# Partial differential equations

Mariel Aulie Hinderaker

December 2019

FYS3150 - Project 5

## 1 Abstract

The three finite difference methods forward Euler, backward Euler and Crank-Nicolson are studied in this project. The latter method is the best approximation, as it has smaller truncation error. The algorithms shows that the function u approaches a linear curve as t $\rightarrow \infty$.

## 2 Introduction

This project concerns partial differential equations, which in many cases can be impossible to solve. The solution is then to approach the impossible solution using other numerical methods. There are some different methods for solving this, and one of the most used is called finite difference methods because it is relatively simple to implement and understand. In this project three finite difference methods will be used: the forward Euler method, the backward Euler method and the Crank-Nicolson method. These will be explored in a 1+1 dimension and a 2+1 dimension, and the methods will be compared to one another.

1- dimensional parabolic equation

## 3 Methods

For all the three algorithms it is assumed that the density u obeys the Gauss-Greens theorem, $u_{xx} = u_t$. They are also given initial conditions and boundary conditions:

$$u(0,t) = a(t) = 0, \ for \ t \geq 0 \tag{1}$$

$$u(L,t) = b(t) = 1, \ for \ t \geq 0 \tag{2}$$

$$u(x,0) = g(x) = 0, \ for \ 0 < x < L \tag{3}$$

The algorithms for the three different finite difference methods is:

Forward Euler:

$$u_t \approx \frac{u(x,t+\Delta t)-u(x,t)}{\Delta t} = \frac{u(x_i,t_j+\Delta t)-u(x_i,t_j)}{\Delta t}$$

$$u_{xx} \approx \frac{u(x_i+\Delta x,t_j)-2u(x_i,t_j)+u(x_i-\Delta x,t_j)}{\Delta x^2} \tag{4}$$

Solving the algorithm for the forward Euler method leads to an explicit scheme.

$$\frac{u(x_i,t_j+\Delta t)-u(x_i,t_j)}{\Delta t} = \frac{u(x_i+\Delta x,t_j)-2u(x_i,t_j)+u(x_i-\Delta x,t_j)}{\Delta x^2}$$

Which can be simplified as followed:

$$\frac{u_{i,j+1}-u_{i,j}}{\Delta t} = \frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\Delta x^2}$$

$$u_{i,j+1} = \frac{\Delta t}{\Delta x^2}(u_{i+1,j}-2u_{i,j}+u_{i-1,j})+u_{i,j}$$

An $\alpha$-variable can be defined:
Defining

$$\alpha = \frac{\Delta t}{\Delta x^2} \tag{5}$$

$$u_{i,j+1} = \alpha u_{i-1,j}+(1-2\alpha)u_{i,j}+\alpha u_{i+1,j} \tag{6}$$

Equation 6 is the equation for the forward Euler to implement.

Backward Euler:

Solving the backward Euler method resolves in an implicit scheme.

$$u_t \approx \frac{u(x,t)-u(x,t-\Delta t)}{\Delta t} = \frac{u(x_i,t_j)-u(x_i,t_j-\Delta t)}{\Delta t}$$

$$u_{xx} \approx \frac{u(x_i+\Delta x,t_j)-2u(x_i,t_j)+u(x_i-\Delta x,t_j)}{\Delta x^2} \tag{7}$$

$$\frac{u_{i,j}-u_{i,j-1}}{\Delta t} = \frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\Delta x^2}$$

Using the definition of 5 *alpha*:

$$u_{i,j-1} = -\alpha u_{i-1,j}+(2\alpha-1)u_{i,j}-\alpha u_{i+1,j} \tag{8}$$

Equation 8 can also be written as a tridiagonal matrix.

$$A = \begin{vmatrix} (2\alpha-1) & -\alpha & 0 & 0 & ... \\ -\alpha & (2\alpha-1) & -\alpha & 0 & ... \\ ... & ... & ... & ... & ... \\ 0 & ... & -\alpha & (2\alpha-1) & -\alpha \\ 0 & ... & ... & -\alpha & (2\alpha-1) \end{vmatrix}$$

