Project 5 - Partial Differencial Equation

Solveig Andrea Devold Fjeld and Sarah Rastad

December 13, 2016

Abstract

1 Introduction

2 Theory

LEGG TIL TEORI PÅ 3D PDE OG BRUK LABEL eq:partDIFF3D VET IKKE HVOR JEG SKAL SETTE LIGNINGEN

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

2.1 Equation

In this project we are solving the partial differencial equation:

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

which can also be written

$$u_{xx} = u_t \tag{3}$$

This partial differencial equation can be seen as the temperature gradient in a rod of lenght L. This equation can be seen as being dimensionless since there are no constant multiplied to the equation and x goes from zero to one.

To solve this equation we are looking for a solution by seperating the variables:

$$u(x,t) = X(x)T(t) \tag{4}$$

If we take the partiall derivatives of this expression we get:

$$u_{xx} = X''(x)T(t), and u_t = X(x)T'(t) \tag{5}$$

So if we set put this in the equation (3) we get:

$$\frac{T'(t)}{T(t)} = \frac{X''(x)}{X(x)} = constant = -\lambda \tag{6}$$

We see that this must be equal to a constant and we see that this is an eigenvalue problem. We put a minus sign infront of the eigenvalue because of convention.

This gives uss the equations:

$$u(0,t) = X(0)T(t) = 0u(1,t) = X(1)T(t) = 0 \tag{7}$$

If we let $T(t) = 0$ we get the trivial solution, which we are not interested int. In two dimensions the same initial conditions require

$$u(0,0,t) = X(0)Y(0)T(t) = 0u(1,0,t) = X(1)Y(0)T(t) = 0u(0,1,t) = X(0)Y(0)T(t) = 0u(1,1,t) = X(1)Y(1 \tag{8}$$

2.2 Algorithms

For solving PDE's we have to different kind of algorithms we have explicit and implicit. An example of an explicit algorithm is Forward Euler. What defines an explicit algorithm is that it takes basis in the forward timestep when differentiating and is therefore straightforward to program. While the implicit scheme we calculate the differencial by using the previous timestep so it becomes a series of matrix equations which can be solved using for example Gaussian elimination or LU-decomposition. Examples of implicit schemes are backward Euler and Crank-Nicolson.

2.2.1 Forward Euler

In forward euler we are approximating the time derivative by:

$$u_t \approx \frac{u(x,t+\Delta t) - u(x,t)}{\Delta t} = \frac{u(x_i,t_j+\Delta t) - u(x_i,t_j)}{\Delta t} \tag{9}$$

This is an explicit scheme because it finds the current time step by looking at the (LES MER PÅ FORSKJELLEN AV IMPLICIT OG EXPLICIT)

We are also using a centered difference in space with the approximation as you can see in equation (1). So setting these to equtions equal to eachother gives:

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} \tag{10}$$

$$\Rightarrow u_{i,j+1} = \alpha u_{i-1,j} + (1-2\alpha)u_{i,j} + \alpha u_{i+1,j} \tag{11}$$

And this is the form we choose for solving this. By looking at this equation we also see that stability requires (eq. (12))

$$\alpha = \frac{\Delta t}{\Delta x^2} < 0.5 \tag{12}$$

Else the second term vanishes, and our solution for the new time step is wrong.

We can implement this as a algorithm just by looping over the timesteps, for so to loop over the x values where $x \in [0,1]$.

2.2.2 Backward Euler

This is an implicit scheme where we approximating the time derivative by:

$$u_t \approx \frac{u(x,t) - u(x,t-\Delta t)}{\Delta t} = \frac{u(x_i,t_j) - u(x_i,t_j-\Delta t)}{\Delta t} \tag{13}$$

And by setting $u_t = u_x x$ we get the equation:

$$u_{i,j-1} = \alpha u_{i-1,j} + (1-2\alpha)u_{i,j} - \alpha u_{j+1,i} \tag{14}$$

We then introduce the matrix:

$$\begin{bmatrix} 1+2\alpha & -\alpha & 0 & 0 & \dots & 0 \\ -\alpha & 1+2\alpha & -\alpha & 0 & \dots & 0 \\ 0 & -\alpha & 1+2\alpha & -\alpha & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & & 1+2\alpha \end{bmatrix}$$

