COMPUTATIONAL METHODS FOR ENGINEERING APPLICATIONS

Lecture Notes by Prof. Dr. Siddhartha Mishra

January 25, 2019

Contents

1 Introduction

Many fundamental problems in engineering involve the study of physical quantities that evolve with time or change with spatial position. Such relationships can be expressed mathematically as *differential equations*. The aim of this course is to present numerical (computational) methods for discretising i.e. approximating and simulating differential equations that arise in engineering.

1.1 What is a Differential Equation?

Let $d \in \{1, 2, 3\}$ and let $x \in \mathbb{R}^d$ denote the spatial position of our system. Thus depending on the dimension of our problem, x could be a scalar or a vector. In general we may write $x = \{x_i\}_{i=1,2,3}$. Let $T \in (0, \infty)$ denote some fixed, final time and let $t \in [0, T]$ denote the time variable. Let $m \in \{1, 2, \ldots\}$ and let $u = u(x, t) \colon \mathbb{R}^d \times [0, T] \to \mathbb{R}^m$ be a function. Thus once again depending on our problem, u could be a scalar i.e. $u(x, t) \in \mathbb{R}$ or u could be a vector i.e. $u(x, t) \in \mathbb{R}^m$ for some $m > 1$.

Let

$$u_t = \frac{\partial u}{\partial t}, \quad u_{tt} = \frac{\partial^2 u}{\partial t^2}, \quad \ldots, \quad \underbrace{u_{tt\ldots t}}_{k \text{ times}} = \frac{\partial^k u}{\partial t^k}, \quad \ldots$$

$$u_{x_i} = \frac{\partial u}{\partial x_i}, \quad u_{x_i x_j} = \frac{\partial^2 u}{\partial x_i \partial x_j}, \quad \ldots$$

denote the partial derivatives of u. Then a **differential equation** is an equation of the form

$$\mathbb{F}\big(x, t, u, u_t, u_{x_i}, u_{tt}, u_{x_i x_j}, \ldots\big) = 0, \tag{1.1}$$

where $\mathbb{F}$ is some general, known function.

Equation (1.1) implies that the function u and its partial derivatives are related in some way. Therefore, the task of solving Equation (1.1) amounts

to finding the unknown function u, given information on the relation between its partial derivatives. Of course, the abstract form of the differential equation (1.1) is not very instructive, and it is more helpful to consider concrete cases of differential equations.

1.2 Ordinary Differential Equations (ODEs)

ODEs are the simplest type of differential equations. Essentially, we assume that the unknown of interest u is a function of exactly one variable (say time). More formally, we assume that $u = u(t)\colon [0, T] \to \mathbb{R}^m$ where $m = 1$ if u is a scalar-valued function and $m > 1$ if u is a vector-valued function. In either case equation (1.1) reduces to

$$\mathbb{F}\big(t, u, u_t, u_{tt}, \dots\big) = 0. \tag{1.2}$$

Thus, the task of solving Equation (1.2) amounts to finding the unknown function u, given information on the relation between its *derivatives*.

It is instructive to consider a simple example of Equation (1.2).

Example 1.1 *Consider the ordinary differential equation given by*

$$u_t = 1. \tag{1.3}$$

Clearly, all solutions of (1.3) *are of the form*

$$u(t) = t + C,$$

where $C \in \mathbb{R}$ is some fixed constant. Therefore, in order to find a unique solution to (1.3)*, it is necessary to specify a value for the constant C. A natural way to do so is to set the value of C equal to the initial value of the function u, i.e., by specifying*

$$u(0) = C.$$

This leads us to the extremely important concept of initial value problems (IVPs).

1.3 Initial Value Problems (IVPs) for ODEs

Let

$$\begin{aligned} u' &= u_t, \\ u'' &= u_{tt}, \\ &\vdots \\ u^{(k)} &= \underbrace{u_{tt\ldots t}}_{k \text{ times}} \end{aligned}$$

denote the first-order, second-order and higher order time derivatives of the function u. Then, the general form of an initial value problem (IVP) for an ODE is given by

$$\begin{aligned} \mathbb{F}\big(t, u, u', u'', u^{(3)}, \ldots, u^{(k)}\big) &= 0, \\ u(0) &= u_0, \\ u'(0) &= u_1, \\ &\vdots \\ u^{(k-1)}(0) &= u_{k-1}. \end{aligned} \tag{1.4}$$

We observe that solving the IVP (1.4) amounts to finding a function $u = u(t)$ given a relation between its time derivatives and their initial values.

Initial value problems for ordinary differential equations arise in a wide variety of models in engineering. We give a few prototypical examples in the next subsections.

1.3.1 Single Body Dynamics

Consider a single particle of mass $m = 1$ with position $x = x(t)$ and velocity $v = v(t)$ at time t. Then, the trajectory of the particle is given by

$$\begin{aligned} x'(t) &= v(t), && \text{(Definition of the velocity)} \\ v'(t) &= f, && \text{(Newton's Second Law)} \end{aligned}$$

where f is the force acting on the particle. Of course, the force $f = f(t, u(t))$ needs to be specified and for simplicity we consider a given external force $f = f(t)$. The single body dynamics are then completely specified by the IVP

$$\begin{aligned} x'(t) &= v(t), & x(0) &= x_0, \\ v'(t) &= f, & v(0) &= v_0. \end{aligned} \tag{1.5}$$

Here, x_0 and v_0 are the initial position and initial velocity of the particle respectively. It is possible to rewrite the IVP (1.5) in a slightly different form. Indeed, let

$$U = [x, v],$$
$$F = [v, f].$$

Then the IVP (1.5) can be rewritten in the compact form

$$U' = F(t, U), \qquad U(0) = [x_0, v_0]. \tag{1.6}$$

Note that U is a vector and the force $f = f(t)$.

1.3.2 N-Body Dynamics

A more general situation arises if we consider N bodies with masses $(m_1, \ldots, m_N)$, positions $(x_1, \ldots, x_N)$ and velocities $(v_1, \ldots, v_N)$. Then, applying Newton's Second Law to this collection of N bodies results in the following system of $2N$ equations:

$$\begin{aligned} x_i' &= v_i, \qquad i = 1, 2, \ldots, N, \\ m_i v_i' &= f_i, \qquad i = 1, 2, \ldots, N. \end{aligned}$$

We assume here that the only force acting on the particles is gravity and furthermore that we are in three spatial dimensions. Then by Newton's Law of Gravitation for all $i = 1, 2, \ldots, N$, it holds that the force of attraction f_i exerted on the i^{th} particle by the other $N - 1$ particles is given by

$$f_i = \sum_{i=1,\ j \neq i}^{N} \frac{G m_i m_j (x_i - x_j)}{|x_i - x_j|^3},$$

where G is the Gravitational constant. Next, let

$$U = [x_1, \ldots, x_N, v_1, \ldots, v_N],$$
$$F = [v_1, \ldots, v_N, f_1, \ldots, f_N],$$

and we again obtain an IVP of the form (1.6)

$$U'(t) = F(U(t)), \qquad U(0) = [x_1^0, \ldots, x_N^0, v_1^0, \ldots v_N^0]. \tag{1.7}$$

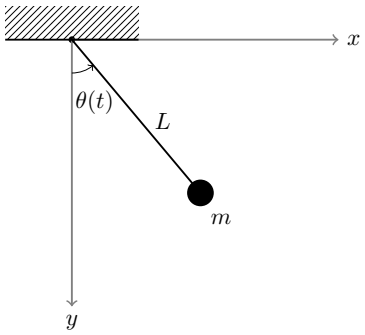


Figure 1.1: An example of a simple pendulum

1.3.3 A Simple Pendulum

Consider a pendulum of mass m at the end of a rigid, massless bar of length L (see Figure 1.1). The motion of the pendulum is described in terms of the angle $\theta(t)$ and can be derived using the Newton's Second Law applied to curvilinear motion:

$$\theta''(t) = -\frac{g}{L}\sin\big(\theta(t)\big).$$

For simplicity, we may assume that $L = g$ and we therefore obtain the pendulum equation

$$\begin{aligned} \theta''(t) &= -\sin\big(\theta(t)\big), \\ \theta(0) = \theta_0, \qquad \theta'(0) &= v_0. \end{aligned} \tag{1.8}$$

Exercise 1.2 *Show that for small values of the angle, $\theta(t) \ll 1$, the IVP* (1.8) *can be approximated by*

$$\begin{aligned} \theta''(t) &= -\theta(t), \\ \theta(0) = \theta_0, \qquad \theta'(0) &= v_0. \end{aligned}$$

Note that in order to solve the IVP (1.8), it is necessary to specify the initial angle $\theta(0)$ and the initial angular velocity $\theta'(0)$.

1.3.4 A Population (Concentration) Model from Cell Biology

Proteins are functional units of living cells. New proteins in cells are manufactured by the sequential processes of transcription and translation:

- In the process of transcription, a gene is expressed in terms of *messenger RNA* (mRNA).

- In the process of translation, the mRNA is used to synthesis proteins.

We consider a simple mathematical model of protein biosynthesis. Let $X = X(t)$ denote the concentration of a gene (X), let $Y = Y(t)$ denote the concentration of mRNA (Y) and let $Z = Z(t)$ denote the concentration of protein (Z). Then we have the following reaction scheme:

$$X \xrightarrow{transcription} Y \xrightarrow{translation} Z.$$

In the simplest mathematical model, the gene concentration is given a priori and the mRNA concentration satisfies the following ODE:

$$Y'(t) = F_1(X) - \alpha_1 Y(t), \tag{1.9}$$

where $\alpha_1 > 0$ is a decay constant and the function F_1 models the production of the mRNA as a function of the gene concentration X and is given by

$$F_1(X) = \frac{\beta_1 X^n}{K_1^n + X^n}.$$

Here, $n \in \mathbb{N}$ and $\beta_1 > 0$ and $K_1 > 0$ are constants that characterise the mRNA production rate and the mRNA production threshold respectively. We also remark that for any natural number $n \in \mathbb{N}$ the function f_n given by

$$f_n(x) = \frac{x^n}{k_1^n + x^n} \tag{1.10}$$

is known as a n-Hill function. This function is widely used in biochemistry to model the binding of ligands such as enzymes to macromolecules and usually the values $n = 4$ or $n = 8$ are chosen. Examples of n-Hill functions for some values of n are displayed in Figure 1.2.

Similarly, the kinetics of the protein concentration is modelled by the following ODE:

$$Z'(t) = F_2(Y) - \alpha_2 Z(t), \tag{1.11}$$

with α_2 a decay constant and the function F_2 given by

$$F_2(Y) = \frac{\gamma \beta_2 Y^n}{K_2^n + Y^n},$$

where $\beta_2 > 0$ is the maximum protein production rate, $K_2 > 0$ is the protein activation threshold and $\gamma > 0$ is the transcription rate.

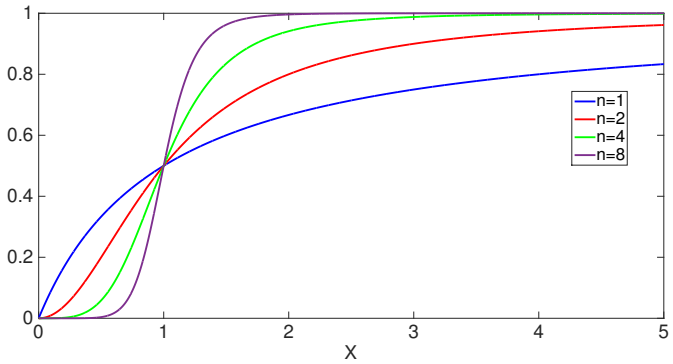


Figure 1.2: Examples of n-Hill functions for various values of n.

We may combine (1.9) and (1.11) into a more compact form in the following manner. Let

$$U = [Y, Z],$$

$$F = \left[\frac{\beta_1 X^n}{K_1^n + X^n} - \alpha_1 Y, \frac{\gamma\beta_2 Y^n}{K_2^n + Y^n} - \alpha_2 Z\right],$$

and we again obtain an IVP of the form (1.6)

$$U'(t) = F(t, U(t)), \qquad U(0) = [Y_0, Z_0].$$

1.3.5 Models of Chemical Kinetics

EXAMPLE 1

We consider the following set of chemical reactions:

$$A \xrightarrow{\text{rate of reaction } k_1} B \xrightarrow{\text{rate of reaction } k_2} C.$$

Here, A, B and C are chemical reactants with concentrations u_1, u_2 and u_3 respectively. We assume that the chemical reactant A decays to the chemical reactant B at the constant rate $k_1 > 0$ and the chemical reactant B decays to the chemical reactant C at the rate $k_2 > 0$. Therefore, our model satisfies the following system of ODEs:

$$\begin{aligned} u_1'(t) &= -k_1 u_1(t), \\ u_2'(t) &= k_1 u_1(t) - k_2 u_2(t), \\ u_3'(t) &= k_2 u_2(t). \end{aligned} \tag{1.12}$$

Let $U = [u_1, u_2, u_3]$ and let the 3×3 matrix A be given by

$$A = \begin{bmatrix} -k_1 & 0 & 0 \\ k_1 & -k_2 & 0 \\ 0 & k_2 & 0 \end{bmatrix}.$$

Then we obtain the following IVP for the chemical kinetics of this set of chemical reactions:

$$\begin{aligned} U'(t) &= F(U(t)) = AU(t), \\ U(0) &= [u_1^0, u_2^0, u_3^0]. \end{aligned} \tag{1.13}$$

EXAMPLE 2: Compound Formation

We next consider the following chemical reaction:

$$A + B \underset{k_2}{\overset{k_1}{\rightleftharpoons}} AB$$

Here, A and B are chemical reactants that react together to produce the chemical compound AB. Let A, B and AB have concentrations u_1, u_2 and u_3 respectively and assume that the reactants A and B combine to form the compound AB at the rate k_1 and the compound AB breaks down into the reactants A and B at the rate k_2. Our model then satisfies the following system of ODEs:

$$\begin{aligned} u_1'(t) &= -k_1 u_1(t) u_2(t) + k_2 u_3(t), \\ u_2'(t) &= -k_1 u_1(t) u_2(t) + k_2 u_3(t), \\ u_3'(t) &= \quad k_1 u_1(t) u_2(t) - k_2 u_3(t). \end{aligned} \tag{1.14}$$

Let $U = [u_1, u_2, u_3]$ and let the function F be given by

$$F(U) = \Big[k_2 u_3 - k_1 u_1 u_2,\ k_2 u_3 - k_1 u_1 u_2,\ k_1 u_1 u_2 - k_2 u_3\Big].$$

Then we obtain the following IVP for the chemical kinetics of this chemical reaction:

$$\begin{aligned} U'(t) &= F(U(t)), \\ U(0) &= [u_1^0, u_2^0, u_3^0]. \end{aligned} \tag{1.15}$$

1.4 Generic Form of IVPs for ODEs

The examples of Section 1.3, chosen from different fields, suggest that the generic form of an IVP

$$\begin{aligned} U'(t) &= F\big(t, U(t)\big), \\ U(0) &= u_0, \end{aligned} \tag{1.16}$$

represents nearly all of them. Of course this is not strictly true for the case of the simple pendulum in Section 1.3.3. We recall that the motion of a simple pendulum is described by the IVP given by

$$\begin{aligned} \theta''(t) &= -\sin\big(\theta(t)\big), \\ \theta(0) = \theta_0, &\quad \theta'(0) = v_0. \end{aligned} \tag{1.17}$$

It turns out that a simple change of variables in Equation (1.17) can cast it in the generic form of Equation (1.16). To this end, let $v(t) = \theta'(t)$. Then it holds that

$$\begin{aligned} \theta'(t) &= v(t), \\ v'(t) &= -\sin\big(\theta(t)\big), \\ \theta(0) = \theta_0, &\quad v(0) = v_0. \end{aligned} \tag{1.18}$$

Therefore introducing the vectors

$$\begin{aligned} U &= [\theta, v], \\ F &= [v, -\sin(\theta)], \end{aligned}$$

we obtain the following IVP:

$$\begin{aligned} U'(t) &= F\big(t, U(t)\big), \\ U(0) &= u_0 \end{aligned} \tag{1.19}$$

and clearly, Equation (1.19) is of the generic form (1.16).

Equation (1.16) is known as the *first-order form* of the initial value problem for ODEs. In general any initial value problem (IVP) for an ODE can be recast in the first-order form (1.16). Let us consider a second simple example.

Example 1.3 *Consider the following initial value problem:*

$$\begin{aligned} u^{(4)}(t) &= 3u'''(t)u'(t) + 2\big(u''(t)\big)^3 - \sin(t), \\ u(0) = x_0, &\quad u'(0) = x_1, \quad u''(0) = x_3, \quad u'''(0) = x_4. \end{aligned} \tag{1.20}$$

We introduce auxiliary variables given by

$$\begin{aligned} u_1(t) &= u(t), \quad & u_2(t) &= u'(t), \\ u_3(t) &= u''(t), \quad & u_4(t) &= u'''(t). \end{aligned}$$

Let the vectors U and F be given by

$$\begin{aligned} U &= [u_1, u_2, u_3, u_4], \\ F &= [u_2, u_3, u_4,\ 3u_2u_4 - 2u_3^2 - \sin(t)] \end{aligned}$$

Then the IVP (1.20) *can be rewritten as a first-order IVP given by*

$$\begin{aligned} U'(t) &= F\big(t, U(t)\big), \\ U(0) &= [x_0, x_1, x_2, x_3]. \end{aligned}$$

1.5 Types of ODEs

In this section we describe different classifications of ordinary differential equations (ODEs).

- **Autonomous and Non-Autonomous ODEs**

Consider a first order ODE of the form (1.16). This ordinary differential equation is termed as *autonomous* if for all $t \in [0, T]$ it holds that $F(t, U(t)) \equiv F(U(t))$ i.e., the right hand side of Equation (1.16) does not depend explicitly on time. Thus, the IVP for an autonomous ODE is given by

$$\begin{aligned} U'(t) &= F\big(U(t)\big), \\ U(0) &= u_0. \end{aligned} \tag{1.21}$$

An ODE which is not autonomous is termed non-autonomous. Examples of autonomous ODEs include N-body dynamics (Section 1.3.2), the pendulum equation (Section 1.3.3) and the models of chemical kinetics (Section 1.3.5). On the other hand, examples of non-autonomous ODEs include the protein biosynthesis model (Section 1.3.4) and the model ODE (1.20).

Interestingly, a simple trick allows us to convert any non-autonomous ODE into an autonomous ODE.

Algorithm for converting non-autonomous ODEs: Consider a non-autonomous IVP of the form (1.16) with $u = [u_1, u_2, \ldots, u_m] \in \mathbb{R}^m$ and $F = [F_1, F_2, \ldots, F_m] \in \mathbb{R}^m$. Let the function u_{m+1} be given by $u_{m+1}(t) = t$ and let the vectors W and G be given by

$$\begin{aligned} W &= [u_1, u_2, \ldots, u_m, u_{m+1}], \\ G(W) &= \big[F_1(u_1, \ldots, u_{m+1}),\ F_2(u_1, \ldots, u_{m+1}), \ldots, F_m(u_1, \ldots, u_{m+1}), 1\big]. \end{aligned}$$

Then the IVP (1.16) can be rewritten as the following autonomous IVP:

$$\begin{aligned} W'(t) &= G(W(t)), \\ W(0) &= [u_1(0), u_2(0), \ldots, u_m(0), 0]. \end{aligned} \tag{1.22}$$

To check this, it is sufficient to observe that $u'_{m+1}(t) = 1$ and $u_{m+1}(0) = 0$ together imply that $u_{m+1}(t) = t$, and therefore the autonomous system (1.22) is consistent with the non-autonomous system (1.16).

- **Scalar ODEs and Systems of ODEs**

Depending on the dimensions of the ODE, the unknown function of interest u in the IVP (1.16) could be scalar-valued or vector-valued. For instance, in the case of the pendulum equation (Section 1.3.3), it holds that $u \in \mathbb{R}$ and therefore Equation (1.8) is an example of a scalar ODE. On the other hand, many physical problems are modelled by systems of ODEs in which case it holds that $u \in \mathbb{R}^m$ for some $m > 1$. Examples of systems of ODEs include N-body dynamics (Section 1.3.2), the protein biosynthesis model (Section 1.3.4) and the models of chemical kinetics (Section 1.3.5). Furthermore, even the pendulum equation, written in the first-order form (1.19) is a system of ODEs.

- **Linear and Non-Linear ODEs**

The ODE (1.16) is termed *linear* if the right-hand side function F is linear in U. In particular, a sufficient condition for the linearity of the ODE (1.16) is that the function F can be written as

$$F(t, U(t)) = A(t)U(t) + C(t),$$

where $U \in \mathbb{R}^m$ is an m-dimensional vector, $A \in \mathbb{R}^{m \times m}$ is an $m \times m$ matrix and $C \in \mathbb{R}^m$ is an m-dimensional vector. Note that if $C(t) \equiv 0$, then the ODE (1.16) is termed *homogeneous* and any linear combination of solutions to such an ODE is also a solution to the ODE.

An ODE which is not linear is termed non-linear. Examples of linear ODEs include the models of chemical kinetics (Section 1.3.5). On the other hand, examples of non-linear ODEs include N-body dynamics (Section 1.3.2), the pendulum equation (Section 1.3.3) and the protein biosynthesis model (Section 1.3.4). We remark that most interesting phenomenon in nature are modelled by non-linear differential equations.

1.6 Explicit Solutions of ODEs

It is possible to obtain explicit solutions for a few ODEs. In this section, we consider some simple examples of such ODEs.

Example 1.4 (Linear Scalar ODE) *Consider a linear, scalar, autonomous IVP given by*

$$\begin{aligned} u'(t) &= \lambda u(t), \\ u(0) &= u_0, \end{aligned} \tag{1.23}$$

where $u \in \mathbb{R},\ u_0 \in \mathbb{R}$ and $\lambda \in \mathbb{R}$ is a constant.

The solution of the IVP (1.23) is then given by

$$u(t) = u_0 e^{\lambda t}. \tag{1.24}$$

Example 1.5 (Linear System of ODEs) *Consider the following IVP involving a linear system of ODEs:*

$$\begin{aligned} u'(t) &= Au(t), \\ u(0) &= u_0, \end{aligned} \tag{1.25}$$

where $u \in \mathbb{R}^m,\ u_0 \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times m}$ is a constant $m \times m$ matrix.

Assume that the matrix A is diagonalisable i.e., there exists a set of eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ and a set of eigenvectors $\{r_1, r_2, \ldots, r_m\}$ such that $\{r_i\}_{i=1}^m$ forms a basis of $\mathbb{R}^m$ and such that

$$A = R\Lambda R^{-1}, \tag{1.26}$$

where

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \lambda_m \end{pmatrix} = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$$

and

$$R = \left[\, r_1 \,\middle|\, r_2 \,\middle|\, \ldots \,\middle|\, r_m \,\right]$$

is the $m \times m$ matrix whose columns consist of the eigenvectors $\{r_i\}_{i=1}^m$.

Then it holds that

$$\begin{aligned} \left(R^{-1}u(t)\right)' &= R^{-1}(R\Lambda R^{-1})u(t) \\ &= \Lambda R^{-1}u(t). \end{aligned}$$

Next, let $w = R^{-1}u$. *Then the IVP* (1.25) *can be rewritten as*

$$\begin{aligned} w'(t) &= \Lambda w(t), \\ w(0) &= R^{-1}u_0. \end{aligned} \tag{1.27}$$

Now the point is that matrix Λ *is diagonal, and therefore the system* (1.27) *consists of an uncoupled set of* m*-scalar equations:*

$$w_i'(t) = \Lambda_i w_i, \qquad w_i(0) = R^{-1}u_{0i}, \qquad i = 1, \ldots, m.$$

Hence, for all $i = 1, 2, \ldots, m$ *it holds that*

$$w_i(t) = w_i(0)e^{\lambda_i t},$$

and therefore

$$w(t) = \begin{pmatrix} e^{\lambda_1 t} & 0 & \ldots & 0 \\ 0 & e^{\lambda_2 t} & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & e^{\lambda_m t} \end{pmatrix} \begin{bmatrix} w_1(0) \\ w_2(0) \\ \vdots \\ w_m(0) \end{bmatrix}.$$

Next, let $e^{\Lambda t} = diag(e^{\lambda_1 t}, \ldots, e^{\lambda_m t})$. *Then it follows that*

$$\begin{aligned} w(t) &= e^{\Lambda t}w(0) \\ &= e^{\Lambda t}R^{-1}u_0. \end{aligned}$$

And hence,

$$\begin{aligned} R^{-1}u(t) &= e^{\Lambda t}R^{-1}u_0. \\ \Longrightarrow \qquad u(t) &= Re^{\Lambda t}R^{-1}u_0 \\ &= e^{At}u_0. \end{aligned}$$

It remains to define the notion of a matrix exponential for a general matrix. To this end, we define for a given $m \times m$ *matrix* A *the matrix exponential as*

$$e^{At} = I + At + \frac{1}{2}A^2t^2 + \ldots + \frac{1}{k!}A^kt^k + \ldots. \tag{1.28}$$

The last equality then follows from the results of Exercise 1.6.

Exercise 1.6 *Let* $A \in \mathbb{R}^{m\times m}$ *be a diagonalisable matrix such that Equation* (1.26) *holds. Show that for all* $t \in [0, T]$ *it holds that*

$$e^{At} = Re^{\Lambda t}R^{-1}.$$

Example 1.7 (Duhamel's Principle (Variation of Constants))
We consider the following linear, scalar, non-autonomous IVP:

$$\begin{aligned} u'(t) &= \lambda u(t) + g(t), \\ u(0) &= u_0, \end{aligned} \tag{1.29}$$

where $u \in \mathbb{R}$, $u_0 \in \mathbb{R}$, $\lambda \in \mathbb{R}$ is a constant and $g\colon [0,T] \to \mathbb{R}$ is a continuous function.

We begin by defining $v(t) = u(t)e^{-\lambda t}$. Hence, it holds that $u(t) = v(t)e^{\lambda t}$, and therefore we have assumed that the solution u consists of a variable constant $v(t)$ multiplied with the homogeneous solution $e^{\lambda t}$. This assumption on the form of the solution motivates the name variation of constants*. It then follows that*

$$\begin{aligned} v'(t) &= u'(t)e^{-\lambda t} - \lambda u(t)e^{-\lambda t} = \big(u'(t) - \lambda u(t)\big)e^{-\lambda t} \\ &\overset{(1.29)}{=} g(t)e^{-\lambda t}. \end{aligned}$$

It therefore follows that

$$v(t) = u_0 + \int_0^t g(s)e^{-\lambda s} ds.$$

Hence,

$$\begin{aligned} u(t)e^{-\lambda t} &= u_0 + \int_0^t g(s)e^{-\lambda s} ds. \\ \implies \qquad u(t) &= u_0 e^{\lambda t} + \int_0^t g(s)e^{\lambda(t-s)} ds. \end{aligned}$$

Exercise 1.8 *Consider the following IVP involving a linear, non-autonomous system of ODEs:*

$$\begin{aligned} u'(t) &= Au(t) + g(t), \\ u(0) &= u_0, \end{aligned} \tag{1.30}$$

where $u \in \mathbb{R}^m$, $u_0 \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times m}$ is a constant, diagonalisable $m \times m$ matrix and $g\colon [0,T] \to \mathbb{R}^m$ is a continuous function.

Show that the solution to the IVP (1.30) *is given by*

$$u(t) = e^{At}u_0 + \int_0^t e^{A(t-s)} g(s) ds. \tag{1.31}$$

We observe that all of the examples considered in this section have involved linear ODEs. Unfortunately, explicit solutions can only be obtained for very few non-linear ODEs. Therefore, solving non-linear IVPs necessitates the use of numerical methods to approximate these ODEs. This has resulted in an extremely rich literature on the numerical analysis of ordinary differential equations.

1.7 Well-Posedness of Initial Value Problems for ODEs

In this section, we briefly explore the questions of existence and uniqueness of solutions to initial value problems for ordinary differential equations. The main result on well-posedness of IVPs for ODEs is the Cauchy-Lipschitz Theorem (also known as the Picard-Lindelöf Theorem).

Theorem 1.9 (Cauchy-Lipschitz) *Let $U \subseteq \mathbb{R}^m$ be an open set and consider the following non-autonomous IVP:*

$$\begin{aligned} u'(t) &= F\big(t, u(t)\big), \\ u(0) &= u_0, \end{aligned} \tag{1.32}$$

where $u_0 \in U \subseteq \mathbb{R}^m$ is a constant, and $F\colon [0,T]\times\mathbb{R}^m \to \mathbb{R}^m$ and $u\colon [0,T] \to \mathbb{R}^m$ are continuous functions in t.

Suppose that the function F is Lipschitz continuous on the set $[0,T] \times U$, i.e., there exists $L > 0$ such that for all $(t,u),(t,u^*) \in [0,T] \times U$ it holds that

$$\|F(t,u) - F(t,u^*)\| \leq L\|u - u^*\| \tag{1.33}$$

where $\|\cdot\|$ is any vector norm. Then there exists $T^* \in (0,T)$ such that the IVP (1.32) has a unique solution in the time interval $[0,T^*]$.

We now attempt to check the hypothesis of Theorem 1.9 for some simple ODEs.

Example 1.10 *Consider the linear, autonomous ODE given by*

$$u'(t) = Au(t),$$

where $u \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times m}$ is a constant $m \times m$ matrix.

Clearly for all $u, u^* \in \mathbb{R}^m$ it holds that

$$\|Au - Au^*\|_{\mathbb{R}^m} \leq \|A\|_2\|u - u^*\|_{\mathbb{R}^m}.$$

where $\|\cdot\|_{\mathbb{R}^m}$ *is the Euclidean norm and* $\|\cdot\|_2$ *is the spectral norm. Since the spectral norm* $\|A\|_2$ *is bounded, we can successfully apply Theorem 1.9 in this case.*

Example 1.11 *Consider the first-order Pendulum equation* (1.19)*. In this case the function* $F\colon \mathbb{R}^2 \to \mathbb{R}^2$ *is given by*

$$F(\theta, v) = [v, \sin(\theta)].$$

It follows that for all $(\theta, v), (\theta^*, v^*) \in \mathbb{R}^2$ *it holds that*

$$\begin{aligned}\|F(\theta, v) - F(\theta^*, v^*)\|_1 &= \|\big(v - v^*,\ \sin(\theta) - \sin(\theta^*)\big)\|_1 \\ &= |v - v^*| + |\sin(\theta) - \sin(\theta^*)| \\ &= |v - v^*| + |\cos(\bar{\theta})||\theta - \theta^*|,\end{aligned}$$

where $\bar{\theta} \in \big[\min\{\theta, \theta^*\},\ \max\{\theta, \theta^*\}\big]$ *and the last equality is a consequence of the Mean Value Theorem. Hence for all* $(\theta, v),\ (\theta^*, v^*) \in \mathbb{R}^2$ *it holds that*

$$\|F(\theta, v) - F(\theta^*, v^*)\|_1 \leq |v - v^*| + |\theta - \theta^*| = \|(\theta, v) - (\theta^*, v^*)\|_1.$$

Thus the function F *is Lipschitz continuous with Lipschitz constant* 1*. Therefore, we can once again successfully apply Theorem 1.9. We remark that in this example we used the* l_1 *norm which resulted in a Lipschitz constant of exactly* 1*. Using a different norm could potentially result in a Lipschitz constant greater than* 1*.*

Example 1.12 *Consider a non-linear, scalar, autonomous IVP given by*

$$\begin{aligned}u'(t) &= \big(u(t)\big)^2, \\ u(0) &= u_0,\end{aligned} \tag{1.34}$$

where $u \in \mathbb{R}$ *and* $u_0 \in \mathbb{R}$ *is a positive constant.*

We immediately observe that the function $F\colon \mathbb{R} \to \mathbb{R}$ *given by* $F(u) = u^2$ *is not globally Lipschitz continuous. Therefore, we can only appeal to Theorem 1.9 to establish uniqueness locally, but* not *globally. Indeed, the explicit solution in this case is given by*

$$u(t) = \frac{u_0}{1 - u_0 t}.$$

Clearly, $\lim_{t \to 1/u_0{}^-} u(t) = \infty$ *and therefore the solution only exists for*

$$0 \leq t < \frac{1}{u_0}.$$

Example 1.13 *Finally, consider a non-linear, scalar, autonomous IVP given by*

$$\begin{aligned} u'(t) &= \sqrt{|u(t)|}, \\ u(0) &= 0, \end{aligned} \tag{1.35}$$

where $u \in \mathbb{R}$.

We observe once again that the function $F \colon \mathbb{R} \to \mathbb{R}$ *given by* $F(u) = \sqrt{|u|}$ *is not Lipschitz continuous in a neighbourhood of* $u = 0$. *Therefore, once again we cannot apply Theorem 1.9 to prove the existence of a unique solution to the IVP* (1.35). *Indeed, there exist at least two different solutions to this IVP:*

$$u(t) \equiv 0,$$

and also

$$u(t) = \frac{1}{4}t^2.$$

Thus we require Lipschitz continuity in order to guarantee uniqueness of solutions to IVPs.

2 Numerical Methods for Ordinary Differential Equations

In this chapter we will describe different methods to numerically approximate solutions of the standard first-order initial value problem for ODEs

$$\begin{aligned} u'(t) &= F(t, u(t)), \\ u(0) &= u_0, \end{aligned} \tag{2.1}$$

where $u_0 \in \mathbb{R}^m$ is a constant, $u\colon [0,T] \to \mathbb{R}^m$ and $F\colon [0,T] \times \mathbb{R}^m \to \mathbb{R}^m$. Depending on the dimension of our problem and hence the value of $m \in \mathbb{N}$, both u and F can be scalar or vector-valued functions.

Recall that we claimed in Section 1.6 that it is only possible to find explicit solutions for the IVP (2.1) in a handful of cases and therefore we require numerical methods to obtain approximate solutions to the IVP (2.1).

2.1 Time Discretisation

We consider the IVP (2.1) in the time interval [0, T] where $T \in (0, \infty)$ is some fixed, final time. The first step in the numerical analysis of this IVP involves the discretisation of the time domain. Therefore, let $\Delta t > 0$ be the time step size, let $N = \frac{T}{\Delta T}$ be the total number of time steps and consider the partition of the time domain given by

$$[0, T) = \cup_{n=0}^{N-1} [t^n, t^{n+1}),$$

where $t^n = n\Delta t$. Thus, we have divided the time period $[0, T]$ into N equally spaced intervals (see Figure 2.1). The time steps $\{t^n\}_{n=0}^{N}$ are also termed *time levels* and we use the notation t^n to refer to the n^{th} time level.

Our aim is to now numerically approximate the exact solution u of the IVP

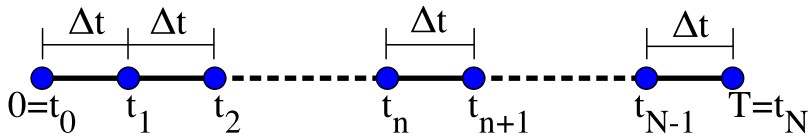


Figure 2.1: Time discretisation of the domain $[0, T]$.

(2.1) at these time levels. We therefore denote by $\{U_n\}_{n=0}^N$ the approximations of the exact solution u at each time level:

$$U_n \approx u(t^n) \tag{2.2}$$

We conclude that numerically solving the IVP (2.1) amounts to calculating the quantities $\{U_n\}_{n=0}^N$. In addition, we observe that the initial condition of the IVP (2.1) implies that $U_0 \approx u(t^0) = u(0) = u_0$ and hence we may set $U_0 = u_0$.

We remark that throughout this section we have assumed for simplicity that the time steps are of equal spacing, i.e. for all $n = 1, \ldots, N$ it holds that $t^n - t^{n-1} = \Delta t$ where Δt is a fixed constant. We will consider the more general situation of a non-uniform time discretisation in later parts of this chapter.

2.2 Forward Euler Method

In order to obtain the approximate solutions at each time level $\{U_n\}_{n=0}^N$, it is necessary to approximate the IVP (2.1). Hence, the next step is to obtain suitable approximations of the time derivative u' and the right-hand side $F\big(t, u(t)\big)$.

The simplest discretisations of the derivative are given by the finite difference formulae. The most obvious choice is the so-called forward difference approximation. Thus for each $n \in \{0, \ldots, N-1\}$ we approximate the time derivative by

$$u'(t^n) \approx \frac{u(t^{n+1}) - u(t^n)}{\Delta t} \approx \frac{U_{n+1} - U_n}{\Delta t}.$$

In addition, for each $n \in \{0, \ldots, N-1\}$ the right-hand side F is approximated by

$$F(t^n, u(t^n)) \approx F(t^n, U_n).$$

Therefore, for each $n \in \{0, \ldots, N-1\}$ we obtain the following approximation of the IVP (2.1):

$$\begin{aligned} \frac{U_{n+1} - U_n}{\Delta t} &= F(t^n, U_n), \\ U_0 &= u_0. \end{aligned} \tag{2.3}$$

A simple rearrangement of the terms in Equation (2.3) yields for each $n \in \{0, \ldots, N-1\}$

$$\begin{aligned} U_{n+1} &= U_n + \Delta t F(t^n, U_n), \\ U_0 &= u_0. \end{aligned} \tag{2.4}$$

The numerical scheme given by (2.3) or (2.4) is known as the *Forward Euler* method and is one of the simplest examples of a numerical scheme for approximating solutions to the IVP (2.1). A characteristic of the Forward Euler method (2.4) is that the method can be implemented as a time marching scheme

$$U_0 \mapsto U_1 \mapsto U_2 \mapsto \ldots U_n \mapsto U_{n+1} \ldots \mapsto U_N.$$

In other words, for each $n \in \{0, \ldots, N-1\}$, we can compute the approximate solution U_{n+1} using the previous value U_n and a simple function evaluation given by Equation (2.4). In particular, the Forward Euler method is also called the *Explicit* Euler method.

2.3 Backward Euler Method

The forward difference approximation for time derivatives considered in Section 2.2 is not our only choice. Another possibility is to employ the so-called backward difference approximation instead of the forward difference approximation. Thus for each $n \in \{0, \ldots, N-1\}$,we approximate the time derivative u' by

$$u'(t^{n+1}) \approx \frac{u(t^{n+1}) - u(t^n)}{\Delta t} \approx \frac{U_{n+1} - U_n}{\Delta t},$$

and for each $n \in \{0, \ldots, N-1\}$ we approximate the right-hand side F by

$$F(t^{n+1}, u(t^{n+1})) \approx F(t^{n+1}, U_{n+1}).$$

Note that both u' and the right-hand side F are approximated at the time level t^{n+1}. Thus, for each $n \in \{0, \ldots, N-1\}$ we obtain the following approximation of the IVP (2.1):

$$\begin{aligned} \frac{U_{n+1} - U_n}{\Delta t} &= F(t^{n+1}, U_{n+1}), \\ U_0 &= u_0. \end{aligned} \tag{2.5}$$

Once again a simple rearrangement of the terms in Equation (2.5) yields for each $n \in \{0, \ldots, N-1\}$

$$\begin{aligned} U_{n+1} - \Delta t F(t^{n+1}, U_{n+1}) &= U_n, \\ U_0 &= u_0. \end{aligned} \tag{2.6}$$

The numerical scheme given by Equation (2.5) or Equation (2.6) is known as the *Backward Euler* method and is another example of a numerical scheme for approximating solutions to the IVP (2.1). Note that the Backward Euler method (2.6) can also be implemented as a time marching scheme

$$U_0 \mapsto U_1 \mapsto U_2 \mapsto \ldots U_n \mapsto U_{n+1} \ldots \mapsto U_N.$$

That is, for each $n \in \{0, \ldots, N-1\}$, we can compute the approximate solution U_{n+1} using the previous value U_n and Equation (2.6). However, it is important to note that Equation (2.6) is, in general, a non-linear system of equations and therefore needs to be solved numerically using, e.g., the Newton method. In particular, the Backward Euler method is also called the *Implicit* Euler method.

2.4 Trapezoidal Method

Let us consider the IVP (2.1) and apply the Fundamental theorem of Calculus to obtain for each $n \in \{0, \ldots, N-1\}$

$$\begin{aligned} u(t^{n+1}) - u(t^n) &= \int_{t^n}^{t^{n+1}} u'(s) ds, \qquad \text{(Fundamental Theorem of Calculus)}, \\ &= \int_{t^n}^{t^{n+1}} F(s, u(s)) ds, \qquad \text{(IVP (2.1))}. \end{aligned} \tag{2.7}$$

Of course, in general, the function F may be highly non-linear and therefore the integral in Equation (2.7) may be difficult to compute exactly. An

alternative approach however, could be to approximate the integral using numerical quadrature and one possibility is to use the so-called *Trapezoidal rule*:

$$\int_a^b f(x)dx \approx (b-a)\Big(\frac{f(a)+f(b)}{2}\Big).$$

Equation (2.7) then reduces to

$$\frac{u(t^{n+1})-u(t^n)}{\Delta t} \approx \frac{1}{2}\Big(F\big(t^n,u(t^n)\big)+F\big(t^{n+1},u(t^{n+1})\big)\Big).$$

Therefore, using the approximation (2.2), i.e., for each $n \in \{0,\dots,N-1\}$ setting $U_n \approx u(t^n)$, we obtain the following approximation of the IVP (2.1):

$$\begin{aligned} \frac{U_{n+1}-U_n}{\Delta t} &= \frac{1}{2}\big(F(t^n,U_n)+F(t^{n+1},U_{n+1})\big), \\ U_0 &= u_0. \end{aligned} \tag{2.8}$$

Finally, rewriting Equation (2.8) in the time marching format, we obtain for each $n \in \{0,\dots,N-1\}$

$$\begin{aligned} U_{n+1} &= U_n + \frac{\Delta t}{2}F(t^n,U_n) + \frac{\Delta t}{2}F(t^{n+1},U_{n+1}), \\ U_0 &= u_0. \end{aligned} \tag{2.9}$$

The numerical scheme for approximating solutions to the IVP (2.1) given by Equation (2.8) or Equation (2.9) is known as the *Trapezoidal method*. Like the Backward Euler method, the Trapezoidal method is also an example of an implicit scheme.