For the implicit scheme yields:

$$V_j = A^{-1}V_{j-1} = ... = A^{-j}V_0$$

Crank-Nicolson scheme:

This method combines both explicit and implicit schemes and bisect the equation.

$$u_t \approx \frac{u(x,t+\Delta t)-u(x,t)}{\Delta t} = \frac{u(x_i,t_j+\Delta t)-u(x_i,t_j)}{\Delta t}$$

$$u_{xx} \approx \frac{1}{2}(\frac{u(x_i+\Delta x,t_j)-2u(x_i,t_j)+u(x_i-\Delta x,t_j)}{\Delta x^2}$$
$$+\frac{u(x_i+\Delta x,t_j+\Delta t)-2u(x_i,t_j+\Delta t)+u(x_i-\Delta x,t_j+\Delta t)}{\Delta x^2}) \qquad (9)$$

$$\frac{1}{2\Delta x^2}(u_{i+1,j}-2u_{i,j}+u_{i-1,j}+u_{i-1,j-1}+2u_{i,j-1}+u_{i+1,j-1}) = \frac{1}{\Delta t}(u_{i,j}-u_{i,j-1})$$

Using equation 5:

$$-\alpha u_{i-1,j}+(2\alpha+2)u_{i,j}-\alpha(u_{i+1,j} = \alpha u_{i-1,j-1}+(2-2\alpha)+\alpha u_{i+1,j-1}$$

Which can also be written as

$$(2I+\alpha B)V_j) = (2I-\alpha B)V_{j-1}$$

where B is

$$B = \begin{vmatrix} 2 & -1 & 0 & 0 & ... \\ -1 & 2 & -1 & 0 & ... \\ ... & ... & ... & ... & ... \\ 0 & ... & -1 & 2 & -1 \\ 0 & ... & ... & -1 & 2 \end{vmatrix}$$

This resolves in a definition of $V_j$:

$$V_j = (2I+\alpha B)^{-1}(2I-\alpha B)V_{j-1}$$

For the forward Euler method, the local truncation error is O(h2). Hence, the method is referred to as a first order technique. In general, a method with O(k+1) truncation error is said to be of kth order.

The truncation error is defined in 10

$$y_{i+1} = y(t_i) + h(y_i' + ... + y^p(t_i)\frac{h^{(p-1)}}{p!}) + O(h^p) \tag{10}$$

For the Forward Euler 4 and Backward Euler it is easy to see that for $u_t$ the equation yields for the first derivative and the truncation error is therefore $O(h^{p=1}) = O(h)$. The equation for $u_x$ contains in both cases of first order and the truncation error is therefor $O(h^{p=2} = O(h^2))$. Since the Crank-Nicolson method is a combination of these two we have to approach the truncation error in a different way. The Crank-Nicolson Scheme could be Taylor expanded, which would show that the truncation errors is O($\Delta x^2$) and O($\Delta t^2$).

The Forward Euler approximation is only stable if $\alpha$ is smaller or equal to 1/2 while the Backward Euler and the Crank-Nicolson approximation is stable for all values of $\Delta t$ and $\Delta x$. The stability is limited by the spectral radius condition:

$$\rho(A) < 1 \tag{11}$$

$$\rho(A) max|\lambda| : det(A - \lambda I) = 0$$

This is achieved if the matrix A is positive definite, if all the eigenvalues to A is positive. This is the case for the Backward Euler and the Crank-Nicolson approximation, but not for the Forward Euler method.

