes & elements
loK=zeros(nel,4);my=ny+1;                     % localization vectors for T model
for i=1:nx;     % Nodes & elements are numbered: bottom - top, left - right
  for j=1:ny;loK((i-1)*ny+j,1) = my*(i-1)+j;
  loK((i-1)*ny+j,2) = loK((i-1)*ny+j,1)+my;
  loK((i-1)*ny+j,3) = loK((i-1)*ny+j,2)+1;
  loK((i-1)*ny+j,4) = loK((i-1)*ny+j,1)+1;end;end;
K=zeros((nx+1)*(ny+1),(nx+1)*(ny+1));        % Full K matrix initialization
for n=1:nel; for i=1:4; for j=1:4;                 % Assembling structural K
K(loK(n,i),loK(n,j))=K(loK(n,i),loK(n,j))+Kca(i,j);end;end;end
Ksca=K(1:(nx+1)*(ny+1)-ny-1,1:(nx+1)*(ny+1)-ny-1);     % Sub-matrix T model
gca=zeros(Ntca,1);                                  % Second member T model
for i = ny+2:my:(ny-1)*my+1; gca(i)=2*qin/nx;gca(i+ny)=2*qin/nx;end
gca(1)=qin/nx;gca(1+ny)=qin/nx;
gca((ny+3)*my)=f1;gca((ny+2)*my+1)=f2;                    % Additional jets
gca((ny+4)*my)=f1;gca((ny+3)*my+1)=f2;                    % Additional jets
gca(ny*my+1)=qin/nx;gca(ny*my+1+ny)=qin/nx;     % End second member T model
tca    = Ksca\gca;disca = gca'*tca/2;       % Temperature: tca & Dissipation
tK=zeros((nx+1)*(ny+1),1);tK(1:Ntca)=tca;gK=K*tK; % T full set & reactions
Kcl= K(1:(ny-1)*(nx+1)-ny+1,nx*(ny-1)+1:(ny+1)*(nx+1));
figure('Position',[1 1 1200 620]);Bt=zeros(my,nx+1);    % Isotherms drawing
ii=0;B=zeros(my,nx+1);x=zeros(my,nx+1);y=zeros(my,nx+1); % initializations
for j=1:nx+1;for i=1:my;ii=ii+1;Bt(i,j)=tK(ii);gapt=pa*qin/(kt);
x(i,j)=(j-1)*2/nx;y(i,j)=(i-1)*2/nx;end;end;telapsed=toc(tstart);
