Discretization and Numerical Solution of Partial Differential Equation

Luo Tao, Wenchao Zhang, Qikun Wu, Jiale Wang
South University of Science and Technology of China

January 3, 2013

Contents

Abstract

A thorough exploration of discretization method and MatLab implementation of PDE is illustrated. Methods of Dirichlet, Neumann, Forward-Difference and Crank-Nicolson are presented in detail to deal with three kinds of PDE: Hyperbolic, Parabolic and Elliptic. Besides, a general solution Runge-Kutta method is showed to solve ODE system after the discretization of PDE.

1 Background of PDE

1.1 Hyperbolic Equation

Equation

$$\frac{d^2u}{dt^2} = c^2\nabla^2 u \tag{1.1}$$

is called a hyperbolic equation.

1.1.1 wave propagation

consider the longitudinal wave propagating in a rod, we divide it into infinitesimal elements. for each element $i|i = 1.2...n$ start from $x_i$to x_{i+1}, equation of motion is

$$\mathrm{m_i}\frac{\mathrm{d^2u}}{\mathrm{dt^2}} = \mathrm{k(u_{i+1} - u_i) - (u_i - u_{i-1})} \tag{1.2}$$

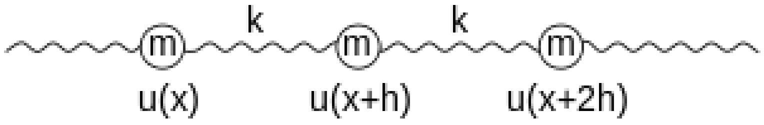
where u is the displacement along the rod.as $K = k/N, \triangle x = l/N, m_i = M/N$, we get

$$\frac{d^2u}{dt^2} = \frac{KL^2}{M}\frac{(u_{i+1} - 2u_i + u_{i-1})}{\triangle x^2}$$

which becomes

$$\frac{d^2u}{dt^2} = \frac{KL^2}{M}\frac{d^2u}{dx^2} \tag{1.3}$$

when $n \to +\infty$



1.1.2 Transverse Wave Propagation

In Fig 1.1, consider the transverse wave propagating in a string,assume the tension is f, and vibration don't have a spatial frequency such that $u_x$is small enough. we divide it into infinitesimal elements. for each point i, the net force perpendicular to string is $\frac{f}{\Delta x}(u_{i-1} - 2u_i + u_{i+1})$, pay attention that here we use the approximation $\cos(\theta) \approx tan(\theta)$ as $\theta \to 0$.equation of motion holds

$$m_i\frac{d^2u}{dt^2} = \frac{f}{\Delta x}(u_{i-1} - 2u_i + u_{i+1})$$

replace m_i by $\Delta x\rho$ and let $\Delta x \to 0$, we get

$$\frac{d^2u}{dt^2} = \frac{f}{\rho}\frac{d^2u}{dx^2} \tag{1.4}$$

1.1.3 High dimension Wave Propagation

Similarly, in high dimension case, wave function become $\frac{d^2u}{dt^2} = c^2\nabla^2 u$ where $c^2$is a positive constant.

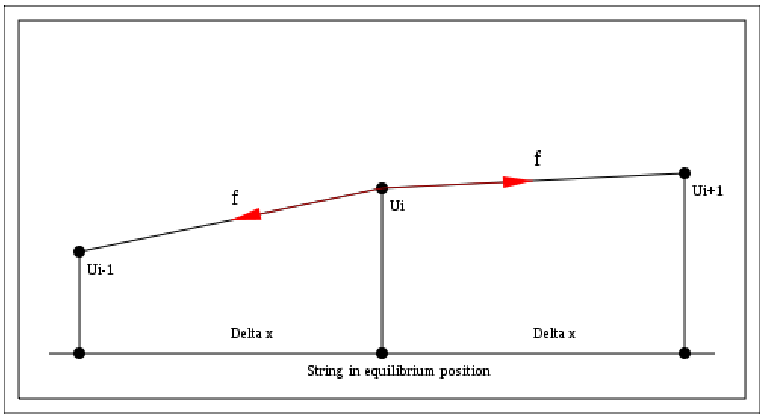


Figure 1.1: String Motion in Equilibrium

1.2 Parabolic Equation

Equation

$$\frac{du}{dt} = \alpha \nabla^2 u \tag{1.5}$$

where α is a positive constant, is called a parabolic equation (also heat equation).

By Fourier's Law$V_q = -k\nabla u$, take the surface integral on both sides we get

$$\oiint V_q d\sigma = -k \oiint \nabla u d\sigma \tag{1.6}$$

The left side means the heat flow out of the surface i.e. $-\iiint q_t dv$.
The right side, by Gauss's Law, can be written as volumn integral $-k\iiint \nabla^2 u dv$. On the other side, $\Delta q = c_p \rho \Delta u$ hence $q_t = c_p \rho u_t$.so just by arranging equation 1.6,it becomes

$$u_t = \frac{k}{c_p \rho} \nabla^2 u \tag{1.7}$$

where $\frac{k}{c_p\rho} = \alpha$.

heat function can also appear in difusion problem, the only difference is to consider u to be particle number density.the famous Schrodinger equation for a free particle is also heat equation basicly, except that α here is a complex number.

1.3 Elliptic Equation

Laplace equation

$$\Delta\varphi = 0 \tag{1.8}$$

and Possion equation

$$\Delta\varphi = f \tag{1.9}$$

are both elliptic equation.

In Electrostatics, $\varphi$is electrical potential and $\Delta\varphi$means charge density. So above equation means no charge case and charge with f distribution case.
In Newtonian gravity, f is density and $\varphi$is gravitational potential.

2 Definite Problem of Partial Differential Equation

2.1 Elliptical Partial Differential Equation

The process that fixes time can be described as an elliptical partial differential equation.The most typical and simplest form of elliptical equation is Poisson's equation

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) \tag{2.10}$$

Particularly,if $f(x,y) \equiv 0$,then the equation is called Laplace's equation.

More over, if we give some definite conditions to the equation then the problem become definite problem.Such definite conditions are initial condition and boundary condition.The boundary value problem of Poisson equation is [4]

$$\begin{cases} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x,y) & (x,y) \in \Omega \\ u(x,y)\big|_{(x,y)\in\Gamma} = \varphi(x,y) & \Gamma = \partial\Omega \end{cases}$$

where Ω is a bounded(by Γ) area, Γ is segmental smooth curve.$f(x,y)$ and $\varphi(x,y)$ are continous function.

Another famous elliptical partial differential equation is Helmholtz equation:$\Delta u + f(x,y)\,u = g(x,y)$[5] .

2.2 Parabolic Partial Differential Equation

When we study the problem like heat conduction,we may face such parabolic equation.The simplest form of parabolic equation is one order form parabolic equation:

$$\frac{\partial u}{\partial t} - a\frac{\partial^2 u}{\partial x^2} = 0\,(a > 0) \tag{2.11}$$

Equation 2.11 has two different definite problem: initial value problem(Cauchy Problem) and initial boudary value problem:

1. Initial Value Problem(IVP)

$$\begin{cases} \frac{\partial u}{\partial t} - a\frac{\partial^2 u}{\partial x^2} = 0 & t > 0, -\infty < x < +\infty \\ u(x,0) = \varphi(x) & -\infty < x < +\infty \end{cases} \tag{2.12}$$

2. Initial Boundary Value Problem(BVP)

$$\begin{cases} \frac{\partial u}{\partial t} - a\frac{\partial^2 u}{\partial x^2} = 0 & 0 < t < T, 0 < x < l \\ u(x,0) = \varphi(x) & 0 \leqslant x \leqslant l \\ u(0,t) = g_1(t), u(l,t) = g_2(t) & 0 \leqslant t \leqslant T \end{cases} \tag{2.13}$$

where $\varphi(x), g_1(x), g_2(x)$ are known and $\varphi(0) = g_1(0), \varphi(l) = g_2(0)$.