The matrix A **??** is possible to rewrite as: $A = I - \alpha B$.

$$B = \begin{vmatrix} 2 & -1 & 0 & 0 & ... \\ -1 & 2 & -1 & 0 & ... \\ ... & ... & ... & ... & ... \\ 0 & ... & -1 & 2 & -1 \\ 0 & ... & ... & -1 & 2 \end{vmatrix} \tag{12}$$

The eigenvalues of B is $my_i$, and the eigenvalues of A is $\lambda_i = 1 - \alpha\mu_i$. The matrix elements in B can be written as:

$$b_{i,j} = 2\delta_{i_j} - \delta_{i+1,j} - \delta_{i-1,j}$$

It is known that $(Bx)_i = \mu_i x_i$, which can be written in terms of the matrix elements:

$$(2\delta_{i_j} - \delta_{i+1,j} - \delta_{i-1,j})x_j = \mu_i x_i$$

and if it could be assumed that x can be written as a sinus range, it would resolve in

$$2sin(i\theta) - sin((i+1)\theta) - sin((i-1)\theta) = \mu_i sin(i\theta)$$

Hence, the eigenvalues of B is $\mu_i = 2 - 2cos(\theta)$, and testing this for the spectral radius conditions 11 resolves in the stability condition:

$$-1 < 1 - \alpha\mu_i < 1$$
$$-1 < 1 - \alpha 2 - 2cos(\theta) < 1$$
$$\alpha < \frac{1}{1 - cos(\theta)}$$

Which means that the algorithm is only stable for $\alpha \leq 1/2$
The analytical solution in one dimension of the following PDE:

$$\frac{\partial^2 u(x,y)}{\partial^2 x} = \frac{\partial u(x,y,t)}{\partial t}, t > 0, x \in [0,1] \tag{13}$$

A general solution can be written as:

$$u(x,t) = cx + F(x)G(t) \tag{14}$$

Equation 13 can be rewritten, whereas both sides equals the same constant (naming it $-\lambda^2$ for this equation):

$$\frac{F''}{F} = \frac{G'}{G} = -\lambda^2$$

This resolves in the two different equations:

$$F'' + F\lambda^2 = 0$$
$$G' + G\lambda^2 = 0$$

For these two differential equations the general solutions are:

$$F(x) = Asin(\lambda x) + Bcos(\lambda x)$$
$$G(t) = Ce^{-\lambda^2 t}$$

From the boundary conditions we see that B is zero and that $\lambda = n\pi/L$we can now rewrite equation 14:

$$u(x,t) = cx + A_n sin(n\pi x/L)e^{-n^2\pi^2 t/L^2}$$

Which is more correctly written as the sum of all the different values, as a superposition.

$$u(x,t) = cx + \sum_{n=1}^{\infty} A_n sin(n\pi x/L)e^{-n^2\pi^2 t/L^2} \tag{15}$$

The boundary condtion 2 provides a possibility to find the constant c.

$$1 = cL + \sum_{n=1}^{\infty} A_n sin(n\pi L/L)e^{-n^2\pi^2 t/L^2}$$

$\sin(n\pi) = 0$ for n $\in Z$

$$1 = cL \;\longrightarrow\; c = \frac{1}{L}$$

What remains is to solve for $A_n$, which is doable when t=0, using the initial condition, equation 3.

$$0 = \frac{x}{L} + \sum_{n=1}^{\infty} A_n sin(n\pi x/L)$$

$$-\frac{x}{L} = \sum_{n=1}^{\infty} A_n sin(n\pi x/L)$$

$$A_n = \frac{2}{L}\int_0^L g(x) sin(n\pi x/L)dx$$

Since g(x) is always constant, so it can be put out of the integral.

$$A_n = \frac{2g(x)}{L}\int_0^L sin(n\pi x/L)dx$$

$$= -\frac{2}{L}\frac{L}{n\pi}cos(n\pi x/L)\Big|_0^L$$

$$A_n = (-1)^n \frac{2}{n\pi}$$

Equation 15 can thus be rewritten with the known constants:

$$u(x,t) = \frac{x}{L}\sum_{n=1}^{\infty}(-1)^n \frac{2}{n\pi} sin(n\pi x/L) e^{-n^2\pi^2 t/L^2}$$

$$u(x,0) = g(x) = \frac{x}{L} + \sum_{n=1}^{\infty}(-1)^n \frac{2}{n\pi} sin(n\pi x/L)$$