% br56;colormap(br56);
[CS,H]=contour(x,y,Bt,(0.0:gapt:max(tca)),'b','LineWidth',0.5);hold on;
clabel(CS,H,[100 200 300 400 500 600 700]);  % End of isotherms drawing
title(['ny = ',num2str(ny),', D = ',num2str(disca,'%0.6g'),', cpu = ',...
num2str(telapsed,'%0.2g'),', Iso-T, from 0., steps = ', num2str(gapt),...
', to max = ',num2str(max(tca),'%0.3g')],'fontsize', 15);  % End =========
```

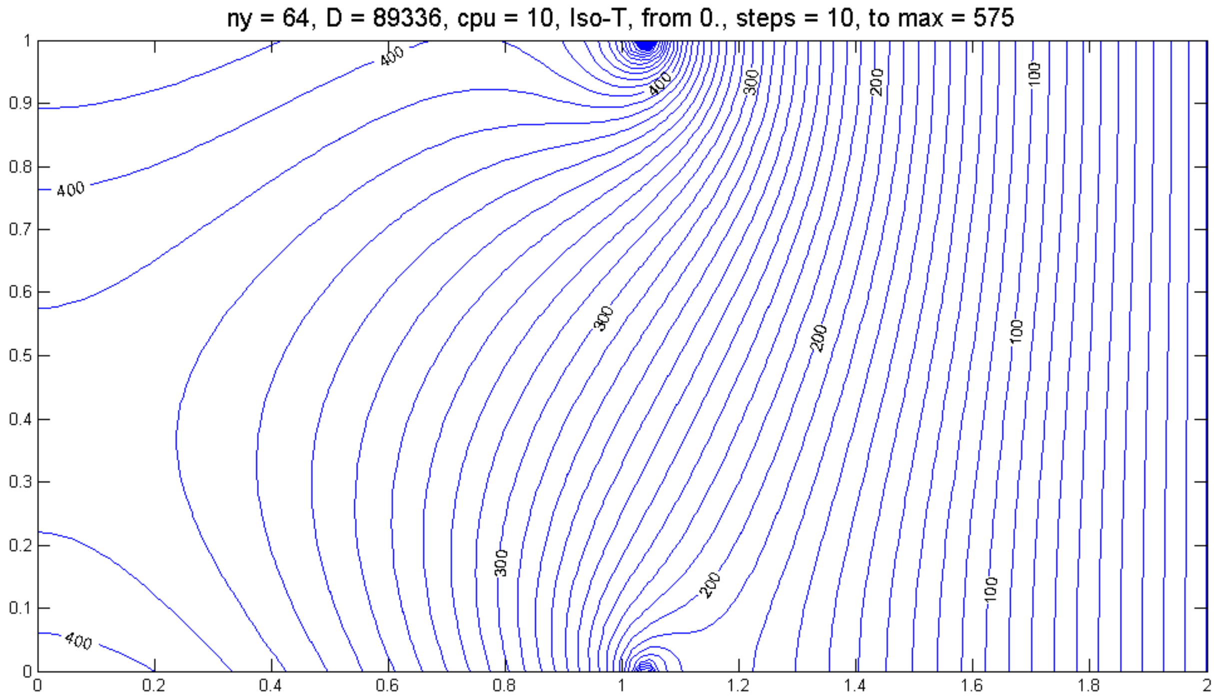


Figure 4: Isothermal curves for application 2

Like in the first application, if the function "*contour*" is replaced by "*contourf*" in line 29 of *Table 3*, we obtain:

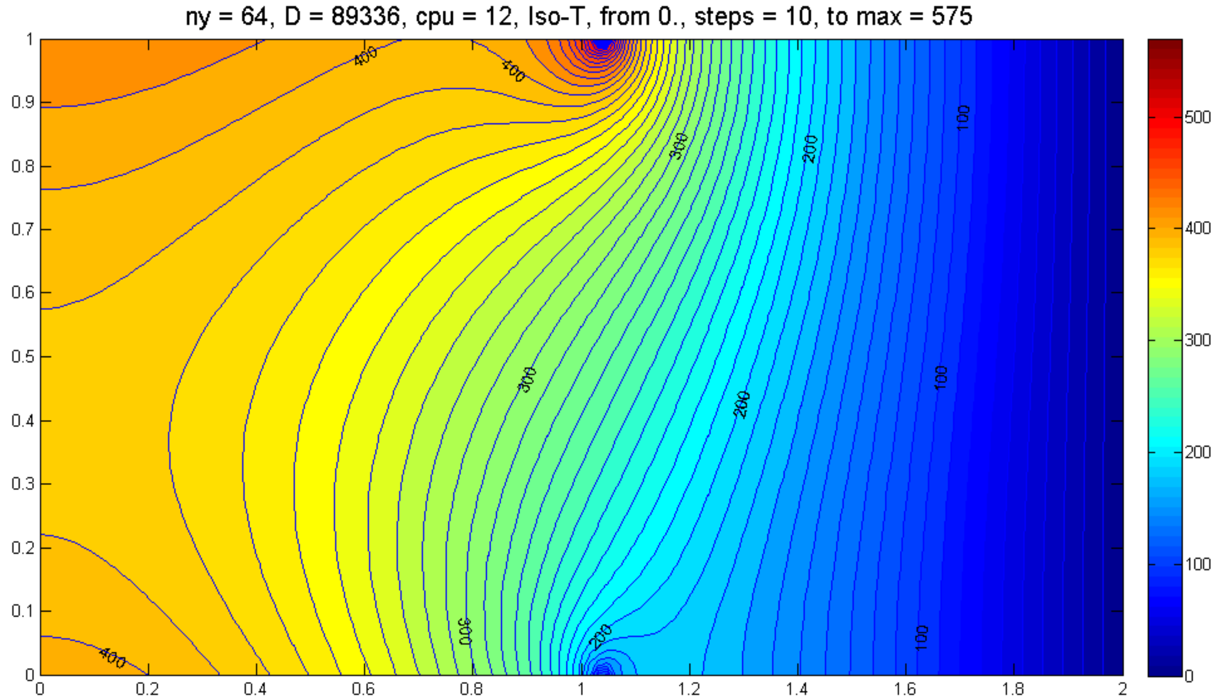


Figure 5: Isothermal curves - Matlab© default color map

With the modified color map activated in line 28 of *Table 3*, we have:

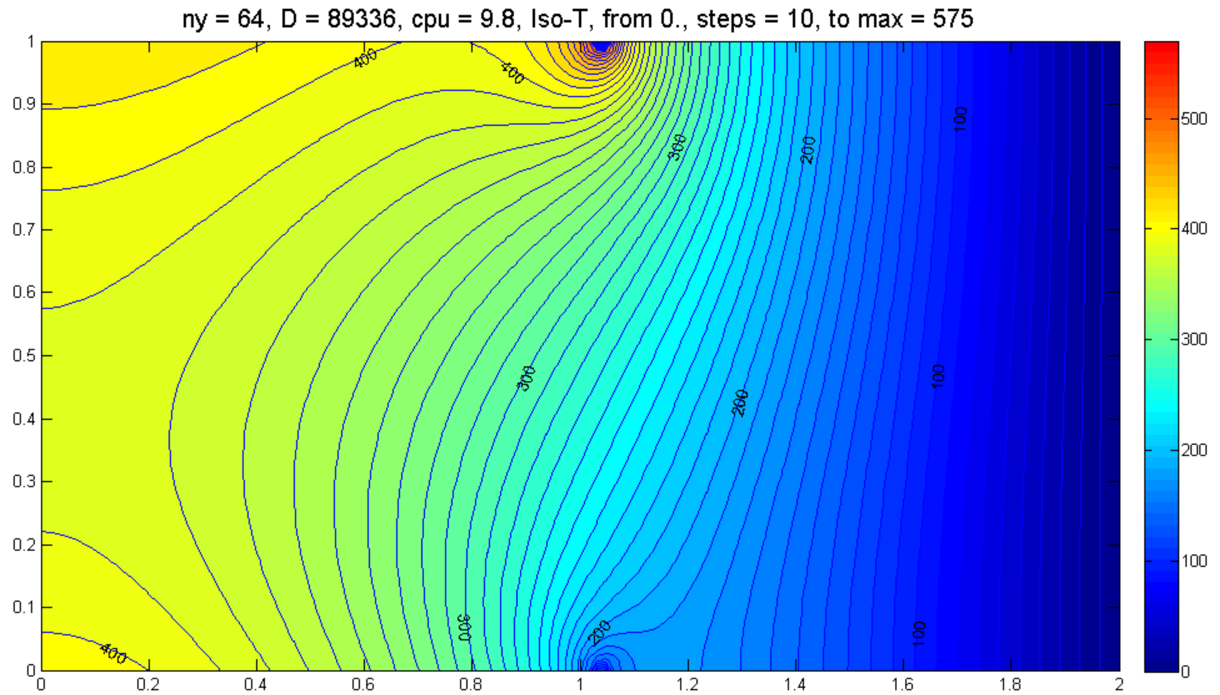


Figure 6: Isothermal curves – modified color map

In the next step, we use the same finite element model in order to discretize and to compute the "stream function" with the aim of describing the heat fluxes (*Table 4*).

In *2D*, the temperature and the stream function are both scalar, but the gradients of the first one have to satisfy the continuity equations while the derivatives of the second are built so as to give an irrotational vector field. Both functions satisfy the Laplace equation.

Table 4: Procedure for computing the stream function in application 1

```
clear all;tstart=tic;k=100;th=0.01;qin=1000;pa=0.025;kt=k*th;% Input data
Kca=kt/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4]; % Element conduct.
