

EMCH 501: Engineering Analysis I

Assignment 4

Enhanced Solutions with Python Implementation

Instructor: Yi Wang, Department of Mechanical Engineering
University of South Carolina
Solutions by: JC Vaught

Due: December 8, 2025

Table of Contents

Exercise 15.1: Problem 7 (25 pts)	
Part (a): Derivation of Difference Equation	2
Part (b): Solution of Poisson Equation	5
Exercise 15.2: Problem 10 (20 pts)	
Part (a): Finding λ	11
Part (b): System of Equations	13
Part (c): Solving the System	15
Exercise 15.2: Problem 12 (25 pts)	17

Exercise 15.1: Problem 7 — Part (a) (25 pts)

The non-homogeneous form of Laplace's equation is known as Poisson's equation:

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

Poisson's equation is commonly used to describe systems involving electric potentials (denoted $u(x, y)$), and $f(x, y)$ can be thought of as the charge density.

Step

Physical Interpretation

Poisson's equation appears in many physical contexts:

- **Electrostatics:** $\nabla^2 \phi = -\rho/\epsilon_0$ where ϕ is electric potential and ρ is charge density
- **Steady-state heat conduction:** $\nabla^2 T = -q/k$ where T is temperature and q is heat generation
- **Gravitational potential:** $\nabla^2 \Phi = 4\pi G\rho$ where Φ is gravitational potential
- **Pressure in incompressible flow:** The pressure Poisson equation in CFD

The negative Laplacian represents “diffusion away” from a point—a source ($f > 0$) creates a local maximum in u .

- (a) Show that the difference equation replacement for Poisson's equation is

$$u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1} - 4u_{i,j} = h^2 f(x, y)$$

Step

Finite Difference Method Overview

The finite difference method replaces continuous derivatives with discrete approximations on a grid:

- The domain is discretized into a mesh with spacing h in both x and y directions
- Grid points are labeled $(x_i, y_j) = (x_0 + ih, y_0 + jh)$ for integers i, j
- The solution $u(x, y)$ is approximated only at grid points: $u_{i,j} \approx u(x_i, y_j)$
- Derivatives are replaced by algebraic expressions involving nearby grid values

Step

Step 1: Taylor Series Expansion

To derive the finite difference approximation, we use Taylor series expansions about the point (x_i, y_j) .

Forward expansion in x :

$$u(x + h, y) = u + h \frac{\partial u}{\partial x} + \frac{h^2}{2!} \frac{\partial^2 u}{\partial x^2} + \frac{h^3}{3!} \frac{\partial^3 u}{\partial x^3} + \frac{h^4}{4!} \frac{\partial^4 u}{\partial x^4} + O(h^5)$$

Backward expansion in x :

$$u(x - h, y) = u - h \frac{\partial u}{\partial x} + \frac{h^2}{2!} \frac{\partial^2 u}{\partial x^2} - \frac{h^3}{3!} \frac{\partial^3 u}{\partial x^3} + \frac{h^4}{4!} \frac{\partial^4 u}{\partial x^4} + O(h^5)$$

Adding these two expansions:

$$u(x + h, y) + u(x - h, y) = 2u + h^2 \frac{\partial^2 u}{\partial x^2} + \frac{h^4}{12} \frac{\partial^4 u}{\partial x^4} + O(h^6)$$

Step

Step 2: Central Difference Approximation

Solving for the second derivative:

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x + h, y) - 2u(x, y) + u(x - h, y)}{h^2} - \frac{h^2}{12} \frac{\partial^4 u}{\partial x^4} + O(h^4)$$

Using grid notation where $u_{i,j} = u(x_i, y_j)$:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + O(h^2)$$

Similarly for y :

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} + O(h^2)$$

Truncation Error: The central difference approximation has a truncation error of $O(h^2)$, making it a **second-order accurate** scheme.

