On Approximating Solution of One-Dimensional Heat Equation by Using Finite Difference Method

Nguyen Quan Ba Hong
Students at Faculty of Math and Computer Science,
Ho Chi Minh University of Science, Vietnam
email. nguyenquanbahong@gmail.com
blog. http://hongnguyenquanba.wordpress.com *

December 8, 2016

Abstract

This assignment aims at solving one-dimensional heat equation by using finite difference methods.

We also implement one-dimensional heat equation by Matlab scripts at the end of this context.

Contents

1 Theoretical Problems **5**

2 Numerical Scheme **5**
2.1 Mesh 5
2.2 The θ Method 7

3 Local Truncation Errors and Order of Accuracy **10**

4 Method of Lines Discretizations **16**

5 Stability Theory **18**

6 Forward Euler Method **19**
6.1 Local Truncation Errors and Order of Accuracy 19

7 Backward Euler Method **19**

8 Crank-Nicolson Method **19**

9 Practical Problems **20**

List of Figures

1 Stencil for the natural methods. 6
2 Method of lines interpretation. $U_i(t)$ is the solution along the line forward in time at the grid point x_i. 17

Diffusion Equations and Parabolic Problems

We now begin to study finite difference methods for time-dependent partial differential equations (PDEs), where variations in space are related to variations in time. We begin with the heat equationn (or diffusion equation)

$$u_t = (\kappa u_x)_x + f \tag{0.1}$$

This is the classical example of a *parabolic* equation, and many of the general properties seen here carry over to the design of numerical methods for other parabolic equations. We will assume $\kappa > 0$. If $\kappa < 0$, then (0.1) would be a "backward heat equation", which is an ill-posed problem.

Along with this equation we need initial conditions at some time t_0, which we typically take to be $t_0 = 0$,

$$u(x, 0) = u_0(x) \tag{0.2}$$

and also boundary conditions if we are working on a bounded domain, e.g., the Dirichlet conditions

$$u(0, t) = g_0(t), t > 0 \tag{0.3}$$
$$u(1, t) = g_1(t), t > 0 \tag{0.4}$$

if $0 \leq x \leq 1$.

Remark 0.1. We now consider the one-dimensional heat equation with homogeneous Dirichlet boundary conditions

$$u(0, t) = 0, t > 0 \tag{0.5}$$
$$u(1, t) = 0, t > 0 \tag{0.6}$$

because we just need some small modifications in both theoretical establishment and practical implementations for inhomogenous one.

1 Theoretical Problems

Problem 1.1 (Heat Equation in 1D). *For $f \in L^2([0,1] \times [0,T])$ and $\kappa(x) \in C^1([0,1])$ and there exist consider strictly positive constants c_1, c_2 such that*

$$c_1 \leq \kappa(x) \leq c_2, \forall x \in [0,1] \tag{1.1}$$

We consider heat equation

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(\kappa(x)\frac{\partial u}{\partial x}\right) + f(x,t), \forall (x,t) \in (0,1) \times (0,T] \tag{1.2}$$

Along with this equation we need initial conditions at time 0.

$$u(x,0) = u_0(x) \tag{1.3}$$

and also boundary conditions if we are working on a bounded domain, e.g., the Dirichlet conditions

$$u(0,t) = 0, t > 0 \tag{1.4}$$
$$u(1,t) = 0, t > 0 \tag{1.5}$$

1. *Construct finite difference scheme for previous equation (Forward Euler, Backward Euler and Crank-Nicolson).*
2. *Calculate and estimate the local truncation error for 3 methods.*
3. *With $f = 0$ and κ be constant, finding the stable condition for 3 methods.*

2 Numerical Scheme

2.1 Mesh

In practice we generally apply a set of finite difference equations on a discrete grid with grid points (x_i, t_n), where $N_x + 2$ space points x_i are given by

$$x_i = ih, \quad j = 0, 1, \ldots, N_x + 1 \tag{2.1}$$

where $h = \dfrac{1}{N_x + 1}$ is the space step, and with $N_t + 2$ time points

$$t_n = nk, \quad n = 0, 1, \ldots, N_t + 1 \tag{2.2}$$

where $k = \dfrac{1}{N_t + 1}$ is the time step.

We make an assumption that

$$k = O(h) \tag{2.3}$$

This assumption is used in Taylor series expansion to estimate the local truncation error later.

Return to our problem, let

$$U_i^n \approx u(x_i, t_n) \tag{2.4}$$

represent the numerical approximation at grid point (x_i, t_n) for $i = 0, 1, \ldots, N_x + 1$ and $n = 0, 1, \ldots, N_t + 1$.

Since the heat equation is an evolution equation that can be solved forward in time, we set up our difference equations in a form where we can march forward in time, determining the values U_i^{n+1} for all i from the values U_i^n at the previous time level, or perhaps using also values at earlier time levels with a multistep formula.

We rewritte (1.2) more explicitly as

$$\frac{\partial u}{\partial t} = \kappa'(x) \frac{\partial u}{\partial x} + \kappa(x) \frac{\partial^2 u}{\partial x^2} + f(x,t), \forall (x,t) \in (0,1) \times (0,T] \tag{2.5}$$

From (2.5), we deduce

$$\frac{\partial u}{\partial t}(x_i, t_n) = \kappa'(x_i) \frac{\partial u}{\partial x}(x_i, t_n) + \kappa(x_i) \frac{\partial^2 u}{\partial x^2}(x_i, t_n) + f(x_i, t_n) \tag{2.6}$$

for $j = 0, 1, \ldots, N_x + 1, n = 0, 1, \ldots, N_t + 1$.

As an example, one natural discretization of (1.2) would be

$$\frac{U_i^{n+1} - U_i^n}{k} = \kappa'(x_i) \frac{U_{j+1}^n - U_{j-1}^n}{2h} + \kappa(x_i) \frac{U_{j-1}^n - 2U_i^n + U_{j+1}^n}{h^2} + f(x_i, t_n) \tag{2.7}$$

for $j = 0, 1, \ldots, N_x + 1, n = 0, 1, \ldots, N_t + 1$.