Exercise 2.1 *Show that the Trapezoidal method given by Equation* (2.9) *is formally, the average of the Forward Euler method* (2.4) *and the Backward Euler method* (2.6)*.*

2.5 Mid-point Rule

Recall that the Forward Euler method (Section 2.2) and the Backward Euler method (Section 2.3) were derived using the forward difference approximation and the backward difference approximation for the time derivative u' respectively. Of course these are not the only possibilities and a third option is to use so-called central differences to approximate the time derivative u'. Hence, for each $n \in \{0,\dots,N-1\}$ we approximate the time derivative u' by

$$u'(t^n) \approx \frac{u(t^{n+1})-u(t^{n-1})}{2\Delta t} \approx \frac{U_{n+1}-U_{n-1}}{2\Delta t},$$

and for each $n \in \{0, \ldots, N-1\}$ we approximate the right-hand side F by

$$F(t^n, u(t^n)) \approx F(t^n, U_n).$$

Using the above approximations we obtain the following approximation of the IVP (2.1):

$$\begin{aligned} \frac{U_{n+1} - U_{n-1}}{2\Delta t} &= F(t^n, U_n), \\ U_0 &= u_0. \end{aligned} \tag{2.10}$$

Finally, rearranging the terms in Equation (2.10), we obtain for each $n \in \{1, \ldots, N-1\}$

$$\begin{aligned} U_{n+1} &= U_{n-1} + 2\Delta t F(t^n, U_n), \\ U_0 &= u_0. \end{aligned} \tag{2.11}$$

The numerical scheme given by Equation (2.10) or Equation (2.11) is known as the *Mid-point* method[1]. The Mid-point method is again an example of an explicit numerical scheme and like the Forward Euler, Backward Euler and Trapezoidal methods, the Mid-point method (2.11) can also be implemented as a time marching scheme. Indeed, given the values of the approximate solution U_{n-1}, U_n we can compute the approximate solution at the next time level U_{n+1} using a simple function evaluation involving Equation (2.11).

We observe however that in contrast to the previous schemes, the Mid-point method uses the approximate solution at *two* previous time levels in order to compute the approximate solution at the new time level. In particular, this implies that we need to approximate U_1 using some other numerical scheme in order to begin time marching with the Mid-point scheme. An attractive choice is the Forward Euler methods, which yields

$$U_1 = u_0 + \Delta t F(0, u_0).$$

2.6 Numerical Experiments

We consider the following scalar IVP:

$$\begin{aligned} u'(t) &= -\big(u(t) - \sin(t)\big) + \cos(t), \\ u(0) &= 0. \end{aligned} \tag{2.12}$$

[1]Beware that there are several different numerical schemes known as the Mid-point method in the literature.

The exact solution of the IVP (2.12) is given by $u(t) = \sin(t)$. We use different numerical methods to obtain approximate solutions to this IVP and use the exact solution to estimate the global errors associated with the various numerical schemes we have introduced so far. Table 2.1 contains our results and indicates that the Trapezoidal method is significantly more accurate than the Forward Euler and the Backward Euler methods. Figure

	FE		BE		TR	
N	Error	Rate	Error	Rate	Error	Rate
20	0.083		0.074		0.004	
40	0.040	1.04	0.038	0.96	0.001	2.00
80	0.020	1.02	0.019	0.98	2.57×10^{-4}	2.00
160	0.010	1.01	0.010	0.99	6.41×10^{-5}	2.00

Table 2.1: Error Table of the Forward Euler method, the Backward Euler method and the Trapezoidal rule for the scalar IVP (2.12).

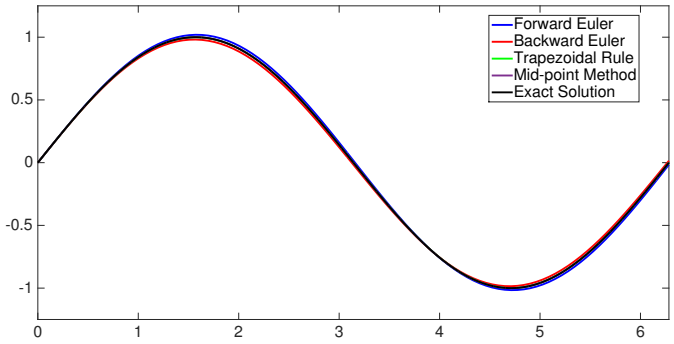


Figure 2.2: Approximate solution produced by different numerical methods for the scalar IVP (2.12) using N=100 points.

2.2, which displays the plots of the approximate solutions produced by these schemes, indicates that all the numerical schemes are able to produce accurate solutions if the total number of time points N is sufficiently large. On the other hand, Figure 2.3 also indicates that while using a smaller N produces less accurate solutions, increasing the number of points does improve the accuracy of the solution. This observation is confirmed by the results of the errors displayed in Table 2.1. Next, we consider an IVP

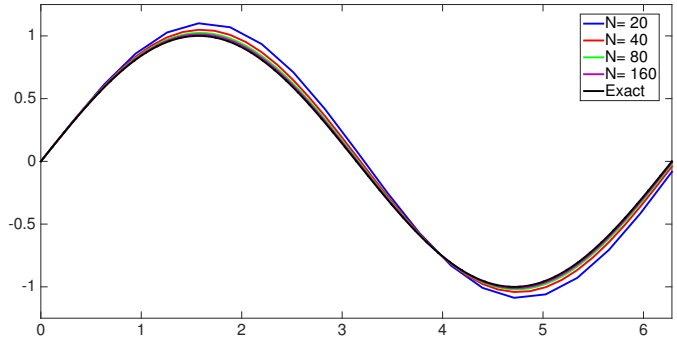


Figure 2.3: Approximate solutions produced by the Forward Euler method for the scalar IVP (2.12).

involving a system of ODEs given by

$$\begin{aligned} \theta'(t) &= v, \\ v'(t) &= -\sin(\theta), \\ \theta(0) &= 0, \\ v(0) &= 1. \end{aligned} \tag{2.13}$$

Recall that this is the so-called Pendulum equation introduced in the previous chapter. Once again, we use the Forward Euler method and the Backward Euler methods to approximate solutions to this systems of ODEs.

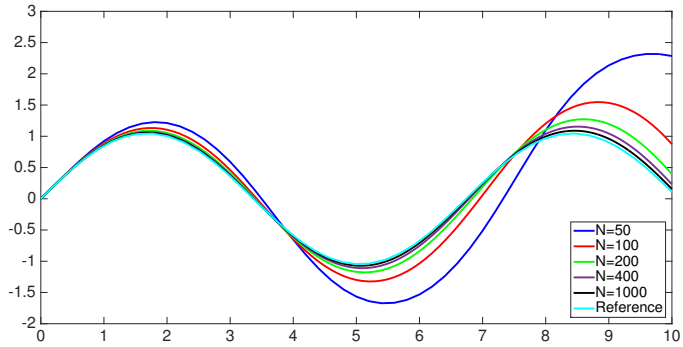


Figure 2.4: Plots of the approximate solution variable θ for the IVP (2.13) using the Forward Euler method.

Our results indicate that a very small time step size Δt or conversely a very large number of points N are required to produce accurate solutions to the IVP (2.13) using the first-order Forward Euler and Backward Euler schemes.

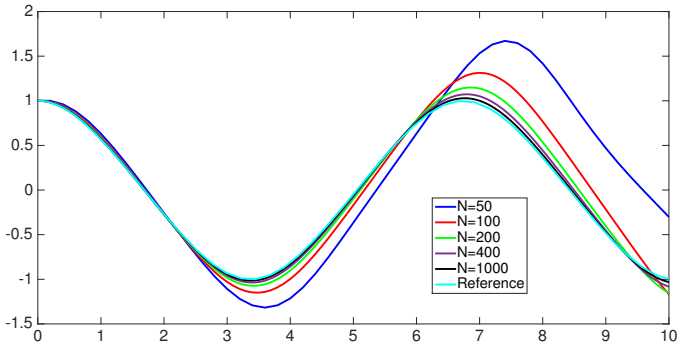


Figure 2.5: Plots of the approximate solution variable v for the IVP (2.13) using the Forward Euler method.

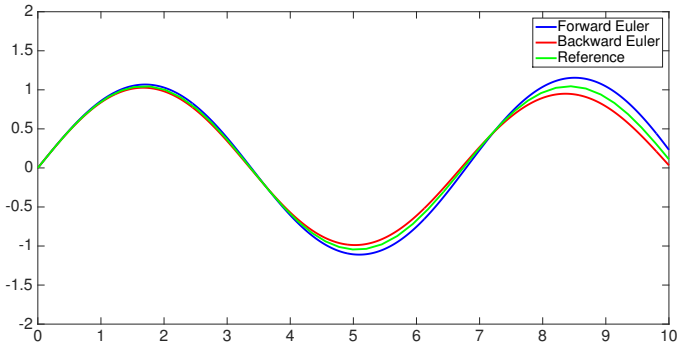


Figure 2.6: Plots of the approximate solution variable θ for the IVP (2.13) using the Forward Euler and the Backward Euler methods with N=400 points.

2.7 Truncation Error

The discussion in Sections 2.2-2.5 has been limited to introducing different numerical methods for approximating solutions to the IVP (2.1) and there has been little comparative examination of these numerical methods. Indeed, we have not described any systematic procedure to distinguish between these schemes or to establish the superiority of one scheme over another. The next step therefore is to analyse these numerical methods in more detail. For this purpose we first introduce the concept of the *truncation error*.

First, observe that the numerical schemes (2.3), (2.5), (2.8) and (2.10) are all consistent with the form of the IVP (2.1). In order to calculate the truncation error, we simply insert the exact solution $u(t)$ of the IVP (2.1) in the consistent forms of the numerical schemes (2.3), (2.5), (2.8) and (2.10).

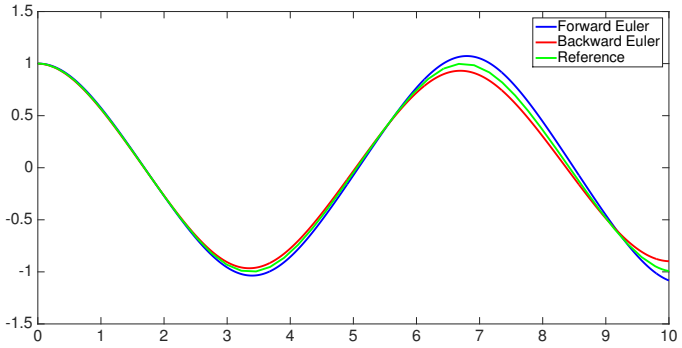


Figure 2.7: Plots of the approximate solution variable v for the IVP (2.13) using the Forward Euler and the Backward Euler methods with N=400 points.

In other words, the truncation error for a numerical scheme is the error made when the exact solution is inserted in the consistent form of numerical methods. As an example, let us compute the truncation error of the Forward Euler method using the Taylor expansions.

Example 2.2 *Consider IVP* (2.1) *and the Forward Euler method* (2.4). *Then the truncation error at each time level* t^n *is given by*

$$\begin{aligned} T_n &= \frac{u(t^{n+1}) - u(t^n)}{\Delta t} - F(t^n, u(t^n)) \\ &= \frac{u(t^n) - u(t^n) + \Delta t u'(t^n) + \frac{(\Delta t)^2}{2} u''(t^n) + \mathcal{O}\big((\Delta t)^3\big)}{\Delta t} - F(t^n, u(t^n)) \\ &= \underbrace{u'(t^n) - F(t^n, u(t^n))}_{=0 \text{ using IVP (2.1)}} + \frac{(\Delta t)}{2} u''(t^n) + \mathcal{O}\big((\Delta t)^2\big) = \mathcal{O}(\Delta t). \end{aligned}$$

Therefore the truncation error of the Forward Euler method is $\mathcal{O}(\Delta t)$.

As a second example, let us consider the Mid-point method.

Example 2.3 *Consider IVP* (2.1) *and the Mid-point method* (2.11). *Then*

the truncation error at each time level t^n is given by

$$\begin{aligned}
T_n &= \frac{u(t^{n+1}) - u(t^n)}{\Delta t} - \frac{u(t^{n-1}) - u(t^n)}{\Delta t} - 2F(t^n, u(t^n)) \\
&= \frac{u(t^{n+1}) - u(t^{n-1})}{\Delta t} - 2F(t^n, u(t^n)) \\
&= \frac{u(t^n) - u(t^n) + \Delta t u'(t^n) + \Delta t u'(t^n) + \frac{(\Delta t)^2}{2}u''(t^n) - \frac{(\Delta t)^2}{2}u''(t^n)}{\Delta t} \\
&+ \frac{\frac{(\Delta t)^3}{6}u'''(t^n) + \frac{(\Delta t)^3}{6}u'''(t^n) + \mathcal{O}\big((\Delta t)^4\big)}{\Delta t} - 2F(t^n, u(t^n)) \\
&= 2\underbrace{\big(u'(t^n) - F(t^n, u(t^n))\big)}_{=0 \text{ using IVP (2.1)}} + \frac{(\Delta t)^2}{3}u'''(t^n) + \mathcal{O}\big((\Delta t)^3\big) \\
&= \mathcal{O}\big((\Delta t)^2\big).
\end{aligned}$$

Therefore the truncation error of the Mid-point method is $\mathcal{O}\big((\Delta t)^2\big)$.

Exercise 2.4 *Show that the truncation error of the Backward Euler method* (2.6) *is* $\mathcal{O}(\Delta t)$.

Exercise 2.5 *Show that the truncation error of the Trapezoidal method* (2.9) *is* $\mathcal{O}\big((\Delta t)^2\big)$.

In general, if a numerical method for approximating solutions to the IVP (2.1) has truncation error $\mathcal{O}\big((\Delta t)^k\big)$ for some integer $k \in \mathbb{N}$, we say that this numerical method is k-order accurate. Thus, for example, the Forward Euler method is first-order accurate while the Mid-point method is second-order accurate.

2.8 One-Step Error

The truncation error of a scheme is closely related to the concept of the so-called *one-step error* of a numerical scheme.

Observe that the numerical schemes (2.4), (2.6), (2.9) and (2.11) are in the so-called update form. The one-step error associated with these numerical schemes is obtained by inserting the exact solution $u(t)$ of the IVP (2.1) in the update forms (2.4), (2.6), (2.9) and (2.11). Therefore, the one-step

error for a numerical scheme is the error made when the exact solution is inserted in the *update form* of the numerical scheme. As an example, let us compute the one-step error of Forward Euler method using the Taylor expansions.

Example 2.6 *Consider IVP* (2.1) *and the Forward Euler method* (2.4)*. Then the one-step error at each time level t^n is given by*

$$\begin{aligned} L_n &= u(t^{n+1}) - u(t^n) - \Delta t F\big(t^n, u(t^n)\big) \\ &= u(t^n) - u(t^n) + \Delta t u'(t^n) + \frac{(\Delta t)^2}{2} u''(t^n) + \mathcal{O}\big((\Delta t)^3\big) \\ &\quad - \Delta t F\big(t^n, u(t^n)\big) \\ &= \Delta t\big(\underbrace{u'(t^n) - F\big(t^n, u(t^n)\big)}_{=0 \text{ using IVP (2.1)}} \big) + \frac{(\Delta t)^2}{2} u''(t^n) + \mathcal{O}\big((\Delta t)^3\big) \\ &= \mathcal{O}\big((\Delta t)^2\big). \end{aligned}$$

Therefore the one-step error of the Forward Euler method is $\mathcal{O}\big((\Delta t)^2\big)$.

Exercise 2.7 *Show that the one-step error of the Backward Euler method* (2.4) *is $\mathcal{O}\big((\Delta t)^2\big)$.*

Exercise 2.8 *Show that the one-step error of the Trapezoidal method* (2.9) *is $\mathcal{O}\big((\Delta t)^3\big)$.*

We observe that there is a clear relation between the one-step error and the truncation error for a one-step numerical method: A truncation error of $\mathcal{O}\big((\Delta t)^k\big)$ implies a one-step error of $\mathcal{O}\big((\Delta t)^{k+1}\big)$.

Finally, we remark that the title of the one-step error in the following way: Assume that the exact solution of the IVP (2.1) at the time level t^{n+1} is known and is given by $u(t^{n+1})$. Recall that the update form of the Forward Euler method for approximating the IVP (2.1) is given by

$$U_{n+1} = U_n + \Delta t F\big(t^n, U_n\big).$$

Therefore, the error introduced by the Forward Euler method in this single step is given by

$$\begin{aligned} u(t^{n+1}) - U^{n+1} &= u(t^{n+1}) - u(t^n) - \Delta t F\big(t^n, u(t^n)\big) \\ &= L_n. \end{aligned} \tag{2.14}$$

Hence, the one-step error L_n is exactly the error introduced by a single step of the numerical scheme and this nomenclature is therefore justified.

2.9 Global Error

Both the truncation error (Section 2.7) and the one-step error (Section 2.8) associated with a numerical scheme are examples of *local* errors. In other words, both the truncation error and the one-step error compute the error due to the approximation of the IVP by the numerical scheme at a *single* time step. In this section we consider a heuristic argument for estimating the *global error* associated with a numerical scheme.

Consider the IVP (2.1), let $u(t^N)$ and U_N denote the exact solution of the IVP and the approximate solution produced by a numerical scheme at the final time level $t^N = T$ respectively. Then the global error of the numerical scheme is defined as

$$E_N := u(t^N) - U_N. \tag{2.15}$$

Let us now consider the following heuristic argument: For each $j \in \{0, \dots, N-1\}$, let L_j denote the one-step error at the time level t^j. Then, the following approximate estimate should hold:

$$E_N \approx \sum_{j=0}^{N-1} L_j \quad \Longrightarrow \quad |E_N| \leq \sum_{j=0}^{N-1} |L_j|.$$

Now assume that the one-step error associated with the numerical scheme satisfies

$$L_j = \mathcal{O}\big((\Delta t)^{q+1}\big),$$

for some $q \in \mathbb{N}$. Then it holds that

$$\begin{aligned} E_N &\approx \mathcal{O}\big((\Delta t)^{q+1}\big) \cdot N = \mathcal{O}\big((\Delta t)^{q+1}\big) \cdot \frac{T}{\Delta t} \\ &\approx \mathcal{O}\big((\Delta t)^{q}\big) \end{aligned}$$

Thus, if exactly the same amount of error is produced at each time step, i.e., there is no amplification of errors over time steps, then the global error scales like the truncation error. In particular this indicates that both the Trapezoidal method and the Mid-point method should be more accurate than the Forward Euler and Backward Euler methods.

We remark however, that the above heuristic calculation is very informal. Indeed, it is by no means clear that there is no amplification of errors across time steps, particularly for non-linear ODEs. These ideas are related to the concept of *stability* of a numerical scheme and will be discussed in detail in a later chapter.

2.10 Taylor Expansion Methods

In view of the discussion in Sections 2.7-2.9, the most obvious way to improve the order of accuracy of a numerical method would be to use Taylor expansions in order to reduce truncation errors. We demonstrate this methodology using a simple example.

Example 2.9 *Consider a scalar, autonomous IVP given by*

$$\begin{aligned} u'(t) &= f(u(t)), \\ u(0) &= u_0. \end{aligned} \tag{2.16}$$

The Taylor expansions imply that for all time levels t^n it holds that

$$\begin{aligned} u(t^{n+1}) =& u(t^n) + \Delta t u'(t^n) + \frac{(\Delta t)^2}{2} u''(t^n) + \frac{(\Delta t)^3}{6} u'''(t^n) \\ &+ \frac{(\Delta t)^4}{24} u^{(4)}(t^n) + \mathcal{O}\big((\Delta t)^5\big). \end{aligned} \tag{2.17}$$

In addition, Equation (2.16) *implies that $u'(t^n) = f(u(t^n))$ and therefore it holds that*

$$\begin{aligned} u''(t^n) &= f'(u(t^n)) \cdot f(u(t^n)), \\ u'''(t^n) &= f''(u(t^n)) \cdot f(u(t^n))^2 + f'(u(t^n))^2 f(u(t^n)). \end{aligned}$$

Next, we insert the expression for the derivatives into the Taylor expansions (2.17) *to obtain the following numerical method for approximating the IVP* (2.1)*:*

$$\begin{aligned} U_{n+1} &= U_n + \Delta t f(U_n) + \frac{(\Delta t)^2}{2} f(U_n) f'(U_n) + \frac{(\Delta t)^3}{6} f'(U_n)^2 f(U_n) \\ &\quad + \frac{(\Delta t)^3}{6} \big(f''(U_n) f(U_n)^2\big), \\ U_0 &= u_0. \end{aligned} \tag{2.18}$$

The results of Exercise 2.10 imply that this numerical scheme could be a third-order accurate numerical method.

Exercise 2.10 *Show that the numerical scheme given by Equation* (2.18) *has truncation error $T_n = \mathcal{O}\big((\Delta t)^3\big)$ and one-step error $L_n = \mathcal{O}\big((\Delta t)^4\big)$.*

Higher order numerical methods can be obtained in a similar fashion. Unfortunately, using Taylor expansions and substituting higher order derivative terms becomes increasingly difficult for non-autonomous ODEs or systems of ODEs. We therefore employ different approaches to obtain higher-order accurate numerical schemes for approximating the IVP (2.1), and we consider these in the next chapter.

3 Higher-Order Methods for ODEs

Consider the following standard first-order initial value problem for ODEs:

$$\begin{aligned} u'(t) &= F(t, u(t)), \\ u(0) &= u_0, \end{aligned} \tag{3.1}$$

where $u_0 \in \mathbb{R}^m$ is a constant, $u\colon [0,T] \to \mathbb{R}^m$ and $F\colon [0,T] \times \mathbb{R}^m \to \mathbb{R}^m$.

As mentioned briefly in Section 2.10, it is difficult to use Taylor expansions in order to derive high-order accurate numerical schemes for approximating solutions to the IVP (3.1). We therefore require alternative means of designing high-order accurate numerical methods. We begin by describing a very popular family of high-order methods known as the Runge-Kutta (RK) methods.

3.1 The Runge-Kutta-2 (RK-2) Method

We consider the IVP (3.1) in the time interval [0, T] where $T \in (0, \infty)$ is some fixed time and we assume the time-discretisation described in Section 2.1. Then for all time levels $t^n,\ n \in \{0, 1, \ldots, N-1\}$, it holds that

$$\begin{aligned} u(t^{n+1}) - u(t^n) &= \int_{t^n}^{t^{n+1}} u'(s)ds, \qquad \text{(Fundamental Theorem of Calculus)}, \\ &= \int_{t^n}^{t^{n+1}} F(s, u(s))ds, \quad \text{(IVP (3.1))}. \end{aligned} \tag{3.2}$$

We can then use numerical quadrature to approximate (3.2). In particular, using the so-called mid-point rule yields

$$\int_{t^n}^{t^{n+1}} F(s, u(s))ds \approx \Delta t F\big(t^n + {}^{\Delta t}/_2, u(t^n + {}^{\Delta t}/_2)\big). \tag{3.3}$$

Of course the mid-point value of u is still unknown so we can perform a further approximation of this value using the Forward Euler method, i.e.,

$$u(t^n + {}^{\Delta t}/_2) \approx u(t^n) + \tfrac{\Delta t}{2} F\big(t^n, u(t^n)\big). \tag{3.4}$$

Finally, combining (3.3) and (3.4) we obtain a so-called **two-stage** numerical scheme for approximating solutions to the IVP (3.1):

$$\begin{aligned} Y_1 &= U_n, \\ Y_2 &= U_n + \frac{\Delta t}{2} F(t^n, Y_1), \\ U_{n+1} &= U_n + \Delta t F(t^n + {}^{\Delta t}/_2, Y_2), \\ U_0 &= u_0. \end{aligned} \tag{3.5}$$

Then the numerical scheme given by (3.5) is termed the standard 2-stage Runge-Kutta (RK-2) method.

The 2-stages refer to the calculation of the two terms Y_1, Y_2 in Equation (3.5) in order to compute the solution at the next time level.

We remark that the RK-2 method (3.5) can be re-written in the update form as

$$\begin{aligned} U_{n+1} &= U_n + \Delta t F\big(t^n + {}^{\Delta t}/_2, U_n + {}^{\Delta t}/_2 F(t^n, U_n)\big), \\ U_0 &= u_0, \end{aligned} \tag{3.6}$$

and therefore represents a time marching scheme.

3.1.1 Order of Accuracy of the RK-2 Method

The order of accuracy of the the RK-2 methods can easily be determined using the following strategy.

- **Step 1** Consider an IVP involving a linear scalar ODE given by
$$\begin{aligned} u'(t) &= \lambda u(t), \\ u(0) &= u_0, \end{aligned}$$
with exact solution given by $u(t) = u_0 e^{\lambda t}$.
- **Step 2** Compute the approximate solution at the time level t^{n+1} using the update form (3.6) of the RK-2 scheme:
$$U_{n+1} = \Big(1 + \lambda \Delta t + \tfrac{\lambda^2 (\Delta t)^2}{2}\Big) U_n.$$
- **Step 3** Compute the corresponding one-step error of the RK-2 scheme:
$$\begin{aligned} L_n = u(t^{n+1}) - U_{n+1} &= u(t^n) e^{\lambda \Delta t} - \Big(1 + \lambda \Delta t + \tfrac{\lambda^2 (\Delta t)^2}{2}\Big) u(t^n) \\ &= \big(e^{\lambda \Delta t} - 1 - \lambda \Delta t - \tfrac{\lambda^2 (\Delta t)^2}{2}\big) u(t^n) \\ &= \tfrac{\lambda^3 (\Delta t)^3}{6} u(t^n) + \mathcal{O}\big((\Delta t)^4\big) \\ &= \mathcal{O}\big((\Delta t)^3\big). \end{aligned}$$

- **Step 4** Based on the discussion in Section 2.9, we can conclude that the RK-2 method (under the assumption of numerical 'stability') has global error of the order $\mathcal{O}\big((\Delta t)^2\big)$ and is therefore second-order accurate.
- **Step 5** The order of accuracy of the RK-2 method then carries over from the simple linear scalar ODE to IVPs involving a general systems of ODEs of the form (3.1).

3.2 The Classical Runge-Kutta-4 (RK-4) Method

An even higher-order accurate numerical method for approximating solutions to the IVP (3.1) can be obtained by considering the following 4-stage numerical method:

$$\begin{aligned}
Y_1 &= U_n, \\
Y_2 &= U_n + \frac{\Delta t}{2} F(t^n, Y_1), \\
Y_3 &= U_n + \frac{\Delta t}{2} F(t^n + {}^{\Delta t}/_2, Y_2), \\
Y_4 &= U_n + \Delta t F(t^n + {}^{\Delta t}/_2, Y_3), \\
U_{n+1} &= U_n + \frac{\Delta t}{6}\big(F(t^n, Y_1) + 2F(t^n + {}^{\Delta t}/_2, Y_2) \\
&\qquad + 2F(t^n + {}^{\Delta t}/_2, Y_3) + F(t^n + \Delta t, Y_4)\big), \\
U_0 &= u_0.
\end{aligned} \tag{3.7}$$

Then the numerical scheme given by (3.7) is termed the classical 4-stage Runge-Kutta (RK-4) method.

We remark that this scheme can also be implemented as a time marching scheme similar to the RK-2 numerical method.

3.2.1 Order of Accuracy of the RK-4 Method

In order to determine the order of accuracy of the RK-4 numerical scheme, we follow the procedure outlined in Section 3.1.1. Hence we begin by considering an IVP involving a linear scalar ODE given by

$$\begin{aligned}
u'(t) &= \lambda u(t), \\
u(0) &= u_0.
\end{aligned}$$

Then, the RK-4 numerical method (3.7) implies that for each $n \in \{0, \dots, N-1\}$ it holds that

$$\begin{aligned}
Y_1 &= U_n, \\
Y_2 &= U_n + \frac{\lambda \Delta t}{2} U_n, \\
Y_3 &= U_n + \frac{\lambda \Delta t}{2}\big(U_n + \frac{\lambda \Delta t}{2} U_n\big) \\
&= U_n + \frac{\lambda \Delta t}{2} U_n + \frac{\lambda^2 (\Delta t)^2}{4} U_n, \\
Y_4 &= U_n + \lambda \Delta t\big(U_n + \frac{\lambda \Delta t}{2} U_n + \frac{\lambda^2 (\Delta t)^2}{4} U_n\big) \\
&= U_n + \lambda \Delta t U_n + \frac{\lambda^2 (\Delta t)^2}{2} U_n + \frac{\lambda^3 (\Delta t)^3}{4} U_n,
\end{aligned}$$

and therefore the approximate solution to this IVP obtained using the RK-4 numerical method is given by

$$\begin{aligned}
U_{n+1} &= U_n + \lambda \Delta t U_n + \tfrac{\lambda^2 (\Delta t)^2}{2} U_n + \tfrac{\lambda^3 (\Delta t)^3}{6} U_n + \tfrac{\lambda^4 (\Delta t)^4}{24} U_n \\
&= \Big(1 + \lambda \Delta t + \tfrac{\lambda^2 (\Delta t)^2}{2} + \tfrac{\lambda^3 (\Delta t)^3}{6} + \tfrac{\lambda^4 (\Delta t)^4}{24}\Big) U_n.
\end{aligned}$$

Thus, the one-step error associated with the RK-4 method is given by

$$\begin{aligned}
L_n &= u(t^{n+1}) - \Big(1 + \lambda \Delta t + \tfrac{\lambda^2 (\Delta t)^2}{2} + \tfrac{\lambda^3 (\Delta t)^3}{6} + \tfrac{\lambda^4 (\Delta t)^4}{24}\Big) u(t^n) \\
&= \Big(e^{\lambda \Delta t} - 1 - \lambda \Delta t - \tfrac{\lambda^2 (\Delta t)^2}{2} - \tfrac{\lambda^3 (\Delta t)^3}{6} - \tfrac{\lambda^4 (\Delta t)^4}{24}\Big) u(t^n) \\
&= \mathcal{O}\big((\Delta t)^5\big).
\end{aligned}$$

Hence, the RK-4 numerical method (3.7) (under the assumption of numerical 'stability') is expected to have global error of the order $\mathcal{O}\big((\Delta t)^4\big)$ and therefore is expected to be fourth-order accurate.

3.3 Numerical Experiments

At this point, it would be instructive to test these higher order Runge-Kutta schemes on the initial value problems introduced in Section 2.6. We first consider the scalar IVP (2.12) given by

$$\begin{aligned}
u'(t) &= -\big(u(t) - \sin(t)\big) + \cos(t), \\
u(0) &= 0.
\end{aligned}$$

In Section 2.6, we used the the Forward Euler, the Backward Euler, the Trapezoidal Rule and the Mid-point method to approximate solutions to this IVP. We repeat the same numerical experiments using the second-order RK-2 method and the fourth-order RK-4 method.

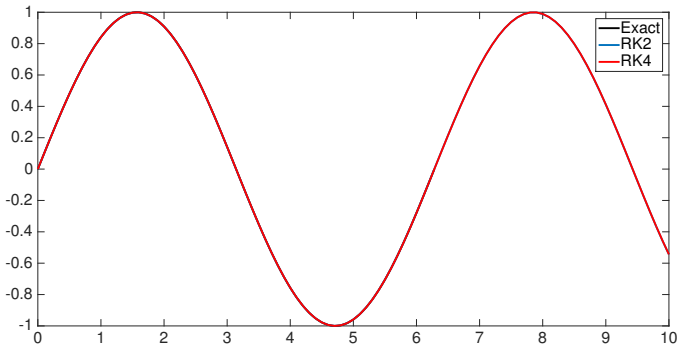


Figure 3.1: Approximate solution produced by the RK-2 and RK-4 methods for the scalar IVP (2.12) using N=100 points.

	FE		RK2		RK4	
N	Error	Rate	Error	Rate	Error	Rate
20	0.083		0.011		3.93×10^{-5}	
40	0.040	1.041	0.0023	2.176	2.04×10^{-6}	4.264
80	0.020	1.020	5.44×10^{-4}	2.087	1.16×10^{-7}	4.137
160	0.010	1.010	1.32×10^{-4}	2.043	6.92×10^{-9}	4.070

Table 3.1: Error Table of the Forward Euler method, the RK2 method and the RK4 method for the scalar IVP (2.12).

Figure 3.1 displays the approximate solution produced by the RK-2 and RK-4 methods using $N = 100$ points. Clearly, both the RK-2 and RK-4 schemes produce highly accurate solutions even for a smaller value of N. Our conclusion is supported by the errors associated with both schemes, which are displayed in Table 3.1. Indeed, the error of the RK-2 and the RK-4 methods is significantly smaller than the error of the Forward Euler method.

Next, we consider the IVP (2.13) given by

$$\begin{aligned}\theta'(t) &= v,\\ v'(t) &= -\sin(\theta),\\ \theta(0) &= 0,\\ v(0) &= 1.\end{aligned}$$

Once again we observe that both the RK-2 and the RK-4 method are able to produce accurate solutions to this system of ODEs. Indeed, as shown in Figures 3.2 and 3.3, the approximate solution produced by these Runge-Kutta methods is significantly more accurate than the approximate solution produced by the Forward Euler method.

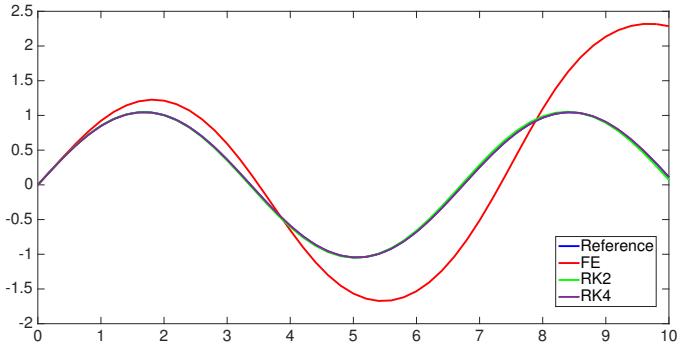


Figure 3.2: Plots of the approximate solution variable θ for the IVP (2.13) using different numerical schemes with N=100 points.

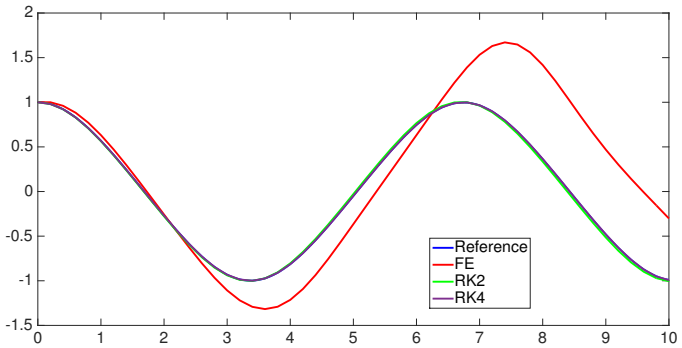


Figure 3.3: Plots of the approximate solution variable v for the IVP (2.13) using different numerical schemes with N=100 points.

3.4 General Form of the Runge-Kutta Methods

The RK-2 method (Section 3.1) and the RK-4 method (Section 3.2) suggest a general form of the s-stage Runge-Kutta methods for approximating solutions to the IVP (3.1):

Let $s \in \mathbb{N}$ be an integer, let $\{a_{ij}\}_{i,j=1}^{s}$, $\{b_i\}_{i=1}^{s}$ and $\{c_i\}_{i=1}^{s}$ be real numbers and define for all time levels $t^n,\ n \in \{0, 1, \ldots, N-1\}$

$$\begin{aligned}
Y_1 &= U_n + \Delta t \sum_{j=1}^{s} a_{1j} F(t^n + c_j \Delta t, Y_j), \\
Y_2 &= U_n + \Delta t \sum_{j=1}^{s} a_{2j} F(t^n + c_j \Delta t, Y_j), \\
&\vdots \quad \vdots \qquad \vdots \qquad\qquad \vdots \\
Y_s &= U_n + \Delta t \sum_{j=1}^{s} a_{sj} F(t^n + c_j \Delta t, Y_j), \\
U_{n+1} &= U_n + \Delta t \sum_{j=1}^{s} b_j F(t^n + c_j \Delta t, Y_j). \\
U_0 &= u_0.
\end{aligned} \tag{3.8}$$

Then the coefficients $\{a_{ij}\}_{i,j=1}^{s}$, $\{b_i\}_{i=1}^{s}$ and $\{c_i\}_{i=1}^{s}$ uniquely specify an s-stage Runge-Kutta method given by Equation (3.8).

For increased clarity and simplicity, the coefficients $\{a_{ij}\}_{i,j=1}^{s}$, $\{b_i\}_{i=1}^{s}$ and $\{c_i\}_{i=1}^{s}$ associated with an s-stage RK method are usually presented in tabular format as a so-called Butcher Tableau. The Butcher Tableau for an s-stage RK method is given by

$$\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & \ldots & a_{1s} \\
c_2 & a_{21} & a_{22} & \ldots & a_{2s} \\
\vdots & \vdots & \vdots & & \vdots \\
c_s & a_{s1} & a_{s2} & \ldots & a_{ss} \\
\hline
 & b_1 & b_2 & \ldots & b_s
\end{array} \tag{3.9}$$

We present some simple examples of Butcher tableaux for the Runge-Kutta methods we have previously introduced.

Example 3.1 *The Butcher tableau for the RK-2 method* (3.5) *is given by*

$$\begin{array}{c|cc}
0 & 0 & 0 \\
1/2 & 1/2 & 0 \\
\hline
 & 0 & 1
\end{array}$$

Example 3.2 *The Butcher tableau for the RK-4 method* (3.7) *is given by*

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

3.5 Consistency Conditions for Runge-Kutta methods

In general, an arbitrary combination of the coefficients $\{a_{ij}\}_{i,j=1}^{s}$, $\{b_i\}_{i=1}^{s}$ and $\{c_i\}_{i=1}^{s}$ will not specify a 'consistent' s-stage Runge-Kutta method (3.8). Instead we need to impose certain conditions on these coefficients in order to ensure consistency. In particular, we must assume that for all $i \in \{1, \ldots, s\}$

$$\sum_{j=1}^{s} a_{ij} = c_i, \tag{3.10}$$

and moreover

$$\sum_{j=1}^{s} b_j = 1. \tag{3.11}$$

Under the assumptions (3.10) and (3.11), the associated s-stage RK method for approximating solutions to the IVP (3.1) is consistent.

The simplest way to see that the conditions (3.10) and (3.11) must be satisfied in order to ensure consistency is to consider the simple case in which the IVP (3.1) consists of a scalar ODE. Now recall from Section 1.5 that the non-autonomous IVP (3.1) can be rewritten as an autonomous IVP by defining

$$w = [w_1, w_2], \quad G = [F(w_2, w_1), 1],$$

and considering the following IVP:

$$\begin{aligned} w'(t) &= G(w(t)), \\ w(0) &= [u_0, 1]. \end{aligned} \tag{3.12}$$

Then the autonomous IVP (3.12) is completely equivalent to the non-autonomous IVP (3.1) in the sense that for every time level $t^{n+1} = t^n + \Delta t,\ n \in \{0, \ldots, N-1\}$ it holds that

$$\begin{aligned} w_1(t^{n+1}) &= u(t^{n+1}), \\ w_2(t^{n+1}) &= t^{n+1} = t^n + \Delta t. \end{aligned}$$

Then, using the s-stage Runge-Kutta method (3.8) to approximate the IVP (3.12), we observe that for all $n \in \{0, \ldots, N-1\}$ it holds that

$$Z_i = \begin{bmatrix} Z_u^i \\ Z_t^i \end{bmatrix} = \begin{bmatrix} U_n + \Delta t \sum_{j=1}^s a_{ij} F\big(Z_t^i, Z_u^i\big) \\ t^n + \Delta t \sum_{j=1}^s a_{ij} \end{bmatrix},$$

and therefore, (3.13)

$$W^{n+1} = \begin{bmatrix} W_1^{n+1} \\ W_2^{n+1} \end{bmatrix} = \begin{bmatrix} U_n + \Delta t \sum_{j=1}^s b_j F\big(Z_t^i, Z_u^i\big) \\ t^n + \Delta t \sum_{j=1}^s b_j \end{bmatrix},$$

We must now show that the approximate solutions produced by Equation (3.13) is consistent with the approximate solutions produced by Equation (3.8). In other words, we must show that for all time level $t^{n+1} = t^n + \Delta t,\ n \in \{0, \ldots, N-1\}$, the approximate solutions produced by Equation (3.13) satisfy

$$\begin{aligned} W_1^{n+1} &= U_{n+1}, \\ W_2^{n+1} &= t^{n+1} = t^n + \Delta t. \end{aligned} \tag{3.14}$$

An inspection of Equation (3.13) indicates that the following condition must be satisfied in order for the condition (3.14) to hold:

$$\sum_{j=1}^s b_j = 1.$$

Furthermore, by setting for each time level $t^n,\ n \in \{1, \ldots, N\}$ and for each $i \in \{1, \ldots, s\}$

$$Z_t^i = t^n + c_i \Delta t,$$

we obtain for each $i \in \{1, \ldots, s\}$ the condition

$$\sum_{j=1}^s a_{ij} = c_i.$$

These two conditions together ensure that for each time level $t^{n+1},\ n \in \{0, \ldots, N-1\}$ and for each $i \in \{1, \ldots, s\}$ it holds that $Z_u^i = Y_i$ and $W_1^{n+1} = U_{n+1}$.

3.6 Examples of Runge-Kutta methods

We consider two main classes of Runge-Kutta methods in this section.

- **Explicit Runge-Kutta Methods**
  Consider an s-stage Runge-Kutta method of the form (3.8) for approximating solutions to the IVP (3.1) with the property that for all $i, j \in \{1, \ldots, s\}$

  $$a_{ij} = 0 \text{ if } j \geq i.$$

  Then this numerical method is termed an *explicit* RK method.

  We observe that by definition of explicit RK schemes, each stage $Y_i,\ i \in \{2, \ldots, s\}$ can be computed using only the previous stages $Y_j,\ j < i$, and therefore an explicit RK method can be implemented as a time marching scheme:

  $$U_n = Y_1 \mapsto Y_2 \mapsto Y_3 \mapsto \ldots \mapsto Y_{s-1} \mapsto Y_s \mapsto U_{n+1}.$$

  We also remark that the matrix $A = \{a_{ij}\}_{i,j=1}^{s}$ in the Butcher tableau (3.9) associated with an explicit RK scheme has a strictly lower triangular structure with zero diagonal entries. Examples of Explicit RK schemes include the RK-2 and the RK-4 numerical methods.

