Chapter 10

Introduction to Initial Boundary Value Problems

We now want to combine our knowledge from solving boundary value problems (BVPs) and initial value problems (IVPs) to solve an initial boundary value problem (IBVP). These equations are PDEs by definition because the differential equation is a function of time and at least one spatial coordinate. Both parabolic and hyperbolic problems fit into this framework. We will see that solving linear parabolic equations is a straightforward extension of what we have learned but hyperbolic problems present some significant difficulties. We will also see how to solve nonlinear problems which we encountered for IVPs but not for BVPs.

10.1 Prototype Parabolic Problems

The prototype problem for a parabolic IBVP is called the **heat** or **diffusion** equation. In one spatial dimension it is governed by the problem

$$u_t(x,t) - \nu u_{xx}(x,t) = f(x,t) \quad a < x < b, \quad 0 < t \leq T \tag{10.1}$$

along with an initial condition $u(x,0) = u_0(x)$ and boundary conditions specified at $x = a$ and $x = b$. Here ν is the diffusion coefficient. Note that for simplicity of exposition we have chosen the initial time to be zero. As we have seen before, these boundary conditions can be Dirichlet, Neumann or a combination of the two. The boundary conditions we specify have to be compatible with the initial conditions. For example, if we impose $u(a,t) = \alpha(t)$ and $u(b,t) = \beta(t)$ then we must have that $u_0(a) = \alpha(0)$ and $u_0(b) = \beta(0)$. Otherwise the IBVP is not well-posed.

In two or three spatial dimensions the differential equation is replaced by

$$u_t(\mathbf{x},t) - \nu \Delta u(\mathbf{x},t) = f(\mathbf{x},t) \quad \mathbf{x} \in \Omega, \quad 0 < t < T \tag{10.2}$$

along with an initial condition $u(\mathbf{x},t_0) = u_0(\mathbf{x})$ and boundary conditions specified on Γ, the entire boundary of the domain Ω.

The heat/diffusion equation is important in modeling various phenomena. For example, $u(\mathbf{x}, t)$ can represent the temperature in a region Ω; if Dirichlet boundary conditions are imposed then the temperature is set on the boundaries and the initial temperature distribution is given. The function $f(\mathbf{x}, t)$ models a heat source or sink. The equation then gives the time evolution of the temperature distribution in the region Ω. Oftentimes the temperature reaches what is called a *steady state solution*, i.e., one in which the solution is no longer varying in time. In fact, this equation can model processes exhibiting diffusive-like behavior for a wide range of applications.

The heat equation satisfies a **maximum principle**. If there are no heat sources or sinks, i.e., $f = 0$, then the maximum temperature for all time has to occur either initially or on the boundary. One can demonstrate a corresponding minimum principle by considering a maximum principle for $-u(\mathbf{x}, t)$. As an example when $f \neq 0$, suppose we specify the temperature on the boundary to be zero and set an initial condition $u_0(\mathbf{x})$ with a temperature sink $f(\mathbf{x}, t) \leq 0$. Then the maximum principle states that the temperature in the domain Ω satisfies

$$\max_{(\mathbf{x},t)\in\bar{\Omega}\times[0,T]} u(\mathbf{x}, t) \leq \max\left[0, \max_{\vec{x}\in\bar{\Omega}} u_0(\mathbf{x})\right]$$

which implies that the highest temperature in the interior of the domain can't be larger than a temperature determined by whichever is higher, the value on the boundary (zero in this case) or the highest initial temperature. Clearly we would like our numerical approximations to satisfy this maximum principle. Note that if it does satisfy such a condition then it tells us that the solution is bounded above by either the boundary or initial condition.

A second example that we will consider is a system of parabolic equations. So far we have only considered the situation where we have a single unknown. A simple example of a linear parabolic system is to find $u(\mathbf{x}, t)$ and $v(\mathbf{x}, t)$ for $\mathbf{x} \in \Omega$, $0 < t \leq T$ satisfying

$$\begin{array}{rcll} u_t - \nu_1 \Delta u + v & = & f(\mathbf{x}, t) & \mathbf{x} \in \Omega, 0 < t \leq T \\ v_t - \nu_2 \Delta v + u & = & g(\mathbf{x}, t) & \mathbf{x} \in \Omega, 0 < t \leq T \end{array} \tag{10.3}$$

along with initial conditions $u(\mathbf{x}, 00) = u_0(\mathbf{x})$, $v(\mathbf{x}, 0) = v_0(\mathbf{x})$ and boundary conditions. Note that this system is coupled so we have to solve for both u and v simultaneously. We do not have to impose the same boundary conditions on both unknowns. Many mathematical models involve coupled system of PDEs so it is important to see how to solve systems. We will see that in the FD or FE approach straightforward modifications to our algorithms allow us to solve coupled systems. However, when we have twice as many unknowns the resulting matrix is doubled in size.

A third example that we will consider is a nonlinear equation where we add a nonlinear function $g(u)$ to the heat equation. Specifically we consider the differential equation

$$u_t(x, t) - \nu u_{xx}(x, t) + g(u) = f(x, t) \quad a < x < b, \quad 0 < t \leq T$$

For example, we could take the nonlinear term $g(u)$ to be u^k for $k > 1$. We will see that when we discretize this problem we are lead to a system of nonlinear algebraic equations so we need to see how to solve these nonlinear equations. Typically an iteration is required to solve the resulting nonlinear systems so at each time step we will need to add an interior loop for the iterative method.

10.2 Prototype Hyperbolic Problems

The prototype second order hyperbolic equation is the **wave equation** which models wave phenomena and so it is useful in acoustics, fluid dynamics, electromagnetics, etc. Specifically we seek $u(\mathbf{x}, t)$ satisfying

$$u_{tt}(\mathbf{x}, t) - \nu \Delta u(\mathbf{x}, t) = f(\mathbf{x}, t) \quad 0 < t \leq T, \mathbf{x} \in \Omega \tag{10.4}$$

along with the initial conditions

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}) \quad \text{and} \quad u_t(\mathbf{x}, 0) = v_0(\mathbf{x})$$

and boundary conditions specified on all of the boundary of Ω. Notice in this case we need two initial conditions because the equation contains a second derivative in time.

