An Introduction to Numerical Methods for Differential Equations

Janet Peterson

Fall 2015

Chapter 1

Introduction

Differential equations arise in many disciplines such as engineering, mathematics, sciences such as physical, chemical and biological as well as economics. These differential equations often result from modeling some phenomenon whether it is a simple process such as population modeling or a complicated process such as weather forecasting. Simple differential equations which only involve one independent variable can sometimes be solved by evaluating an integral, but, in general, even when the unknown involves a single variable analytic solutions are not easily obtained. When the unknown in the differential equation is a function of more than one independent variable, then finding an analytic solution is typically much more difficult and even if an exact solution is found, it is often in terms of an infinite series. In reality, the solutions to most differential equations are approximated rather than solved exactly.

In this chapter we give a brief introduction to differential equations and some of the terminology and notation that is commonly used. We then give a brief introduction to the concept of discretization and discuss how the numerical methods we derive can be implemented on a computer and verified that they are producing valid results. Lastly, the strategy we take for studying this broad topic of numerically approximating the solution to differential equations is outlined.

1.1 Differential equations

A differential equation (DE) is simply an equation which involves one or more derivatives of an unknown function; however, this function may depend on one or more independent variables. If the unknown is a function of a single variable then the derivatives occurring in the equation are ordinary derivatives and the equation is called an **ordinary differential equation (ODE)**. If the unknown is a function of more than one independent variable then the derivatives are partial derivatives and the equation is called a **partial differential equation (PDE)**.

First consider the following two examples of ODEs where the unknown depends

only on one independent variable:

$$y'(t) = y^2(t)e^{-t} \tag{1.1}$$

and

$$-u''(x) + 2u(x) = 3\sin x\,. \tag{1.2}$$

In (1.1) the independent variable is denoted by t and the dependent variable is denoted by y; in (1.2) the independent variable is denoted by x and the dependent variable is denoted by u. How we denote the dependent and independent variables doesn't matter; for example, $w'(x) = w^2(x)e^{-x}$ is the same equation as (1.1). In the sequel we will use notation such as dy/dt and $y'(t)$ interchangeably.

In many situations the dependent variable doesn't simply depend on one independent variable. For example, suppose we let u denote the temperature in a rod at any time t. Then clearly u depends upon both the location in the rod and the time. If we assume the temperature in the rod is constant in any cross section so that the spatial location only depends on x then $u = u(x,t)$. A differential equation can be obtained to model the heat transfer in this rod but it will be a PDE because the unknown u is a function of two independent variables. Recall from calculus that when a function depends on two or more independent variables, then we must use partial differentiation. Throughout this text we use the standard notation for partial derivatives such as

$$\frac{\partial u}{\partial x} = u_x, \qquad \frac{\partial^2 u}{\partial x^2} = u_{xx}, \qquad \frac{\partial^2 u}{\partial y \partial x} = u_{xy}\,.$$

See Appendix I for a review of partial differentiation. Using this notation, the PDE that models the heat transfer in a rod where the temperature is constant in any cross section is given by

$$u_t - u_{xx} = 0 \tag{1.3}$$

where there are no heat sources or sinks.

1.1.1 Initial and boundary value problems

In calculus we learned how to obtain analytic solutions for simple ODEs such as

$$y'(t) = -\sin t + t\,.$$

For this ODE we simply find a function whose derivative is $-\sin t + t$ which can be done by inspection or formally by integrating both sides of the equation with respect to t. We have

$$\int y'(t)\,dt = \int (-\sin t + t)\,dt \Rightarrow y(t) + C_1 = \cos t + \frac{t^2}{2} + C_2 \Rightarrow y(t) = \cos t + \frac{t^2}{2} + C$$

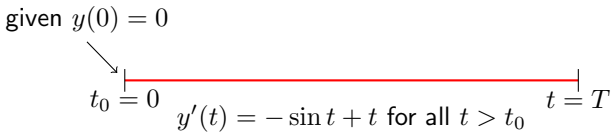
for arbitrary constants C_1, C_2, C. Consequently the solution is only determined up to a constant. In order to uniquely determine $y(t)$ we need to specify an *auxiliary condition* such as specifying y at some point. For example, if we specify $y(0) = 0$

then $y(t) = \cos t + t^2/2 - 1$ because $y(0) = \cos 0 + 0 + C = 0$ implies $C = -1$. This auxiliary condition is called an **initial condition** and the problem

$$\begin{aligned} y'(t) &= -\sin t + t \quad 0 < t \leq T \\ y(0) &= 0 \end{aligned}$$

is called an **initial value problem (IVP)**; here T denotes the final time. In general, for an IVP where the highest derivative occurring in the equation is one we are given the unknown y at some time $t = t_0$ and are asked to determine y for subsequent times from the fact that we know its first derivative. A generic IVP is illustrated pictorially below.