Then we see that we can formulate this as a matrix multiplication problem:

$$\hat{A}V_j = V_{j-i} \tag{15}$$

Which means we can rewrite our differential equation problem to:

$$V_j = \hat{A}^{-1}V_{j_1} = \hat{A}^{-1}(\hat{A}^{-1}V_{j_2}) = ... = \hat{A}^{-j}V_0 \tag{16}$$

To solve this matrix equation we utilize the Gaussian elimination for tridiagonal matrixes which we solved in project 1.

2.2.3 Crank Nicolson

In Cranc-Nicolson we use a time centered scheme where

$$u(x_i, t_{j+1/2}) \approx \tag{17}$$

This gives us the equation :

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = \frac{1}{2}(\frac{u_{i+1,j+1} - 2u_{i,j+1} + 2u_{i-1,j+1}}{(\Delta x)^2} + \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} \tag{18}$$

This we can write as:

$$-\alpha u_{i+1,j+1} + (1+2\alpha)u_{i,j+1} - \alpha u_{i-1,j-1} = \alpha u_{i+1,j} + (1-2\alpha)u_{i,j} + \alpha u_{i-1,j} \tag{19}$$

This we can write as an matrix equation:

$$\hat{A}V_{j+1} = \hat{B}V_i \tag{20}$$

Dette kan vi skrive som :

$$\hat{A}V_{j+1} = b_j \tag{21}$$

Where we find V_{j+1} by using forward euler and then solve the matrix equation as in backward euler by using Gaussian elimination.

2.3 Jacobi

For an explicit solution to the 2-dimensional problem, U_xx and U_yy are given by eq(11). Combining this results in the Jakobi-algorithm (eq.SETT INN)

$$u(i,j,t{+}dt) = u(i,j,t){+}alpha{*}(u(i{+}1,j,t){+}u(i{-}1,j,t){-}4{*}(i,j,t){+}u(i,j{+}1,t){+}u(i,j{-}1,t)); \tag{22}$$

3 Execution

3.0.1 Forward Euler

For the forward Euler algorithm we start by solving U(x,0), hereby referenced as U0, and define α as given by eq(11) with dx=0.1 and dt=dx*dx*0.25, as dictated by the restrictions for the explicit scheme (eq(12). We then call the forward step method (see below) for a given number of timesteps, each run increasing the total time T by dt.

```
vec forward_step(double n, double alpha, vec u, vec unew) {
    for (int i=1; i<n; i++) {
        unew(i) = alpha*u(i-1) + (1-2*alpha) * u(i) + alpha*u(i+1);
    }
    return unew;
}
```

3.0.2 Backward Euler

In the implicit Backward Euler scheme we use Gaussian elimination to advance in space and time, implemented in code below. Here, as eq (14) shows, b-value is defined as 1+2α, and a=c=-α, v being the solution given at a previous timestep, with the same initial condition as for the forward Euler scheme. We run the Gaussian elimination for each timestep dt until T(i) = final T.

```
 Forward Substitution
    double m;
    for (int k=2; k<=n; k++) {
        m = a/b(k-1);
        b(k) = b_value - m*c;
        v(k) -= m*v(k-1);
    }

 Backward Substitution
    u(n)= v(n)/b(n);
    for (int k= n-1; k>0; k--) {
        u(k) = (1.0/b(k))*(v(k) - c*u(k+1));
    }

    u(0) = 0;
    u(n) = 0;
```

3.0.3 Crank-Nicolson

Crank-Nicolson, being a combination of the explicit and implicit schemes, first runs forward step and then uses this updated solution v in the gaussian elimination for each timestep T(i).

