SF2521 : Numerical Solutions of Differential Equations

Homework 1

Thomas Garrett & Weicker David

8th February 2016

Introduction

This report presents the results for the first homework of SF2521.

1 Conservation laws

We are given the following Euler equations in $u = (\rho, \rho v, E)$,

$$\rho_t + (\rho v)_x = 0 \quad \text{(Conservation of mass)} \tag{1}$$
$$(\rho v)_t + (\rho v^2 + p)_x = 0 \quad \text{(Conservation of momentum)} \tag{2}$$
$$E_t + (v(E+p))_x = 0 \quad \text{(Conservation of energy)} \tag{3}$$

and the variable change $u = (\rho, \rho v, E) = (u_1, u_2, u_3)$. We then have

$$f(u) = (f_1, f_2, f_3)^T = (u_2, u_2^2/u_1 + p(u_1), u_2(u_3 + p(u_1))/u_1)$$

Using the u coordinates, we can write equations (1), (2), (3) as

$$u_t + f(u)_x = 0$$

From the chain rule, we can rewrite our equation as

$$u_t + \frac{\partial f}{\partial u} u_x = 0 \tag{4}$$

We now wish to find the eigenvalues and eigenvectors of the matrix

$$\frac{\partial f}{\partial u} = \begin{pmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \frac{\partial f_1}{\partial u_3} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \frac{\partial f_2}{\partial u_3} \\ \frac{\partial f_3}{\partial u_1} & \frac{\partial f_3}{\partial u_2} & \frac{\partial f_3}{\partial u_3} \end{pmatrix}$$

Using simple partial differentiation, we calculate

$$\frac{\partial f}{\partial u} = \begin{pmatrix} 0 & 1 & 0 \\ a & b & 0 \\ c & d & e \end{pmatrix}$$

with $a = -(\frac{u_2}{u_1})^2 + p'(u_1)$,
$b = 2(\frac{u_2}{u_1})$,
$c = \frac{u_2 p'(u_1)}{u_1} - \frac{u_2(u_3+p(u_1))}{u_1^2}$,
$d = \frac{u_3+p(u_1)}{u_1}$,
and $e = \frac{u_2}{u_1}$

To get the eigenvalues, we look at the roots of the characteristic polynomial of this matrix

$$\lambda^3 - e\lambda^2 - b\lambda^2 + be\lambda - a\lambda + ae = 0$$

which gives us the three eigenvalues

$$\lambda_1 = \frac{1}{2}(b - \sqrt{4a + b^2})$$
$$\lambda_2 = \frac{1}{2}(b + \sqrt{4a + b^2})$$
$$\lambda_3 = e$$

Plugging in the actual values of a, b, and e, we get

$$\lambda_1 = \frac{u_2}{u_1} - \sqrt{p'(u_1)}$$
$$\lambda_2 = \frac{u_2}{u_1} + \sqrt{p'(u_1)}$$
$$\lambda_3 = \frac{u_2}{u_1}$$

We compute the eigenvectors as follows

$$\begin{pmatrix} 0 & 1 & 0 \\ a & b & 0 \\ c & d & e \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \lambda_1 \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

which gives the system of equations

$$y = (\frac{u_2}{u_1} - \sqrt{p'(u_1)})x$$
$$ax + by = (\frac{u_2}{u_1} - \sqrt{p'(u_1)})y$$
$$cx + dy + ez = (\frac{u_2}{u_1} - \sqrt{p'(u_1)})z$$

We pick $x = 1$, and after solving we get

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{u_2}{u_1} - \sqrt{p'(u_1)} \\ \frac{u2\sqrt{p'(u_1)} - (u_3 + p(u_2)))}{u_1} \end{pmatrix}$$

We compute the second eigenvector similarly and get

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{u_2}{u_1} + \sqrt{p'(u_1)} \\ \frac{u2\sqrt{p'(u_1)} + (u_3 + p(u_2)))}{u_1} \end{pmatrix}$$

The last eigenvalue is computed from the system of equations

$$y = ex$$
$$ax + by = ey$$
$$cx + dy + ez = ez$$

from which we get

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

1.1 Linearisation

We have

$$u_t + \frac{\partial f}{\partial u} u_x = 0 \tag{5}$$

where $\frac{\partial f}{\partial u}$ is dependent on u. This equation immediately becomes linear if we fix $\frac{\partial f}{\partial u}$ at a specific point. We call it u_0, and we denote $\frac{\partial f}{\partial u}(u_0)$ as $\frac{\partial f}{\partial u}$ evaluated at this point. Thus, we have a linear equation

$$u_t + \frac{\partial f}{\partial u}(u_0) u_x = 0$$

which behaves like the nonlinear equation in a neighbourhood around u_0.

1.2 Hyperbolicity

We know that an equation of this form is hyperbolic if $\frac{\partial f}{\partial u}$ has real eigenvalues and linearly independent eigenvectors. It can be seen easily that the eigenvalues are real as long as $p'(u_1) \geq 0$. We can also see that the eigenvectors are independent as long as either $p'(u_1) \neq 0$ or $u_3 + p(u_2) \neq 0$

1.3 Transport Equation

We now derive conditions on a and b for the two-dimensional transport equation

$$u_t + a(x,y)u_x + b(x,y)u_y = 0$$

to be a conservation law. we look at the basic form of a conservation law in two dimensions

$$u_t + \nabla f(u) = 0 \tag{6}$$

and we find conditions which enable us to write our transport equation in this form. We assume

$$f = \begin{pmatrix} a(x,y) & b(x,y) \end{pmatrix}$$

so plugging into (5) we get

$$u_t + \nabla(\begin{pmatrix} a(x,y) & b(x,y) \end{pmatrix} u) = u_t + (a(x,y))_x u + a(x,y)u_x + (b(x,y))_y u + b(x,y)u_y = 0$$

We now notice that if

$$(a(x,y))_x = -(b(x,y))_y$$

we get

$$u_t + \nabla(\begin{pmatrix} a(x,y) & b(x,y) \end{pmatrix} u) = u_t + a(x,y)u_x + b(x,y)u_y = 0$$

which is our two dimensional transportation equation. Thus,

$$(a(x,y))_x = -(b(x,y))_y$$

is our conservation law condition.

2 Heat Equation

2.1 Flux vector

From the course text book, 'Finite Volume Methods for Hyperbolic Problems', we know that

$$\text{flux} = -\beta q_x$$

when the heat equation is in the form

$$q_t = \beta q_{xx} + S$$

Comparing with our equation

$$q_t = \nabla \cdot (\nabla q) + S$$

we can see that $\beta = 1$. Therefore the flux vector is $-\nabla q$.