We will concentrate our efforts for hyperbolic equations on the one-dimensional linear advection equation which is closely related to the wave equation and demonstrates some of the issues that arise in solving hyperbolic equations. We seek $u(x, t)$ satisfying

$$u_t(x, t) + c u_x(x, t) = 0 \tag{10.5}$$

Advection is a transport mechanism; for example we say that pollutants in a river are advected by the water flow. This equation models the passive advection of a field $u(x, t)$ carried along by a flow of constant speed c.

A second example we will consider is Burgers' equation which is a fundamental nonlinear PDE that is used in modeling dynamics such as gas flow or even the flow of traffic. Specifically we seek $u(x, t)$ satisfying

$$u_t(x, t) - \nu u_{xx}(x, t) + u(x, t) u_x(x, t) = f(x, t) \quad a < x < b, \quad t_0 < t \leq T \tag{10.6}$$

along with an initial condition $u(x, 0) = u_0(x)$ and boundary conditions specified at $u = a$ and $u = b$. As before $\nu > 0$ in order that the problem is well-posed for forward time. The term u_{xx} is the diffusion term we saw in the heat equation and the additional term uu_x models nonlinear transport.

We will only look at finite difference methods for solving hyperbolic equations. Finite difference or finite volume methods are commonly used for hyperbolic equations; finite element methods are used less often.

10.3 Discretization

When we discretize IBVPs we discretize the spatial domain as we did when we solve BVPs but now we also have to include a time step to evolve the

solution in time. For example, if we want to solve the heat equation in one spatial dimension then we discretize the domain by introducing grid points $a = x_0, x_1, \ldots, x_{n+1} = b$ but we also must include a time domain $[0, T]$ (where $t_0 = 0$) as indicated in Figure 10.1; the locations where the initial conditions are imposed are indicated by a double circle and the boundary conditions by a solid circle.

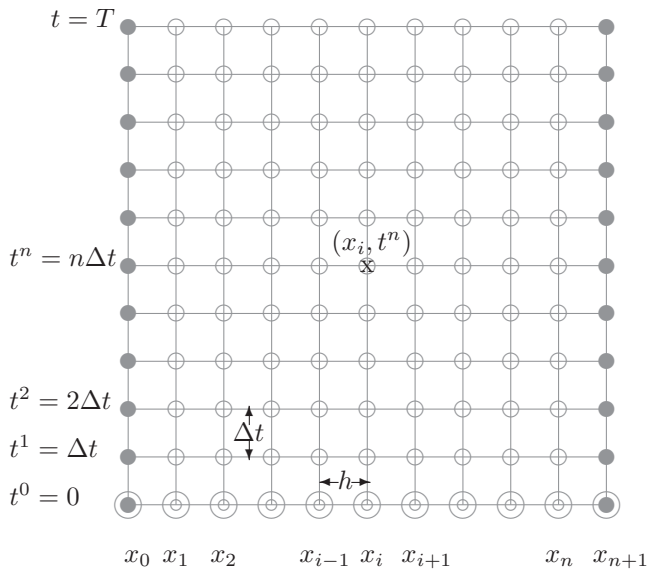

Figure 10.1: The spatial grid is denoted x_i, $i = 0, 1, \ldots, n+1$ and the temporal grid by t^n where the initial time is $t^0 = 0$ with we impose a uniform time step of Δt. The location of the enforcement of initial conditions is denoted by a double circle and boundary conditions by a solid circle.

We are given the solution at $t = 0$ at every grid point x_i, $i = 0, \ldots, n+1$ and our goal is to use a method to use this solution to approximate the solution at $t = \Delta t$. We will view the problem as having to solve a BVP for each fixed time. For the spatial discretization we can use finite differences or finite elements and for the temporal discretization we use a method such as a Runge-Kutta method or a multistep method or simply an Euler approximation. The concept of explicit or implicit time discretization to provide stable numerical results will be important here just as it was in the study of IVPs.

10.4 Mathematical background

When we discretize a nonlinear equation in a straightforward manner with finite differences or finite elements the resulting algebraic system of equations will be nonlinear. One option for solving this nonlinear system is to use the Newton-Rhapson method. Suppose we have a system of n nonlinear equations $f_i(\xi_1, \ldots, \xi_n) = 0$ in n independent variables ξ_i which we write in vector form $\mathbf{F}(\xi) = 0$ where the ith component of $\mathbf{F}$ is f_i. The Newton-Rhapson method is an iterative method which takes the form

$\mathbf{x}^0 = (x_1^0, x_2^0, \ldots, x_n^0)$ given
for $k = 0, 1, \ldots$
$\mathbf{x}^{k+1} = \mathbf{x}^k - J^{-1}(\mathbf{x}^k)F(\mathbf{x}^k)$

where $J(\mathbf{x})$ is the Jacobian matrix of partial derivatives with

$$J_{ij}(\mathbf{x}) = \frac{\partial f_i}{\partial x_j}$$

Typically one applies the method by solving the linear system

$$J(\mathbf{x}^k)(\mathbf{x}^{k+1} - \mathbf{x}^k) = -F(\mathbf{x}^k).$$

A disadvantage to Newton's method is that the Jacobian changes at each iteration so a different linear system must be solved for each k. The advantage is that if it converges, it typically converges quadratically so rapid convergence is achieve. If explicit formulas for the partial derivatives are not available then finite differences can be used to estimate the derivatives.

Chapter 11

Parabolic Problems

In this chapter we consider solving some standard parabolic equations with finite difference and finite element methods. In particular, we will look at the heat equation in one and two spatial domains, a nonlinear IBVP and a coupled system of IBVPs. We can view these types of problems as solving a BVP for each time step. Issues such as stability of time stepping schemes are important here just like they were for IVPs. We will compare the stability restrictions for both Forward and Backward Euler approximations to the time derivative term and will look at a second order in time scheme called Crank-Nicolson. We first consider finite differences for parabolic equations and then finite element methods. We will see that many results are a straightforward extension of our previous work.