Sample IVP: Find $\mathbf{y(t)}$ for all $\mathbf{t_0 < t \leq T}$



If our differential equation is $y''(t) = -\sin(t) + t$ instead of $y'(t) = -\sin(t) + t$ then we would expect to have a solution involving two independent constants; the general solution to this equation is $y(t) = \sin t + t^3/6 + C_1 t + C_2$ which is easily found by integrating the equation twice. An example of an IVP for this equation is to specify both y and y' at t_0. An IVP requires the auxiliary conditions to all be specified at the same time t_0. Note that we have used t as the independent variable in these examples but we could have easily used x and assumed the unknown was given at some location x_0; then the goal would be to determine its position for subsequent locations.

An example of an IVP is modeling the growth of a population where we know the initial population size, we have information on how the population grows with time, (i.e., we have information on the first derivative) and we want to determine the population at subsequent times. Another example is the motion of a simple mechanical system such as a mass suspended from a spring where we want to know the position (e.g., the displacement from the rest position of the spring) at subsequent times. This system is modeled using Newton's second law of motion and thus an equation containing a second derivative of the displacement is given; in this case we typically specify the initial displacement of the spring due to the mass and the initial velocity.

Initial value problems can be either ordinary or partial differential equations; however, here we focus on ODEs. In addition to IVPs being important in their own right, understanding of this type of problem will form a foundation for the study of time dependent partial differential equations.

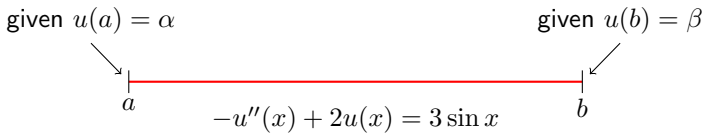
Oftentimes the differential equation models some phenomenon that does not depend on time such as a state of equilibrium. For example, suppose we take the differential equation given in (1.2) and require it to hold for $a < x < b$. Because the equation has a second derivative we know that we need to specify two

auxiliary equations to completely determine the solution. One option is to specify the unknown u at the boundary, i.e., specify $u(a)$ and $u(b)$. These are called **boundary conditions** and the problem

$$\begin{aligned} -u''(x) + 2u(x) &= 3\sin x \quad a < x < b \\ u(a) &= \alpha \\ u(b) &= \beta \end{aligned}$$

is called a **boundary value problem (BVP)**. This BVP in one dimension is illustrated pictorially below.

Sample BVP in 1D: Find $\mathbf{u(x)}$ for all $\mathbf{a < x < b}$

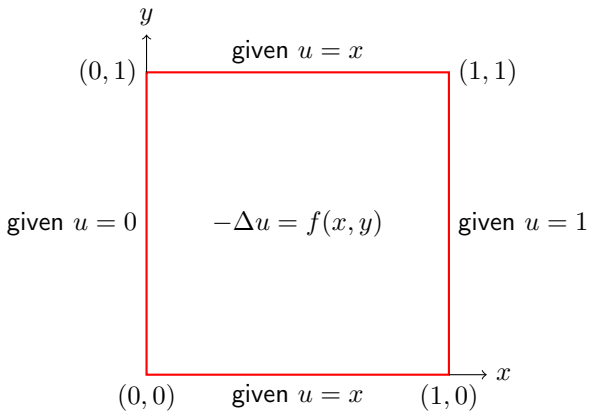


Boundary value problems can be PDEs too. As an example, suppose our unknown is $u(x, y)$ and our domain is the unit square $(0, 1) \times (0, 1)$ and we have the BVP

$$\begin{aligned} -\Delta u = -(u_{xx} + u_{yy}) &= f(x, y) \quad 0 < x < 1, 0 < y < 1 \\ u &= 0 \quad \text{for } x = 0 \qquad u = 1 \quad \text{for } x = 1 \\ u &= x \quad \text{for } y = 0, 1\,. \end{aligned}$$

This BVP is illustrated pictorially below. Note that the boundary conditions are compatible in the sense that the boundary condition $u = x$ at the top and bottom is zero at $x = 0$ and one at $x = 1$ so it agrees with the boundary conditions at the left and right of the domain.