2.3 Hyperbolic Partial Differential Equation

Wave equation is a classic form of Hyperbolic equation with two order.The simplest form of hyperbolic equation is one order form:

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \tag{2.14}$$

The form we are familiar in physical: one dimensional shock and wave equation is following:

$$\frac{\partial^2 u}{\partial t^2} = a^2\frac{\partial^2 u}{\partial x^2} \tag{2.15}$$

Similarly with Parobolic form,we have IVP and BVP:

1. Initial Value Problem(IVP)

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} = a^2\frac{\partial^2 u}{\partial x^2} & t > 0, -\infty < x < +\infty \\ u(x,0) = \varphi(x) & -\infty < x < +\infty \\ \frac{\partial u}{\partial t}|_{t=0} = \phi(x) & -\infty < x < +\infty \end{cases} \tag{2.16}$$

2. Initial Boundary Value Problem(BVP)

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} = a^2\frac{\partial^2 u}{\partial x^2} & t > 0, 0 < x < l \\ u(x,0) = \varphi(x), \frac{\partial u}{\partial t}|_{t=0} = \phi(x) & 0 \leqslant x \leqslant l \\ u(0,t) = g_1(t), u(l,t) = g_2(t) & 0 \leqslant t \leqslant T \end{cases} \tag{2.17}$$

where $\varphi(x), g_1(x), g_2(x)$ are known and $\varphi(0) = g_1(0), \varphi(l) = g_2(0)$.

3 solution to ODE system [1][2]

3.1 matrix multiplication

3.1.1 norm

Let X be a real vector space. A norm on X is a map $\|.\| : X \to [0, \infty)$ satisfying the following requirements:

I. $\|0\| = 0, \|x\| > 0$ for$x \in X \backslash \{0\}$

II. $\|\lambda x\| = |\lambda| \, \|x\|$ for $\lambda \in \mathbb{R}$ and $x \in X$

III. $\|x + y\| \leq \|x\| + \|y\|forx, y \in X$

The pair $(X, \|.\|)$ is called a normed vector space. Given a normed vector space X, we have the concept of convergence and of a Cauchy sequence in this space. The normed vector space is called complete if every Cauchy sequence converges. A complete normed vector space is called a Banach space. For any vector x, three kinds of norm (1-norm, 2-norm and ∞-norm) are usually used. In general, the p-norm of a vector is represented as:

$$\|x\|_p = \left[\sum_{i=1}^{n} |x_i|^p \right]^{\frac{1}{p}} \tag{3.18}$$

Since $\mathbb{C}^n$ is algebraically closed, we use $\mathbb{C}^n$ as underlying vector space rather than $\mathbb{R}^n$.

The norm of matrix is introduced as follows:

$$\|\mathbf{A}\| = \sup_{x:|x|=1} |\mathbf{A}x| \tag{3.19}$$

We can then see that the space of $n \times n$ matrices becomes a Banach space. Since matrix norm is just a kind of norm, we can see that it also satisfy the nature of norm:

I. $\|\mathbf{0}\| = 0, \|\mathbf{A}\| > 0$ for$\mathbf{A} \in \mathbf{L}(\mathbf{R}^n) \backslash \{\mathbf{0}\}$

II. $\|\lambda \mathbf{A}\| = |\lambda| \, \|\mathbf{A}\|$ for $\lambda \in \mathbf{R}$ and $\mathbf{A} \in \mathbf{L}(\mathbf{R}^n)$

III. $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|for\mathbf{A}, \mathbf{B} \in \mathbf{L}(\mathbf{R}^n)$

So we have following Lemma:

Lemma 1 *for* $\forall \mathbf{A}, \mathbf{B} \in \mathbf{L}(\mathbf{R}^n)$ *and* $x \in \mathbf{R}^n$*:*

I. $|\mathbf{A}x| \leq \|\mathbf{A}\| \, |x|$

II. $\|\mathbf{AB}\| \leq \|\mathbf{A}\| \, \|\mathbf{B}\|$

III. $\left\|\mathbf{A}^k\right\| \leq \|\mathbf{A}\|^k$

Proof.
(i)by definition, $\|\mathbf{A}\| \geq \left|\mathbf{A}\frac{x}{|x|}\right| = \frac{|\mathbf{A}x|}{|x|}$
(ii)from (i) we have for $|x| \leq 1$:

$$|\mathbf{AB}x| \leq \|\mathbf{A}\| \, |\mathbf{B}x| \leq \|\mathbf{A}\| \, \|\mathbf{B}\| \, |x|$$

(iii) from (ii)we can easily proof use deduction ■

So the convergence of the series $\sum\limits_{k=0}^{\infty} \frac{\mathbf{A}^k}{k!}$ can be proofed in the following theorem:

Theorem 2 *For* $\forall \mathbf{A} \in \mathbf{L}(\mathbf{R}^n)$ *and* $t_0 > 0$ *,*

$$\sum_{k=0}^{\infty} \frac{\mathbf{A}^k t^k}{k!}$$

converge uniformly for $\forall \, |t| < t_0$

The theorem is easily proofed by using

$$\left\|\mathbf{A}^k\right\| \leq \|\mathbf{A}\|^k$$

We can then define matrix exponential.

3.1.2 matrix exponential

A matrix exponential of a matrix A is given by:

$$e^{\mathbf{A}} = \exp(\mathbf{A}) = \sum_{i=0}^{\infty} \frac{1}{i!}\mathbf{A}^i$$

However, note that in general, $\exp(\mathbf{A}+\mathbf{B}) \neq \exp(\mathbf{A})\exp(\mathbf{B})$ Unless the communicator $[\mathbf{A},\mathbf{B}] = \mathbf{AB}-\mathbf{BA} = \mathbf{0}$ By the theorem 2, we can see that $\exp(\mathbf{A}) = \sum\limits_{i=0}^{\infty} \frac{1}{i!}\mathbf{A}^i$converge for $\forall \mathbf{A}$. Moreover, we can have some properties:

(i) if $\mathbf{B} = \mathbf{PAP^{-1}}$ then $e^{\mathbf{B}}= \mathbf{P}e^{\mathbf{A}}\mathbf{P^{-1}}$

(ii) if $\mathbf{A} = \mathbf{P}diag\left(\lambda_j\right)\mathbf{P^{-1}}$, then $e^{\mathbf{A}t} = \mathbf{P}diag\left(e^{\lambda_j t}\right)\mathbf{P}^{-1}$

The first property can be done easily by calculation:

$$e^{\mathbf{B}}= \exp(\mathbf{B}) = \sum_{i=0}^{\infty}\frac{1}{i!}\mathbf{B}^i = \sum_{i=0}^{\infty}\frac{1}{i!}\left(\mathbf{PAP^{-1}}\right)^i = \sum_{i=0}^{\infty}\frac{1}{i!}\mathbf{PA}^i\mathbf{P^{-1}} = \mathbf{P}e^{\mathbf{A}}\mathbf{P^{-1}}$$

And the second property can be directly gained from the first property.

3.1.3 Jordan canonical form[3]

We can use Jordan canonical form to solve a wide range of ODE system $x. = Ax$

Theorem 3 *(Cayley-Hamilton) Every matrix satisfies its own characteristic equation*

$$\chi_A(\mathbf{A}) = 0 \tag{3.20}$$

Proof. Assume that $\alpha_1, \alpha_2, \ldots \alpha_n$ is the base of linear space V, and A is the matrix representation of linear transformation $\mathbf{A}$, then we have:

$$(\mathbf{A}\alpha_1, \mathbf{A}\alpha_2, \ldots \mathbf{A}\alpha_n) = (\alpha_1, \alpha_2, \ldots \alpha_n)A \tag{3.21}$$

Which can be written as:

$$\begin{pmatrix} \mathbf{A} & & & \\ & \mathbf{A} & & \\ & & \ddots & \\ & & & \mathbf{A} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} = A^T \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} \tag{3.22}$$

We denote the equation above as:

$$\begin{pmatrix} \lambda & & & \\ & \lambda & & \\ & & \ddots & \\ & & & \lambda \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} = A^T \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} \tag{3.23}$$

Then we simply have:

$$(\lambda I - A^T)\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} = 0, \chi_A(A) = \det\left|\lambda I - A^T\right| = 0 \tag{3.24}$$

■

The aim of Jordan canonical form is to decompose the linear space. We first should spilt our space into some subspace and then apply a more accurate decomposition to each subspace.