- **Diagonally Implicit Runge-Kutta (DIRK) Methods**
  Consider an s-stage Runge-Kutta method of the form (3.8) for approximating solutions to the IVP (3.1) with the property that for all $i, j \in \{1, \ldots, s\}$

  $$a_{ij} = 0 \text{ if } j > i,$$

  and with the property that there exists some $i \in \{1, \ldots, s\}$ such that

  $$a_{ii} \neq 0.$$

  Then this numerical method is termed a *diagonally implicit* RK method.

  We observe that by definition of diagonally implicit RK schemes, each stage $Y_i,\ i \in \{1, \ldots, s\}$ can be computed using the stages $Y_j,\ j \leq i$ and therefore computing each stage Y_i requires the solution of a non-linear equation of the form

  $$Y_i - \Delta t a_{ii} F(t^n + c_i \Delta t, Y_i) = U_n + \Delta t \sum_{j=1, j<i}^{s} a_{ij} F(t^n + c_i \Delta t, Y_i),$$

  where t^n is the current time level. In particular, this implies that we must use some numerical method such as, e.g., Newton's method to find the approximate solution to one or more non-linear equations at each time step.

Example 3.3 *As a concrete example, we consider the* 3*-stage second-order accurate DIRK method given by*

$$\begin{aligned} Y_1 &= U_n, \\ Y_2 &= U_n + \frac{\Delta t}{4}\big(F(t^n, Y_1) + F(t^n + \frac{\Delta t}{2}, Y_2)\big), \\ Y_3 &= U_n + \frac{\Delta t}{3}\big(F(t^n, Y_1) + F(t^n + \frac{\Delta t}{2}, Y_2) + F(t^n + \Delta t, Y_3)\big), \\ U_{n+1} &= U_n + \frac{\Delta t}{3}\big(F(t^n, Y_1) + F(t^n + \frac{\Delta t}{2}, Y_2) + F(t^n + \Delta t, Y_3)\big), \\ U_0 &= u_0. \end{aligned} \tag{3.15}$$

The Butcher Tableau for the DIRK-2 is therefore given by

0	0	0	0
1/2	1/4	1/4	0
1	1/3	1/3	1/3
	1/3	1/3	1/3

We remark that the DIRK-2 method is also known as the Trapezoidal Rule with the second order Backward Difference Formula (TR-BDF2).

3.7 Order of Accuracy of General RK Methods

Consider the IVP (3.1) and let an s-stage Runge-Kutta scheme for approximating solutions to this IVP be specified by the coefficients $\{a_{ij}\}_{i,j=1}^{s}$, $\{b_i\}_{i=1}^{s}$ and $\{c_i\}_{i=1}^{s}$.

As mentioned in Section 3.5, we require the consistency conditions (3.10) and (3.11) to be satisfied in order to ensure that these approximate solutions converge to the exact solution of the IVP (3.1). However, we also require additional constraints in order to guarantee that this Runge-Kutta scheme has truncation order of, say, $\mathcal{O}\big((\Delta t)^\gamma\big)$ where $\gamma > 1$.

In general, these additional constraints become increasingly complicated for higher-order accurate Runge-Kutta scheme ($\gamma \geq 4$) and these constraints are therefore outside the scope of these notes. However, we consider the simpler cases of $\gamma = 2, 3$ in this section.

Example 3.4 (Second-Order Runge Kutta Methods) *In order to obtain a second-order accurate numerical method, the following condition must*

be imposed on the coefficients associated with a consistent s-stage Runge-Kutta method:

$$\sum_{j=1}^{s} b_j c_j = \frac{1}{2}. \tag{3.16}$$

Exercise 3.5 *Show that both the RK-2 and the DIRK-2 numerical methods satisfy the condition* (3.16)*. Conclude that both these schemes are at least second-order accurate.*

Example 3.6 (Third-Order Runge Kutta Methods) *In order to obtain a third-order accurate numerical method, in addition to the condition* (3.16)*, the following additional conditions must be imposed on the coefficients associated with a consistent s-stage Runge-Kutta method:*

$$\begin{aligned} \sum_{j=1}^{s} b_j c_j^2 &= \frac{1}{3}, \\ \sum_{j=1}^{s} \sum_{i=1}^{s} b_i a_{ij} c_j &= \frac{1}{6}. \end{aligned} \tag{3.17}$$

Exercise 3.7 *Show that the RK-4 numerical method satisfies the conditions* (3.17) *while both the RK-2 and the DIRK-2 numerical methods do not satisfy the condition* (3.17)*. Conclude that the RK-4 scheme is at least third-order accurate while the RK-2 and DIRK-2 schemes are both exactly second-order accurate.*

We conclude this section by mentioning that, for explicit Runge-Kutta methods, in order to obtain γ-order accuracy where $\gamma < 5$, it is necessary to consider a method with at least γ stages. On the other hand in order to obtain γ-order accuracy where $\gamma \geq 5$, it is necessary to consider a Runge-Kutta method with strictly more than γ stages.

4 Multi-Step Methods for Solving ODEs

The Runge-Kutta methods introduced in Chapter 3 can be used to numerically approximate solutions to the initial value problem

$$\begin{aligned} u'(t) &= F\big(t, u(t)\big), \\ u(0) &= u_0. \end{aligned} \tag{4.1}$$

However, these Runge-Kutta schemes are all examples of a multi-stage **one-step** method. Indeed, we observe that only the approximate solution U_n at the time level t^n is required in order to compute the approximate solution U_{n+1} at the next time level t^{n+1}.

One potential problem with multi-stage numerical schemes such as the RK-4 method (3.7) is that a large number of function evaluations might be required in order to compute the approximate solution at each time step. In particular, this will be the case if the function u is a high-dimensional vector or the number of stages s is large.

An alternative approach for obtaining high-order numerical methods is to use **multi-step** methods. The basic idea behind multi-step methods is to compute the approximate solution $U_{n+\gamma}$ at the time level $t^{n+\gamma}$ using the approximate solutions $U_n, U_{n+1}, \ldots, U_{n+\gamma-1}$ at the previous γ time levels.

The simplest examples of such multi-step methods are the so-called *linear* multi-step methods of the form

$$\sum_{j=0}^{\gamma} \alpha_j U_{n+j} = \Delta t \sum_{j=0}^{\gamma} \beta_j F\big(t^{n+j}, U_{n+j}\big), \tag{4.2}$$

for the coefficients $\{\alpha_j\}_{j=0}^{\gamma}$ and $\{\beta_j\}_{j=0}^{\gamma}$. We observe that the γ-step method (4.2) provides a linear relation between the approximate solution values at γ time steps.

Special cases of the linear multi-step method (4.2) are the explicit multi-step methods obtained by setting the coefficient $\beta_\gamma = 0$. In general however, (4.2) results in an implicit method.

4.1 Adams Methods

The so-called Adams methods are a special class of the linear multi-step methods (4.2) obtained by setting the coefficients

$$\begin{aligned} \alpha_\gamma &= 1, \\ \alpha_{\gamma-1} &= -1, \\ \alpha_j &= 0, \qquad \forall\ j < \gamma - 1. \end{aligned}$$

Thus (4.2) takes the form

$$U_{n+\gamma} = U_{n+\gamma-1} + \Delta t \sum_{j=0}^{\gamma} \beta_j F\big(t^{n+j}, U_{n+j}\big). \tag{4.3}$$

Therefore, focusing on the special case of an autonomous ODE, i.e.,

$$F\big(t, u(t)\big) = F(u),$$

we obtain the following form of the Adams methods:

$$U_{n+\gamma} = U_{n+\gamma-1} + \Delta t \sum_{j=0}^{\gamma} \beta_j F(U_{n+j}). \tag{4.4}$$

4.2 Adams-Bashforth Methods

The explicit versions of the Adams methods (4.4) are obtained by setting the coefficient $\beta_\gamma = 0$ and are therefore of the form

$$U_{n+\gamma} = U_{n+\gamma-1} + \Delta t \sum_{j=0}^{\gamma-1} \beta_j F(U_{n+j}). \tag{4.5}$$

The explicit numerical methods given by (4.5) are known as Adams-Bashforth methods. The coefficients $\{\beta_j\}_{j=0}^{\gamma-1}$ in the Adams-Bashforth methods (4.5) can be computed in an appropriate manner to ensure the correct order of accuracy.

One possible approach to compute these coefficients is to consider the following calculation:

$$\begin{aligned} u(t^{n+\gamma}) - u(t^{n+\gamma-1}) &= \int_{t^{n+\gamma-1}}^{t^{n+\gamma}} u'(s)ds \\ &= \int_{t^{n+\gamma-1}}^{t^{n+\gamma}} F\big(u(s)\big)ds. \end{aligned} \tag{4.6}$$

The integral given by Equation (4.6) can then be approximated using numerical quadrature rules. In particular, we may approximate the function $F(u)$ using a polynomial $p(t)$ of degree $\gamma - 1$, interpolated from the approximate solution values at the time levels $t^n, t^{n+1}, \ldots, t^{n+\gamma-1}$, and then integrate the polynomial $p(t)$. This calculation then results in the following Adams-Bashforth methods for $\gamma = 1, 2, 3$:

$$\begin{aligned}
&\text{(AB1)} \qquad U_{n+1} = U_n + \Delta t F(U_n), \\
&\text{(AB2)} \qquad U_{n+2} = U_{n+1} + \tfrac{\Delta t}{2}\big(- F(U_n) + 3F(U_{n+1})\big), \\
&\text{(AB3)} \qquad U_{n+3} = U_{n+2} + \tfrac{\Delta t}{12}\big(5F(U_n) - 16F(U_{n+1}) + 23F(U_{n+2})\big).
\end{aligned}$$

We observe that the AB1 method is in fact the Forward Euler method (2.4).

4.3 Adams-Moulton Methods

The implicit version of the Adams methods (4.4) is obtained by setting the coefficient $\beta_\gamma \neq 0$. These implicit numerical schemes are known as the Adams-Moulton methods.

Once again, the coefficients $\{\beta_j\}_{j=0}^{\gamma}$ in the Adams-Moulton methods can be computed by repeating the calculation (4.6), approximating the function $F(u)$ with a polynomial $q(t)$ of degree γ, interpolating the approximate solution values at the time levels $t^n, t^{n+1}, \ldots, t^{n+\gamma}$, and then integrating the polynomial $q(t)$. Note that the interpolation here includes the values $U_{n+\gamma}$ and $t^{n+\gamma}$, in contrast to the Adams-Bashforth methods. The resulting methods are then $(\gamma+1)$-order accurate. Some examples of Adams-Moulton methods are given below:

$$\begin{aligned}
&\text{(AM1)} \quad U_{n+1} = U_n + \tfrac{\Delta t}{2}\big(F(U_n) + F(U_{n+1})\big), \\
&\text{(AM2)} \quad U_{n+2} = U_{n+1} + \tfrac{\Delta t}{12}\big(- F(U_n) + 8F(U_{n+1}) + 5F(U_{n+2})\big), \\
&\text{(AM3)} \quad U_{n+3} = U_{n+2} + \tfrac{\Delta t}{24}\big(F(U_n) - 5F(U_{n+1}) + 19F(U_{n+2}) + 9F(U_{n+3})\big).
\end{aligned}$$

We observe that the AM1 method is in fact the Trapezoidal rule (2.9).

4.4 Truncation Error

The truncation error associated with a linear multi-step method of the form (4.2) is defined as

$$\begin{aligned} T_{n+\gamma} &= \frac{1}{\Delta t}\Big(\sum_{j=0}^{\gamma}\alpha_j u(t^{n+j}) - \Delta t\sum_{j=0}^{\gamma}\beta_j F(u(t^{n+j}))\Big) \\ &\underbrace{=}_{(4.1)} \frac{1}{\Delta t}\Big(\sum_{j=0}^{\gamma}\alpha_j u(t^{n+j}) - \Delta t\sum_{j=0}^{\gamma}\beta_j u'(t^{n+j})\Big). \end{aligned} \tag{4.7}$$

Taylor Expansions then imply that for all time levels it holds that

$$\begin{aligned} u(t^{n+j}) &= u(t^n) + j\Delta t u'(t^n) + \tfrac{j^2(\Delta t)^2}{2}u''(t^n) + \ldots + \tfrac{j^k(\Delta t)^k}{k!}u^{(k)}(t^n) + \ldots, \\ u'(t^{n+j}) &= u'(t^n) + j\Delta t u''(t^n) + \tfrac{j^2(\Delta t)^2}{2}u'''(t^n) + \ldots + \tfrac{j^k(\Delta t)^k}{k!}u^{(k+1)}(t^n) + \ldots. \end{aligned}$$

Therefore, substituting these expressions into Equation (4.7) and collecting terms we obtain

$$\begin{aligned} T_{n+\gamma} = \frac{1}{\Delta t}\Big(\sum_{j=0}^{\gamma}\alpha_j\Big)u(t^n) + \Big(\sum_{j=0}^{\gamma} j(\alpha_j - \beta_j)\Big)u'(t^n) \\ + \Delta t\Big(\sum_{j=0}^{\gamma}\big(\tfrac{j^2\alpha_j}{2} - j\beta_j\big)\Big)u''(t^n) + \ldots \\ + (\Delta t)^{k-1}\Big(\sum_{j=0}^{\gamma}\big(\tfrac{j^k\alpha_j}{k!} - \tfrac{j^{k-1}\beta_j}{(k-1)!}\big)\Big)u^{(k)}(t^n) + \ldots. \end{aligned}$$

In order to ensure consistency, me must impose the conditions

$$\begin{aligned} \sum_{j=0}^{\gamma}\alpha_j &= 0, \\ \sum_{j=0}^{\gamma} j\alpha_j &= \sum_{j=0}^{\gamma}\beta_j. \end{aligned}$$

Finally, a truncation error of order $(\Delta t)^k$ is obtained by setting

$$\sum_{j=0}^{\gamma}\frac{j^q}{q!}\alpha_j = \sum_{j=0}^{\gamma}\frac{j^{q-1}}{(q-1)!}\beta_j, \qquad \forall\ q \le k+1.$$

4.5 Starting Values

Clearly, a linear γ-step numerical method of the form (4.2) requires γ starting values $U_0, U_1, \ldots, U_{\gamma-1}$ in order to be implemented as a time marching scheme. The initial condition of the IVP (4.1) allows us to specify U_0 but we require some concrete method of specifying the other starting values.

The usual strategy to specify the remaining starting values $U_1, \ldots, U_{\gamma-1}$ is to use a Runge-Kutta method of order $\gamma - 1$ in the case of an explicit Adams-Bashforth method, or to use a Runge-Kutta method of order γ in the case of an implicit Adams-Moulton method.

We observe that in both cases, we employ an RK method with order of accuracy exactly one less than that of the multi-step method. To observe why it is sufficient to use an RK method of such order of accuracy, consider the case of an explicit γ-step Adams-Bashforth method. We then employ an RK method of order $\gamma - 1$. This leads to a one-step error of order γ. Therefore, the total error due to the first γ steps (assuming that the method is stable) will be of order $(\gamma - 1) \cdot \mathcal{O}\big((\Delta t)^\gamma\big)$ and therefore the global error of the multi-step method remains $\mathcal{O}\big((\Delta t)^\gamma\big)$.

4.6 Concluding Remarks

We end this section by listing some advantages and disadvantages of using multi-step methods to approximate solutions to the IVP (4.1).

- The major advantage of using explicit multi-step methods such as Adams-Bashforth methods is that the right-hand side function F only needs to be evaluated once at each time step. In contrast, Runge-Kutta methods generally require multiple function evaluations at each time step which adds to the computational complexity of such methods.
- On the other hand, a significant disadvantage of using multi-step methods is that variable time-steps are difficult to implement. In addition, as mentioned previously multi-step methods require several starting values in order to be implemented as a time-marching scheme.

For further examples of multi-step methods, see Section 5.8 on the so-called BDF methods.

5 Stability of Numerical Methods for ODEs

Consider the following standard first-order initial value problem for ODEs:

$$\begin{aligned} u'(t) &= F(t, u(t)), \\ u(0) &= u_0. \end{aligned} \tag{5.1}$$

All the numerical methods discussed so far compute the approximate solution value U_N of the IVP (5.1) at the time level $t^N = N\Delta t = T$.

A numerical method is said to converge if it holds that

$$\lim_{\substack{\Delta t \to 0, \\ N \cdot \Delta t = T}} U_N = u(T) \tag{5.2}$$

where u is the exact solution of the IVP (5.1).

Clearly, for a one-step method, the starting value coincides with the initial condition u_0 of the IVP (5.1). On the other hand, for a γ-step method, we also have to account for the starting values $U_1, U_2, \ldots, U_{\gamma-1}$. Therefore, we require a γ-step method to satisfy the condition

$$\lim_{\Delta t \to 0} U_j(\Delta t) = u_0, \qquad \forall \;\; 0 \leq j \leq \gamma - 1. \tag{5.3}$$

In light of this discussion, we have the following definition:

Definition 5.1 (Convergent Numerical Method) *A numerical method for approximating solutions to the IVP* (5.1) *is said to be* ***convergent*** *if the computed solution U_N satisfies conditions* (5.2) *and* (5.3) *for every fixed, final time $T > 0$.*

Clearly, convergence is, at the very least, a key requirement for a 'good' numerical method.

5.1 Convergence of Forward Euler for Linear ODEs

We examine the question of convergence in the case of a linear, scalar IVP:

$$\begin{aligned} u'(t) &= \lambda u(t) + g(t), \\ u(0) &= u_0 \end{aligned} \tag{5.4}$$

where $\lambda \in \mathbb{R}$ is some constant.

The simplest numerical method for approximating solutions to the IVP (5.4) is the Forward Euler method (2.4):

$$\begin{aligned} U_{n+1} &= U_n + \Delta t\big(\lambda U_n + g(t^n)\big) \\ &= \big(1 + \lambda\Delta t\big)U_n + \Delta t g(t^n), \\ U_0 &= u_0. \end{aligned} \tag{5.5}$$

It is immediately clear that the condition (5.2) is satisfied by the Forward Euler method (5.5).

Our aim is to calculate the error given by

$$E_N = u(t^N) - U_N.$$

We recall from Chapter 2 that the truncation error associated with the Forward Euler method (5.5) is defined as

$$T_n = \frac{u(t^{n+1}) - u(t^n)}{\Delta t} - \lambda u(t^n) - g(t^n). \tag{5.6}$$

Exercise 5.2 *Show that the truncation error* (5.6) *associated with the Forward Euler method* (5.5) *can be written as*

$$T_n = \frac{\Delta t}{2}u''(t^n) + \mathcal{O}\big((\Delta t)^3\big). \tag{5.7}$$

Next, we rewrite (5.6) as

$$u(t^{n+1}) = \big(1 + \lambda\Delta t\big)u(t^n) + \Delta t g(t^n) + \Delta t T_n. \tag{5.8}$$

Equations (5.5) and (5.8) together then imply that

$$u(t^{n+1}) - U_{n+1} = \big(1 + \lambda\Delta t\big)\big(u(t^n) - U_n\big) + \Delta t T_n \tag{5.9}$$

and therefore using the definition of the error E_n we obtain the following:

$$E_{n+1} = \big(1 + \lambda\Delta t\big)E_n + \Delta t T_n. \tag{5.10}$$

Thus, the error at a given time level depends on the error at the previous time level and the truncation error.

Next, observe that we can rewrite (5.10) as

$$\begin{aligned} E_n &= (1+\lambda\Delta t)E_{n-1} + \Delta t T_{n-1} \\ &= (1+\lambda\Delta t)\big((1+\lambda\Delta t)E_{n-2} + \Delta t T_{n-2}\big) + \Delta t T_{n-1} \\ &= (1+\lambda\Delta t)^2 E_{n-2} + \Delta t(1+\lambda\Delta t)T_{n-2} + \Delta t T_{n-1}. \end{aligned} \tag{5.11}$$

Repeating this argument N times, we obtain

$$E_N = (1+\lambda\Delta t)^N E_0 + \Delta t \sum_{m=1}^{N} (1+\lambda\Delta t)^{N-m} T_{m-1}. \tag{5.12}$$

Equation (5.12) clearly indicates the contribution of the local truncation error to the global error. Indeed, we observe that the local truncation error at each time level t^m contributes an error of $(1+\lambda\Delta t)^{N-m}T_{m-1}$ to the global error. Next, observe that for all $\Delta t > 0$, it holds that

$$\begin{aligned} |1+\lambda\Delta t| &\leq e^{|\lambda|\Delta t} \\ \implies |1+\lambda\Delta t|^N &\leq e^{N|\lambda|\Delta t} = e^{|\lambda|T}, \qquad \text{where } N\Delta t = T. \end{aligned}$$

Similarly, for all $m < N$ it holds that

$$|1+\lambda\Delta t|^{N-m} \leq e^{(N-m)|\lambda|\Delta t} \leq e^{N|\lambda|\Delta t} \leq e^{|\lambda|T}. \tag{5.13}$$

Therefore, we obtain the following upper bound for the global error (5.12):

$$\begin{aligned} |E_N| &\leq |1+\lambda\Delta t|^N |E_0| + \Delta t \sum_{m=1}^{N} |1+\lambda\Delta t|^{N-m} |T_{m-1}| \\ &\leq e^{|\lambda|T}\Big(|E_0| + \Delta t \sum_{m=1}^{N} \max_m |T_{m-1}|\Big). \end{aligned}$$

Next, let

$$\|\tau\|_\infty = \max_{0\leq m\leq N-1} |T_m|.$$

Then it holds that

$$|E_N| \leq e^{|\lambda|T}\Big(|E_0| + T\|\tau\|_\infty\Big).$$

Note that for the Forward Euler method (5.5) it holds that

$$\|\tau\|_\infty = \max_{1\leq m\leq N-1} |T_m| \approx \tfrac{\Delta t}{2}\|u''\|_\infty = \mathcal{O}(\Delta t).$$

Finally, we observe that since $E_0 = 0$, the global error of the Forward Euler method is bounded by

$$|E_N| \leq Te^{|\lambda|T}\mathcal{O}(\Delta t),$$

and therefore

$$\lim_{\Delta t \to 0} E_N \to 0.$$

Hence, the Forward Euler method indeed satisfies condition (5.2) and is therefore **convergent**. Furthermore, $|E_N| = \mathcal{O}(\Delta t)$ implies that the Forward Euler is a first-order accurate method.

5.2 Convergence of Forward Euler for Non-Linear ODEs

We next consider the case of a non-linear, autonomous IVP:

$$\begin{aligned} u'(t) &= F(u(t)), \\ u(0) &= u_0. \end{aligned} \tag{5.14}$$

In order to ensure well-posedness of solutions to the IVP (5.14), we must assume that the function F is Lipschitz continuous. Once again we use the Forward Euler method (2.4) to approximate solutions to the IVP (5.14) and obtain

$$\begin{aligned} U_{n+1} &= U_n + \Delta t F(U_n), \\ U_0 &= u_0. \end{aligned} \tag{5.15}$$

The truncation error associated with the Forward Euler method (5.15) is then given by

$$T_n = \frac{u(t^{n+1}) - u(t^n)}{\Delta t} - F(u(t^n)). \tag{5.16}$$

and therefore it holds that

$$u(t^{n+1}) = u(t^n) + \Delta t F(u(t^n)) + \Delta t T_n. \tag{5.17}$$

Equations (5.14) and (5.17) together then imply that

$$u(t^{n+1}) - U_{n+1} = u(t^n) - U_n + \Delta t\big(F(u(t^n)) - F(U_n)\big) + \Delta t T_n$$

and therefore using the definition of the error E_n we obtain the following:

$$E_{n+1} = E_n + \Delta t\big(F(u(t^n)) - F(U_n)\big) + \Delta t T_n.$$

We thus obtain the following upper bound for the error:

$$\|E_{n+1}\| \leq \|E_n\| + \Delta t\|F(u(t^n)) - F(U_n)\| + \Delta t\|T_n\| \tag{5.18}$$

where $\|\cdot\|$ is a vector norm.

Next, since the function F is Lipschitz continuous, there exists some constant $L \in \mathbb{R}$ such that

$$\|F(u(t^n)) - F(U_n)\| \leq L\|u(t^n) - U_n\| \leq L\|E_n\|.$$

Hence, it holds that

$$\|E_{n+1}\| \leq \big(1 + \Delta t L\big)\|E_n\| + \Delta t\|T_n\|. \tag{5.19}$$

Therefore, the error associated with the Forward Euler method (5.15) at the time level t^n is bounded by

$$\|E_n\| \leq \big(1 + \Delta t L\big)\|E_{n-1}\| + \Delta t\|T_{n-1}\|. \tag{5.20}$$

We observe that Inequality (5.20) is extremely similar to Equation (5.10). Indeed, apart from the fact that (5.20) is an inequality and makes use of a vector norm, the two expressions are virtually identical. We therefore proceed in a similar fashion as before and iterate Inequality (5.20) N times to obtain

$$\|E_N\| \leq \big(1 + \Delta t L\big)^N\|E_0\| + \Delta t \sum_{m=1}^{N} \big(1 + \Delta t L\big)^{N-m}\|T_{m-1}\|. \tag{5.21}$$

Next, the bound (5.13) and Equation (5.21) together imply that

$$\|E_N\| \leq e^{LT}\Big(\|E_0\| + T\|\tau\|_\infty\Big) \tag{5.22}$$

where

$$\|\tau\|_\infty = \max_{0 \leq m \leq N-1} \|T_m\|.$$

Finally using the fact that $E_0 = 0$ and $\|\tau\|_\infty = \mathcal{O}(\Delta t)$ we obtain

$$\|E_N\| \leq T e^{LT} \cdot \mathcal{O}(\Delta t).$$

Therefore, it holds that

$$\lim_{\Delta t \to 0} E_N = 0$$

and therefore the Forward Euler method is convergent in the sense of (5.2) for the general IVP (5.14).

5.3 Convergence of Consistent One-Step Methods

It turns out that explicit one-step methods for approximating the IVP (5.1) can be written in the following general form:

$$\begin{aligned} U_{n+1} &= U_n + \Delta t \Phi(U_n, t^n, \Delta t), \\ U_0 &= u_0. \end{aligned} \tag{5.23}$$

Example 5.3 *Consider the 2-stage standard Runge-Kutta method* (3.5)*. Recall that the RK2 method can be written in the form* (3.6)*:*

$$\begin{aligned} U_{n+1} &= U_n + \Delta t F\big(t^n + \tfrac{\Delta t}{2}, U_n + \tfrac{\Delta t}{2} F(t^n, U_n)\big), \\ U_0 &= u_0. \end{aligned}$$

Hence, the RK2 method can be written in the form (5.23) *with*

$$\Phi(U_n, t^n, \Delta t) = F\big(t^n + \tfrac{\Delta t}{2}, U_n + \tfrac{\Delta t}{2} F(t^n, U_n)\big).$$

We define a numerical method of the form (5.23) to be consistent if it holds for all time t that

$$\Phi\big(u(t), t, 0\big) = F\big(t, u(t)\big).$$

Exercise 5.4 *Show that the 2-stage standard Runge-Kutta method* (3.5) *is consistent.*

We can now readily define the truncation error of a consistent one-step method as

$$T_n := \frac{u(t^{n+1}) - u(t^n)}{\Delta t} - \Phi(u(t^n), t^n, \Delta t). \tag{5.24}$$

We assume that the function Φ is Lipschitz in u. Then Equations (5.23) and (5.24) together imply that

$$u(t^{n+1}) - U_{n+1} = u(t^n) - U_n + \Delta t \Big(\Phi(u(t^n), t^n, \Delta t) - \Phi(U_n, t^n, \Delta t) \Big) + \Delta t T_n.$$

Lipschitz continuity of the function Φ then implies that

$$\|\Phi(u(t^n), t^n, \Delta t) - \Phi(U_n, t^n, \Delta t)\| \leq L \|u(t^n) - U_n\|,$$

and therefore the error E_n associated with the numerical method (5.23) satisfies the bound

$$\|E_{n+1}\| \leq \big(1 + \Delta t L\big) \|E_n\| + \Delta t \|T_n\|,$$

which is identical to the inequality bound (5.19).

Hence, it holds that

$$\|E_N\| \leq Te^{LT} \cdot \mathcal{O}(\Delta t),$$

and therefore,

$$\lim_{\Delta t \to 0} E_N = 0.$$

This establishes that general explicit one-step methods of the form (5.23) are convergent if they are consistent. We remark that the convergence of multi-step methods is considerably more difficult to establish and is outside the scope of these notes. In addition, note that a truncation error of $\mathcal{O}\big((\Delta t)^p\big)$ leads to $|E_N| = \mathcal{O}\big((\Delta t)^p\big)$, and thus the numerical method is p^{th}-order accurate.

5.4 Why Convergence is Not Enough

Convergence in the sense of (5.2) is merely a necessary condition for a 'good' numerical method but is by no means a sufficient condition. The following example helps illustrate this.

Example 5.5 *Consider the scalar IVP*

$$\begin{aligned} u'(t) &= \lambda\big(u(t) - \sin(t)\big) + \cos(t), \\ u(0) &= 0, \end{aligned} \tag{5.25}$$

where $\lambda \in \mathbb{R}$ is a constant.

Exercise 5.6 *Show that $u(t) = \sin(t)$ is the unique solution of the IVP (5.25) for any value of the constant λ.*

We compute the numerical solution of the IVP (5.25) using the Forward Euler method (2.4) for $\lambda = -100$ and different values of the time step Δt up to the final time $T = 10$. Our results are displayed in Table 5.1.
Clearly, the global error associated with the Forward Euler method for different values of Δt is very large. Indeed, Figure 5.1 displays the results for $N = \frac{T}{\Delta t} = 300$ points and indicates that the approximate solution contains very large oscillations and seems to blow up.
The behaviour of the Forward Euler method is very surprising considering that we have shown that this method produces approximate solutions that converge to the exact solution as $\Delta t \to 0$. Interestingly, this result is still true for very small values of the time step Δt. Indeed Figure 5.2 displays the results for $N = \frac{\Delta t}{T} = 400$ points and indicates that the approximate

N	Error	$\lvert 1 + \lambda\Delta t \rvert$
100	6.70×10^{66}	5.283
200	1.04×10^{59}	2.141
300	1.79×10^{5}	1.094
320	2.29×10^{-6}	0.964
400	3.74×10^{-7}	0.571

Table 5.1: Error Table for the Forward Euler method for the stiff IVP (5.25).

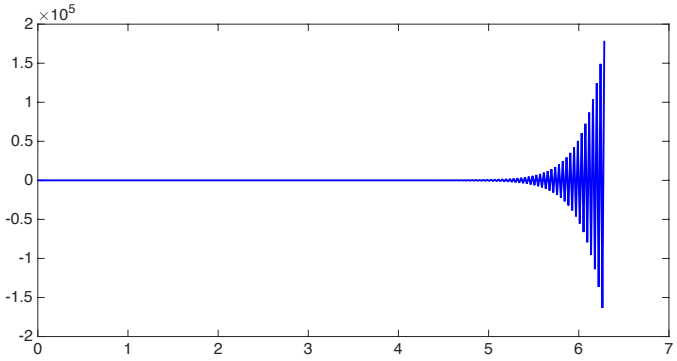


Figure 5.1: Approximate solution produced by the Forward Euler method for the stiff IVP (5.25) using N=300 points.

solution in this case is very close to the exact solution as evidenced by a global error of 3.74×10^{-7}.

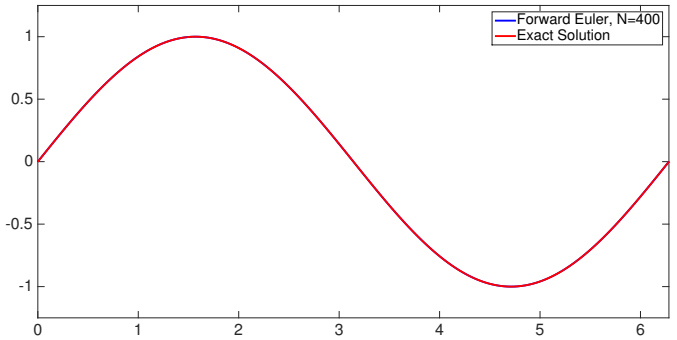


Figure 5.2: Approximate solution produced by the Forward Euler method for the stiff IVP (5.25) using N=400 points.

Table 5.1 contains a clue pertaining to this behaviour. Based on our numerical experiments it seems that the value $\|1 + \lambda\Delta t\|$ plays a significant role

in the value of the global error associated with the Forward Euler method as Δt is varied.

Finally, we remark that the Backward Euler method is able to produce accurate solutions to the IVP (5.25) even for very small values of the number of points N, as shown in Figure 5.3.

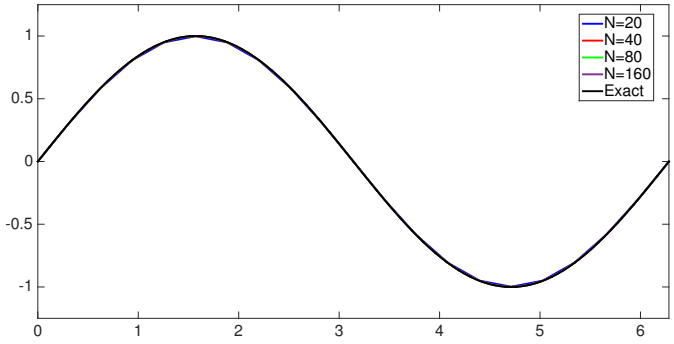


Figure 5.3: Approximate solutions produced by the Backward Euler method for the stiff IVP (5.25).

5.5 Absolute Stability

The discussion in the previous sections provides motivation for a stronger notion of stability for numerical methods. Consider the following model IVP:

$$\begin{aligned} u'(t) &= \lambda u(t), \\ u(0) &= u_0 \end{aligned} \tag{5.26}$$

where $\lambda \in \mathbb{R}^-$ is a negative constant. Note that the exact solution of the IVP (5.26) is given by $u(t) = u_0 e^{\lambda t}$ and therefore the solution will decay rapidly to zero for any initial condition as long as $\lambda < 0$.

Applying the Forward Euler method (2.4) to approximate solutions to the IVP (5.26) leads to

$$U_{n+1} = \big(1 + \lambda \Delta t\big) U_n. \tag{5.27}$$

We say that the Forward Euler method (2.4) is absolutely stable if it holds that

$$|1 + \lambda \Delta t| \leq 1, \tag{5.28}$$

or equivalently if it holds that

$$|U_{n+1}| \leq |U_n|. \tag{5.29}$$

Remark 5.7 *We observe that this notion of stability only makes sense when the constant $\lambda \leq 0$ as the exact solution in this case is monotonically decreasing.*

Equation (5.28) implies that the time step Δt must be chosen such that

$$-2 \leq \lambda \Delta t \leq 0 \quad \text{(See Figure 5.4)}$$

Hence, the Forward Euler method (2.4) is absolutely stable only if the time step Δt, relative to the constant λ, is sufficiently small. This provides justification for the results of our numerical experiments in Section 5.4.

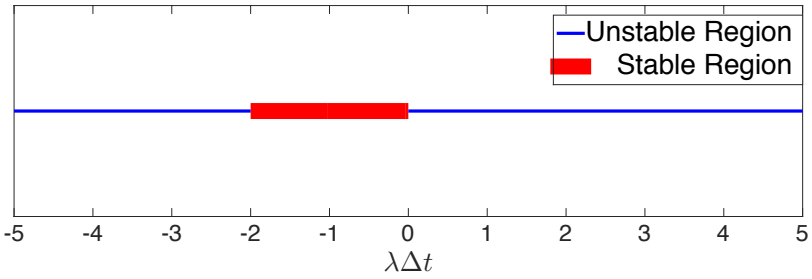


Figure 5.4: Stable and Unstable regions of $\lambda \Delta t$ for the Forward Euler scheme.

5.5.1 Absolute Stability of Backward Euler Method

Applying the Backward Euler method (2.6) to approximate solutions to the IVP (5.26) leads to

$$\frac{U_{n+1} - U_n}{\Delta t} = \lambda U_{n+1}$$

$$\implies U_n = \big(1 - \lambda \Delta t\big) U_{n+1}$$

$$\implies U_{n+1} = \frac{1}{1 - \lambda \Delta t} U_n.$$

Thus, the Backward Euler method (2.6) is absolutely stable if Equation (5.29) holds, i.e., if it holds that

$$\frac{1}{|1 - \lambda \Delta t|} \leq 1, \tag{5.30}$$

or equivalently if it holds that

$$\lambda \Delta t \in (-\infty, 0] \cup [2, \infty) \quad \text{(See Figure 5.5).} \tag{5.31}$$

The large stability region for the Backward Euler scheme given by (5.31) indicates that the Backward Euler scheme remains stable even for large values of the time step Δt. This explains why the Backward Euler method performs well in the numerical experiments considered in the previous sections.

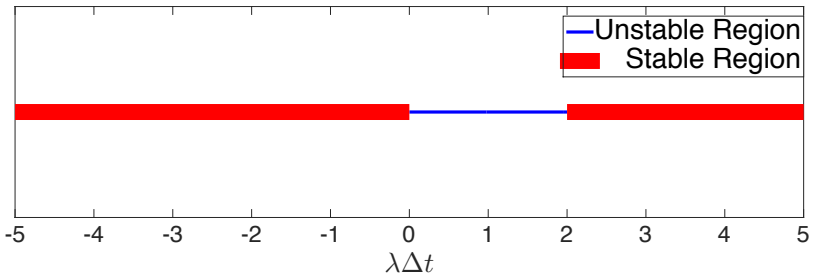


Figure 5.5: Stable and Unstable regions of $\lambda\Delta t$ for the Backward Euler scheme.

5.5.2 Absolute Stability of Trapezoidal Rule

Applying the Trapezoidal rule (2.9) to approximate solutions to the IVP (5.26) leads to

$$\frac{U_{n+1} - U_n}{\Delta t} = \tfrac{1}{2}\big(\lambda U_n + \lambda U_{n+1}\big)$$

$$\implies \big(1 - \tfrac{\lambda\Delta t}{2}\big)U_{n+1} = \big(1 + \tfrac{\lambda\Delta t}{2}\big)U_n$$

$$\implies U_{n+1} = \frac{1 + \frac{\lambda\Delta t}{2}}{1 - \frac{\lambda\Delta t}{2}} U_n.$$

Thus, the Trapezoidal rule (2.9) is A-stable if Equation (5.29) holds, i.e., if it holds that

$$\left|\frac{1 + \frac{\lambda\Delta t}{2}}{1 - \frac{\lambda\Delta t}{2}}\right| \leq 1. \tag{5.32}$$

Exercise 5.8 *Show that a sufficient condition for the stability Equation* (5.32) *to hold is that* $\lambda\Delta t \in (-\infty, 0]$.

Stability regions for more complicated Runge-Kutta and multi-step methods can similarly be computed.

5.6 Absolute Stability of Systems of ODEs

Consider the linear system of ODEs

$$u'(t) = Au(t), \tag{5.33}$$

where $A \in \mathbb{R}^{m\times m}$ is a constant diagonalisable matrix (see Section 1.6). Recall that the matrix A can then be written as

$$A = R\Lambda R^{-1},$$

where

$$\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m),$$

and

$$R = \left[\, r_1 \,\middle|\, r_2 \,\middle|\, \ldots \,\middle|\, r_m \,\right],$$

with $(\lambda_i, r_i),\ i = 1, \ldots, m$ the i^{th} eigenvalue and eigenvector of the matrix A respectively, i.e., for all $i = 1, \ldots, m$ it holds that

$$Ar_i = \lambda_i r_i.$$

Next, similar to Section 1.6, we may use the substitution $w = R^{-1}u$ to rewrite Equation (5.33) as

$$w'(t) = \Lambda w(t).$$

Thus the system of ODEs (5.33) decouples into m scalar ODEs of the form

$$w'(t) = \lambda_i w_i, \quad i = 1, \ldots, m. \tag{5.34}$$

Then, applying the Forward Euler method (2.4) to the system of ODEs (5.33) and using I to denote the $m \times m$ identity matrix, we obtain

$$\begin{aligned} U_{n+1} &= \big(\mathrm{I} + \Delta t A\big)U_n \\ \implies U_{n+1} &= \big(\mathrm{I} + \Delta t R\Lambda R^{-1}\big)U_n \\ \implies R^{-1}U_{n+1} &= R^{-1}U_n + \Delta t \Lambda R^{-1}U_n. \end{aligned}$$

Therefore, letting $W_n = R^{-1}U_n$, we obtain

$$W_{n+1} = W_n + \Delta t \Lambda W_n.$$

Since Λ is a diagonal $m\times m$ matrix, we obtain m-scalar, decoupled difference equations of the form

$$w^i_{n+1} = w^i_n + \Delta t \lambda_i w^i_n, \quad i = 1, \ldots, m. \tag{5.35}$$

Thus, absolute stability of a numerical method for approximating solutions to the system of ODEs (5.33) can be ensured by enforcing the following condition:

$$|w_{n+1}^i| \leq |w_n^i|, \quad \forall\ i = 1, \ldots, m. \tag{5.36}$$

For the Forward Euler method, the condition (5.36) is satisfied if for all $i = 1, \ldots, m$ it holds that

$$|1 + \Delta t \lambda_i| \leq 1 \tag{5.37}$$

Note however, that the eigenvalues $\{\lambda_i\}_{i=1}^m$ of the matrix A are, in general, complex-valued. Thus, stability regions for various numerical schemes are drawn in the complex plane. Some examples of such complex stability regions are given in Figure 5.6. Stability regions for more complicated Runge-Kutta and multi-step methods can also be drawn.

5.7 Stiff Problems

The discussion on stability in the previous section indicates that Forward Euler method can require the use of very small time steps, particularly if the absolute value of an eigenvalue of the system (5.33) is very large. On the other hand, the Backward Euler method and the Trapezoidal rule remain stable even for large values of the time step Δt.

A natural question to ask therefore is whether such problems involving very large (negative) eigenvalues actually occur. The simplest example of such problems is the model IVP (5.25) with a large negative value of the constant λ. We have already seen that the Forward Euler method and the Backward Euler method produce approximate solutions to the IVP (5.25) that are completely different.