Sample BVP in 2D: Find $\mathbf{u(x, y)}$ for all $\mathbf{0 < x, y < 1}$



Of course there are other types of boundary conditions than specifying the unknown on the boundary. We can also specify the derivative of the unknown or a combination of the unknown and a derivative. It is important to understand the distinction between BVPs and IVPs because we will see that the way we numerically approximate the solution of an IVP and a BVP are very different.

BVPs require auxiliary conditions to be imposed at the extremes of the domain (i.e., at the boundaries) whereas IVPs require auxiliary conditions to be imposed at the same point.

Suppose we return to (1.3) which is a PDE that models heat transfer in a rod under certain assumptions. For this problem the unknown is a function of both time and space so it is neither an IVP nor a BVP but rather a combination of the two. We call this an **initial boundary value problem (IBVP)** where we specify both boundary conditions and initial condition(s). An example of an IBVP is the heat equation

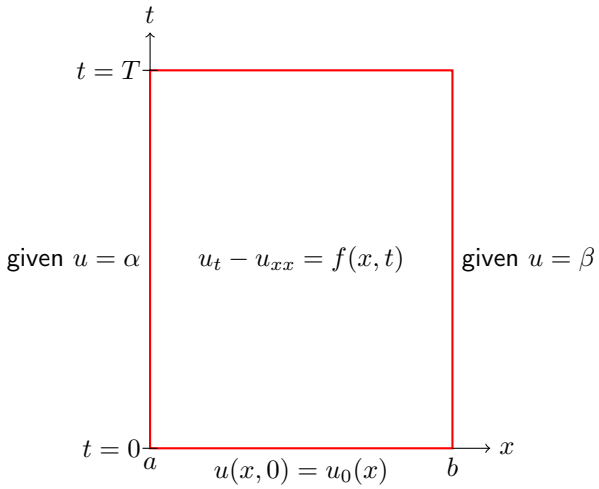
$$u_t - u_{xx} = f(x,t) \quad a < x < b, 0 < t \leq T \tag{1.4a}$$

$$u(x,0) = u_0(x) \quad a < x < b \tag{1.4b}$$

$$u(a) = \alpha, \quad u(b) = \beta\,, \tag{1.4c}$$

where $f(x,t)$, $u_0(x)$, α, and β are given. The initial condition is specified in (1.4b) and the two boundary conditions are specified in (1.4c). This IBVP is described pictorially below. Note that the initial condition and the boundary conditions should be compatible in the sense that $u_0(a) = \alpha$ and $u_0(b) = \beta$.

Sample IBVP in 1D: Find $\mathbf{u(x,t)}$ for all $\mathbf{a < x < b}$, $\mathbf{0 < t \leq T}$



Example 1.1. **Classification of DEs and their auxiliary conditions.**

Determine (i) if each differential equation is an ODE or a PDE, then (ii) classify each problem as an IVP, a BVP or an IBVP and (iii) identify which conditions are initial conditions and which are boundary conditions. If the problem is an IBVP then verify the compatibility between the initial and boundary conditions.

1. Let $u = u(x)$, then
$$u'''' - 2u'' + u = 0 \quad 0 < x < 1$$
$$u(0) = u(1) = 1, \quad u''(0) = 0, \quad u''(1) = -1$$

2. Let $u = u(x)$, then
$$u'' = x^3 u^2 \quad 1 < x < 10$$
$$u(1) = 4, \quad u'(1) = 0$$

3. Let $u = u(x, y)$, then
$$-(u_{xx} + u_{yy}) = f(x, y) \quad 0 < x < 1, 0 < y < 2$$
$$u(0, y) = u(1, y) = 1 \quad u_y(x, 0) = u_y(x, 2) = 0$$

4. Let $u = u(x, t)$, then
$$u_t + uu_x - u_{xx} = f(x, t) \quad 0 < x < 2, 0 < t \leq 1$$
$$u(0, t) = \cos \pi t, \quad u(2, t) = 9$$
$$u(x, 0) = x^3 + 1\,.$$

1. This problem is an ODE because the differential equation only involves ordinary derivatives. It is a BVP because it has boundary conditions given at the extremities of the domain; in this case at $x = 0$ and $x = 1$. Note that four boundary conditions are specified because the highest derivative occurring in the equation is four.