ny=32;if ny>64;ny=64;end;if ny<2;ny=2;end;nx=ny*2;% Definition of the mesh
Ntca= (ny+1)*(nx-1);nel=nx*ny;loK = zeros(nel,4);my=ny+1;    % loc vectors
for i=1:nx;for j=1:ny;
 loK((i-1)*ny+j,1)=my*(i-1)+j;loK((i-1)*ny+j,2)=loK((i-1)*ny+j,1)+my;
 loK((i-1)*ny+j,3)=loK((i-1)*ny+j,2)+1;
 loK((i-1)*ny+j,4)=loK((i-1)*ny+j,1)+1;end;end;
ii=0;lopsi=zeros((ny+1)*(nx+1),1);lop=zeros((ny+1)*(nx+1),1); % initialize
for j=2:ny     ;for i=1:nx+1;ii=ii+1;lopsi(ii)=(i-1)*(ny+1)+j;end;end
for j=1:ny:ny+1;for i=1:nx+1;ii=ii+1;lopsi(ii)=(i-1)*(ny+1)+j;end;end
lop(lopsi(1:(ny+1)*(nx+1)))=(1:(ny+1)*(nx+1));
loF=ones(nel,4);loF(1:nel,1:4)=lop(loK(1:nel,1:4));
K=zeros((nx+1)*(ny+1),(nx+1)*(ny+1));         % Full matrix initialization
for n=1:nel; for i=1:4; for j=1:4;
K(loF(n,i),loF(n,j))=K(loF(n,i),loF(n,j))+Kca(i,j);end;end;end
psi = ones((nx+1)*(ny+1),1)*.25*qin;              % Dirichlet B.C. top right
psi((nx+1)*(ny-1)+ny+1:(nx+1)*(ny-1)+2*ny+1)=-.25*qin; % B.C. bottom right
for i=1:ny;
    psi((nx+1)*(ny-1)+i)=(.75+(i-1)*(-.25-.75)/(ny))*qin;% B.C. top left
    psi((nx+1)*(ny)+i)   =(-.75+(i-1)*(.25+.75)/(ny))*qin;% B.C.bottom left
end                                    % End of Dirichlet boundary conditions
Kcl     = K(1:(ny-1)*(nx+1),(ny-1)*(nx+1)+1:(ny+1)*(nx+1));
Kpsi    = K(1:(ny-1)*(nx+1),1:(ny-1)*(nx+1));
psij    = psi((nx+1)*(ny-1)+1:(nx+1)*(ny+1));
psii    = Kpsi\(-Kcl*psij);psit=[psii;psij];tsa=zeros(Ntca,1);
dispsi = .5*psit'*K*psit/kt^2;                          % End of computation
figure('Position',[1 1 1200 620]);x=zeros(my,nx+1);y=zeros(my,nx+1);
for j=2:nx+1;for i=1:my;x(i,j)=(j-1)*2/nx;y(i,j)=(i-1)*2/nx;end;end
y(:,1)=y(:,2);gapf=pa*qin;br56;colormap(br56);
for i=1:my*(nx+1);tsa(lopsi(i))=psit(i);end;ii=0;% Stream function drawing
B=zeros(my,nx+1);for j=1:nx+1;for i=1:my;ii=ii+1;B(i,j)=tsa(ii);end;end
contourf(x,y,B,(-.75*qin:gapf:.75*qin),'r','LineWidth',1);hold on;% End SF
```

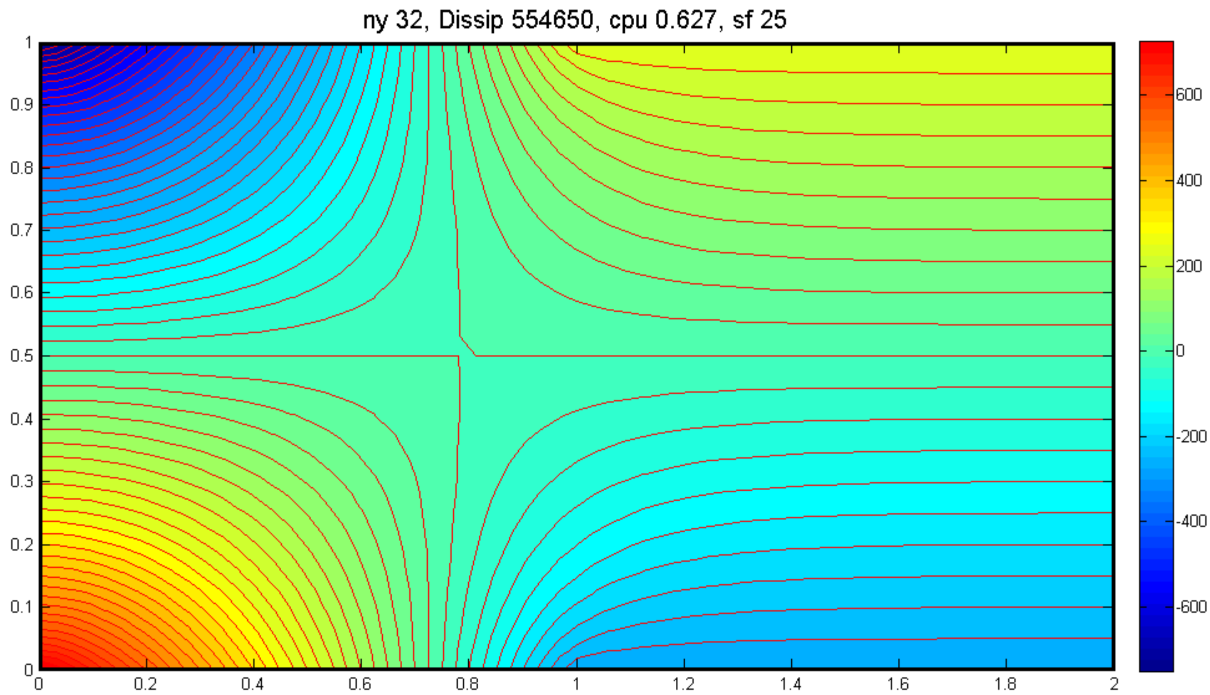


Figure 7: Streamlines superposed to the levels of the stream function

In *Figure 7*, on the left vertical side, the stream function is varying between 750 and -750 while it is varying between -250 and 250 on the right vertical side. Note that the difference of stream function values between two points is giving the normal flux through any line that connects them.

Following anticlockwise the boundary of the domain, a decreasing stream function means a heat inflow, a constant value means that there is no flow and an increasing stream function means an outflow.

It is now possible to put together the procedures of *Table 1* and *Table 4* in order to create a 2 x 33 lines procedure (*Table 5*) where we compute the potential (or temperature) and the stream function with the same element model (*Figure 8*).