This uses the standard centered difference in space and a forward difference in time. This is an *explicit* method since we can compute each U_i^{n+1} explicitly in terms of the previous data

$$U_i^{n+1} = k \left(\frac{\kappa'(x_i)}{2h} + \frac{\kappa(x_i)}{h^2} \right) U_{j+1}^n + \left(1 - \frac{2k\kappa(x_i)}{h^2} \right) U_i^n \tag{2.8}$$

$$+ k \left(\frac{\kappa(x_i)}{h^2} - \frac{\kappa'(x_i)}{2h} \right) U_{j-1}^n + k f(x_i, t_n) \tag{2.9}$$

Figure 1 shows the *stencil* of this method.

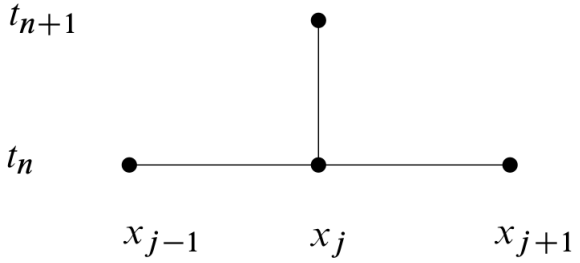

Figure 1: Stencil for the natural methods.

This is a one-step method in time, which is also called a two-level method in the context of PDEs since it involves the solution at two different time levels.

Another one-step method, which is much more useful in practice, as we will see below, is the *Crank-Nicolson* method,

$$\frac{U_j^{n+1}-U_j^n}{k}=\kappa'(x_j)\left(\frac{U_{j+1}^n-U_{j-1}^n}{4h}+\frac{U_{j+1}^{n+1}-U_{j-1}^{n+1}}{4h}\right)+f(x_j,t_n) \tag{2.10}$$

$$+\kappa(x_j)\left(\frac{U_{j-1}^n-2U_j^n+U_{j+1}^n}{2h^2}+\frac{U_{j-1}^{n+1}-2U_j^{n+1}+U_{j+1}^{n+1}}{2h^2}\right) \tag{2.11}$$

This is an *implicit* method and gives a tridiagonal system of equations to solve for all the values U_i^{n+1} simultaneously.

Instead of continuing to discuss the Crank-Nicolson method, we now consider the more general θ method, which contains Crank-Nicolson method as a special case.

2.2 The θ Method

The finite difference discretization of the θ method reads

$$\frac{U_i^{n+1}-U_i^n}{k}=\kappa'(x_i)\left((1-\theta_1)D_1U_i^n+\theta_1D_1U_i^{n+1}\right) \tag{2.12}$$

$$+\kappa(x_i)\left((1-\theta_2)D_2U_i^n+\theta_2D_2U_i^{n+1}\right)+f(x_i,t_n) \tag{2.13}$$

where

$$D_1U_i^n=\frac{U_{j+1}^n-U_{j-1}^n}{2h} \tag{2.14}$$

$$D_1U_i^{n+1}=\frac{U_{j+1}^{n+1}-U_{j-1}^{n+1}}{2h} \tag{2.15}$$

$$D_2U_i^n=\frac{U_{j-1}^n-2U_i^n+U_{j+1}^n}{h^2} \tag{2.16}$$

$$D_2U_i^{n+1}=\frac{U_{j-1}^{n+1}-2U_i^{n+1}+U_{j+1}^{n+1}}{h^2} \tag{2.17}$$

Inserting (2.14)-(2.28) into (2.12)-(2.13), we can rewrite (2.12)-(2.13) as

$$\frac{U_i^{n+1}-U_i^n}{k} \tag{2.18}$$

$$=\kappa'(x_i)\left((1-\theta_1)\frac{U_{j+1}^n-U_{j-1}^n}{2h}+\theta_1\frac{U_{j+1}^{n+1}-U_{j-1}^{n+1}}{2h}\right)+f(x_i,t_n) \tag{2.19}$$

$$+\kappa(x_i)\left((1-\theta_2)\frac{U_{j-1}^n-2U_i^n+U_{j+1}^n}{h^2}+\theta_2\frac{U_{j-1}^{n+1}-2U_i^{n+1}+U_{j+1}^{n+1}}{h^2}\right) \tag{2.20}$$

This method includes some common methods.

1. Forward Euler method. Equation (2.18)-(2.20) becomes

$$\frac{U_i^{n+1}-U_i^n}{k}=\kappa'(x_i)\frac{U_{j+1}^n-U_{j-1}^n}{2h} \tag{2.21}$$

$$+\kappa(x_i)\frac{U_{j-1}^n-2U_i^n+U_{j+1}^n}{h^2}+f(x_i,t_n) \tag{2.22}$$

2. Backward Euler method. Equation (2.18)-(2.20) becomes

$$\frac{U_i^{n+1}-U_i^n}{k}=\kappa'(x_i)\frac{U_{j+1}^{n+1}-U_{j-1}^{n+1}}{2h} \tag{2.23}$$

$$+\kappa(x_i)\frac{U_{j-1}^{n+1}-2U_i^{n+1}+U_{j+1}^{n+1}}{h^2}+f(x_i,t_n) \tag{2.24}$$

3. Crank-Nicolson method. Equation (2.18)-(2.20) becomes

$$\frac{U_i^{n+1}-U_i^n}{k} \tag{2.25}$$

$$=\kappa'(x_i)\left(\frac{U_{j+1}^n-U_{j-1}^n}{4h}+\frac{U_{j+1}^{n+1}-U_{j-1}^{n+1}}{4h}\right)+f(x_i,t_n) \tag{2.26}$$

$$+\kappa(x_i)\left(\frac{U_{j-1}^n-2U_i^n+U_{j+1}^n}{2h^2}+\frac{U_{j-1}^{n+1}-2U_i^{n+1}+U_{j+1}^{n+1}}{2h^2}\right) \tag{2.27}$$