2.2 Q(t)

We are now interested in computing :

$$Q(t) = \int_0^1 \int_0^1 q(x, y, t) dxdy$$

Using the fundamental theorem of calculus, we have :

$$Q(t) = Q(0) + \int_0^t Q'(\tau) d\tau$$

The initial condition gives :

$$Q(0) = \int_0^1 \int_0^1 q(x, y, 0) dxdy = \int_0^1 \int_0^1 0 \, dxdy = 0$$

We also know that (using the definition of Q) :

$$Q'(t) = \frac{d}{dt}(\int_0^1 \int_0^1 q(x, y, t) dxdy) = \int_0^1 \int_0^1 q_t(x, y, t) dxdy$$

We are now going to use the PDE to substitute q_t. This yields :

$$Q'(t) = \int_0^1 \int_0^1 q_t(x, y, t) dxdy = \int_0^1 \int_0^1 \nabla \cdot (\nabla q(x, y, t)) dxdy + \int_0^1 \int_0^1 S(x, y, t) dxdy$$

With divergence theorem, and if S means the boundary of the domain and $\mathbf{n}$ the outward unit normal to the boundary, we have :

$$Q'(t) = \oint_S \nabla q(x, y, t) \cdot \mathbf{n} \, ds + \int_0^1 \int_0^1 S(x, y, t) dxdy$$

Because the given boundary condition given is :

$$\nabla q(x, y, t) \cdot \mathbf{n} = 0$$

We finally have :

$$Q'(t) = \int_0^1 \int_0^1 S(x, y, t) dxdy$$

This yields an expression for Q as a function of t.

$$Q(t) = \int_0^t \int_0^1 \int_0^1 S(x, y, \tau) dxdyd\tau$$

Because S is a given function, this expression can always be computed as a function of t.

3 Discretization and implementation

In this section, we will implement a finite volume method to solve the problem given.

3.1 Finite Volume method

First, we need to define the method used. Let us start with the PDE and take the average over cell (i, j).

$$\frac{1}{\Delta x \Delta y} \iint_{(i,j)} q_t dx dy = \frac{1}{\Delta x \Delta y} (\iint_{(i,j)} \nabla \cdot (\nabla q) dx dy + \iint_{(i,j)} S \, dx dy)$$

Using the definition of Q_{ij}, divergence theorem and defining $S_{ij}(t) = \frac{1}{\Delta x \Delta y} \iint_{(i,j)} S \, dx dy$, we get :

$$\frac{dQ_{ij}}{dt} = \frac{1}{\Delta x \Delta y} \oint_{(i,j)} \nabla q \cdot \mathbf{n} \, ds + S_{ij}(t)$$

Because each cell is a rectangle, the first term in the right-hand side can be expressed as the sum of the integral evaluated at each side (East, North, West and South) :

$$\oint_{(i,j)} \nabla q \cdot \mathbf{n} \, ds = \int_E q_x dy + \int_N q_y dx - \int_W q_x dy - \int_S q_y dx$$

The next step is to discretize those integral. We will use finite differences. Let us assume that we are not on the boundary.

$$\begin{aligned}
\int_E q_x dy &\approx \Delta y \frac{Q_{i+1,j} - Q_{i,j}}{\Delta x} \\
\int_W q_x dy &\approx -\Delta y \frac{Q_{i-1,j} - Q_{i,j}}{\Delta x} \\
\int_N q_y dx &\approx \Delta x \frac{Q_{i,j+1} - Q_{i,j}}{\Delta y} \\
\int_S q_y dx &\approx -\Delta x \frac{Q_{i,j-1} - Q_{i,j}}{\Delta y}
\end{aligned}$$

If we are on the boundary, we use the boundary condition to have :

$$\begin{aligned}
i = 1 &\implies \int_W q_y dx = 0 \\
i = m &\implies \int_E q_y dx = 0 \\
j = 1 &\implies \int_S q_x dy = 0 \\
j = n &\implies \int_N q_x dy = 0
\end{aligned}$$

Using the finite differences above, in general, we have the following discrete equation :

$$\frac{dQ_{ij}}{dt} = \frac{aQ_{i+1,j} + bQ_{i,j} + cQ_{i-1,j}}{\Delta x^2} + \frac{dQ_{i,j+1} + eQ_{i,j} + fQ_{i,j-1}}{\Delta y^2} + S_{ij}(t)$$

Where a, b, c, d, e, f are given in the two tables below :

	a	b	c
i = 1	1	-1	0
i = 2:m-1	1	-2	1
i = m	0	-1	1

	d	e	f
j = 1	1	-1	0
j = 2:n-1	1	-2	1
j = n	0	-1	1

In order to solve this numerically, we also need to rewrite the unknown matrix Q as a vector. We do this by defining $Qvect$ as :

$$Qvect_k = Q_{i,j} \iff k = (j-1)m + i$$

So we now get a system of linear ODE's to solve (where M is a $mn \times mn$ matrix) :

$$\frac{dQvect}{dt} = MQvect + S$$

The only remaining thing to do is to build M. In order to do this let us define $I(c)$ the $c \times c$ identity matrix and $A(c)$ the following $c \times c$:

$$A = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 1 & -2 & 1 \\ 0 & \cdots & 0 & 1 & -1 \end{pmatrix}$$

We can now build B, representing the x-derivative :

$$B = \frac{1}{\Delta x^2} I(n) \otimes A(m)$$

Where $\otimes$ is the kronecker product. We can check that the size of B is indeed $mn \times mn$ because $A(m)$ is of size $m \times m$.

We can also build C, representing the y-derivative :

$$C = \frac{1}{\Delta y^2} \begin{pmatrix} -I(m) & I(m) & 0 & \cdots & 0 \\ I(m) & -2I(m) & I(m) & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & I(m) & -2I(m) & I(m) \\ 0 & \cdots & 0 & I(m) & -I(m) \end{pmatrix} = \frac{1}{\Delta y^2} A(n) \otimes I(m)$$

Finally :

$$M = B + C$$

3.2 Implicit Euler

Now that the problem is spatially discretized, we can start solving the system of ODE's. Let us first introduce the time step Δt and some notation :

$$Qvect^{(T)} \approx Qvect(T\Delta t)$$

$$S_k^{(T)} \approx \frac{1}{\Delta x \Delta y} \iint_{(i,j)} S(x, y, T\Delta t) dx dy$$

Because, for the second heat source, S depends on t. We also have to numerically compute the mean of the heat source on each cell. We will use the classic central point approximation.

$$S_k^{(T)} = S(x_i, y_j, T\Delta t)$$

We can now use implicit euler for the time derivative. We also have to remember that M does not depends on t and thus :

$$\frac{Qvect^{(T+1)} - Qvect^{(T)}}{\Delta t} = MQvect^{(T+1)} + S^{(T+1)}$$

If Id is the $mn \times mn$ identity matrix, we have :

$$(Id - \Delta t M)Qvect^{(T+1)} = Qvect^{(T)} + S^{(T+1)}$$

$Qvect^{(0)}$ is filled with zeroes because of the initial condition and the equation above gives the recursion to compute every $Qvect$. We can then reshape $Qvect$ to obtain the matrix Q.

Here, implicit euler is used because the problem is stiff (M has eigenvalues of very different magnitudes). Thus, an explicit method would require us to take a very little time step in order to have a stable solution. The use of an implicit method allows us to take a much bigger time step and thus to be more efficient.

3.3 Problem in matrix form

It is convenient to use the matrix form because we do not have to think about changing lines to take the boundary conditions into account and we thus avoid kronecker products.

If we use this to state the problem, we have :

$$\frac{dQ}{dt} = QT_x + T_yQ + S$$

Using implicit euler and the same notation as before, we get :

$$\frac{Q^{(T+1)} - Q^{(T)}}{\Delta t} = Q^{(T+1)}T_x + T_yQ^{(T+1)} + S$$

$$Q^{(T+1)} - \Delta t Q^{(T+1)}T_x - \Delta t T_y Q^{(T+1)} = Q^{(T)} + \Delta t S$$

This recursion equation is unfortunately not solvable because $Q^{(T+1)}$ is multiplied on the left once and on the right another time. This is why we have to write Q as a vector to solve the problem with implicit euler.