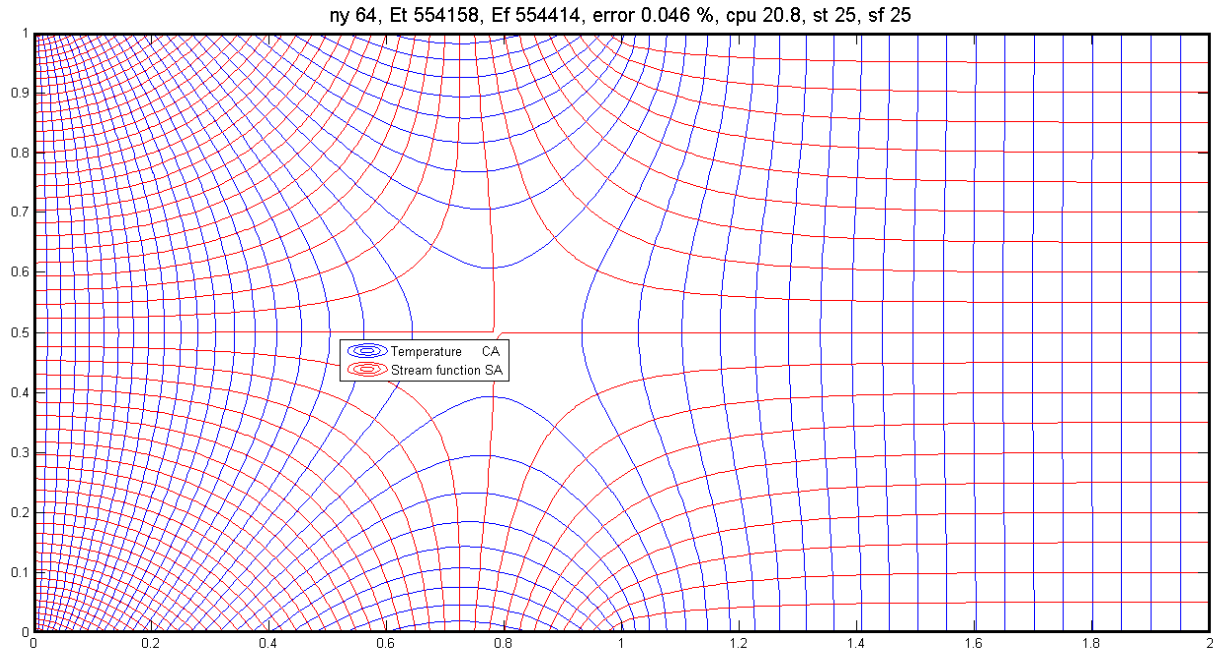


Figure 8: Superposition of isothermal curves (blue) and streamlines (red)

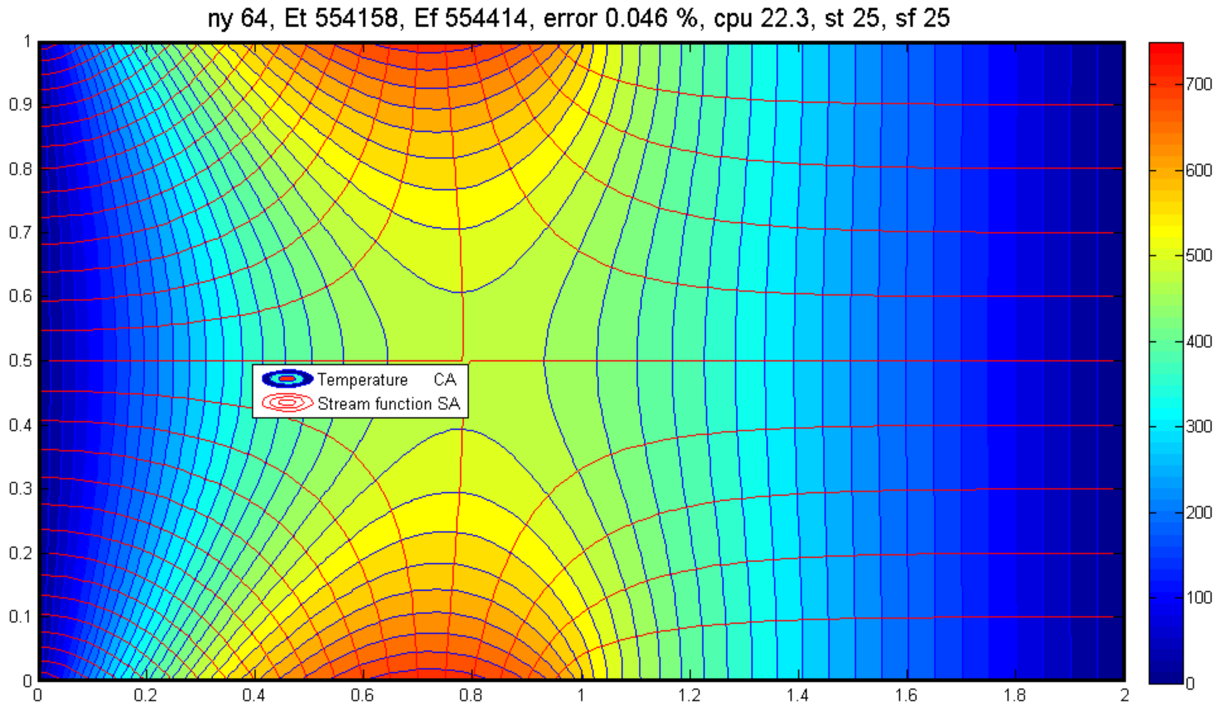
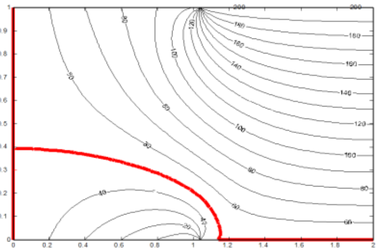


Figure 9: Isothermal curves, streamlines and levels

In the second application where two new flows are concentrated in the width of a single element (1/128 of the width of the domain), we obtain the results of *Figure 10* and *Figure 11*. The dissipation functions is equal to **89336 WK** for the temperature model and **95385 WK** for the heat flow or stream function model. In *Figure 12*, we draw much more streamlines separated by unity. We observe the amazing presence of the streamline "50" perpendicular to the left vertical wall but, if we compare with the streamlines 49 and 51, we see that this wall also corresponds to the streamline "50" what means that it is an insulated wall.



In this application there are 2 flows separated by the streamline "50". The upper flow is filling the main part of the domain. The second one is limited to less than a quarter of the domain area. The inferior right boundary of the domain also corresponds to a streamline "50".