Moving to two dimension gives another differential equation:

$$\frac{\partial^2 u(x,y,t)}{\partial^2 x} + \frac{\partial^2 u(x,y,t)}{\partial^2 y} = \frac{\partial u(x,y,t)}{\partial t}, t > 0, x, y \in [0,1] \tag{16}$$

For the explicit scheme the equation is extended with an y-variable so the equations needed is as followed (in the discretized verison):

$$u_{xx} \approx \frac{u^l_{i+1,j} - 2u^l_{i,j} + u^l_{i-1,j}}{\Delta x^2}$$

$$u_{yy} \approx \frac{u^l_{i,j+1} - 2u^l_{i,j} + u^l_{i,j-1}}{\Delta y^2}$$

$$u_t \approx \frac{u_{i,j}^{l+1} + u_{i,j}^{l}}{\Delta t}$$

We assume an equal number of mesh points for x and y and we have: $x_i = x_0 + ih$ and $y_j = y_0 + *jh$.

If we now solve equation 16 for $u_{i,j}^{l+1}$:

$$u_{i,j}^{l+1} \approx u_{i,j}^{l} + \alpha[u_{i+1,j}^{l} + u_{i-1,j}^{l} + u_{i,j+1}^{l} + u_{i,j-1}^{l} - 4u_{i,j}^{l}] \tag{17}$$

In two dimensions it is convenient to know about the Laplace and the Poissons rule, respectively:

$$\nabla^2 u(x) = u_{xx} + u_{yy} = 0$$

$$u_{xx} + u_{yy} = -\rho(x, y)$$

The value h is set to be $h = \Delta x = \Delta y = \frac{L}{n+1}$.

The equation needed to solve is:

$$u_{i,j} = \frac{1}{4}[u_{i,j+1} + u_{i,j-1} + u_{i+1,j} + u_{i-1,j}] + h^2 \rho_{i,j}$$

To simplify, a value $\rho^{\sim}$ is defined as $\rho^{\sim} = h^2 \rho$.

$$4u_{i,j} - [u_{i,j+1} + u_{i,j-1} + u_{i+1,j} + u_{i-1,j}] = -\rho^{\sim}$$

This leads to the following equations:

$$\begin{aligned}
4u_{1,1} - u_{1,2} - u_{1,0} - u_{2,1} - u_{0,1} &= -\rho_{1,1}^{\sim} \\
4u_{1,2} - u_{1,3} - u_{1,1} - u_{2,2} - u_{0,2} &= -\rho_{1,2}^{\sim} \\
4u_{2,1} - u_{2,2} - u_{2,0} - u_{3,1} - u_{1,1} &= -\rho_{2,1}^{\sim} \\
4u_{2,2} - u_{2,3} - u_{2,1} - u_{3,2} - u_{1,2} &= -\rho_{2,2}^{\sim}
\end{aligned} \tag{18}$$

The boundary conditions are given:

$$u_{i,0} = u_{i,L} = g_{i,0}, 0 \le i \le n + 1$$

and

$$u_{0,j} = g_{0,j}$$

,

$$u_{L,j} = g_{L,j}$$

The equations in 18 can therefore be written as a matrix equation Ax = b, taking into account for the known boundary conditions in the right side of the equation.

$$\begin{vmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{vmatrix} \begin{vmatrix} u_{1,1} \\ u_{1,2} \\ u_{2,1} \\ u_{2,2} \end{vmatrix} = \begin{vmatrix} u_{0,1} + u_{1,0} - \rho_{1,1}^{\sim} \\ u_{1,3} + u_{0,2} - \rho_{1,2}^{\sim} \\ u_{2,0} + u_{3,1} - \rho_{2,1}^{\sim} \\ u_{2,3} + u_{3,2} - \rho_{2,2}^{\sim} \end{vmatrix} \tag{19}$$