3.1 Jacobi

Implementing the jacobi-algorithm is quite straight forward using eq.(22). U(x,y,0)= A old = solution for t=0 found by separation of variables.

```
void jakobi_solver (double dx, double dt) {
    double n = 1.0/dx;
    double t_steps = 1000;
    mat A_old = zeros<mat>(n+1,n+1);
    for (int i=1;i<n;i++) {
        for (int j=1;j<n;j++) {
            A_old(i,j) = func(dx,i,j);
        }
    }

    double alpha = dt/(dx*dx);
    mat A_new = zeros<mat>(n+1,n+1);
    for (int t=1;t<=t_steps;t++) {
        for (int i=1;i<n;i++) {
            for (int j=1;j<n;j++) {
                A_new(i,j) = A_old(i,j) + alpha*(A_old(i+1,j)
                    + A_old(i-1,j) - 4*A_old(i,j) + A_old(i,j+1) + A_old(i,j-1));
            }
        }
        A_old = A_new;
        }
    }
}
```

4 Results

4.1 Closed form solutions

4.1.1 Solution to the 1D heat equation

To solve the equation (2) we need to look for seperable solutions on the form:

$$u(x,t) = X(x)T(t) \tag{23}$$

If we set this in in the equation (2) we get:

$$\frac{\partial}{\partial t}(X(x)T(t)) = \frac{\partial^2}{\partial x^2}(X(x)T(t)) \tag{24}$$

To simplify the notation we write:

$$T'(t)X(x) = T(t)X''(x) \tag{25}$$

Which we can write:

$$\frac{T'(t)}{T(t)} = \frac{X''(x)}{X(x)} \tag{26}$$

We see that each side depends on a different variable R.H.S depends on x and L.H.S depends on t, so therefor this mus be equal to a constant. This is because if we change one and keep the other fixed the value must be the same. This constant we set to $-\lambda$ by convention so the equations to solve becomes:

$$X''(x) + \lambda X(x) = 0 \tag{27}$$

$$T'(t) + \lambda T(t) = 0 \tag{28}$$

With the boundary conditions:

$$u(0,t) = X(0)T(t) = 0 \tag{29}$$

$$u(1,t) = X(1)T(t) = 0 \tag{30}$$

From these boundary conditions we see that it must be $X(0) = X(1) = 0$ because if $T(t) = 0$ we would only get the triviall solutions which we are not interested in.

So we solve the $X(x)$ equation first.

This is a equation which we have solved nmany times before. First we have the case $\lambda < 0$ which gives the solution:

$$X(x) = Ae^{\sqrt{k}x} + Be^{-\sqrt{k}x}, \lambda = -k \tag{31}$$

if we set in the boundary conditions we get that $X(0) = A + B$ and then $X(1) = Ae^{\sqrt{k}} - Ae^{\sqrt{k}} = A(e^{2*\sqrt{k}}$ and since k must be positive this gives that $A = B = 0$ which is the trivial solution which we are not interested in.

When $\lambda = 0$ this gives $A = B = 0$ which also is the trivial solutions.

The last possibility is the harmonic equation which is:

X(x) = Acos($\sqrt{x} + Bsin(\sqrt{\lambda x}(32)$

And with our boundary conditions it gives $X(0) = A = 0$ and $X(1) = Bsin(\sqrt{\lambda}) = 0$ This means that $sin = 0$ This gives us the eigenvalue $\lambda = (n\pi)^2$ for any positive integer. This gives the solution:

$$X(x) = b_n sin(n\pi x) \tag{33}$$

The solution for $T(t)$ is then given by:

$$T'(t) = -n^2 * \pi^2 T(t) \tag{34}$$

Which is well known as

$$T(t) = c_n e^{-(n*pi)^2 t} \tag{35}$$

So the the solution becomes:

$$u(x,t) \approx f(x) * sin(x) e^{-(\pi^2 t)} \tag{36}$$

Where we have used that $f(x) = constant = 1$

4.2 3D- Heat equation

4.2.1 Analytical Solution

Here we have the equation (**??**) which we solve as the 2D equation by seperable solutions:

$$u(x,y,t) = X(x)Y(y)T(t) \tag{37}$$

With the boundary conditions $u(0,y,t) = u(1,y,t) = 0$ and $u(x,0,t) = u(x,1,t) = 0$ So when we set this in the equation we get:

$$\frac{X''(x)}{X(x)} + \frac{Y''(y)}{Y(y)} = \frac{T'(t)}{T(t)} \tag{38}$$

So by the same logic as for 2D this becomes:

$$\frac{X''(x)}{X(x)} + \frac{Y''(y)}{Y(y)} = -\lambda \tag{39}$$

If we first keep y constant and varies x we get the equation:

$$X''(x) + (\lambda + \frac{Y''(y)}{Y(y)})X(x) = 0 \Rightarrow X''(x) + (\lambda +)X(x) = 0 \tag{40}$$