Next, equation (2.18)-(2.20) is equivalent to

$$\left(\frac{\theta_1 k\kappa'(x_i)}{2h}-\frac{\theta_2 k\kappa(x_i)}{h^2}\right)U_{j-1}^{n+1}+\left(1+\frac{2\theta_2 k\kappa(x_i)}{h^2}\right)U_i^{n+1} \tag{2.28}$$

$$-\left(\frac{\theta_1 k\kappa'(x_i)}{2h}+\frac{\theta_2 k\kappa(x_i)}{h^2}\right)U_{j+1}^{n+1} \tag{2.29}$$

$$=\left(\frac{(1-\theta_2)k\kappa(x_i)}{h^2}-\frac{(1-\theta_1)k\kappa'(x_i)}{2h}\right)U_{j-1}^n \tag{2.30}$$

$$+\left(1-\frac{2(1-\theta_2)k\kappa(x_i)}{h^2}\right)U_i^n \tag{2.31}$$

$$+\left(\frac{(1-\theta_1)k\kappa'(x_i)}{2h}+\frac{(1-\theta_2)k\kappa(x_i)}{h^2}\right)U_{j+1}^n+kf(x_i,t_n) \tag{2.32}$$

For simplicity, we now put

$$r_i=\frac{k\kappa'(x_i)}{2h} \tag{2.33}$$

$$s_i=\frac{k\kappa(x_i)}{h^2} \tag{2.34}$$

then (2.28)-(2.32) becomes

$$(r_i\theta_1-s_i\theta_2)U_{j-1}^{n+1}+(1+2s_i\theta_2)U_i^{n+1}-(r_i\theta_1+s_i\theta_2)U_{j+1}^{n+1} \tag{2.35}$$

$$=(s_i(1-\theta_2)-r_i(1-\theta_1))U_{j-1}^n+(1-2s_i(1-\theta_2))U_i^n \tag{2.36}$$

$$+(r_i(1-\theta_1)+s_i(1-\theta_2))U_{j+1}^n+kf(x_i,t_n) \tag{2.37}$$

Equation (2.35)-(2.37) is the scheme of the θ method.

This method includes some common methods.

1. Case $\theta_1=\theta_2=0$ (Forward Euler). (2.35)-(2.37) becomes

$$U_i^{n+1}=(s_i-r_i)U_{j-1}^n+(1-2s_i)U_i^n+(r_i+s_i)U_{j+1}^n+kf(x_i,t_n) \tag{2.38}$$

2. Case $\theta_1 = \theta_2 = 1$ (Backward Euler). (2.35)-(2.37) becomes

$$U_i^n = (r_i - s_i) U_{j-1}^{n+1} + (1 + 2s_i) U_i^{n+1} - (r_i + s_i) U_{j+1}^{n+1} - kf(x_i, t_n) \tag{2.39}$$

3. Case $\theta_1 = \theta_2 = \frac{1}{2}$ (Crank-Nicolson). (2.35)-(2.37) becomes

$$\frac{1}{2}(r_i - s_i) U_{j-1}^{n+1} + (1 + s_i) U_i^{n+1} - \frac{1}{2}(r_i + s_i) U_{j+1}^{n+1} \tag{2.40}$$

$$= \frac{1}{2}(s_i - r_i) U_{j-1}^{n} + (1 - s_i) U_i^{n} + \frac{1}{2}(r_i + s_i) U_{j+1}^{n} + kf(x_i, t_n) \tag{2.41}$$

We can rewrite the scheme of the θ method more compactly by putting

$$R = \begin{bmatrix} 0 & r_1 & & & \\ -r_2 & 0 & r_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -r_{N_x-1} & 0 & r_{N_x-1} \\ & & & -r_{N_x} & 0 \end{bmatrix} \tag{2.42}$$

$$S = \begin{bmatrix} -2s_1 & s_1 & & & \\ s_2 & -2s_2 & s_2 & & \\ & \ddots & \ddots & \ddots & \\ & & s_{N_x-1} & -2s_{N_x-1} & s_{N_x-1} \\ & & & s_{N_x} & -2s_{N_x} \end{bmatrix} \tag{2.43}$$

Then (2.35)-(2.37) can be rewritten as following

$$(I - \theta_1 R - \theta_2 S) U^{n+1} = (I + (1 - \theta_1) R + (1 - \theta_2) S) U^n + kF \tag{2.44}$$

Under this notation, (2.38)-(2.41) becomes

1. Forward Euler method.

$$U^{n+1} = (I + R + S) U^n + kF \tag{2.45}$$

2. Backward Euler method.

$$U^{n} = (I - R - S) U^{n+1} + kF \tag{2.46}$$

3. Crank-Nicolson method.

$$\left(I - \frac{1}{2}R - \frac{1}{2}S\right) U^{n+1} = \left(I + \frac{1}{2}R + \frac{1}{2}S\right) U^n + kF \tag{2.47}$$

These linear finite difference equations can be solved formally as

$$U^{n+1} = AU^n + kF \tag{2.48}$$

where

$$U^n = \begin{bmatrix} U_1^n \\ U_2^n \\ \vdots \\ U_{N_x-1}^n \\ U_{N_x}^n \end{bmatrix} \tag{2.49}$$

for $n = 0, 1, \ldots, N_t + 1$ and

1. Forward Euler method.

$$A = I + R + S \tag{2.50}$$

2. Backward Euler method.

$$A = (I - R - S)^{-1} \tag{2.51}$$

3. Crank-Nicolson method.

$$A = \left(I + \frac{1}{2}R + \frac{1}{2}S\right)\left(I - \frac{1}{2}R - \frac{1}{2}S\right)^{-1} \tag{2.52}$$

Remark. In the inhomogeneous boundary conditions case, not how the boundary conditions (0.3)-(0.4) come into these equations.