However, the matrix form allows us to prove that the finite volume method is exactly conservative. Indeed, since the walls are insulated, if the source term is 0, we should keep the same amount of heat in the domain.

$$\frac{d}{dt}\int_0^1\int_0^1 Q dx dy = 0$$

Let us see if that is indeed the case for our method. Because Q_{ij} is defined as the average on cell (i, j), we have that :

$$\int_0^1\int_0^1 Q dx dy = \Delta x \Delta y \sum_i \sum_j Q_{ij}$$

So we only have to prove that, in the absence of a source term, the sum of all entries does not vary in time. Since summing a matrix is multiply it by vectors filled with zeroes :

$$\frac{d}{dt}(\begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} Q \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}) = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} \frac{dQ}{dt} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

Using our scheme (and remembering that the source term is 0...), we get :

$$\begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} \frac{dQ}{dt} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} QT_x \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} + \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} T_y Q \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

Because of the structure of T_x and T_y, we also have :

$$T_x \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} T_y = \begin{pmatrix} 0 & \cdots & 0 \end{pmatrix}$$

So the right hand side is equal to zero.
We can then conclude that the finite volume method is exactly conservative.

3.4 Approximation of the dirac

Remember that the discretized problem says :

$$S_k^{(T)} = S(x_i, y_j, T\Delta t)$$

But, when S is the dirac function, we cannot do that because it is impossible to evaluate the dirac function. So, we will approximate it by the given function.

3.5 Implementation

The Matlab code for the implementation of the finite volume method for this problem can be found at the end of the report.

4 Numerical results

4.1 Figures

As we can see in Figure (1) with the constant heat source, the total temperature is constantly increasing with the hottest part being at heat source. Up until $T = 0.25$, we observe the same phenomena in Figure (2) that we observe for the smooth heat source. However, once we pass $T = 0.25$, the total heat stays the same and starts to level out.

4.2 Convergence

Because we do not know the true solution to the problem, we study ratios, which we denote by R, of the differences of the computed values i.e.

$$R_h = \frac{Q'_{h/2,t} - Q'_{h,t}}{Q'_{h/4,t} - Q'_{h/2,t}}$$

where, for example, $Q'_{h/2,t}$ is our computed solution using $\Delta x = \Delta y = h/2$ and $\Delta t = t$. We wish to show that the error behaves as $O(\Delta t^p) + O(\Delta h^r)$ and find p and r. Letting $Q*$ denote the true solution, we write write our ratio as

$$R_h = \frac{Q* + O(\Delta t^p) + O(\Delta (h/2)^r) - (Q* + O(\Delta t^p) + O(\Delta h^r))}{Q* + O(\Delta t^p) + O(\Delta (h/4)^r) - (Q* + O(\Delta t^p) + O(\Delta (h/2)^r))} = \frac{O(\Delta (h/2)^r) - O(\Delta h^r)}{O(\Delta (h/4)^r) - O(\Delta (h/2)^r)}$$

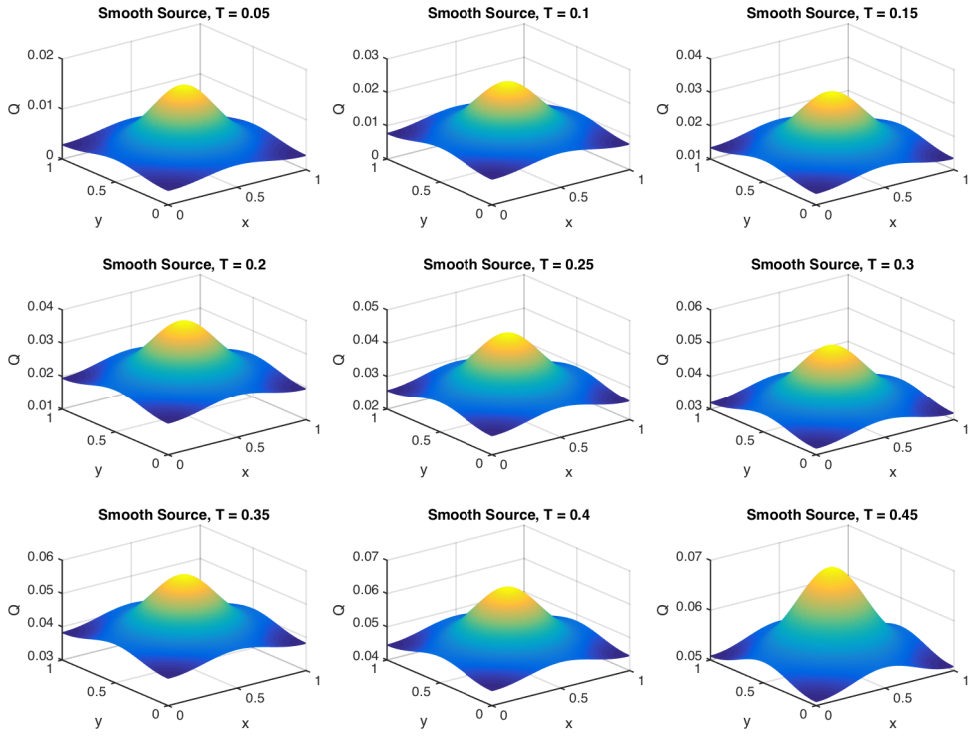

Figure 1: Smooth Source Solutions at Different Times

Table 1: Order of Convergence in h

	h=0.025	h=0.0125	0.00625
$log_2(R)$	1.9962	1.9990	1.9986

To get the order of convergence from R, we take the log base 2. This gives us

$$\begin{aligned} log_2(R_h) &= log_2\left(\frac{O(\Delta(h/2)^r) - O(\Delta h^r)}{O(\Delta(h/4)^r) - O(\Delta(h/2)^r)}\right) \\ &= log_2\left(\frac{2^{-r}O(\Delta h^r) - O(\Delta h^r)}{2^{-r}(2^{-r}O(\Delta h^r) - O(\Delta h^r))}\right) \\ &= log_2(2^r) \\ &= r \end{aligned}$$

We compute several values of $log_2(R_h)$ using different h values. We get the values in Table 1 from which we can infer that $r = 2$. We do the same thing in t with

$$R_t = \frac{O(\Delta(t/2)^p) - O(\Delta t^p)}{O(\Delta(t/4)^p) - O(\Delta(t/2)^p)}$$

and obtain the values in Table 2 from which we infer that $p = 1$. This values are in agreement with numerical theory because the derivative with respect to time is computed with the first order Euler scheme, which has first order convergence, and second order finite difference for the second derivative with respect to time, which has second order convergence.