Definition 4 *(generalized eigenspace)*
A generalized eigenspace is the kernel of polynomial $f(\lambda)$:
$\ker f = \ker f(\lambda) = \{\alpha \in V | f(\boldsymbol{A})\alpha = 0\}$

Under such definition, we can first apply a primary decomposition to our linear space V:

Theorem 5 *(primary decomposition)*
let $m(\lambda)$ be the minimal annihilator of linear space V on field F.

$$m(\lambda) = p_1(\lambda)^{r_1} p_2(\lambda)^{r_2} \dots p_s(\lambda)^{r_s} \tag{3.25}$$

Then the generalized eigenspace of $p_i(\lambda)^{r_i}$, saying W_i, is an invariant subspace of V and

$$V = W_1 \oplus W_2 \oplus \dots \oplus W_s \tag{3.26}$$

The proof is omitted since it is not what we are focus on. After the primary decomposition, we know that the characteristic equation of a $n \times n$ matrix A can be written as

$$f(\lambda) = (\lambda - \lambda_1)^{d_1} (\lambda - \lambda_2)^{d_2} \dots (\lambda - \lambda_s)^{d_s} \tag{3.27}$$

where $d_1 + d_2 + \dots + d_s = n$. Moreover, the minimal annihilator is

$$m(\lambda) = (\lambda - \lambda_1)^{r_1} (\lambda - \lambda_2)^{r_2} \dots (\lambda - \lambda_s)^{r_s} \tag{3.28}$$

where $r_i \leq d_i$ for $i = 1 \dots s$ Our target is to find the Jordan canonical form J and the P satisfying $P^{-1}AP = J$. The method can be illustrated in the following way:

I. Write down the characteristic equation $f(\lambda)$ of A. we have

$$f(\lambda) = (\lambda - \lambda_1)^{d_1} (\lambda - \lambda_2)^{d_2} \dots (\lambda - \lambda_s)^{d_s}, \sum d_i = n$$

Then we can know that

$$A \sim diag\{J_{d_1}(\lambda_1), J_{d_2}(\lambda_2), \dots, J_{d_s}(\lambda_s)\} = J$$

II. Then we look into $J_{d_i}(\lambda_i)$. We know that $J_{d_i}(\lambda_i)$ is consists of several Jordan blocks. Furthermore, the number of Jordan blocks is the number of linear independent eigenvector whose eigenvalue is λ_i. This number also equals to the dimension of $\ker(A - \lambda I)$.

III. After we know the number of Jordan blocks for each eigenvalue, we should exactly know the number of $k \times k$ Jordan block N_{k,λ_i} which eigenvalue is λ_i. We have:

$$N_{k,\lambda_i} = [rank(A - \lambda_i I)^{k-1} - rank(A - \lambda_i I)^k] - [rank(A - \lambda_i I)^k - rank(A - \lambda_i I)^{k+1}]$$

IV. We choose a set of vector $\{a\}$ which satisfies

$$a \in \ker{(A - \lambda I)^k}, a \notin \ker{(A - \lambda I)^{k-1}}$$

And gain a Jordan chain

$$a_j, (A - \lambda_i I)a_j, \dots, (A - \lambda_i I)^{k-1} a_j$$

for each $a_j \in \{a\}$. Then for all eigenvalues, the Jordan chain produce the P we wanted.

3.2 Solution to ODE systems

3.2.1 homogeneous linear system $x' = Ax$

We first consider the scenario that our first order linear ODEs system is in the form $\mathbf{x}' = \mathbf{A}\mathbf{x}$ where $\mathbf{A}$ is time-independent. Like the first order ODE and second order ODE, there is theorem claims about the existence and uniqueness of the solution:

Theorem 6 *(Existence and uniqueness)* $\mathbf{A}$ *is a* $n \times n$ *matrix which is time-independent. Then for* $\forall \mathbf{x_0} \in \mathbf{R}^n$*, there exist a unique solution to initial value problem for*

$$\begin{cases} \mathbf{x}' = \mathbf{A}\mathbf{x} \\ \mathbf{x}(0) = \mathbf{x_0} \end{cases} \tag{3.29}$$

And the solution is: $\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x_0}$

The proof of the theorem require the flowing lemma:

Lemma 7 $\frac{d}{dt}e^{\mathbf{A}t} = \mathbf{A}e^{\mathbf{A}t}$

Proof.
$\frac{d}{dt}e^{\mathbf{A}t} = \lim\limits_{\Delta t \to 0} \frac{e^{\mathbf{A}(t+\Delta t)} - e^{\mathbf{A}t}}{\Delta t} = e^{\mathbf{A}t} \lim\limits_{\Delta t \to 0} \frac{e^{\mathbf{A}\Delta t} - I}{\Delta t} = e^{\mathbf{A}t} \lim\limits_{\Delta t \to 0} \sum\limits_{i=1}^{\infty} \frac{1}{i!}\mathbf{A}^i \Delta t^{i-1} = e^{\mathbf{A}t}\mathbf{A}$ ■

Now use this lemma we can proof the theorem we mentioned above:

Proof.
(Existence) Using the lemma, if $\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x_0}$ then $\frac{d}{dt}\mathbf{x}(t) = \frac{d}{dt}e^{\mathbf{A}t}\mathbf{x_0} = \mathbf{A}e^{\mathbf{A}t}\mathbf{x_0} = \mathbf{A}\mathbf{x}(t)\ for\ \forall t \in \mathbf{R}$ And $\mathbf{x}(0) = I\mathbf{x_0} = \mathbf{x_0}$. So $\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x_0}$ is indeed a solution to the IVP. The existence is then been proofed.
(Uniqueness) we assume that there exist another solution to such IVP, let' s say it is $\mathbf{z}(t)$. We let $\mathbf{y}(t) = e^{-\mathbf{A}t}\mathbf{z}(t)$, then: $\mathbf{y}'(t) = -\mathbf{A}e^{-\mathbf{A}t}\mathbf{z}(t) + e^{-\mathbf{A}t}\mathbf{z}'(t) = -\mathbf{A}e^{-\mathbf{A}t}\mathbf{z}(t) + e^{-\mathbf{A}t}\mathbf{A}\mathbf{z}(t) = \mathbf{0}$ Which means in fact $\mathbf{y}(t)$is a constant. And by the initial condition, $\mathbf{y}(t) \equiv \mathbf{y}(0) = \mathbf{x_0}$. Then is it obvious that $\mathbf{z}(t) = e^{\mathbf{A}t}\mathbf{y}(0) = e^{\mathbf{A}t}\mathbf{x_0}$, the uniqueness is then proofed. ■

With the theorem about the existence and uniqueness of the solution, we can then simply calculate the solution of our IVP by first transform our matrix $\mathbf{A}$ in a simplified form(where the most complicated form is Jordan canonical form), then calculate $\exp(\mathbf{A})$ and finally get our answer.

3.2.2 nonhomogeneous linear system $x' = Ax + B(t)$

We have discussed in previous subsection how to get a solution to homogeneous linear system $x. = Ax$ where $\mathbf{A}$ is time independent and we now solve for a more general case where external force is involved and is presented as a time-dependent form $\mathbf{B}(t)$. We first gain a **fundamental matrix solution** Φ by taking n linear independent solutions $\phi_1, \phi_2 \dots \phi_n$ where their **Wronski determinant**:

$$W(t) = \det(\phi_1, \phi_2 \dots \phi_n)$$

Is not equal to 0 at any point $t \in I$. Then $\Phi = (\phi_1, \phi_2 \dots \phi_n)$. We can know that $\Phi(t)\Phi^{-1}(0) = e^{\mathbf{A}t}$. We use the method of variation of constants (also variation of parameters) by assuming the particular solution to the nonhomogeneous system is:

$$\mathbf{x}(t) = \Phi(t)\mathbf{c}(t) \tag{3.30}$$

Then:

$$\begin{aligned} \mathbf{x}'(t) &= \Phi'(t)\mathbf{c}(t) + \Phi(t)\mathbf{c}'(t) \\ &= \mathbf{A}\Phi(t)\mathbf{c}(t) + \Phi(t)\mathbf{c}'(t) \\ &= \mathbf{A}\mathbf{x}(t) + \Phi(t)\mathbf{c}'(t) \\ \mathbf{x}'(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}(t) \end{aligned} \tag{3.31}$$

Which explicitly say that$\Phi(t)\mathbf{c}'(t) = \mathbf{B}(t)$. Then

$$\mathbf{c}'(t) = \Phi^{-1}(t)\mathbf{B}(t)$$

and we integral both sides yields:

$$\mathbf{c}(t) = \Phi^{-1}(t_0)\mathbf{x}_0 + \int\limits_{t_0}^{t} \Phi^{-1}(s)\mathbf{B}(s)ds \tag{3.32}$$

Then the general solution to nonhomogeneous system

$$\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}(t)$$

is given by:

$$\mathbf{x}(t) = \Phi(t)\Phi^{-1}(t_0)\mathbf{x}_0 + \int_{t_0}^{t} \Phi(t)\Phi^{-1}(s)\mathbf{B}(s)ds \tag{3.33}$$

