



ROYAL UNIVERSITY OF PHNOM PENH

FACULTY OF SCIENCE

Bachelor Thesis

1D Simulation to Semilinear Poisson Problem

Author:
SAN RADY

Supervisor:
Dr. SEAM Ngonn

*A thesis submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science in Mathematics*

Academic Year: 2025–2026

Advisor's Endorsement

Program Title: Bachelor of Science in Mathematics

Researcher's Name: SAN RADY

Thesis Title: 1D Simulation to Semilinear Poisson Problem

This is to certify that the thesis referenced above was undertaken and successfully completed by the aforementioned researcher under my direct supervision. I have overseen the entire research process from its initial stage to final submission. I hereby affirm that the work presented fulfills the academic standards required by the program.

Supervisor's Signature: _____

Date: _____

Supervisor's Name: Dr. SEAM Ngonn

Researcher's Declaration

I, San Rady, hereby solemnly declare that the research study entitled:

1D Simulation to Semilinear Poisson Problem

was independently carried out by me as part of the requirements for the Bachelor of Science in Mathematics at the Royal University of Phnom Penh. This research was conducted under the direct supervision and academic guidance of **Dr. SEAM Ngonn.**

I affirm that this thesis has not been submitted, in whole or in part, for the fulfillment of any degree or diploma at any other university or institution. Furthermore, no section of this thesis may be reproduced, quoted, or cited without proper acknowledgment.

Researcher's Signature: _____

Date: _____

Name of Researcher: SAN RADY

Abstract

1D Simulation of Semilinear Poisson Problem

In this thesis, we simulate numerically the solution of the one-dimensional semilinear Poisson problem $-u''(x) = f(u(x))$, for $x \in (0, 1)$ with Dirichlet boundary condition. We use the finite difference method to discretize the problem and apply the Newton-Raphson algorithm to solve the resulting nonlinear system. Numerical results are presented for two choices of f : a linear case $f(u) = x(1 - x)$ and a nonlinear case $f(u) = u(1 - u)$.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisor, **Dr. SEAM Ngonn**, for his guidance, support, and encouragement throughout this thesis. His valuable advice and suggestions have been essential to the completion of this work.

I would also like to thank the **Department of Mathematics, Faculty of Science, Royal University of Phnom Penh** for providing me with the opportunity to pursue this research.

Finally, I would like to express my deepest gratitude to my family and friends for their endless support and encouragement throughout my studies.

Contents

Advisor’s Endorsement	1
Researcher’s Declaration	2
Abstract	3
Acknowledgments	4
1 Motivation	7
1.1 Introduction	7
2 Approximation of Derivatives	9
2.1 Taylor Expansion	9
2.2 Approximation of the First Derivative	10
2.2.1 Forward Difference	10
2.2.2 Backward Difference	10
2.2.3 Central Difference	11
2.3 Approximation of the Second Derivative	12
2.3.1 Three-Point Formula	12
3 Approximation of 1D Semilinear Problem	15
3.1 Continuous Problem	15
3.2 Partition of Segment	16
3.3 Discrete Scheme	16
3.4 Newton–Raphson Algorithm	18
3.4.1 Jacobian Matrix	19
3.4.2 Example	21
4 Simulation	24
4.1 Case 1: $f(u(x)) = x(1 - x)$	24
4.2 Case 2: $f(u(x)) = u(x)(1 - u(x))$	25

5	Appendix Python Code	27
5.1	Case 1: $f(u(x)) = x(1 - x)$	27
5.2	Case 2: $f(u) = u(1 - u)$	28

Chapter 1

Motivation

1.1 Introduction

Differential equations are used to describe many problems in physics and engineering, such as how heat spreads through a material or how an electric field behaves in space. One of the simplest and most studied equations of this type is the Poisson equation in high dimension :

$$-\Delta u = f.$$

The finite difference method for solving such problems has been widely studied [1, 5].

In one dimensional space, this reduces to a second order ordinary differential equation. In this thesis, we study the one-dimensional semilinear Poisson problem:

$$-u''(x) = f(u(x)), \quad x \in (0, 1),$$

with boundary condition.

This work aims to simulate numerically the solution of this problem with the Dirichlet boundary condition. We discretize the problem using the finite difference method to obtain a nonlinear system of equations, and apply the Newton-Raphson algorithm to solve this system by computing the residual function F and the Jacobian matrix J .

This thesis is organized as follows:

- **Chapter 2** introduces the approximation of derivatives using the finite difference method, including the three-point formula for the second derivative.

- **Chapter 3** formulates the one-dimensional semilinear Poisson problem and derives the discrete scheme. It applies the Newton-Raphson method to solve the resulting nonlinear system with the Dirichlet boundary condition, where the residual function F and the Jacobian matrix J are computed explicitly.
- **Chapter 4** presents the numerical simulation for two choices of f : a linear case $f(u) = x(1 - x)$ and a nonlinear case $f(u) = u(1 - u)$.
- **Chapter 5** presents the Python code used to generate the numerical results.

Chapter 2

Approximation of Derivatives

In this chapter, we introduce finite difference approximations of derivatives. We follow the approach of LeVeque [1] and Ndayisenga et al. [5] to derive the forward, backward, central, and three-point difference formulas used to discretize the problem.