Since a tridiagonal system of N_x equations can be solved with $O(N_x)$ work, this method is essentially as efficient per time step as an explicit method. Moreover, the heat equation is "stiff", and hence this implicit method, which allows much larger time steps to be taken than an explicit method, is a very efficient method for the heat equation.

Solving a parabolic equation with an implicit method requires solving a system of equations with the same structure as the two-point boundary value problem. Similarly, a multidimensional parabolic equation requires solving a problem with the structure of a multidimensional elliptic equation in each time step.

3 Local Truncation Errors and Order of Accuracy

We can define the local truncation error as usual, we insert the exact solution $u(x,t)$ of the PDE into the finite difference equation and determine by how much it fails to satisfy the discrete equation.

The local truncation error of the θ method is based on the form

$$\tau_i^n = \frac{u(x_i, t_{n+1}) - u(x_i, t_n)}{k} - f(x_i, t_n) \tag{3.1}$$

$$- \kappa'(x_i) \begin{pmatrix} (1-\theta_1)\dfrac{u(x_{j+1}, t_n) - u(x_{j-1}, t_n)}{2h} \\ +\theta_1 \dfrac{u(x_{j+1}, t_{n+1}) - u(x_{j-1}, t_{n+1})}{2h} \end{pmatrix} \tag{3.2}$$

$$-\kappa\left(x_i\right)\left(\begin{array}{l}\left(1-\theta_2\right)\dfrac{u\left(x_{j-1},t_n\right)-2u\left(x_i,t_n\right)+u\left(x_{j+1},t_n\right)}{h^2}\\ +\theta_2\dfrac{u\left(x_{j-1},t_{n+1}\right)-2u\left(x_i,t_{n+1}\right)+u\left(x_{j+1},t_{n+1}\right)}{h^2}\end{array}\right) \tag{3.3}$$

or in general notation

$$\tau\left(x,t\right)=\frac{u\left(x,t+k\right)-u\left(x,t\right)}{k}-f \tag{3.4}$$

$$-\kappa'\left(\begin{array}{l}\left(1-\theta_1\right)\dfrac{u\left(x+h,t\right)-u\left(x-h,t\right)}{2h}\\ +\theta_1\dfrac{u\left(x+h,t+k\right)-u\left(x-h,t+k\right)}{2h}\end{array}\right) \tag{3.5}$$

$$-\kappa\left(\begin{array}{l}\left(1-\theta_2\right)\dfrac{u\left(x-h,t\right)-2u\left(x,t\right)+u\left(x+h,t\right)}{h^2}\\ +\theta_2\dfrac{u\left(x-h,t+k\right)-2u\left(x,t+k\right)+u\left(x+h,t+k\right)}{h^2}\end{array}\right) \tag{3.6}$$

Again we should be careful to use the form that directly models the differential equation in order to get powers of k and h that agree with what we hope to see in the global error. Although we do not know $u\left(x,t\right)$ in general, if we assume that it is smooth and use Taylor series expansions about $u\left(x,t\right)$, we can estimate the local truncation errors.

We will make the convention that all functions $u_{x^at^b}$ below are considered at the point (x,t). Now focusing on (3.4)-(3.6), we need

1. Taylor expansion $O\left(k^3\right)$ for the term $u\left(x,t+k\right)$ in (3.4).

$$u\left(x,t+k\right)=u+ku_t+\frac{k^2}{2}u_{tt}+O\left(k^3\right) \tag{3.7}$$

2. Taylor expansion $O\left(h^3\right)$ for the term $u\left(x+h,t\right)$ in (3.5).

$$u\left(x+h,t\right)=u+hu_x+\frac{h^2}{2}u_{xx}+O\left(h^3\right) \tag{3.8}$$

3. Taylor expansion $O\left(h^3\right)$ for the term $u\left(x-h,t\right)$ in (3.5).

$$u\left(x-h,t\right)=u-hu_x+\frac{h^2}{2}u_{xx}+O\left(h^3\right) \tag{3.9}$$

4. Taylor expansion $O\left(h^3+k^3\right)$ for the term $u\left(x+h,t+k\right)$ in (3.5).

$$u\left(x+h,t+k\right)=u+hu_x+ku_t+\frac{h^2}{2}u_{xx}+hku_{xt}+\frac{k^2}{2}u_{tt}+O\left(h^3+k^3\right) \tag{3.10}$$

5. Taylor expansion $O\left(h^3+k^3\right)$ for the term $u\left(x-h,t+k\right)$ in (3.5).

$$u\left(x-h,t+k\right)=u-hu_x+ku_t+\frac{h^2}{2}u_{xx}-hku_{xt}+\frac{k^2}{2}u_{tt}+O\left(h^3+k^3\right) \tag{3.11}$$

6. Taylor expansion $O\left(h^4\right)$ for the term $u\left(x-h,t\right)$ in (3.6).

$$u\left(x-h,t\right)=u-hu_x+\frac{h^2}{2}u_{xx}-\frac{h^3}{6}u_{xxx}+O\left(h^4\right) \tag{3.12}$$

7. Taylor expansion $O\left(h^4\right)$ for the term $u\left(x+h,t\right)$ in (3.6).

$$u\left(x+h,t\right)=u+hu_x+\frac{h^2}{2}u_{xx}+\frac{h^3}{6}u_{xxx}+O\left(h^4\right) \tag{3.13}$$

8. Taylor expansion $O\left(h^4+k^4\right)$ for the term $u\left(x-h,t+k\right)$ in (3.6).

$$u\left(x-h,t+k\right) \tag{3.14}$$

$$=u-hu_x+ku_t+\frac{h^2}{2}u_{xx}-hku_{xt}+\frac{k^2}{2}u_{tt} \tag{3.15}$$

$$-\frac{h^3}{6}u_{xxx}+\frac{h^2k}{2}u_{xxt}-\frac{hk^2}{2}u_{xtt}+\frac{k^3}{6}u_{ttt}+O\left(h^4+k^4\right) \tag{3.16}$$