11.1 Finite difference methods for the heat equation

When we studied the FDM for the two-point BVP or the Poisson equation we found that typically we discretized the $u''(x)$ term or the laplacian term Δu by using a second centered difference quotient in each independent variable. For example, in $\mathbb{R}^2$ we found that the finite difference equation at (x_i, y_j) for Poisson's equation is just

$$-U_{i-1,j} + 4U_{i,j} - U_{i+1,j} - U_{i,j-1} - U_{i,j+1} = h^2 f(x_i, y_j)$$

where $U_{ij} \approx u(x_i, y_j)$ and $\Delta x = \Delta y = h$. We found that if we impose Dirichlet boundary data and we have $M + 2$ grid points in the x-direction and $N + 2$ in the y-direction then the resulting matrix is $MN \times MN$ and is a block tridiagonal matrix. If $M = N$ then the matrix has dimension N^2 and has a total bandwidth of $\mathcal{O}(2N)$. We view solving an IBVP as solving a BVP at each time so we will rely on our previous results for BVPs. In this section we consider different approaches to handling the time term u_t. One obvious choice is to just

use an Euler approximation but recall that it is just first order. Also if we use the explicit Forward Euler method there are stability issues which require restrictions on the size of the time step Δt. We will also consider a scheme which will be second order accurate.

11.1.1 Euler time discretization

We begin by discretizing the heat equation in one spatial dimension

$$u_t - \nu u_{xx} = 0 \quad (x) \in [a, b], 0 < t \leq T \tag{11.1}$$

with homogeneous Dirichlet boundary data and an initial condition $u(x, 0) = u_0(x)$. For now we have assumed $f(x, t) = 0$ but a nonzero f is easily incorporated. We have already seen that a second order (in h) approximation to the u_{xx} is the second centered difference; consequently we will concentrate on choices for approximating the u_t term.

To discretizate u_t we want to simply choose a difference quotient and incorporate it into the equation. We let $U_i^n \approx u(x_i, , t^n)$ and at each time t^n we write a difference equation at all interior nodes We first choose a Forward Euler approximation for u_t and then compare this with a Backward Euler approximation.

Recall that the Forward Euler method for an IVP approximates the time derivative by looking forward in time from a point say t^n. As before we let $\Delta t = t^{n+1} - t^n$. Our difference equation is written at t^n and we write an equation at each interior grid point $(x_i,)$, $i = 1, 2, \ldots, N$ because we are imposing Dirichlet boundary conditions. Using Forward Euler we have

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} + \nu \frac{-U_{i-1}^n + 2U_i^n - U_{i+1j}^n}{h^2} = 0$$

or letting $\lambda = \nu \Delta t / h^2$ and simplifying we have

$$U_i^{n+1} = \lambda U_{i-1}^n + (1 - 2\lambda) U_i^n + \lambda U_{i+1j}^n \tag{11.2}$$

This is an *explicit* scheme because the unknown is given explicitly in terms of known quantities. We are given the solution at $t = 0$ so we set $U_i^0 = u_0(x_i)$ for $i = 0, N+1$ and then we simply "march in time". We know the solution at each $(x_i, 0)$ so to find the solution U_1^1 at $(x_1, \Delta t)$ we have

$$U_1^1 = \lambda U_0^0 + (1 - 2\lambda) U_1^0 + \lambda U_2^0$$

All terms on the right hand side are known from either the initial condition or the boundary conditions. We impose the difference equation at each $i = 1, 2, \ldots, N$ and determine all U_i^1. Then we use (11.2) with $n = 1$ to find the solution at U_i^2 for all $i = 1, 2, \ldots, N$. Note that we do not have to solve a linear system of equations.

Recall that when we used the Forward Euler method to solve an IVP we saw that the local truncation error was $\mathcal{O}(\Delta t^2)$ and the global error was $\mathcal{O}(\Delta t)$.

We know that the global error of the second centered difference is $\mathcal{O}(\Delta x^2)$. The following lemma shows that the local truncation error for the method is $\mathcal{O}(\Delta t(\Delta t + \Delta x^2))$.

Lemma 11.1. *Let $u(x,t)$ be the exact solution to the IBVP (11.1) and assume that it possesses four continuous derivatives in x and two in t. Then there exists a positive constant C, independent of u, Δt, and Δx such that*

$$\begin{aligned}\tau = \max_{\substack{1\le i\le N\\ 0\le n\le M}} \left|u(x_i,t^{n+1}) - [\lambda u(x_{i+1},t^n) + (1-2\lambda)u(x_i,t^n) + \lambda u(x_{i-1},t^n)]\right| \\ \le C\Delta t\big(\Delta t + (\Delta x)^2\big)\max_{x,t}\left|\frac{\partial^4 u}{\partial x^4}\right|\end{aligned} \tag{11.3}$$

where we have that $M\Delta t = T$.

We know that the local truncation error is the difference in the exact solution at time t^{n+1} and the value computed by the difference scheme starting from the exact solution. The difference now is that we have N values to compute at each time level. So we simply perform the calculation at a generic x_i and then take the maximum over all i, $i = 1,\ldots,N$. We have

$$\tau_i^{n+1} = \left|u(x_i,t^{n+1}) - \left[\lambda u(x_{i+1},,t^n) + (1-2\lambda)u(x_i,,t^n) + \lambda u(x_{i-1},,t^n)\right]\right|$$

As in our previous calculations in finding the local truncation error, we expand all terms in Taylor series; in this case we expand about the point (x_i,t^n). To simplify the exposition we use the notation $u_i^n = u(x_i,t^n)$. We first expand $u(x_i,t^{n+1})$ in terms of t

$$u_i^{n+1} = u(x_i,,t^n+\Delta t) = u_i^n + \Delta t(u_i^n)_t + \frac{\Delta t^2}{2}(u_i^n)_{tt} + \mathcal{O}(\Delta t^3)$$

and u_{i+1}^n and u_{i-1}^n in terms of x to get

$$u_{i+1}^n = u_i^n + \Delta x(u_i^n)_x + \frac{\Delta x^2}{2}(u_i^n)_{xx} + \frac{\Delta x^3}{3!}(u_i^n)_{xxx} + \frac{\Delta x^4}{4!}u_{xxxx}(\theta_i,t^n)$$

$$u_{i-1}^n = u_i^n - \Delta x(u_i^n)_x + \frac{\Delta x^2}{2}(u_i^n)_{xx} - \frac{\Delta x^3}{3!}(u_i^n)_{xxx} + \frac{\Delta x^4}{4!}u_{xxxx}(\rho_i,t^n)$$