2.1 Taylor Expansion

Let $u \in C^\infty([a, b])$ and $h > 0$. For $x \in (a, b)$ with $x + h, x - h \in (a, b)$, Taylor's theorem gives the infinite series expansions:

$$u(x + h) = u(x) + h u'(x) + \frac{h^2}{2!} u''(x) + \cdots + \frac{h^k}{k!} u^{(k)}(x) + \cdots \quad (2.1)$$

$$u(x - h) = u(x) - h u'(x) + \frac{h^2}{2!} u''(x) - \cdots + \frac{(-h)^k}{k!} u^{(k)}(x) + \cdots \quad (2.2)$$

If $u \in C^k([a, b])$ for some integer $k \geq 1$, Taylor's theorem with Lagrange remainder we can write:

$$u(x + h) = u(x) + h u'(x) + \frac{h^2}{2!} u''(x) + \cdots + \frac{h^{k-1}}{(k-1)!} u^{(k-1)}(x) + \frac{h^k}{k!} u^{(k)}(\xi_1) \quad (2.3)$$

$$u(x - h) = u(x) - h u'(x) + \frac{h^2}{2!} u''(x) - \cdots + \frac{(-h)^{k-1}}{(k-1)!} u^{(k-1)}(x) + \frac{(-h)^k}{k!} u^{(k)}(\xi_2) \quad (2.4)$$

where $\xi_1 \in (x, x + h)$ and $\xi_2 \in (x - h, x)$.

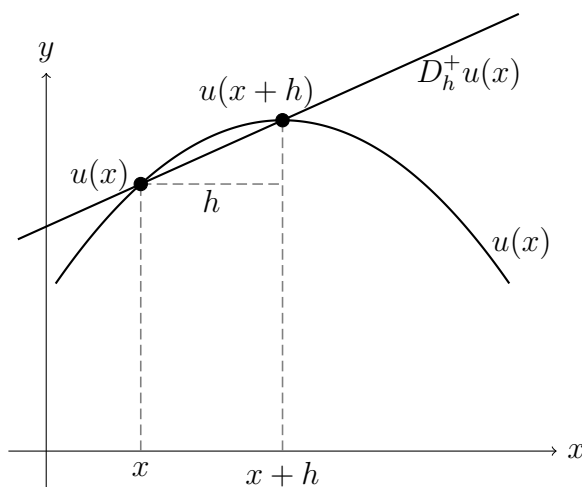
2.2 Approximation of the First Derivative

2.2.1 Forward Difference

Definition 2.1. Let $u : [a, b] \rightarrow \mathbb{R}$ and $h > 0$. For $x \in (a, b)$ with $x + h \in (a, b)$, the forward difference quotient of u at x is defined by:

$$D_h^+ u(x) = \frac{u(x + h) - u(x)}{h}.$$

$D_h^+ u(x)$ is interpreted as the slope of the line passing through the two points $(x, u(x))$ and $(x + h, u(x + h))$.

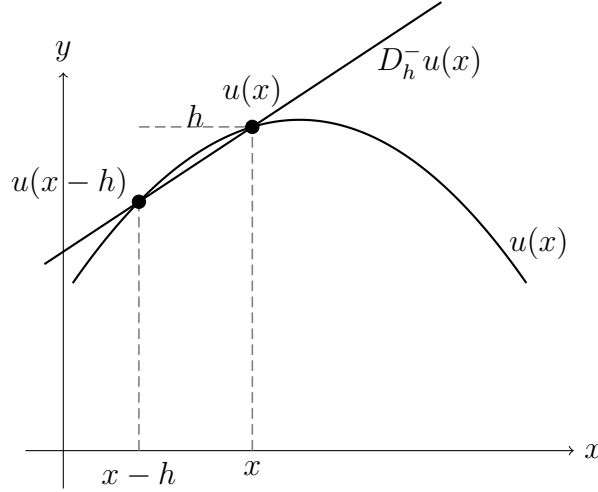


2.2.2 Backward Difference

Definition 2.2. Let $u : [a, b] \rightarrow \mathbb{R}$ and $h > 0$. For $x \in (a, b)$ with $x - h \in (a, b)$, the backward difference quotient of u at x is defined by :

$$D_h^- u(x) = \frac{u(x) - u(x - h)}{h}.$$

$D_h^- u(x)$ is interpreted as the slope of the line passing through the two points $(x - h, u(x - h))$ and $(x, u(x))$.