2. This problem is an ODE because the differential equation only involves ordinary derivatives. It is an IVP because its auxiliary conditions are given at the same initial point.

3. This problem is a PDE because the differential equation involves partial derivatives. It is a BVP because the auxiliary conditions are given at the boundary of the domain and there is no time dependence in the equation.

4. This problem is a PDE because the differential equation involves partial derivatives. It is an IBVP in one spatial dimension because it has the boundary conditions $u(0, t) = \cos \pi t$ and $u(2, t) = 9$ imposed and the initial condition $u(x, 0) = x^3 + 1$. Note that at $x = 0$ the initial condition is $u(0, 0) = 0 + 1 = 1$ which agrees with the initial condition at $x = 0$, i.e., $u(0, 0) = \cos 0 = 1$. Also at $x = 2$ the initial condition is $x^3 + 1 = 9$ which agrees with the boundary condition there.

1.1.2 Terminology

Much of the terminology for ODEs and PDEs is the same because it is terminology used for differential equations in general. This terminology is useful because it can simplify the description of a problem.

The **order** of a differential equation is the highest degree derivative which appears in the equation. For example, we classify (1.1) as a first order equation and (1.2), (1.3) as second order equations.

A differential equation is said to be **homogeneous** if the right hand side is zero when all terms involving the unknown(s) are moved to the left hand side of the equation. For example, (1.3) is homogeneous whereas (1.2) is not. We also refer to boundary conditions as being homogeneous. This simply means that we are setting the boundary condition to be zero; it doesn't matter whether we are setting the unknown or its derivative to zero, the term still applies.

An important distinction in differential equations is whether the equation is **linear** or **nonlinear**. Recall that a single linear algebraic equation (such as $x+7=4$) is trivial to solve and a system of linear algebraic equations is straightforward to solve by a technique such as Gauss elimination. However, a nonlinear algebraic equation (such as $\sin x = 2x^2$) or a system of nonlinear algebraic equations is more difficult to solve and requires an iterative method such as Newton's method. The same difficulty holds for differential equations so it is important to recognize whether the equation is linear or nonlinear. A differential equation is nonlinear if the *unknown* or its derivatives appear nonlinearly; otherwise it is linear. Nonlinear terms include terms like u^3, e^u or $(u_x)^2$. It is also nonlinear if there is a term involving the product of the unknown and its derivative such as the term uu_x.

There is some specific terminology that is used with boundary conditions. When we specify the value of the function on the boundary this type of constraint is called a **Dirichlet**[1] **boundary condition**. A second type of boundary condition is to specify a derivative of the unknown function on the boundary; this type of constraint is called a **Neumann**[2] **boundary condition**. A third type of boundary condition is to specify a weighted combination of the function value and its derivative at the boundary; this is called a **Robin**[3] **boundary condition** or **mixed boundary condition**.

Using common terminology assists us in describing a problem succinctly. For example, suppose we say that we are considering a second order linear BVP in one dimension with homogeneous Dirichlet boundary data. We know that this is a linear ODE with the second derivative the highest derivative occurring in the equation and that we are setting the unknown to be zero at the two endpoints of the interval.

Example 1.2. Classification of DEs by order and linearity; classification of boundary conditions

[1] Named after the German mathematician Gustav Lejeune Dirichlet (1805-1859)
[2] Named after the German mathematician Carl Neumann (1832-1925)
[3] Named after the French mathematician Victor Gustave Robin (1855-1897)

For each of the problems in Example 1.1 determine (i) the order of the differential equation, (ii) whether it is linear or nonlinear, and (iii) classify each boundary condition.

1. This differential equation is fourth order and linear. The first set of boundary conditions $u(0) = u(1) = 1$ are Dirichlet whereas the other two are Neumann.
2. This differential equation is a linear second order equation and because it is an IVP there are no boundary conditions applied.
3. This differential equation is a linear second order equation with homogeneous Dirichlet boundary conditions imposed at the left and right of the domain and inhomogeneous Neumann conditions at the top and bottom.
4. This differential equation is second order and it is nonlinear due to the uu_x term. Inhomogeneous Dirichlet boundary conditions are imposed.