3.2.3 homogeneous linear system $x' = A(t)x$

In this section, we talk about a more general case where $\mathbf{A}(t)$is a time-dependent matrix. First we have so usual rule like $\frac{d}{dt}\mathbf{A}(t)\mathbf{B}(t) = \mathbf{A}'(t)\mathbf{B}(t) + \mathbf{A}(t)\mathbf{B}'(t)$ and if $\det(\mathbf{A}) \neq 0, \frac{d}{dt}\mathbf{A}(t)^{-1} = -\mathbf{A}(t)^{-1}\mathbf{A}'(t)\mathbf{A}(t)^{-1}$ The following theorem claim the existence and uniqueness of the solution to the IVP $\mathbf{x}'(t) = \mathbf{A}(t)\mathbf{x}(t)$:

Theorem 8 *Suppose $\mathbf{A}(t)$ is continuous on an open interval I which contains the point t_0. Then the linear first-order ODE system $\mathbf{x}'(t) = \mathbf{A}(t)\mathbf{x}(t)$ has a unique solution satisfying the initial condition $\mathbf{x}(t_0) = \mathbf{x_0}$. Moreover, this solution is defined for all t $\in I$.*

Using the superposition principle, we know that the set of all solutions forms a vector space. So if we could find all the solutions which are the bases of this vector space, then all the solution can be expressed as the linear combination of the bases. A matrix $\prod(t, t_0)$ is called a **principle matrix solution** at t_0 and it solves the matrix valued initial value problem:

$$\begin{cases} \prod'(t, t_0) = \mathbf{A}(t)\prod(t, t_0) \\ \prod(t_0, t_0) = I \end{cases} \tag{3.34}$$

The existence and uniqueness of the solution to IVP ensure the existence and uniqueness of the principle matrix solution. So we can easily see that $\prod(t, t_0)\mathbf{x}_0$ is exactly the solution to the IVP:

$$\begin{cases} \mathbf{x}'(t) = \mathbf{A}(t)\mathbf{x}(t) \\ \mathbf{x}(t_0) = \mathbf{x}_0 \end{cases} \tag{3.35}$$

So the main problem is to find$\prod(t, t_0)$. In particular, the principal matrix solution can be obtained from any fundamental matrix solution via $\prod(t, t_0) = \Phi(t)\Phi^{-1}(t_0)$ A simple example is illustrated below: consider the linear first-order ODE system: $x' = \begin{pmatrix} 2 & t \\ 0 & 3 \end{pmatrix} x$ .which can be detailed as:

$$\begin{cases} {x_1}' = 2x_1 + tx_2 \\ {x_2}' = 3x_2 \end{cases} \tag{3.36}$$

We must solve for two IVP: $x = (1, 0)'$ and $x = (0, 1)'$. For initial condition$x = (1, 0)'$, we know that $x_2 = 0$ and therefore$x_1 = e^{2(t-t_0)}$. As for the initial condition$x = (0, 1)'$, we get $x_2 = e^{3(t-t_0)}$. Plug the solution into the first equation ${x_1}' = 2x_1 + tx_2$ and we get $x_1 = (1 - t_o)e^{2(t-t_o)} + (t - 1)e^{3(t-t_o)}$ using the initial condition $x_1(t_0) = 0$. Hence we get:

$$\prod(t, t_0) = \begin{pmatrix} e^{2(t-t_0)} & (1 - t_o)e^{2(t-t_o)} + (t - 1)e^{3(t-t_o)} \\ 0 & e^{3(t-t_0)} \end{pmatrix} \tag{3.37}$$

3.2.4 nonhomogeneous linear system $x' = A(t)x + B(t)$

We have discussed the solution to homogeneous linear system. To gain a particular solution to a nonhomogeneous system

$$\mathbf{x}'(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)$$

we use the method of variation of constants (also variation of parameters) by assuming the particular solution to the system is:

$$\mathbf{x}(t) = \prod(t, t_0)\mathbf{c}(t), \mathbf{c}(t_0) = \mathbf{x_0} \tag{3.38}$$

Differentiate both sides we get

$$\begin{aligned} \mathbf{x}'(t) &= \prod'(t,t_0)\mathbf{c}(t) + \prod(t,t_0)\mathbf{c}'(t) \\ &= \mathbf{A}(t)\prod(t,t_0)\mathbf{c}(t) + \prod(t,t_0)\mathbf{c}'(t) \\ &= \mathbf{A}(t)\mathbf{x}(t) + \prod(t,t_0)\mathbf{c}'(t) \end{aligned} \tag{3.39}$$

And the comparison with $\mathbf{x}'(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)$ yields

$$\prod(t,t_0)\mathbf{c}'(t) = \mathbf{B}(t) \tag{3.40}$$

Note that $\prod(t,t_0)^{-1} = \prod(t_0,t)$since$\prod(t,t_1)\prod(t_1,t_0) = \prod(t,t_0)$, and let $t = t_0$we can get $\prod(t_0,t_1)\prod(t_1,t_0) = \prod(t_0,t_0) = I$ for $\forall t_1 \in I$, which means $\prod(t_0,t)\prod(t,t_0) = I$. Then $\prod(t,t_0)\mathbf{c}'(t) = \mathbf{B}(t)$ is equivalent to

$$\mathbf{c}'(t) = \prod(t_0,t)\mathbf{B}(t) \tag{3.41}$$

Integral both sides yields

$$\mathbf{c}(t) = \mathbf{x}_0 + \int_{t_0}^{t} \prod(t_0,s)\mathbf{B}(s)ds \tag{3.42}$$

Then the general solution to nonhomogeneous system $\mathbf{x}'(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)$is given by:

$$\begin{aligned} \mathbf{x}(t) &= \prod(t,t_0)\mathbf{x}_0 + \prod(t,t_0)\int_{t_0}^{t} \prod(t_0,s)\mathbf{B}(s)ds \\ &= \prod(t,t_0)\mathbf{x}_0 + \int_{t_0}^{t} \prod(t,s)\mathbf{B}(s)ds \end{aligned} \tag{3.43}$$

4 Numerical Solution for PDE

4.1 Runge-Kutta method

From the previous discussion about, we explore the method solving different kinds of first-order ODE system. However, to solve it in computer is another story-we need numerical method. Recall that we once use Runge-Kutta method to solve the ODE equation:

$$\frac{dy}{dx} = f(x,y) \tag{4.44}$$

Now we reuse this method, but in a way that the variable is represented as matrix. The Runge-Kutta method is used to simplify Taylor series. It can be seen that Runge-Kutta method is just another form of Taylor series method while there is no need to calculate the derivatives. Instead, iteration is used. The general form of calculating $y(x+h)$ using this method is provided as follows:

$$y(x_0+h) = y(x_0) + \frac{1}{6}(\omega_1 + 2\omega_2 + 2\omega_3 + \omega_4) \tag{4.45}$$

Where:

$$\begin{aligned} \omega_1 &= hf(x_0,y_0) \\ \omega_2 &= hf\left(x_0 + \tfrac{1}{2}h, y_0 + \tfrac{1}{2}\omega_1\right) \\ \omega_3 &= hf\left(x_0 + \tfrac{1}{2}h, y_0 + \tfrac{1}{2}\omega_2\right) \\ \omega_4 &= hf(x_0 + h, y_0 + \omega_3) \end{aligned} \tag{4.46}$$

This is equivalent to the Taylor polynomial with degree 4 so the error is just $O\left(h^5\right)$.

And we interpret this method into a <u>matrix</u> form:
Consider a initial value problem of first order ODE system:

$$\left\{ \begin{array}{l} \frac{dy_1}{dx} = f_1(y_1,y_2,\ldots,y_n,x) \\ \frac{dy_2}{dx} = f_2(y_1,y_2,\ldots,y_n,x) \\ \vdots \\ \frac{dy_n}{dx} = f_n(y_1,y_2,\ldots,y_n,x) \end{array} \right. \quad with \left\{ \begin{array}{l} y_1(x_0) = y_1^0 \\ y_2(x_0) = y_2^0 \\ \vdots \\ y_n(x_0) = y_n^0 \end{array} \right. \tag{4.47}$$

We rewrite this system into $\mathbf{Y}' = \mathbf{F}(x, \mathbf{Y})$ with $\mathbf{Y}(x_0) = \mathbf{Y_0}$. Then by the previous result we can guess the solution to $\mathbf{Y}(x_0 + h)$:

$$\mathbf{Y}\left(x_0 + h\right) = \mathbf{Y}\left(x_0\right) + \frac{1}{6}\left(\mathbf{\Omega_1} + 2\mathbf{\Omega_2} + 2\mathbf{\Omega_3} + \mathbf{\Omega_4}\right) \tag{4.48}$$