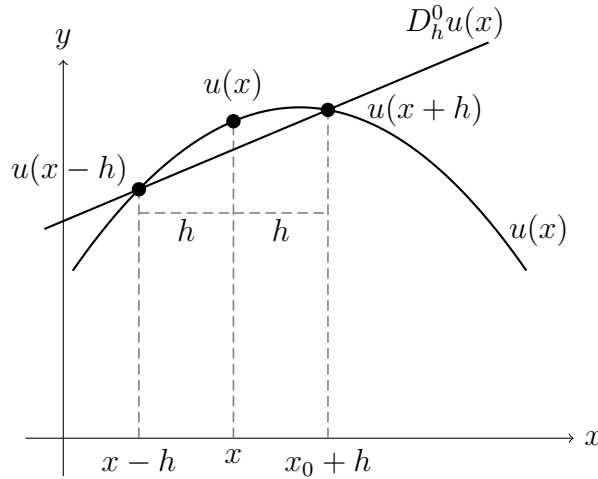


2.2.3 Central Difference

Definition 2.3. Let $u : [a, b] \rightarrow \mathbb{R}$ and $h > 0$. For $x \in (a, b)$ with $x - h$ and $x + h \in (a, b)$, the central difference quotient of u at x is defined by:

$$D_h^0 u(x) = \frac{u(x + h) - u(x - h)}{2h}.$$

$D_h^0 u(x)$ is interpreted as the slope of the line passing through the two points $(x - h, u(x - h))$ and $(x + h, u(x + h))$.



Property 2.2.1. The central difference quotient satisfies:

$$D_h^0 u(x) = \frac{1}{2}(D_h^+ u(x) + D_h^- u(x)).$$

Proof. By definition of D_h^+ and D_h^- , we have

$$\begin{aligned}
\frac{1}{2}(D_h^+ u(x) + D_h^- u(x)) &= \frac{1}{2} \left[\frac{u(x+h) - u(x)}{h} + \frac{u(x) - u(x-h)}{h} \right] \\
&= \frac{1}{2} \cdot \frac{u(x+h) - u(x-h)}{h} \\
&= \frac{u(x+h) - u(x-h)}{2h} \\
&= D_h^0 u(x).
\end{aligned}$$

□

2.3 Approximation of the Second Derivative

2.3.1 Three-Point Formula

Definition 2.4. Let $u : [a, b] \rightarrow \mathbb{R}$ and $h > 0$. For $x \in (a, b)$ with $x - h$ and $x + h \in (a, b)$, the three-point difference quotient of u at x is defined by:

$$D_h^2 u(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}.$$

Remark 2.3.1. The three-point operator can be viewed as applying the difference operator twice. For a finite difference operator D , we define

$$D^2 = D \cdot D,$$

that is, $D^2 u(x) = D(Du(x))$.

Property 2.3.1. The three-point difference quotient satisfies:

$$D_h^2 u(x) = D_h^+ D_h^- u(x).$$

Proof. By definition of D_h^+ and D_h^- , we have

$$\begin{aligned}
D_h^+(D_h^- u(x)) &= D_h^+\left(\frac{u(x) - u(x-h)}{h}\right) \\
&= \frac{1}{h} [D_h^+ u(x) - D_h^+ u(x-h)] \\
&= \frac{1}{h} \left[\frac{u(x+h) - u(x)}{h} - \frac{u(x) - u(x-h)}{h} \right] \\
&= \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} \\
&= D_h^2 u(x).
\end{aligned}$$

□

Property 2.3.2. *The operator D_h^2 satisfies:*

$$D_h^2 u(x) = D_h^- D_h^+ u(x).$$

Proof. We have

$$\begin{aligned}
D_h^-(D_h^+ u(x)) &= D_h^-\left(\frac{u(x+h) - u(x)}{h}\right) \\
&= \frac{1}{h} [D_h^- u(x+h) - D_h^- u(x)] \\
&= \frac{1}{h} \left[\frac{u(x+h) - u(x)}{h} - \frac{u(x) - u(x-h)}{h} \right] \\
&= \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} \\
&= D_h^2 u(x).
\end{aligned}$$

□

Summary of All Approximations

Name	Notation	Formula
Forward (u')	$D_h^+ u(x)$	$\frac{u(x+h) - u(x)}{h}$
Backward (u')	$D_h^- u(x)$	$\frac{u(x) - u(x-h)}{h}$
Central (u')	$D_h^0 u(x)$	$\frac{u(x+h) - u(x-h)}{2h}$
Three-point (u'')	$D_h^2 u(x)$	$\frac{u(x+h) - 2u(x) + u(x-h)}{h^2}$

Table 2.1: All finite difference approximations at $x \in (a, b)$, $x+h, x-h \in (a, b)$.

Chapter 3

Approximation of 1D Semilinear Problem

In this chapter, we present the continuous problem and derive the discrete scheme using the finite difference approximations from Chapter 2.

3.1 Continuous Problem

Let $[a, b]$ be an interval of $\mathbb{R}$ and $f : \mathbb{R} \rightarrow \mathbb{R}$ be a given function. We consider the one-dimensional semilinear Poisson problem.

$$-u''(x) = f(u(x)), \quad a < x < b, \quad (3.1)$$

subject to one of the following boundary conditions.

1. Dirichlet Boundary Condition

$$u(a) = g_0 \quad \text{and} \quad u(b) = g_1, \quad (3.2)$$

where $g_0, g_1 \in \mathbb{R}$ are given values.

2. Neumann Boundary Condition

$$u'(a) = h_0 \quad \text{and} \quad u'(b) = h_1, \quad (3.3)$$

where $h_0, h_1 \in \mathbb{R}$ are given values.

3. Mixed Boundary Condition

$$u(a) = g_0 \quad \text{and} \quad u'(b) = h_1, \quad (3.4)$$

or

$$u'(a) = h_0 \quad \text{and} \quad u(b) = g_1, \quad (3.5)$$

where $g_0, g_1, h_0, h_1 \in \mathbb{R}$ are given values.