And this we can solve as we did in 2D the same for when we keep x constant:

$$Y''(y) + (\lambda + \mu)Y(y) = 0 \tag{41}$$

These two equations becomes:

$$X(x) = b_n sin(n\pi x) \tag{42}$$

$$Y(y) = c_m sin(m\pi y) \tag{43}$$

And the time equation then becomes:

$$T(t) = d_{n,m} e^{-(m^2\pi^2 + n^2\pi^2)t}(44)$$

So the equation becomes with $m = n = 1$ and $b_n c_n d_{n,m} = 1$

$$u(x,y,t) = sin(\pi x) sin(\pi y) e^{-2pi^2 t} \tag{45}$$

4.2.2 Error Analysis

To calculate the error we use taylor expansion which are defined:

$$u_n = \frac{f^{(n)}(b)}{n!} \tag{46}$$

So to calculate the error in the forward difference for u'(t) we make a Taylor expansion around t_n:

$$u(t_{n+1}) = u(t_n) + u'(t_n)\Delta t + \frac{1}{2}u''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^3) \tag{47}$$

This gives the error:

$$R = \frac{1}{2}u''(t_n)\Delta t + \mathcal{O}\Delta t^2 \tag{48}$$

This means that the forward euler has an error in time in the first order. For backwards euler we taylor expand $u(t_{n-1}$ and get:

$$R = \frac{1}{2}u''(t_n)\Delta t + \mathcal{O}\Delta t^2 \tag{49}$$

So the same as in the Forward Euler scheme

In Crank-Nicolson we use a time centered scheme so we have to taylor expand $u_{n+1/2}$ and $u_{n-1/2}$ and combine these.

$$u(t_{n+1/2}) = u(t_n)+\frac{1}{2}u'(t_n)\Delta t+\frac{1}{4}u''(t_n)\Delta t^2+\frac{1}{12}u'''(t_n)\Delta t^3+\frac{1}{48}u^{(4)}(t_n)\Delta t^4+\frac{1}{240}u^{(5)}(t_n)\Delta t^5+\mathcal{O}(\Delta t^6) \tag{50}$$

and

$$u(t_{n-1/2}) = u(t_n)-\frac{1}{2}u'(t_n)\Delta t+\frac{1}{4}u''(t_n)\Delta t^2-\frac{1}{12}u'''(t_n)\Delta t^3+\frac{1}{48}u^{(4)}(t_n)\Delta t^4-\frac{1}{240}u^{(5)}(t_n)\Delta t^5+\mathcal{O}(\Delta t^6) \tag{51}$$

If we subtract the last from the first we get the error:

$$R = \frac{1}{24}u'''(t_{n+1/2})\Delta t^2 + \mathcal{O}(\Delta t^4) \tag{52}$$

But to get the full error we have to take in consideration the standard arithmetic mean wich is:

$$\frac{1}{2}(u(t_{n+1/2}) + u(t_{n-1/2}) \tag{53}$$

This have the error term:

$$R = \frac{1}{8}u''(t_{n+1/2})\Delta + \frac{1}{384}u''''(t_n)\Delta t^4 + \mathcal{O}(\Delta t^6) \tag{54}$$

If we add these to we get the total error in the time as:

$$R = (\frac{1}{24}u'''(t_{n+1/2}) + \frac{1}{8}u''(t_n))\Delta t^2 + \mathcal{O}\Delta t^4 \tag{55}$$

So the error in the Crank-Nicolson scheme is Δt^2

Figure 1: The time evolution for the analytical solution

Figure 2: The three schemes with the analytical solutiion for when $T = 0.025$ and $dt = 0.0025$

And the error in spatial differencial for all three is equal to by usin taylor expansion:

$$R_x = \frac{1}{12}u''''(x_n)\Delta x^2 + \mathcal{O}(\Delta x^4) \tag{56}$$

So the error on the x is of the second order.