Where:

$$\begin{aligned}
\mathbf{\Omega_1} &= h\mathbf{F}\left(x_0, \mathbf{Y_0}\right)\\
\mathbf{\Omega_2} &= h\mathbf{F}\left(x_0 + \tfrac{1}{2}h, \mathbf{Y_0} + \tfrac{1}{2}\mathbf{\Omega_1}\right)\\
\mathbf{\Omega_3} &= h\mathbf{F}\left(x_0 + \tfrac{1}{2}h, \mathbf{Y_0} + \tfrac{1}{2}\mathbf{\Omega_2}\right)\\
\mathbf{\Omega_4} &= h\mathbf{F}\left(x_0 + h, \mathbf{Y_0} + \mathbf{\Omega_3}\right)
\end{aligned} \tag{4.49}$$

In fact, the $\mathbf{\Omega}$ can be detailed as:

$$\mathbf{\Omega_1} = \begin{pmatrix} hf_1(y_1^0, y_2^0, \dots, y_n^0, x_0) \\ hf_2(y_1^0, y_2^0, \dots, y_n^0, x_0) \\ \vdots \\ hf_n(y_1^0, y_2^0, \dots, y_n^0, x_0) \end{pmatrix} = \begin{pmatrix} f_{11} \\ f_{12} \\ \vdots \\ f_{1n} \end{pmatrix} \tag{4.50}$$

$$\mathbf{\Omega_2} = \begin{pmatrix} hf_1(y_1^0 + \frac{1}{2}f_{11}, y_2^0 + \frac{1}{2}f_{12}, \dots, y_n^0 + \frac{1}{2}f_{1n}, x_0 + \frac{1}{2}h) \\ hf_2(y_1^0 + \frac{1}{2}f_{11}, y_2^0 + \frac{1}{2}f_{12}, \dots, y_n^0 + \frac{1}{2}f_{1n}, x_0 + \frac{1}{2}h) \\ \vdots \\ hf_n(y_1^0 + \frac{1}{2}f_{11}, y_2^0 + \frac{1}{2}f_{12}, \dots, y_n^0 + \frac{1}{2}f_{1n}, x_0 + \frac{1}{2}h) \end{pmatrix} = \begin{pmatrix} f_{21} \\ f_{22} \\ \vdots \\ f_{2n} \end{pmatrix} \tag{4.51}$$

$$\mathbf{\Omega_3} = \begin{pmatrix} hf_1(y_1^0 + \frac{1}{2}f_{21}, y_2^0 + \frac{1}{2}f_{22}, \dots, y_n^0 + \frac{1}{2}f_{2n}, x_0 + \frac{1}{2}h) \\ hf_2(y_1^0 + \frac{1}{2}f_{21}, y_2^0 + \frac{1}{2}f_{22}, \dots, y_n^0 + \frac{1}{2}f_{2n}, x_0 + \frac{1}{2}h) \\ \vdots \\ hf_n(y_1^0 + \frac{1}{2}f_{21}, y_2^0 + \frac{1}{2}f_{22}, \dots, y_n^0 + \frac{1}{2}f_{2n}, x_0 + \frac{1}{2}h) \end{pmatrix} = \begin{pmatrix} f_{31} \\ f_{32} \\ \vdots \\ f_{3n} \end{pmatrix} \tag{4.52}$$

$$\mathbf{\Omega_4} = \begin{pmatrix} hf_1(y_1^0 + \frac{1}{2}f_{31}, y_2^0 + \frac{1}{2}f_{32}, \dots, y_n^0 + \frac{1}{2}f_{3n}, x_0 + \frac{1}{2}h) \\ hf_2(y_1^0 + \frac{1}{2}f_{31}, y_2^0 + \frac{1}{2}f_{32}, \dots, y_n^0 + \frac{1}{2}f_{3n}, x_0 + \frac{1}{2}h) \\ \vdots \\ hf_n(y_1^0 + \frac{1}{2}f_{31}, y_2^0 + \frac{1}{2}f_{32}, \dots, y_n^0 + \frac{1}{2}f_{3n}, x_0 + \frac{1}{2}h) \end{pmatrix} = \begin{pmatrix} f_{41} \\ f_{42} \\ \vdots \\ f_{4n} \end{pmatrix} \tag{4.53}$$

We can see that this method is a general approach to first order ODE system, however since its complexity, computer cannot handle such problem when the dimension of the matrix is bigger than 100 approximately (Using this Runge-Kutta method). In the following discussion, iterative method which can accomplished by MATLAB is demonstrated, which is feasible.

4.2 Elliptical Partial Differential Equation

4.2.1 Dirichlet Method for Laplace's Equation

To approximate the solution of the Laplace's equation $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$ over the rectangle $R = \{(x, y) : 0 \leqslant x \leqslant a, 0 \leqslant y \leqslant b\}$ with $u(x, 0) = g_1(x), u(x, b) = g_2(x), \text{for}\, 0 \leqslant x \leqslant a, \text{and}\, u(0, y) = g_3(y)\, \text{and}\, u(a, y) = g_4(y)$, for $0 \leqslant y \leqslant b$. It is assumed that $\Delta x = \Delta y = h$ and integers n and m exist so that $a = nh$and $b = mh$.

let us consider the large problem and fix our ideas regarding various do-loops that we will use in the program. The grid for this problem is defined by $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$. The east, west, north, and south walls are Dirichlet boundaries. The boundary conditions $\{\varphi\left(1, j\right); j = 2, m-1\}$, $\{\varphi\left(n, j\right); j = 2, m-1\}$, $\{\varphi\left(i, 1\right); i = 2, n-1\}$ and $\{\varphi\left(i, m\right); i = 2, n-1\}$ are inserted as data.

The algorithm for the iterative solution is [6]

Algorithm 1

```
For $i = 2, n-1$
For $j = 2, m-1$
  $\varphi\left(i, j\right) = \frac{1}{4}\left[\varphi\left(i+1, j\right) + \varphi\left(i-1, j\right) + \varphi\left(i, j-1\right) + \varphi\left(i, j+1\right)\right]$
End do
End do
```

We can easily see that the corner values $\varphi(1,1)$, $\varphi(n,1)$, $\varphi(1,m)$, $\varphi(n,m)$ do not appear in the iteration loops of Eqn.(14). These corner values are computed from

$$\begin{aligned}
\varphi(1,1) &= \frac{1}{2}\left[\varphi(1,2)+\varphi(2,1)\right]\\
\varphi(n,1) &= \frac{1}{2}\left[\varphi(n-1,1)+\varphi(n,2)\right]\\
\varphi(1,m) &= \frac{1}{2}\left[\varphi(1,m-1)+\varphi(2,m)\right]\\
\varphi(n,m) &= \frac{1}{2}\left[\varphi(n,m-1)+\varphi(n-1,m)\right]
\end{aligned}$$

It's a statement of continuity of φ as a corner is approached along the two walls meeting at the corner.

4.2.2 Neumann Method for Laplace's Equation

To approximate the solution of the Laplace's equation $\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0$ over the rectangle $R=\{(x,y):0\leqslant x\leqslant a, 0\leqslant y\leqslant b\}$ with $u(x,0)=g_1(x), u(x,b)=g_2(x),\text{for}\, 0\leqslant x\leqslant a,\text{and}\, u(0,y)=g_3(y)\,\text{and}\, u(a,y)=g_4(y)$, for $0\leqslant y\leqslant b$. It is assumed that $\Delta x=\Delta y=h$ and integers n and m exist so that $a=nh$and $b=mh$.

The simple algorithm of Neumann for the iterative solution is

Algorithm 2

```
For $i = 2, n-1$
For $j = 2, m-1$
If $i = 1$ then
  $\varphi(1,j)=\frac{1}{4}\left[2\varphi(2,j)+2hg(1,j)+\varphi(1,j+1)+\varphi(1,j-1)\right]$
Else
  $\varphi(i,j)=\frac{1}{4}\left[\varphi(i+1,j)+\varphi(i-1,j)+\varphi(i,j-1)+\varphi(i.j+1)\right]$
End do
End do
```

Where $\frac{\partial\varphi}{\partial x}|_{(1,j)}=\frac{\varphi(2,j)-\varphi(0,j)}{2h}=-g(1,j)$

4.3 Parabolic Partial Differential Equation

4.3.1 Forward-Difference Method for the Heat Equation

To approximate the solution of the heat equation $\frac{\partial u}{\partial t}=a\frac{\partial^2 u}{\partial x^2}$ over the rectangle $R=\{(x,t):0\leqslant x\leqslant a, 0\leqslant t\leqslant b\}$ with $u(x,0)=\varphi(x)$, for $0\leqslant x\leqslant a$. and $u(0,t)=a_1$ and $u(a,t)=a_2$, for $0\leqslant t\leqslant b$.