We will simulate numerically the solution of problem (3.1) with the Dirichlet boundary condition by the finite difference method.

3.2 Partition of Segment

Let $n \geq 1$ be an integer. We divide $[a, b]$ into n equal sub-intervals defined by

$$I_i = [x_i, x_{i+1}], \quad i = 0, 1, \dots, n-1,$$

with

$$\Delta x_i = x_{i+1} - x_i$$

For a uniform mesh, we obtain

$$x_i = a + i \Delta x, \quad i = 0, 1, \dots, n,$$

with

$$\Delta x = \frac{b - a}{n}$$

3.3 Discrete Scheme

We denote $u_i \approx u(x_i)$ for $i = 0, 1, \dots, n$. Using the three-point formula D_h^2 from Chapter 2, we approximate $u''(x_i)$ by

$$u''(x_i) \approx \frac{u_{i-1} - 2u_i + u_{i+1}}{(\Delta x)^2}.$$

Substituting into problem (3.1), we obtain the discrete scheme at each interior node x_i :

$$-\frac{1}{(\Delta x)^2}u_{i-1} + \frac{2}{(\Delta x)^2}u_i - \frac{1}{(\Delta x)^2}u_{i+1} = f(u_i), \quad i = 1, 2, \dots, n-1. \quad (3.6)$$

Equivalently, this can be written as

$$-\frac{1}{(\Delta x)^2}u_{i-1} + \frac{2}{(\Delta x)^2}u_i - \frac{1}{(\Delta x)^2}u_{i+1} - f(u_i) = 0, \quad i = 1, 2, \dots, n-1. \quad (3.7)$$

The Dirichlet boundary values are given by

$$u(a) = g_0,$$

and

$$u(b) = g_1.$$

Using the finite difference approximation, we obtain the following nonlinear system:

$$\begin{cases} u_0 - g_0 = 0, \\ -\frac{1}{(\Delta x)^2}u_{i-1} + \frac{2}{(\Delta x)^2}u_i - \frac{1}{(\Delta x)^2}u_{i+1} - f(u_i) = 0, & i = 1, 2, \dots, n-1, \\ u_n - g_1 = 0. \end{cases}$$

We obtain a nonlinear system of $n+1$ equations for the $n+1$ unknowns, where $\mathbf{u}$ is the vector of unknowns

$$\mathbf{u} = (u_0, u_1, \dots, u_n)^T.$$

Since $f(u_i)$ depends on the unknown u_i , the system (3.7) is nonlinear and cannot be solved directly. We define the residual function. For the boundary nodes:

$$F_0(\mathbf{u}) = u_0 - g_0,$$

and

$$F_n(\mathbf{u}) = u_n - g_1.$$

For the interior nodes $i = 1, 2, \dots, n-1$:

$$F_i(\mathbf{u}) = -\frac{u_{i-1} - 2u_i + u_{i+1}}{(\Delta x)^2} - f(u_i).$$

We seek $\mathbf{u}$ such that $F(\mathbf{u}) = \mathbf{0}$, where

$$F(\mathbf{u}) = \begin{pmatrix} F_0(\mathbf{u}) \\ F_1(\mathbf{u}) \\ \vdots \\ F_{n-1}(\mathbf{u}) \\ F_n(\mathbf{u}) \end{pmatrix} = \begin{pmatrix} u_0 - g_0 \\ -\frac{1}{(\Delta x)^2}u_0 + \frac{2}{(\Delta x)^2}u_1 - \frac{1}{(\Delta x)^2}u_2 - f(u_1) \\ \vdots \\ -\frac{1}{(\Delta x)^2}u_{n-2} + \frac{2}{(\Delta x)^2}u_{n-1} - \frac{1}{(\Delta x)^2}u_n - f(u_{n-1}) \\ u_n - g_1 \end{pmatrix}.$$

This nonlinear problem is solved by the Newton-Raphson [2] below.

3.4 Newton–Raphson Algorithm

Following Jia [8], we apply the Newton–Raphson method to solve the nonlinear system iteratively.

We seek the solution of the nonlinear system

$$F(\mathbf{u}) = \mathbf{0}.$$

Since the residual function $F(\mathbf{u})$ is nonlinear, the system $F(\mathbf{u}) = \mathbf{0}$ is solved iteratively using the Newton–Raphson method. Applying the first-order Taylor expansion about the current iterate $\mathbf{u}^{(k)}$, we obtain

$$F(\mathbf{u}) \approx F(\mathbf{u}^{(k)}) + J(\mathbf{u}^{(k)}) (\mathbf{u} - \mathbf{u}^{(k)}).$$

Assume that $\mathbf{u} = \mathbf{u}^{(k+1)}$ is a solution of $F(\mathbf{u}) = \mathbf{0}$, then

$$\mathbf{0} = F(\mathbf{u}^{(k)}) + J(\mathbf{u}^{(k)}) (\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}).$$

Let

$$\Delta \mathbf{u}^{(k)} = \mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}.$$

Then,

$$J(\mathbf{u}^{(k)}) \Delta \mathbf{u}^{(k)} = -F(\mathbf{u}^{(k)}).$$

The solution is updated by

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \Delta \mathbf{u}^{(k)}, \quad k = 0, 1, 2, \dots$$

The iteration continues until

$$\|\Delta \mathbf{u}^{(k)}\|_{\infty} < \varepsilon,$$

where $\varepsilon > 0$ is a prescribed tolerance.