9. Taylor expansion $O\left(k^4\right)$ for the term $u\left(x,t+k\right)$ in (3.6).

$$u\left(x,t+k\right)=u+ku_t+\frac{k^2}{2}u_{tt}+\frac{k^3}{6}u_{ttt}+O\left(k^4\right) \tag{3.17}$$

10. Taylor expansion $O\left(H^6\right)$ for the term $u\left(x_{j+1},t_{n+1}\right)$ in (3.6).

$$u\left(x+h,t+k\right) \tag{3.18}$$

$$=u+hu_x+ku_t+\frac{h^2}{2}u_{xx}+hku_{xt}+\frac{k^2}{2}u_{tt} \tag{3.19}$$

$$+\frac{h^3}{6}u_{xxx}+\frac{h^2k}{2}u_{xxt}+\frac{hk^2}{2}u_{xtt}+\frac{k^3}{6}u_{ttt}+O\left(h^4+k^4\right) \tag{3.20}$$

We now have enough Taylor series expansions to estimate the truncation error of the θ method.

Inserting (3.7) into (3.4) yields

$$\frac{u\left(x,t+k\right)-u\left(x,t\right)}{k}=\frac{u+ku_t+\frac{k^2}{2}u_{tt}+O\left(k^3\right)-u}{k} \tag{3.21}$$

$$=u_t+\frac{k}{2}u_{tt}+O\left(k^2\right) \tag{3.22}$$

Inserting (3.8)-(3.9) into $D_1u\left(x,t\right)$ in (3.5) yields

$$\frac{u\left(x+h,t\right)-u\left(x-h,t\right)}{2h} \tag{3.23}$$

$$=\frac{u+hu_x+\frac{h^2}{2}u_{xx}+O\left(h^3\right)-\left(u-hu_x+\frac{h^2}{2}u_{xx}+O\left(h^3\right)\right)}{2h} \tag{3.24}$$

$$=u_x+O\left(h^2\right) \tag{3.25}$$

Inserting (3.10)-(3.11) into $D_1u\left(x,t+k\right)$ in (3.5) yields

$$\frac{u\left(x+h,t+k\right)-u\left(x-h,t+k\right)}{2h} \tag{3.26}$$

$$= \frac{1}{2h}\left(\begin{array}{l} u + hu_x + ku_t + \frac{h^2}{2}u_{xx} + hku_{xt} + \frac{k^2}{2}u_{tt} + O\left(h^3 + k^3\right) \\ -\left(u - hu_x + ku_t + \frac{h^2}{2}u_{xx} - hku_{xt} + \frac{k^2}{2}u_{tt} + O\left(h^3 + k^3\right)\right) \end{array}\right) \tag{3.27}$$

$$= u_x + ku_{xt} + \frac{1}{2h}O\left(h^3 + k^3\right) \tag{3.28}$$

$$= u_x + ku_{xt} + O\left(h^2\right) \tag{3.29}$$

where the last row is deduced from (2.3).

Inserting (3.12)-(3.13) into $D_2u\left(x,t\right)$ in (3.6) yields

$$\frac{u\left(x-h,t\right) - 2u\left(x,t\right) + u\left(x+h,t\right)}{h^2} \tag{3.30}$$

$$= \frac{1}{h^2}\left(\begin{array}{l} u - hu_x + \frac{h^2}{2}u_{xx} - \frac{h^3}{6}u_{xxx} + O\left(h^4\right) - 2u \\ +u + hu_x + \frac{h^2}{2}u_{xx} + \frac{h^3}{6}u_{xxx} + O\left(h^4\right) \end{array}\right) \tag{3.31}$$

$$= u_{xx} + O\left(h^2\right) \tag{3.32}$$

Inserting (3.14)-(3.20) into $D_2u\left(x,t+k\right)$ yields

$$\frac{u\left(x-h,t+k\right) - 2u\left(x,t+k\right) + u\left(x+h,t+k\right)}{h^2} \tag{3.33}$$

$$= \frac{1}{h^2}\left(\begin{array}{l} u - hu_x + ku_t + \frac{h^2}{2}u_{xx} - hku_{xt} + \frac{k^2}{2}u_{tt} - \frac{h^3}{6}u_{xxx} \\ +\frac{h^2k}{2}u_{xxt} - \frac{hk^2}{2}u_{xtt} + \frac{k^3}{6}u_{ttt} + O\left(h^4 + k^4\right) \\ -2\left(u + ku_t + \frac{k^2}{2}u_{tt} + \frac{k^3}{6}u_{ttt} + O\left(k^4\right)\right) \\ +u + hu_x + ku_t + \frac{h^2}{2}u_{xx} + hku_{xt} + \frac{k^2}{2}u_{tt} + \frac{h^3}{6}u_{xxx} \\ +\frac{h^2k}{2}u_{xxt} + \frac{hk^2}{2}u_{xtt} + \frac{k^3}{6}u_{ttt} + O\left(h^4 + k^4\right) \end{array}\right) \tag{3.34}$$

$$= u_{xx} + ku_{xxt} + \frac{1}{h^2}O\left(h^4 + k^4\right) \tag{3.35}$$

$$= u_{xx} + ku_{xxt} + O\left(h^2\right) \tag{3.36}$$

Inserting (3.21)-(3.36) into (3.4)-(3.6) yields

$$\tau\left(x,t\right) = u_t + \frac{k}{2}u_{tt} + O\left(k^2\right) - f \tag{3.37}$$

$$-\kappa'\left(\left(1-\theta_1\right)\left(u_x + O\left(h^2\right)\right) + \theta_1\left(u_x + ku_{xt} + O\left(h^2\right)\right)\right) \tag{3.38}$$

$$-\kappa\left(\left(1-\theta_2\right)\left(u_{xx} + O\left(h^2\right)\right) + \theta_2\left(u_{xx} + ku_{xxt} + O\left(h^2\right)\right)\right) \tag{3.39}$$

$$= u_t + \frac{k}{2}u_{tt} - f - \kappa'u_x - \kappa'\theta_1 ku_{xt} - \kappa u_{xx} - \kappa\theta_2 ku_{xxt} \tag{3.40}$$

$$+ O\left(h^2 + k^2\right) \tag{3.41}$$

We rewrite (1.2) briefly as

$$u_t = \left(\kappa u_x\right)_x + f \tag{3.42}$$

$$= \kappa'u_x + \kappa u_{xx} + f \tag{3.43}$$

Then we need to compute some essential partial derivatives in (3.37)-(3.41).