The calculation of the closed form expression in two dimensions is very similar to the calculation in one dimension. The initial condition is:

$$u(x,y,0) = f(x,y),\ \ x,y \in (0,L)$$
$$f(x,y) = sin(\frac{\pi x}{L})sin(\frac{\pi y}{L})$$

and the boundary contions are:

$$u(0,y,t) = 0,\ \ t > 0$$
$$u(L,y,t) = 0,\ \ t > 0$$
$$u(x,0,t) = 0,\ \ t > 0$$
$$u(x,L,t) = 0,\ \ t > 0$$

Equation 16 has the general solution

$$u(x,y,t) = F(x,y)G(t)$$

$$\frac{\nabla^2 F(x,y}{F(x,y} = \frac{\partial G(t)}{G(t)} = -\lambda^2$$

$$F(x,y) = H(x)R(y)$$

$$H(x) = Acos(kx) + Bsin(kx)$$
$$R(y) = Ccos(qy) + Dsin(qy)$$

The boundary conditions erases the cosinus element in each equation, which leads to:

$$F_{mn}(x,y) = A_n sin(\frac{n\pi x}{L})sin(\frac{m\pi y}{L})$$

$$G_{mn}(t) = e^{\gamma}$$
$$\gamma = \frac{2\pi^2 t}{L^2}$$

Finally the analytical equation is:

$$u(x,y,t) = sin(\frac{n\pi x}{L})sin(\frac{m\pi y}{L}e^(\frac{2\pi^2 t}{L^2})$$

## 4 Results and discussion

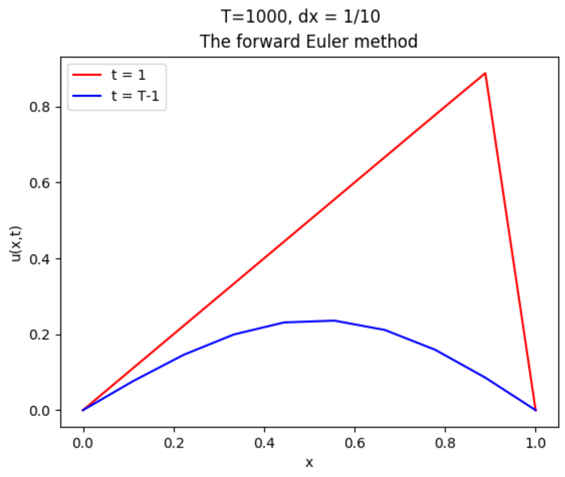

Figure 1: The forward Euler method with dx=1/10

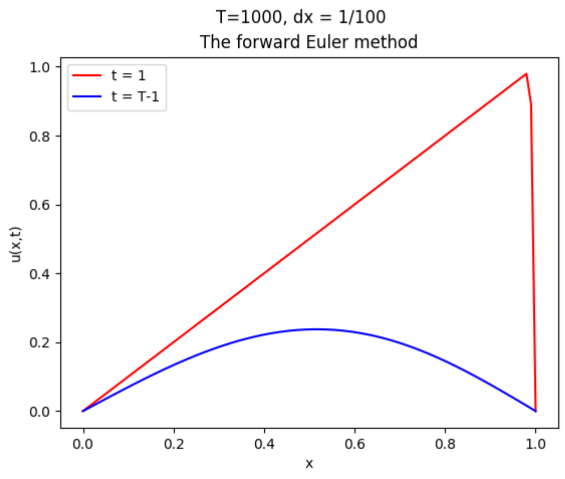

Figure 2: The forward Euler method with dx=1/100

In general, the more mesh points, the better results. The curved u(x,1) is smoother in the figure where dx = 1/100 compared to dx = 1/10. It also shows that for the forward euler method the u(x,T-1) graph goes further on the x-axis before the slope gets negative.