3.4.1 Jacobian Matrix

Definition 3.1. *Given a vector-valued function $\mathbf{F} : \mathbb{R}^n \rightarrow \mathbb{R}^n$, the Jacobian matrix element is defined by:*

$$J_{i,j} = \frac{\partial F_i}{\partial u_j}, \quad i, j = 0, 1, \dots, n$$

where:

- F_i is the i -th component of the vector-valued function $\mathbf{F}$
- u_j is the j -th input variable
- $J_{i,j}$ represents how the i -th output changes with respect to the j -th input

The full matrix takes the form:

$$J = \begin{pmatrix} \frac{\partial F_0}{\partial u_0} & \frac{\partial F_0}{\partial u_1} & \frac{\partial F_0}{\partial u_2} & \cdots & \frac{\partial F_0}{\partial u_{n-1}} & \frac{\partial F_0}{\partial u_n} \\ \frac{\partial F_1}{\partial u_0} & \frac{\partial F_1}{\partial u_1} & \frac{\partial F_1}{\partial u_2} & \cdots & \frac{\partial F_1}{\partial u_{n-1}} & \frac{\partial F_1}{\partial u_n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{\partial F_{n-1}}{\partial u_0} & \frac{\partial F_{n-1}}{\partial u_1} & \cdots & \frac{\partial F_{n-1}}{\partial u_{n-2}} & \frac{\partial F_{n-1}}{\partial u_{n-1}} & \frac{\partial F_{n-1}}{\partial u_n} \\ \frac{\partial F_n}{\partial u_0} & \frac{\partial F_n}{\partial u_1} & \frac{\partial F_n}{\partial u_2} & \cdots & \frac{\partial F_n}{\partial u_{n-1}} & \frac{\partial F_n}{\partial u_n} \end{pmatrix}$$

1. Boundary Node Derivatives ($i = 0, n$) Differentiating the boundary residuals directly gives:

$$\frac{\partial F_0}{\partial u_j} = \frac{\partial}{\partial u_j}(u_0 - g_0) = \begin{cases} 1 & \text{if } j = 0, \\ 0 & \text{if } j \neq 0, \end{cases}$$

$$\frac{\partial F_n}{\partial u_j} = \frac{\partial}{\partial u_j}(u_n - g_1) = \begin{cases} 1 & \text{if } j = n, \\ 0 & \text{if } j \neq n. \end{cases}$$

2. Interior Node Derivatives ($i = 1, \dots, n-1$) **Otherwise** ($|i - j| > 1$):

$$\frac{\partial F_i}{\partial u_j} = 0.$$

3. Complete Matrix System The full Jacobian $J \in \mathbb{R}^{(n+1) \times (n+1)}$ is:

$$J = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ \frac{\partial F_1}{\partial u_0} & \frac{\partial F_1}{\partial u_1} & \frac{\partial F_1}{\partial u_2} & \cdots & 0 & 0 \\ 0 & \frac{\partial F_2}{\partial u_1} & \frac{\partial F_2}{\partial u_2} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial F_{n-1}}{\partial u_{n-2}} & \frac{\partial F_{n-1}}{\partial u_{n-1}} & \frac{\partial F_{n-1}}{\partial u_n} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Therefore,

$$J = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -\frac{1}{(\Delta x)^2} & \frac{2}{(\Delta x)^2} - \frac{\partial f(u_1)}{\partial u_1} & -\frac{1}{(\Delta x)^2} & \cdots & 0 & 0 \\ 0 & -\frac{1}{(\Delta x)^2} & \frac{2}{(\Delta x)^2} - \frac{\partial f(u_2)}{\partial u_2} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -\frac{1}{(\Delta x)^2} & \frac{2}{(\Delta x)^2} - \frac{\partial f(u_{n-1})}{\partial u_{n-1}} & -\frac{1}{(\Delta x)^2} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

3.4.2 Example

In this section, we apply the Dirichlet boundary condition to two specific choices of f .

Case 1: $f(u(x)) = x(1 - x)$

Since f does not depend on u , we have $f(u(x)) = f(x) = x(1 - x)$. We consider the linear problem

$$\begin{cases} -u''(x) = x(1 - x), & x \in (0, 1), \\ u(0) = 0, \quad u(1) = 0. \end{cases}$$