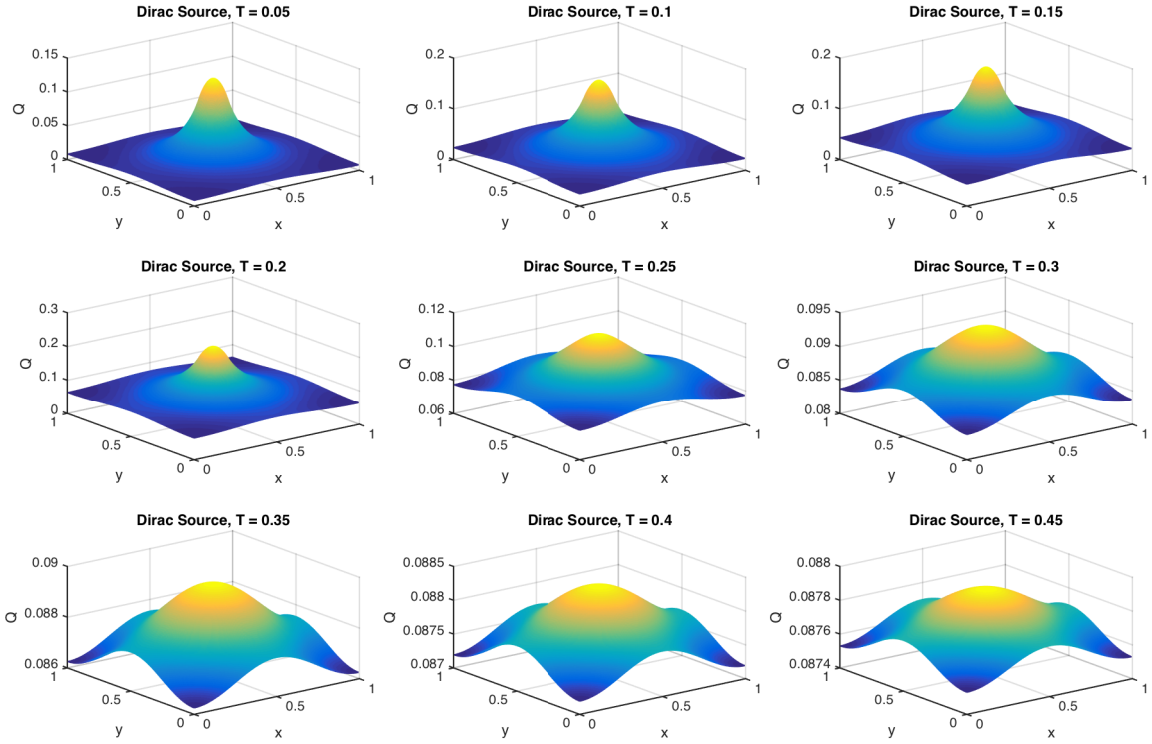

Figure 2: Dirac Source Solutions at Different Times

Table 2: Order of Convergence in t

	$t = 4 * 10^{-4}$	$t = 2 * 10^{-4}$	$t = 1 * 10^{-4}$
$log_2(R_t)$	1.0000	1.0000	1.0000

With the time variable source, we would expect the same r and p, because we would use the same numerical methods. We obtain the same p right away, although we do not obtain the same r value. This is a result of the source term being dependent on Δh, and therefore changing with each of the different Δh values we use. However, when solving for r if we intervene in the program to keep the ϵ value the same for the different values of h, we can obtain the expected quadratic convergence in space. With a fixed $\epsilon = \sqrt{\frac{1}{40}}$ we get $log_2(R_h) = 1.9978$ when $h = \frac{1}{160}$. All the convergence tests are included in `conv.m`, available at the end of the report. Because the ∞ and L^2 norms are not defined for the Dirac, we do not look at them for the error. We now investigate the effect of refining our approximation of the Dirac function. We do this by changing ϵ to $\alpha\sqrt{h}$ where $\alpha = 1, 0.1, 0.01, 0.0001$, and 10^{-10}. We include the following plots, Figure (3) and (4), with $\Delta h = 3.125 * 10^{-3}$ and $T = 0.145$ to show the effect of a more refined Dirac function.

As we can see, the peak becomes thinner and lower as α becomes smaller. We can see why this is when we look at the definition of the Dirac function. When α becomes smaller by a factor of 10, $\delta_\epsilon(r)$ increases by a factor of 10. But when doing this, we also cut down the accepted radius by a factor of 10. This radius defines an area that the heat source effects, which now has been but down by a factor of 100. This is why the peak becomes thinner and lower. Also, if α

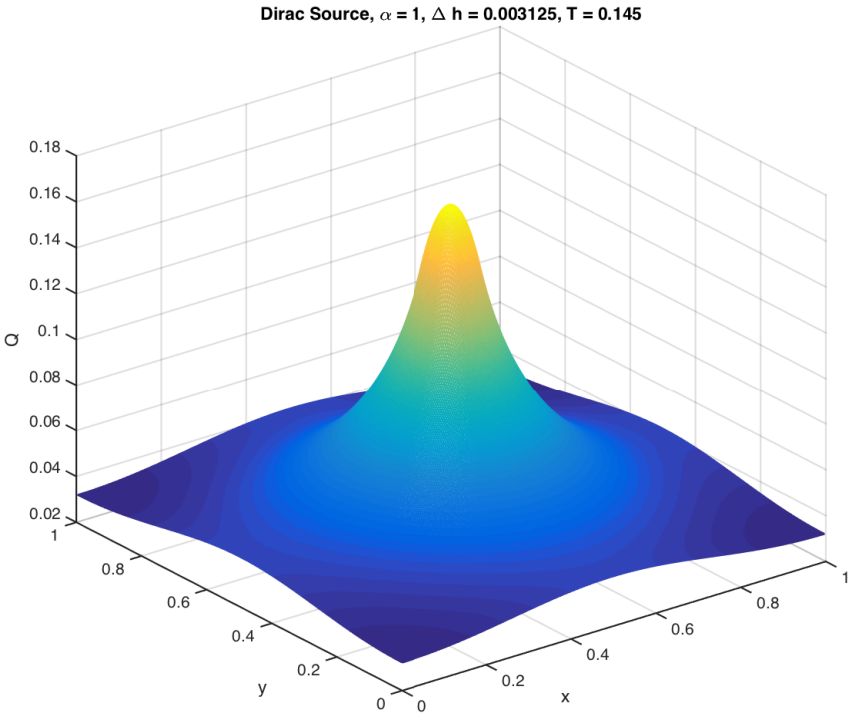

Figure 3: $\alpha = 1$

gets too small the discretization cannot pick up the heat source and we get a flat plot.

4.3 Conservative

In `conserv.m` we show that the method is conservative by looking at

$$\int q dx dy = \Delta x \Delta y \sum Q_{ij}$$

We need to show that

$$\int q dx dy = \int_0^T \int S dx dy dt \rightarrow$$
$$\Delta x \Delta y \sum Q_{ij} = \Delta x \Delta y \int_0^T \sum S_{ij}$$

Because S is a known function, we can easily compute the integral. For both cases, it is done in *conserv.m*. The function returns the absolute values of the difference between the heat in the system at $t = 2$ and the heat given to the system from $t = 0$ to $t = 2$.