Algorithm 3

```
For $i = 2, n$
For $j = 2, m$
$u(i,j)=(1-2r)\,u(i,j-1)+r\left(u(i-1,j-1)+u(i+1,j-1)\right)$
End do
End do
```

where $r$satisfies $h^2 r=ak$, h is the difference of x,and k is the difference of t.

4.3.2 Crank-Nicolson Method

An implicit scheme, invented by John Crank(1916-) and Phyllis Nicolson(1917-1968),is based on numerical approximations for solutions at the point $(x,t+k/2)$ that lies between the rows in the grid. Specifically, the approximation used for $u_t(x,t+k/2)$ is obtained from the central-difference formula,

$$u_t\left(x,t+\frac{k}{2}\right)=\frac{u(x,t+k)-u(x,t)}{k}+O\left(k^2\right)$$

The equation for Crank–Nicolson method is a combination of the forward Euler method at n and the backward Euler method at $n+1$ (note, however, that the method itself is not simply the average of those two methods, as the equation has an implicit dependence on the solution),which can make it as a tridiagonal problem, so that it may be efficiently solved by using the tridiagonal matrix algorithm in favor of a much more costly matrix inversion.

4.4 Hyperbolic Partial Differential Equation

4.4.1 Forward-Difference Method for the Wave Equation

To approximate the solution of the heat equation $\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$ over the rectangle $R = \{(x,t) : 0 \leqslant x \leqslant a, 0 \leqslant t \leqslant b\}$ with $u(x,0) = \varphi(x), \frac{\partial u}{\partial t}|_{t=0} = \phi(x)$, for $0 \leqslant x \leqslant a$. and $u(0,t) = g_1(t), u(a,t) = g_2(t)$, for $0 \leqslant t \leqslant b$.

Algorithm 4

```
    For $i = 2, n$
    If $j = 2$
    $u(i,2) = (1 - r^2)\varphi(i) + k\phi(i) + \frac{r^2}{2}(\varphi(i+1) + \varphi(i-1))$
    For $j = 3, m$
    $u(i,j) = (2 - 2r^2)u(i,j-1) + r^2(u(i+1,j-1) + u(i-1,j-1)) - u(i,j-2)$
    End do
    End do
```

5 The MATLAB Application of Partial Differential Equation

The numerical method we use here is called MOL(Method of lines). And here we shall give different code aiming at different types of PDE.

5.1 Elliptic PDE

As an example, we will deal with a special type of elliptic equation called Helmholtz' s equation, which is written as:

$$\nabla^2 u(x,y) + g(x,y)u(x,y) = f(x,y) \tag{5.54}$$

over a domain $\mathrm{D} = \{(\mathrm{x},\mathrm{y})|\mathrm{x}_0 \leq x \leq x_f, y_0 \leq y \leq y_f\}$ with some boundary conditions of

$$\begin{aligned} u(x_0,y) = b_{x_0}(y),\ u(x_f,y) = b_{x_f}(y), \\ u(x,y_0) = b_{y_0}(x),\ u(x,y_f) = b_{y_f}(x) \end{aligned} \tag{5.55}$$

Because we have already discussed the three-point central difference approximation above, so we would like to pass it here and we will discuss its application.
To apply the difference method, we divide the domain into M_x sections, each of length $\Delta x = (x_f - x_0)/M_x$ along the x-axis and into M_y sections, each of length $\Delta y = (y_f - y_0)/M_y$ along the y-axis, respectively, and then by applying three-point central difference approximation, it is easy to get

$$\begin{aligned} \left.\frac{\partial^2 u(x,y)}{\partial x^2}\right|_{x_j,y_j} &\cong \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta x^2} with\ x_j = x_0 + j\Delta x, y_j = y_0 + i\Delta y \\ \left.\frac{\partial^2 u(x,y)}{\partial y^2}\right|_{x_j,y_j} &\cong \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta y^2} with\ u_{i,j} = u(x_j,y_i) \end{aligned} \tag{5.56}$$

so that, for every interior point (x_i, y_i) with $1 \leq i \leq M_y - 1$ and $1 \leq j \leq M_x - 1$, we obtain the finite difference equation

$$\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta x^2} + \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta y^2} + g_{i,j}u_{i,j} = f_{i,j} \tag{5.57}$$

where

$$u_{i,j} = u(x_j,y_i),\ f_{i,j} = f(x_j,y_i)\ and\ g_{i,j} = g(x_j,y_i) \tag{5.58}$$

These equation can somehow be arranged into a linear system of simultaneous equations with respect to $(M_y - 1)(M_x - 1)$ variables, and here if we use Runge-Kutta method to solve the matrix, MATLAB would be in trouble as M_x and M_y become larger. A simpler way is to use the iterative methods. To do so, we first need to shape the equations and the boundary conditions into the following form:

$$u_{i,j} = r_y(u_{i,j+1} + u_{i,j-1}) + r_x(u_{i+1,j} + u_{i-1,j}) + r_{xy}(g_{i,j}u_{i,j} - f_{i,j}) \tag{5.59}$$

with

$$u_{i,0} = b_{x_0}(y_i),\ u_{i,M_x} = b_{x_f}(y_i),\ u_{0,j} = b_{y_0}(x_j),\ u_{M_y,j} = b_{y_f}(x_j) \tag{5.60}$$

where

$$r_y = \frac{\Delta y^2}{2(\Delta x^2 + \Delta y^2)},\ r_x = \frac{\Delta x^2}{2(\Delta x^2 + \Delta y^2)},\ r_{xy} = \frac{\Delta x^2 \Delta y^2}{2(\Delta x^2 + \Delta y^2)} \tag{5.61}$$

How do we initialize this algorithm? If we have no priori knowledge about the solution, it is reasonable to take the average value of the boundary values as the initial values of $u_{i,j}$.
Here, the code is showed in `Helmhotz.m`:

```
function [u,x,y]=Helmhotz(f,g,bx0,bxf,by0,byf,D,Mx,My,tol,MaxIter)
%solve u_xx + u_yy + g(x,y)u = f(x,y)
% over the region D = [x0,xf,y0,yf] = {(x,y) |x0 <= x <= xf, y0 <= y <= yf}
% with the boundary Conditions:
% u(x0,y) = bx0(y), u(xf,y) = bxf(y)
% u(x,y0) = by0(x), u(x,yf) = byf(x)
% Mx = # of subintervals along x axis
% My = # of subintervals along y axis
% tol : error tolerance
% MaxIter: the maximum # of iterations
x0 = D(1); xf = D(2); y0 = D(3); yf = D(4);
dx = (xf - x0)/Mx; x = x0 + [0:Mx]*dx;
dy = (yf - y0)/My; y = y0 + [0:My]*dy;
Mx1 = Mx + 1; My1 = My + 1;
%Boundary conditions
for m = 1:My1, u(m,[1 Mx1])=[bx0(y(m)) bxf(y(m))]; end %left/right side
for n = 1:Mx1, u([1 My1],n) = [by0(x(n)); byf(x(n))]; end %bottom/top
%initialize as the average of boundary values
sum_of_bv = sum(sum([u(2:My,[1 Mx1]) u([1 My1],2:Mx)']));
u(2:My,2:Mx) = sum_of_bv/(2*(Mx + My - 2));
for i = 1:My
for j = 1:Mx
F(i,j) = f(x(j),y(i)); G(i,j) = g(x(j),y(i));
end
end
dx2 = dx*dx; dy2 = dy*dy; dxy2 = 2*(dx2 + dy2);
rx = dx2/dxy2; ry = dy2/dxy2; rxy = rx*dy2;
for itr = 1:MaxIter
for j = 2:Mx
for i = 2:My
u(i,j) = ry*(u(i,j + 1)+u(i,j - 1)) + rx*(u(i + 1,j)+u(i - 1,j))...
+ rxy*(G(i,j)*u(i,j)- F(i,j)); %Eq.(9.1.5a)
end
end
if itr > 1 & max(max(abs(u - u0))) < tol, break; end
u0 = u;
end
```

And we just need to give different boundary condition to get different results restricted under the error we give.
As an example, here:
Example 1 Poisson's Equation

Consider Poisson's equation

$$\begin{aligned}&\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0,\ 0<x,y<\pi\\ &u(x,0)=\sin(x),\ u(x,\pi)=0,\ 0\le x\le\pi\\ &u(0,y)=\sin(y),\ u(\pi,y)=0,\ 0\le y\le\pi\end{aligned} \tag{5.62}$$