Exact Solution. Since the right-hand side depends only on x , we obtain the exact solution by direct integration. Rewriting the equation as

$$u''(x) = -x + x^2,$$

and integrating once gives

$$u'(x) = -\frac{x^2}{2} + \frac{x^3}{3} + C_1.$$

Integrating again:

$$u(x) = -\frac{x^3}{6} + \frac{x^4}{12} + C_1x + C_2.$$

Applying $u(0) = 0$ gives $C_2 = 0$, and applying $u(1) = 0$ gives

$$-\frac{1}{6} + \frac{1}{12} + C_1 = 0 \implies C_1 = \frac{1}{12}.$$

Hence the exact solution is

$$u(x) = -\frac{x^3}{6} + \frac{x^4}{12} + \frac{x}{12}.$$

Jacobian Matrix. The full Jacobian $J \in \mathbb{R}^{(n+1) \times (n+1)}$ becomes:

$$J = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -\frac{1}{(\Delta x)^2} & \frac{2}{(\Delta x)^2} - \frac{\partial f(x_1)}{\partial u_1} & -\frac{1}{(\Delta x)^2} & \cdots & 0 & 0 \\ 0 & -\frac{1}{(\Delta x)^2} & \frac{2}{(\Delta x)^2} - \frac{\partial f(x_2)}{\partial u_2} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -\frac{1}{(\Delta x)^2} & \frac{2}{(\Delta x)^2} - \frac{\partial f(x_{n-1})}{\partial u_{n-1}} & -\frac{1}{(\Delta x)^2} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Since $f(x) = x(1 - x)$ does not depend on u , and therefore:

$$J = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -\frac{1}{(\Delta x)^2} & \frac{2}{(\Delta x)^2} & -\frac{1}{(\Delta x)^2} & \cdots & 0 & 0 \\ 0 & -\frac{1}{(\Delta x)^2} & \frac{2}{(\Delta x)^2} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -\frac{1}{(\Delta x)^2} & \frac{2}{(\Delta x)^2} & -\frac{1}{(\Delta x)^2} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Case 2: $f(u(x)) = u(x)(1 - u(x))$

We consider the nonlinear problem

$$\begin{cases} -u''(x) = u(x)(1 - u(x)), & x \in (0, 1), \\ u(0) = 0, \quad u(1) = 1. \end{cases}$$

Exact Solution. Since $f(u) = u(1 - u)$ is nonlinear in u , the equation becomes

$$u''(x) = -u(x) + u(x)^2,$$

which is a nonlinear second-order ODE. No closed-form exact solution exists for this problem. Therefore, Newton–Raphson iteration is applied to approximate the solution numerically.

Jacobian Matrix. The full Jacobian $J \in \mathbb{R}^{(n+1) \times (n+1)}$ is:

$$J = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -\frac{1}{(\Delta x)^2} & \frac{2}{(\Delta x)^2} - \frac{\partial f(u_1)}{\partial u_1} & -\frac{1}{(\Delta x)^2} & \cdots & 0 & 0 \\ 0 & -\frac{1}{(\Delta x)^2} & \frac{2}{(\Delta x)^2} - \frac{\partial f(u_2)}{\partial u_2} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -\frac{1}{(\Delta x)^2} & \frac{2}{(\Delta x)^2} - \frac{\partial f(u_{n-1})}{\partial u_{n-1}} & -\frac{1}{(\Delta x)^2} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Since $f(u) = u(1 - u)$, and therefore:

$$J = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -\frac{1}{(\Delta x)^2} & \frac{2}{(\Delta x)^2} - (1 - 2u_1) & -\frac{1}{(\Delta x)^2} & \cdots & 0 & 0 \\ 0 & -\frac{1}{(\Delta x)^2} & \frac{2}{(\Delta x)^2} - (1 - 2u_2) & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -\frac{1}{(\Delta x)^2} & \frac{2}{(\Delta x)^2} - (1 - 2u_{n-1}) & -\frac{1}{(\Delta x)^2} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Chapter 4

Simulation

In this chapter, we simulate numerically the solution of the one-dimensional semi-linear problem

$$-u''(x) = f(u(x)), \quad x \in (0, 1),$$

with the Dirichlet boundary condition $u(0) = g_0$ and $u(1) = g_1$, for two choices of f .

4.1 Case 1: $f(u(x)) = x(1 - x)$

We numerically solve the boundary value problem,

$$\begin{cases} -u''(x) = x(1 - x), & x \in (0, 1), \\ u(0) = 0 \\ u(1) = 0. \end{cases}$$