1.1.3 Classification of PDEs

Second order PDEs are classified into three broad types. It is important to be able to recognize the type of an equation because it will indicate a lot about how to solve it.

A second order linear PDE in two independent variables (ξ, η) has the general form

$$au_{\xi\xi} + bu_{\xi\eta} + cu_{\eta\eta} + du_{\xi} + eu_{\eta} + gu = f(\xi, \eta) \tag{1.5}$$

where a, b, c, d, e, g are given coefficients (they can be constants or functions of (ξ, η)) and $f(\xi, \eta)$ is a given right hand side or source term. The equation (1.5) is classified as **elliptic**, **parabolic**, or **hyperbolic** in a domain based upon the sign of the discriminant $b^2 - 4ac$. The equation is classified as

elliptic if $b^2 - 4ac < 0$ for all points in the domain

parabolic if $b^2 - 4ac = 0$ for all points in the domain

hyperbolic if $b^2 - 4ac > 0$ for all points in the domain

The names elliptic, parabolic, and hyperbolic come from the classification of a conic section $ax^2 + bxy + cy^2 + dx + ey + g$ which is classified as elliptic if $b^2 - 4ac < 0$, etc. For example, for the unit circle $x^2 + y^2 = 1$ we have $b^2 - 4ac = -4 < 0$ so it is elliptic.

Different kinds of phenomena are modeled by each type of equation. Elliptic equations model states of equilibrium and thus do not include time evolution. Elliptic problems are BVPs. Parabolic and hyperbolic equations involve time evolution as well as spatial dependence and so they both lead to IBVPs.

1.1.4 Systems of differential equations

Oftentimes when we study some real-world process there is more than one unknown which is involved. For example, in a predator-prey model there are at least two unknowns (the prey and predator populations) and in a complex model such as weather forecasting there are typically many quantities of interest. Such models often lead to a system of differential equations. If the equations are *uncoupled* as in the system of IVPs

$$\begin{aligned} u'(t) &= t^3 e^{-u(t)} \quad t > t_0 \\ v'(t) &= 2v(t) \quad t > t_0 \\ w'(t) &= w^2(t) \quad t > t_0 \end{aligned}$$

then we can simply solve each equation separately along with its given auxiliary condition. The equations are uncoupled because each differential equation only involves one unknown. However, in most models the equations are *coupled*. This means that the equation for one unknown also includes terms involving one or more of the remaining unknowns. The following predator-prey equations are an example of a coupled system of IVPs:

$$\begin{aligned} p'(t) &= p(\alpha - \beta q) \quad t > t_0 \\ q'(t) &= -q(\gamma - \delta p) \quad t > t_0 \,, \end{aligned}$$

where $p(t)$ represents the number of prey at time t, $q(t)$ represents the quantity of predator at time t and α, β, γ and δ are parameters governing the interaction of the two species. Of course, initial conditions for both p and q must be specified at the same point t_0.

Oftentimes a model may be represented as a single equation but it is really a system because the unknown is a vector or perhaps a complex variable which has both a real and imaginary part. Also a system may consist of a combination of PDEs and ODEs.

Typically we will be able to easily extend our numerical schemes for a single differential equation to a system. Consequently in the sequel we will first discuss methods for solving a single differential equation before considering systems.

1.2 Discretization

For most differential equations it is impossible to find a closed form solution. Even if a solution can be found, it is often in terms of an infinite series which must be truncated to approximate the solution. In addition, even when an analytic solution is available, a modification of the domain or a change from constant coefficients in the equation to variable coefficients renders an analytic solution unattainable. For these reasons, it is important to study numerical techniques for approximating the solution of differential equations.

If we use numerical methods to approximate the solution of a differential equation, we have to give up the idea of finding a solution which holds for all points

in the domain. Instead, we typically overlay the domain with a finite number of points and use a method for finding an approximation to the exact solution of the differential equation at each of the discrete points. Our hope is that as we add more and more points in a uniform manner the approximate solution gets closer to the exact solution.