We check for $n = m = 200$ and $h_t = 0.005$ and get the following values:

Source term	error
Smooth	$3.4506 * 10^{-13}$
Dirac	0.0022

In the first case, we can consider that the difference is zero. For the dirac function though, it seems that our method is not conservative. This is due to the fact that the heat source is not

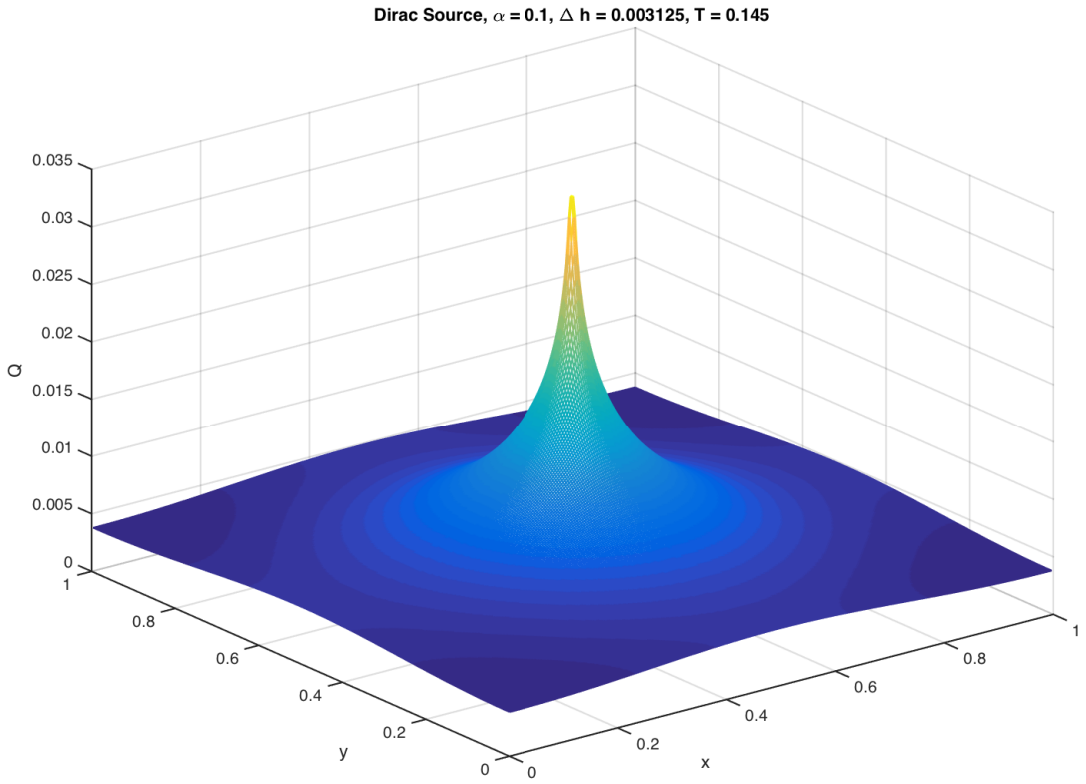

Figure 4: $\alpha = 0.1$

continuous. Therefore, we will get a bigger error while using implicit Euler. We can see that if we refine the time step, we get a smaller error. For example, for $m = n = 10$ and $ht = 0.00005$, we get an error of $8 * 10^{-4}$.

5 Refinements

5.1 Variable coefficients

In this part, we will change the equation a little and add variable coefficients so that the equation becomes :

$$q_t = a(y)q_{xx} + b(x)q_{yy} + S$$

5.1.1 Finite volume method

Using the same principles as in section 3, we get the following :

$$\frac{1}{\Delta x \Delta y} \iint_{(i,j)} q_t dx dy = \frac{1}{\Delta x \Delta y} \iint_{(i,j)} \nabla \cdot \begin{pmatrix} a(y)q_x \\ b(x)q_y \end{pmatrix} dx dy + \frac{1}{\Delta x \Delta y} \iint_{(i,j)} S dx dy$$

$$\frac{dQ_{ij}}{dt} = \frac{1}{\Delta x \Delta y}(\int_E a(y)q_x dy + \int_N b(x)q_y dx - \int_W a(y)q_x dy - \int_S b(x)q_y dx) + S_{ij}$$

Using finite differences to evaluate q_x and q_y, we get :

$$\frac{dQ_{ij}}{dt} = E\frac{aQ_{i+1,j} + bQ_{i,j} + cQ_{i-1,j}}{\Delta x^2} + F\frac{dQ_{i,j+1} + eQ_{i,j} + fQ_{i,j-1}}{\Delta y^2} + S_{ij}$$

Where a, b, c, d, e, f are the same coefficients as those defined in section 3.1. E and F take into account the variable coefficient and are defined by :

$$E = \frac{1}{\Delta y}\int_E a(y)dy = \frac{1}{\Delta y}\int_W a(y)dy$$

$$F = \frac{1}{\Delta x}\int_N b(x)dx = \frac{1}{\Delta x}\int_S b(x)dx$$

So, if we use the same notations as in section 3.1, the only thing we have to do is premultiply B and C with the corresponding matrix and after that, the implementation is unchanged. We can note that if $a(y) = b(x) = 1$ then E and F are identity matrices and we have the same rule as previously.

In this problem, we have always used the smooth source term. We also have used the trapezoidal rule to evaluate the integral. Let us define $A1$ the $n \times n$ matrix as :

$$A1 = diag(0.5(a(0) + a(\Delta y)), 0.5(a(\Delta y) + a(2\Delta y)), ..., 0.5(a((n-1)\Delta y) + a(1)))$$

Which is a diagonal matrix containing the different evaluations of the integral of a. With this matrix, we can build $B1$ as :

$$B1 = (A1 \otimes I(m)) * B$$

Identically, we can define $A2$ as :

$$A2 = diag(0.5(b(0) + b(\Delta x)), 0.5(b(\Delta x) + b(2\Delta x)), ..., 0.5(b((m-1)\Delta x) + b(1)))$$

And then :

$$C1 = (I(n) \otimes A2) * C$$

And finally :

$$M = B1 + C1$$

Once we have build M, the methodology is exactly the same as before. We get the following ODE's system :

$$\frac{dQvect}{dt} = MQvect + S$$

And we can use the same technique as in section 3.

The implementation of the method is done in $variableCoeff.m$.

5.1.2 Plots, convergence and conservation

The matlab code called $varCoeffConv.m$ plots the solution and returns the order of convergence for space and time.

For our implementation, we have used the following function a and b :