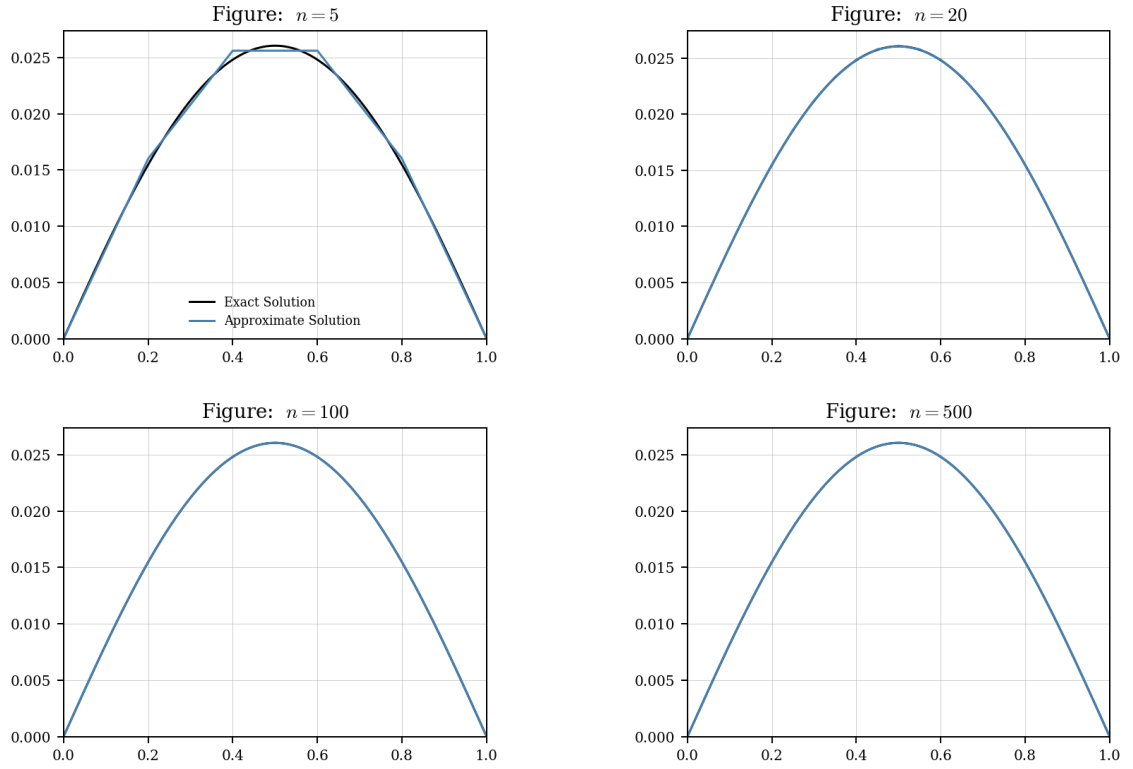


Figure 4.1: Approximate solution vs exact solution for $n = 5, 20, 100, 500$.

4.2 Case 2: $f(u(x)) = u(x)(1 - u(x))$

We consider the following boundary value problem,

$$\begin{cases} -u''(x) = u(x)(1 - u(x)), & x \in (0, 1), \\ u(0) = 0 \\ u(1) = 1. \end{cases}$$

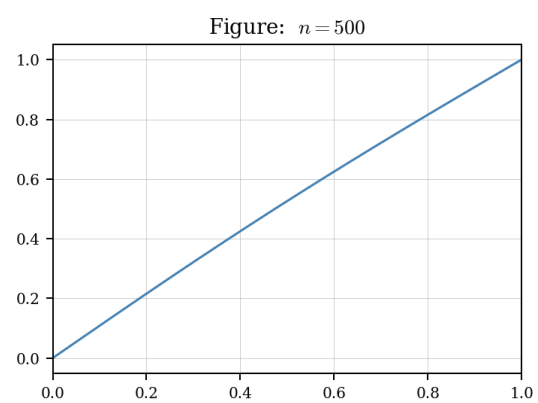
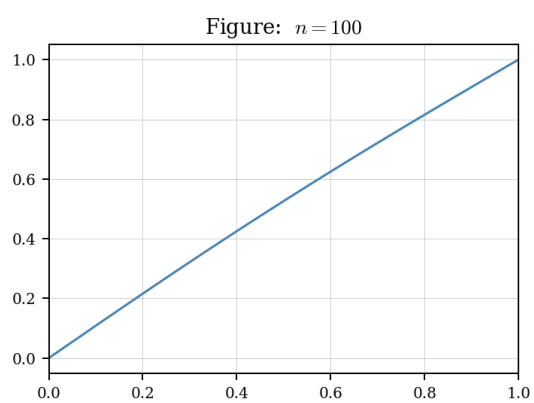
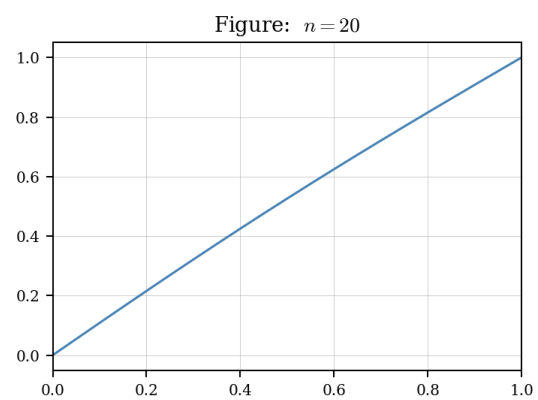
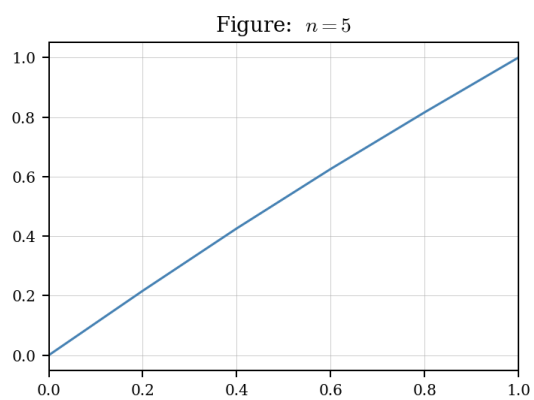


Figure 4.2: Numerical solution for $n = 5, 20, 100, 500$.