$$u_{xt} = \left(\kappa'u_x + \kappa u_{xx} + f\right)_x \tag{3.44}$$

$$= \kappa'' u_x + 2\kappa' u_{xx} + \kappa u_{xxx} + f_x \tag{3.45}$$

$$u_{xxt} = \left(\kappa'' u_x + 2\kappa' u_{xx} + \kappa u_{xxx} + f_x\right)_x \tag{3.46}$$

$$= \kappa''' u_x + 3\kappa'' u_{xx} + 3\kappa' u_{xxx} + \kappa u_{xxxx} + f_{xx} \tag{3.47}$$

$$u_{tt} = \left(\kappa' u_x + \kappa u_{xx} + f\right)_t \tag{3.48}$$

$$= \kappa' u_{xt} + \kappa u_{xxt} + f_t \tag{3.49}$$

$$= \kappa' \left(\kappa'' u_x + 2\kappa' u_{xx} + \kappa u_{xxx} + f_x\right) \tag{3.50}$$

$$+ \kappa \left(\kappa''' u_x + 3\kappa'' u_{xx} + 3\kappa' u_{xxx} + \kappa u_{xxxx} + f_{xx}\right) + f_t \tag{3.51}$$

$$= \kappa'\kappa'' u_x + \kappa\kappa''' u_x + 2(\kappa')^2 u_{xx} + 3\kappa\kappa'' u_{xx} \tag{3.52}$$

$$+ 4\kappa\kappa' u_{xxx} + \kappa' f_x + \kappa^2 u_{xxxx} + \kappa f_{xx} + f_t \tag{3.53}$$

Inserting (3.44)-(3.53) into (3.37)-(3.41) yields

$$\tau(x,t) = \kappa' u_x + \kappa u_{xx} + f \tag{3.54}$$

$$+ \frac{k}{2} \begin{pmatrix} \kappa'\kappa'' u_x + \kappa\kappa''' u_x + 2(\kappa')^2 u_{xx} + 3\kappa\kappa'' u_{xx} \\ +4\kappa\kappa' u_{xxx} + \kappa' f_x + \kappa^2 u_{xxxx} + \kappa f_{xx} + f_t \end{pmatrix} \tag{3.55}$$

$$- f - \kappa' u_x - \kappa'\theta_1 k \left(\kappa'' u_x + 2\kappa' u_{xx} + \kappa u_{xxx} + f_x\right) \tag{3.56}$$

$$- \kappa u_{xx} - \kappa\theta_2 k \left(\kappa''' u_x + 3\kappa'' u_{xx} + 3\kappa' u_{xxx} + \kappa u_{xxxx} + f_{xx}\right) \tag{3.57}$$

$$+ O\left(h^2 + k^2\right) \tag{3.58}$$

$$= k \begin{pmatrix} \left(\frac{1}{2} - \theta_1\right)\kappa'\kappa'' u_x + \left(\frac{1}{2} - \theta_2\right)\kappa\kappa''' u_x \\ +3\left(\frac{1}{2} - \theta_2\right)\kappa\kappa'' u_{xx} + 2\left(\frac{1}{2} - \theta_2\right)(\kappa')^2 u_{xx} \\ +4\left(\frac{1}{2} - \theta_2\right)\kappa\kappa' u_{xxx} + \left(\frac{1}{2} - \theta_2\right)\kappa^2 u_{xxxx} \end{pmatrix} \tag{3.59}$$

$$+ \left(\frac{1}{2} - \theta_1\right) k\kappa' f_x + \left(\frac{1}{2} - \theta_2\right) k\kappa f_{xx} + \frac{k}{2} f_t \tag{3.60}$$

$$+ O\left(h^2 + k^2\right) \tag{3.61}$$

We now define the dominant term in the above truncation error by

$$\mu(u,\kappa;\theta_1,\theta_2) = \left(\frac{1}{2} - \theta_1\right)\kappa'\kappa'' u_x + \left(\frac{1}{2} - \theta_2\right)\kappa\kappa''' u_x \tag{3.62}$$

$$+ 3\left(\frac{1}{2} - \theta_2\right)\kappa\kappa'' u_{xx} + 2\left(\frac{1}{2} - \theta_2\right)(\kappa')^2 u_{xx} \tag{3.63}$$

$$+ 4\left(\frac{1}{2} - \theta_2\right)\kappa\kappa' u_{xxx} + \left(\frac{1}{2} - \theta_2\right)\kappa^2 u_{xxxx} \tag{3.64}$$

and consider two cases with respect to $\mu(u,\kappa;\theta_1,\theta_2)$.

1. Case $\mu(u,\kappa;\theta_1,\theta_2) \not\equiv 0$ in $(0,1)\times[0,T]$. In this case, we only obtain

$$\tau(x,t) = O\left(h^2 + k\right) \tag{3.65}$$

   Our θ method in this case is said to be *second order accurate in space* and *first order accurate in time* since the truncation error is $O\left(h^2 + k\right)$.