A more practical example is provided by the chemical kinetics model of Section 1.3.5, which models the reactions

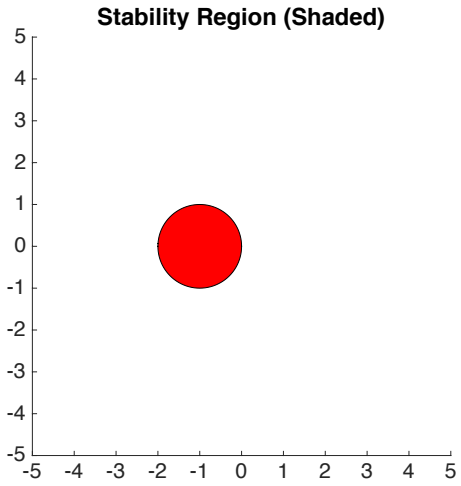
$$A \xrightarrow{\text{rate of reaction } k_1} B \xrightarrow{\text{rate of reaction } k_2} C.$$

and is given by the system of ODEs

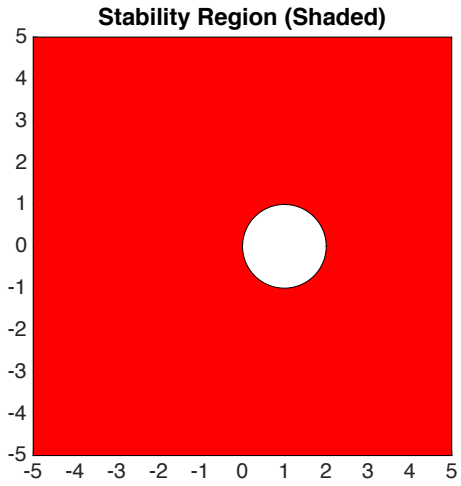
$$u'(t) = Au(t),$$

where

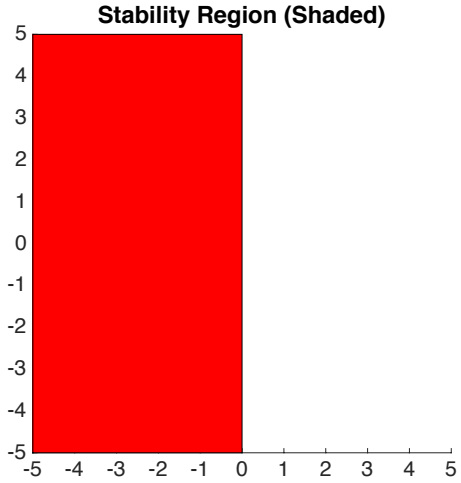
$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}, \qquad A = \begin{bmatrix} -k_1 & 0 & 0 \\ k_1 & -k_2 & 0 \\ 0 & k_2 & 0 \end{bmatrix}.$$



(a) Forward Euler method



(b) Backward Euler method



(c) Trapezoidal method

Figure 5.6: Stability regions of several methods

The eigenvalues of the matrix A are then given by $0, -k_1, -k_2$ and in many problems in chemistry, $k_1 \gg k_2$. For instance, we could have $k_1 = 10^6, k_2 = 1$. Such a problem is a classic example of so-called *stiff* problems.

In such a case, the Forward Euler method requires the use of a very small time step Δt of $\mathcal{O}(10^{-6})$, whereas the Backward Euler method works even for a large time step Δt of $\mathcal{O}(1)$.

Methods whose stability region contains the entire negative (real) half of the complex plane, i.e., $\{z \in \mathbb{C} \colon \mathrm{Re} z \leq 0\}$, are called *A-stable* numerical

methods. Examples of such methods include implicit schemes such as the Backward Euler method and the Trapezoidal method. A-stable methods are particularly well-suited for approximating solutions to stiff problems, as indicated by our results in the previous section.

5.8 BDF Methods

Higher order versions of the Backward Euler method are provided by the Backward difference formula (BDF) methods, which are of the general form

$$\alpha_0 U_n + \alpha_1 U_{n+1} + \ldots + \alpha_\gamma U_{n+\gamma} = \Delta t \beta_\gamma F(U_{n+\gamma}) \tag{5.38}$$

where $F(u) = u'$.

Thus, Equation (5.38) computes an approximation of the time derivative at the time level $t^{n+\gamma}$ using the approximate solution values at the previous γ time levels, $t^n, t^{n+1}, \ldots, t^{n+\gamma-1}$. This provides justification for the name *BDF* methods. Clearly, the Backward Euler method (2.6) is a BDF-1 method. Other examples of BDF methods include

BDF-2 $3U_{n+2} - 4U_{n+1} + U_n = 2\Delta t F(U_{n+2})$,

BDF-3 $11U_{n+3} - 18U_{n+2} + 9U_{n+1} - 2U_n = 6\Delta t F(U_{n+3})$.

BDF methods are well suited for stiff problems, particularly in chemistry.

6 The Poisson Equation

Let $\Omega \subseteq \mathbb{R}^n, (n = 1, 2, 3)$ be an open set with boundary $\partial\Omega$ as shown in Figure 6.1.

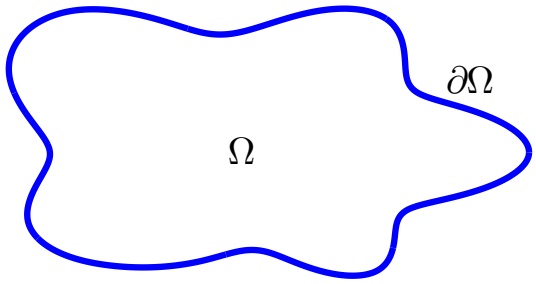


Figure 6.1: Example of the domain set Ω with boundary $\partial\Omega$.

Let $u\colon \Omega \to \mathbb{R}$ be an unknown scalar or vector function. We denote the derivatives of u in the following way:

$$\begin{aligned} \nabla u &= \left(\tfrac{\partial u}{\partial x_1}, \ldots, \tfrac{\partial u}{\partial x_n}\right), \\ \mathrm{D}^2 u &\quad \text{denotes the second derivative of } u, \\ \mathrm{D}^k u &\quad \text{denotes the } k^{th} \text{ derivative of } u. \end{aligned}$$

Then a **partial differential equation** (PDE) is an equation of the form

$$\mathbb{F}\left(x, u, \nabla u,\ \mathrm{D}^2 u,\ \mathrm{D}^k u, \ldots\right) = 0 \tag{6.1}$$

where $\mathbb{F}$ is some general, known function.

The task of solving Equation (6.1) therefore amounts to finding the unknown function u, given a complicated, non-linear relationship between a combination of its derivatives.

PDEs are ubiquitous in models in physics and engineering. We will study some of the most important PDEs that arise in engineering and design efficient numerical methods to solve them.

6.1 Derivation of Poisson's Equation

An important example of a PDE is the so-called Poisson's Equation given by

$$-\Delta u = f \tag{6.2}$$

where $u\colon \Omega \to \mathbb{R}$ is an unknown scalar-valued function, $f\colon \Omega \to \mathbb{R}$ is a given source function and Δ is the so-called **Laplace operator** given by

$$\Delta u = \sum_{i=1}^{n} u_{x_i x_i},$$

i.e., the trace of $\mathrm{D}^2 u$.

6.1.1 A Variational Principle

In many problems in physics, the mathematical model boils down to choosing one configuration of a system from amongst many possible configurations. The sought for configuration is usually a minimiser (or maximiser) for some variational problem as in the following example.

Consider an elastic body defined on a domain Ω. The elastic body can, for instance, be a membrane clamped at the boundary.

The unknown in this case is the vertical displacement $u = u(x), \quad x \in \Omega$. Furthermore, clamping at the boundary $\partial\Omega$ implies so-called Dirichlet boundary conditions

$$u|_{\partial\Omega} \equiv 0.$$

The total elastic energy can then be modelled by

$$J(u) = \frac{1}{2}\int_{\Omega} |\nabla u|^2 dx - \int_{\Omega} u f dx, \tag{6.3}$$

where f is some known, load function.

The sought for configuration of the system is a minimiser of the energy J given by (6.3). We remark that the function J is termed the Dirichlet energy.

As in calculus, we can calculate the minimiser of the energy J by solving the so-called Euler-Lagrange equations:

$$J'(u, v) = \lim_{\tau \to 0} \frac{J(u + \tau v) - J(u)}{\tau} = 0.$$

We therefore perform the following (formal) calculation:

$$\frac{J(u+\tau v)-J(u)}{\tau}=\frac{1}{\tau}\Big(\frac{1}{2}\int_\Omega|\nabla(u+\tau v)|^2dx-\frac{1}{2}\int_\Omega|\nabla u|^2dx-\tau\int_\Omega fvdx\Big).$$

Next, we use the fact that $|\nabla w|^2=\langle\nabla w,\nabla w\rangle$ where $\langle\cdot,\cdot\rangle$ is the usual inner product in $\mathbb{R}^n$, and we obtain

$$\begin{aligned}\frac{J(u+\tau v)-J(u)}{\tau}=&\frac{1}{\tau}\left(\frac{1}{2}\int_\Omega|\nabla u|^2dx+\tau\int_\Omega\langle\nabla u,\nabla v\rangle dx\right.\\&\left.+\frac{\tau^2}{2}\int_\Omega|\nabla v|^2dx-\frac{1}{2}\int_\Omega|\nabla u|^2dx-\tau\int_\Omega fvdx\right)\\=&\int_\Omega\langle\nabla u,\nabla v\rangle dx+\frac{\tau}{2}\int_\Omega|\nabla v|^2dx-\int_\Omega fvdx.\end{aligned}$$

Taking the limit we obtain

$$\lim_{\tau\to0}\frac{J(u+\tau v)-J(u)}{\tau}=\int_\Omega\langle\nabla u,\nabla v\rangle dx-\int_\Omega fvdx.$$

and hence,

$$J'(u,v)=\int_\Omega\langle\nabla u,\nabla v\rangle dx-\int_\Omega fvdx.$$

Therefore, the Euler-Lagrange equations are given by

$$\int_\Omega\langle\nabla u,\nabla v\rangle dx-\int_\Omega fvdx=0.$$

Next, using integration by parts we obtain

$$-\int_\Omega v\Delta udx-\int_\Omega vfdx+\int_{\partial\Omega}v\frac{\partial u}{\partial\nu}ds(x)=0,$$

where $\frac{\partial u}{\partial\nu}=\nabla u\cdot\nu$, and ν is the unit outward vector, normal to the boundary $\partial\Omega$.

Using the fact that any admissible configuration is clamped at the boundary, i.e., $v\equiv0$ on $\partial\Omega$, we therefore obtain

$$\int_\Omega(-\Delta u-f)vdx=0,\quad\forall v.$$

Therefore,

$$\begin{aligned} -\Delta u &= f, \\ u|_{\partial\Omega} &\equiv 0. \end{aligned} \tag{6.4}$$

We have thus derived Poisson's equation (6.2).

We remark that Poisson's equation (6.2) can also be derived as a steady state of the heat equation (see Chapter 10) as well as the potential flow equations of fluid dynamics.

6.2 The Poisson Equation in One-Space Dimension

In one space dimension, with the domain $\Omega = [0, 1]$, the Poisson Equation (6.4) takes the form

$$\begin{aligned} -u''(x) &= f(x), \quad \forall\ x \in (0, 1), \\ u(0) &= u(1) = 0. \end{aligned} \tag{6.5}$$

Note that Equation (6.5) is an example of a two-point boundary value problem (BVP) for ODEs.

It is possible to find an explicit formula for the solutions to the BVP (6.5). Indeed, using the Fundamental theorem of Calculus, we may write

$$\begin{aligned} u'(y) &= c_2 + \int_0^y u''(z)dz \qquad (c_2 \text{ is a constant}) \\ &= c_2 - \int_0^y f(z)dz \qquad (\text{using (6.5)}), \end{aligned}$$

and furthermore

$$\begin{aligned} u(x) &= c_1 + \int_0^x u'(y)dy \qquad (c_1 \text{ is a constant}) \\ &= c_1 + c_2 x - \int_0^x \int_0^y f(z)dzdy. \end{aligned} \tag{6.6}$$

We want to write this solution in a different form. Thus, let $F(y) = \int_0^y f(z)dz$ so that $F'(y) = f(y)$. Integration by parts then implies that

$$\begin{aligned} \int_0^x F(y)dy \overset{y'=1}{=} \int_0^x y'F(y)dy &= xF(x) - \int_0^x yF'(y)dy \\ &= x\int_0^x f(y)dy - \int_0^x yf(y)dy \\ &= \int_0^x (x - y)f(y)dy. \end{aligned}$$

Hence, Equation (6.6) can be rewritten as

$$u(x) = c_1 + c_2 x - \int_0^x (x - y) f(y) dy. \tag{6.7}$$

In addition, the constants c_1, c_2 can be determined using the boundary conditions in (6.5):

$$\begin{aligned} 0 &= u(0) = c_1, \\ 0 &= u(1) = c_2 - \int_0^1 (1 - y) f(y) dy \\ &\implies c_2 = \int_0^1 (1 - y) f(y) dy. \end{aligned}$$

Hence, the solution u of Equation (6.5) is given by

$$u(x) = \int_0^1 x(1 - y) f(y) dy - \int_0^x (x - y) f(y) dy. \tag{6.8}$$

Next, we define

$$G(x, y) = \begin{cases} y(1 - x) & 0 \leq y \leq x \\ x(1 - y) & x \leq y \leq 1, \end{cases} \tag{6.9}$$

and we can check that (6.8) is equivalent to

$$u(x) = \int_0^1 G(x, y) f(y) dy. \tag{6.10}$$

The function G in Equation (6.10) is termed a *Green's function* and provides an explicit formula for the solution to the 1-D Poisson equation (6.5).

6.2.1 Limitations of the Green's Function Representation

Unfortunately, the explicit formula (6.10) is not very useful due to the following reasons:

- The integral in (6.10) is not possible to evaluate exactly for complicated source (load) functions f. In such cases, a numerical quadrature rule would need to be used.

- A slight perturbation in the form of the Poisson equation may result in our inability to find a solution formula similar to (6.10). For instance, many applications use a modified version of the Poisson equation given by

$$\begin{aligned} -\big(a(x)u'(x)\big)'+b(x)u'(x)+c(x)u(x) &= f(x), \quad \forall\ x \in (0,1) \\ u(0) = u(1) &= 0, \end{aligned} \tag{6.11}$$

  where a, b, c are coefficient functions. It is not possible to find explicit formulae for solutions to (6.11) even in the simple case $b(x) \equiv 0$ and $c(x) = c$.

- Green's function representations are not available in the case of the two-dimensional and three-dimensional Poisson equation, except for very simple domains such as a ball.

6.3 Finite Difference Methods

Given the limitations of explicit solution formulae, we must resort to numerical methods to approximate solutions to the Poisson equation. The simplest numerical method is to approximate the 1-D Poisson equation (6.5) using the so-called finite difference method.

6.3.1 Discretising the domain

Let $\Delta x > 0$ and let $N = \frac{1}{\Delta x} - 1$. Then we discretise the domain $[0, 1]$ into $N + 2$ points by setting

$$x_0 = 0, \quad x_{N+1} = 1, \quad x_j = j\Delta x,\ j = 1, \ldots, N.$$

Figure 6.2 displays an example of the discretised domain.

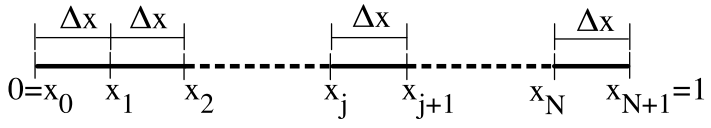


Figure 6.2: An example of a one-dimensional mesh for the domain $\Omega = (0, 1)$.

6.3.2 Discretising the Derivatives

Our aim will be to approximate the function u, which solves Equation (6.5), with point values, i.e., by setting

$$u_j \approx u(x_j),$$

and similarly, we define

$$f_j = f(x_j).$$

Hence, we need to approximate the derivatives that appear in Equation (6.5) with finite differences. The obvious choice is to use a simple central difference approximation of the second derivative of u given by

$$u''(x_j) \approx \frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta x^2}. \tag{6.12}$$

Exercise 6.1 *Check that for a sufficiently smooth function $u\colon \mathbb{R} \to \mathbb{R}$, there exists some constant $C \in \mathbb{R}$ such that for all $\Delta x > 0$ it holds that*

$$\left| u''(x_j) - \frac{u(x_j + \Delta x) - 2u(x_j) + u(x_j - \Delta x)}{\Delta x^2} \right| \leq C\Delta x^2. \tag{6.13}$$

6.3.3 The finite Difference Scheme

A finite difference scheme for approximating (6.5) is then given by

$$-u_{j+1} + 2u_j - u_{j-1} = \Delta x^2 f_j, \qquad \forall\ j = 1, \ldots, N.$$

Furthermore, using the boundary condition $u_0 = u(0) = 0$, we obtain

$$2u_1 - u_2 = \Delta x^2 f_1,$$

and similarly, using the boundary condition $u_{N+1} = u(1) = 0$, we obtain

$$-u_{N-1} + 2u_N = \Delta x^2 f_N.$$

Then, introducing the vectors

$$U = \big[u_1, u_2, \ldots, u_N\big]^\intercal, \qquad F = \Delta x^2 \big[f_1, f_2, \ldots, f_N\big]^\intercal,$$

we observe that the above finite difference scheme can be recast as the following matrix equation:

$$AU = F, \tag{6.14}$$

where A is the $N \times N$ matrix given by

$$A = \begin{bmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{bmatrix}.$$

6.3.4 Solving the Matrix Equation

The previous section indicates that the finite difference scheme for approximating solutions to the Poisson equation (6.5) reduces to solving the matrix equation (6.14). We observe that the matrix A is tridiagonal and diagonally dominant, and is therefore invertible. The matrix equation (6.14) can therefore be solved using methods learnt in numerical linear algebra.

6.4 Numerical Results

As a first numerical example, we consider the following two-point boundary value problem:

$$\begin{aligned} -u''(x) &= (3x + x^2)\exp(x), \\ u(0) &= u(1) = 0. \end{aligned} \tag{6.15}$$

It can be show that the exact solution to the BVP (6.15) is given by $u(x) = x(1-x)\exp(x)$. We use the one-dimensional finite difference method to approximate solutions to this problem for different values of the number of mesh points N.

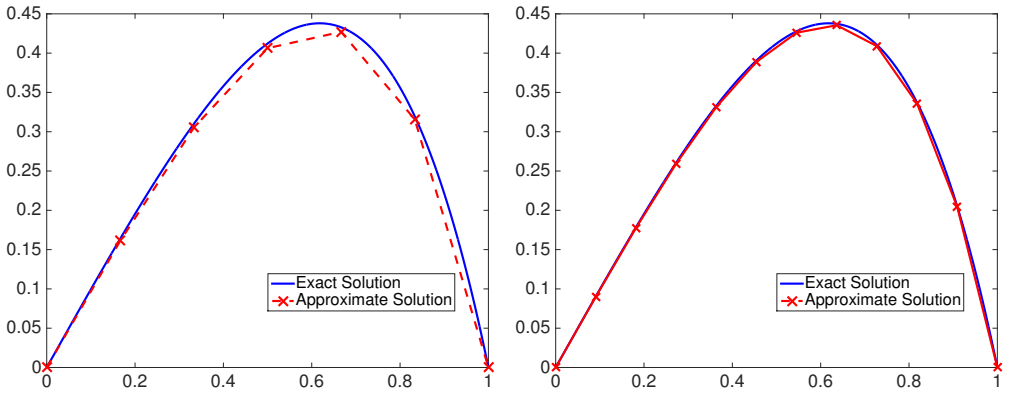
Figure 6.3 displays our results and indicates that the finite difference method can approximate solutions to the BVP (6.15) very well. Indeed, the associated errors given in Table 6.1 indicate that the approximate solutions seem to converge to the exact solution as the number of mesh points is increased.

In fact, using a discrete version of the Green's function representation, it can be proven that for any $\Delta x > 0$, the following stability estimate holds:

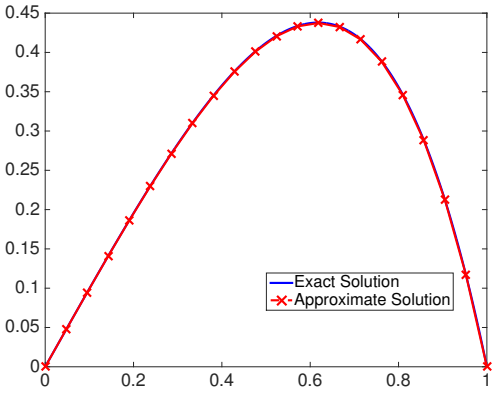
$$\|u^{\Delta x}\|_\infty \leq \frac{1}{8}\|f^{\Delta x}\|_\infty,$$

where

$$u^{\Delta x} = \big[u_1, u_2, \dots, u_N\big], \qquad f^{\Delta x} = \big[f_1, f_2, \dots, f_N\big],$$



(a) N=5. (b) N=10.



(c) N=20.

Figure 6.3: Exact solution and approximate solution plots for the finite difference method using different values of the number of mesh points N.

with $u^{\Delta x}$ solving the matrix equation (6.14) and the norm $\|\cdot\|_\infty$ is defined as

$$\|u^{\Delta x}\|_\infty = \max_{1\leq j\leq N} |u_j|,$$
$$\|f^{\Delta x}\|_\infty = \max_{1\leq j\leq N} |f_j|.$$

It is also possible to prove the following error estimate: Let

$$E_j = u(x_j) - u_j^{\Delta x}, \quad \forall\ j = 1, \ldots, N$$

where $u_j^{\Delta x}$ is the approximate solution produced by the finite difference scheme for a given value of Δx.

N	Error in Max-norm	EOC
5	5.890×10^{-3}	1.969
10	1.178×10^{-3}	1.996
20	4.911×10^{-4}	2.000
40	1.288×10^{-4}	2.000
80	3.302×10^{-5}	

Table 6.1: Error Table of the finite difference method for the BVP (6.15).

Next, let

$$E^{\Delta x} = [E_1, \ldots, E_N].$$

Then it holds that

$$\|E^{\Delta x}\|_\infty \leq \frac{\Delta x^2}{96} \max_{0 \leq x \leq 1} |f''(x)|. \tag{6.16}$$

The error estimate (6.16) therefore justifies the second order convergence observed in the numerical examples (see Table 6.1).

6.5 Finite Difference Schemes for the 2-D Poisson Equation

In this section, we consider the two-dimensional version of the Poisson equation on the unit square domain, i.e., with the domain $\Omega = (0,1)^2$:

$$\begin{aligned} -\big(u_{xx}(x) + u_{yy}(x)\big) &= f(x), \qquad \text{for } x \in \Omega = (0,1)^2, \\ u(x) &\equiv 0, \qquad \text{for } x \in \partial\Omega. \end{aligned} \tag{6.17}$$

We can now formulate a finite difference scheme for the Equation (6.17). Indeed, let $\Delta x, \Delta y > 0$ and and let $N = \frac{1}{\Delta x} - 1, M = \frac{1}{\Delta y} - 1$. We can then discretise the domain Ω into a set of $(N+2) \times (M+2)$ points by setting

$$\begin{aligned} x_i &= i\Delta x, \quad \forall\ 1 \leq i \leq N \\ x_0 &= 0, \quad x_{N+1} = 1, \\ y_j &= j\Delta y, \quad \forall\ 1 \leq j \leq M \\ y_0 &= 0, \quad y_{M+1} = 1. \end{aligned}$$

Figure 6.4 displays an example of the discretised two-dimensional domain.

The aim of a finite difference scheme is then to approximate

$$u_{ij} \approx u(x_i, y_j).$$

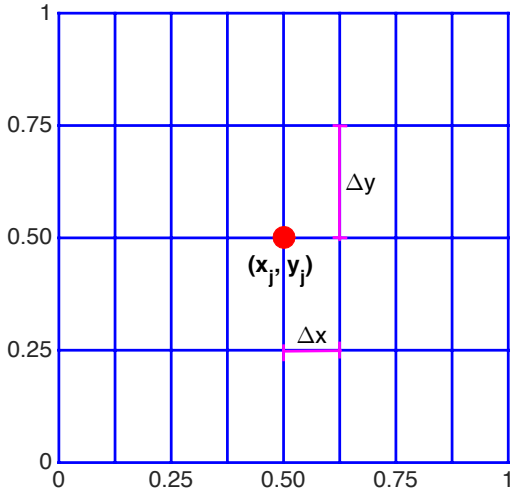


Figure 6.4: An example of a two-dimensional mesh for the domain $\Omega = (0,1)^2$ using $\Delta x = \frac{1}{8}, \Delta y = \frac{1}{4}$.

We similarly define

$$f_{ij} = f(x_i, y_j).$$

Once again, we discretise the Laplacian operator with a central difference approximation, i.e., we set

$$\begin{aligned} u_{xx}(x_i, y_j) &\approx \frac{u(x_i + \Delta x, y_j) - 2u(x_i, y_j) + u(x_i - \Delta x, y_j)}{\Delta x^2} \\ &\approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}, \end{aligned}$$

and similarly

$$\begin{aligned} u_{yy}(x_i, y_j) &\approx \frac{u(x_i, y_j + \Delta y) - 2u(x_i, y_j) + u(x_i, y_j - \Delta y)}{\Delta y^2} \\ &\approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}. \end{aligned}$$

Hence, a finite difference scheme for approximating (6.17) is given by

$$-\left(\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}\right) = f_{ij} \qquad (6.18)$$

for all $i = 1, \ldots, N$, $j = 1, \ldots, M$. In addition, the boundary conditions are imposed by specifying

$$\begin{aligned} u_{0,j} = u_{N+1,j} = 0, &\quad \forall\ j = 0, 1, \ldots, M+1, \\ u_{i,0} = u_{i,M+1} = 0, &\quad \forall\ i = 0, 1, \ldots, N+1. \end{aligned}$$

For simplicity, we may set $\Delta x = \Delta y$ and therefore $N = M$. Then, introducing the vectors

$$\begin{aligned} U &= \big[u_{1,1}, u_{2,1}, \ldots, u_{N,1}, u_{1,2}, u_{2,2}, \ldots, u_{N,2}, \ldots, u_{1,N}, u_{2,N}, \ldots, u_{N,N}\big]^\intercal, \\ F &= \Delta x^2 \big[f_{1,1}, f_{2,1}, \ldots, f_{N,1}, f_{1,2}, f_{2,2}, \ldots, f_{N,2}, \ldots, f_{1,N}, f_{2,N}, \ldots, f_{N,N}\big]^\intercal, \end{aligned}$$

we observe that the finite difference scheme (6.18) can be recast as the following matrix equation:

$$AU = F, \tag{6.19}$$

where A is the $N^2 \times N^2$ block matrix given by

$$A = \begin{bmatrix} B & -I & 0 & \ldots & 0 \\ -I & B & -I & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -I & B & -I \\ 0 & \ldots & 0 & -I & B \end{bmatrix}.$$

Here, I is the $N \times N$ identity matrix and B is the $N \times N$ matrix given by

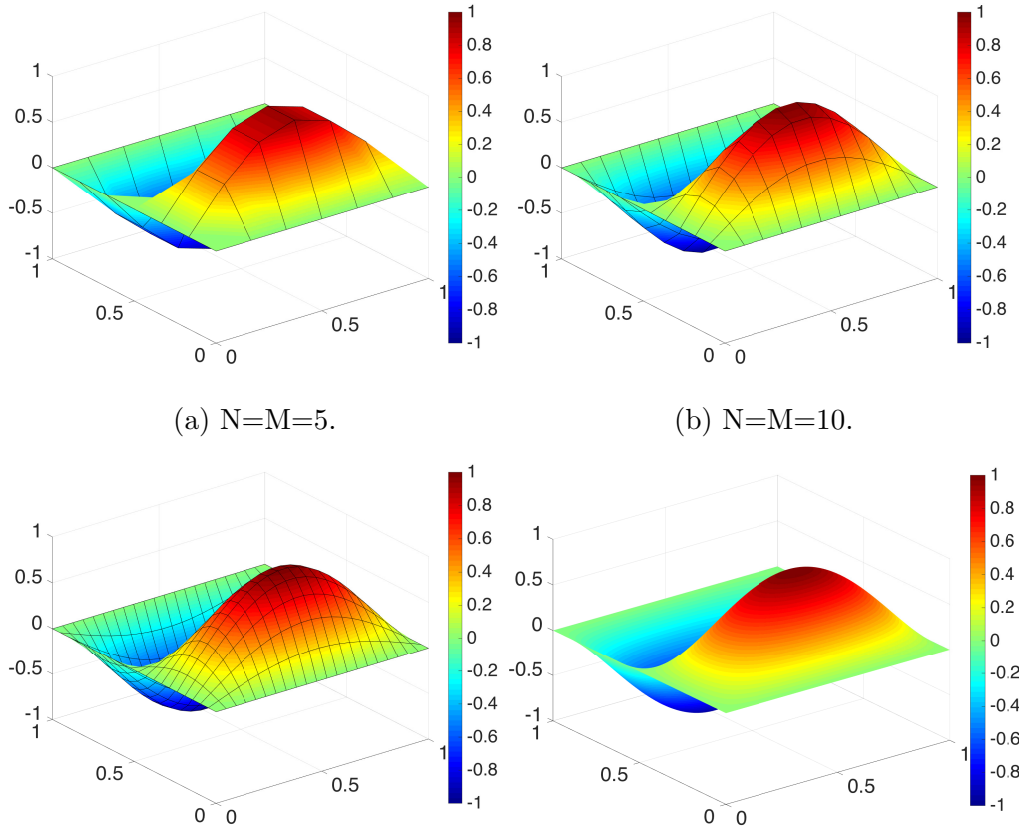
$$B = \begin{bmatrix} 4 & -1 & 0 & \ldots & 0 \\ -1 & 4 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 4 & -1 \\ 0 & \ldots & 0 & -1 & 4 \end{bmatrix}.$$

Finite difference schemes can also be defined for other simple two-dimensional domains such as rectangles. Unfortunately, it is significantly more complicated to define finite difference schemes for even slightly more complex geometries such as circles. We must therefore find an alternative strategy. One possibility is provided by the so-called *finite element methods*, which are considered in the next chapter.

6.5.1 Numerical Results in 2-D

We conclude this chapter by considering the following boundary value problem involving the two-dimensional Poisson equation:

$$\begin{aligned} -\big(u_{xx} + u_{yy}\big) &= 5\pi^2 \sin(\pi x)\sin(2\pi y), \\ u(0,y) = u(1,y) = u(x,0) = u(x,1) &= 0, \qquad \forall\ x, y \in [0,1]. \end{aligned} \tag{6.20}$$



(a) N=M=5. (b) N=M=10.

(c) N=M=20. (d) Exact Solution.

Figure 6.5: Exact solution and approximate solution plots for the two-dimensional finite difference method using different values of the number of interior mesh points in one direction $N = M$.

It can be show that the exact solution to the BVP (6.20) is given by $u(x, y) = \sin(\pi x)\sin(2\pi y)$. We use the two-dimensional finite difference method to approximate solutions to this problem for different values of the number of interior mesh points in each one direction $N = M$.

Figure 6.5 displays our results and indicates that the two-dimensional finite difference method can approximate solutions to the BVP (6.20) very well. Indeed, the associated errors given in Table 6.2 indicate that the approximate solutions seem to converge to the exact solution as the number of mesh points in each direction is increased. This conclusion is supported by Figure 6.6, which displays a logarithmic plot of the errors as a function of N. Furthermore, we observe once again that the experimental order of convergence EOC ≈ 2.

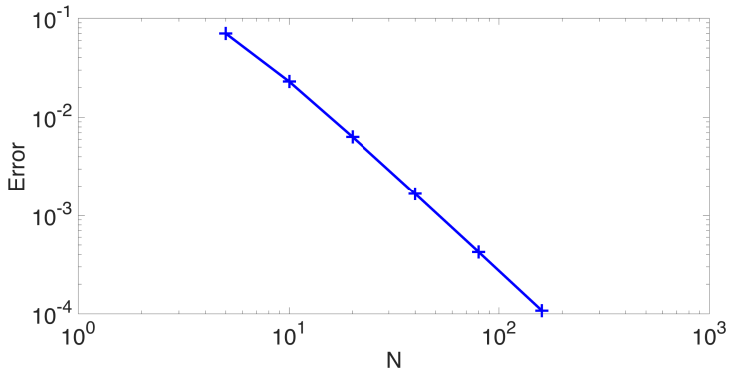


Figure 6.6: Logarithmic plot of the max-error norm for the approximate solutions vs. the number of interior mesh points in one direction $N = M$.

N=M	Error in Max-norm	EOC
5	7.023×10^{-2}	1.847
10	2.293×10^{-2}	1.991
20	6.328×10^{-3}	1.998
40	1.663×10^{-3}	1.999
80	4.262×10^{-4}	2.000
160	1.079×10^{-4}	

Table 6.2: Error Table of the two-dimensional finite difference method for the BVP (6.20).

Remark 6.2 *It is important to note that in the two-dimensional case, there is no longer a linear relationship between the number of interior mesh points in each direction, $N = M$, and the amount of computational work that needs to be done in order to obtain the finite difference approximation of the solution. Indeed, suppose our computational domain consists of N interior mesh points in each direction. Hence, our mesh consists of a total of N^2 interior mesh points. Thus, due to the two-dimensional nature of the problem, the amount of computational work required will be at least of the order $\mathcal{O}(N^2)$.*

Let us now assume that we double the number of interior mesh points in each direction. This results in a total of $2N \times 2N = 4N^2$ interior mesh points and hence, we must now perform 4 times as much computational work in order to approximate the solution (if we assume that the linear system is solved in an optimal way). This is in contrast to the one-dimensional case where the computational work is of the order $\mathcal{O}(N)$ and therefore doubling

the number of mesh points results in only twice the computational work. Therefore, in order to decrease the error by a factor of 4*, we must perform* 2 *times the computational work in a one-dimensional problem but* 4 *times the computational work in a two-dimensional problem.*

7 Finite Element Methods for the 1-D Poisson Equation

Finite element methods (FEM) are a powerful and heavily used alternative to the finite difference methods introduced in Chapter 6. We begin with a description of FEM in one space dimension.

Consider the one-dimensional Poisson equation given by

$$\begin{aligned} -u''(x) &= f(x), \quad \forall x \in (0,1), \\ u(0) &= u(1) = 0. \end{aligned} \tag{7.1}$$

In Chapter 6, we derived the Poisson equation as the Euler-Lagrange equations corresponding to the solutions of the following variational problem:

$$\min_u J(u), \tag{7.2}$$

where

$$J(u) = \frac{1}{2}\int_0^1 |u'(x)|^2 dx - \int_0^1 u(x)f(x)dx,$$

and we have assumed that $u(0) = u(1) = 0$.

We first study the variational problem (7.2) in more detail.

7.1 Variational Principles

The first question we must ask concerning the variational problem (7.2) is: in what set (class) of functions do we seek a minimiser of the Dirichlet energy J?

Clearly, the boundary conditions impose a restriction on the set of admissible functions in which a minimiser is sought. It is therefore natural to ask if there are other constraints as well.

Indeed, a natural constraint should be to restrict u to the class of functions for which the Dirichlet energy $J(u)$ is well-defined. This can be ensured if it holds that

$$\int_0^1 |u'(x)|^2 dx < \infty, \tag{7.3}$$

and

$$\left| \int_0^1 u(x) f(x) dx \right| < \infty. \tag{7.4}$$

In particular, the conditions (7.3) and (7.4) ensure that the Dirichlet energy $J(u)$ is mathematically well-defined. We can therefore narrow down the set of admissible functions by imposing (7.3) and (7.4).
We define the set

$$H_0^1\big([0,1]\big) := \Big\{ u\colon [0,1] \to \mathbb{R}\colon u(0) = u(1) = 0 \text{ and } \int_0^1 |u'(x)|^2 dx < \infty \Big\}.$$

Thus, $H_0^1\big([0,1]\big)$ is the set of all functions that vanish at the boundary and have the property that the integral of the square of their derivative is bounded.

We can also define a norm on $H_0^1\big([0,1]\big)$ by setting

$$\|u\|_{H_0^1([0,1])} := \Big(\int_0^1 |u'(x)|^2 dx \Big)^{1/2}.$$

Note that this norm is well defined for all $u \in H_0^1\big([0,1]\big)$ since condition (7.3) is automatically satisfied for all $u \in H_0^1\big([0,1]\big)$.

$H_0^1\big([0,1]\big)$ is a prototypical example of a *Sobolev space*. We now give some examples of functions that belong to the set $H_0^1\big([0,1]\big)$.

Example 7.1 *Let $u\colon [0,1] \to \mathbb{R}$ be the function given by*

$$u(x) = x(1-x).$$

Then, clearly $u(0) = u(1) = 0$. Furthermore, it holds that

$$\int_0^1 |u'(x)|^2 dx = \int_0^1 |1-2x|^2 dx \leq 2.$$

Hence, $u \in H_0^1\big([0,1]\big)$.

Example 7.2 *Let* $u\colon [0,1] \to \mathbb{R}$ *be the function given by*

$$u(x) = \begin{cases} x & \text{if } x \in [0, \frac{1}{2}], \\ 1 - x & \text{if } x \in (\frac{1}{2}, 1]. \end{cases}$$

Check that $u \in H_0^1\big([0,1]\big)$.

Clearly, condition (7.3) is automatically satisfied for all $u \in H_0^1\big([0,1]\big)$. A natural question to ask is if the condition (7.4) is also satisfied by such functions. Let us therefore consider the following calculation: Let $u \in H_0^1\big([0,1]\big)$. Then, the Cauchy-Schwarz inequality implies that

$$\begin{aligned} \left| \int_0^1 u(x) f(x) dx \right| &\leq \int_0^1 |u(x)||f(x)| dx \\ &\leq \Big(\int_0^1 |u(x)|^2 dx \Big)^{1/2} \cdot \Big(\int_0^1 |f(x)|^2 dx \Big)^{1/2}. \end{aligned} \tag{7.5}$$

To bound the above integrals, we continue by defining the set

$$L^2\big([0,1]\big) := \Big\{ g\colon [0,1] \to \mathbb{R} \colon \int_0^1 |g(x)|^2 dx < \infty \Big\}.$$

and we define a norm on the set $L^2\big([0,1]\big)$ by setting

$$\|g\|_{L^2([0,1])} = \Big(\int_0^1 |g(x)|^2 dx \Big)^{1/2}.$$

Hence, under the assumption that both functions $u, f \in L^2\big([0,1]\big)$, Inequality (7.5) implies that

$$\left| \int_0^1 u(x) f(x) dx \right| \leq \|u\|_{L^2([0,1])} \cdot \|f\|_{L^2([0,1])} < \infty.$$

Therefore, the Dirichlet energy J given by (7.2) is well-defined if the following constraints on the functions u and f hold:

$$\begin{aligned} &u \in H_0^1\big([0,1]\big), \quad u \in L^2\big([0,1]\big), \\ &f \in L^2\big([0,1]\big). \end{aligned}$$

As a matter of fact, we require even less restrictive constraints on u and f to guarantee that the Dirichlet energy $J(u)$ is well defined. Indeed, consider the following simple calculation:

Let $u \in H_0^1\big([0,1]\big)$. Hence $u(0) = 0$ and by the fundamental theorem of calculus it holds that

$$\begin{aligned} u(x) &= \int_0^x u'(s)ds, \qquad \forall\ x \in [0,1], \\ \implies |u(x)| &\leq \int_0^x |u'(s)|ds \\ &\leq \int_0^1 |u'(s)|ds, \quad \forall\ x \in [0,1], \\ \implies |u(x)| &\leq \int_0^1 1 \cdot |u'(s)|ds \\ &\leq \Big(\int_0^1 |1|^2 ds\Big)^{1/2} \cdot \Big(\int_0^1 |u'(s)|^2 ds\Big)^{1/2}, \end{aligned}$$

where the last inequality follows from the Cauchy-Schwarz inequality.

Therefore, for all $x \in [0,1]$, it holds that

$$|u(x)|^2 \leq \int_0^1 |u'(s)|^2 ds.$$

Integrating both sides of this inequality over the interval $[0,1]$, we obtain

$$\int_0^1 |u(x)|^2 dx \leq \int_0^1 |u'(x)|^2 dx, \tag{7.6}$$

and therefore it holds that

$$\|u\|_{L^2([0,1])} \leq \|u\|_{H_0^1([0,1])}. \tag{7.7}$$

Inequality (7.7) is an example of a *Poincaré inequality*. In particular, Inequality (7.7) implies that it is sufficient to impose the constraints $u \in H_0^1\big([0,1]\big)$ and $f \in L^2\big([0,1]\big)$ in order to ensure that the energy functional (7.2) is well-defined.

The precise statement of the variational principle is therefore:

Given $f \in L^2\big([0,1]\big)$, find $u \in H_0^1\big([0,1]\big)$, such that u minimises the energy functional $J(v)$ given by (7.2) for all $v \in H_0^1\big([0,1]\big)$, i.e.,

find $u \in H_0^1\big([0,1]\big)$ such that

$$J(u) = \min_{v \in H_0^1([0,1])} J(v). \tag{7.8}$$

7.2 A Variational Formulation

As is standard in the Calculus of Variations, we will compute the minimiser u of (7.8) using the Euler-Lagrange equations, i.e.,

Find $u \in H_0^1\big([0,1]\big)$ such that for all $v \in H_0^1\big([0,1]\big)$ it holds that

$$J'(u,v) = 0,$$

where

$$J'(u,v) = \lim_{\tau \to 0} \frac{J(u+\tau v) - J(u)}{\tau}.$$

We recall that the formal calculations in Section 6.1.1 imply that for all $v \in H_0^1\big([0,1]\big)$ the minimiser u must satisfy the equation

$$\int_0^1 u'(x)v'(x)dx = \int_0^1 v(x)f(x)dx. \tag{7.9}$$

Several remarks are in order here.

Remark 7.3 *Equation* (7.9) *is termed the* variational formulation *of the one-dimensional Poisson equation. It is also known as the principle of virtual work in structural mechanics.*

Remark 7.4 *We observe that* (7.9) *is well-defined. Indeed, let* $u, v \in H_0^1\big([0,1]\big)$*. Then, the Cauchy-Schwarz inequality implies that*

$$\left| \int_0^1 u'(x)v'(x)dx \right| \leq \|u\|_{H_0^1([0,1])} \cdot \|v\|_{H_0^1([0,1])} < \infty.$$

and similarly, $f \in L^2\big([0,1]\big)$ *implies that*

$$\begin{aligned} \left| \int_0^1 v(x)f(x)dx \right| &\leq \|v\|_{L^2([0,1])} \cdot \|f\|_{L^2([0,1])} \qquad (\textit{Cauchy-Schwarz}) \\ &\leq \|v\|_{H_0^1([0,1])} \cdot \|f\|_{L^2([0,1])} \qquad (\textit{Poincaré Inequality}) \\ &< \infty. \end{aligned}$$

Remark 7.5 *Since* $u \in H_0^1\big([0,1]\big)$*, we can set* $v = u$ *in Equation* (7.9)*. Then it holds that*

$$\begin{aligned} \|u\|^2_{H_0^1([0,1])} = \int_0^1 |u'(x)|^2 dx &= \int_0^1 u(x)f(x)dx \\ &\leq \|u\|_{L^2([0,1])} \cdot \|f\|_{L^2([0,1])} \qquad (\textit{Cauchy-Schwarz}) \\ &\leq \|u\|_{H_0^1([0,1])} \cdot \|f\|_{L^2([0,1])} \qquad (\textit{Poincaré Inequality}), \end{aligned}$$

and therefore, it holds that

$$\|u\|_{H_0^1([0,1])} \leq \|f\|_{L^2([0,1])} \tag{7.10}$$

Thus, (7.10) provides a stability estimate on the solution of the variational formulation of the one-dimensional Poisson equation in terms of the given data function f.