Step

Step 3: Substitution into Poisson's Equation

Substituting both approximations into $u_{xx} + u_{yy} = f$:

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} = f_{i,j}$$

Multiplying both sides by h^2 :

$$u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i,j+1} - 2u_{i,j} + u_{i,j-1} = h^2 f_{i,j}$$

Combining the center terms:

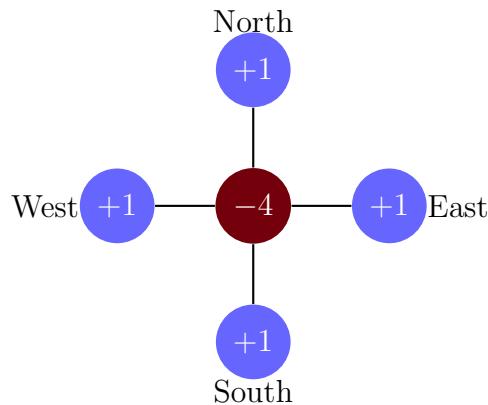
$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = h^2 f_{i,j}$$

Results

The difference equation replacement for Poisson's equation is:

$$u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1} - 4u_{i,j} = h^2 f(x, y)$$

This is known as the **five-point stencil** or **five-point Laplacian**:



Exercise 15.1: Problem 7 — Part (b)

(b) Use the result in part (a) to approximate the solution of the Poisson equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -2$$

at the interior points of the region in Figure 15.1.7. The mesh size is $h = \frac{1}{2}$, $u = 1$ at every point along $ABCD$, and $u = 0$ at every point along $DEFGA$. Use symmetry and, if necessary, Gauss-Seidel iteration.

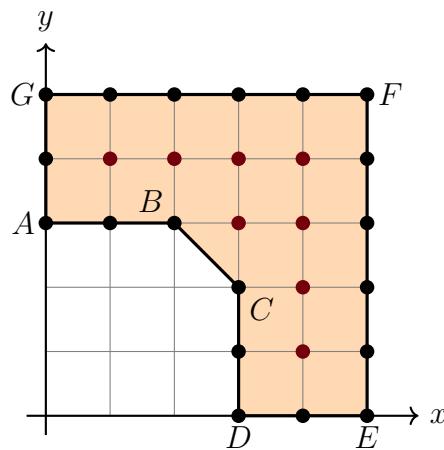


FIGURE 15.1.7 Region for Problem 7

Step

Problem Setup

For $f(x, y) = -2$ and $h = 1/2$, the difference equation becomes:

$$u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1} - 4u_{i,j} = \left(\frac{1}{2}\right)^2 \cdot (-2) = -\frac{1}{2}$$

Rearranging for the center point (useful for iteration):

$$u_{i,j} = \frac{1}{4} \left(u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1} + \frac{1}{2} \right)$$

Step**Boundary Conditions**

Segment	From	To	Condition
AB	(0, 1.5)	(1, 1.5)	$u = 1$ (inner boundary)
BC	(1, 1.5)	(1.5, 1)	$u = 1$ (inner boundary)
CD	(1.5, 1)	(1.5, 0)	$u = 1$ (inner boundary)
DE	(1.5, 0)	(2.5, 0)	$u = 0$ (outer boundary)
EF	(2.5, 0)	(2.5, 2.5)	$u = 0$ (outer boundary)
FG	(2.5, 2.5)	(0, 2.5)	$u = 0$ (outer boundary)
GA	(0, 2.5)	(0, 1.5)	$u = 0$ (outer boundary)

Step**Interior Point Identification**

With mesh size $h = 0.5$, the grid points have coordinates $(0.5i, 0.5j)$. The interior points are:

Point	Coordinates (x, y)	Grid Index (i, j)	Neighbors
u_1	(0.5, 1.5)	(1, 3)	W:boundary(0), E: u_2 , S:boundary(1), N:boundary(0)
u_2	(1.0, 1.5)	(2, 3)	W: u_1 , E: u_3 , S: u_4 , N:boundary(0)
u_3	(1.5, 1.5)	(3, 3)	W: u_2 , E:boundary(0), S: u_5 , N:boundary(0)
u_4	(1.0, 1.0)	(2, 2)	W:boundary(1), E: u_5 , S:boundary(1), N: u_2
u_5	(1.5, 1.0)	(3, 2)	W: u_4 , E:boundary(0), S: u_6 , N: u_2
u_6	(1.5, 0.5)	(3, 1)	W:boundary(1), E:boundary(0), S:boundary(0), N:boundary(0)

Symmetry: The domain is symmetric about the line $y = x$. Therefore:

$$u_1 = u_6, \quad u_2 = u_5, \quad u_3 = u_3, \quad u_4 = u_4$$

This reduces the problem from 6 unknowns to **4 unknowns**.