Combining these results gives

$$\begin{aligned}\tau_i^{n+1} \quad = \quad & \Big|u_i^n + \Delta t(u_i^n)_t + \frac{\Delta t^2}{2}(u_i^n)_{tt} + \mathcal{O}(\Delta t^3) - (1-2\lambda)u_i^n \\ & -\lambda\Big[u_i^n + \Delta x(u_i^n)_x + \frac{\Delta x^2}{2}(u_i^n)_{xx} + \frac{\Delta x^3}{3!}(u_i^n)_{xxx} + \frac{\Delta x^4}{4!}u_{xxxx}(\theta_i,t^n)\Big] \\ & -\lambda\Big[u_i^n - \Delta x(u_i^n)_x + \frac{\Delta x^2}{2}(u_i^n)_{xx} - \frac{\Delta x^3}{3!}(u_i^n)_{xxx} + \frac{\Delta x^4}{4!}u_{xxxx}(\rho_i,t^n)\Big]\Big|\end{aligned}$$

Collecting terms and using the differential equation $u_t = \nu u_{xx}$ we have

$$\tau_i^{n+1} \quad = \quad \Big|u_i^n\big(1 - 1 + 2\lambda - 2\lambda\big)$$

$$
\begin{aligned}
&+\Delta x(u_i^n)_x\big(-\lambda+\lambda\big)+(u_i^n)_{xx}\Big(\nu\Delta t-\frac{\lambda\Delta x^2}{2}-\frac{\lambda\Delta x^2}{2}\Big)\\
&+(u_i^n)_{xxx}\lambda\Big[\frac{\Delta x^3}{3!}-\frac{\Delta x^3}{3!}\Big]\\
&+\Big[\frac{\Delta t^2}{2}(u_i^n)_{tt}-\lambda\frac{\Delta x^4}{4!}u_{xxxx}(\theta_i,t^n)+\lambda\frac{\Delta x^4}{4!}u_{xxxx}(\rho_i,t^n)\Big]\Big|
\end{aligned}
$$

The term multiplying $(u_i^n)_{xx}$ can be simplified by recalling that $\lambda=\nu\Delta t/\Delta x^2$; we have

$$\nu\Delta t-\lambda\Delta x^2=\nu\Delta t-\big(\frac{\nu\Delta t}{\Delta x^2}\big)(\Delta x)^2=0$$

Now all terms cancel except the last terms involving the second derivative in time and the fourth derivative in space. Note first that

$$\lambda\frac{\Delta x^4}{4!}=\frac{\nu\Delta t}{\Delta x^2}\frac{\Delta x^4}{4!}=C\Delta t(\Delta x)^2$$

so that these terms can be bounded by a constant times the maximum value of u_{xxxx} times $\Delta t(\Delta x)^2$. The other term is a constant times $(\Delta t)^2$ times the maximum value of the u_{tt}. Using the differential equation we see that $u_t=u_{xx}$ implies $u_{tt}=u_{xxt}=(u_t)_{xx}=\nu(u_{xx})_{xx}=\nu u_{xxxx}$ and so we can bound that term as $\mathcal{O}(\Delta t^2)$ times the maximum value of the fourth derivative of u with respect to x and the result is proved by taking the maximum over all points $1\le i\le N$ and all time.

Because the local truncation is is $\mathcal{O}\big(\Delta t(\Delta t+(\Delta x)^2\big)$ we expect the global error to be linear in time and quadratic in space; i.e., $\mathcal{O}\big(\Delta t+(\Delta x)^2\big)$. We will not prove this here but it is what we expect from our previous work and this statement can be made rigorous.

Recall from our study of IVPs that the forward Euler method has strict stability requirements. The following lemma gives a condition on λ that guarantees a stable solution. For simplicity we are assuming that homogeneous Dirichlet boundary conditions are imposed and we are taking no sources or sinks in the heat equation; modifications to the results can be made to handle $f\neq 0$ and inhomogeneous boundary conditions. The result says that if $\lambda\le\frac{1}{2}$ then the solution at any x_i and time t^n is bounded by the maximum value of the initial condition at any grid point, i.e., it satisfies a maximum principle and is thus stable.

Lemma 11.2. *Let $\lambda=\frac{\nu\Delta t}{\Delta x^2}\le\frac{1}{2}$. Then the solution U_i^n of the difference equation (11.2) satisfies the inequality*

$$\max_{\substack{1\le i\le N\\0\le n\le M}}|U_i^n|\le\max_{1\le i\le N}|u_0(x_i)|$$

Our difference equation is

$$U_i^{n+1}=\lambda U_{i+1}^n+(1-2\lambda)U_i^n+\lambda U_{i+1}^n$$

for $i=1,2,\ldots,N$. Because $0<\lambda\le\frac{1}{2}$, $1-2\lambda>0$ and we have

$$|U_i^{n+1}|\quad\le\quad\big|\lambda U_{i+1}^n+(1-2\lambda)U_i^n+\lambda U_{i+1}^n\big|$$

$$\begin{aligned} &\leq \lambda|U_{i+1}^n| + (1-2\lambda)|U_i^n| + \lambda|U_{i+1}^n| \\ &\leq \Big(\lambda + 1 - 2\lambda + \lambda\Big) \max_i |U_i^n| \\ &\leq \max_i |U_i^n| \end{aligned}$$

Applying this result recursively gives

$$|U_i^{n+1}| \leq \max_i |U_i^n| \leq \max_i |U_i^{n-1}| \leq \cdots \leq \max_i |U_i^0| = \max_i |u_0(x_i)|$$

This stability requirement is very stringent and depends on the rate of diffusion ν. For example if $\nu = 1$ the requirement on Δt is that $\Delta t \leq \frac{(\Delta x)^2}{2}$ so if , e.g., $\Delta x = 0.01$ then this requires $\Delta t \leq 5 \times 10^{-5}$. If the diffusion rate is rapid, i.e., a large value of ν, then even a smaller time step must be taken. For example, if $\nu = 100$ and $\Delta x = 0.01$ then $\Delta t \leq 5 \times 10^{-7}$. For these reasons implicit methods are widely used.