1.2.1 Discretizing the domain

Typically, the first step in approximating the solution of a differential equation is to overlay the domain with a **grid** or **mesh**. If the problem is an IVP then the domain is usually the time interval $[t_0, T]$ so we add points t_i, $i = 0, 1, N$ which subdivide the domain into intervals for a total of $N + 1$ points and N subintervals. If these subintervals are the same size, say $\Delta t = (T - t_0)/N$ then we call the subdivision **uniform**; otherwise it is **nonuniform**. For an IVP the increment Δt is typically called the **time step**.

For BVPs the spatial dimension and domain dictate what type of grids can be used. For example, for an ODE BVP we know that the domain is an interval, say $[a, b]$, so we overlay the domain with a grid using $N + 1$ points

$$x_0 = a, \quad x_1 = x_0 + \Delta x_1, \quad \cdots \quad x_i = x_{i-1} + \Delta x_i, \quad \cdots \quad x_N = b\,.$$

Here Δx_i is the grid spacing and if $\Delta x_i = \Delta x$ for all $i = 1, \ldots, N$ then the grid is uniform. The $N + 1$ points x_i are called the **grid points** or **nodes** of the mesh which divide the domain into N **cells** or **elements**. If the BVP is a PDE then the spatial dimension is typically two or three. If the domain in $\mathbb{R}^2$ is a rectangle then we can overlay it with a Cartesian grid, i.e., with lines parallel to the coordinate axes which results in dividing the domain into rectangles. However, if the domain is a circle, a triangle, or some complicated polygon then this is not a viable approach. For a general polygon it is often advantageous to use a triangular mesh whereas to completely cover a circle we need elements which have curved sides. In $\mathbb{R}^3$ the grid can be even more complicated. For example, if the domain is a model of a car or an airplane then clearly obtaining grids is an involved process. In addition, it is important to be able to automatically *refine* the grid, i.e., to decrease the discretization spacing in a uniform manner. To complicate matters further, for complex problems it is often advantageous to have an adaptive grid which changes over time.

There are many available software packages to automatically discretize a domain. Also software for solving PDEs often contains capabilities for automatically generating grids. The area of generating efficient grids is a vast one and the interested reader is referred to the literature. For the most part, we will be satisfied with simple Cartesian grids.

1.2.2 Approximating the differential equation

The one thing that we know how to do well on a computer is to solve algebraic equations so our goal is to approximate the differential equation(s) in such a way that

we get either a single or a system of algebraic equations to solve. Of course, if the original differential equations are nonlinear, the approximation will typically result in a system of nonlinear algebraic equations but these can be solved by schemes such as the Newton-Raphson method, quasi-Newton methods, etc.

There are two major approaches to turn the differential equation into algebraic equations. The first approach is to approximate the *derivatives* in the equation by **difference quotients** which just contain differences in function values. The second approach is to approximate the *solution* by simpler functions such as polynomials or trigonometric functions. Of course, these two approaches are related but this distinction gives a useful way to categorize methods.

To compare these two approaches consider the simple IVP $y'(t) = \sin t$, $y(0) = 2$, $0 < t \leq 1$. Assume we have discretized $[0, 1]$ into N subintervals of length Δt using the grid points $t_0, t_1, \ldots, t_N$. Using the definition of derivative

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h} \,,$$

we approximate $y'(t_i)$ by the difference quotient

$$y'(t_i) \approx \frac{y(t_{i+1}) - y(t_i)}{\Delta t} \,.$$

The idea is to write an algebraic equation which approximates the solution at t_1 given the known initial condition at t_0. Once this approximation is computed it is used to approximate the solution at t_2 and the process is continued until the final time is reached.

Now $y(t_n)$ represents the exact solution at time t_n so we need new notation to represent the approximation there. To this end, let Y^n denote the approximation to $y(t_n)$. Then we use the differential equation at t_0, i.e., $y'(t_0) = \sin t_0 = 0$, and the approximation

$$y'(t_0) \approx \frac{y(t_1) - y(t_0)}{\Delta t}$$

to get

$$\frac{Y^1 - Y^0}{\Delta t} = \sin(t_0) \Rightarrow Y^1 = Y^0 + \Delta t \sin(t_0) \,.$$

This is called a **difference equation** because it involves differences in function values. The next step is to write the difference equation at t_1 and get the approximation Y^2; we have

$$\frac{Y^2 - Y^1}{\Delta t} = \sin(t_1) \Rightarrow Y^2 = Y^1 + \Delta t \sin(t_1) \,.$$

Because we have an IVP which consists of a single equation this approach yields a single algebraic equation. However, if we have a system of IVPs we get a system of algebraic equations.