Chapter 5

Appendix Python Code

5.1 Case 1: $f(u(x)) = x(1 - x)$

Below is a Python driver code to solve of

$$\begin{cases} -u''(x) = x(1 - x), & x \in (0, 1), \\ u(0) = 0 \\ u(1) = 0. \end{cases}$$

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # Exact solution
5 def exact_solution(x):
6     return -x**3/6 + x**4/12 + x/12
7
8 # Solve Au = b (linear system)
9 # Input: number of subintervals n
10 n = 100
11
12 h = 1.0 / n
13 x = np.linspace(0, 1, n + 1)
14
15 # Build tridiagonal matrix A (interior nodes only)
16 A = np.zeros((n-1, n-1))
17 for i in range(n-1):
18     A[i, i] = 2.0 / h**2
19     if i > 0:
```

```

20     A[i, i-1] = -1.0 / h**2
21     if i < n-2:
22         A[i, i+1] = -1.0 / h**2
23
24     # Build right-hand side b
25     xi = x[1:n]
26     b = xi * (1 - xi)
27
28     # Solve Au = b
29     u_int = np.linalg.solve(A, b)
30
31     # Add boundary values u(0)=0, u(1)=0
32     u = np.concatenate([[0], u_int, [0]])
33
34     # Plot approximate vs exact
35     x_fine = np.linspace(0, 1, 500)
36     u_ex = exact_solution(x_fine)
37
38     plt.plot(x_fine, u_ex, color='black', linewidth=1.2, label='Exact Solution')
39     plt.plot(x, u, color='steelblue', linewidth=1.2, label='Approximate Solution')
40     plt.title(f'Figure: $n = {n}$')
41     plt.xlim(0, 1)
42     plt.grid(True, linewidth=0.4, alpha=0.6)
43     plt.legend(fontsize=8, frameon=False)
44     plt.tight_layout()
45     plt.savefig(f'case1_n{n}.png', dpi=180, bbox_inches='tight')
46     plt.show()

```

Listing 5.1: Linear case $f(x) = x(1 - x)$

5.2 Case 2: $f(u) = u(1 - u)$

Below is a Python driver code to solve of

$$\begin{cases} -u''(x) = u(x)(1 - u(x)), & x \in (0, 1), \\ u(0) = 0 \\ u(1) = 1. \end{cases}$$

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # Solve by Newton-Raphson (nonlinear)

```

```

5  # Input: number of subintervals n
6  n = 100
7  g0 = 0      # u(0) = 0
8  g1 = 1      # u(1) = 1
9  tol = 1e-10
10
11 h = 1.0 / n
12 x = np.linspace(0, 1, n + 1)
13
14 # Linear initial guess
15 u = np.linspace(g0, g1, n + 1)
16
17 # Newton-Raphson iteration
18 for _ in range(100):
19     F = np.zeros(n + 1)
20     J = np.zeros((n + 1, n + 1))
21
22     # Interior nodes i = 1, ..., n-1
23     for i in range(1, n):
24         F[i] = -(u[i-1] - 2*u[i] + u[i+1]) / h**2 - u[i]*(1 - u[i])
25         J[i, i-1] = -1.0 / h**2
26         J[i, i] = 2.0 / h**2 - (1 - 2*u[i])
27         J[i, i+1] = -1.0 / h**2
28
29     # Boundary conditions
30     F[0] = u[0] - g0; J[0, 0] = 1.0
31     F[n] = u[n] - g1; J[n, n] = 1.0
32
33     # Solve J * delta_u = -F
34     delta_u = np.linalg.solve(J, -F)
35     u = u + delta_u
36     if np.max(np.abs(delta_u)) < tol:
37         break
38
39 # Plot solution
40 plt.plot(x, u, color='steelblue', linewidth=1.2)
41 plt.title(f'Figure: $n = {n}$')
42 plt.xlim(0, 1)
43 plt.grid(True, linewidth=0.4, alpha=0.6)
44 plt.tight_layout()
45 plt.savefig(f'case2_n{n}.png', dpi=180, bbox_inches='tight')
46 plt.show()

```

Listing 5.2: Nonlinear case $f(u) = u(1 - u)$

Bibliography

- [1] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations*, SIAM, Philadelphia, 2007.
- [2] C. T. Kelley, *Solving Nonlinear Equations with Newton's Method*, SIAM, Philadelphia, 2003.
- [3] R. L. Burden and J. D. Faires, *Numerical Analysis*, 10th ed., Cengage Learning, Boston, 2015.
- [4] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, 2nd ed., SIAM, Philadelphia, 2004.
- [5] S. Ndayisenga, L. A. Sevastianov, and K. P. Lovetskiy, *Finite-Difference Methods for Solving 1D Poisson Problem*, Discrete and Continuous Models and Applied Computational Science, 2022.
- [6] G. Nagy, *Ordinary Differential Equations*, Michigan State University Lecture Notes, 2021.
- [7] P. Knabner and L. Angermann, *Numerical Methods for Elliptic and Parabolic Partial Differential Equations*, Springer, New York, 2003.
- [8] H. Jia, *Newton–Raphson Method and Arithmetic Mean Newton's Method for Solving Systems of Nonlinear Equations*, Honours Seminar Project, Department of Mathematical Sciences, Lakehead University, Thunder Bay, Canada, 2018.