We now turn to implementing the Backward Euler approximation to u_t. We have

$$\frac{U_i^n - U_i^{n-1}}{\Delta t} + \nu \frac{-U_{i-1}^n + 2U_i^n - U_{i+1}^n}{h^2} = 0$$

or putting all known quantities on the right hand side and letting $\lambda = \frac{\nu \Delta t}{\Delta x^2}$ gives

$$-\lambda U_{i-1}^n + \ (1+2\lambda)U_i^n - \lambda U_{i+1}^n = U_i^{n-1}$$

To solve a BVP using the FD method we saw that it was necessary to solve a linear system of equations; the same is true for IBVPs when one uses an implicit time approximation.

To see the structure of the resulting matrix, we look at (11.2) and see clearly that it is tridiagonal as was the matrix for solving the corresponding two-point BVP. For the two-dimensional heat equation we get the same block tridiagonal structure that we had for the Poisson equation in two dimensions. To modify a code which solves the Poisson equation using a second centered difference quotient so that it solves the heat equation using the Forward Euler method we see that all that has to be done is to add a loop over the number of time steps and to modify the entries of the matrix and to adjust the right hand side by the term $\frac{h^2}{\Delta t} U_{ij}^{n-1}$. Of course now one has to also keep the solution from the previous time step so at any one time you must store solutions at time t^n and the new solution that is computed at time t^{n+1}. Note that you should not store all solutions for all time because the storage requirements would become onerous with a large number of time steps. To plot the solution or make a movie, the approximations should be written to file.

It is important to note that although we have to solve a linear system at each time step, the coefficient matrix does NOT change from one time step to another. Thus an efficient approach is to form the matrix outside of the time stepping loop and then perform a factorization such as a Cholesky LL^T. Then at each time step a new right hand side is formed and we only have to perform one backward solve and one forward solve which is efficient.

It can be shown that the Backward Euler time discretization in the heat equation results in the same local truncation error as the Forward Euler approximation. However, one can demonstrate that the Backward Euler approximations results in a convergence scheme with no restrictions on λ whereas the Forward Euler required $\lambda \leq \frac{1}{2}$ to guarantee stability and thus convergence.

What should we do if we want to use a second order in time difference quotient for u_t? If we use the centered difference quotient

$$u_t(x_i, t^n) \approx \frac{U_i^{n+1} - U^{n-1}}{2\Delta t}$$

then when we implement this scheme we have

$$\frac{U_i^{n+1} - U_i^{n-1}}{2\Delta t} + \frac{-U_{i-1}^n + 2U_i^n - U_{i+1}^n}{h^2} = 0$$

which is a three time level explicit scheme because we have terms at times t^{n-1}, t^n and t^{n+1}. So just as in the case of multistep methods for IVPs we would need the solution at two time steps to use this approach.

11.1.2 A stable two time level second order accurate scheme

Our goal now is to derive a second order accurate in time scheme which only involves two time levels and which has no stability requirements. The scheme we derive is called the **Crank-Nicolson method**. The key to deriving the scheme is to write a centered difference approximation at the time $t^{n+\frac{\Delta t}{2}}) = t^{n+\frac{1}{2}}$ so that we have the time levels t^n and T^{n+1}. When we do this, then the approximation in two spatial dimensions is

$$u_t(x_i, y_j, t^{n+\frac{1}{2}}) \approx \frac{u(x_i, y_j, t^{n+1}) - u(x_i, y_j, t^n)}{\Delta t}$$

is a centered difference scheme and is thus second order accurate. Note that the denominator is Δt not $2\Delta t$ because the spacing between time levels is $2(\frac{\Delta t}{2})$.

What do we do with the spatial terms? If we approximate $u_{xx}(x_i, y_j, t^{n+\frac{1}{2}})$ by a second centered difference at $t^{n+\frac{1}{2}}$ then we are introducing unknowns there. The way to get around this is to take the approximation to, e.g., $u_{xx}(x_i, y_j, t^{n+\frac{1}{2}})$ as the average of the centered difference approximation at (x_i, y_j, t^n) and the approximation at (x_i, y_j, t^{n+1}). For example,

$$\begin{aligned} u_{xx}(x_i, y_j, t^{n+\frac{1}{2}}) \quad \approx \quad & \frac{\nu}{2}\left(\frac{-U_{i-1,j}^{n+1} + 4U_{i,j}^{n+1} - U_{i+1,j}^{n+1} - U_{i,j-1}^{n+1} - U_{i,j+1}^{n+1}}{h^2}\right. \\ & \left. + \frac{-U_{i-1,j}^{n} + 4U_{i,j}^{n} - U_{i+1,j}^{n} - U_{i,j-1}^{n} - U_{i,j+1}^{n}}{h^2}\right) \end{aligned}$$

All terms in the equation are approximated at $t^{n+\frac{1}{2}}$ so if we have a term u as in the Helmholtz BVP we simply approximate it by averaging its value at the $(n+1)$st time level with the value at the nth time level.