In Chapter **??** we will see that discretizing a BVP even in one spatial dimension leads to a *system* of algebraic equations. The general technique of approximating derivatives by difference quotients is called the **finite difference method**.

The second approach is to approximate the unknown solution by simpler functions such as polynomials or trigonometric functions. How does this get rid of the derivatives in the equation? Suppose we have quadratic polynomials $\phi_i(x)$, $i = 1, \ldots, N$ that we want to use to approximate the solution, say $u(x)$. We write $u(x) \approx \sum_{i=1}^{N} c_i \phi_i(x)$ and so $u'(x) \approx \sum_{i=1}^{N} c_i \phi_i'(x)$. But each $\phi(x)$ is a given quadratic polynomial so we just differentiate it to get a linear polynomial. The unknowns in the equation are the c_i, $i = 1, \ldots, N$ and we get a system of algebraic equations for these unknowns. The way this approach is implemented is slightly different from the way described here and it will be discussed in detail in Chapter **??**. Common methods such as finite element methods, spectral methods, wavelet-Galerkin methods, etc. take this approach of approximating the solution rather than replacing the derivative with difference quotients.

We will spend a considerable effort understanding how to derive various schemes for approximating the solution of differential equations rather than treating the schemes as "black boxes". In addition, we will be concerned with how fast a numerical approximation approaches the exact solution as the mesh spacing is decreased uniformly. In many cases we will prove this theoretical rate which can be used to help validate our code.

1.2.3 Numerical computations

Once we have chosen an appropriate scheme to approximate the solution of a problem, we must implement the method efficiently and verify that the computer code is producing accurate results for a problem where the exact solution is known. Once this process is complete, then the code can be used with confidence for a similar problem where the exact solution is not known. It is important to realize that numerical results for one choice of the mesh spacing are somewhat meaningless. We need to perform several approximations for decreasing values of the mesh spacing and demonstrate that the approximate solutions are converging.

The theoretical rates of convergence that we derive will be of the form $C(\Delta t^r)$ for temporal approximations and $C(h^r)$ for spatial approximations where h is a measure of the spatial discretization. The notation that we will use is $\mathcal{O}(\Delta t^r)$ or equivalently $\mathcal{O}(h^r)$. This "big O" notation just means a constant times that power of Δt or h. If $r = 1$ this says that the error is a constant times the mesh spacing so the convergence is *linear*. So if we half the spacing then we would expect the error to be approximately half the previous error. If $r = 2$ then we say the method converges *quadratically* and if we half the mesh spacing we expect the error to be reduced by approximately four. Of course, the higher r is, the faster the method converges to the solution. The constant C typically involves higher derivatives of the solution.

For a fixed value of the mesh spacing we don't expect the error in the actual solution and the approximate solution to obey exactly the theoretical rate. However, as the mesh spacing goes to zero uniformly we expect it to get closer to that rate. To verify this, we compute a sequence of approximations for decreasing mesh spacing for a problem where the exact solution is known and then compute the actual error

for each choice of the mesh spacing. We use these results to calculate a sequence of numerical rates which should converge to the theoretical rate.

To determine the actual numerical rate we compute the numerical error using step size h_1. Assume that we know that theoretically the error, E_1 should satisfy $E_1 = Ch_1^r$. We can't solve for r from this equation because we don't know C. However, if we look at two calculations

$$E_1 = Ch_1^r \quad \text{and} \quad E_2 = Ch_2^r$$

then we can solve for r from these. Assuming the constant C is fixed, we have

$$\frac{E_1}{h_1^r} = \frac{E_2}{h_2^r} \Rightarrow \frac{E_1}{E_2} = \left(\frac{h_1}{h_2}\right)^r .$$

Using properties of logarithms we get

$$r = \frac{\ln \frac{E_1}{E_2}}{\ln \frac{h_1}{h_2}} . \tag{1.6}$$

If the spacing is halved this gives the formula

$$r = \frac{\ln \frac{E_1}{E_2}}{\ln 2} .$$

We will use (1.6) to calculate the numerical rate of convergence for the computational examples in subsequent chapters.