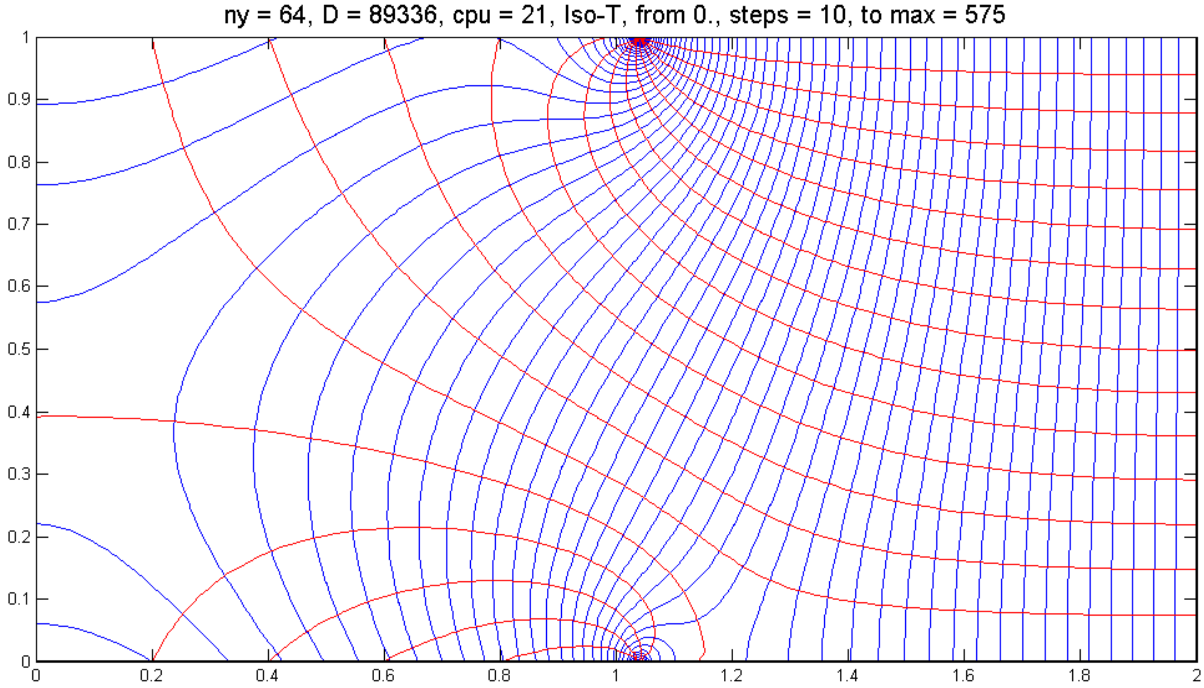


Figure 10: Superposition of isothermal curves (blue) and streamlines (red) – application 2

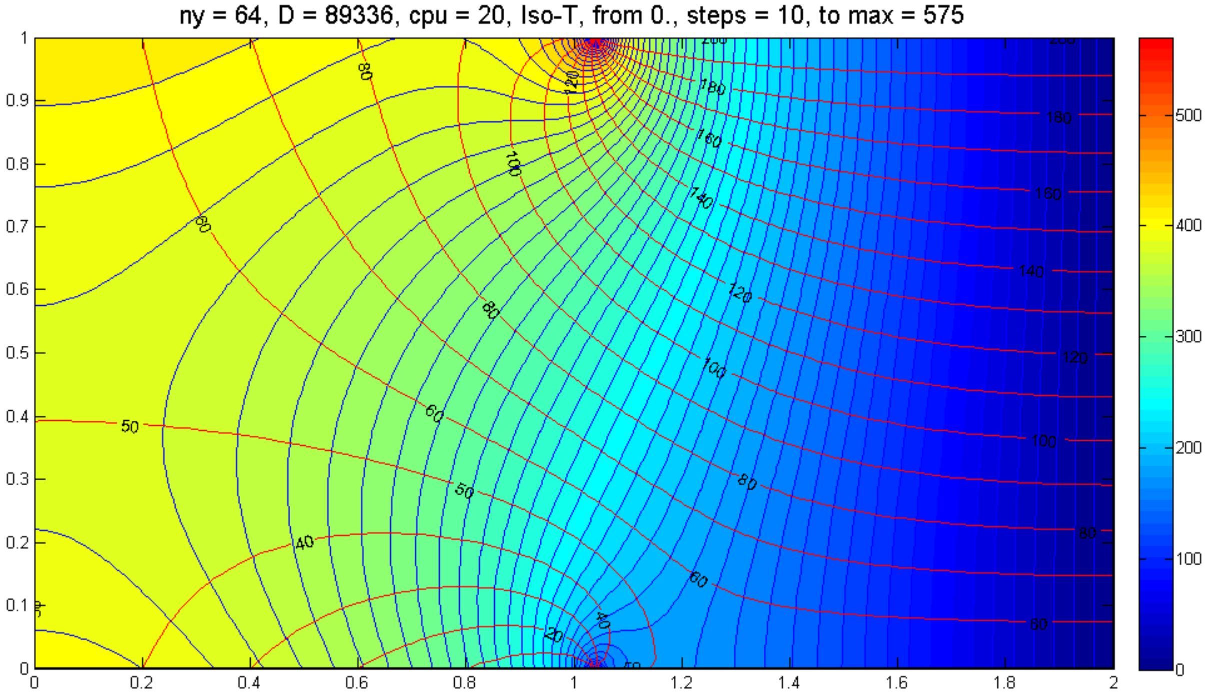


Figure 11: Streamlines superposed to the levels of the stream function in application 2

The two (2 x 33) procedures leading to *Figure 8*, *Figure 9* and *Figure 10*, *Figure 11* are given in *Table 5* and *Table 6*.