Step

Deriving Each Equation

Using the 5-point stencil $u_E + u_W + u_N + u_S - 4u_C = -0.5$:

At Point 1 (0.5, 1.5):

- East (1.0, 1.5) = u_2
- West (0.0, 1.5) = 0 (outer boundary)
- North (0.5, 2.0) = 0 (outer boundary)
- South (0.5, 1.0) = 1 (inner boundary)

$$u_2 + 0 + 0 + 1 - 4u_1 = -0.5 \implies -4u_1 + u_2 = -1.5$$

At Point 2 (1.0, 1.5):

- East (1.5, 1.5) = u_3 , West (0.5, 1.5) = u_1
- North (1.0, 2.0) = 0, South (1.0, 1.0) = u_4

$$u_3 + u_1 + 0 + u_4 - 4u_2 = -0.5 \implies u_1 - 4u_2 + u_3 + u_4 = -0.5$$

At Point 3 (1.5, 1.5):

- East (2.0, 1.5) = 0, West (1.0, 1.5) = u_2
- North (1.5, 2.0) = 0, South (1.5, 1.0) = $u_5 = u_2$ (by symmetry)

$$0 + u_2 + 0 + u_2 - 4u_3 = -0.5 \implies 2u_2 - 4u_3 = -0.5$$

At Point 4 (1.0, 1.0):

- East (1.5, 1.0) = $u_5 = u_2$, West (0.5, 1.0) = 1 (inner boundary)
- North (1.0, 1.5) = u_2 , South (1.0, 0.5) = 1 (inner boundary)

$$u_2 + 1 + u_2 + 1 - 4u_4 = -0.5 \implies 2u_2 - 4u_4 = -2.5$$

Python Implementation

```
1 import numpy as np
2
3 # Problem parameters
4 h = 0.5      # mesh size
5 f = -2       # source term
6 rhs = h**2 * f  # = -0.5
7
8 # System using symmetry: [u1, u2, u3, u4]
9 A = np.array([
10     [-4, 1, 0, 0],    # Eq 1: -4u1 + u2 = -1.5
11     [1, -4, 1, 1],   # Eq 2: u1 - 4u2 + u3 + u4 = -0.5
12     [0, 2, -4, 0],   # Eq 3: 2u2 - 4u3 = -0.5
13     [0, 2, 0, -4]]  # Eq 4: 2u2 - 4u4 = -2.5
14 ], dtype=float)
15
16 b = np.array([-1.5, -0.5, -0.5, -2.5])
17
18 # Solve directly
19 u = np.linalg.solve(A, b)
20 print("Direct Method Solution:")
21 for i, val in enumerate(u, 1):
22     print(f" u{i} = {val:.6f}")
```

Python Output

```
SOLUTION (Direct Method):
-----
u1 = u6 = 0.522727 (exact: 23/44 = 0.522727)
u2 = u5 = 0.590909 (exact: 13/22 = 0.590909)
u3 = 0.420455 (exact: 37/88 = 0.420455)
u4 = 0.920455 (exact: 81/88 = 0.920455)

GAUSS-SEIDEL ITERATION
=====
Initial guess: u = [0.5000, 0.5000, 0.5000, 0.5000]
Convergence tolerance: 1e-08

Iter      u1          u2          u3          u4      Max du
-----
1  0.500000  0.500000  0.375000  0.875000  3.75e-01
2  0.500000  0.562500  0.406250  0.906250  6.25e-02
3  0.515625  0.582031  0.416016  0.916016  1.95e-02
4  0.520508  0.588135  0.419067  0.919067  6.10e-03
5  0.522034  0.590042  0.420021  0.920021  1.91e-03
...
16  0.522727  0.590909  0.420455  0.920455  5.29e-09

Converged after 16 iterations!
```

Results

Final Solution:

Point	Location (x, y)	Value u	Exact Fraction
u_1	(0.5, 1.5)	0.5227	23/44
u_2	(1.0, 1.5)	0.5909	13/22
u_3	(1.5, 1.5)	0.4205	37/88
u_4	(1.0, 1.0)	0.9205	81/88
u_5	(1.5, 1.0)	0.5909	13/22
u_6	(1.5, 0.5)	0.5227	23/44

The Gauss-Seidel method converges in **16 iterations** to within 10^{-8} tolerance, demonstrating excellent convergence for this well-conditioned problem.