Remark 7.6 *Let u be the solution to Equation* (7.9) *and furthermore suppose that u'' exists and is bounded. Applying integration by parts in Equation* (7.9) *implies that for all $v \in H_0^1\big([0,1]\big)$ it holds that*

$$\begin{aligned} & -\int_0^1 u''(x)v(x)dx = \int_0^1 v(x)f(x)dx, \\ \Longrightarrow \quad & \int_0^1 \big(-u''(x) - f(x)\big)v(x)dx = 0, \\ \Longrightarrow \quad & -u''(x) = f(x). \end{aligned}$$

We have therefore recovered the pointwise form of the one-dimensional Poisson equation (7.1). *We remark that Equation* (7.1) *is also termed the* strong form *of the one-dimensional Poisson equation. However, the variational form* (7.9) *is more fundamental since it is derived directly from the variational principle* (7.2). *The Finite Element method (FEM) essentially amounts to a discretisation of this variational formulation.*

7.3 The Finite Element Formulation

Let $V = H_0^1\big([0,1]\big)$. Then the variational formulation of the one-dimensional Poisson equation is:

Find $u \in V$ such that for all $v \in V$ it holds that

$$\int_0^1 u'(x)v'(x)dx = \int_0^1 v(x)f(x)dx. \tag{7.11}$$

Using the notation

$$\big(g,h\big) = \int_0^1 g(x)h(x)dx,$$

we may write Equation (7.11) concisely as

$$\big(u',v'\big) = \big(f,v\big), \qquad \forall\ v \in V. \tag{7.12}$$

Observe that the function space V is infinite-dimensional. The Finite Element method replaces this infinite-dimensional space V with a suitable finite-dimensional subspace $V^h \subseteq V$ and attempts to find a function $u_h \in V^h$ such that (7.12) holds for all $v \in V^h \subseteq V$.

To be more concrete, we consider a discretisation of the domain $\Omega = [0,1]$. Let $h > 0$ and let $N = \frac{1}{h} - 1$. Then we discretise the domain $[0,1]$ into $N+2$ points by setting

$$x_0 = 0, \quad x_{N+1} = 1, \quad x_j = jh, \ j = 1, \dots, N.$$

Figure 7.1 displays an example of the discretised domain. We remark that this discretisation is exactly the same as the discretisation considered in the case of the finite difference method.

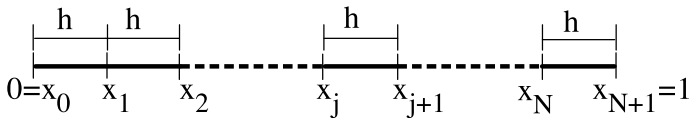


Figure 7.1: An example of a one-dimensional mesh for the domain $\Omega = (0,1)$.

In addition, let

$$V^h = \big\{ w \colon [0,1] \to \mathbb{R} \colon w \text{ is continuous}, w(0) = w(1) = 0, \\ \text{and } w|_{[(j-1)h, jh]} \text{ is linear } \ \forall\, j \in \{1, \dots, N+1\} \big\}.$$

In other words, V^h is the set of all continuous, piecewise linear functions on $[0,1]$ with respect to the partition

$$[0,1] = \bigcup_{j=1}^{N+1} [(j-1)h, jh].$$

A typical example of a function $w \in V^h$ is shown in Figure 7.2.

Exercise 7.7 *Check that for all $h > 0$, it holds that $V^h \subseteq V = H_0^1\big([0,1]\big)$.*

Remark 7.8 *Consider, for $j \in \{1, \dots, N\}$, the continuous and piecewise linear function $\phi_j(x)$ given by*

$$\phi_j(x_i) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

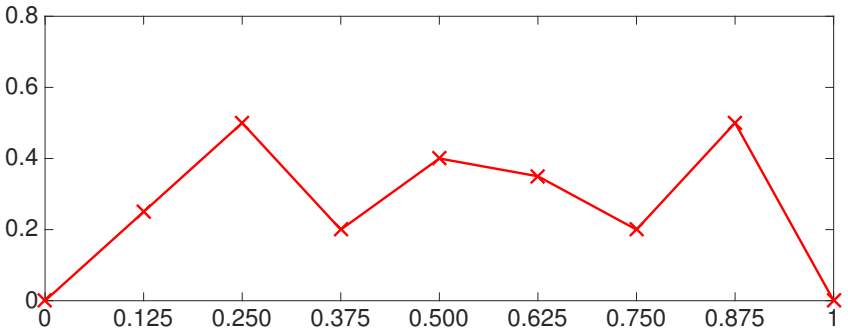


Figure 7.2: An example of a continuous, piecewise linear function in the space V^h for the mesh-width $h = \frac{1}{8}$.

More explicitly, we can calculate the functions to be

$$\phi_j(x) = \begin{cases} \frac{x - x_{j-1}}{h} & x \in [x_{j-1}, x_j), \\ \frac{x_{j+1} - x}{h} & x \in [x_j, x_{j+1}), \\ 0 & \textit{otherwise.} \end{cases}$$

Figure 7.3 displays an example of such a function. The functions ϕ_j, $j = 1, \ldots, N$ are termed ***hat*** *or* ***tent*** *functions.*

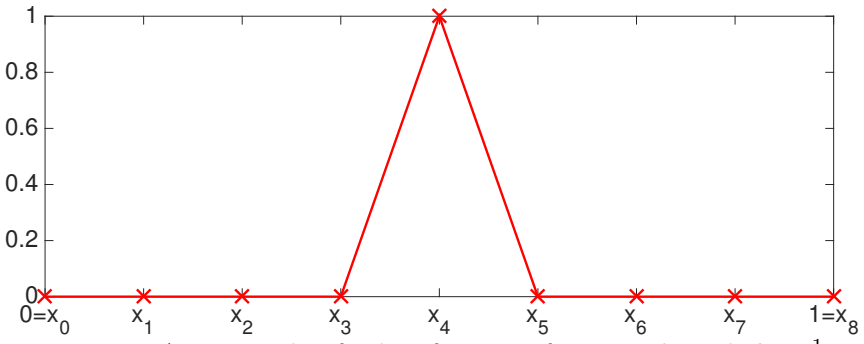


Figure 7.3: An example of a hat function for a mesh with $h = \frac{1}{8}$.

It can easily be checked that the set $\{\phi_j\}_{j=1}^N$ forms a basis of the space V^h. Indeed, given a function $w \in V^h$, it clearly holds that

$$w(x) = \sum_{j=1}^{N} w_j \phi_j(x), \tag{7.13}$$

where $w_j = w(x_j)$.

It therefore follows that the space V^h is finite-dimensional and in fact has dimension N. The FEM for the one-dimensional Poisson equation thus consists of the following variational formulation:

Find $u_h \in V^h$ such that for all $v \in V^h$ it holds that

$$\big(u_h', v'\big) = \big(f, v\big). \tag{7.14}$$

Thus the finite element approximation u_h is a continuous, piecewise linear function that approximates the solution of the Poisson equation. This should be contrasted to the finite difference method where point values of u are approximated.

7.3.1 Concrete Realisation of FEM

Given $v \in V^h$, it follows from (7.13) that v can be written as

$$v = \sum_{j=1}^{N} v_j \phi_j(x).$$

Therefore, (7.14) implies that

$$\begin{aligned} &\big(u_h', v'\big) = \big(f, v\big), \\ \iff &\left(u_h', \Big(\sum_{j=1}^{N} v_j \phi_j(x)\Big)'\right) = \left(f, \sum_{j=1}^{N} v_j \phi_j(x)\right). \end{aligned}$$

Next, using the linearity of the inner product $\big(\cdot, \cdot\big)$, it holds that

$$\sum_{j=1}^{N} v_j \big(u_h', \phi_j'\big) = \sum_{j=1}^{N} v_j \big(f, \phi_j\big)$$

for all $v_j \in \mathbb{R}$.

This final equation is satisfied if and only if for all $j = 1, \ldots, N$, it holds that

$$\big(u_h', \phi_j'\big) = \big(f, \phi_j\big). \tag{7.15}$$

Furthermore, since $u_h \in V^h$, it holds that

$$u_h = \sum_{i=1}^{N} u_i \phi_i(x).$$

Thus, (7.15) reduces to

$$\sum_{i=1}^{N} u_i\big(\phi_i', \phi_j'\big) = \big(f, \phi_j\big).$$

Next, we define the $N \times N$ matrix A as

$$A = \{A_{ij}\}_{i,j=1,\dots,N},$$

where

$$A_{ij} = \big(\phi_i', \phi_j'\big),$$

and the vectors $U = \{u_j\}_{j=1}^N$ and $F = \{F_j\}_{j=1}^N$, where

$$F_j = \big(f, \phi_j\big).$$

Hence, Equation (7.15) reduces to the matrix equation

$$AU = F. \tag{7.16}$$

The matrix A is termed the *stiffness matrix*, the Vector F is termed the *load vector* and the vector U is termed the *solution vector*. Thus, the Finite Element method also reduces to a matrix equation.

Of course, it remains to establish that the matrix equation (7.16) has a unique solution. To this end, observe that the matrix A is symmetric since

$$\begin{aligned} A_{ij} = \big(\phi_i', \phi_j'\big) &= \int_0^1 \phi_i'(x)\phi_j'(x)dx \\ &= \int_0^1 \phi_j'(x)\phi_i'(x)dx = \big(\phi_j', \phi_i'\big) = A_{ji}. \end{aligned}$$

Furthermore, denoting by $\langle \cdot, \cdot \rangle$ the usual inner product on $\mathbb{R}^N$, it holds for any $w \in \mathbb{R}^N$ that

$$\begin{aligned} \langle Aw, w\rangle = \sum_{i,j=1}^{N} w_i A_{ij} w_j &= \sum_{i,j=1}^{N} w_i\big(\phi_i', \phi_j'\big)w_j \\ &= \left(\sum_{i=1}^{N} w_i\phi_i', \sum_{j=1}^{N} w_j\phi_j'\right) \\ &= \big(\bar{w}', \bar{w}'\big) > 0, \quad \text{if } \bar{w} \neq 0, \end{aligned}$$

where $\bar{w} = \sum_{i=1}^N w_i\phi_i$. Thus, the matrix A is positive-definite.

Given that the matrix A is symmetric and positive definite, it follows that A is invertible, i.e. the linear system (7.16) is solvable and has a unique solution. Thus, the Finite Element method (7.14) is well-defined.

7.3.2 Computing the Stiffness Matrix and the Load Vector

We can compute the stiffness matrix explicitly in the following manner:

$$\phi_j(x)' = \begin{cases} \frac{1}{h}, & \text{if } x \in [(j-1)h, jh], \\ -\frac{1}{h}, & \text{if } x \in [jh, (j+1)h], \\ 0, & \text{otherwise.} \end{cases}$$

Hence, for all $i, j = 1, \ldots, N$ it holds that

$$\begin{aligned} A_{ij} &= 0, \qquad \text{if } |i-j| > 1, \\ A_{j-1,j} &= A_{j,j-1} = -\frac{1}{h}, \\ A_{j,j} &= \frac{1}{h^2} \int_{x_{j-1}}^{x_{j+1}} dx = \frac{2}{h}. \end{aligned}$$

Therefore, the matrix A is given by

$$A = \frac{1}{h} \begin{bmatrix} 2 & -1 & 0 & \ldots & 0 \\ -1 & 2 & -1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & -1 & 2 & -1 \\ 0 & \ldots & 0 & -1 & 2 \end{bmatrix}.$$

Thus, up to a scaling factor, the stiffness matrix A in FEM is identical to the matrix A that arises in a finite difference method.

Next, in order to evaluate the load vector F, observe that

$$F_j = \int_0^1 f(x)\phi_j(x)dx,$$

and this integral can be computed using a numerical quadrature rule. In particular, using the trapezoidal method on each element, we obtain

$$\begin{aligned} F_j &= \int_0^1 f(x)\phi_j(x)dx = \int_{x_{j-1}}^{x_j} f(x)\phi_j(x)dx + \int_{x_j}^{x_{j+1}} f(x)\phi_j(x)dx \\ &\approx \frac{h}{2}\Big(\underbrace{\phi_j(x_{j-1})}_{=0} f_{j-1} + 2\underbrace{\phi_j(x_j)}_{=1} f_j + \underbrace{\phi_j(x_{j+1})}_{=0} f_{j+1} \Big) = h f_j. \end{aligned}$$

7.4 Convergence Analysis

Let u_h be the FEM solution of Equation (7.14) and define

$$e_h = u - u_h.$$

Thus, $e_h \in V$ is the error function. Note that (7.12) implies that for all $v \in V^h \subseteq V$, it holds that

$$\big(u', v'\big) = \big(f, v\big),$$

and similarly (7.14) implies that for all $v \in V^h$, it holds that

$$\big(u_h', v'\big) = \big(f, v\big).$$

Therefore, the linearity of the inner product $\big(\cdot, \cdot\big)$ implies that for all $v \in V^h$ it holds that

$$\big((u - u_h)', v'\big) \equiv 0,$$

i.e.,

$$(e_h', v') \equiv 0. \tag{7.17}$$

Identity (7.17) is termed *Galerkin orthogonality* and it implies that the error e_h is orthogonal to the subspace V^h with respect to the $\big(\cdot, \cdot\big)$ inner product. Next, given any $v \in V^h$, we define $w = u_h - v \in V^h$. Then it holds that

$$\begin{aligned}
\|e_h\|^2_{H_0^1([0,1])} &= \int_0^1 |e_h'(x)|^2 dx \\
&= \big(e_h', e_h'\big) \\
&= \big(e_h', e_h'\big) + (e_h', w') \quad \text{(Galerkin Orthogonality (7.17))} \\
&= \big(e_h', (e_h + w)'\big) \qquad \text{(Linearity of } \big(\cdot, \cdot\big)\text{)} \\
&= \big(e_h', (u - v)'\big) \qquad \text{(Definition of } e_h\text{)} \\
&= \int_0^1 e_h'(x)\big(u'(x) - v'(x)\big) dx \\
&\leq \left(\int_0^1 |e_h'(x)|^2 dx \right)^{1/2} \cdot \left(\int_0^1 |u'(x) - v'(x)|^2 dx \right)^{1/2} \\
&= \|e_h\|_{H_0^1([0,1])} \cdot \|u - v\|_{H_0^1([0,1])},
\end{aligned}$$

where the last inequality follows from the Cauchy-Schwarz inequality.

Hence, for all $v \in V^h$ it holds that

$$\|e_h\|_{H_0^1([0,1])} \leq \|u - v\|_{H_0^1([0,1])}.$$

In other words, for all $v \in V^h$ it holds that

$$\|u - u_h\|_{H_0^1([0,1])} \leq \|u - v\|_{H_0^1([0,1])}. \tag{7.18}$$

The inequality (7.18) is known as Céa's Lemma. It is essentially an optimality result which states that u_h is the best approximation of the exact solution u in the space V^h with respect to the $\|\cdot\|_{H_0^1([0,1])}$ norm. Inequality (7.18) is therefore an extremely powerful tool for proving error estimates. Indeed, we can choose $v \in V^h$ to be the piecewise linear interpolant of u, i.e., given a grid, v is a continuous, piecewise linear function such that for all $j = 1, \ldots, N$ it holds that

$$\begin{aligned} v(x_j) &= u(x_j), \\ v(0) &= v(1) = 0. \end{aligned}$$

We denote such a function v by $I_h u$. Figure 7.4 displays an example of $I_h u$ for the function $u(x) = \sin(x)$.

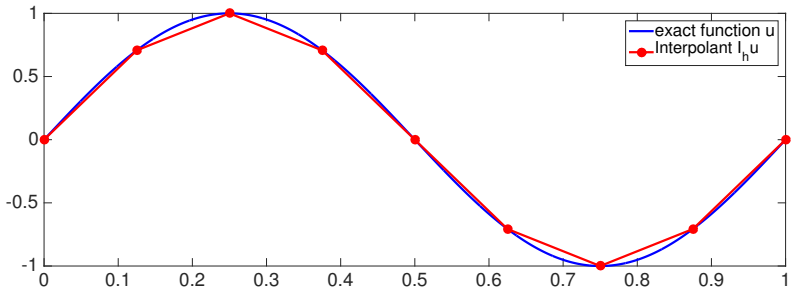


Figure 7.4: Plot of the function $u = \sin(x)$ and the interpolant $I_h u$ for a mesh with $h = \frac{1}{8}$.

This interpolation allows us to calculate that for all $x \in [0, 1]$ it holds that

$$|u(x) - I_h u(x)| \leq \frac{h^2}{8} \max_{0 \leq y \leq 1} |u''(y)|,$$

and

$$|u'(x) - I_h u'(x)| \leq \tilde{C} h \max_{0 \leq y \leq 1} |u''(y)|,$$

where $\tilde{C} \in \mathbb{R}$ is some constant.

Next, squaring and integrating the above inequalities results in

$$\|u - I_h u\|_{H_0^1([0,1])} \leq Ch. \tag{7.19}$$

where C is a constant that depends on $\max_{0 \leq y \leq 1} |u''(y)|$.

Therefore, (7.18) and (7.19) together imply that

$$\|u - u_h\|_{H_0^1([0,1])} \leq Ch. \tag{7.20}$$

Thus, the FEM converges to the exact solution as $h \to 0$ and the rate of convergence in the $H_0^1\big([0,1]\big)$ norm is at least 1. In order to obtain a corresponding error estimate for the $L^2\big([0,1]\big)$ norm, we can use the Poincaré inequality (7.7) to obtain

$$\|u - u_h\|_{L^2([0,1])} \leq Ch.$$

Therefore, the rate of convergence in the $L^2\big([0,1]\big)$ norm is also at least 1. However, this last error estimate is extremely crude and in practice we often observe an EOC ≈ 2 in the $L^2\big([0,1]\big)$ norm.

7.5 Numerical Experiments

As a practical example, let us consider the following two-point boundary value problem:

$$\begin{aligned} -u''(x) &= \sin(x), \quad -\pi < x < \pi, \\ u(-\pi) &= u(\pi) = 0. \end{aligned} \tag{7.21}$$

It can be shown that exact solution of (7.21) is given by $u(x) = \sin(x)$. We use the Finite Element method to approximate the solution to this problem using $N = 100$ mesh points.

Figure 7.5 displays our results and indicates that the Finite Element method can approximate solutions to the BVP (7.21) very well. Indeed the associated errors given in Table 7.1 indicate that the approximate solutions seem to converge to the exact solution as the number of mesh points is increased. Furthermore, our results agree with the error estimate (7.20) and we observe linear convergence in the H_0^1 norm. On the other hand, we obtain EOC ≈ 2 in the L^2 norm.

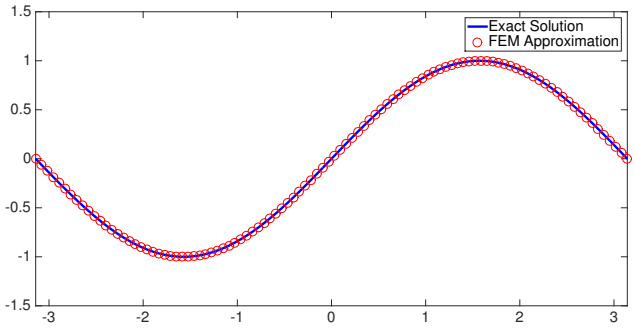


Figure 7.5: Exact solution and approximate solution plot for the Finite Element method using $N = 100$ mesh points.

N	Error in L^2 norm	EOC	Error in H_0^1-norm	EOC
25	8.610×10^{-3}	1.998	0.1235	0.999
50	2.241×10^{-3}	2.000	0.0630	1.000
100	5.716×10^{-4}	2.000	0.0318	1.000
200	1.443×10^{-4}		0.0160	

Table 7.1: Error Table of the Finite Element method for the BVP (7.21).

As a second example, we consider the following two-point boundary value problem:

$$\begin{aligned} -u''(x) &= \cos(x), \quad -\pi < x < \pi, \\ u(-\pi) &= -1, \quad u(\pi) = \frac{1}{2}. \end{aligned} \tag{7.22}$$

Note that this requires changes in the FEM methodology we have introduced since the boundary conditions are no longer zero.

It can be shown that the exact solution to the BVP (7.22) is given by $u(x) = \cos(x) + \frac{3x}{4\pi} + \frac{3}{4}$. Once again, we use FEM to approximate the solution to this problem using $N = 100$ mesh points.

Figure 7.6 displays our results and indicates that FEM can also approximate solutions to the BVP (7.22) very well. Indeed the associated errors given in Table 7.2 once again indicate that the approximate solutions seem to converge to the exact solution as the number of mesh points is increased. Once again, we observe EOC ≈ 1 in the H_0^1 norm and EOC ≈ 2 in the L^2 norm.

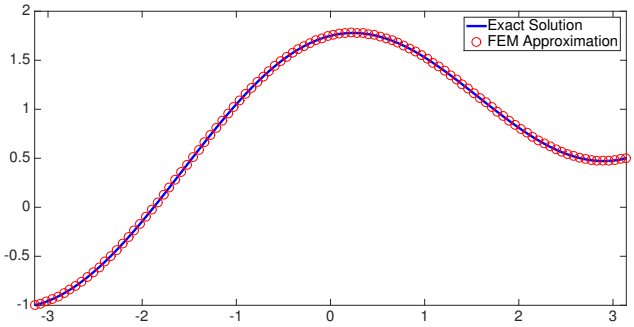


Figure 7.6: Exact solution and approximate solution plot for the Finite Element method using $N = 100$ mesh points.

N	Error in L^2 norm	EOC	Error in H_0^1-norm	EOC
25	8.610×10^{-3}	1.998	0.1235	0.999
50	2.241×10^{-3}	2.000	0.0630	1.000
100	5.716×10^{-4}	2.000	0.0318	1.000
200	1.443×10^{-4}		0.0160	

Table 7.2: Error Table of the Finite Element method for the BVP (7.22).

8 Finite Element Methods for the 2-D Poisson Equation

In this Chapter, we extend the Finite Element method for solving the Poisson equation to two dimensions. The formulation of FEM in two dimensions is analogous to the one-dimensional case. We therefore begin by stating some definitions.

Let $\Omega \subset \mathbb{R}^2$ (or $\mathbb{R}^n$, for any $n \in \mathbb{N}$) be a bounded set. Then we define the function space

$$L^2(\Omega) = \Big\{ v \colon \Omega \to \mathbb{R} \colon \int_\Omega |v|^2 dx < \infty \Big\}.$$

In addition, we define a norm on the space $L^2(\Omega)$ by setting for all $v \in L^2(\Omega)$

$$\|v\|_{L^2(\Omega)} = \left(\int_\Omega |v|^2 dx \right)^{1/2},$$

and we define an inner product by setting for all $u, v \in L^2(\Omega)$

$$\big(u, v\big)_{L^2(\Omega)} = \int_\Omega u(x) v(x) dx.$$

Similarly, we define the function space

$$H_0^1(\Omega) = \Big\{ v \colon \Omega \to \mathbb{R} \colon \int_\Omega |\nabla v|^2 dx < \infty \text{ and } v = 0 \text{ on } \partial\Omega \Big\}.$$

We again define a norm on this function space by setting for all $v \in H_0^1(\Omega)$

$$\|v\|_{H_0^1(\Omega)} = \left(\int_\Omega |\nabla v|^2 dx \right)^{1/2},$$

and we define an inner product by setting for all $u, v \in H_0^1(\Omega)$

$$\big(u, v\big)_{H_0^1(\Omega)} = \int_\Omega \langle \nabla u, \nabla v \rangle dx.$$

Finally, we recall that the norms we have introduced are related through the Poincaré inequality: for all $v \in H_0^1(\Omega)$, there exists some constant $C \in \mathbb{R}$ such that

$$\int_\Omega |v|^2 dx \leq C \int_\Omega |\nabla v|^2 dx. \tag{8.1}$$

8.1 The two-dimensional Poisson Equation

Let $\Omega \subseteq \mathbb{R}^2$ be an open set and let $f\colon \Omega \to \mathbb{R}$ be some known function. Then the strong form of the two-dimensional Poisson equation is given by

$$\begin{aligned} -\Delta u &= f, \qquad \text{on } \Omega, \\ u &= 0, \qquad \text{on } \partial\Omega. \end{aligned} \tag{8.2}$$

Similar to the one-dimensional case however, the variational formulation of the two-dimensional Poisson equation is more fundamental than the strong form.

8.1.1 Variational Formulation

Let $\Omega \subseteq \mathbb{R}^2$ be an open set, let $f\colon \Omega \to \mathbb{R}$ be some known function and let $V = H_0^1(\Omega)$. Then the weak form of the two-dimensional Poisson equation (8.2) is as follows:

Find $u \in V$ such that for all $v \in V$ it holds that

$$\big(u, v\big)_{H_0^1(\Omega)} = \big(f, v\big)_{L^2(\Omega)}, \tag{8.3}$$

or equivalently

$$\int_\Omega \langle \nabla u, \nabla v \rangle dx = \int_\Omega f(x) v(x) dx.$$

A few remarks are now in order.

Remark 8.1 *If the function $u \in H_0^1(\Omega)$ is sufficiently regular, then the weak form* (8.3) *is equivalent to the strong form* (8.2)*. This assertion can be checked by repeating the calculations carried out in Remark 7.6 in two dimensions, i.e. by applying integration by parts in two dimensions.*

Remark 8.2 *Let the energy functional J be defined as*

$$J(w) = \frac{1}{2}\int_\Omega |\nabla w|^2 dx - \int_\Omega w(x) f(x) dx,$$

and let $u \in H_0^1(\Omega)$ be the function with the property that for all $w \in H_0^1(\Omega)$ it holds that

$$J(u) \leq J(w).$$

Then, it can be shown that u satisfies the variational formulation (8.3).

8.2 The Finite Element Formulation

Similar to the one-dimensional case, the Finite Element method in two dimensions consists of solving the variational formulation on a finite-dimensional subspace $V^h \subseteq V = H_0^1(\Omega)$.

8.2.1 Triangulations

In order to define the finite-dimensional subspace V^h, we consider a polygonal domain Ω. Thus, $\partial\Omega$ may consist of a large number of sides. An example of such a domain is displayed in Figure 8.1.

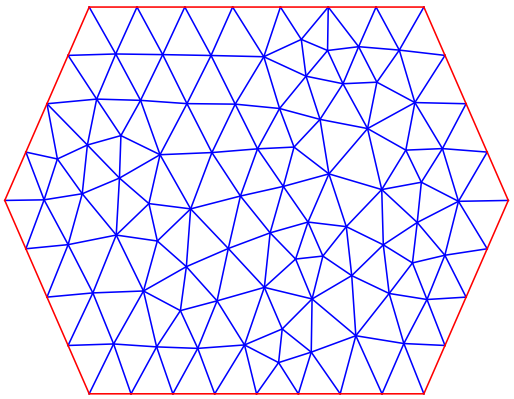


Figure 8.1: Two-dimensional triangular mesh for a hexagonal domain Ω.

We consider a triangulation T_h of the domain Ω of the form

$$\bar{\Omega} = \bigcup_{K \in T_h} \bar{K} = \bar{K}_1 \cup \bar{K}_2 \cup \ldots \cup \bar{K}_M,$$

where $K_i,\ i = 1, \ldots, M$ are non-overlapping triangles such that no vertex of a triangle lies on the interior of an edge of another triangle. An example of such a triangulation is displayed in Figure 8.1.

Next, let h be the mesh width, i.e.

$$h = \max_{K \in T_h} \operatorname{diam}(K),$$

where $\operatorname{diam}(K)$ is the length of the longest edge of the triangle K.

Then, we define the finite-dimensional subspace V^h as

$$V^h = \Big\{ v \colon \Omega \to \mathbb{R} \colon v \text{ continuous}, v|_K \text{ linear for each } K \in T_h,\ v = 0 \text{ on } \partial\Omega \Big\}.$$

In other words, V^h is the space of continuous, piecewise linear functions that vanish on the boundary $\partial\Omega$. It is easy to check that $V^h \subseteq H_0^1(\Omega)$.

The resulting Finite Element formulation is therefore:

Find $u_h \in V^h$ such that for all $v \in V^h$ it holds that

$$\big(u_h, v\big)_{H_0^1(\Omega)} = \big(f, v\big)_{L^2(\Omega)}, \tag{8.4}$$

or equivalently

$$\int_\Omega \langle \nabla u_h, \nabla v \rangle dx = \int_\Omega f(x) v(x) dx.$$

8.2.2 Concrete Realisation of FEM

As in the one-dimensional case, we can define a basis for the finite-dimensional subspace V^h. Indeed, let $\{\mathcal{N}_i\}_{i=1}^N$ denote the set of interior nodes of the triangles in T_h. Then we define the so-called hat functions

$$\phi_j(x) \in V^h, \quad j = 1, \ldots, N,$$

where

$$\phi_j(\mathcal{N}_i) = \begin{cases} 1 & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases}$$

An example of such a hat function is displayed in Figure 8.2. Note that the support of each hat function ϕ_j is restricted to triangles $K \in T_h$ that have the node j as a vertex.

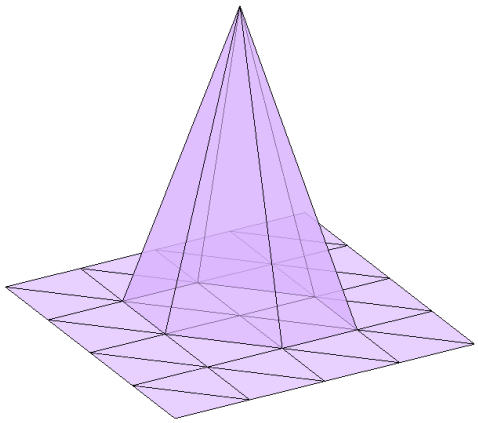


Figure 8.2: An example of a two-dimensional hat function.

Next, using the fact that the hat functions $\{\phi_j\}_{j=1}^N$ define a basis for the space V^h, we may write each function $v \in V^h$ as

$$v(x) = \sum_{j=1}^{N} v_j \phi_j(x),$$

where $v_j = v(\mathcal{N}_j)$ for all $j = 1, \ldots, N$.

The two-dimensional Finite Element method can therefore be formulated as

$$\big(u_h, \phi_j\big)_{H_0^1(\Omega)} = \big(f, \phi_j\big)_{L^2(\Omega)}, \quad \forall\ 1 \leq j \leq N,$$

or equivalently,

$$\int_\Omega \langle \nabla u_h, \nabla \phi_j \rangle dx = \int_\Omega f(x)\phi_j(x)dx, \quad \forall\ 1 \leq j \leq N. \tag{8.5}$$

Since $u_h \in V^h$, it follows that there exist coefficients $\{u_i\}_{i=1}^N$ such that

$$u_h = \sum_{i=1}^{N} u_i \phi_i.$$

Therefore, Equation (8.5) implies that for all $j = 1, \ldots, N$ it holds that

$$\sum_{i=1}^{N} u_i \int_\Omega \langle \nabla \phi_i, \nabla \phi_j \rangle dx = \int_\Omega f(x)\phi_j(x)dx. \tag{8.6}$$

Next, we define the vectors

$$U = \{u_i\}_{i=1}^N, \qquad F = \{F_j\}_{j=1}^N,$$

and the $N \times N$ matrix

$$A = \{A_{ij}\}_{i,j=1}^N,$$

where

$$F_j = \int_\Omega f(x)\phi_j(x)dx,$$
$$A_{ij} = \int_\Omega \langle \nabla\phi_i, \nabla\phi_j \rangle dx.$$

Equation (8.6) can then be written as the matrix equation

$$AU = F. \tag{8.7}$$

Thus, the two-dimensional Finite Element method also reduces to a matrix equation for the vector of unknowns U. This vector U can now be obtained by inverting the stiffness matrix A and multiplying it with the load vector F. We remark that similar to the one-dimensional case, the stiffness matrix A is symmetric and positive-definite and therefore invertible. Hence, the FEM equation (8.7) has a unique solution. Furthermore, since each basis function ϕ_j is supported on few triangles, the matrix A has a sparse structure.

We conclude this section by considering a concrete example. We consider the simple case of a domain consisting of the unit square in 2-D. Therefore, let $\Omega = (0,1)^2$ and consider the *uniform* triangulation of this domain as displayed in Figure 8.3.

It can then be shown that the stiffness matrix A is the $n^2 \times n^2$ block matrix given by

$$A = \begin{bmatrix} B & -I & 0 & \dots & 0 \\ -I & B & -I & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -I & B & -I \\ 0 & \dots & 0 & -I & B \end{bmatrix}.$$

Here, I is the $n \times n$ identity matrix and B is the $n \times n$ matrix given by

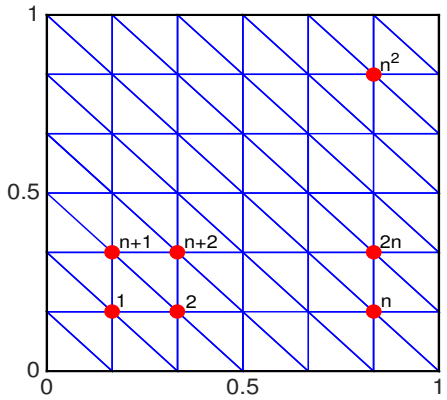


Figure 8.3: Uniform triangular mesh on the domain $\Omega = (0,1)^2$ consisting of $N = n^2$ interior nodes.

$$B = \begin{bmatrix} 4 & -1 & 0 & \dots & 0 \\ -1 & 4 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 4 & -1 \\ 0 & \dots & 0 & -1 & 4 \end{bmatrix}.$$

Therefore, the stiffness matrix A is identical to the matrix obtained in the two-dimensional finite difference method.

In general, the stiffness matrix A can be computed in a straightforward manner. We first observe that

$$A_{ij} = \int_\Omega \langle \nabla \phi_i, \nabla \phi_j \rangle dx = \sum_{K \in T_h} \int_K \langle \nabla \phi_i, \nabla \phi_j \rangle dx.$$

The above equation suggests that we can compute a local integral for each triangle $K \in T_h$ and then sum over all triangles to obtain each entry of the stiffness matrix A.

Therefore, let $\mathcal{N}_a, \mathcal{N}_b, \mathcal{N}_c$ be the three nodes of some triangle $K \in T_h$. It then follows from the definition of the basis functions that

$$\int_K \langle \nabla \phi_i, \nabla \phi_j \rangle dx = 0.$$

unless both i and j are a vertex of K, i.e. one gets non-zero contributions only if both i and j are in $\{a, b, c\}$.

Thus, it is sufficient to consider for each triangle $K \in T_h$, the symmetric, 3×3 *element stiffness matrix* A^K given by

$$A^K = \begin{bmatrix} \int_K \langle \nabla\phi_a, \nabla\phi_a \rangle dx & \int_K \langle \nabla\phi_a, \nabla\phi_b \rangle dx & \int_K \langle \nabla\phi_a, \nabla\phi_c \rangle dx \\ \int_K \langle \nabla\phi_b, \nabla\phi_a \rangle dx & \int_K \langle \nabla\phi_b, \nabla\phi_b \rangle dx & \int_K \langle \nabla\phi_b, \nabla\phi_c \rangle dx \\ \int_K \langle \nabla\phi_c, \nabla\phi_a \rangle dx & \int_K \langle \nabla\phi_c, \nabla\phi_b \rangle dx & \int_K \langle \nabla\phi_c, \nabla\phi_c \rangle dx \end{bmatrix}.$$

The global stiffness matrix A can then be computed by combining all the element stiffness matrices in the correct manner. This process is termed *assembly*. A detailed explanation of the assembly process can be found in Chapter 9. The load vector F can also be computed in a similar manner.

Remark 8.3 *Similar to the one-dimensional case, it can be shown that the two-dimensional Finite Element method satisfies the error estimate*

$$\|u - u_h\|_{H_0^1(\Omega)} \leq Ch\|\mathrm{D}^2 u\|_{L^2(\Omega)}.$$

8.2.3 Numerical Experiments in 2-D

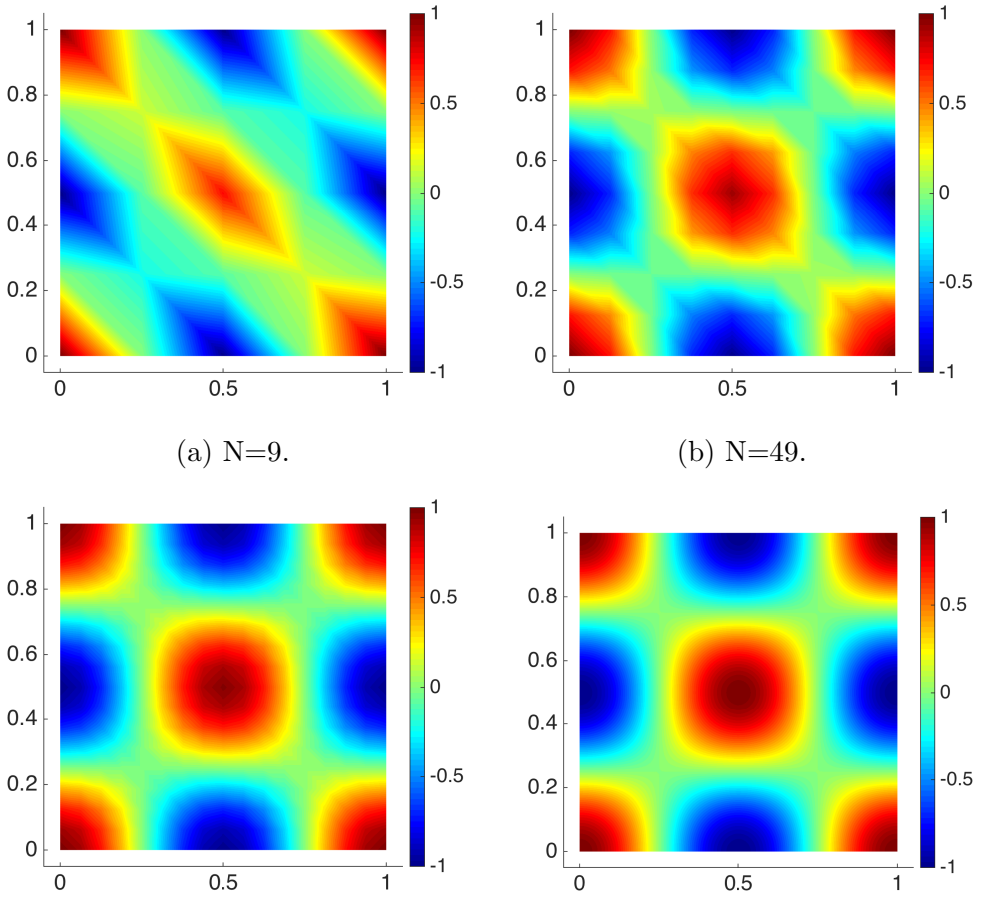
Example 1

As a first numerical example, we consider the following boundary value problem involving the two-dimensional Poisson equation:

$$\begin{aligned} -\big(u_{xx} + u_{yy}\big) &= 8\pi^2 \cos(2\pi x)\cos(2\pi y), && \forall\ x, y \in (0,1), \\ u(0,y) = u(1,y) &= \cos(2\pi y), && \forall\ y \in [0,1], \\ u(x,0) = u(x,1) &= \cos(2\pi x), && \forall\ x \in [0,1]. \end{aligned} \tag{8.8}$$

It can be shown that the exact solution to the BVP (8.8) is given by $u(x,y) = \cos(2\pi x)\cos(2\pi y)$. We use the two-dimensional Finite Element method to approximate solutions to this problem for different values of the number of interior nodes N. Note that the boundary conditions are no longer zero so we require some changes in the FEM methodology in order to account for these inhomogeneous boundary conditions. A comprehensive discussion on the treatment of inhomogeneous boundary conditions can be found in Section 9.1.

Figure 8.4 displays our results and indicates that the two-dimensional Finite Element method can approximate solutions to the BVP (8.8) very well. Indeed, the associated errors given in Table 8.1 indicate that the approximate solutions seem to converge to the exact solution as the number of interior



(a) N=9. (b) N=49.

(c) N=225. (d) Exact Solution.

Figure 8.4: Exact solution and approximate solution plots for the two-dimensional Finite Element method using different values of the number of interior nodes N.

nodes is increased. This conclusion is supported by Figure 8.5 and Figure 8.6, which display a logarithmic plot of the errors as a function of the number of interior nodes N and the mesh width h respectively. Furthermore, we observe once again that the experimental order of convergence (EOC) in the $H_0^1\big([0,1]^2\big)$ norm approaches 1 as the number of interior nodes is increased. On the other hand, the EOC in the $L^2\big([0,1]^2\big)$ norm approaches 2 as the number of interior nodes is increased.

Remark 8.4 *The EOCs depicted in Table 8.1 are with respect to the mesh width h. One could also compute the EOCs with respect to the number of degrees of freedom N, and these approach $\frac{1}{2}$ in the $H_0^1\big([0,1]^2\big)$ norm and*

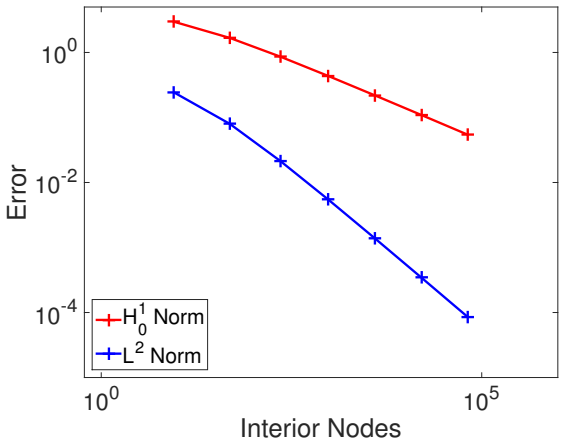


Figure 8.5: Logarithmic plot of the error in the H_0^1 norm and L^2 norm for the approximate solutions vs. the number of interior nodes N.