2. Case $\mu(u,\kappa;\theta_1,\theta_2) \equiv 0$ in $(0,1)\times[0,T]$. Since u is unknown in general, a sufficient condition for which

$$\mu(u,\kappa;\theta_1,\theta_2) \equiv 0 \text{ in } (0,1)\times[0,T] \tag{3.66}$$

holds is

$$\left(\frac{1}{2}-\theta_1\right)\kappa'\kappa''+\left(\frac{1}{2}-\theta_2\right)\kappa\kappa'''=0 \tag{3.67}$$

$$3\left(\frac{1}{2}-\theta_2\right)\kappa\kappa''+2\left(\frac{1}{2}-\theta_2\right)(\kappa')^2=0 \tag{3.68}$$

$$4\left(\frac{1}{2}-\theta_2\right)\kappa\kappa'=0 \tag{3.69}$$

$$\left(\frac{1}{2}-\theta_2\right)\kappa^2=0 \tag{3.70}$$

in $(0,1)$.

Due to (1.1), $\kappa>0$. Combining this with (3.70) yields

$$\theta_2=\frac{1}{2} \tag{3.71}$$

Under (3.71), (3.67)-(3.70) remains only

$$\left(\frac{1}{2}-\theta_1\right)\kappa'\kappa''=0 \tag{3.72}$$

or equivalently,

$$\left(\theta_1=\frac{1}{2}\right)\vee(\kappa'=0)\vee(\kappa''=0) \tag{3.73}$$

Hence, if the given function κ is a constant or a linear function in $(0,1)$, (3.67)-(3.70) still holds for $\theta_2=\frac{1}{2}$ and an arbitrary θ_1.

On the other hand, if the given function κ satisfying $\kappa''\neq 0$, then $\theta_1=\frac{1}{2}$ must holds for which (3.67)-(3.70) still holds.

If we move all the terms involving f in the local truncation error (3.54)-(3.61), we will obtain the following result.

Proposition. *If κ,θ_1,θ_2 satisfy one of two cases below*

(a) $\theta_1=\frac{1}{2},\theta_2=\frac{1}{2}$

(b) $\theta_2=\frac{1}{2}$ *and κ is a constant or a linear function on* $(0,1)$.

then

$$\tau(x,t)=O\left(h^2+k^2\right) \tag{3.74}$$

Our θ method in these special cases is said to be *second order accurate in space* and *second order accurate in time* since the truncation error is $O\left(h^2+k^2\right)$.

Remark. The Crank-Nicolson method is centered in both space and time, and the above analysis of its local truncation error shows that it is second order accurate in both space and time,

$$\tau(x,t)=O\left(h^2+k^2\right) \tag{3.75}$$

A method is said to be *consistent* if

$$\tau(x,t) \to 0 \text{ as } h,k \to 0 \tag{3.76}$$

Just as in the other cases we have studied (boundary value problems and initial value problems for ordinary differential equations (ODEs)), we expect that consistency, plus some form of stability, will be enough to prove that the method converges at each fixed point (X,T) as we refine the grid in both space and time. Moreover, we expect that for a stable method the global order of accuracy will agree with the order of the local truncation error, e.g., for Crank-Nicolson we expect that

$$U_i^n - u(X,T) = O\left(k^2 + h^2\right) \tag{3.77}$$

as $k,h \to 0$ when $ih \equiv X$ and $nk \equiv T$ are fixed.

For linear PDEs, the fact that consistency plus stability is equivalent to convergence is known as the *Lax equivalence theorem*. As usual, it is the definition and study of stability that is the hard (and interesting) part of this theory.

4 Method of Lines Discretizations

To understand how stability theory for time-dependent PDEs relates to the stability theory we have already developed for time-dependent ODEs, see [1], it is easiest to first consider the so-called method of lines (MOL) discretization of the PDE. In this approach we first discretize in space alone, which gives a large system of ODEs with each component of the system corresponding to the solution at some grid point, as a function of time. The system of ODEs can then be solved using one of the methods for ODEs.

This system of ODEs is also often called a *semidiscrete* method, since we have discretized in space but not yet in time.

For our problem, we might discretize the heat equation (1.2) in space at grid point x_i by

$$U_i'(t) = \kappa'(x_i)\frac{U_{i+1}(t) - U_{i-1}(t)}{2h} + \kappa(x_i)\frac{U_{i-1}(t) - 2U_i(t) + U_{i+1}(t)}{h^2} \tag{4.1}$$

for $i = 1, 2, \ldots, N_x$, where prime now means differentiation with respect to time. We can view this as a coupled system of m ODEs for the variables $U_i(t)$, which vary continuously in time along the lines shown in Figure 2.

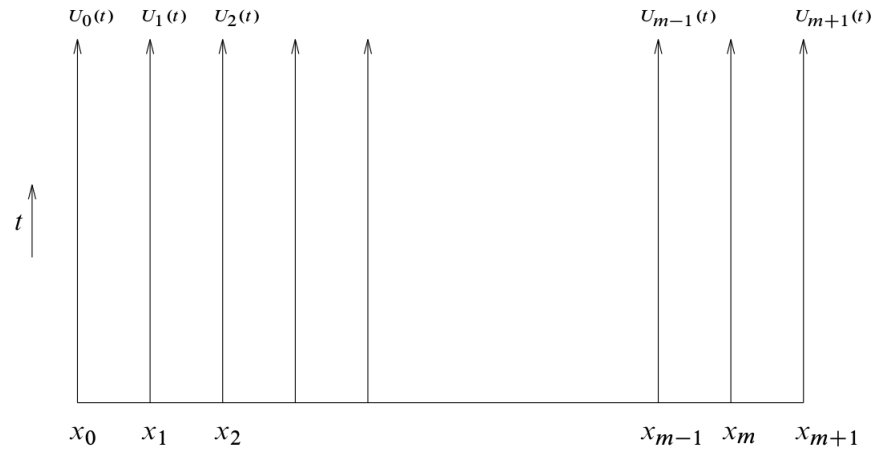

Figure 2: Method of lines interpretation. $U_i(t)$ is the solution along the line forward in time at the grid point x_i.