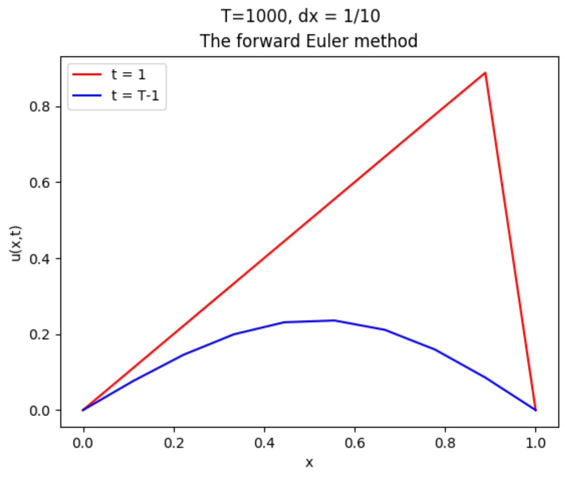

Figure 3: The forward Euler method with dx=1/10, and dt=0.001. The stability condition is not obeyed.

Figure 3 shows a calculation where the stability condition is exceeded.

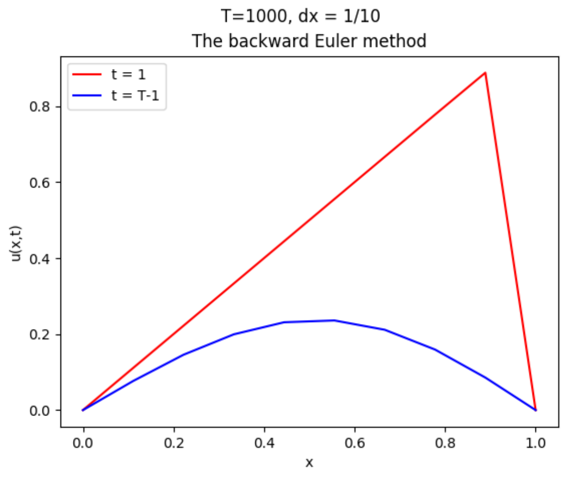

Figure 4: The backward Euler method with dx=1/10

For the backward Euler method the difference in where the slope changes is not as significant.