In figure (1) we see the time evolution for the analytical solution which we used to get the time points to analyze the numerical calculations. In figure (2) and (3) we see the numerical calculations against the analytical for two times $t_1 = 0.025$ and $t_2 = 0.65$. And in table (1) we see the realtive error at the different time points. Where the relative error is calculated with the max value and $\epsilon = |1 - u_{num}/u_{analytical}|$.

4.3 2D PDE

5 Discussion

6 Conclusion

Figure 3: The three schemes with the analytical solutiion for when $T = 0.025$ and $dt = 0.65$

Table 1: Table with the relative error of the different schemes calculating it by using the max value and $\epsilon = |1 - u_{num}/u_{analytical}|$

	$t_1 = 0.025$	$t_2 = 0.065$
Forward Euler	0.000023	0.026
Backward Euler	0.000631	0.4227
Crank Nicolson	0.01261	0.1341

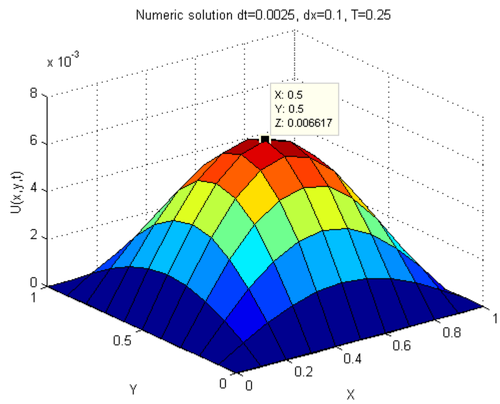

Figure 4: Explicit solution for dx=0.1, $\alpha = 0.25$, T=0.25

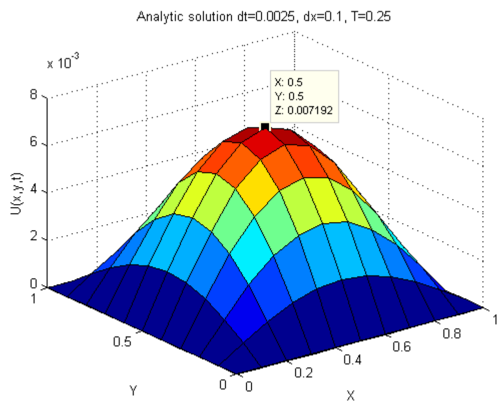

Figure 5: Analytical solution for dx=0.1, $\alpha = 0.25$, T=0.25

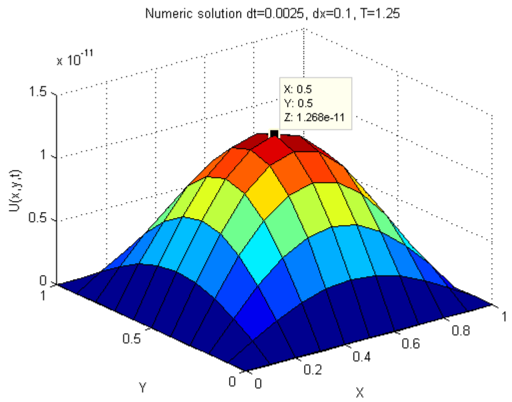

Figure 6: Explicit solution for dx=0.1, α = 0.25, T=1.25

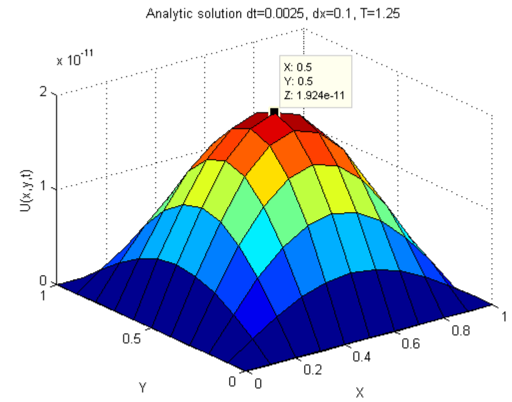

Figure 7: Analytical solution for dx=0.1, α = 0.25, T=1.25

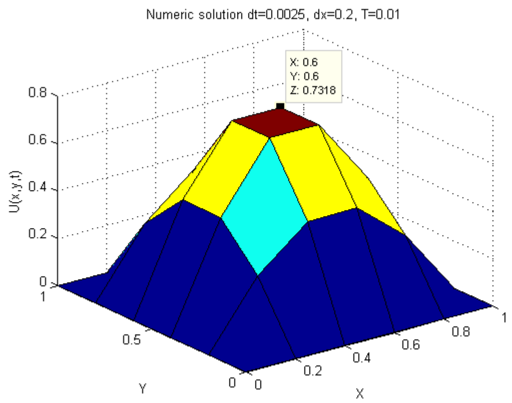

Figure 8: Explicit solution for dx=0.2, α = 0.25, T=1.0

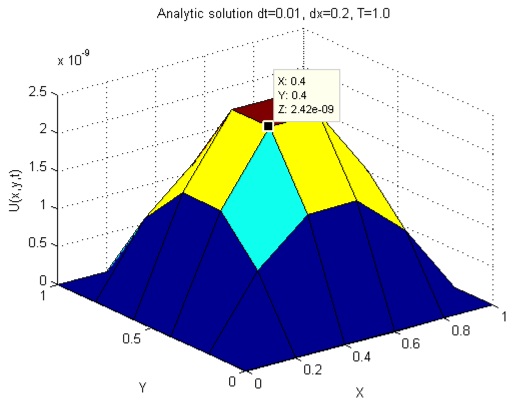

Figure 9: Analytical solution for dx=0.2, α = 0.25, T=1.0

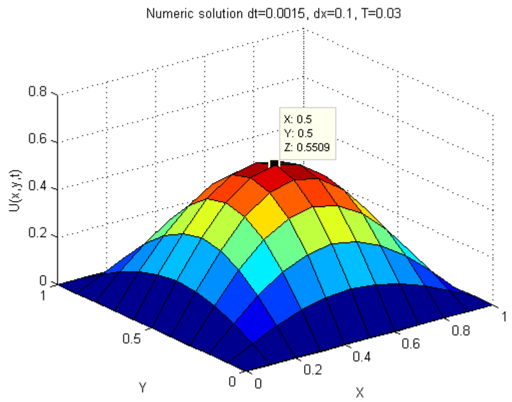