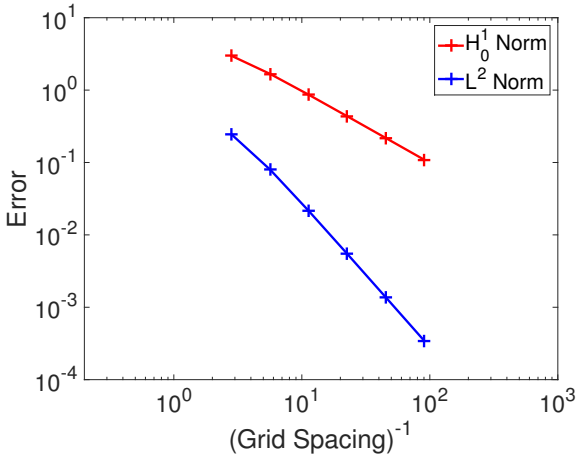


Figure 8.6: Logarithmic plot of the error in the H_0^1 norm and L^2 norm for the approximate solutions vs. the mesh-width h.

1 *in the* $L^2\big([0,1]^2\big)$ *norm. The reason is the following: in meshes without triangles with very acute angles and where all the triangles in the mesh have roughly the same size, it holds that*

$$h = \mathcal{O}\left(N^{-1/2}\right). \tag{8.9}$$

You can check that this holds for the uniform mesh depicted in Figure 8.3. This implies that the order of convergence with respect to N *is only half as big as the convergence with respect to* h*. Indeed, this relationship can be observed by examining Figure 8.5 and Figure 8.6. Moreover, this is*

one of the reasons why computations in two dimensions are much more computationally intensive than in one dimension, where $h = \mathcal{O}\left(N^{-1}\right)$.

N	h	Error in L^2 norm	EOC	Error in H_0^1 norm	EOC
9	0.354	0.2435	1.61	2.971	0.83
49	0.177	0.0796	1.89	1.672	0.95
225	0.088	0.0215	1.97	0.8629	0.99
961	0.044	5.469×10^{-3}	1.99	0.4350	1.00
3969	0.022	1.374×10^{-3}	2.00	0.2719	1.00
16129	0.011	3.439×10^{-4}		0.1090	

Table 8.1: Error Table of the two-dimensional Finite Element method for the BVP (8.8).

Example 2

A significant advantage of the Finite Element method is its ability to approximate solutions to PDEs on complicated domains. In order to illustrate this aspect of FEM, we next consider a boundary value problem involving the two-dimensional Poisson equation on a non-square domain. Thus, let the domain $\Omega = (-1,1)^2 \setminus \big([0,1] \times [-1,0]\big)$ and consider the BVP given by

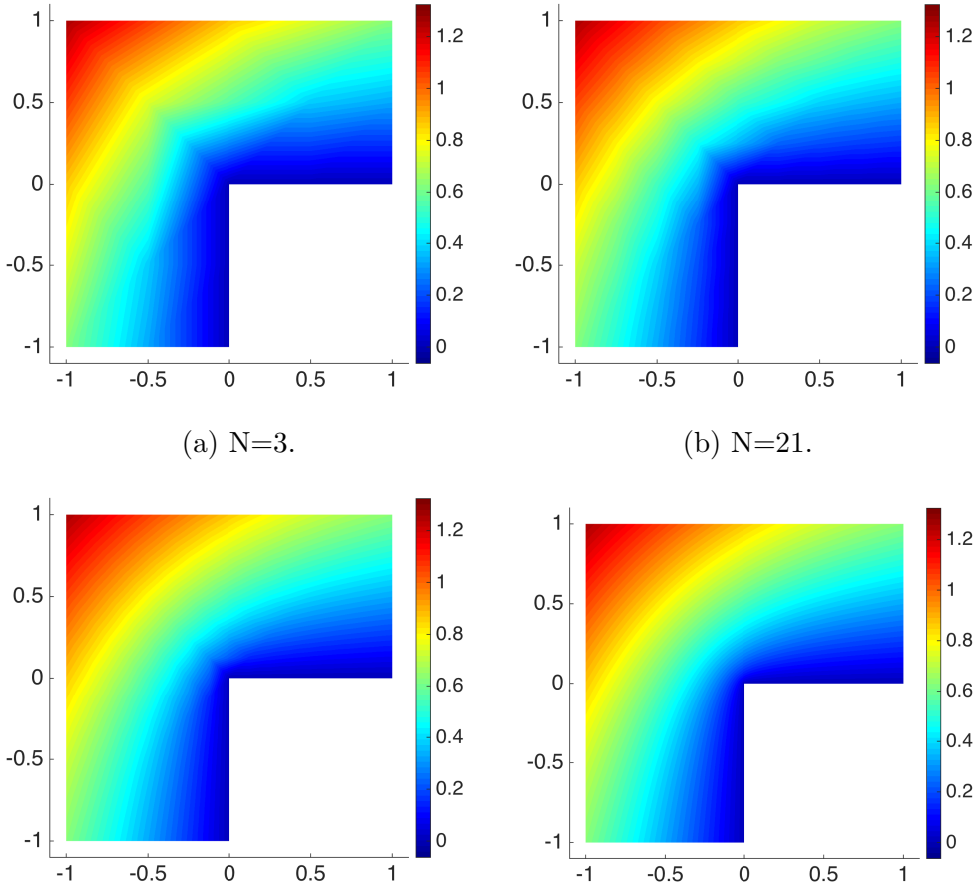
$$\begin{aligned} -\big(u_{xx} + u_{yy}\big) &= 1, && \forall\ (x,y) \in \Omega, \\ u(r,\theta) &= r^{\frac{2}{3}} \sin\big(\tfrac{2\theta}{3}\big), && \forall\ (r,\theta) \in \partial\Omega, \end{aligned} \tag{8.10}$$

where we have used polar coordinates (r,θ) to specify the boundary conditions.

N	h	Error in L^2 norm	EOC	Error in H_0^1 norm	EOC
3	1.000	0.0473	1.22	0.3554	0.62
21	0.500	0.0203	1.20	0.2320	0.63
105	0.250	8.818×10^{-3}	1.22	0.1498	0.64
1953	0.125	3.774×10^{-3}	1.25	0.0611	0.65
8001	0.063	1.585×10^{-3}	1.27	0.0387	0.66
32385	0.031	6.556×10^{-4}		0.0245	

Table 8.2: Error Table of the two-dimensional Finite Element method for the BVP (8.10).

Note that the domain in this case is now *L-shaped*. The exact solution to the BVP (8.10) is given in polar coordinates as $u(r,\theta) = r^{\frac{2}{3}} \sin(\frac{2\theta}{3})$. We



(a) N=3.

(b) N=21.

(c) N=105.

(d) Exact Solution.

Figure 8.7: Exact solution and approximate solution plots for the two-dimensional Finite Element method using different values of the number of interior nodes N.

once again use the two-dimensional Finite Element method to approximate solutions to this BVP for different values of the number of interior nodes N. We remark that similar to the previous example, the boundary conditions are not zero (except near the reentrant corner) so we must take into account the inhomogeneous nature of the boundary conditions (see Section 9.1).

Figure 8.7 displays our results and indicates once again that the two-dimensional Finite Element method can approximate solutions to the BVP (8.10) very well. Indeed, the associated errors given in Table 8.2 indicate that the approximate solutions seem to converge to the exact solution as the number of interior nodes is increased. This conclusion is supported by

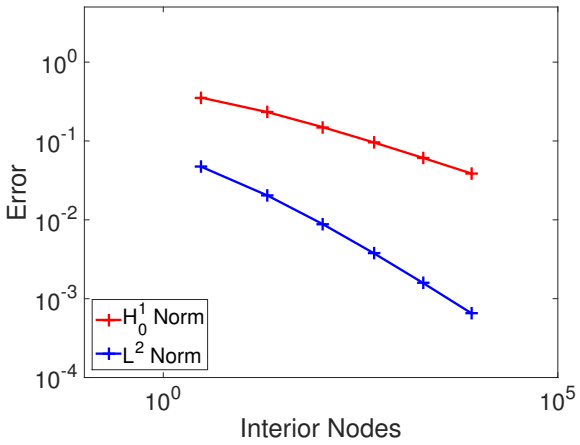


Figure 8.8: Logarithmic plot of the error in the H_0^1 norm and L^2 norm for the approximate solutions vs. the number of interior nodes N.

Figure 8.8, which displays a logarithmic plot of the errors as a function of the number of interior nodes N.

Based on the previous numerical examples in both 1-D and 2-D, we would expect to observe an experimental order of convergence (EOC) with respect to the mesh width h in the $H_0^1(\Omega)$ norm and the $L^2(\Omega)$ norm of approximately 1 and 2, respectively. However, in a significant departure from the previous numerical examples, we observe (see Table 8.2) that the EOC in both cases is much less. Further experiments reveal that the EOC in the $H_0^1(\Omega)$ norm and the $L^2(\Omega)$ norm approaches $\frac{2}{3}$ and $\frac{4}{3}$ respectively as the number of interior nodes is increased.

This reduced rate of convergence is essentially due to a so-called *corner singularity* that arises when considering a non-convex domain, and which leads to a solution with reduced regularity. More information on corner singularities can be found in any standard textbook on the Finite Element Method.

9 Implementation of the Finite Element Method

In this chapter, we describe a simple implementation of FEM for the two-dimensional Poisson equation with Dirichlet boundary conditions on a polygonal domain Ω with boundary Γ.

We recall that the variational formulation of the Finite Element method is of the form:

Find $u_h \in V^h$ such that for all $v \in V^h$ it holds that

$$\big(u_h, v\big)_{H_0^1(\Omega)} = \big(f, v\big)_{L^2(\Omega)}, \tag{9.1}$$

where $\big(\cdot,\cdot\big)_{H_0^1(\Omega)}$ is the H_0^1 inner product and $\big(\cdot,\cdot\big)_{L^2(\Omega)}$ is the L^2 inner product.

Thus, choosing the function space V^h to be the span of the hat functions, it can be shown that the discrete formulation (9.1) reduces to the matrix equation

$$AU = F, \tag{9.2}$$

where U is the vector of unknowns:

$$U = \{u_j\}_{j=1}^N,$$

$A = \{A_{ij}\}_{i,j=1}^N$ is the global stiffness matrix given by

$$A_{ij} = \int_\Omega \langle \nabla\phi_i, \nabla\phi_j \rangle dx,$$

and $F = \{F_j\}_{j=1}^N$ is the global load vector given by

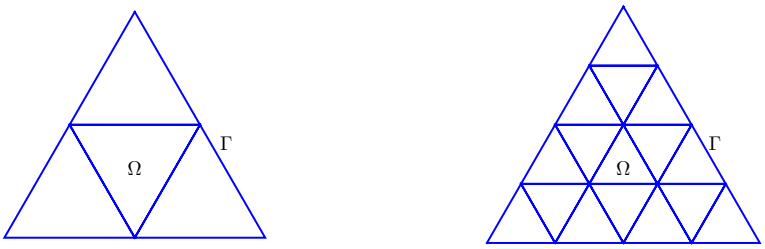
$$F_j = \big(f, \phi_j\big) = \int_\Omega f(x)\phi_j(x)dx.$$

The Finite Element method can then be implemented using the following steps:

Step 1 Triangulation (Mesh Generation)

Given a polygonal domain Ω, the first step is to divide this domain into a set of non-overlapping triangles. In general, this is a non-trivial, sophisticated process, but various efficient algorithms are available for this purpose.

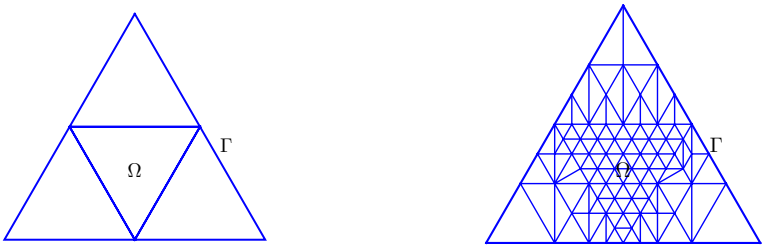
A popular algorithm involves starting with a coarse triangulation and then refining the triangles successively. An example of such a process is displayed in Figure 9.1. This process results in a so-called quasi-uniform mesh, i.e., the resulting triangles are of approximately the same size.



(a) Coarse triangulation. (b) Fine triangulation.

Figure 9.1: An example of a coarse and a fine quasi-uniform mesh.

Depending on the nature of our problem, it may be necessary to refine the mesh in some regions of the domain and allow it to remain coarse in other regions. Such meshes are typically referred to as adapted meshes. An example of adaptive mesh refinement is displayed in Figure 9.2.



(a) Coarse triangulation. (b) Adaptive triangulation.

Figure 9.2: An example of adaptive mesh refinment.

In either case, one relies on free or commercial mesh generators such as, e.g., *DistMesh (MATLAB), NETGEN, GID, DELAUNDO*, etc.

Let us now assume that a triangulation T_h of the domain Ω is available. The mesh information is then stored in the following manner:

Let $\{\mathcal{N}_i\}_{i=1}^{N}$ denote the nodes of the triangulation T_h and let $\{K_j\}_{j=1}^{M}$ denote the triangles in T_h. Then the mesh generator provides a numbering of the nodes and the triangles as shown in Figure 9.3.

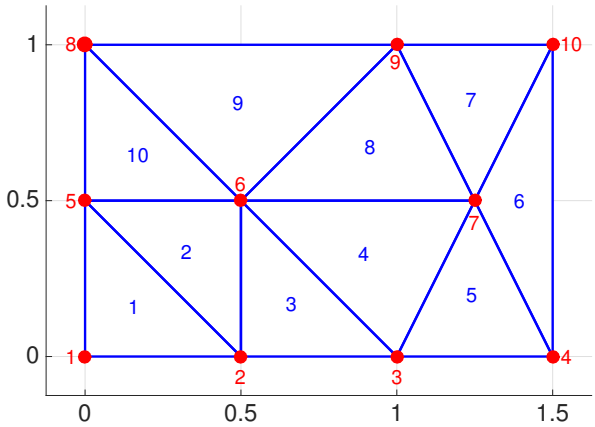


Figure 9.3: Labelling of the nodes (red) and triangles (blue) in a triangulation T_h.

In particular, the following two arrays are stored by the mesh generator:

1. Z is the $2 \times N$ array of node values such that
   - $\mathrm{Z}(\cdot, j)$ refers to the node $\mathcal{N}_j$,
   - $\mathrm{Z}(1, j)$, $\mathrm{Z}(2, j)$ represent the x- and y- coordinates respectively of the node $\mathcal{N}_j$.

2. T is the $3 \times M$ array of triangles such that
   - $\mathrm{T}(\cdot, j)$ refers to the j^{th} triangle K_j,
   - $\mathrm{T}(i, j)$, $(i = 1, 2, 3)$ represent the three node vertices corresponding to the triangle K_j.

Note that we have so far not distinguished between interior and boundary nodes, and indeed the array Z contains boundary nodes as well. Often, an

additional array containing flags that correspond to boundary nodes is also generated. This is important because it allows us to implement boundary conditions.

As a concrete example, the node array Z and the triangle array T corresponding to the mesh displayed in Figure 9.3 are given by

$$\mathrm{Z} = \begin{bmatrix} 0 & \frac{1}{2} & 1 & \frac{3}{2} & 0 & \frac{1}{2} & \frac{5}{4} & 0 & 1 & \frac{3}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 1 & 1 & 1 \end{bmatrix},$$

$$\mathrm{T} = \begin{bmatrix} 1 & 2 & 2 & 3 & 3 & 4 & 7 & 6 & 6 & 5 \\ 2 & 5 & 3 & 6 & 4 & 7 & 9 & 7 & 8 & 6 \\ 5 & 6 & 6 & 7 & 7 & 10 & 10 & 9 & 9 & 8 \end{bmatrix}.$$

Note that once the array value $\mathrm{T}(i,j)$ is obtained, the coordinates of the vertex i of the triangle K_j can be obtained by calling the array values $\mathrm{Z}\big(\cdot, \mathrm{T}(i,j)\big)$.

Most mesh generators provide the arrays Z and T and some additional information about the mesh. Mesh generators also aim to number the nodes and triangles as efficiently as possible in order to minimise the band width of the resulting matrices.

Step 2 Building Element Stiffness Matrices and Element Load Vectors

The structure of the global stiffness matrix A implies that each element of the matrix is given by

$$A_{ij} = \int_\Omega \langle \nabla\phi_i, \nabla\phi_j \rangle dx = \sum_{m=1}^{M} \int_{K_m} \langle \nabla\phi_i, \nabla\phi_j \rangle dx.$$

where K_m is the m^{th} triangle in the triangulation T_h.

Note that the definition of the basis functions $\{\phi_i\}_{i=1}^N$ then implies that $\int_{K_m} \langle \nabla\phi_i, \nabla\phi_j \rangle dx \neq 0$ if and only if $\mathcal{N}_i$ and $\mathcal{N}_j$ are both node vertices of the triangle K_m.

Next, observe that since K_m is the m^{th} triangle in the triangulation T_h, it follows that $\mathrm{T}(\alpha, m)$ denotes the labels of the vertices of the triangle K_m. Furthermore, the coordinates of each vertex $\mathrm{T}(\alpha, m)$ are given by

$$\mathrm{Z}\big(i, \mathrm{T}(\alpha, m)\big), \quad i = 1, 2.$$

Once these coordinates are available, we can construct the so-called *local shape functions* ϕ_α, $(\alpha = 1, 2, 3)$: we define ϕ_α as a linear function on K_m with the property that

$$\phi_\alpha\big(\mathcal{N}_{\mathrm{T}(\beta,m)}\big) = \begin{cases} 1, & \text{if } \alpha = \beta, \\ 0, & \text{if } \alpha \neq \beta. \end{cases}$$

An example of a local shape function is displayed in Figure 9.4.

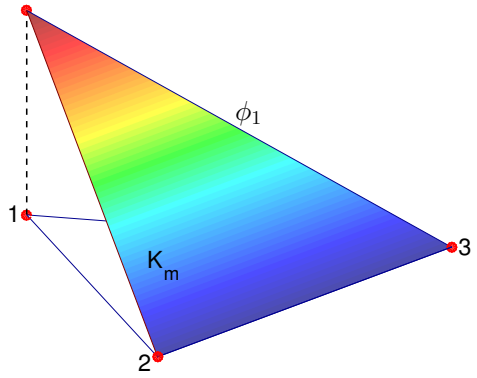


Figure 9.4: An example of the local shape function ϕ_1 for the triangle K_m.

Thus, for the triangle K_m, we compute the element stiffness matrix $A^m = \{A^m_{\alpha,\beta}\}^3_{\alpha,\beta=1}$ as

$$A^m_{\alpha,\beta} = \int_{K_m} \langle \nabla\phi_\alpha, \nabla\phi_\beta \rangle dx. \tag{9.3}$$

We observe that A^m is a symmetric, 3×3 matrix. We also remark that the integral in Equation (9.3) is usually computed using a quadrature rule.

In a similar manner, we compute the element load vector $F^m = \{F^m_\alpha\}^3_{\alpha=1}$ corresponding to the triangle K_m as

$$F^m_\alpha = \int_{K_m} f(x)\phi_\alpha(x)dx.$$

Finally, we can loop over all $m = 1, \ldots, M$ and calculate the element stiffness matrix A^m and the element load vector F^m for each triangle in the triangulation T_h.

Remark 9.1 (Parametric Finite Elements)

In practice, one usually considers a reference element $\hat{K}$ and a mapping $\Phi_K \colon \hat{K} \to K$ that maps the reference element $\hat{K}$ to any given triangle $K \in T_h$ as shown in Figure 9.5.

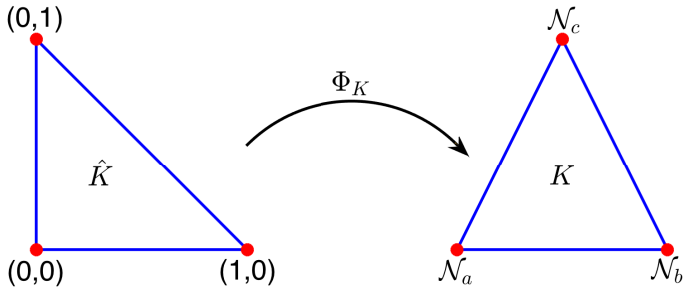


Figure 9.5: Reference element $\hat{K}$ and the mapping Φ_K for some triangle $K \in T_h$.

Here, the function Φ_K is an affine mapping with the property that for all $\hat{x} \in \hat{K}$ it holds that

$$\begin{aligned} x = \Phi_K(\hat{x}) &= \begin{pmatrix} \mathcal{N}_b - \mathcal{N}_a & \mathcal{N}_c - \mathcal{N}_a \end{pmatrix} \hat{x} + \mathcal{N}_a \\ &= \mathrm{J}_K \hat{x} + \mathcal{N}_a. \end{aligned}$$

where $\mathcal{N}_a, \mathcal{N}_b, \mathcal{N}_c$ are the vertices of the triangle $K \in T_h$.

All computations involving the element load vector and element stiffness matrix associated with the triangle K are then performed on the reference element $\hat{K}$, i.e.,

$$F_\alpha^K = \int_K f(x)\phi_\alpha(x)dx = \int_{\hat{K}} f\big(\Phi_K(\hat{x})\big)\hat{\phi}_\alpha(\hat{x})|\mathrm{detJ}_K|d\hat{x}.$$

and similarly,

$$\begin{aligned} A_{\alpha,\beta}^K &= \int_K \langle \nabla\phi_\alpha, \nabla\phi_\beta \rangle dx \\ &= \int_{\hat{K}} \left\langle \mathrm{J}_K^{-\mathsf{T}} \hat{\nabla}\hat{\phi}_\alpha, \mathrm{J}_K^{-\mathsf{T}} \hat{\nabla}\hat{\phi}_\beta \right\rangle \Big|\mathrm{detJ}_K\Big| d\hat{x}. \end{aligned}$$

where $\hat{\phi}_\alpha$, $(\alpha = 1, 2, 3)$ are the local shape functions of the reference element and it can be checked that $\phi_\alpha(x) = \hat{\phi}_\alpha(\hat{x})$ and $\nabla\phi_\alpha(x) = \mathrm{J}_K^{-\mathsf{T}}\hat{\nabla}\hat{\phi}_\alpha(\hat{x})$. Here, $\mathrm{J}_K^{-\mathsf{T}}$ is a short hand notation for $(\mathrm{J}_K^{-1})^{\mathsf{T}}$.

Step 3 Assembly

We next have the key step where the results of the local computations on each triangle are assembled (combined) together to obtain the global

stiffness matrix A and the global load vector F. We use the following pseudo-code (MATLAB notation) to illustrate the assembly process:

Pseudo-Code

A=zeros(N,N);

f=zeros(N);

for $m = 1 : M$ (loop over all triangles)

fetch $A^m = \{A^m_{\alpha,\beta}\}$, $F^m = \{F^m_\alpha\}$, $\alpha, \beta = 1, 2, 3.$

$$A\big(\mathrm{T}(\alpha, m), \mathrm{T}(\beta, m)\big) = A\big(\mathrm{T}(\alpha, m), \mathrm{T}(\beta, m)\big) + A^m_{\alpha,\beta}, \ \alpha, \beta = 1, 2, 3.$$

$$F\big(\mathrm{T}(\alpha, m)\big) = F\big(\mathrm{T}(\alpha, m)\big) + F^m_\alpha, \qquad \alpha = 1, 2, 3.$$

end;

Thus, the non-zero contributions of each triangle K_m, $(m = 1, \ldots, M)$ are assembled into the global stiffness matrix and the global load vector.

Remark 9.2 *For clarity of exposition, we have used a dense matrix format for the global stiffness matrix A in the above pseudo-code. In practice, for the sake of computational efficiency, one should always use a sparse matrix format for A.*

Remark 9.3 *In the case of homogenous boundary conditions, all contributions from boundary nodes should be skipped when assembling the global stiffness matrix A and the global load vector F (i.e. the corresponding rows and columns should be skipped). In pseudo-code this reads:*

Pseudo-Code

compute A and F from above

compute FreeDofs

A=A(FreeDofs,FreeDofs)

F=F(FreeDofs)

Here, FreeDofs is the index set of all interior nodes. For example for the mesh in Figure 9.3, we have FreeDofs$= \{6, 7\}$.

Step 4 Solving the Linear System

Once the global stiffness matrix A and the global load vector F have been obtained, the matrix equation (9.2) can be solved using either a direct solver or an iterative method. More information on such methods can be found in any standard textbook on numerical linear algebra.

Finally, once we have obtained the vector of unknowns $U = \{u_i\}_{i=1}^N$, the FEM approximation of the solution u_h can be computed as

$$u_h(x) = \sum_{i=1}^{N} u_i \phi_i(x).$$

9.1 Treatment of Inhomogeneous Boundary Conditions

We consider the following inhomogeneous boundary value problem involving the Poisson equation:

$$\begin{aligned} -\Delta u &= f \qquad \text{in } \Omega, \\ u &= g \qquad \text{on } \partial\Omega, \end{aligned} \tag{9.4}$$

where $g\colon \partial\Omega \to \mathbb{R}$ is the given boundary data.

We can approach this problem as follows: we consider an extension of the Dirichlet data:

$$\tilde{g}\colon \Omega \to \mathbb{R}, \qquad \tilde{g}\big(\partial\Omega\big) = g.$$

Next, we define the function $u_0\colon \Omega \to \mathbb{R}$ as

$$u_0 = u - \tilde{g}.$$

This implies that

$$u = u_0 + \tilde{g},$$

and furthermore,

$$u_0|_{\partial\Omega} = u|_{\partial\Omega} - \tilde{g} = g - g = 0.$$

Thus, (9.4) can be rewritten as

$$f = -\Delta u = -\Delta\big(u_0 + \tilde{g}\big).$$

We may therefore consider the following variational problem:

Find $u_0 \in H_0^1\big(\Omega\big)$ *such that for all* $v \in H_0^1(\Omega)$ *it holds that*

$$\big(u_0, v\big)_{H_0^1(\Omega)} = \big(f, v\big)_{L^2(\Omega)} - \big(\tilde{g}, v\big)_{H_0^1(\Omega)} \tag{9.5}$$

where $\big(\cdot,\cdot\big)_{H_0^1(\Omega)}$ *is the* H_0^1 *inner product and* $\big(\cdot,\cdot\big)_{L^2(\Omega)}$ *is the* L^2 *inner product.*

Note that (9.5) is very similar to the weak form of the Poisson equation with homogeneous boundary conditions (8.3). Indeed, only the right side of both equations is different.

We can now solve the variational problem (9.5). The solution u of the inhomogeneous boundary value problem (9.4) is then given by $u = u_0 + \tilde{g}$.

9.1.1 Finite Element Formulation

We now mimic the preceding calculations on the discrete level:

We define $\tilde{g}_h \colon \Omega \to \mathbb{R}$ as the continuous, piecewise linear function with the property that for all nodes $\{x_i\}_{i=1}^N$ of the triangulation it holds that

$$\tilde{g}_h(x_i) = \begin{cases} g(x_i) & \text{if } x_i \text{ is a boundary node,} \\ 0 & \text{otherwise.} \end{cases}$$

Thus, it holds that

$$\tilde{g}_h(x) = \sum_{\substack{x_i \text{ is bd.} \\ \text{node}}} g(x_i)\phi_i(x).$$

Note that since the function g is not necessarily a piecewise linear function, we don't have $g|_{\partial\Omega} = \tilde{g}_h|_{\partial\Omega}$, in general, but only $g|_{\partial\Omega} \approx \tilde{g}_h|_{\partial\Omega}$.

Then, the variational formulation of the Finite Element method is of the form:

Find $u_{0,h} \in V_h$ *such that for all* $v \in V_h$ *it holds that*

$$\big(u_{0,h}, v\big)_{H_0^1(\Omega)} = \big(f, v\big)_{L^2(\Omega)} - \big(\tilde{g}_h, v\big)_{H_0^1(\Omega)}. \tag{9.6}$$

It is now a simple matter of solving the above variational problem. The solution u_h is then given by

$$u_h = u_{0,h} + \tilde{g}_h.$$

Note that we can also write the solution u_h as

$$u_h(x) = \sum_{i=1}^{N} u_i \phi_i(x),$$

where $u_i = g(x_i)$ for all boundary nodes x_i.

Algorithmically, the finite element formulation described above leads to the following pseudo-code:

Pseudo-Code

compute A and F from above

compute FreeDofs

U=zeros(N,1)

U(i)=$g(x_i)$ for all boundary nodes x_i.

F=F−AU

A=A(FreeDofs,FreeDofs)

F=F(FreeDofs)

Solve AU(FreeDofs)=F

10 Parabolic Partial Differential Equations

The *Heat equation* (or diffusion equation) is the prototypical example of a parabolic partial differential equation. Let $\Omega \subseteq \mathbb{R}^m$ for some $m \in \mathbb{N}$, let $T \in [0, \infty)$ be some fixed, final time and let $u_0 \colon \Omega \to \mathbb{R}$ be some known function. Then the heat equation in several space dimensions is given by

$$\begin{aligned} u_t - \Delta u &= 0, && \text{on } \Omega \times (0, T), \\ u(x, 0) &= u_0(x), && \text{on } \Omega. \end{aligned} \tag{10.1}$$

Note that Equation (10.1) must be supplemented with suitable boundary conditions.

We begin by considering the case of one spatial dimension ($m = 1$) and set the domain $\Omega = (0, 1)$. Then, the heat equation in one spatial dimension is given by

$$\begin{aligned} u_t - u_{xx} &= 0, && \text{on } (0, 1) \times (0, T), \\ u(x, 0) &= u_0(x), && \text{on } (0, 1), \\ u(0, t) &= u(1, t) = 0, && \text{on } (0, T), \end{aligned} \tag{10.2}$$

where we have assumed homogeneous Dirichlet boundary conditions.

10.1 Exact Solutions to the Heat Equation

In order to obtain an exact, analytical solution for the heat equation, we may use the so-called method of *separation of variables*. We therefore use the ansatz

$$u = u(x, t) = \mathcal{T}(t)\mathcal{X}(x).$$

Then, it holds that

$$\begin{aligned} u_t &= \mathcal{T}'(t)\mathcal{X}(x), \\ u_{xx} &= \mathcal{T}(t)\mathcal{X}''(x). \end{aligned}$$

Therefore, the PDE (10.2) reduces to

$$\mathcal{T}'(t)\mathcal{X}(x) = \mathcal{T}(t)\mathcal{X}''(x),$$

and hence, it holds that

$$\frac{\mathcal{T}'(t)}{\mathcal{T}(t)} = \frac{\mathcal{X}''(x)}{\mathcal{X}(x)}. \tag{10.3}$$

Next, observe that the left side of Equation (10.3) is a function of only time (t) and the right side of Equation (10.3) is a function of only the spatial variable (x). Thus, Equation (10.3) can only be satisfied if there exists some constant $\lambda_k \in \mathbb{R}$ such that for all $x \in (0,1)$ and for all $t \in (0,T)$ it holds that

$$\frac{\mathcal{T}'(t)}{\mathcal{T}(t)} = \frac{\mathcal{X}''(x)}{\mathcal{X}(x)} = -\lambda_k. \tag{10.4}$$

Now, observe that Equation (10.4) combined with the boundary conditions from (10.2) implies that

$$\begin{aligned} \mathcal{X}''(x) + \lambda_k \mathcal{X}(x) &= 0, \qquad \text{for } x \in (0,1), \\ \mathcal{X}(0) = \mathcal{X}(1) &= 0. \end{aligned} \tag{10.5}$$

Similarly, Equation (10.4) also implies that

$$\mathcal{T}'(t) + \lambda_k \mathcal{T}(t) = 0, \quad \text{for } t \in (0,T), \tag{10.6}$$

Thus, using the separation of variables ansatz has allowed us to reduce the PDE (10.2) to a pair of ODEs, which can be solved exactly. Indeed, it can be shown that the ODE (10.5) has a general solution given by

$$\mathcal{X}_k(x) = \sin(k\pi x), \quad \lambda_k = (k\pi)^2 \quad \forall k \in \mathbb{Z}. \tag{10.7}$$

Plugging (10.7) into the ODE (10.6), we obtain that for all $k \in \mathbb{Z}$ it holds that

$$\mathcal{T}'(t) = -(k\pi)^2 \mathcal{T}(t) \implies \mathcal{T}(t) = \mathrm{e}^{-(k\pi)^2 t}. \tag{10.8}$$

Finally, combining (10.7) and (10.8) into the separation of variables ansatz, we obtain that

$$u(x,t) = \mathcal{X}(x)\mathcal{T}(t) = \mathrm{e}^{-(k\pi)^2 t} \sin(k\pi x),$$

is a general solution of the PDE

$$\begin{aligned} u_t - u_{xx} &= 0, && \text{on } (0,1) \times (0,T), \\ u(0,t) = u(1,t) &= 0, && \text{on } (0,T). \end{aligned}$$

It now remains to obtain a unique, particular solution for the given initial data. To this end, we observe that for all $k, m \in \mathbb{N}$ it holds that

$$\int_0^1 \sin(k\pi x)\sin(m\pi x)dx = \begin{cases} 0, & \text{if } k \neq m, \\ \frac{1}{2}, & \text{if } k = m. \end{cases}$$

Hence, the set $\{\sin(k\pi x)\}_{k\in\mathbb{N}}$ (known as the *Fourier sine series*) forms a basis for the function space $L^2\big((0,1)\big)$. Thus, for any function $f \in L^2\big((0,1)\big)$, we have the *Fourier expansion* given by

$$f(x) = \sum_{k=1}^{\infty} f_k \sin(k\pi x),$$

where

$$f_k = 2\int_0^1 f(x)\sin(k\pi x)dx.$$

Thus, for any initial condition u_0 in (10.2) such that $u_0 \in L^2\big((0,1)\big)$ it holds that

$$u_0(x) = \sum_{k=1}^{\infty} u_k^0 \sin(k\pi x),$$

where

$$u_k^0 = 2\int_0^1 u_0(x)\sin(k\pi x)dx.$$

Note that the convergence of the above infinite series is a consequence of the theory of Fourier series.

It can now be shown that the solution u of the heat equation in one spatial dimension (10.2) is given by

$$u(x,t) = \sum_{k=1}^{\infty} u_k^0 \mathrm{e}^{-(k\pi)^2 t}\sin(k\pi x). \tag{10.9}$$

Indeed, we observe that the time derivative and second spatial derivative of u are given by

$$u_t = \sum_{k=1}^{\infty} -(k\pi)^2 u_k^0 \mathrm{e}^{-(k\pi)^2 t}\sin(k\pi x),$$

$$u_{xx} = \sum_{k=1}^{\infty} -(k\pi)^2 u_k^0 \mathrm{e}^{-(k\pi)^2 t}\sin(k\pi x),$$

and therefore $u_t = u_{xx}$. Furthermore, for all $t \in (0, T)$ it holds that

$$u(0, t) = u(1, t) = 0.$$

Finally, for all $x \in (0, 1)$ it holds that

$$u(x, 0) = \sum_{k=1}^{\infty} u_k^0 \sin(k\pi x) = u_0(x).$$

In fact, it can be shown that (10.9) is the unique solution of the heat equation (10.2).

10.1.1 Evaluation of the Exact Solution

The following algorithm can be used to evaluate the exact solution (10.9) of the heat equation (10.2):

Given $u_0 \in L^2\big((0, 1)\big)$

Step 1

Expand u_0 using the truncated Fourier series i.e.,

$$u_0^N(x) = \sum_{k=1}^{N} u_k^0 \sin(k\pi x).$$

We remark that the error $|u_0 - u_0^N|$ is small for large values of N if the function u_0 is smooth and satisfies $u_0(0) = u_0(1) = 0$.

Next, in order to calculate the coefficients $\{u_k^0\}_{k=1}^N$, we use a quadrature rule to approximate

$$u_k^0 = 2 \int_0^1 u_0(x) \sin(k\pi x) dx.$$

Step 2

The approximate solution to the heat equation (10.2) can then be obtained by truncating the formula (10.9):

$$u_N(x, t) = \sum_{k=1}^{N} u_k^0 \mathrm{e}^{-(k\pi)^2 t} \sin(k\pi x).$$

Note that this implies that there are now **two** sources of error in the approximate solution:

- Error due to the finite-truncation of the Fourier sine series.
- Error due to the use of quadrature rules to approximate integrals.

Given these multiple sources of errors, we might as well replace the exact solution with an approximate solution obtained using some numerical method, which can then be applied in a more general setting.

In order to design suitable numerical method however, we require some qualitative properties of solutions to the heat equation (10.2).

10.2 Energy Estimate

Let u be a solution of the heat equation (10.2). We define the energy function $\mathcal{E}\colon [0,T] \to \mathbb{R}$ as

$$\mathcal{E}(t) := \frac{1}{2}\int_0^1 |u(x,t)|^2 dx.$$

It then follows that

$$\begin{aligned}
\frac{d\mathcal{E}}{dt} &= \frac{1}{2}\int_0^1 (u^2)_t dx \\
&= \int_0^1 u u_t dx && \text{(Using the Chain Rule)} \\
&= \int_0^1 u u_{xx} dx && \text{(Using Equation (10.2))} \\
&= -\int_0^1 u_x^2 dx + u u_x|_0^1 && \text{(Integration by parts)} \\
&= -\int_0^1 u_x^2 dx && \text{(Using Boundary Conditions).}
\end{aligned}$$

We can thus conclude that

$$\frac{d\mathcal{E}}{dt} = -\int_0^1 u_x^2 dx \leq 0.$$

Therefore, for all $t \in (0,T)$ it holds that

$$\mathcal{E}(t) \leq \mathcal{E}(0). \tag{10.10}$$

In other words, the energy of the solution u to the heat equation (10.2) decreases in time.

10.2.1 Consequence of the Energy Estimate

Uniqueness

Let $u, \bar{u}$ be two solutions of the heat equation (10.2) for the same initial and boundary conditions, and let $w := u - \bar{u}$.

Then, clearly w satisfies the following heat equation:

$$\begin{aligned} w_t - w_{xx} &= 0, && \text{on } (0,1) \times (0,T), \\ w(x,0) &\equiv 0, && \text{on } (0,1), \\ w(0,t) = w(1,t) &= 0, && \text{on } (0,T). \end{aligned}$$

Next, we define the energy function $\bar{\mathcal{E}}$ associated with w as

$$\bar{\mathcal{E}}(t) = \frac{1}{2} \int_0^1 w^2(x,t) dx.$$

The energy estimate (10.10) then implies that for all $t \in (0,T)$ it holds that

$$\bar{\mathcal{E}}(t) \leq \bar{\mathcal{E}}(0).$$

Therefore, it follows that for all $t \in (0,T)$ it holds that

$$\begin{aligned} & \int_0^1 w^2(x,t) dx \leq \int_0^1 w^2(x,0) dx \\ \implies & \int_0^1 w^2(x,t) dx \leq 0 \quad \text{(Using the Initial Conditions)} \\ \implies & w(x,t) \equiv 0 \\ \implies & u(x,t) = \bar{u}(x,t). \quad \text{(Using the Definition of } w\text{)}. \end{aligned}$$

Hence, the heat equation (10.2) has a unique solution.

10.3 Maximum Principles

We observe that the energy estimate described in the previous section implies a bound on the L^2 norm of the solution u to the heat equation (10.2) at any given time $t \in (0,T)$:

$$\sup_{0 \leq t \leq T} \|u(\cdot,t)\|_{L^2((0,1))} \leq \|u_0\|_{L^2((0,1))}.$$

In addition to satisfying an L^2 bound, the solution u also satisfies a *maximum principle*, which provides bounds on the L^∞ (maximum) norm of the solution. Indeed, we have the following lemma regarding a maximum principle:

Lemma 10.1 (Maximum Principle) *Let u be a solution of the heat equation (10.2). Then for all $x \in [0,1]$ and for all $t \in [0,T]$ it holds that*

$$\min\left(0, \min_{\tilde{x}}(u_0(\tilde{x}))\right) \leq u(x,t) \leq \max\left(0, \max_{\tilde{x}}(u_0(\tilde{x}))\right). \tag{10.11}$$

We observe that this lemma implies that the maximum (and minimum) of the solution u of the heat equation (10.2) is attained on the parabolic boundary i.e., the initial line $t = 0$ or the two side boundaries $x = 0,\ x = 1$, as shown in Figure 10.1.

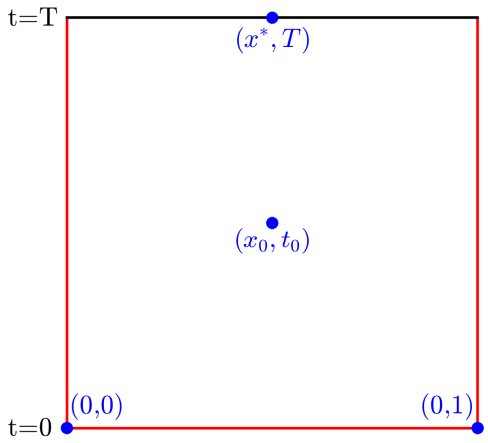


Figure 10.1: Domain for the heat equation (10.2) with the parabolic boundary in red.

Note that for simplicity, we will restrict our proof to the maximum principle but the minimum principle can be proven analogously.

Proof The proof proceeds by contradiction. Let $x_0 \in (0,1)$ and $t_0 \in (0,T)$ (see Figure 10.1) and assume that (x_0, t_0) is a strict maximal point of the function u. In other words, we assume that for all $x \in (0,1)$ and for all $t \in (0,T)$, it holds that

$$u(x_0, t_0) > u(x,t).$$

Clearly, it holds that

$$\begin{aligned} u_t(x_0, t_0) &\equiv 0, \\ u_{xx}(x_0, t_0) &< 0. \quad \big((x_0, t_0) \text{ is a maximal point}\big), \end{aligned}$$

and hence,

$$u_t(x_0, t_0) - u_{xx}(x_0, t_0) > 0.$$

This is a contradiction as u solves the heat equation (10.2). Thus, a strict maximum cannot be attained at an interior point $(x_0, t_0) \in (0, 1) \times (0, T)$.