This system can be written as

$$U'(t) = AU(t) \tag{4.2}$$

where the tridiagonal matrix A is

$$A \tag{4.3}$$

$$= \begin{bmatrix} -\frac{2\kappa(x_1)}{h^2} & \frac{\kappa'(x_1)}{2h} + \frac{\kappa(x_1)}{h^2} & & \\ -\frac{\kappa'(x_2)}{2h} + \frac{\kappa(x_2)}{h^2} & -\frac{2\kappa(x_2)}{h^2} & \frac{\kappa'(x_2)}{2h} + \frac{\kappa(x_2)}{h^2} & \\ & \ddots & \ddots & \ddots \\ & & -\frac{\kappa'(x_{N_x})}{2h} + \frac{\kappa(x_{N_x})}{h^2} & -\frac{2\kappa(x_{N_x})}{h^2} \end{bmatrix} \tag{4.4}$$

Remark. In the inhomogeneous case, the system can be written as

$$U'(t) = AU(t) + g(t) \tag{4.5}$$

where the tridiagonal matrix A is given by (4.3)-(4.4) and $g(t)$ includes the terms needed for the boundary conditions,

$$U_0(t) \equiv g_0(t) \tag{4.6}$$

$$U_{N_x+1}(t) \equiv g_1(t) \tag{4.7}$$

i.e.,

$$g(t) = \begin{bmatrix} \left(\frac{\kappa(x_1)}{h^2} - \frac{\kappa'(x_1)}{2h}\right) g_0(t) \\ 0 \\ \vdots \\ 0 \\ \left(\frac{\kappa(x_{N_x})}{h^2} + \frac{\kappa'(x_{N_x})}{2h}\right) g_1(t) \end{bmatrix} \tag{4.8}$$

This MOL approach is sometimes used in practice by first discretizing in space and then applying a software package for systems of ODEs. There are also packages that are specially designed to apply MOL. This approach has the advantage of being relatively easy to apply to a fairly general set of time-dependent PDEs, but the resulting method is often not as efficient as specially designed methods for the PDE.

As a tool in understanding stability theory, however, the MOL discretization is extremely valuable, and this is the main use we will make of it. We know how to analyze the stability of ODE methods applied to a linear system of the form (4.2) and (4.5) based on the eigenvalues of the matrix A, which now depend on the spatial discretization.

If we apply an ODE method to discretize the system (4.2) and (4.5), we will obtain a fully discrete method which produces approximations

$$U_i^n \approx U_i(t_n) \tag{4.9}$$

at discrete points in time which are exactly the points (x_i, t_n) of the grid that we introduced at the beginning of this context.

For example, applying Euler's method

$$U^{n+1} = U^n + kF(U^n) \tag{4.10}$$

to this linear system results in the fully discrete method (2.8)-(2.9). Applying instead the trapezoidal method results in the Crank-Nicolson method (2.10)-(2.11). Applying a higher order linear multistep or Runge-Kutta method would give a different method, although with the spatial discretization (4.1) the overall method would be only second order accurate in space. Replacing the right-hand side of (4.1) with a higher order approximation to $u_x(x_i)$ and $u_{xx}(x_i)$ and then using a higher order time discretization would give a more accurate method.

5 Stability Theory

6 Forward Euler Method

6.1 Local Truncation Errors and Order of Accuracy

We can define the local truncation error as usual, we insert the exact solution $u(x,t)$ of the PDE into the finite difference equation and determine by how much it fails to satisfy the discrete equation.

The local truncation error of the Forward Euler method is based on the form

$$\tau_i^n = \frac{u(x_i, t_{n+1}) - u(x_i, t_n)}{k} - \kappa'(x_i) \frac{u(x_{j+1}, t_n) - u(x_{j-1}, t_n)}{2h} \tag{6.1}$$

$$- \kappa(x_i) \frac{u(x_{j-1}, t_n) - 2u(x_i, t_n) + u(x_{j+1}, t_n)}{h^2} - f(x_i, t_n) \tag{6.2}$$

Again we should be careful to use the form that directly models the differential equation in order to get powers of k and h that agree with what we hope to see in the global error. Although we do not know $u(x,t)$ in general, if we assume that it is smooth and use Taylor series expansions about $u(x,t)$,

$$f \tag{6.3}$$

7 Backward Euler Method

8 Crank-Nicolson Method

9 Practical Problems

Problem. *For $f \in L^2([0,1] \times [0,T])$ and we consider heat equation*

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + f(x,t), \forall (x,t) \in (0,1) \times (0,T] \tag{9.1}$$

Along with this equation we need initial conditions at time 0.

$$u(x,0) = u_0(x) \tag{9.2}$$

and also boundary conditions if we are working on a bounded domain, e.g., the Dirichlet conditions

$$u(t,0) = 0 \tag{9.3}$$
$$u(t,1) = 0 \tag{9.4}$$

1. *Find the discrete solution using finite difference scheme with 3 methods (Forward Euler, Backward Euler and Crank-Nicolson) and consider them when*

$$u_{ex}(x,t) = x(1-x)^2 e^{-2t} \tag{9.5}$$
$$f(x,t) = \left(2 - 8x + 4x^2 - 2x^3\right) e^{-2t} \tag{9.6}$$

2. *Compute the error for 3 methods on space with discrete H_0^2- and L^2-norms when $k = 0.01, k = 0.005, k = 0.0001$ and $k = h^2$. Do you consider about these errors.*

References

[1] Randall J. Leveque, *Finite Difference Methods for Ordinary and Partial Differential Equations*, SIAM Society for Industrial and Applied Mathematics, 2007.

[2] Le Anh Ha, *Heat Equation in 1D*, University of Sciences, October 31, 2015.

[3] https://hongnguyenquanba.wordpress.com/2016/10/17/assignment-on-approximating-solutions-of-one-dimensional-boundary-value-problems-by-

[4] https://hongnguyenquanba.wordpress.com/2016/11/29/assignment-on-approximating-solutions-of-two-dimensional-boundary-value-problems-by-