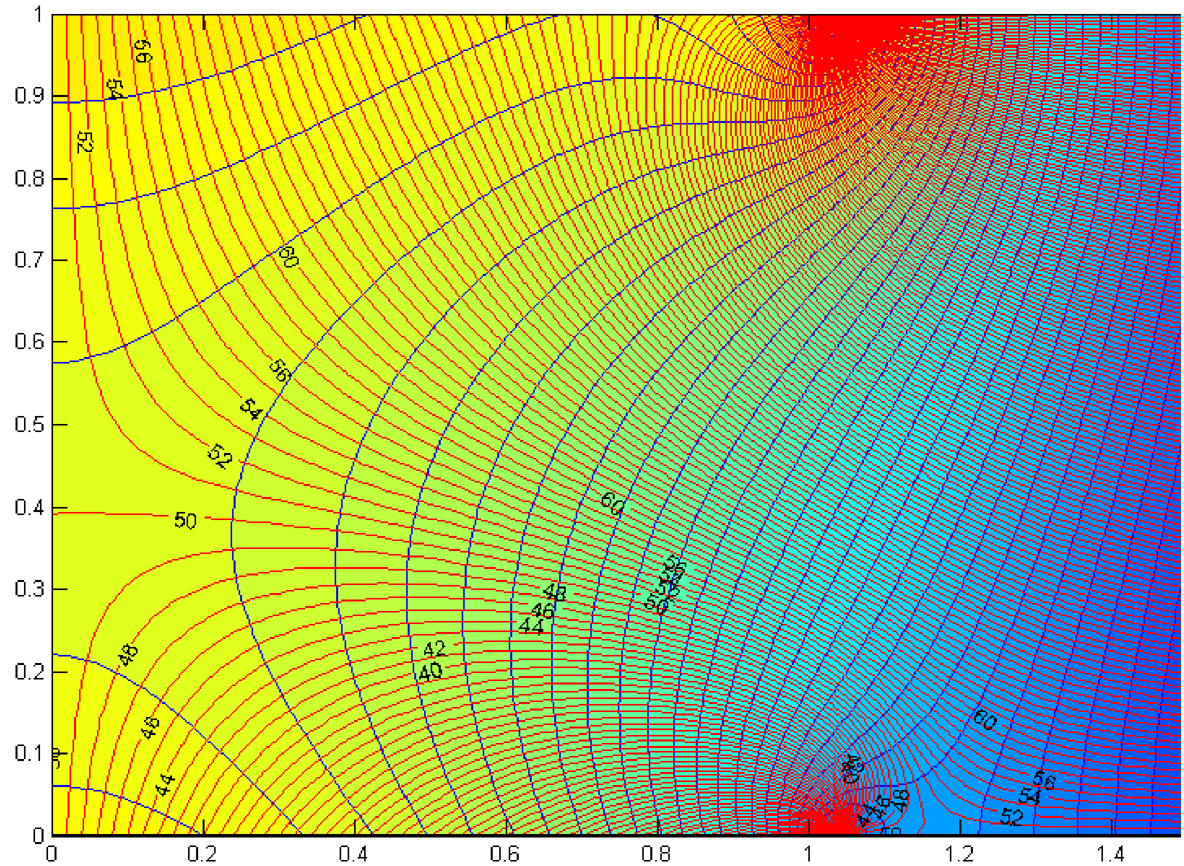


Figure 12: Zoom on the high density streamlines to show the zero flow on the left vertical side.

Final remarks:

All the procedures presented in this report are using the same finite element model that can be qualified as temperature model (or potential model in the frame of perfect fluid dynamics). We also call it cinematically admissible model, because in elasticity, it is based on the discretization of the displacement field. The dual element based on heat flow discretization is presented in the referred paper [Beckers & Beckers 2015].

In the procedure of *Table 4*, the same element is used to discretize the stream function while in *Table 5* and *Table 6*, the same finite element mesh is used two times with different boundary conditions in order to solve the dual problems expressed in terms of temperature and stream function.

Reference

Pierre Beckers, Benoit Beckers, "A 66 line heat transfer finite element code to highlight the dual approach", *Computers & Mathematics with Applications*, Volume 70 Issue 10, November 2015, Pages 2401-2413.

Annexes

Three Matlab© procedures including that of the above paper are listed in the following tables. The Matlab© script can be created by simple "copy and paste" operations.

Table 5: Procedure for computing the results of Figure 8

```
% Beckers - 2016 ======== Laplace_66 procedure ===========================
clear all;tstart=tic;k=100;th=0.01 ;qin=1000;pa=0.025;kt=k*th;% Input data
Kca=kt/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4]; % Element conduct.
ny=64;if ny>64;ny=64;end;if ny<2;ny=2;end;nx=ny*2;% Definition of the mesh
Ntca=(ny+1)*(nx-1);nel=nx*ny; % Numbers of DOF of C.A. model & of elements
loK=zeros(nel,4);my=ny+1;               % localization vectors for T model
for i=1:nx;     % nodes & elements are numbered: bottom - top, left - right
    for j=1:ny;
        loK((i-1)*ny+j,1) = my*(i-1)+j;
        loK((i-1)*ny+j,2) = loK((i-1)*ny+j,1)+my;
        loK((i-1)*ny+j,3) = loK((i-1)*ny+j,2)+1;
        loK((i-1)*ny+j,4) = loK((i-1)*ny+j,1)+1;
    end
end;
K=zeros((nx+1)*(ny+1),(nx+1)*(ny+1));       % Full K matrix initialization
for n=1:nel; for i=1:4; for j=1:4;               % Assembling structural K
K(loK(n,i),loK(n,j))=K(loK(n,i),loK(n,j))+Kca(i,j);end;end;end
Ksca=K(ny+2:(nx+1)*(ny+1)-ny-1,ny+2:(nx+1)*(ny+1)-ny-1);  % sub-matrix phi
gca=zeros(Ntca,1);                                  % Second member T model
for i = 1:my:(ny-2)*my+1; gca(i)=2*qin/nx;gca(i+ny)=2*qin/nx;end
gca((ny-1)*my+1)=qin/nx;gca((ny-1)*my+1+ny)=qin/nx;    % End 2d mem T model
tca    = Ksca\gca;                                      % Solution T model
disca = gca'*tca/2;                             % Dissipation energy T model
tK=zeros((nx+1)*(ny+1),1);tK(my+1:Ntca+my)=tca;gK=K*tK;         % Reactions
ii=0;lopsi=zeros((ny+1)*(nx+1),1);lop=zeros((ny+1)*(nx+1),1); % initialize
for j=2:ny     ;for i=1:nx+1;ii=ii+1;lopsi(ii)=(i-1)*(ny+1)+j;end;end
for j=1:ny:ny+1;for i=1:nx+1;ii=ii+1;lopsi(ii)=(i-1)*(ny+1)+j;end;end
lop(lopsi(1:(ny+1)*(nx+1)))=(1:(ny+1)*(nx+1));
loF=ones(nel,4);loF(1:nel,1:4)=lop(loK(1:nel,1:4));
K=zeros((nx+1)*(ny+1),(nx+1)*(ny+1));           % Full matrix initialization
for n=1:nel; for i=1:4; for j=1:4;
K(loF(n,i),loF(n,j))=K(loF(n,i),loF(n,j))+Kca(i,j);end;end;end
psi = ones((nx+1)*(ny+1),1)*.25*qin;               % Dirichlet B.C. top right
psi((nx+1)*(ny-1)+ny+1:(nx+1)*(ny-1)+2*ny+1)=-.25*qin; % B.C. bottom right
for i=1:ny;
    psi((nx+1)*(ny-1)+i)=(.75+(i-1)*(-.25-.75)/(ny))*qin;  % B.C. top left
    psi((nx+1)*(ny)+i)  =(-.75+(i-1)*(.25+.75)/(ny))*qin;% B.C.bottom left
end                                  % End of Dirichlet boundary conditions
Kcl     = K(1:(ny-1)*(nx+1),(ny-1)*(nx+1)+1:(ny+1)*(nx+1));
Kpsi    = K(1:(ny-1)*(nx+1),1:(ny-1)*(nx+1));
psij    = psi((nx+1)*(ny-1)+1:(nx+1)*(ny+1));
psii    = Kpsi\(-Kcl*psij);
psit    = [psii;psij];tsa=zeros(Ntca,1);
dispsi = .5*psit'*K*psit/kt^2;                          % End of computation
figure('Position',[1 1 1200 620]);                       % Isotherms drawing