$$a(y) = y$$

$$b(x) = x$$

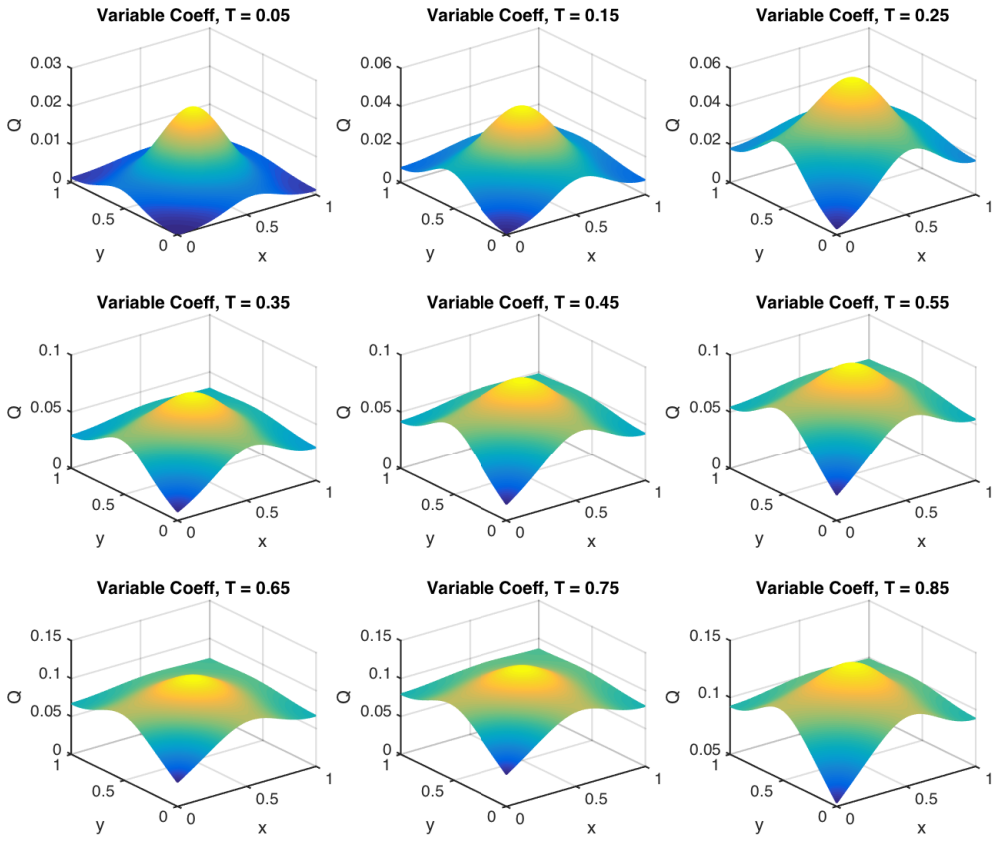

Figure 5: Results with Variable Coefficients

Figure (5) shows the solution for different times. We can see that at first the source term is dominant. After that, it is the conduction. The functions a and b can be seen as the conduction coefficients. The higher they are, the quicker the heat will travel in that direction. It is easily seen in the plots. Heat "has trouble" reaching $(0, 0)$ because there a and b are small. On the other end, where a and b are close to one, the conduction is really quick.

Let us now look at the convergence. The code used, $varCoeffConv.m$, is available at the end of the report.

First, the spatial convergence. Let us define :

$$R = log_2(\frac{Q_h - Q_{h/2}}{Q_{h/2} - Q_{h/4}})$$

Where Q_h is our solution evaluated at $(1, 1)$ (the last cell) at $t = 1$ for a grid such that $h = \Delta x = \Delta y$.

Our code gives the following values :

h	$\frac{1}{40}$	$\frac{1}{80}$	$\frac{1}{160}$
R	1.9953	1.9991	1.9999

We can see that the value seems to converge to 2. Hence, we can conclude that the order of spatial convergence is 2.

Let us now look at the time convergence. Let us define :

$$H = log_2(\frac{Q_{\Delta t} - Q_{\Delta t/2}}{Q_{\Delta t/2} - Q_{\Delta t/4}})$$

Where $Q_{\Delta t}$ is our solution evaluated at $(1,1)$ and $t = 0.4$ and $m = n = 200$ with the time step Δt. This yields the results :

Δt	0.08	0.04	0.02
H	1.0680	0.9884	0.9801

So we can conclude that the order of convergence is 1. This is to be expected since implicit Euler is used.

Let us finally look at the conservation. Our code returns the absolute value of the difference between the total heat given to the system by the heat source and the heat contained in the system at final time. Because the walls are insulated, those two values should be equal.

Running our code with $m = n = 400$, $\Delta t = 0.01$ and the final time being 0.5, we get that the difference $\Delta heat$ is :

$$\Delta heat = 5.6150 * 10^{-14}$$

Which is very close to zero and should be considered as such. We have thus verified that our scheme is numerically conservative.

5.2 Boundary conditions

We are now going to try and change the boundary conditions to see what happens. We have a Neuman condition on the left and right boundaries (just as before except it is no longer homogeneous). And we now have a Dirichlet condition on the upper and lower boundaries. Because in the previous grid, we had no point exactly on the boundary, we are going to shift the grid of $\frac{\Delta y}{2}$ in the y-direction. That way, we will have center of cells at $y = 0$ and $y = 1$. If we still keep $n\Delta y = 1$, that means that we will now have $m(n-1)$ unknowns since the value on the upper and lower boundaries are known.

5.2.1 Finite volume method

We use the same methodology as before. We rewrite Q as $Qvect$ and we will show that we will get a system of ODE's of the following type :

$$\frac{dQvect}{dt} = A_x Qvect + A_y Qvect + b_x + b_y + S$$

We already know the definition of S, it is unchanged. A_x and b_x represents the derivative in the x-direction (b_x is not zero because of the non homogeneous boundary conditions). Respectively, A_y and b_y take the y-derivative into account.

The x-derivative operator does not change but an additional term appears. Indeed, for a cell that is on the right boundary, we no longer no flux on the East edge but instead :

$$\int_E q_x dy = \int_E (-1) dy = -\Delta y$$

In the same way, for a cell on the left boundary, we have the following West flux :

$$-\int_W q_x dy = -\int_W (-1) dy = \Delta y$$

So, if we define T_x as the following $m \times m$ matrix :

$$T_x = \frac{1}{\Delta x^2} \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 1 & -2 & 1 \\ 0 & \cdots & 0 & 1 & -1 \end{pmatrix}$$

We can use kronecker products to express the x-derivative :

$$A_x = I(n-1) \otimes T_x$$

Where $I(n-1)$ is the $(n-1)\times(n-1)$ identity matrix. We have to take the non homogeneous conditions into account and so :

$$b_x = ones(n-1,1) \otimes \begin{pmatrix} -\frac{1}{\Delta x} \\ 0 \\ \vdots \\ 0 \\ \frac{1}{\Delta x} \end{pmatrix}$$

The work for the y-derivative is quite similar. But because we are only solving for the interior cells, we have "ghost cells" on the upper and lower boundary. This means that the $(n-1) \times (n-1)$ matrix T_y changes a bit :

$$T_y = \frac{1}{\Delta y^2} \begin{pmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 1 & -2 & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{pmatrix}$$

And A_y becomes :

$$A_y = T_y \otimes I(m)$$

To take the Dirichtlet conditions into account we have to use b_y defined by :

$$b_y = \begin{pmatrix} q(x_1,0,t) & \cdots & q(x_m,0,t) & 0 & \cdots & 0 & q(x_1,1,t) & \cdots & q(x_m,1,t) \end{pmatrix}^T$$

Where x_i means the value of x in the center of cell $(i,1)$ or $(i,n-1)$.
Finally, we can define $M = A_x + A_y$ and $S_{new} = S + b_x + b_y$, we get a system of the form :

$$\frac{dQvect}{dt} = MQvect + S_{new}$$

This system is of the same form as in section 3 so we can use the same method to implement implicit Euler.

5.2.2 Plots, convergence and conservation

The matlab code called *boundConv.m* plots the solution and returns the order of convergence for space and time.

We used the given boundary conditions and the results are given in figure (6). We can see the boundary conditions given at $y = 0$ and $y = 1$. We can also see the evolution of