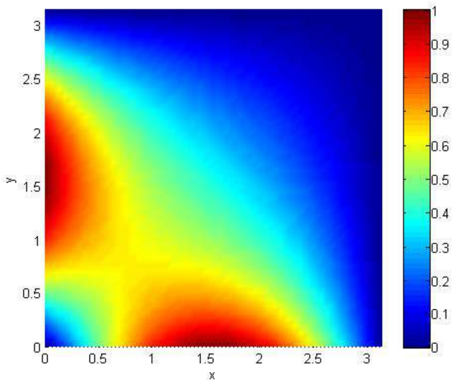
The real solution of this is:

$$u(x,y)=\frac{1}{\sinh(\pi)}[\sin(x)\sinh(\pi-y)+\sin(y)\sinh(\pi-x)] \tag{5.63}$$

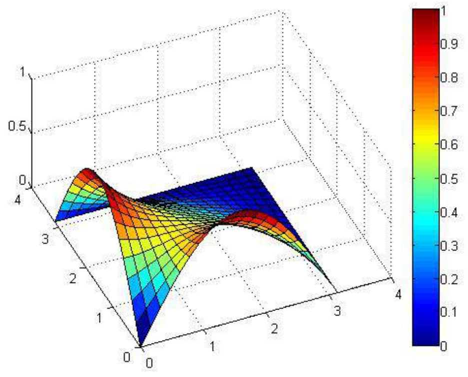
And the code is:

```
f = inline('0','x','y'); g = inline('0','x','y');
x0 = 0; xf = pi; Mx = 50; y0 = 0; yf = pi; My = 50;
bx0 = inline('sin(y)','y'); %(E9.1.2a)
bxf = inline('0','y'); %(E9.1.2b)
by0 = inline('sin(x)','x'); %(E9.1.3a)
byf = inline('0','x'); %(E9.1.3b)
D = [x0 xf y0 yf]; MaxIter = 1000; tol = 1e-4;
[U,x1,y1] = Helmhotz(f,g,bx0,bxf,by0,byf,D,Mx,My,tol,MaxIter);
real=inline('1/sinh(pi)*(sin(x)*sinh(pi-y)+sin(y)*sinh(pi-x))','x','y');
error=0;
for i=1:(Mx+1)
    for j=1:(My+1)
        error=error+(real(x1(i),y1(j))-U(i,j))^2;
    end
end
error
[X1,Y1]=meshgrid(x1,y1);
Z1=griddata(x1,y1,U,X1,Y1,'v4');
figure,surf(X1,Y1,Z1);
shading interp;
xlabel('x');ylabel('y');
colormap('jet');
colorbar;
```

Here, we can see that in order to control the accuracy level of the result, we just need to change the value of tol, and we shall see later from the numerical result that the SSE(sum square of error) of the numerical solution is positive correlated with the value of *tol*.



(a) The plan view of the temperature distribution



(b) The 3-dimensional view of the temperature distribution

Figure 5.2: The numerical result of the Poisson Equation

Thus, it is easy to observe the positive correlation between tol and SSE, also perror and err-order. Also, from the data, we may give an assumption that when M_x and M_y become larger, the increasing velocity of its accuracy level would become lower.

Table 5.1: The Positive Correlation between tol and SSE

	M_x	M_y	tol	SSE	perror	err-order
Case 1	50	50	10^{-4}	0.1487	0.0172	-1.8476
Case 2	50	50	10^{-5}	0.0033	0.0023	-4.7496
Case 3	50	50	10^{-6}	0.0011	0.0014	-5.5097
Case 4	20	20	10^{-4}	1.0859×10^{-4}	0.0014	-3.0674
Case 5	20	20	10^{-5}	9.3671×10^{-5}	9.1363×10^{-4}	-0.5107

5.2 Parabolic PDE

As an example, we will deal with a special type of parabolic equation called one-dimensional heat equation, which is written as:

$$A\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t} \quad for\ 0 \le x \le x_f,\ 0 \le t \le T \tag{5.64}$$

with some boundary conditions(Dirichlet type) of $u(0,t) = b_0(t),\ u(x_f,t) = b_{x_f}(t)$, as well as the initial condition $u(x,0) = i_0(x)$. And we will solve this equation using three methods which we mentioned above:
1. The Explicit Forward Euler Method
Here, we still use the iterative method:
first, define the function:

```
function [u,x,t] = heat_exp(a,xf,T,it0,bx0,bxf,M,N)
%solve a u_xx = u_t for 0 <= x <= xf, 0 <= t <= T
% Initial Condition: u(x,0) = it0(x)
% Boundary Condition: u(0,t) = bx0(t), u(xf,t) = bxf(t)
% M = # of subintervals along x axis
% N = # of subintervals along t axis
```

second, we need to assign the value to dx and dt:

```
dx = xf/M; x = [0:M]'*dx;
dt = T/N; t = [0:N]*dt;
```

third, using the iterative method to do the numerical calculation:

```
for i = 1:M + 1, u(i,1) = it0(x(i)); end
for n = 1:N + 1, u([1 M + 1],n) = [bx0(t(n)); bxf(t(n))]; end
r = a*dt/dx/dx, r1 = 1 - 2*r;
for k = 1:N
for i = 2:M
u(i,k+1) = r*(u(i + 1,k) + u(i-1,k)) + r1*u(i,k); %Eq.(9.2.3)
end
end
```

The MATLAB routine `heat_exp.m` has been composed to implement this algorithm.
2. The Implicit Backward Euler Method
Similarly, first, define the function:

```
function [u,x,t] = heat_imp(a,xf,T,it0,bx0,bxf,M,N)
%solve a u_xx = u_t for 0 <= x <= xf, 0 <= t <= T
% Initial Condition: u(x,0) = it0(x)
% Boundary Condition: u(0,t) = bx0(t), u(xf,t) = bxf(t)
% M = # of subintervals along x axis
% N = # of subintervals along t axis
```

second, we need to assign the value to dx and dt:

```
dx = xf/M; x = [0:M]'*dx;
dt = T/N; t = [0:N]*dt;
```

third, we need to do the matrix calculation, be careful that here we cannot use the iterative method we used before because the boundary condition is different.

```
for i = 1:M + 1, u(i,1) = it0(x(i)); end
for n = 1:N + 1, u([1 M + 1],n) = [bx0(t(n)); bxf(t(n))]; end
r = a*dt/dx/dx; r2 = 1 + 2*r;
for i = 1:M - 1
A(i,i) = r2;
if i > 1, A(i - 1,i) = -r; A(i,i - 1) = -r; end
end
for k = 2:N + 1
b = [r*u(1,k); zeros(M - 3,1); r*u(M + 1,k)] + u(2:M,k - 1); %Eq.(9.2.9)
u(2:M,k) = trid(A,b);
end
```

The MATLAB routine `heat_imp.m` has been composed to implement this algorithm.

3. The Crank Nicholson Method

Similarly, first, define the function:

```
function [u,x,t] = heat_CN(a,xf,T,it0,bx0,bxf,M,N)
%solve a u_xx = u_t for 0 <= x <= xf, 0 <= t <= T
% Initial Condition: u(x,0) = it0(x)
% Boundary Condition: u(0,t) = bx0(t), u(xf,t) = bxf(t)
% M = # of subintervals along x axis
% N = # of subintervals along t axis
```

second, we need to assign the value to dx and dt:

```
dx = xf/M; x = [0:M]'*dx;
dt = T/N; t = [0:N]*dt;
```

third, we need to do the matrix calculation, like the implicit Euler method, here we cannot use the iterative method.

```
for i = 1:M + 1, u(i,1) = it0(x(i)); end
for n = 1:N + 1, u([1 M + 1],n) = [bx0(t(n)); bxf(t(n))]; end
r = a*dt/dx/dx;
r1 = 2*(1 - r); r2 = 2*(1 + r);
for i = 1:M - 1
A(i,i) = r1;
if i > 1, A(i - 1,i) = -r; A(i,i - 1) = -r; end
end
for k = 2:N + 1
b = [r*u(1,k); zeros(M - 3,1); r*u(M + 1,k)] ...
+ r*(u(1:M - 1,k - 1) + u(3:M + 1,k - 1)) + r2*u(2:M,k - 1);
u(2:M,k) = trid(A,b); %Eq.(9.2.17)
end
```

The MATLAB routine `heat_CN.m` has been composed to implement this algorithm.

And we will use an example to show you a comparison of these three methods.

Example2 One-Dimensional Parabolic PDE: Heat Flow Equation.