Figure 10: Explicit solution for dx=0.1, α = 0.15, T=0.03

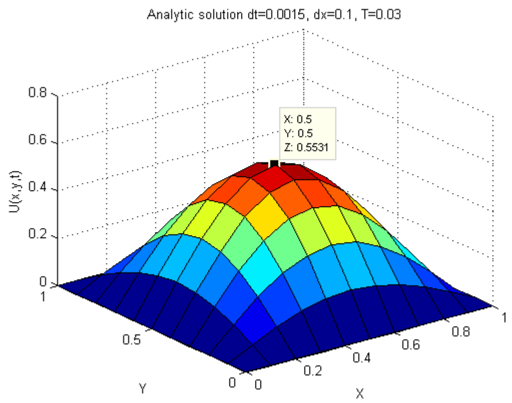

Figure 11: Analytical solution for dx=0.1, α = 0.15, T=0.03

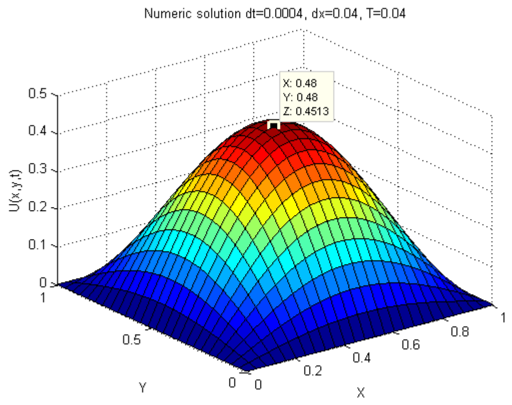

Figure 12: Explicit solution for dx=0.04, α = 0.25, T=0.04

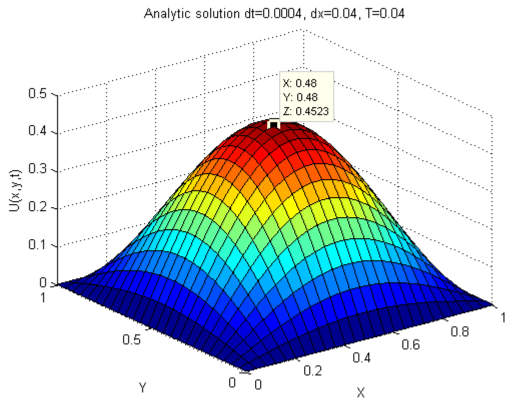

Figure 13: Analytic solution for dx=0.04, α = 0.25, T=0.04

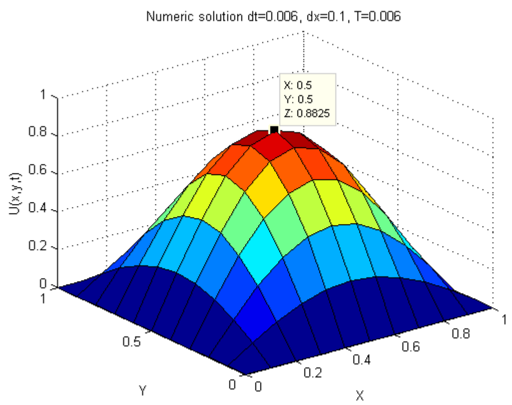

Figure 14: Explicit solution for dx=0.1, α = 0.6, T=0.006

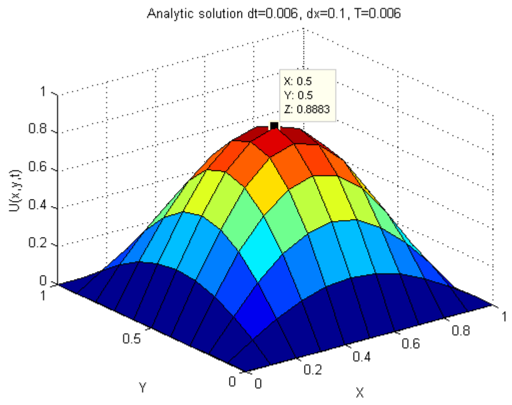

Figure 15: Analytical solution for dx=0.1, α = 0.6, T=0.006

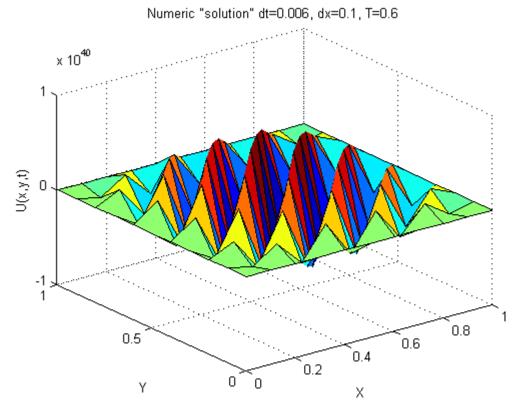

Figure 16: Explicit solution for dx=0.1, α = 0.6, T=0.6

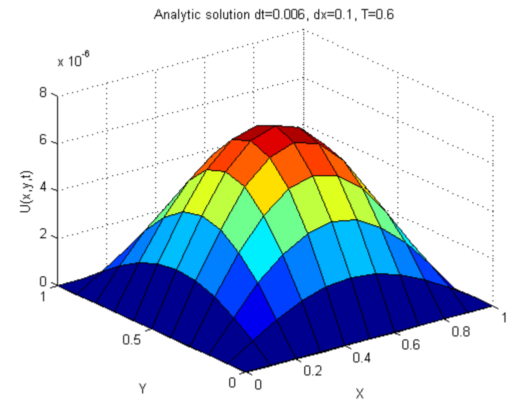

Figure 17: Analytical solution for dx=0.1, α = 0.25, T=0.6