To solve the heat equation in two spatial dimensions (10.2) using the Crank-Nicolson method we have the generic difference equation

$$\begin{aligned}\frac{U_{i,j}^{n+1}-U_{i,j}^{n}}{\Delta t}+\frac{1}{2}\Bigg(&\frac{-U_{i-1,j}^{n+1}+4U_{i,j}^{n+1}-U_{i+1,j}^{n+1}-U_{i,j-1}^{n+1}-U_{i,j+1}^{n+1}}{h^2}\\&+\frac{-U_{i-1,j}^{n}+4U_{i,j}^{n}-U_{i+1,j}^{n}-U_{i,j-1}^{n}-U_{i,j+1}^{n}}{h^2}\Bigg)\\&=\frac{1}{2}\left(f(x_i,y_j,t^n)+f(x_i,y_j,t^{n+1})\right)\end{aligned}$$

Bringing the known terms to the right hand side of the equation and letting $\lambda=\nu\frac{\Delta t}{h^2}$ produces the difference scheme

$$\begin{aligned}-\lambda U_{i-1,j}^{n+1}+\big(2+4\lambda\big)U_{i,j}^{n+1}-\lambda U_{i+1,j}^{n+1}-\lambda U_{i,j-1}^{n+1}-\lambda U_{i,j+1}^{n+1}&\\=\Delta t\big(f(x_i,y_j,t^n)+f(x_i,y_j,t^{n+1})\big)&\\+\lambda U_{i-1,j}^{n}+\big(2-4\lambda\big)U_{i,j}^{n}+\lambda U_{i+1,j}^{n}+\lambda U_{i,j-1}^{n}+\lambda U_{i,j+1}^{n}&\end{aligned}$$

The coefficient matrix has the same block tridiagonal structure as when we solved the elliptic Poisson equation. As in the case of Forward Euler method one has to store the solution at the previous time at all grid points. Note that the matrix on the left only differs from that for Forward Euler in the diagonal element but the right hand side is considerably different. Once again the coefficient matrix does not change in time as long as the time step remains constant so that it can be factored once and at each time step a forward and back solve are required.

The Crank-Nicolson scheme has an accuracy of $\mathcal{O}(\Delta t^2+\Delta x^2)$. One can demonstrate that the Crank-Nicolson scheme is unconditionally stable so we do not have to worry about choosing Δt small enough to avoid numerical instabilities. If we choose $\Delta t=\Delta x$ then the scheme will be second order overall in both space and time. If we make $\Delta t << \Delta x$ then it is wasteful in the sense that the dominant error will be $\mathcal{O}(\Delta x^2)$.

11.1.3 A coupled system of PDEs

We now consider a simple coupled system of parabolic equations. Specifically we seek $u(\mathbf{x},t)$ and $v(\mathbf{x},t)$ satisfying

$$\begin{aligned}u_t-\nu\Delta u+v &= f(\mathbf{x},t) \quad \mathbf{x}\in\Omega, 0<t\le T\\ v_t-\nu\Delta v+u &= g(\mathbf{x},t) \quad \mathbf{x}\in\Omega, 0<t\le T\\ u(\mathbf{x},0) &= u_0(\mathbf{x}) \qquad v(\mathbf{x},0)=v_0(\mathbf{x})\\ u(x,t) &= 0, v(x,t)=0 \quad \mathbf{x}\in\Gamma, t\ge 0\end{aligned} \tag{11.4}$$

We discretize using a second centered difference for the laplacian term and the implicit Backward Euler for the time derivative. Similar to our previous work we have the difference equations at (x_i,y_j,t^n)

$$-\lambda U_{i-1,j}^{n}+(1+4\lambda)\,U_{i,j}^{n}-\lambda U_{i+1,j}^{n}-\lambda U_{i,j-1}^{n}-\lambda U_{i,j+1}^{n}+\Delta t V_{i,j}^{n} \;=\; \Delta t f(x_i,y_j,t^n)+U_{ij}^{n-1}$$

$$-\lambda V^n_{i-1,j} + (1+4\lambda)\, V^n_{i,j} - \lambda V^n_{i+1,j} - \lambda V^n_{i,j-1} - \lambda V^n_{i,j+1} + \Delta t U_{i,j} \quad = \quad \Delta t g(x_i, y_j, t^n) + V^{n-1}_{ij}$$

where $U^n_{i,j} \approx u(x_i, y_j, t^n)$ and $V^n_{i,j} \approx v(x_i, y_j, t^n)$ and $\lambda = \nu \Delta t / h^2$. As we have seen before it is typically better to number the unknowns alternating between u and v instead of including all the unknowns for u and then for v. So if we have $N+2$ nodes on each side with N^2 interior nodes, the resulting matrix is $2N^2 \times 2N^2$. The first $2N$ rows of the coefficient matrix are given here to illustrate its structure for the case $N = 3$. We indicate above the matrix the unknown associated with each column for clarity; the rows are numbered in the same way. We have the first six rows as follows:

U^n_{11}	V^n_{11}	U^n_{21}	V^n_{21}	U^n_{31}	V^n_{31}	U^n_{12}	V^n_{12}	U^n_{22}	V^n_{22}	U^n_{32}	V^n_{32}
$4\lambda+1$	Δt	$-\lambda$	0	0	0	$-\lambda$	0	0	0	0	0
Δt	$4\lambda+1$	0	$-\lambda$	0	0	0	$-\lambda$	0	0	0	0
$-\lambda$	0	$4\lambda+1$	Δt	$-\lambda$	0	0	0	$-\lambda$	0	0	0
0	$-\lambda$	Δt	$4\lambda+1$	0	$-\lambda$	0	0	0	$-\lambda$	0	0
0	0	$-\lambda$	0	$4\lambda+1$	Δt	$-\lambda$	0	0	0	$-\lambda$	0
0	0	0	$-\lambda$	Δt	$4\lambda+1$	0	$-\lambda$	0	0	0	$-\lambda$

Consequently our block structure will be

$$\begin{pmatrix} S & -\lambda I & & \\ -\lambda I & S & -\lambda I & \\ & \ddots & \ddots & \ddots \\ & & -\lambda I & S \end{pmatrix}$$

where each block is $2N \times 2N$ and we have N rows of block matrices. The matrix S has the structure

$$S = \begin{pmatrix} 4\lambda+1 & \Delta t & -\lambda & & & & \\ \Delta t & 4\lambda+1 & 0 & -\lambda & & & \\ -\lambda & 0 & 4\lambda+1 & \Delta t & -\lambda & & \\ 0 & -\lambda & \Delta t & 4\lambda+1 & 0 & -\lambda & \\ & & \ddots & \ddots & \ddots & & \\ 0 & 0 & -\lambda & 0 & 4\lambda+1 & \Delta t & \\ 0 & 0 & 0 & -\lambda & \Delta t & 4\lambda+1 & \end{pmatrix}$$

with a total bandwidth of five and is symmetric.