Consider the Parabolic PDE:

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial^2 u(x,t)}{\partial t^2} \quad for\ 0 \le x \le 1,\ 0 \le t \le 0.2 \tag{5.65}$$

with the initial condition and the boundary conditions:

$$u(x,0) = \sin \pi x,\ u(0,t) = 0,\ u(1,t) = 0 \tag{5.66}$$

The real solution of this equation is:

$$u(x,t) = e^{-\pi^2 t} \sin(\pi x) \tag{5.67}$$

Note that $\Delta x = 1/M$ and $\Delta t = 0.2/N$, thus the stability condition become:

$$\gamma = \frac{0.2M^2}{N} \le \frac{1}{2} \tag{5.68}$$

Choose $M = 3, 6, 12$ and corresponding $N = 6, 24, 96$, so in this case $\gamma = 0.3$, which satisfies the stability condition. And the code of the example is:

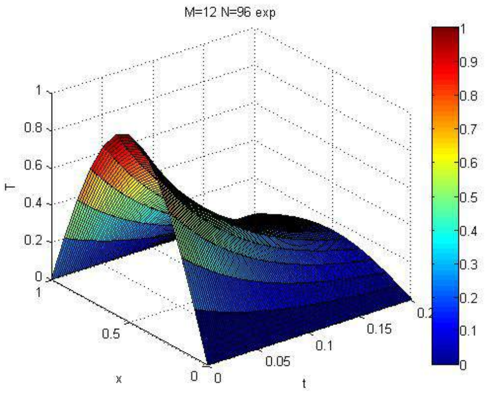
```
%solve_heat
a = 1; %the parameter of (E9.2.1)
it0 = inline('sin(pi*x)','x'); %initial condition
bx0 = inline('0'); bxf = inline('0'); %boundary condition
xf = 1; M = 12; T = 0.2; N = 96; %r = 0.3
%analytical solution
uo = inline('sin(pi*x)*exp(-pi*pi*t)','x','t');
[u1,x,t] = heat_exp(a,xf,T,it0,bx0,bxf,M,N);
figure(1)
surf(t,x,u1);
colormap('jet');
view(3);
colorbar;
title('M=12 N=96 exp');
xlabel('t');ylabel('x');zlabel('T');
[u2,x,t] = heat_imp(a,xf,T,it0,bx0,bxf,M,N); %converge unconditionally
figure(2)
surf(t,x,u2);
colormap('jet');
view(3);
colorbar;
title('M=12 N=96 imp');
xlabel('t');ylabel('x');zlabel('T');
[u3,x,t] = heat_CN(a,xf,T,it0,bx0,bxf,M,N); %converge unconditionally
figure(3)
surf(t,x,u3);
colormap('jet');
view(3);
colorbar;
title('M=12 N=96 CN');
xlabel('t');ylabel('x');zlabel('T');
Uo = uo(x,t);
```

Calculate the error:

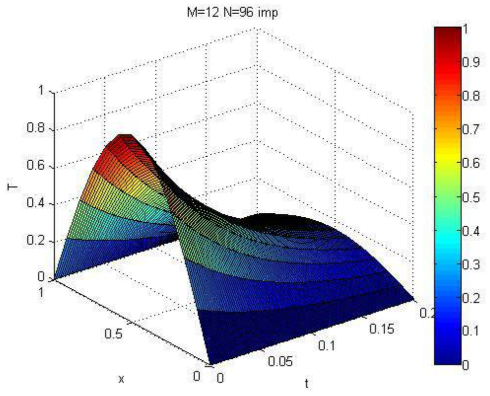
```
error=0;
perror=0;
for i=1:(M+1)
    for j=1:(N+1)
        error=error+(Uo(i,j)-u1(i,j))^2;
        perror(i,j)=abs(Uo(i,j)-u1(i,j));
    end
end
error
k1=max(max(perror))
error=0;
perror=0;
for i=1:(M+1)
    for j=1:(N+1)
        error=error+(Uo(i,j)-u2(i,j))^2;
        perror(i,j)=abs(Uo(i,j)-u2(i,j));
    end
end
error
k1=max(max(perror))
error=0;
perror=0;
for i=1:(M+1)
    for j=1:(N+1)
```

```
        error=error+(Uo(i,j)-u3(i,j))^2;
        perror(i,j)=abs(Uo(i,j)-u3(i,j));
    end
end
error
k1=max(max(perror))
```

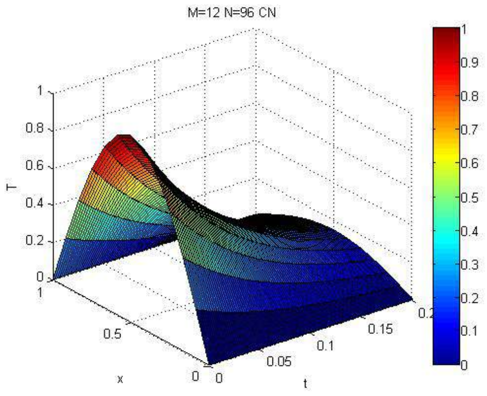
And the result is:



(a) Explicit Euler



(b) Implicit Euler



(c) Crank – Nicholson Method

Figure 5.3: Numerical result of the three methods

The comparison of different methods: The comparison of different M and N: So, from the two tables,

Table 5.2: The comparison of these three methods with M=12, N=96

	Explicit Euler	Implicit Euler	CN Method
Point Error	0.0017	0.0058	0.0021
SSE	0.0012	0.0039	0.0018

Table 5.3: The comparison of different step length(CN Method)

	M=3, N=6	M=6, N=24	M=12, N=96
Point Error	0.0269	0.0082	0.0021
SSE	0.0066	0.0035	0.0018

we may give out the assumption that:
1. CN method is a quite good method(at least better than implicit Euler method), and this is theoretically correct.
2. Smaller step length leads to higher accuracy level, which is quite reasonable because we can utilize more

information of the PDE.
Also, if we let the γ become very close to 0.5 or even larger than it, we would see that the error would become very large for explicit euler method:

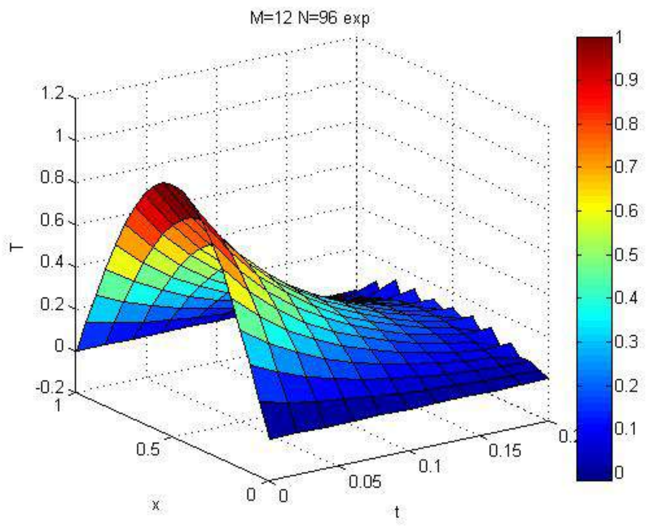


Figure 5.4: The result of the explicit euler

which obviously show the large error when t become very close to 0.2.
This tells us that stability condition is very important to explicit euler method, however, the other two methods still working efficiently when it breaks up. Besides, if it converges, its accuracy may be better than that of the implicit backward Euler method, but generally no better than that of the Crank – Nicholson method.

6 Contribution

Luo Tao: Backgrounds
Wenchao Zhang: Definite Problem of Partial Differential Equation, numerical solution of iteration for PDE
Qikun Wu: Runge-Kutta method, matrix manipulation, solution to ODE system, typesetting with $\mathcal{AMS}$-LaTeX
Jiale Wang: MatLab implementation of Elliptic and Parabolic PDE

References

[1] Gerald Teschl *Ordinary Differential Equations and Dynamical Systems.* American Mathematical Society.

[2] 贺小明,彭名书 常微分方程与动力系统概论 北京理工大学出版社,2010.

[3] 张贤科,许甫华 高等代数学 清华大学出版社,2008.

[4] 陆金甫,关治 偏微分方程数值解法 清华大学出版社,2004.

[5] Peaceman, D.W. and Rachford Jr, H.H. *The numerical solution of parabolic and elliptic differential equations* Journal of the Society for Industrial & Applied Mathematics, vol 3, 28-41, 1955, SIAM.

[6] Ambar K. Mitra *Finite Difference Method for the Solution of Laplace Equation* Department of Aerospace Engineering, Iowa State University,2008.