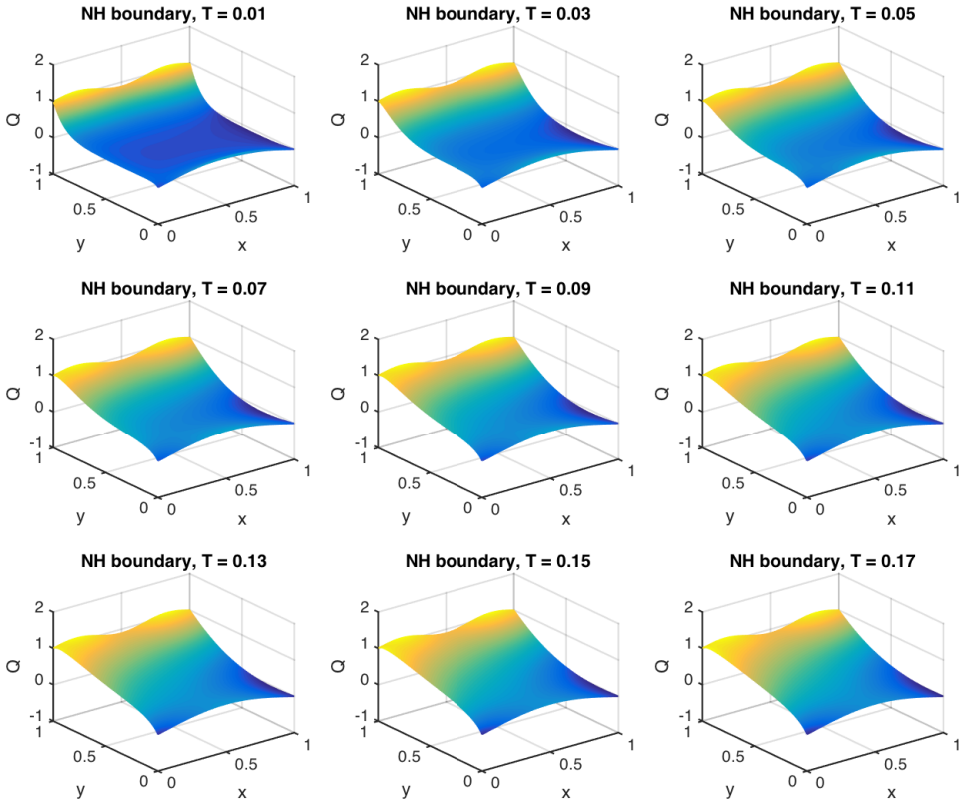

Figure 6: Results with Non-homogeneous B.C.'s

the temperature. First, the gradients are really steep near the boundaries because the initial condition is zero. Then, the conduction acts and the gradients become less steep.

Let us now look at the convergence. Let us define :

$$R = log_3(\frac{Q_h - Q_{h/3}}{Q_{h/3} - Q_{h/9}})$$

Where Q_h is our solution evaluated at $(0.95, 0.9)$ (it is no longer a good idea to use a boundary points because of the Dirichlet condition) at $t = 0.2$ for a grid such that $h = \Delta x = \Delta y$. We get the following table :

h	$\frac{1}{10}$	$\frac{1}{30}$	$\frac{1}{90}$
R	2.1555	2.0284	2.0033

Once again, we can conclude that we have an order 2 convergence in space.

Let us now look at the convergence. With the same definition for H as in section 5.1, we get the following table :

Δt	0.08	0.04	0.02
H	1.1255	1.0951	1.0587

We find back the order 1 convergence for implicit Euler.

Finally, a word about conservation. Let us first look at the left and right boundaries. There, $q_x = -1$ so the heat flux is the same at the left and right boundaries. However, at the upper and lower boundaries, we have Dirichlet conditions, meaning that there is also a heat flux at those boundaries but they are not equivalent. Thus, we no longer have conservation. It is easy to see on the plots. Even though the heat source is not zero, the temperature distribution looks like it is reaching a stationary state. The heat has thus to get out of the domain.