11.1.4 A nonlinear parabolic equation

The last example we look at for discretizing the heat equation using finite differences is to add a nonlinear term to the heat equation. For simplicity we consider the heat equation in one spatial dimension. We seek a solution to

$$\begin{aligned} u_t - u_{xx} + u^2 &= f(x,t) \\ u(x,0) &= u_0(x) \\ u(0) &= u(1) = 0 \end{aligned} \tag{11.5}$$

If we use Backward Euler in time and a second centered difference in space we have the difference equation

$$-\lambda U_{i-1}^n + (1+2\lambda)\,U_i^n - \lambda U_{i+1}^n + \Delta t\big(U_i^n\big)^2 = \Delta t f(x_i, t^n) + U_i^{n-1}$$

at the point (x_i, t^n). Note that this is a nonlinear difference equation due to the term $(U_i^n)^2$ so the difference equations can't be written as a linear system $A\mathbf{x} = \mathbf{f}$. Consequently we must use a method to linearize this problem and we expect that we will have to perform an iteration at each time step to solve these nonlinear equations.

One approach that is sometimes used is to "lag" the nonlinear term. By this we mean to set up an iteration where the nonlinear term $(U_i^n)^2$ is evaluated at the previous iteration so that the system becomes linear. The initial guess for this term is just the value of U_i at the previous time step.

A common way to solve this nonlinear system is to use a method such as the Newton-Rhapson method or a Quasi-Newton method for solving a system of nonlinear algebraic equations. To use the Newton method we must be able to compute the Jacobian. Recall that Newton's method will converge in one iteration for a linear system so in that case the Jacobian is just the same as the coefficient matrix. This means that all of the linear terms in our equation will be the same as in the linear case. For example, when we compute $\frac{\partial}{\partial U_{i-1}^n}$ we just get $-\lambda$ which is the usual term in the coefficient matrix. The difference comes when we calculate $\frac{\partial}{\partial U_i^n}$ which includes the nonlinear term. In this case we get $(2\lambda + 1) + 2(U_i)^k$ where the term $(U_i)^k$ denotes the solution at the previous Newton iterate. Consequently, for the Jacobian we just have to modify the diagonal entry of the standard coefficient matrix. We iterate at each time step until the normalized difference in successive iterates is sufficiently small. As an initial guess to Newton's method we use the solution at the previous time step. If Newton's method fails to converge then we should reduce the time step and try again. This gives us a natural way to determine the appropriate step size. If Newton's method converges in one or two steps then we can probably increase the time step. If it fails to converge, we must reduce the time step. The following example illustrates the solution of the nonlinear system for small N for a one-dimensional heat equation.

11.2 Finite element methods for the heat equation

We now want to solve the heat equation using finite element methods. Recall that the first step in the FEM is to write an appropriate weak formulation.

Seek u satisfying

$$\int_0^1 u_t v\,dx + \nu \int_0^1 u_x v_x\,dx = \int_0^1 f v\,dx \tag{11.6}$$

or in higher dimensions

$$\int_\Omega u_t v \, d\Omega + \nu \int_\Omega \nabla u \cdot \nabla v \, d\Omega = \int_\Omega f v \, d\Omega \tag{11.7}$$

We now choose a finite dimensional subspace of our underlying solution space and write the corresponding discrete weak formulation. Now for each time t we let $u^h(x,t) = \sum_{j=1}^N c_j(t)\phi_j(\mathbf{x})$ and we see that in this case our coefficients are functions of time so when we compute $\frac{\partial}{\partial t}u^h$ we get $\sum_{j=1}^N c_j'(t)\phi_j(\mathbf{x})$. Thus for the two-dimensional heat equation we have

$$\sum_{j=1}^N c_j'(t) \int_\Omega \phi_j(\mathbf{x})\phi_i(\mathbf{x}) \, d\Omega + \nu \sum_{j=1}^N c_j(t) \int_\Omega \nabla\phi_j(\mathbf{x}) \cdot \nabla\phi_i(\mathbf{x}) \, d\Omega = \int_\Omega f\phi_i(\mathbf{x}) \, d\Omega$$

for $i = 1, \ldots, N$. This is just a system of IVPs for the unknowns c_j at any time t. Consequently we can use any of our schemes that we learned for solving IVPs. So one approach to solving this is to discretize in space using FEMs and solve the resulting IVP for the unknowns $c_j(t)$ at time t. For the initial conditions for the system of IVPs we use $u^h(\mathbf{x},0) = \sum_{j=1}^N c_j(0)\phi_j(\mathbf{x}) = u_0(\mathbf{x})$ so we take $c_j(0)$, $j = 1, 2, \ldots, N$ as the nodal values of the initial condition.

Another approach is to actually discretize the temporal term u_t so that we have a fully discrete method, i.e., everything has been discretized. We could choose to use finite elements in time but typically this is not often done because in most situations we gain nothing and it merely succeeds in complicating the situation. The common approach is to discretize the temporal derivative using a finite difference quotient. We first choose a Backward Euler approximation because we know that it is stable and so we don't have restrictions on the time step. If we let our FEM approximation at time level t^n be expressed as $U^n = \sum_{j-1}^N c_j^n \phi_i(\mathbf{x})$ then we have the full discrete problem

$$\int_\Omega \frac{U^n - U^{n-1}}{\Delta t}\phi_i(\mathbf{x}) \, d\Omega + \nu \int_\Omega \nabla U^n \cdot \nabla\phi_i(\mathbf{x}) \, d\Omega = \int_\Omega f\phi_i(\mathbf{x}) \, d\Omega \quad \text{for } i = 1, \ldots, N$$

or equivalently

$$\sum_{j=1}^N c_j^n \int_\Omega \phi_j(\mathbf{x})\phi_i(\mathbf{x}) \, d\Omega + \nu\Delta t \sum_{j=1}^N c_j^n \int_\Omega \nabla\phi_j(\mathbf{x}) \cdot \nabla\phi_i(\mathbf{x}) \, d\Omega = \Delta t \int_\Omega f\phi_i(\mathbf{x}) \, d\Omega + \sum_{j=1}^N c_j^{n-1} \int_\Omega \phi_j(\mathbf{x})\phi_i(\mathbf{x}) \, d\Omega$$

for $i = 1, \ldots, N$. If we let M be the mass matrix and K the stiffness matrix with entries

$$M_{ij} = \int_\Omega \phi_j(\mathbf{x})\phi_i(\mathbf{x}) \, d\Omega \quad \text{and} \quad K_{ij} = \int_\Omega \nabla\phi_j(\mathbf{x}) \cdot \nabla\phi_i(\mathbf{x}) \, d\Omega$$

we can write the resulting system as

$$(M + \nu\Delta t K)\mathbf{c}^n = \Delta t\mathbf{F} + M\mathbf{c}^{n-1}$$

where $F_i = \int_\Omega f\phi_i(\mathbf{x})\, d\Omega$.