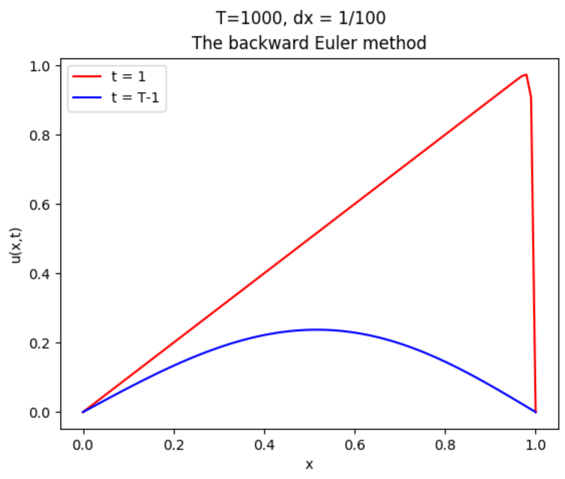

Figure 5: The backward Euler method with dx=1/100

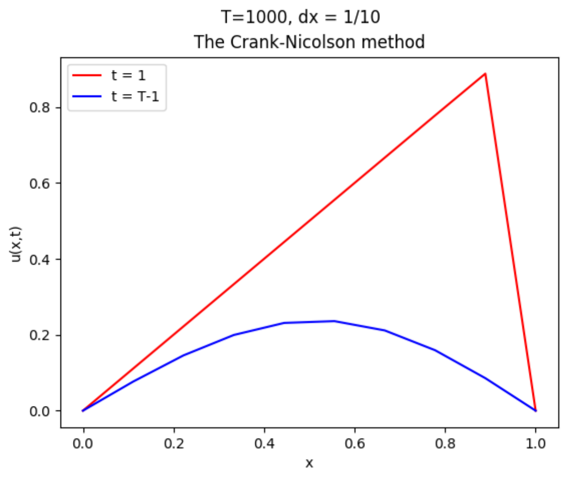

Figure 6: The Crank-Nicolson method with dx=1/10

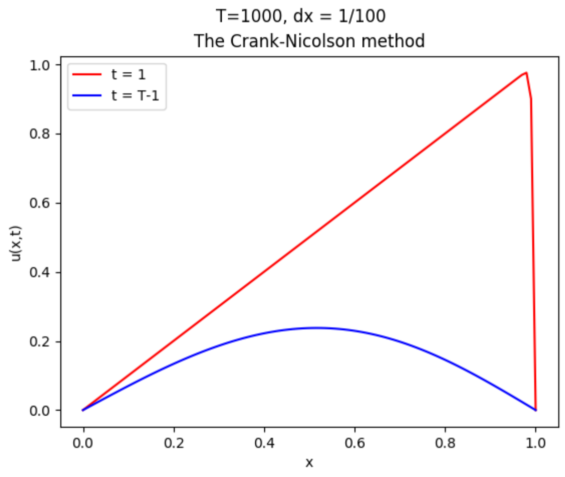

Figure 7: The Crank-Nicolson method with dx=1/100

There is a bit more difference in the Crank-Nicolson scheme, which is natural as the method is a combination of the backward and the forward euler method. For this method the error is smaller because it is of $O(\Delta t^2)$, whereas the other methods are of$O(\Delta t)$.

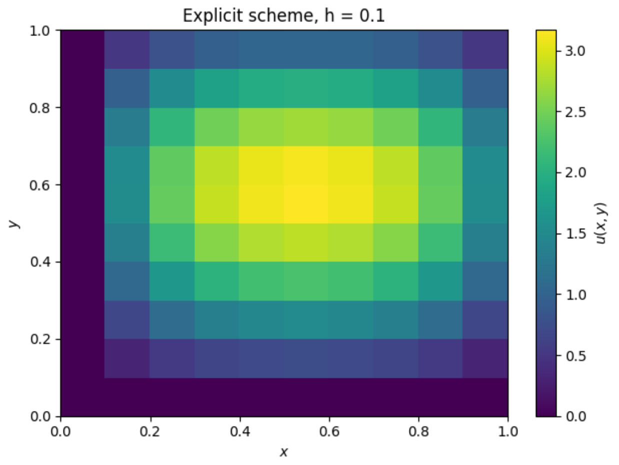

Figure 8: Two dimensional equation with h = 1/10

Figure 8 shows a diagram of u(x,y,0), iterated approximately 14000 times to reach the approved convergence tolerance. There were some issues implementing the time-loop, t = 0 in this diagram. There were also some issues outputting files, which is why there are so few mesh points.

## 5 Conclusion

The finite difference methods that has been discussed in this project are all good approximations, and it is obvious that as $u(x, t \to \infty)$ the slope gets more and more linear. The best of the three methods is the Crank-Nicolson method, because it has smaller truncation error. The Crank-Nicolson method and the backward Euler method might be easier to implement sometimes because they are not limited by the spectral radius condition. The forward Euler method however is.

## 6 Improvements

This is a very time-consuming project, and if I were to have more time there are a couple of tasks I would have improved. I would have calculated the analytical values in one and two dimension for a certain x, (y), and t to compare it with the numerical solution for the same variable values. To make even better plots I would have tried to animate the plots in time.

## List of Figures

## 7

## References

[1] Morten Hjorth-Jensen. *Computational Physics. Lecture Notes Fall 2015.* Department of Physics, University of Oslo, 2015.

[2] Morten Hjorth-Jensen. *Computational Physics Lectures: Partial differential equations.*
`http://compphysics.github.io/ComputationalPhysics/doc/pub/pde/html/pde.html`

[3] Department of physicsm, University of Oslo. *Project 5. Partial differential equations.*
`http://compphysics.github.io/ComputationalPhysics/doc/Projects/2019/Project5/DiffusionEqu`

[4] Granville Sewell *The Numerical solution of Ordinary and Partial Differential Equations, second edition.*