Next, let $x^* \in (0, 1)$ and suppose that a strict maximum of the function u is attained at the point (x^*, T) (shown in Figure 10.1), where T is the final time.

Clearly, it holds that

$$u_{xx}(x^*, T) < 0$$

and furthermore,

$$u_t(x^*, T) = \lim_{h^+ \to 0} \frac{u(x^*, T) - u(x^*, T - h)}{h} > 0.$$

Hence, one again we have that

$$u_t(x_0, t_0) - u_{xx}(x_0, t_0) > 0,$$

which contradicts the fact that u solves the heat equation (10.2). Thus, a strict maximum cannot be attained at the final time $t = T$.

We can therefore conclude that a strict maximum of the function u can only be attained at the parabolic boundary (shown in Figure 10.1). However, we have not ruled out the possibility of a *non-strict* maximum at the interior point (x_0, t_0) or the point (x^*, T). To this end, we proceed as follows:

Let $\epsilon > 0$ and define the function $u^\epsilon \colon [0, 1] \times [0, T] \to \mathbb{R}$ as

$$u^\epsilon(x, t) = u(x, t) - \epsilon t.$$

Assume that for a fixed ϵ, the function u^ϵ attains a maximum at the point (x_0, t_0). It then follows that

$$\begin{aligned} 0 = u^\epsilon_t(x_0, t_0) &= u_t(x_0, t_0) - \epsilon \\ \implies u_t(x_0, t_0) &= \epsilon > 0. \end{aligned}$$

Moreover, it also holds that

$$\begin{aligned} & u_{xx}(x_0, t_0) = u^{\epsilon}_{xx}(x_0, t_0) \leq 0 \\ \Longrightarrow\ & u_{xx}(x_0, t_0) \leq 0. \end{aligned}$$

Hence, it follows that

$$u_t(x_0, t_0) - u_{xx}(x_0, t_0) \geq \epsilon > 0,$$

which once again contradicts the fact that u solves the heat equation (10.2). Therefore, u^{ϵ} cannot attain a maximum at the point (x_0, t_0).

A similar argument also holds for the point (x^*, T). We therefore conclude that for all $\epsilon > 0$ it holds that

$$\max_{\substack{0 \leq x \leq 1, \\ 0 \leq t \leq T}} u^{\epsilon}(x, t) \leq \max\big(0, \max_{x}(u^{\epsilon}(x, 0))\big),$$

and therefore since

$$u^{\epsilon}(x, 0) = u(x, 0) - \epsilon \cdot 0 = u(x, 0),$$

it holds that

$$\max_{\substack{0 \leq x \leq 1, \\ 0 \leq t \leq T}} u^{\epsilon}(x, t) \leq \max\big(0, \max_{x}(u(x, 0))\big).$$

As the right side of the last inequality is now independent of ϵ, we can take the limit $\epsilon \to 0$ on the left side to obtain

$$\max_{\substack{0 \leq x \leq 1, \\ 0 \leq t \leq T}} u(x, t) \leq \max\big(0, \max_{x}(u_0(x))\big).$$

This proves the maximum principle. □

We are now in a position to discuss numerical methods for approximating solutions to the heat equation (10.2). Given the above results, we would like the approximate solutions produced by our numerical methods to satisfy a discrete version of the energy inequality and/or the maximum principle.

10.4 Finite Difference Schemes for the Heat Equation

We consider the one-dimensional heat equation:

$$\begin{aligned} u_t - u_{xx} &= 0, && \text{on } (0,1)\times(0,T), \\ u(x,0) &= u_0(x), && \text{on } (0,1), \\ u(0,t) &= u(1,t) = 0, && \text{on } (0,T), \end{aligned} \tag{10.12}$$

In order to approximate solutions to the heat equation (10.12), we derive a finite difference scheme using the following steps:

Step 1 Discretising the Domain

Let the grid size $\Delta x > 0$ and let $N = \frac{1}{\Delta x} - 1$. Then we discretise the spatial domain $[0,1]$ into $N+2$ equally spaced points by setting

$$\begin{aligned} x_0 &= 0, \\ x_j &= j\Delta x, \quad j = 1,\dots,N, \\ x_{N+1} &= 1. \end{aligned}$$

Similarly, let the time step $\Delta t > 0$ and let $M = \frac{T}{\Delta t} - 1$. Then we discretise the temporal domain $[0,T]$ into $M+2$ equally spaced points by setting

$$\begin{aligned} t_0 &= 0, \\ t^n &= n\Delta t, \quad n = 1,\dots,M, \\ t^{M+1} &= T. \end{aligned}$$

An example of the resulting grid is displayed in Figure 10.2.

Step 2 Discretising the Solution u

We next approximate the function u, which solves the heat equation (10.12), with point values by setting

$$U_j^n \approx u(x_j, t^n).$$

Step 3 Discretising the Derivatives

In addition to discretising the solution u, it is also necessary to discretise the spatial and temporal derivatives of the function u.

Spatial derivative Similar to the case of the Poisson equation, we approximate the spatial derivative using a central difference approximation:

$$u_{xx}(x_j, t^n) \approx \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{\Delta x^2}.$$

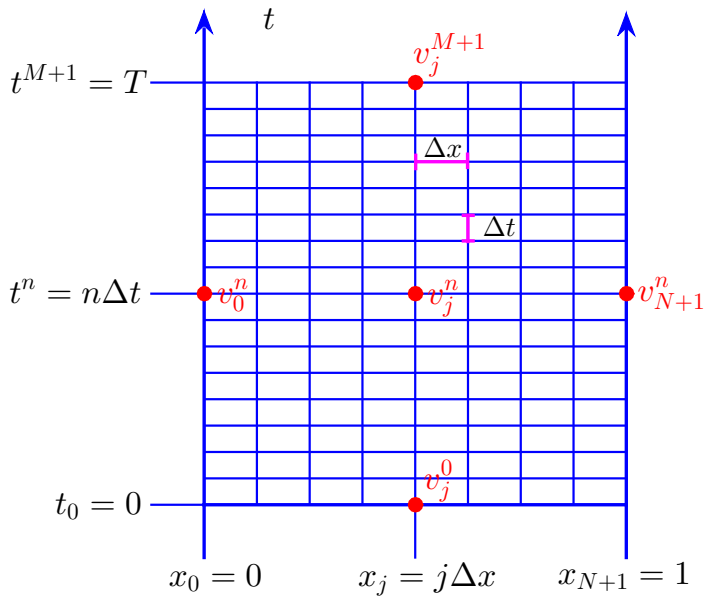


Figure 10.2: An example of a grid for the one-dimensional heat equation (10.12)

Temporal derivative There are many possible choices for approximating the time derivative of the solution u. The simplest choice is to use a forward difference approximation:

$$u_t(x_j, t^n) \approx \frac{U_j^{n+1} - U_j^n}{\Delta t}.$$

Step 4 The Finite Difference Scheme

A finite difference for approximating solutions to the heat equation (10.12) is then given by

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} - \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{\Delta x^2} = 0, \tag{10.13}$$

for all $j = 1, \ldots, N$ and for all $n = 0, \ldots M$.

Defining the constant $\lambda = \frac{\Delta t}{\Delta x^2}$, we can rewrite (10.13) as

$$U_j^{n+1} = (1 - 2\lambda)U_j^n + \lambda U_{j+1}^n + \lambda U_{j-1}^n.$$

Furthermore, the initial conditions and the boundary conditions of the heat equation (10.12) can be implemented by setting

$$\begin{aligned} U_j^0 &= u(x_j, 0) = u_0(x_j), \quad && \forall\ 1 \le j \le N, \\ U_0^n &= U_{N+1}^n \equiv 0 && \forall\ 0 \le n \le M + 1. \end{aligned}$$

The implementation of the finite difference scheme (10.13) is straightforward. Given the approximate solution values $\{U_j^n\}_{1\leq j\leq N}$ at the time level t^n, we compute the approximate solutions values $\{U_j^{n+1}\}_{1\leq j\leq N}$ at the next time level t^{n+1} using the update formula (10.13) and the boundary conditions.

Remark 10.2 (Terminology)

The finite difference scheme (10.13) *is termed an Explicit finite difference scheme since the Explicit (Forward) Euler method is used for time stepping.*

Remark 10.3 (Notation) *In order to simplify the notation in some future sections, we introduce the following notation*

$$\begin{aligned}
\textit{Forward difference in space:} \quad & \mathrm{D}_x^+ w_j^n = \frac{w_{j+1}^n - w_j^n}{\Delta x} \\
\textit{Backward difference in space:} \quad & \mathrm{D}_x^- w_j^n = \frac{w_j^n - w_{j-1}^n}{\Delta x} \\
\textit{Forward difference in time:} \quad & \mathrm{D}_t^+ w_j^n = \frac{w_j^{n+1} - w_j^n}{\Delta t} \\
\textit{Backward difference in time:} \quad & \mathrm{D}_t^- w_j^n = \frac{w_j^n - w_j^{n-1}}{\Delta t}
\end{aligned}$$

The finite difference scheme (10.13) *can then be recast as*

$$\mathrm{D}_t^+ U_n^j - \mathrm{D}_x^- \mathrm{D}_x^+ U_n^j = 0, \tag{10.14}$$

for all $j = 1, \ldots, N$ *and for all* $n = 0, \ldots M$.

10.4.1 Numerical Results

Example 1

As a first numerical example, we consider the following boundary value problem involving the one-dimensional heat equation:

$$\begin{aligned}
u_t - u_{xx} &= 0, && \text{on } (0,1)\times(0,T), \\
u(x,0) &= \sin(2\pi x), && \text{on } (0,1), \\
u(0,t) &= u(1,t) = 0, && \text{on } (0,T),
\end{aligned} \tag{10.15}$$

It can be shown that the exact solution to the BVP (10.15) is given by $u(x,t) = \mathrm{e}^{-4\pi^2 t}\sin(2\pi x)$. We use the explicit finite difference method (10.13) to approximate solutions to this problem for different values of the time step Δt and grid size Δx.

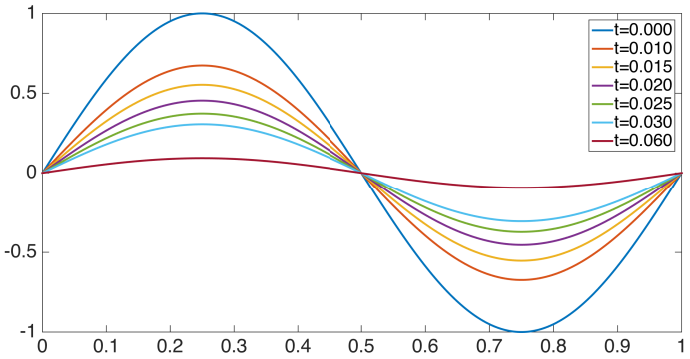
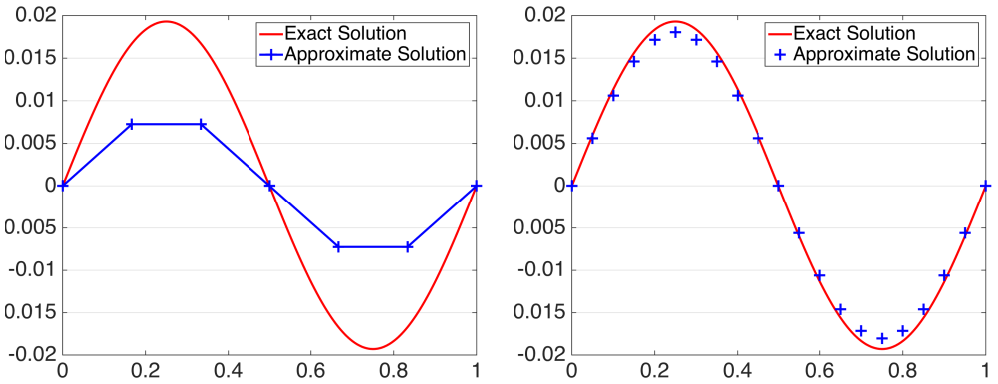


Figure 10.3: Exact solution plots of the BVP (10.15) at different times t.

Figure 10.3 displays the exact solution of the BVP (10.15) at different times. Clearly, the solution decays to zero over time.

Figure 10.4 displays the exact and approximate solution at time $t = 0.1$, and indicates that the quality of the solution approximation improves as the grid spacing and time step are refined.



(a) $\Delta x = \frac{1}{6}$, $\Delta t = \frac{1}{80} \implies \lambda = 0.45$. (b) $\Delta x = \frac{1}{20}$, $\Delta t = \frac{1}{800} \implies \lambda = 0.5$.

Figure 10.4: Exact solution and approximate solution plots at time $t = 0.1$ for the BVP (10.15) using the explicit finite difference method for two different values of $\Delta x, \Delta t$.

Example 2

As a second numerical example, we consider the following boundary value

problem involving the one-dimensional heat equation:

$$\begin{aligned} u_t - u_{xx} &= 0, && \text{on } (0,1)\times(0,T), \\ u(x,0) &= \min\big(2x, 2-2x\big), && \text{on } (0,1), \\ u(0,t) &= u(1,t) = 0, && \text{on } (0,T), \end{aligned} \tag{10.16}$$

The exact solution to the BVP (10.16) can be calculated using the Fourier sine series and is given by

$$u(x,t) = \sum_{k \text{ even},\, k=1}^{\infty} \left(\frac{(-1)^{\frac{k-1}{2}}}{k^2\pi^2} 8 \sin(k\pi x) \mathrm{e}^{-k^2\pi^2 t} \right).$$

We once again use the explicit finite difference method (10.13) to approximate solutions to this problem for different values of the time step Δt and grid size Δx.

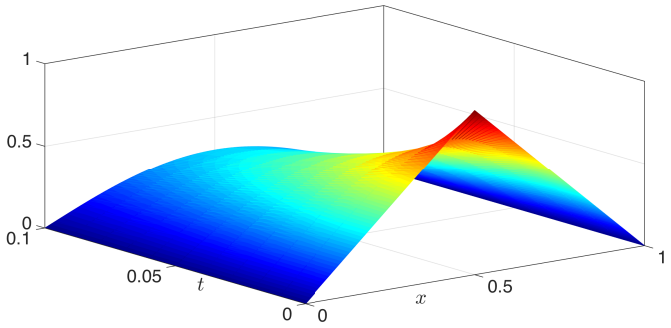


Figure 10.5: 3-dimensional plot of the exact solution to the BVP (10.16).

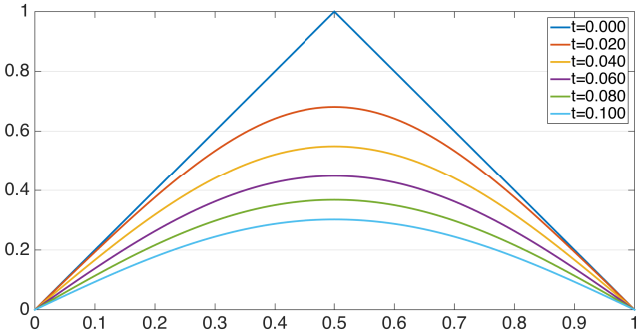
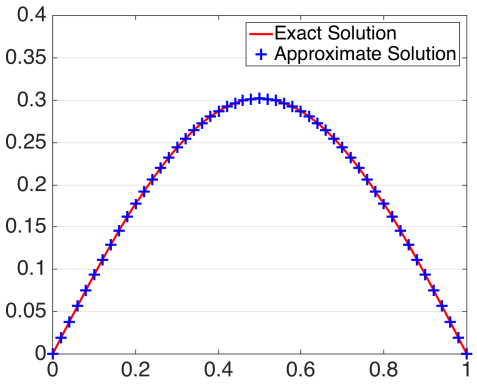


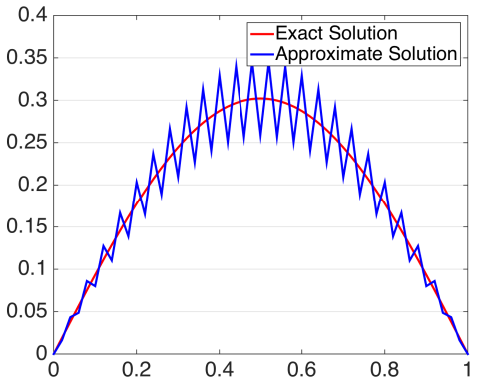
Figure 10.6: Exact solution plots of the BVP (10.16) at different times t.

Figure 10.5 displays a 3-dimensional plot of the exact solution of the BVP (10.15) as a function of space (x) and time (t). In addition, Figure 10.6

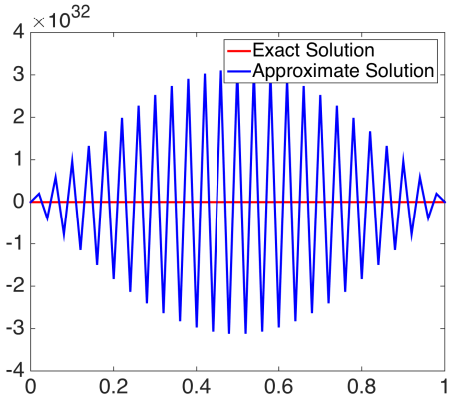
displays the exact solution of the BVP (10.16) at different times. Once again, we observe that the solution decays to zero over time.



(a) $\Delta x = \frac{1}{50}$, $\Delta t = \frac{1}{5000} \implies \lambda = 0.5$.



(b) $\Delta x = \frac{1}{50}$, $\Delta t = \frac{1}{4975} \implies \lambda = 0.503$.



(c) $\Delta x = \frac{1}{50}$, $\Delta t = \frac{2}{9091} \implies \lambda = 0.55$.

Figure 10.7: Exact solution and approximate solution plots at time $t = 0.1$ for the BVP (10.16) using the explicit finite difference method for 3 different values of $\Delta t, \lambda$.

Figure 10.7 displays the exact and approximate solution at time $t = 0.1$ for different values of the time step Δt and the parameter λ. We immediately observe that the approximate solution produced by the finite difference scheme (10.13) can become unstable for certain values of Δt and λ.

The results of the numerical experiments therefore indicate that the quality of the solution approximation strongly depends on the choice of the time step Δt or, equivalently, the choice of the parameter λ.

In order to gain a clearer understanding of these results, it is necessary to

perform a stability analysis of the finite difference scheme (10.13).

10.4.2 Discrete Energy Stability

Recall that we have previously shown in Section 10.2 that the exact solutions of the heat equation (10.2) are energy stable, i.e., for all $t \in (0, T)$, it holds that

$$\mathcal{E}(t) \leq \mathcal{E}(0), \tag{10.17}$$

where

$$\mathcal{E}(t) = \frac{1}{2} \int_0^1 |u(x,t)|^2 dx.$$

Therefore, we should analyse the approximate solutions produced by the finite difference scheme to determine if these solutions satisfy a discrete version of the energy inequality (10.17).

To this end, we define for all $n = 0, \ldots, M+1$, the discrete energy $\mathcal{E}^n$ as

$$\mathcal{E}^n = \frac{\Delta x}{2} \sum_{j=1}^{N} \left(U_j^n\right)^2. \tag{10.18}$$

Next, we state some elementary identities that we require for our calculations. We remark that these identities are straightforward to prove.

Discrete Chain Rule

For all $n = 0, \ldots, M$ and for all $j = 1, \ldots, N$, it holds that

$$w_j^n \mathrm{D}_t^+ w_j^n = \frac{1}{2} \mathrm{D}_t^+ \left(\left(w_j^n\right)^2 \right) - \frac{\Delta t}{2} \left(\mathrm{D}_t^+ w_j^n \right)^2. \tag{10.19}$$

Proof By definition, it holds that

$$\begin{aligned} w_j^n \mathrm{D}_t^+ w_j^n &= \frac{w_j^n}{\Delta t} \left(w_j^{n+1} - w_j^n \right) \\ &= \frac{1}{2\Delta t} \left(2 w_j^n w_j^{n+1} - 2\left(w_j^n\right)^2 - \left(w_j^{n+1}\right)^2 + \left(w_j^{n+1}\right)^2 \right) \\ &= \frac{1}{2\Delta t} \left(\left(w_j^{n+1}\right)^2 - \left(w_j^n\right)^2 - \left(w_j^{n+1} - w_j^n\right)^2 \right) \\ &= \frac{1}{2} \mathrm{D}_t^+ \left(\left(w_j^n\right)^2 \right) - \frac{\Delta t}{2} \left(\mathrm{D}_t^+ w_j^n \right)^2. \end{aligned}$$

□

Summation by Parts

For all $n = 0, \ldots, M+1$ and for all $j = 1, \ldots, N$, it holds that

$$\sum_{j=1}^{N} w_j^n \mathrm{D}_x^- \mathrm{D}_x^+ w_j^n = -\sum_{j=0}^{N} \Big(\mathrm{D}_x^+ w_j^n\Big)^2 + \frac{1}{\Delta x}\Big(w_{N+1}^n \mathrm{D}_x^+ w_N^n - w_0^n \mathrm{D}_x^+ w_0^n\Big). \tag{10.20}$$

The summation by parts formula (10.20) can be proven using mathematical induction.

Let us now consider the finite difference scheme (10.14) and mimic the derivation of the energy inequality for the continuous case by multiplying both sides of Equation (10.14) with v_j^n. We then obtain

$$U_j^n \mathrm{D}_t^+ U_j^n = U_j^n \mathrm{D}_x^- \mathrm{D}_x^+ U_j^n.$$

The discrete chain rule (10.19) then implies that

$$\frac{1}{2}\mathrm{D}_t^+\Big(\big(U_j^n\big)^2\Big) = \frac{\Delta t}{2}\Big(\mathrm{D}_t^+ U_j^n\Big)^2 + U_j^n \mathrm{D}_x^- \mathrm{D}_x^+ U_j^n.$$

Next, using (10.14) to expand the first term on the right side, we obtain

$$\begin{aligned}\frac{1}{2}\mathrm{D}_t^+\Big(\big(U_j^n\big)^2\Big) &= \frac{\Delta t}{2}\Big(\mathrm{D}_x^- \mathrm{D}_x^+ U_j^n\Big)^2 + U_j^n\Big)^2 + U_j^n \mathrm{D}_x^- \mathrm{D}_x^+ U_j^n \\ &= \frac{\Delta t}{2\Delta x^2}\Big(\mathrm{D}_x^+ U_j^n - \mathrm{D}_x^- U_j^n\Big)^2 + U_j^n \mathrm{D}_x^- \mathrm{D}_x^+ U_j^n, \end{aligned} \tag{10.21}$$

where the last equation is a consequence of the following identity:

$$\mathrm{D}_x^- \mathrm{D}_x^+ U_j^n = \frac{1}{\Delta x}\Big(\mathrm{D}_x^+ U_j^n - \mathrm{D}_x^- U_j^n\Big).$$

We now multiply both sides of Equation (10.21) with $\Delta x \Delta t$ and sum over all $j = 1, \ldots, N$ to obtain

$$\underbrace{\frac{\Delta x \Delta t}{2}\sum_{j=1}^{N}\mathrm{D}_t^+\Big(\big(U_j^n\big)^2\Big)}_{T_1} = \underbrace{\frac{\Delta t^2}{2\Delta x}\sum_{j=1}^{N}\Big(\mathrm{D}_x^+ U_j^n - \mathrm{D}_x^- U_j^n\Big)^2}_{T_2} + \underbrace{\Delta x \Delta t \sum_{j=1}^{N} U_j^n \mathrm{D}_x^- \mathrm{D}_x^+ U_j^n}_{T_3}.$$

We can now attempt to simplify each term separately:

$$T_1 := \frac{\Delta x \Delta t}{2} \sum_{j=1}^{N} \mathrm{D}_t^+ \left(\left(U_j^n \right)^2 \right) = \frac{\Delta x}{2} \sum_{j=1}^{N} \left(U_j^{n+1} \right)^2 - \frac{\Delta x}{2} \sum_{j=1}^{N} \left(U_j^n \right)^2$$
$$= \mathcal{E}^{n+1} - \mathcal{E}^n.$$

$$T_3 := \Delta x \Delta t \sum_{j=1}^{N} U_j^n \mathrm{D}_x^- \mathrm{D}_x^+ U_j^n$$
$$= - \Delta x \Delta t \sum_{j=0}^{N} \left(\mathrm{D}_x^+ U_j^n \right)^2 \quad \text{(using summation by parts (10.20) and B.C.).}$$

$$T_2 := \frac{\Delta t^2}{2\Delta x} \sum_{j=1}^{N} \left(\mathrm{D}_x^+ U_j^n - \mathrm{D}_x^- U_j^n \right)^2 \leq \frac{\Delta t^2}{\Delta x} \sum_{j=1}^{N} \left(\mathrm{D}_x^+ U_j^n \right)^2 + \frac{\Delta t^2}{\Delta x} \sum_{j=1}^{N+1} \left(\mathrm{D}_x^- U_j^n \right)^2$$
$$\leq \frac{\Delta t^2}{\Delta x} \sum_{j=0}^{N} \left(\mathrm{D}_x^+ U_j^n \right)^2 + \frac{\Delta t^2}{\Delta x} \sum_{j=1}^{N+1} \left(\mathrm{D}_x^- U_j^n \right)^2.$$

This inequality is a consequence of the following simple algebraic relation: for all $a, b \in \mathbb{R}$ it holds that

$$(a-b)^2 \leq 2(a^2 + b^2).$$

Next, observe that a simple change of indices implies that

$$\sum_{j=0}^{N} \left(\mathrm{D}_x^+ U_j^n \right)^2 = \sum_{j=1}^{N+1} \left(\mathrm{D}_x^- U_j^n \right)^2,$$

and therefore it follows that

$$T_2 \leq \frac{2\Delta t^2}{\Delta x} \sum_{j=0}^{N} \left(\mathrm{D}_x^+ U_j^n \right)^2.$$

Hence, combining the simplified forms of T_1, T_2, T_3, we obtain

$$\mathcal{E}^{n+1} \leq \mathcal{E}^n + \left(\frac{2\Delta t^2}{\Delta x} - \Delta x \Delta t \right) \sum_{j=0}^{N} \left(\mathrm{D}_x^+ U_j^n \right)^2. \tag{10.22}$$

Now, observe that under the assumption that

$$\begin{aligned} &\frac{2\Delta t^2}{\Delta x} - \Delta x \Delta t \leq 0 \\ \iff &\frac{\Delta t}{\Delta x^2} \leq \frac{1}{2} \iff \lambda \leq \frac{1}{2}, \end{aligned} \tag{10.23}$$

it holds that

$$\mathcal{E}^{n+1} \leq \mathcal{E}^n \leq \ldots \leq \mathcal{E}^0.$$

In other words, under the assumption that $\lambda \leq \frac{1}{2}$, the discrete energy of the approximate solution does not increase over time. On the other hand, if $\lambda > \frac{1}{2}$, then energy is being added to the approximate solution at each time step and therefore the solution must be unstable.

The condition (10.23) is known as the *Courant-Friedrichs-Levy (CFL) condition*. Clearly, the explicit finite difference scheme (10.13) is stable only if the CFL condition is satisfied. The scheme (10.13) is therefore termed a *conditionally stable* scheme.

Note that the CFL condition (10.23) is very restrictive. Indeed, the time step Δt needs to satisfy the relation $\Delta t \leq \frac{1}{2}\Delta x^2$, and thus for reasonably small grid spacing Δx, we need to take *very* small time steps Δt.

10.4.3 Discrete Maximum Principle

It is also possible to obtain the CFL condition by imposing the constraint that approximate solutions produced by the finite difference scheme (10.13) must satisfy a discrete maximum principle.

To this end, recall that the finite difference scheme (10.13) can be written in the form

$$U_j^{n+1} = (1 - 2\lambda)U_j^n + \lambda U_{j+1}^n + \lambda U_{j-1}^n. \tag{10.24}$$

Next, assume that the CFL condition is satisfied, i.e.,

$$\lambda = \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}.$$

Let $\bar{U}_j^n = \max\left(U_{j-1}^n, U_j^n, U_{j+1}^n\right)$ and observe that by definition, it holds that

$$U_{j-1}^n \leq \bar{U}_j^n, \quad U_j^n \leq \bar{U}_j^n, \quad U_{j+1}^n \leq \bar{U}_j^n.$$

Hence, using the fact that $\lambda > 0$ and $1 - 2\lambda \geq 0$, Equation (10.24) implies that

$$\begin{aligned} U_j^{n+1} &\leq \lambda \bar{U}_j^n + (1-2\lambda)\bar{U}_j^n + \lambda \bar{U}_j^n \\ &= \bar{U}_j^n, \end{aligned}$$

or equivalently,

$$U_j^{n+1} \leq \max\left(U_{j-1}^n, U_j^n, U_{j+1}^n\right).$$

Similarly, let $\underline{U}_j^n = \min\left(U_{j-1}^n, U_j^n, U_{j+1}^n\right)$. Then, Equation (10.24) implies that

$$U_j^{n+1} \geq \underline{U}_j^n = \min\left(U_{j-1}^n, U_j^n, U_{j+1}^n\right).$$

Thus, under the CFL condition, the approximate solution produced by the finite difference scheme (10.13) satisfies the following discrete maximum principle:

$$\min\left(U_{j-1}^n, U_j^n, U_{j+1}^n\right) \leq U_j^{n+1} \leq \max\left(U_{j-1}^n, U_j^n, U_{j+1}^n\right),$$

for all $n = 0, \ldots, M$ and for all $j = 1, \ldots, N$.

In addition, iterating over all indices n and j, we obtain that the approximate solution U_j^{n+1} satisfies the inequality

$$\min\left(0, \min_j U_j^0\right) \leq U_j^{n+1} \leq \max\left(0, \max_j U_j^0\right), \tag{10.25}$$

for all $n = 0, \ldots, M$ and for all $j = 1, \ldots, N$.

Inequality (10.25) is a discrete version of the maximum principle. Approximate solutions produced by the finite difference scheme (10.13) will satisfy (10.25) if the CFL condition (10.23) holds.

10.4.4 Truncation Error

Let $u_j^n = u(x_j, t^n)$, where u is the exact solution of the heat equation (10.12). Then the truncation error of the finite difference scheme (10.13) is defined as

$$\tau_j^n = \frac{u_j^{n+1} - u_j^n}{\Delta t} - \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{\Delta x^2},$$

or equivalently,

$$\tau_j^n = \mathrm{D}_t^+ u_j^n - \mathrm{D}_x^- \mathrm{D}_x^+ u_j^n.$$

It can be shown that there exists some constant $C \in \mathbb{R}$ such that for all $n = 0, \ldots, M$ and for all $j = 1, \ldots, N$, the truncation error τ_j^n satisfies the bound

$$|\tau_j^n| \leq C\Big(\Delta t + \Delta x^2\Big).$$

Furthermore, we can use the above bound on the truncation error along with the results on energy stability to prove the following convergence result: there exists some $\bar{C} \in \mathbb{R}$ such that for all $\Delta t, \Delta x > 0$ that satisfy the CFL condition and for all $n = 0, \ldots, M + 1$ it holds that

$$\sqrt{\frac{\Delta x}{2} \sum_{j=1}^{N} \left|u_j^n - U_j^n\right|^2} \leq \bar{C}\Big(\Delta t + \Delta x^2\Big).$$

Hence, the explicit finite difference scheme (10.13) has a first-order rate of convergence in time and a second-order rate of convergence in space.

10.5 An Implicit Finite Difference Scheme

Unfortunately, the CFL condition (10.23) is extremely constraining as the relation $\Delta t \approx \Delta x^2$ results in very small time steps. To remedy this, we can instead use an implicit finite difference scheme:

$$\mathrm{D}_t^- U_j^{n+1} = \mathrm{D}_x^- \mathrm{D}_x^+ U_j^{n+1}, \tag{10.26}$$

for all $j = 1, \ldots, N$ and for all $n = 0, \ldots M$, where we have used backward differences to approximate the time derivative.

Equation (10.26) can be rewritten as

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} = \frac{U_{j-1}^{n+1} - 2U_j^{n+1} + U_{j+1}^{n+1}}{\Delta x^2},$$

or equivalently,

$$-\lambda U_{j-1}^{n+1} + (1 + 2\lambda) U_j^{n+1} - \lambda U_{j+1}^{n+1} = U_j^n,$$

for all $j = 1, \ldots, N$ and for all $n = 0, \ldots M$.

Note that at any given time level t^n, the implicit finite difference scheme (10.26) reduces to a matrix equation:

$$AU^{n+1} = F^n, \tag{10.27}$$

where $U^{n+1} = \left\{U_j^{n+1}\right\}_{j=1}^N$ is the vector of unknowns, $F^n = \left\{U_j^n\right\}_{j=1}^N$ is the right-hand side and $A = \{A_{i,j}\}_{i,j=1}^N$ is the tridiagonal $N \times N$ matrix given by

$$A = \begin{bmatrix} 1+2\lambda & -\lambda & 0 & \dots & 0 \\ -\lambda & 1+2\lambda & -\lambda & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -\lambda & 1+2\lambda & -\lambda \\ 0 & \dots & 0 & -\lambda & 1+2\lambda \end{bmatrix}.$$

Remark 10.4 (Terminology)

The finite difference scheme (10.26) *is termed an Implicit finite difference scheme since the Implicit (Backward) Euler method is used for time stepping.*

10.5.1 Discrete Energy Stability

Similar to the case of the explicit finite difference scheme (10.13), we perform an energy stability analysis of the approximate solutions produced by the implicit finite difference scheme (10.26).

To this end, we multiply both sides of Equation (10.26) with $U_j^{n+1}\Delta t\Delta x$ and sum over all $j = 1, \dots, N$ to obtain

$$\Delta x\Delta t \sum_{j=1}^N U_j^{n+1} \mathrm{D}_t^- U_j^{n+1} = \Delta x\Delta t \sum_{j=1}^N U_j^{n+1} \mathrm{D}_x^- \mathrm{D}_x^+ U_j^{n+1}$$

Next, we apply a variant of the discrete chain rule (10.19):

$$U_j^{n+1} \mathrm{D}_t^+ U_j^{n+1} = \frac{1}{2} \mathrm{D}_t^+ \left(\left(U_j^{n+1}\right)^2 \right) - \frac{\Delta t}{2} \left(\mathrm{D}_t^+ U_j^{n+1} \right)^2,$$

and the summation by parts formula (10.20) together with the boundary conditions to obtain

$$\begin{aligned} \frac{\Delta x\Delta t}{2} \sum_{j=1}^N \mathrm{D}_t^- \left(\left(U_j^{n+1}\right)^2 \right) = &- \Delta x\Delta t \sum_{j=0}^N \left(\mathrm{D}_x^+ U_j^{n+1} \right)^2 \\ &- \frac{\Delta x\Delta t^2}{2} \sum_{j=1}^N \left(\mathrm{D}_t^- U_j^{n+1} \right)^2. \end{aligned}$$

Thus,

$$\mathcal{E}^{n+1} = \mathcal{E}^n - \Delta x \Delta t \sum_{j=0}^{N} \left(\mathrm{D}_x^+ U_j^{n+1} \right)^2 - \frac{\Delta x \Delta t^2}{2} \sum_{j=1}^{N} \left(\mathrm{D}_t^- U_j^{n+1} \right)^2.$$

Therefore, for all $n = 0, \ldots, M$ it holds that

$$\mathcal{E}^{n+1} \leq \mathcal{E}^n.$$

Hence, irrespective of the size of the time step Δt and the grid spacing Δx, the discrete energy of the approximate solution produced by the implicit finite difference scheme (10.26) does not increase over time. Thus, the implicit finite difference scheme (10.26) is termed *unconditionally stable*.

10.5.2 Discrete Maximum Principle

The unconditional stability of the implicit finite difference scheme (10.26) can also be deduced using the discrete maximum principle.

First, observe that the implicit finite difference scheme (10.26) can be rewritten in the form

$$(1 + 2\lambda) U_j^{n+1} = U_j^n + \lambda U_{j-1}^{n+1} + \lambda U_{j+1}^{n+1}. \tag{10.28}$$

Next, let $\bar{U}^{n+1} = \max_{0 \leq j \leq N+1} U_j^{n+1}$. Then, since $\lambda > 0$, Equation (10.28) implies that for all $j = 0, \ldots, N+1$ it holds that

$$(1 + 2\lambda) U_j^{n+1} \leq \bar{U}^n + 2\lambda \bar{U}^{n+1}.$$

Now, using the fact that the right side of the above inequality is independent of j, we obtain that for all $n = 0, \ldots, M$

$$\begin{aligned} \max_{0 \leq j \leq N+1} U_j^{n+1} &\leq \bar{U}^n + 2\lambda \bar{U}^{n+1} \\ \implies (1 + 2\lambda) \bar{U}^{n+1} &\leq \bar{U}^n + 2\lambda \bar{U}^{n+1} \\ \implies \bar{U}^{n+1} &\leq \bar{U}^n. \end{aligned}$$

We can similarly prove a minimum principle for the approximate solution produced by the implicit finite difference scheme (10.26). We therefore conclude that

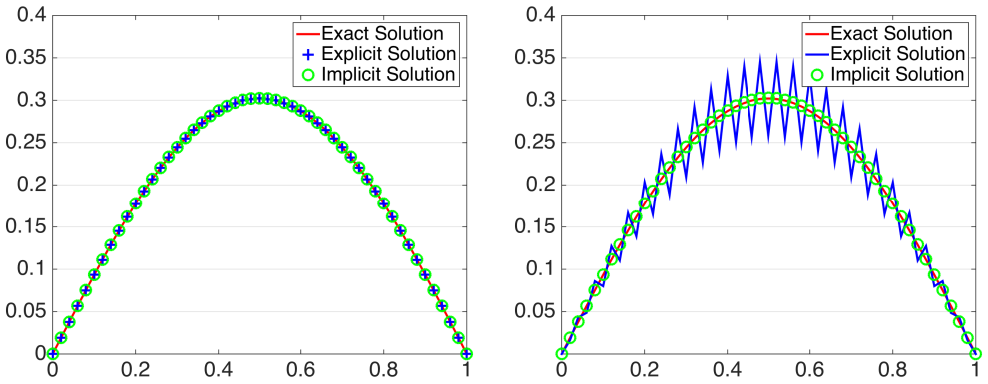
$$\min\left(0, \min_j U_j^0\right) \leq U_j^{n+1} \leq \max\left(0, \max_j U_j^0\right), \tag{10.29}$$

for all $n = 0, \ldots, M$.

Inequality (10.25) is once again a discrete version of the maximum principle. We have thus shown that approximate solutions produced by the implicit finite difference scheme (10.26) will satisfy (10.25) irrespective of the values of Δx and Δt and the CFL condition (10.23). Therefore, we once again conclude that the implicit finite difference scheme (10.26) is unconditionally stable.

10.5.3 Numerical Results

We consider the boundary value problem (10.16) and use the implicit finite difference method (10.26) to approximate solutions to this problem for different values of the time step Δt and the parameter λ. We recall that the explicit finite difference scheme (10.13) is only conditionally stable and therefore produces unstable solutions if the parameter $\lambda > \frac{1}{2}$.



(a) $\Delta x = \frac{1}{50}$, $\Delta t = \frac{1}{5000} \implies \lambda = 0.5$. (b) $\Delta x = \frac{1}{50}$, $\Delta t = \frac{1}{4975} \implies \lambda = 0.503$.

Figure 10.8: Exact solution and approximate solution plots at time $t = 0.1$ for the BVP (10.16) using both the explicit and implicit finite difference method for two different values of Δt and λ.

Figure 10.8 displays the exact and approximate solution at time $t = 0.1$ for the parameter values $\lambda = \frac{1}{2}$ and $\lambda = 0.55$. The figure indicates that while the explicit finite difference scheme (10.13) is unstable for $\lambda = 0.55$, the implicit finite difference scheme (10.26) produces a stable solution for both values of λ.

Furthermore, Figure 10.9 supports our conclusion that the implicit numerical scheme (10.26) produces stable solutions irrespective of the value of λ.

We note however that the accuracy of the approximate solutions decreases for larger values of the parameter λ.

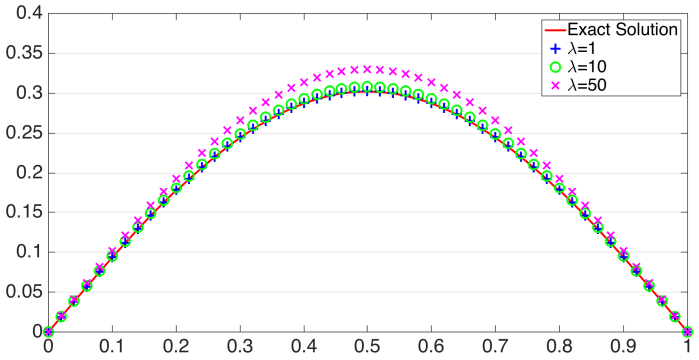


Figure 10.9: Approximate solution plots of the BVP (10.16) for different values of the parameter λ.

Remark 10.5 (Convergence Results) *It can be shown that, similar to the explicit finite difference scheme* (10.13)*, the implicit finite difference scheme* (10.26) *has a first-order rate of convergence in time and a second-order rate of convergence in space.*

10.6 Crank-Nicolson Scheme

Both the explicit and implicit finite difference schemes discussed so far are only first-order accurate in time. An example of a higher-order time accurate scheme is the *Crank-Nicolson* method:

$$\mathrm{D}_t^+ U_j^n = \frac{1}{2}\mathrm{D}_x^- \mathrm{D}_x^+ U_j^n + \frac{1}{2}\mathrm{D}_x^- \mathrm{D}_x^+ U_j^{n+1}, \tag{10.30}$$

for all $j = 1, \ldots, N$ and for all $n = 0, \ldots M$.

Equation (10.30) can be rewritten as

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} = \frac{U_{j-1}^n - 2U_j^n + U_{j+1}^n}{2\Delta x^2} + \frac{U_{j-1}^{n+1} - 2U_j^{n+1} + U_{j+1}^{n+1}}{2\Delta x^2},$$

for all $j = 1, \ldots, N$ and for all $n = 0, \ldots M$.

Note that the right side of Equation (10.30) is simply an average of the spatial derivatives at the t^n and t^{n+1} time levels. Hence, the Crank-Nicolson scheme (10.30) is the formal average of the explicit finite difference scheme (10.14) and the implicit finite difference scheme (10.26).