If we used the explicit Forward Euler discretization then we still have to solve a linear system unlike the FD case. To see this note that with Forward Euler we have

$$\int_\Omega \frac{U^n - U^{n-1}}{\Delta t}\phi_i(\mathbf{x})\, d\Omega + \nu\int_\Omega \nabla U^{n-1}\cdot\nabla\phi_i(\mathbf{x})\, d\Omega = \int_\Omega f\phi_i(\mathbf{x})\, d\Omega \quad \text{for } i = 1,\ldots,N$$

or equivalently

$$\begin{aligned}\sum_{j=1}^N c_j^n \int_\Omega \phi_j(\mathbf{x})\phi_i(\mathbf{x})\, d\Omega &= -\nu\Delta t\sum_{j=1}^N c_j^{n-1}\int_\Omega \nabla\phi_j(\mathbf{x})\cdot\nabla\phi_i(\mathbf{x})\, d\Omega + \Delta t\int_\Omega f\phi_i(\mathbf{x})\, d\Omega \\ &\quad + \sum_{j=1}^N c_j^{n-1}\int_\Omega \phi_j(\mathbf{x})\phi_i(\mathbf{x})\, d\Omega\end{aligned}$$

which leads to the system

$$M\mathbf{c}^n = \Delta t\mathbf{F} + (M - \nu\Delta t K)\mathbf{c}^{n-1}$$

Whether we use the Forward or Backward Euler approximation to u_t the coefficient matrix doesn't change as long as Δt remains constant. Thus an efficient approach is to form the matrix outside of the time stepping loop and then perform a factorization such as LL^T. Then at each time step a new right hand side is formed and we only have to perform one backward solve and one forward solve.

We can also use the Crank-Nicolson approach to get a second order scheme. Recall that we write the equation at $t^{n+\frac{1}{2}}$ so that we get a second order approximation in time. The spatial terms are averaged at time t^n and t^{n+1}. We have

$$\begin{aligned}&\int_\Omega \frac{U^n - U^{n-1}}{\Delta t}\phi_i(\mathbf{x})\, d\Omega + \frac{\nu}{2}\Big[\int_\Omega \nabla U^n\cdot\nabla\phi_i(\mathbf{x})\, d\Omega + \int_\Omega \nabla U^{n-1}\cdot\nabla\phi_i(\mathbf{x})\, d\Omega\Big] \\ &\qquad = \frac{1}{2}\Big[\int_\Omega f\phi_i(\mathbf{x}, t^n)\, d\Omega + \int_\Omega f\phi_i(\mathbf{x}, t^{n-1})\, d\Omega\Big] \quad \text{for } i = 1,\ldots,N\end{aligned}$$

or equivalently

$$\begin{aligned}&\sum_{j=1}^N c_j^n \int_\Omega \phi_j(\mathbf{x})\phi_i(\mathbf{x})\, d\Omega + \frac{\nu\Delta t}{2}\sum_{j=1}^N c_j^n\int_\Omega \nabla\phi_j(\mathbf{x})\cdot\nabla\phi_i(\mathbf{x})\, d\Omega \\ &\qquad = \frac{\Delta t}{2}\Big[\int_\Omega f(\mathbf{x}, t^n)\phi_i(\mathbf{x})\, d\Omega + \int_\Omega f(\mathbf{x}, t^{n-1})\phi_i(\mathbf{x})\, d\Omega\Big] + \sum_{j=1}^N c_j^{n-1}\int_\Omega \phi_j(\mathbf{x})\phi_i(\mathbf{x})\, d\Omega\end{aligned}$$

$$+\frac{\nu\Delta t}{2}\sum_{j=1}^{N} c_j^{n-1}\int_\Omega \nabla\phi_j(\mathbf{x})\cdot\nabla\phi_i(\mathbf{x})\,d\Omega\,.$$

Writing as a matrix equation yields

$$\Big(M+\frac{\nu\Delta t}{2}K\Big)\mathbf{c}^n=\Big(M+\frac{\nu\Delta t}{2}K\Big)\mathbf{c}^{n-1}+\frac{\Delta t}{2}\Big(\mathbf{F}^n+\mathbf{F}^{n-1}\Big)$$

Perhaps a better approach to getting a second order approximation in time is to use a second order Backward Difference Formula (BDF) . Recall that the Backward Euler scheme is a first order BDF and the coefficients for a second and higher order BDFs are given in the IVP chapter. A BDF formula is a multistep formula so one needs a way to get additional starting values. Suppose that you feel that a step size of Δt will be used (at least initially). Then one way to get the starting values is to just take a step of length $(\Delta t)^2$ with Backward Euler. Recall that Backward Euler is linear in the time step so if we take a step of length $(\Delta t)^2$ then the error is $\mathcal{O}(\Delta t^2)$. If we take a step of length Δt with Backward Euler then this would be unacceptable because the error is $\mathcal{O}(\Delta t)$. Of course one would need the BDF formula with variable time steps but in practice a variable time step algorithm should be used to solve IBVPs anyway.

Approximating the parabolic system (11.4) with FEMs is a straightforward exercise. In addition the FEM discretization of the nonlinear problem (11.5) using finite elements leads to a nonlinear algebraic system of equations which can be solved in a manner analogous to the finite difference case.