ii=0;Bt=zeros(my,nx+1);x=zeros(my,nx+1);y=zeros(my,nx+1);% initializations
for j=2:nx+1;
    for i=1:my;
        ii=ii+1;
        if ii < my*(nx-1)+1;Bt(i,j)=tca(ii);end
        x(i,j)=(j-1)*2/nx;y(i,j)=(i-1)*2/nx;
    end
end
Bt(:,1)=0;Bt(:,nx+1)=0;y(:,1)=y(:,2);gapf=pa*qin;gapt=pa*qin/kt;
contourf(x,y,Bt,(0.0:gapt:max(tca)),'b','LineWidth',0.2);hold on;% End isoT
for i=1:my*(nx+1);tsa(lopsi(i))=psit(i);end;ii=0;% Stream function drawing
B=zeros(my,nx+1);for j=1:nx+1;for i=1:my;ii=ii+1;
        B(i,j)=(tsa(ii)+750)/2;end;end;br56;colormap(br56);
contour(x,y,B,(-.75*qin:gapf:.75*qin),'r','LineWidth',1);hold on;% End SF
legend('Temperature     CA','Stream function SA','Location','Best');
Error=(dispsi-disca)*200/(disca+dispsi);telapsed = toc(tstart);
title(['ny ',num2str(ny),', Et ',num2str(disca,'%0.6g'),', Ef ', ...
num2str(dispsi,'%0.6g'),', error ',num2str(Error,'%0.2g'),' %, cpu ',...
num2str(telapsed,'%0.3g'),', st ',num2str(gapt),...
', sf ',num2str(gapf)],'fontsize', 15);hold on;
plot([0 2 2 0 0],[0 0 1 1 0],'k','LineWidth',4);hold on; % End =========
```

Table 6: Procedure for computing the results of Figure 10

```
% Beckers - 2014 ======== Laplace_66 procedure ==========================
clear all;tstart=tic;k=100;th=0.01;qin=100;pa=0.1;kt=k*th;f1=qin;f2=-f1/2;
Kca=kt/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4];  % Conduct element
ny=64;if ny>64;ny=64;end;if ny<2;ny=2;end;nx=ny*2;% Definition of the mesh
Ntca=(ny+1)*nx;nel=nx*ny;                        % Numbers of nodes & elements
loK=zeros(nel,4);my=ny+1;                   % localization vectors for T model
for i=1:nx;     % Nodes & elements are numbered: bottom - top, left - right
    for j=1:ny;
        loK((i-1)*ny+j,1) = my*(i-1)+j;
        loK((i-1)*ny+j,2) = loK((i-1)*ny+j,1)+my;
        loK((i-1)*ny+j,3) = loK((i-1)*ny+j,2)+1;
        loK((i-1)*ny+j,4) = loK((i-1)*ny+j,1)+1;
    end
end;
K=zeros((nx+1)*(ny+1),(nx+1)*(ny+1));         % Full K matrix initialization
for n=1:nel; for i=1:4; for j=1:4;                 % Assembling structural K
K(loK(n,i),loK(n,j))=K(loK(n,i),loK(n,j))+Kca(i,j);end;end;end
Ksca=K(1:(nx+1)*(ny+1)-ny-1,1:(nx+1)*(ny+1)-ny-1);     % Sub-matrix T model
gca=zeros(Ntca,1);                                    % Second member T model
for i = ny+2:my:(ny-1)*my+1; gca(i)=2*qin/nx;gca(i+ny)=2*qin/nx;end
gca(1)=qin/nx;gca(1+ny)=qin/nx;
gca((ny+3)*my)=f1;gca((ny+2)*my+1)=f2;                      % Additional jets
gca((ny+4)*my)=f1;gca((ny+3)*my+1)=f2;                      % Additional jets
gca(ny*my+1)=qin/nx;gca(ny*my+1+ny)=qin/nx;     % End second member T model
tca   = Ksca\gca;                                       % Solution T model
disca = gca'*tca/2;                               % Dissipation energy T model
tK=zeros((nx+1)*(ny+1),1);tK(1:Ntca)=tca;gK=K*tK; % T full set & reactions
ii=0;lopsi=zeros((ny+1)*(nx+1),1);lop=zeros((ny+1)*(nx+1),1);     % Sf init
for j=2:ny      ;for i=2:nx+1;ii=ii+1;lopsi(ii)=(i-1)*(ny+1)+j;end;end
for j=1:ny:ny+1;for i=2:nx+1;ii=ii+1;lopsi(ii)=(i-1)*(ny+1)+j;end;end
lopsi(ii+1:ii+my)=1:my;lop(lopsi(1:(ny+1)*(nx+1)))=(1:(ny+1)*(nx+1));
loF=ones(nel,4);loF(1:nel,1:4)=lop(loK(1:nel,1:4));
K=zeros((nx+1)*(ny+1),(nx+1)*(ny+1));             % Full matrix initialization
for n=1:nel; for i=1:4; for j=1:4;          % Structural matrix assembling
K(loF(n,i),loF(n,j))=K(loF(n,i),loF(n,j))+Kca(i,j);end;end;end
psi = zeros((nx+1)*(ny+1),1);                       % Dirichlet B.C. top right
psi(nx*ny + ny+2+1:nx*(ny+1))=-f2;
psi(nx*(ny-1)+ny:nx*(ny-1)+2*ny)= qin;        % Dirichlet B.C. bottom right
psi(nx*(ny-1) + ny+2+1:nx*ny)= qin +f1;
psi((nx+1)*(ny+1):-1:(nx+1)*(ny+1)-ny)=.5*qin;   % B.C. Left vertical side
for i=1:ny-1;
    psi(nx*(ny  )+i)=(.5-i*.5/(ny))*qin;        % Dirichlet B.C.   top left
    psi(nx*(ny-1)+i)=(.5+i*.5/(ny))*qin;        % Dirichlet B.C.bottom left
end                                   % End of Dirichlet boundary conditions
Kcl   = K(1:(ny-1)*(nx+1)-ny+1,nx*(ny-1)+1:(ny+1)*(nx+1));
Kpsi  = K(1:nx*(ny-1),1:nx*(ny-1)); psij = psi(nx*(ny-1)+1:(nx+1)*(ny+1));