Also note that we can impose the boundary conditions of the heat equation (10.12) by setting for all $n = 1, \ldots, M+1$

$$U_0^n = U_{N+1}^n = 0,$$

and we can impose the initial conditions of the heat equation (10.12) by setting for all $j = 0, \ldots, N+1$

$$U_j^0 = u_j^0 = u_0(x_j).$$

Note that using $\lambda = \frac{\Delta t}{\Delta x^2}$, we can rewrite the scheme (10.30) as

$$-\frac{\lambda}{2}U_{j-1}^{n+1} + (1+\lambda)U_j^{n+1} - \frac{\lambda}{2}U_{j+1}^{n+1} = \frac{\lambda}{2}U_{j-1}^n + (1-\lambda)U_j^n + \frac{\lambda}{2}U_{j+1}^n,$$

for all $j = 1, \ldots, N$ and for all $n = 0, \ldots M$.

Then, similar to the implicit finite difference scheme (10.26), it is possible to rewrite the Crank Nicolson scheme (10.30) at any time level t^n as a matrix equation:

$$AU^{n+1} = F^n, \tag{10.31}$$

where $U^{n+1} = \left\{U_j^{n+1}\right\}_{j=1}^N$ is the vector of unknowns, $F^n = \left\{F_j^n\right\}_{j=1}^N$ is the right-hand side given by

$$F_j^n = \frac{\lambda}{2}U_{j-1}^n + (1-\lambda)U_j^n + \frac{\lambda}{2}U_{j+1}^n,$$

and $A = \{A_{i,j}\}_{i,j=1}^N$ is the tridiagonal $N \times N$ matrix given by

$$A = \begin{bmatrix} 1+\lambda & -\frac{\lambda}{2} & 0 & \ldots & 0 \\ -\frac{\lambda}{2} & 1+\lambda & -\frac{\lambda}{2} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -\frac{\lambda}{2} & 1+\lambda & -\frac{\lambda}{2} \\ 0 & \ldots & 0 & -\frac{\lambda}{2} & 1+\lambda \end{bmatrix}.$$

10.6.1 Discrete Energy Stability

In order to explore the stability of approximate solutions produced by the Crank-Nicolson scheme (10.30), we perform an energy stability analysis.

To this end, we multiply both sides of Equation (10.30) with $\Delta x \Delta t \left(\frac{v_j^{n+1} + v_j^n}{2} \right)$ and sum over all $j = 1, \ldots, N$, to obtain

$$\begin{aligned} \frac{\Delta x}{2} \sum_{j=1}^{N} \left(U_j^{n+1} + U_j^n \right) \left(U_j^{n+1} - U_j^n \right) = \; & \underbrace{\frac{\Delta x \Delta t}{4} \sum_{j=1}^{N} U_j^{n+1} \mathrm{D}_x^- \mathrm{D}_x^+ U_j^{n+1}}_{\bar{T}_1} \\ & + \underbrace{\frac{\Delta x \Delta t}{4} \sum_{j=1}^{N} U_j^{n} \mathrm{D}_x^- \mathrm{D}_x^+ U_j^{n+1}}_{\bar{T}_2} \\ & + \underbrace{\frac{\Delta x \Delta t}{4} \sum_{j=1}^{N} U_j^{n+1} \mathrm{D}_x^- \mathrm{D}_x^+ U_j^{n}}_{\bar{T}_3} \\ & + \underbrace{\frac{\Delta x \Delta t}{4} \sum_{j=1}^{N} U_j^{n} \mathrm{D}_x^- \mathrm{D}_x^+ U_j^{n}}_{\bar{T}_4} . \end{aligned}$$

We can now modify the discrete summation by parts formula (10.20) and use the boundary conditions in order to simplify each of the terms $\bar{T}_1, \bar{T}_2, \bar{T}_3, \bar{T}_4$ individually. We then obtain

$$\begin{aligned} \bar{T}_1 &= -\frac{\Delta x \Delta t}{4} \sum_{j=0}^{N} \left(\mathrm{D}_x^+ U_j^{n+1} \right)^2, \\ \bar{T}_2 = \bar{T}_3 &= -\frac{\Delta x \Delta t}{4} \sum_{j=0}^{N} \left(\mathrm{D}_x^+ U_j^{n} \right) \left(\mathrm{D}_x^+ U_j^{n+1} \right), \\ \bar{T}_4 &= -\frac{\Delta x \Delta t}{4} \sum_{j=0}^{N} \left(\mathrm{D}_x^+ U_j^{n} \right)^2. \end{aligned}$$

Since

$$\frac{\Delta x}{2} \sum_{j=1}^{N} \left(U_j^{n+1} + U_j^n \right) \left(U_j^{n+1} - U_j^n \right) = \; \mathcal{E}^{n+1} - \mathcal{E}^n,$$

it therefore follows that

$$\begin{aligned}\mathcal{E}^{n+1} - \mathcal{E}^n = &- \frac{\Delta x \Delta t}{4} \sum_{j=1}^{N} \left(\mathrm{D}_x^+ U_j^{n+1}\right)^2 \\ &- \frac{\Delta x \Delta t}{2} \sum_{j=1}^{N} \left(\mathrm{D}_x^+ U_j^{n}\right)\left(\mathrm{D}_x^+ U_j^{n+1}\right) \\ &- \frac{\Delta x \Delta t}{4} \sum_{j=1}^{N} \left(\mathrm{D}_x^+ U_j^{n}\right)^2.\end{aligned}$$

Hence, for all $n = 0, \ldots, M$ it holds that

$$\begin{aligned}\mathcal{E}^{n+1} &= \mathcal{E}^n - \frac{\Delta x \Delta t}{4} \sum_{j=1}^{N} \left(\mathrm{D}_x^+ v_j^{n+1} + \mathrm{D}_x^+ v_j^{n}\right)^2 \\ \implies \mathcal{E}^{n+1} &\leq \mathcal{E}^n.\end{aligned}$$

Therefore, irrespective of the size of the time step Δt and the grid spacing Δx, the discrete energy of the approximate solution produced by the Crank-Nicolson scheme (10.30) does not increase over time. Thus, the Crank-Nicolson scheme is also termed *unconditionally stable*.

Remark 10.6 (Discrete Maximum Principle) *Interestingly, despite the unconditional stability of the Crank-Nicolson scheme* (10.30)*, it can nevertheless be shown that the approximate solutions produced by the scheme will satisfy a discrete maximum principle only if* $\lambda = \frac{\Delta t}{\Delta x^2} \leq 1$*. Indeed, for large values of* λ*, approximate solutions may contain spurious oscillations. Therefore, in contrast to the implicit finite difference scheme* (10.26)*, the Crank-Nicolson scheme in general, will not satisfy a discrete maximum principle if there is no constraint on* λ*.*

10.6.2 Truncation Error

Let $u_j^n = u(x_j, t^n)$, where u is the exact solution of the heat equation (10.12). Then the truncation error of the Crank-Nicolson scheme (10.30) is defined as

$$\tau_j^n = \frac{u_j^{n+1} - u_j^n}{\Delta t} - \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{2\Delta x^2} - \frac{u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}}{2\Delta x^2},$$

or equivalently,

$$\tau_j^n = \mathrm{D}_t^+ u_j^n - \left(\frac{1}{2}\mathrm{D}_x^- \mathrm{D}_x^+ u_j^n + \frac{1}{2}\mathrm{D}_x^- \mathrm{D}_x^+ u_j^{n+1}\right).$$

It can be shown using Taylor expansions that there exists some constant $C \in \mathbb{R}$ such that for all $n = 0, \ldots, M$ and for all $j = 1, \ldots, N$, the truncation error τ_j^n satisfies the bound

$$|\tau_j^n| \leq C\Big(\Delta t^2 + \Delta x^2\Big).$$

Furthermore, we can use the above bound on the truncation error along with the results on energy stability to prove that the Crank-Nicolson scheme (10.30) has a second-order rate of convergence in both time and space.

10.7 Convergence Studies

Convergence Study 1 We consider the boundary value problem (10.15) given by

$$\begin{aligned} u_t - u_{xx} &= 0, && \text{on } (0,1) \times (0,T), \\ u(x,0) &= \sin(2\pi x), && \text{on } (0,1), \\ u(0,t) &= u(1,t) = 0, && \text{on } (0,T), \end{aligned}$$

Our goal is to use the different finite difference schemes (10.13), (10.26) and (10.30) to approximate solutions to this problem for different values of the time step Δt and grid size Δx and compare the experimental order of convergence. For the purpose of this experiment, we choose a fixed $\lambda = \frac{1}{2}$ so that $\Delta t = \frac{1}{2}\Delta x^2$. All errors were calculated using the max-norm and a final time $T = 0.1$.

Δx	Δt	Explicit	EOC	Implicit	EOC	Crank-Nicolson	EOC
$\frac{1}{10}$	$\frac{1}{2\cdot 10^2}$	4.63×10^{-3}	1.90	1.05×10^{-2}	2.03	2.27×10^{-3}	1.87
$\frac{1}{20}$	$\frac{1}{2\cdot 20^2}$	1.25×10^{-3}	1.99	2.57×10^{-3}	2.03	6.19×10^{-4}	1.99
$\frac{1}{40}$	$\frac{1}{2\cdot 40^2}$	3.13×10^{-4}	2.00	6.31×10^{-4}	2.00	1.56×10^{-4}	2.00
$\frac{1}{80}$	$\frac{1}{2\cdot 80^2}$	7.83×10^{-5}	2.00	1.57×10^{-4}	2.00	3.91×10^{-5}	2.00
$\frac{1}{160}$	$\frac{1}{2\cdot 160^2}$	1.96×10^{-5}	2.00	3.92×10^{-5}	2.00	9.79×10^{-6}	2.00
$\frac{1}{320}$	$\frac{1}{2\cdot 320^2}$	4.89×10^{-6}		9.79×10^{-6}		2.45×10^{-6}	

Table 10.1: Error Table of the two-dimensional Finite difference methods for the BVP (10.15).

Table 10.1 displays our results and indicates that all scheme have an experimental order of convergence EOC ≈ 2 with respect to Δx. We recall that

each finite difference scheme (10.13), (10.26) and (10.30) is second-order accurate in space. Since, we have chosen $\Delta t = \frac{1}{2}\Delta x^2$, the temporal error is at most of the same order as the spatial error and our results therefore agree with the theoretical convergence analysis.

Convergence Study 2 We once again consider the boundary value problem (10.15) and use the finite difference schemes (10.26) and (10.30) to approximate solutions to this problem. For the purpose of this experiment, we set $\Delta x = \Delta t$ and vary the parameter $\lambda = \frac{1}{\Delta x}$. All errors were calculated using the max-norm and a final time $T = 0.1$.

Table 10.2 displays our results and indicates that the the implicit finite difference scheme (10.26) has an experimental order of convergence EOC ≈ 1, while the Crank-Nicolson scheme (10.30) has an EOC ≈ 2. We recall that the scheme (10.26) is only first-order accurate in time while the Crank-Nicolson scheme (10.30) is second-order accurate in time. Since, we have chosen $\Delta t = \Delta x$, the spatial error is at most of the same order as the temporal error and our results therefore agree with the theoretical convergence analysis.

Δx	Δt	Implicit	EOC	Crank-Nicolson	EOC
$\frac{1}{20}$	$\frac{1}{20}$	9.50×10^{-2}	1.07	1.92×10^{-2}	1.70
$\frac{1}{40}$	$\frac{1}{40}$	4.51×10^{-2}	1.09	5.92×10^{-3}	1.99
$\frac{1}{80}$	$\frac{1}{80}$	2.12×10^{-2}	1.07	1.50×10^{-3}	2.00
$\frac{1}{160}$	$\frac{1}{160}$	1.01×10^{-2}	1.04	3.76×10^{-4}	2.00
$\frac{1}{320}$	$\frac{1}{320}$	4.88×10^{-3}	1.02	9.42×10^{-5}	2.00
$\frac{1}{640}$	$\frac{1}{640}$	2.40×10^{-3}		2.35×10^{-5}	

Table 10.2: Error Table of the two-dimensional Finite difference methods for the BVP (10.15).

11 Linear Transport Equations (Hyperbolic PDEs)

In this chapter, we consider the simplest example of a hyperbolic partial differential equation: the linear transport equation in one spatial dimension, which is given by

$$u_t + a(x,t)u_x = 0 \qquad \forall\ (x,t) \in \mathbb{R} \times \mathbb{R}_+. \tag{11.1}$$

One situation where this equation arises is when modelling the motion of a pollutant in a river. Let u denote the concentration of the pollutant in the river. Assume that the river flows with a velocity field $a(x,t)$ and we know the velocity field at all points in the river. The pollutant will clearly be transported with the velocity of the river at the current position and this then leads to Equation (11.1).

The simplest case of the scalar transport equation actually arises when the velocity field is constant, that is, $a(x,t) \equiv a$. The resulting transport equation is

$$u_t + au_x = 0. \tag{11.2}$$

The rather simple equation (11.2) has served as a crucible for designing highly efficient schemes for much more complicated systems of equations. We concentrate on it for the rest of this chapter.

11.1 Method of characteristics

The initial value problem (or Cauchy problem) for (11.1) consists of finding a solution of (11.1) that also satisfies the initial condition

$$u(x,0) = u_0(x) \qquad \forall\ x \in \mathbb{R}. \tag{11.3}$$

It is well known that the solution of the initial value problem can be constructed by using the *method of characteristics*. The idea underlying this

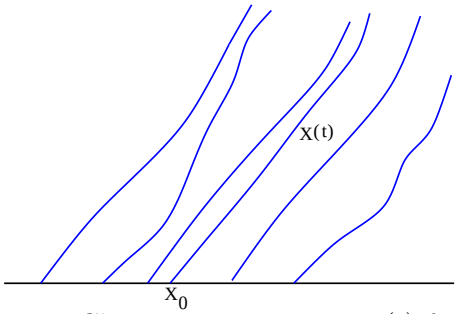


Figure 11.1: Characteristics curves $x(t)$ for (11.1)

method is to reduce a PDE like (11.1) to an ODE by utilizing the structure of the solutions. As an ansatz, assume that we are given some curve $x(t)$, along which the solution u is constant. This means that

$$\begin{aligned} 0 &= \frac{d}{dt}u(x(t),t) && \text{(as u is constant along } x(t)) \\ &= u_t(x(t),t) + u_x(x(t),t)x'(t) && \text{(chain rule).} \end{aligned}$$

We also know that $u_t(x(t),t) + u_x(x(t),t)a(x(t),t) = 0$, since u is assumed to be a solution of (11.1). Therefore, if $x(t)$ satisfies the ODE

$$\begin{aligned} x'(t) &= a(x(t),t) \\ x(0) &= x_0, \end{aligned} \tag{11.4}$$

then $x(t)$ is precisely such a curve. The solution $x(t)$ of this equation is called a *characteristic curve*. From ODE theory, we know that solutions of (11.4) exist provided that a is Lipschitz continuous in both arguments. It may or may not be possible to find an explicit solution formula for (11.4).

The importance of characteristic curves lies in the property that u is constant along them:

$$u(x(t),t) = u(x(0),0) = u_0(x_0).$$

The initial data $u_0(x)$ is already known, so if we can find characteristic curves that go through all points $(x,t) \in \mathbb{R} \times \mathbb{R}_+$, then we have found the solution u at all points in the plane. (See Figure 11.1) for an illustration.)

In the simple case of a constant velocity field $a(x,t) \equiv a$, the characteristic equation (11.4) is explicitly solved as

$$x(t) = x_0 + at.$$

Therefore, given some point (x,t), the unique characteristic that goes through (x,t) (so that $x(t)=x$) has initial value $x_0 = x - at$. Hence, the solution of (11.2) is

$$u(x,t) = u_0(x_0) = u_0(x-at) \tag{11.5}$$

for any $(x,t) \in \mathbb{R} \times R_+$. The solution formula (11.5) implies that the initial data is transported with the velocity a.

In the more general case of (11.1), the characteristic equation (11.4) may not be possible to solve explicitly. Hence, it is essential that we obtain some information about the structure of solutions of (11.1) from the equation itself. This is done by means of an energy estimate. In order to prove this estimate however, we require the important Gronwall lemma:

Theorem 11.1 (Gronwall's inequality) *Let $\beta(t)$ be continuous and $u(t)$ be differentiable on some interval $[a,b]$, and assume that*

$$u'(t) \leq \beta(t)u(t) \qquad \forall\ t \in (a,b).$$

Then

$$u(t) \leq u(a)\exp\left(\int_a^t \beta(t)\right) \qquad \forall\ t \in [a,b].$$

Solutions to the transport equation (11.1) then satisfy the following energy estimate:

Lemma 11.2 *Let $u(x,t)$ be a smooth solution of (11.1) which decays to zero at infinity, i.e, $\lim_{|x|\to\infty} u(x,t) = 0$ for all $t \in \mathbb{R}_+$ and let a be a continuously differentiable function. Then u satisfies the energy bound*

$$\int_{\mathbb{R}} u^2(x,t)dx \leq e^{\|a\|_{C^1}t}\int_{\mathbb{R}} u_0^2(x)dx \tag{11.6}$$

for all times $t > 0$.

Proof The proof of the estimate (11.6) is based on multiplying (11.1) with u on both sides:

$$
\begin{aligned}
u u_t + a(x,t)\, u u_x &= 0 && \text{(multiplying (11.1) by } U\text{)}\\
\left(\frac{u^2}{2}\right)_t + a(x,t)\left(\frac{u^2}{2}\right)_x &= 0 && \text{(chain rule)}\\
\left(\frac{u^2}{2}\right)_t + \left(a(x,t)\frac{u^2}{2}\right)_x &= a_x(x,t)\frac{u^2}{2} && \text{(product rule)}\\
\frac{d}{dt}\int_{\mathbb{R}}\left(\frac{u^2}{2}\right)dx + \int_{\mathbb{R}}\left(a(x,t)\frac{u^2}{2}\right)_x dx &= \int_{\mathbb{R}} a_x(x,t)\frac{u^2}{2}dx && \text{(integrating over space)}\\
\frac{d}{dt}\int_{\mathbb{R}}\left(\frac{u^2}{2}\right)dx + \left(a(x,t)\frac{u^2}{2}\right)\Big|_{x=-\infty}^{x=\infty} &= \int_{\mathbb{R}} a_x(x,t)\frac{u^2}{2}dx &&\\
\frac{d}{dt}\int_{\mathbb{R}}\left(\frac{u^2}{2}\right)dx &= \int_{\mathbb{R}} a_x(x,t)\frac{u^2}{2}dx && \text{(decay to zero at infinity)}\\
&\le \|a\|_{C^1}\int_{\mathbb{R}}\frac{u^2}{2}dx && \text{(regularity of } a\text{)},
\end{aligned}
$$

where $\|a\|_{C^1} = \sup_{x\in\mathbb{R}} a_x(x,t)$. Finally, we can apply Gronwall's inequality (Theorem 11.1) on the interval $[0,t]$ with $\beta \equiv \|a\|_{C^1}$ to obtain

$$\int_{\mathbb{R}} U^2(x,t)dx \le e^{\|a\|_{C^1} t}\int_{\mathbb{R}} U_0^2(x)dx. \qquad \square$$

The quantity $\int_{\mathbb{R}} \frac{u(x,t)^2}{2}dx$ is commonly called the *energy* of the solution. The above lemma shows that the energy of the solutions to the transport equation (11.1) are bounded. The energy estimate is going to be used for designing robust schemes for the transport equation. We remark that the restriction that the solution u decays to zero at infinity may be relaxed by considering a different energy functional.

11.2 Finite difference schemes for the transport equation

It may not be possible to obtain an explicit formula for the solution of the characteristic equation (11.4). For example, the velocity field $a(x,t)$ might have a complicated nonlinear expression. Hence, we have to devise numerical methods for approximating the solutions of (11.1). For simplicity, we consider $a(x,t) \equiv a > 0$ and solve (11.2). It is rather straightforward to extend the schemes to the case of a more general velocity field.

Discretization of the domain

The first step in any numerical method is to discretize both the spatial and temporal parts of the domain. Since $\mathbb{R}$ is unbounded, we have to truncate the domain to some bounded domain $[x_l, x_r]$. This truncation implies that suitable boundary conditions need to be imposed. We discuss the problem of boundary conditions later on.

For the sake of simplicity, the domain $[x_l, x_r]$ is discretized uniformly with a mesh size Δx into a sequence of $N+1$ points x_j such that $x_0 = x_l$, $x_N = x_r$ and $x_{j+1} - x_j = \Delta x$ for all j. A non-uniform discretization can readily be considered.

For the temporal discretization, we choose some terminal time T and divide $[0, T]$ into M points $t^n = n\Delta t$ $(n = 0, \ldots, M)$. The space-time mesh is shown in Figure 11.2. Our aim is obtain an approximation of the form $U_j^n \approx u(x_j, t^n)$. To get from the initial time step t^0 to the terminal time step t^M, we first set the initial data $U_j^0 = u_0(x_0)$ for all j. Then the solution U_j^1 at the next time step is computed using some update formula, again for all j. This process is reiterated until we arrive at the final time step $t^M = T$ with our final solution U_j^M.

A simple centered finite difference scheme

On the mesh, we need to approximate the transport equation (11.2). We do so by replacing both the spatial and temporal derivatives by finite differences. Similar to the case of the Heat equation (10.2), the time derivative is replaced with a forward difference and the spatial derivative with a central difference. The resulting scheme is

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + \frac{a(U_{j+1}^n - U_{j-1}^n)}{2\Delta x} = 0 \qquad \text{for } j = 1, \ldots, N-1. \tag{11.7}$$

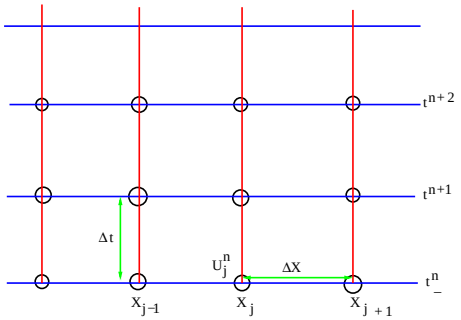


Figure 11.2: A representation of the mesh in space-time

Some special care must be taken when defining the boundary values. We have a consistent discretization of (11.2) that is very simple to implement. We test it on the following numerical example.

A numerical example

Consider the linear transport equation (11.2) in the domain $[0, 1]$ with initial data

$$U_0(x) = \sin(2\pi x). \tag{11.8}$$

Since the data is periodic, it is natural to assume periodic boundary conditions. We implement this numerically by letting

$$U_0^n = U_{N-1}^n, \quad U_N^n = U_1^n.$$

The exact solution is calculated by (11.5) as $u(x, t) = \sin(2\pi(x - at))$. We set $a = 1$ and compute the solutions with the central scheme (11.7) with 100 mesh points, and plot the solution at time $t = 3$ in Figure 11.3. The figure clearly shows that, despite being a consistent approximation, the scheme is unstable, with very large oscillations.

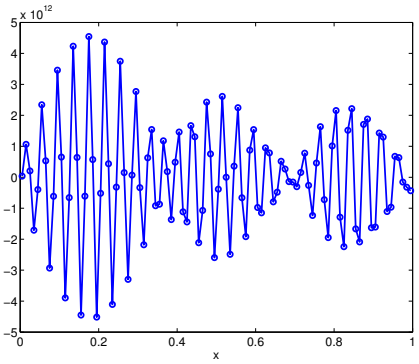


Figure 11.3: Approximate solution for (11.2) with the central scheme (11.7) at time $t = 3$ with 100 mesh points.

A physical explanation

Why do the solutions computed with the central scheme (11.7) blow up? After all, the central scheme seems a reasonable approximation of the transport equation and worked perfectly well for the Heat equation (10.2). A *physical* explanation can be deduced from the following argument: The exact solution moves to the right (as $a > 0$) with a fixed speed. Therefore, information goes from left to right. However, the central scheme (see

Figure 11.4) takes information from both the left and the right, violating the physics. Consequently, the solutions are unstable. This explanation seems intuitive but has to be backed by solid mathematical arguments. We proceed to do so below.

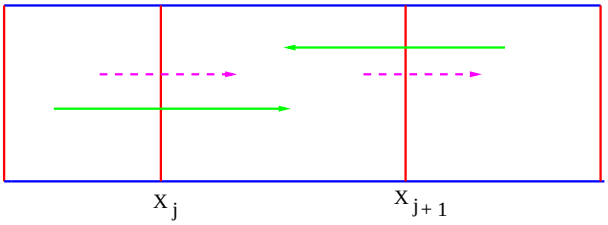


Figure 11.4: The central scheme (11.7). Green arrows indicate numerical propagation and magenta arrows physical propagation.

A mathematical explanation

The observed instability of the central scheme can be explained mathematically in terms of estimates. We recall that the exact solutions have a bounded energy (see estimate (11.6)). It is reasonable to require that the scheme is *energy stable* like the exact solution, that is, a discrete version of energy remains bounded. For a given Δx, we define the discrete version of energy as

$$E^n = \frac{1}{2}\Delta x \sum_{j=1}^{N} \left(U_j^n\right)^2. \tag{11.9}$$

Note that the integral in the energy for the continuous problem has been replaced with a Riemann sum.

Lemma 11.3 *Let U_j^n be the solutions computed with the central scheme* (11.7)*. Then the following estimate holds:*

$$E^{n+1} = E^n + \frac{\Delta x}{2} \sum_{j=1}^{N} \left(U_j^{n+1} - U_j^n\right)^2. \tag{11.10}$$

Consequently, the energy grows at every time step for any choice of $\Delta x, \Delta t$.

Proof We mimic the steps of continuous energy estimate (Lemma 11.2) and multiply both sides of the scheme (11.7) by U_j^n to obtain

$$U_j^n\left(U_j^{n+1} - U_j^n\right) + \frac{a\Delta t}{2\Delta x}\left(U_j^n U_{j+1}^n - U_j^n U_{j-1}^n\right) = 0. \tag{11.11}$$

We have the following elementary identity:

$$d_2(d_1 - d_2) = \frac{(d_1)^2}{2} - \frac{(d_2)^2}{2} - \frac{1}{2}(d_1 - d_2)^2 \tag{11.12}$$

for any two numbers d_1, d_2. Therefore, (11.11) can be rewritten as

$$\frac{\left(U_j^{n+1}\right)^2}{2} - \frac{(U_j^n)^2}{2} - \frac{1}{2}\left(U_j^{n+1} - U_j^n\right)^2 + \frac{a\Delta t}{2\Delta x}\left(U_j^n U_{j+1}^n - U_j^n U_{j-1}^n\right) = 0.$$

Next, we can denote

$$H_{j+1/2} = a\frac{U_j^n U_{j+1}^n}{2}$$

to reduce the above equation to

$$\frac{\left(U_j^{n+1}\right)^2}{2} = \frac{(U_j^n)^2}{2} + \frac{1}{2}\left(U_j^{n+1} - U_j^n\right)^2 - \frac{\Delta t}{\Delta x}(H_{j+1/2} - H_{j-1/2}). \tag{11.13}$$

Summing (11.13) over all j, we obtain

$$\sum_{j=1}^{N} \frac{\left(U_j^{n+1}\right)^2}{2} = \sum_{j=1}^{N} \frac{(U_j^n)^2}{2} + \sum_{j=1}^{N} \frac{1}{2}\left(U_j^{n+1} - U_j^n\right)^2 - \sum_{j=1}^{N} \frac{\Delta t}{\Delta x}(H_{j+1/2} - H_{j-1/2}).$$

Finally, note that the last sum on the right side of the above equation is a telescoping sum and using zero (or periodic) conditions, it reduces to zero. We can therefore multiply both sides by Δx to obtain

$$E^{n+1} = E^n + \frac{\Delta x}{2}\sum_{j=1}^{N}\left(U_j^{n+1} - U_j^n\right)^2. \qquad \square$$

Although we assumed zero or periodic boundary conditions in the proof of this lemma, a variant of the lemma holds for more general boundary conditions, as for the continuous setting in Lemma 11.2.

The above lemma provides a mathematical justification for our physical intuition. The central scheme leads to a growth of energy at every time step and is therefore *unconditionally unstable*. We recall that by contrast, in the case of the heat equation (10.2), the central scheme was *conditionally stable*. We need to find schemes that posses a discrete version of the energy estimate. This use of rigorous mathematical tools like energy analysis to justify physical reasoning will be an essential ingredient of these notes.

11.3 An upwind scheme

The central scheme (11.7) does not respect the direction of propagation of information for the transport equation (11.2). Hence, we must include the correct direction of information propagation and hope that it stabilizes the scheme. This entails using one-sided differences instead of a central difference to approximate the linear transport equation (11.2).

If $a > 0$ and the direction of information propagation is from left to right, then we can use a backward difference in space to obtain the scheme

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + \frac{a(U_j^n - U_{j-1}^n)}{\Delta x} = 0 \qquad \text{for } j = 1, \ldots, N-1, \tag{11.14}$$

and if $a < 0$, we can use the forward difference to obtain:

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + \frac{a(U_{j+1}^n - U_j^n)}{\Delta x} = 0 \qquad \text{for } j = 1, \ldots, N-1. \tag{11.15}$$

Using the notation

$$a^+ = \max\{a, 0\}, \quad a^- = \min\{a, 0\}, \quad |a| = a^+ - a^-,$$

(11.14) and (11.15) can be written together as

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + \frac{a^+(U_j^n - U_{j-1}^n)}{\Delta x} + \frac{a^-(U_{j+1}^n - U_j^n)}{\Delta x} = 0. \tag{11.16}$$

The above scheme takes into account the direction of propagation of information – information is "carried with the wind". Hence, this scheme is termed as the *upwind* scheme.

Using the definition of the absolute value and some simple algebraic manipulations, the upwind scheme (11.16) can be recast as

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + \frac{a(U_{j+1}^n - U_{j-1}^n)}{2\Delta x} = \frac{|a|}{2\Delta x}(U_{j+1}^n - 2U_j^n + U_{j-1}^n) \tag{11.17}$$

(compare to (11.7)). Note that in the above form, the spatial derivatives are the central term and a *diffusion* term. The right hand side of (11.17) approximates $\frac{\Delta x|a|}{2} u_{xx}$. Hence, the upwind scheme (11.17) adds *numerical viscosity* or *diffusion* to the unstable central scheme (11.7). Numerical viscosity plays a crucial role in the design of numerical methods for approximating solutions to more complicated hyperbolic partial differential equations.

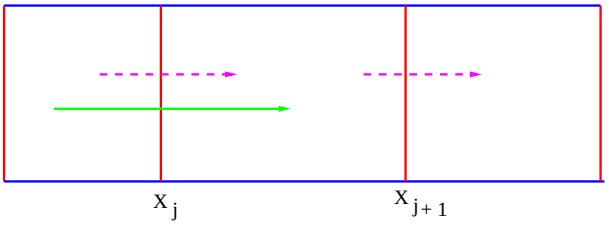
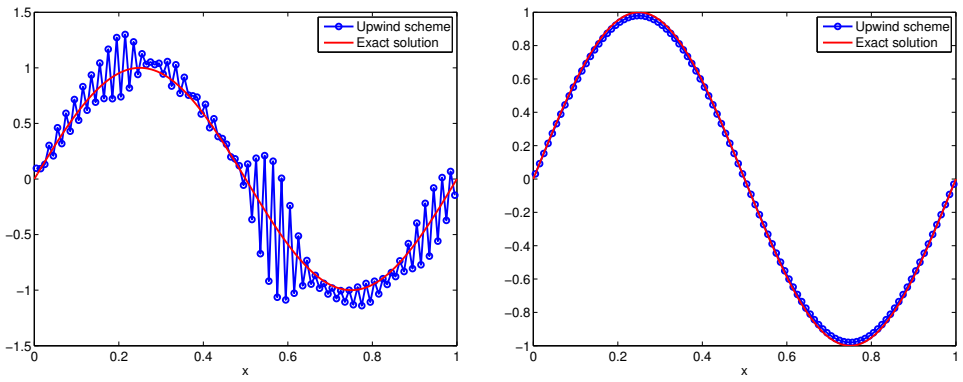


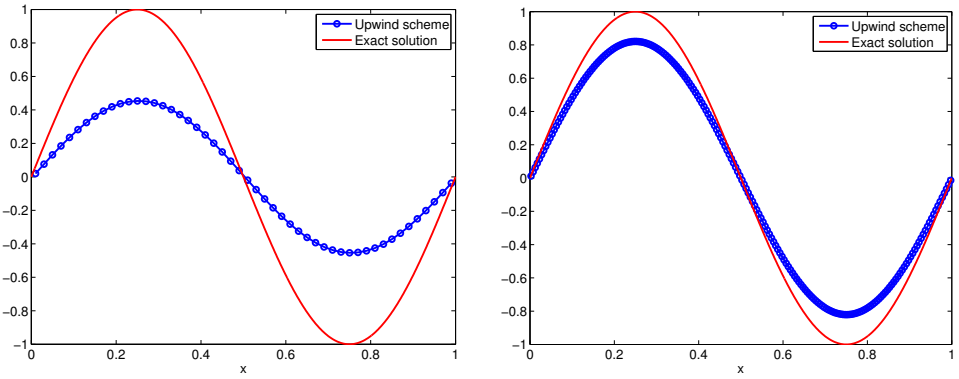
Figure 11.5: The upwind scheme (11.16). Green arrows indicate numerical propagation and magenta arrows physical propagation.

Since the upwind scheme incorporates the correct direction of propagation of information (see Figure 11.5), we expect it to be more stable than the central scheme. This is endorsed by the numerical experiment with initial data (11.8). We take $a = 1$ and compute approximate solutions for the linear transport equation (11.2) on a uniform mesh with 100 mesh points up to $t = 1$. We use two different timesteps: $\Delta t = 1.3\Delta x$ and $\Delta t = 0.9\Delta x$. As seen in Figure 11.6, the results with $\Delta t = 1.3\Delta x$ are still oscillatory and the scheme continues to be unstable. In spite of the upwinding, stability stills seems to be elusive. However, results with $\Delta t = 0.9\Delta x$ are stable. The approximation appears to be good in this case. Much better results are obtained by refining the mesh, while keeping the ratio $\Delta t/\Delta x$ fixed, as is presented in Figure 11.7.



(a) $\Delta t = 1.3\Delta x$ (b) $\Delta t = 0.9\Delta x$

Figure 11.6: Solution with initial data (11.8) at $t = 1$. The ratio $\Delta t/\Delta x$ is important for stability.



(a) 50 mesh points (b) 200 mesh points

Figure 11.7: Solution with initial data (11.8) at $t = 10$. Refining the mesh gives a more accurate solution.

11.4 Stability for the upwind scheme

The numerical results indicate that stability for the upwind scheme is subtle. It is not unconditionally unstable as the central scheme (11.7); instead, stability depends on the parameters $\Delta x, \Delta t$. Numerical results indicate the crucial role played by the ratio $\frac{\Delta t}{\Delta x}$. It seems that one must not only take into account the correct direction of propagation, but also the correct magnitude.

The quantification of stability will involve energy analysis as in the last section. We have the following stability result:

Lemma 11.4 *Let the mesh parameters satisfy the condition*

$$|a|\frac{\Delta t}{\Delta x} \leq 1. \tag{11.18}$$

Then solutions computed with the upwind scheme (11.17) *satisfy the energy estimate*

$$E^{n+1} \leq E^n, \tag{11.19}$$

where the energy is defined as in (11.9)*. The upwind scheme is thus* conditionally stable*.*

Proof For the sake of simplicity, we assume that $a > 0$. Hence the upwind scheme (11.17) reduces to

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + \frac{a(U_{j+1}^n - U_{j-1}^n)}{2\Delta x} = \frac{a}{2\Delta x}(U_{j+1}^n - 2U_j^n + U_{j-1}^n). \tag{11.20}$$

As in the proof of the estimate (11.10) we multiply both sides of the scheme (11.20) by U_j^n to obtain

$$\begin{aligned} U_j^n(U_j^{n+1} - U_j^n) = &-\frac{a\Delta t}{2\Delta x}(U_j^n U_{j+1}^n - U_j^n U_{j-1}^n) \\ &+ \frac{a\Delta t}{2\Delta x}(U_j^n(U_{j+1}^n - U_j^n)) + \frac{a\Delta t}{2\Delta x}(U_j^n(U_{j-1}^n - U_j^n)). \end{aligned} \tag{11.21}$$

Now we use elementary identity (11.12) twice and rewrite (11.21) as

$$\begin{aligned} \frac{(U_j^{n+1})^2}{2} = &\frac{(U_j^n)^2}{2} + \frac{(U_j^{n+1} - U_j^n)^2}{2} - \frac{a\Delta t}{2\Delta x}(U_j^n U_{j+1}^n - U_j^n U_{j-1}^n) \\ &+ \frac{a\Delta t}{4\Delta x}\left((U_{j+1}^n)^2 - (U_j^n)^2\right) - \frac{a\Delta t}{4\Delta x}\left((U_j^n)^2 - (U_{j-1}^n)^2\right) \\ &- \frac{a\Delta t}{4\Delta x}(U_{j+1}^n - U_j^n)^2 - \frac{a\Delta t}{4\Delta x}(U_j^n - U_{j-1}^n)^2. \end{aligned} \tag{11.22}$$

Denoting

$$K_{j+1/2} = \frac{a}{2}(U_j^n U_{j+1}^n) - \frac{a}{4}\left((U_{j+1}^n)^2 - (U_j^n)^2\right),$$

we may rewrite (11.22) as

$$\begin{aligned} \frac{(U_j^{n+1})^2}{2} = &\frac{(U_j^n)^2}{2} + \frac{(U_j^{n+1} - U_j^n)^2}{2} - \frac{a\Delta t}{\Delta x}(K_{j+1/2} - K_{j-1/2}) \\ &- \frac{a\Delta t}{4\Delta x}(U_{j+1}^n - U_j^n)^2 - \frac{a\Delta t}{4\Delta x}(U_j^n - U_{j-1}^n)^2. \end{aligned} \tag{11.23}$$

We may now sum (11.23) over all j to obtain

$$\begin{aligned} \sum_{j=1}^{N} \frac{(U_j^{n+1})^2}{2} = &\sum_{j=1}^{N} \frac{(U_j^n)^2}{2} + \sum_{j=1}^{N} \frac{(U_j^{n+1} - U_j^n)^2}{2} - \sum_{j=1}^{N} \frac{a\Delta t}{\Delta x}(K_{j+1/2} - K_{j-1/2}) \\ &- \sum_{j=1}^{N} \frac{a\Delta t}{4\Delta x}(U_{j+1}^n - U_j^n)^2 - \sum_{j=1}^{N} \frac{a\Delta t}{4\Delta x}(U_j^n - U_{j-1}^n)^2. \end{aligned}$$

Finally, note that the sum involving $K_{j-1/2}$ on the right side of the above equation is a telescoping sum and using zero (or periodic) boundary conditions, it reduces to zero. Furthermore, using a change of indices and zero (or periodic) boundary conditions we obtain that

$$\sum_{j=1}^{N} \frac{a\Delta t}{4\Delta x}(U_{j+1}^n - U_j^n)^2 = \sum_{j=1}^{N} \frac{a\Delta t}{4\Delta x}(U_j^n - U_{j-1}^n)^2$$

We can therefore multiply both sides of the equation with Δx and obtain

$$E^{n+1} \leq E^n + \frac{\Delta x}{2} \sum_j (U_j^{n+1} - U_j^n)^2 - \frac{a\Delta t}{2} \sum_j (U_j^n - U_{j-1}^n)^2. \qquad (11.24)$$

Using the definition of the upwind scheme (11.14) in (11.24) yields

$$E^{n+1} \leq E^n + \left(\frac{a^2 \Delta t^2}{2\Delta x} - \frac{a\Delta t}{2} \right) \sum_j (U_j^n - U_{j-1}^n)^2. \qquad (11.25)$$

Since the term in the sum in (11.25) is positive, we obtain the energy bound (11.19), provided

$$\frac{a^2 \Delta t^2}{\Delta x} \leq a\Delta t,$$

which is precisely the condition (11.18). □

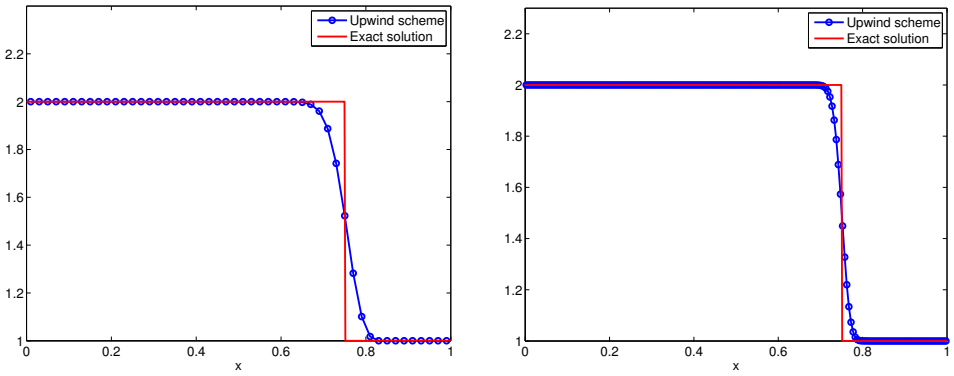
The stability condition (11.18) is termed the *CFL condition* after Courant, Friedrichs and Lewy who first proposed it. Note that the CFL condition (11.18) for the linear transport equation is much less restrictive than the CFL condition (10.23) for the heat equation. Indeed, (11.18) implies that $\Delta x \approx \Delta t$, which means that even in the case of a very fine mesh, the size of the time step Δt is of the order of the size of grid spacing Δx. We remark that the conditional stability of the upwind scheme is confirmed in numerical experiments.

Numerical experiment: Discontinuous data

Consider the transport equation (11.2) with $a = 1$ in the domain $[0, 1]$ and initial data

$$u_0(x) = \begin{cases} 2 & \text{if } x < 0.5 \\ 1 & \text{if } x > 0.5. \end{cases} \qquad (11.26)$$

The initial data and consequently the exact solution (11.5) are discontinuous. We compute with the upwind scheme using 50 and 200 mesh points and display the results in Figure 11.8. The results show that the upwind scheme approximates the solution quite well, at least at a fine resolution. However the errors on a coarse mesh are somewhat large. One can construct more accurate methods but this is far beyond the scope of these notes.



(a) 50 mesh points (b) 200 mesh points

Figure 11.8: The upwind scheme (11.14) for advection (11.2) with discontinuous initial data (11.26). Both results are at time $t = 0.25$.