6 Matlab codes

```
function [Q,X,Y,Tax] = eulerImpl(m,n,ht,T,fct,alpha)
%EULERIMPL Solves the problem with n cells in the x-direction and m cells
%in the y-direction
hx = 1/m;
hy = 1/n;

%Construction of the stencil
e = ones(m,1);
M = spdiags([e -2*e e],-1:1,m,m);
M(1,1) = -1;
M(m,m) = -1;
A = kron(speye(n),M);
f = ones(n,1);
N = spdiags([f -2*f f],-1:1,n,n);
N(1,1) = -1;
N(n,n) = -1;
B = kron(N,speye(m));
C = A/(hx*hx)+B/(hy*hy);

%LU-decomposition
K = speye(n*m)-ht*C;
[L,U,P,V,R] = lu(K);

%Definition of the axis
X = (0.5:m-0.5)'*hx;
Y = (0.5:n-0.5)'*hy;
Tax = (0:T)'*ht;

x = kron(ones(n,1),X);
y = kron(Y,ones(m,1));
q = zeros(n*m,T+1);
Q = zeros(n,m,T+1);
%The heat source is the dirac function
if strcmp(fct,'dirac')
    ep = alpha*sqrt(max([hx hy]));
    r = sqrt((x-0.5).^2+(y-0.5).^2);
    delta = (1+cos(pi*r))./(2*ep).*(r<ep);
    for i = 1:T
        S = 2*delta*(i*ht < 0.25);
        q(:,i+1) = V*(U\(L\(P*((q(:,i)+ht*S)./diag(R)))));
        Q(:,:,i+1) = reshape(q(:,i+1),m,n)';
    end
%The heat source is the smooth function
else
    S = exp(-((x-0.5).^2+(y-0.5).^2)/0.04);
    for i = 1:T
        q(:,i+1) = V*(U\(L\(P*((q(:,i)+ht*S)./diag(R)))));
        Q(:,:,i+1) = reshape(q(:,i+1),m,n)';
    end
end

end
```

```
function [] = conv()
%CONV
% close all;
N = [40 80 160 320 640];
q = zeros(1,length(N));

for i = 1:length(N)
    Q = eulerImpl(N(i),N(i),0.05,30,'smooth');
    q(i) = Q(1,1,30);
end
orderSpace1 = log2(abs(q(1)-q(2))/abs(q(2)-q(3)))
orderSpace2 = log2(abs(q(2)-q(3))/abs(q(3)-q(4)))
orderSpace3 = log2(abs(q(3)-q(4))/abs(q(4)-q(5)))

ht = [0.0004 0.0002 0.0001 0.00005 0.000025];
for i = 1:length(ht)
    T = round(0.3/ht(i));
    Q = eulerImpl(50,25,ht(i),T,'dirac');
    q(i) = Q(1,1,T);

end
orderTemp1 = log2(abs(q(1)-q(2))/abs(q(2)-q(3)))
orderTemp2 = log2(abs(q(2)-q(3))/abs(q(3)-q(4)))
orderTemp3 = log2(abs(q(3)-q(4))/abs(q(4)-q(5)))

end
```

```
function [consSmooth,consDirac] = conserv()
%CONSERV This function checks conservation for both heat source
m =10;
n = 10;
ht = 0.00005;

%Conservation for smooth source
[Q,X,Y,T] = eulerImpl(m,n,ht,2/ht,'smooth',1);
x = kron(ones(n,1),X);
y = kron(Y,ones(m,1));
S = exp(-((x-0.5).^2+(y-0.5).^2)/0.04);
heatGiven = 2*sum(S)/(m*n);
heatSys = sum(sum(Q(:,:,end)))/(m*n);
consSmooth = abs(heatGiven-heatSys);

%Conservation for dirac source
[Q,X,Y,T] = eulerImpl(m,n,ht,2/ht,'dirac',1);
ep = sqrt(max([1/m 1/n]));
r = sqrt((x-0.5).^2+(y-0.5).^2);
delta = (1+cos(pi*r))./(2*ep).*(r<ep);
S = 2*delta;
heatGiven = 0.25*sum(S)/(m*n);
heatSys = sum(sum(Q(:,:,end)))/(m*n);
consDirac = abs(heatGiven-heatSys);

end
```

```
function [ ] = plots4_1( )
% Make plots for problem 4.1
m = 200;
n = 200;
[Q,X,Y,Tax] = eulerImpl(m,n,0.05,10,'dirac');
figure(1)
for i=2:10
    subplot(3,3,i-1)
    mesh(X,Y,Q(:,:,i))
    title(sprintf('Dirac Source, T = %g', Tax(i)))
    xlabel('x')
```

```
    ylabel('y')
    zlabel('Q')
end

[Q,X,Y,Tax] = eulerImpl(m,n,0.05,10,'other');
figure(2)
for i=2:10
    subplot(3,3,i-1)
    mesh(X,Y,Q(:,:,i))
    title(sprintf('Smooth Source, T = %g', Tax(i)))
    xlabel('x')
    ylabel('y')
    zlabel('Q')
end
end
```

```
function [Q,X,Y,Tax,conservCheck] = variableCoeff(m,n,ht,T)
%VARIABLECOEFF We here implement a solver where a(y) and b(x) can be any
%smooth positive function. For the source term, we always use the smooth
%function.
%The output conservCheck the value of the difference between the heat
%given to the system by the heat source and the heat contained in the
%system at final time

hx = 1/m;
hy = 1/n;

%Definition of the axis
X = (0.5:m-0.5)'*hx;
Y = (0.5:n-0.5)'*hy;
Tax = (0:T)'*ht;

%Construction of the stencil
e = ones(m,1);
M = spdiags([e -2*e e],-1:1,m,m);
M(1,1) = -1;
M(m,m) = -1;
A = kron(speye(n),M);
f = ones(n,1);
N = spdiags([f -2*f f],-1:1,n,n);
N(1,1) = -1;
N(n,n) = -1;
B = kron(N,speye(m));
%This part takes a(y) and b(x) into account!
A1 = a(0:hy:1);
A1 = (A1(1:end-1)+A1(2:end))/2;
A1 = kron(diag(A1),speye(m));
B1 = b(0:hx:1);
B1 = (B1(1:end-1)+B1(2:end))/2;
B1 = kron(speye(n),diag(B1));

C = A1*A/(hx*hx)+B1*B/(hy*hy);


%This part is the same as in eulerImpl!
%LU-decomposition
K = speye(n*m)-ht*C;
[L,U,P,V,R] = lu(K);

x = kron(ones(n,1),X);
y = kron(Y,ones(m,1));
q = zeros(n*m,T+1);
Q = zeros(n,m,T+1);

%The heat source is the smooth function
S = exp(-((x-0.5).^2+(y-0.5).^2)/0.04);
for i = 1:T
    q(:,i+1) = V*(U\(L\(P*((q(:,i)+ht*S)./diag(R)))));
```

```
    Q(:,:,i+1) = reshape(q(:,i+1),m,n)';
end

%We check conservation
%The total heat given to the system is :
heatGiven = hx*hy*sum(S)*T*ht;
%The total heat in the system is :
heatSys = hx*hy*sum(sum(Q(:,:,end)));
conservCheck = abs(heatGiven-heatSys);

end

%Definition of a(y) and b(x)
function A = a(y)
A = y;
end
function B = b(x)
B = x;
end
```

```
function [orderSpace,orderTime] = varCoeffConv()
%VARCOEFF This function computes the order of convergence for space and
%time for the equation with variable coefficients. It also plots the
%solution at different times for the most refined mesh.
close all;

%Space convergence
N = [40 80 160 320 640];
q = zeros(size(N));
n = length(N);
for i = 1:n;
    [Q,X,Y,T] = variableCoeff(N(i),N(i),0.05,20);
    q(i) = Q(end,end,end);
end
orderSpace = log2(abs(q(1:n-2)-q(2:n-1))./abs(q(2:n-1)-q(3:n)));

%Time convergence
ht = [0.08 0.04 0.02 0.01 0.005];
tend = 5*ht(1);
p = zeros(size(ht));
n = length(ht);
for i = 1:n
    [P,x,y,t] = variableCoeff(200,200,ht(i),round(tend/ht(i)));
    p(i) = P(end,end,end);
end
orderTime = log2(abs(p(1:n-2)-p(2:n-1))./abs(p(2:n-1)-p(3:n)));

%Plot of the solution
figure;
for i = 1:9;
    subplot(3,3,i);
    mesh(X,Y,Q(:,:,2*i));
    title(sprintf('Variable Coeff, T = %g', T(2*i)));
    xlabel('x');
    ylabel('y');
    zlabel('Q');
end
end
```

```
function [Q,X,Y,Tax] = bound(m,n,ht,T)
%BOUND We implement a solver for the given non homogenous boundary
%conditions given. The solution is computed for the interior points
%(meaning that we have m(n-1) unknows). The source term used is always the
%smooth one.
N = n-1;
hx = 1/m;
```

```
hy = 1/n;

%Definition of the axis
X = (0.5:m-0.5)'*hx;
Y = (1:N)'*hy; %the y-axis is shifted by half a cell
Tax = (0:T)'*ht;

%Construction of the stencil
e = ones(m,1);
M = spdiags([e -2*e e],-1:1,m,m);
M(1,1) = -1;
M(m,m) = -1;
A = kron(speye(N),M);
f = ones(n,1);
O = spdiags([f -2*f f],-1:1,N,N);
B = kron(O,speye(m));

C = A/(hx*hx)+B/(hy*hy);

%LU-decomposition
K = speye(N*m)-ht*C;
[L,U,P,V,R] = lu(K);

x = kron(ones(N,1),X);
y = kron(Y,ones(m,1));
q = zeros(N*m,T+1);
Q = zeros(N,m,T+1);

%The heat source is the smooth function
S = exp(-((x-0.5).^2+(y-0.5).^2)/0.04);
%We have to also add b1 and b2 to S to keep boundary conditions
b1 = kron(ones(N,1),[1 spalloc(1,m-2,0) -1]'/hx);
b2 = [southBound(X);spalloc(m*(N-2),1,0);northBound(X)]/(hy*hy);
S = S+b1+b2;
for i = 1:T
    q(:,i+1) = V*(U\(L\(P*((q(:,i)+ht*S)./diag(R)))));
    Q(:,:,i+1) = reshape(q(:,i+1),m,N)';
end

end

function q0 = southBound(x)
q0 = sin(pi*x)/pi;
end
function q1 = northBound(x)
q1 = sin(3*pi*x)/(3*pi)+1;
end
```

```
function [orderSpace,orderTime] = boundConv()
%BOUNDCONV This function computes the order of convergence in space and
%time for the problem with non homogeneous boundary conditions. It also
%plots the solution at different times for the most refined mesh.
close all;

%Space convergence
N = [10 30 90 270 810];
q = zeros(size(N));
n = length(N);
for i = 1:n;
    [Q,X,Y,T] = bound(N(i),N(i),0.01,20);
    I = (abs(X-0.95)< eps);
    J = (abs(Y-0.9) < eps);
    q(i) = Q(J,I,end);
end
orderSpace = log(abs(q(1:n-2)-q(2:n-1))./abs(q(2:n-1)-q(3:n)))/log(3);

%Time convergence
ht = [0.08 0.04 0.02 0.01 0.005];
```

```
tend = 5*ht(1);
p = zeros(size(ht));
n = length(ht);
for i = 1:n
    [P,x,y,t] = bound(200,200,ht(i),round(tend/ht(i)));
    p(i) = P(100,75,end);
end
orderTime = log2(abs(p(1:n-2)-p(2:n-1))./abs(p(2:n-1)-p(3:n)));

%Plot of the solution
figure;
for i = 1:9;
    subplot(3,3,i);
    mesh(X,Y,Q(:,:,2*i));
    title(sprintf('NH boundary, T = %g', T(2*i)));
    xlabel('x');
    ylabel('y');
    zlabel('Q');
end

end
```