When we test numerical methods we apply the scheme to a problem for which the analytic solution is known. For simple differential equations we can use standard techniques to generate the analytic solution. However, in many cases, we may not be able to solve a problem analytically; in fact, this is why we use numerical methods. An approach to finding a test problem which avoids explicitly solving the problem is called the **method of manufactured solutions**. In this method we begin by choosing a function which satisfies the desired initial and/or boundary conditions and then plug it into the given differential equation to get the right hand side. The following examples illustrate this technique.

Example 1.3. **Method of manufactured solutions for an IVP**

We want to use the method of manufactured solutions to find an IVP of the following form

$$y'(t) = f(t, y), \quad 0 \le t \le 1 \qquad y(0) = 1 .$$

This means that we want to generate $f(t, y)$ and have an exact solution $y(t)$ to this IVP which has an initial value of one.

We begin by choosing a function $y(t)$ which satisfies the initial condition $y(0) = 1$ which is the only auxiliary condition for this problem. There are an infinite number of choices but for specificity we choose $y(t) = \cos \pi t$. Then $y'(t) = -\pi \sin \pi t$ so the model problem is

$$y'(t) = -\pi \sin \pi t \quad 0 \le t \le 1 \qquad y(0) = 1$$

whose solution is $y(t) = \cos \pi t$.

Example 1.4. **Method of manufactured solutions for an IBVP**
We want to use the method of manufactured solutions to find an IBVP which has the following form for $u = u(x,t)$

$$u_t - u_{xx} - u^2 = f(x,t), \quad 0 \le t \le T, \quad 0 < x < 1$$
$$u(x,0) = g(x), \quad u(0,t) = 1, \quad u(1,t) = \sin t\,.$$

We begin by choosing a function $u(x,t)$ which satisfies the two boundary conditions. One choice is $u(x,t) = x\sin t + 1 - x^2$. To get the initial condition we set $t = 0$ and thus $g(x) = 1 - x^2$. Finally, we substitute $u = x\sin t + 1 - x^2$ into the differential equation to generate $f(x,t)$

$$\begin{aligned} f(x,t) = u_t - u_{xx} - u^2 &= \big[x\cos t\big] - \big[-2\big] - \big[x\sin t + 1 - x^2\big]^2 \\ &= x\cos t + 2 + x^2\sin^2 t + (1-x^2)^2 + 2x(1-x^2)\sin t\,. \end{aligned}$$

Thus we have the analytic solution $u(x,t) = x\sin t + 1 - x^2$ to the IBVP

$$u_t - u_{xx} - u^2 = x\cos t + 2 + x^2\sin^2 t + (1-x^2)^2 + 2x(1-x^2)\sin t \quad 0 \le t \le T, \quad 0 < x < 1$$
$$u(x,0) = 1 - x^2, \quad u(0,t) = 1, \quad u(1,t) = \sin t\,.$$

1.3 Strategy for studying numerical differential equations

The strategy that we take for studying numerical differential equations is to divide this broad area into three parts. Initially, we address the solution of IVPs; we will only consider ODEs here. Next, we look at BVPs which consist of both ODEs and PDEs. Finally we combine the results we learned from IVPs and BVPs to solve IBVPs which are PDEs.

For each type of problem we will identify prototype equation(s). Of course we only want to approximate the solution of problems which have a unique solution so for each prototype problem we will state basic results guaranteeing existence and uniqueness of the solution.

We will spend considerable time demonstrating how numerical methods for these prototype equations are derived. This will give us insight into how the methods arise and how new methods might be derived. Once we have derived a numerical method we want to determine its theoretical accuracy. This will tell us how the error should decrease as the mesh size is decreased. Then we need to implement the method efficiently on a computer and be able to verify that the code is working properly by comparing the numerical rate of convergence with the theoretical rate.

There is not one numerical scheme that is the method of choice for each prototype equation. Rather, it depends upon many factors such as accuracy desired, amount of computing power available, shape of the domain, etc. We will look at families of methods for each type of equation and discuss the pros and cons of each.

Not all numerical methods we derive will produce numerically reliable results for all choices of the mesh size. We must investigate this concept of numerical instability carefully so that we can choose an appropriate numerical scheme.