psii  = Kpsi\(-Kcl*psij);            psit =  [psii;psij];tsa=zeros(Ntca,1);
dispsi= 2*psit'*K*psit/kt^2;                           % End of computation
figure('Position',[1 1 1200 620]);Bt=zeros(my,nx+1);   % Isotherms drawing
ii=0;B=zeros(my,nx+1);x=zeros(my,nx+1);y=zeros(my,nx+1); % initializations
for j=1:nx+1;
    for i=1:my;
        ii=ii+1;
        Bt(i,j)=tK(ii);
        x(i,j)=(j-1)*2/nx;    y(i,j)=(i-1)*2/nx;
    end
end
gapf=pa*qin;gapt=pa*qin/kt;br56;colormap(br56)
contour(x,y,Bt,(0.0:gapt:max(tca)),'b','LineWidth',0.5);hold on;
for i=1:my*(nx+1);tsa(lopsi(i))=psit(i);end;    % Stream function drawing
ii=0;for j=1:nx+1;for i=1:my;ii=ii+1;B(i,j)=tsa(ii);end;end
BB(1:my,1:nx+1)=B(my:-1:1,1:nx+1);
contour(x,y,BB,(min(psit):gapf:max(psit)),'r','LineWidth',1);hold on;
% [CS,H]=contour(x,y,BB,(0.0:gapt:max(tca)),'r','LineWidth',0.5);hold on;
% clabel(CS,H,[0 20 40 60 80 100 120 140 160 180 200 220]);
plot([0 2 2 0 0],[0 0 1 1 0],'k','LineWidth',1.5);hold on; % End drawing
```

Table 7: Procedure given in the reference: [Beckers & Beckers 2015]. *This procedure produces the figure 5 of the reference*

```
% Beckers - 2015 =========== Dual_66 free code procedure =================
clear all;tstart=tic;k=100;t=0.01;kt=k*t;qin=1000;              % Input data
ny=32;ny=min(ny,90);ny=round(ny/2)*2;nx=ny*2;nel=nx*ny;% Upper domain mesh
nys=ny/2;Ntsa=nys*(nx-1)+nx*(nys+1);Ntca=(nys+1)*(nx-1);%Numb DOF: SA & CA
% Statement 3: ny forced to be even. Computation limited here to sym. part
Kca=kt/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4];       % K: CA model
lc = zeros(nx*nys,4);my=nys+1;              % Localisation vectors: CA model
for i=1:nx;
for j=1:nys;
lc((i-1)*nys+j,1) = my*(i-1)+j;
lc((i-1)*nys+j,2) = lc((i-1)*nys+j,1)+my;
lc((i-1)*nys+j,3) = lc((i-1)*nys+j,2)+1;
lc((i-1)*nys+j,4) = lc((i-1)*nys+j,1)+1;
end;
end;
K=zeros(Ntca+2*(nys+1),Ntca+2*(nys+1));%Full Conductivity matrix: CA model
for n=1:nx*nys;
for i=1:4;
for j=1:4;
K(lc(n,i),lc(n,j))=K(lc(n,i),lc(n,j))+Kca(i,j);
end;
end;
end
Ksca=K(nys+2:(nx+1)*(nys+1)-nys-1,nys+2:(nx+1)*(nys+1)-nys-1);% Sub-matrix
gca=zeros(Ntca,1);                                % Second member: CA model
for i = 1+nys:my:nys*2*my-my;gca(i)=2*qin/nx;end;gca(ny*my)=qin/nx;
tca =Ksca\gca;disca = gca'*tca;clear Ksca;clear Kc; % Solution of CA model
% ========================================================================
Ksa=kt/2*[5 1 -3 -3; 1 5 -3 -3; -3 -3 5 1; -3 -3 1 5];        % K: SA model
ls = zeros(nx*nys,4);                       % Localisation vectors: SA model
for j=1:nx;
for i=1:nys;
ls((j-1)*nys+i,1)=(j-1)*(ny+1)+nys+i+1;
ls((j-1)*nys+i,2)=ls((j-1)*nys+i,1)-1;
ls((j-1)*nys+i,3)=j*(ny+1)+i;
ls((j-1)*nys+i,4)=ls((j-1)*nys+i,3)-ny-1;
end;
end
K=zeros(Ntsa+ny,Ntsa+ny);                   % Full conductivity matrix; SA model
for n=1:nx*nys;
for i=1:4;
for j=1:4;
K(ls(n,i),ls(n,j))=K(ls(n,i),ls(n,j))+Ksa(i,j);
end;
end;
end;
Kssa=K(nys+1:Ntsa+nys,nys+1:Ntsa+nys);clear K; % Sub-matrix to be inverted
gsa=zeros(Ntsa,1);                          % Second member of the SA model
for i=1+nys:2*nys+1:ny*(ny+1)-nys;gsa(i)=qin/(ny);end;
tsa =Kssa\gsa;dissa = gsa'*tsa;clear Kssa;    % Solution & dissip. function
% ========================================================================
% Drawing CA & SA temperatures along the boundary of the symmetric part
figure('Position',[1 1 1600 800]);axes('fontsize',15)
xnod=0:2/nx:5;px=1/nx:2/nx:5-1/nx;
test=[0 (tca(1:my:(nx-2)*my+1))' (0:2/nx:.5)*0 (tca((nx-1)*my:-my:my))'...
 (0:2/nx:.5)*0];
td=[tsa(1:ny+1:(2*nys+1)*nx-ny)' (0:2/nx:.5-2/nx)*0 ...
tsa((ny+1)*nx-nys :-(ny+1):nys+1 )' (0:2/nx:.5-2/nx)*0];
plot(xnod,test,'k');hold on;
plot(px,td,'.k','MarkerSize',12);hold on;grid on;
axis([-.02 5.02 -max(tca)/100 max(max(tca),max(td))/.98])
Error=(dissa-disca)*200/(dissa+disca);telapsed = toc(tstart);
title([num2str(nx),' x ',num2str(ny),', CA: ',num2str(disca,'%0.6g'),...
', SA: ',num2str(dissa,'%0.6g'),', Er: ',num2str(Error,'%0.2g'),...
' %, ','cpu: ',num2str(telapsed,'%0.3g')],'fontsize', 15)
legend('CA ','SA ','Location','NorthWest')
```