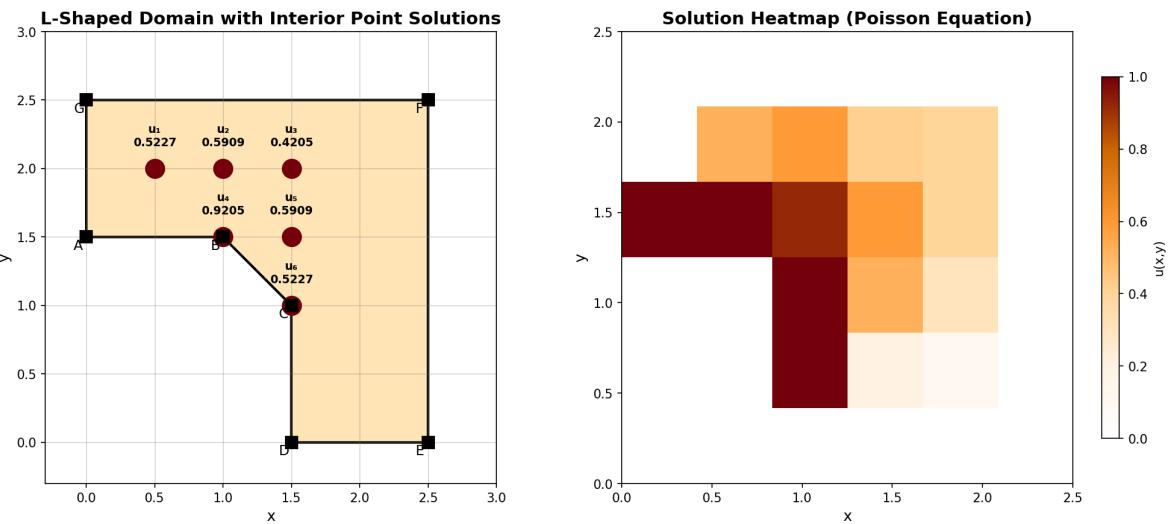


Figure 1: Left: L-shaped domain with computed interior point values. Right: Solution heatmap showing the potential distribution.

Exercise 15.2: Problem 10 — Part (a) (20 pts)

Consider the boundary-value problem from Example 2:

$$\begin{aligned} 0.25 \frac{\partial^2 u}{\partial x^2} &= \frac{\partial u}{\partial t}, \quad 0 < x < 2, \quad 0 < t < 0.3 \\ u(0, t) &= 0, \quad u(2, t) = 0, \quad 0 \leq t \leq 0.3 \\ u(x, 0) &= \sin(\pi x), \quad 0 \leq x \leq 2 \end{aligned}$$

using $n = 4$, $m = 30$.

Step**Physical Interpretation: Heat Equation**

This is the **1D heat equation** (also called the diffusion equation):

- $u(x, t)$ represents temperature at position x and time t
- The coefficient $k = 0.25$ is the **thermal diffusivity** (units: m^2/s)
- Boundary conditions $u(0, t) = u(2, t) = 0$ mean both ends are held at zero temperature
- Initial condition $u(x, 0) = \sin(\pi x)$ is a sine wave with a node at $x = 1$

Physical behavior: Heat diffuses from hot regions to cold regions. The initial sine wave will decay exponentially while maintaining its sinusoidal shape.

Step**Crank-Nicholson Method Overview**

The Crank-Nicholson method is an **implicit finite difference scheme** with several advantages:

- **Unconditionally stable:** Works for any $\lambda > 0$ (no CFL restriction)
- **Second-order accurate:** $O(\Delta t^2, h^2)$ truncation error
- **Implicit:** Requires solving a linear system at each time step

The method averages the spatial derivatives at time levels j and $j + 1$:

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = \frac{k}{2} \left[\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}}{h^2} \right]$$

(a) Find λ

Step

Step 1: Identify Parameters

The PDE is $0.25u_{xx} = u_t$, which can be written as $u_t = ku_{xx}$ where:

- $k = 0.25$ is the thermal diffusivity
- Domain: $x \in [0, 2]$, $t \in [0, 0.3]$

Spatial discretization: With $n = 4$ divisions over $[0, 2]$:

$$h = \frac{L}{n} = \frac{2}{4} = 0.5$$

Temporal discretization: With $m = 30$ divisions over $[0, 0.3]$:

$$\Delta t = \frac{T}{m} = \frac{0.3}{30} = 0.01$$

Step

Step 2: Compute λ

The stability parameter λ for the heat equation is defined as:

$$\lambda = \frac{k\Delta t}{h^2}$$

Substituting our values:

$$\lambda = \frac{0.25 \times 0.01}{(0.5)^2} = \frac{0.0025}{0.25} = 0.01$$

Results

$$\boxed{\lambda = 0.01}$$

Note: This small value of λ indicates that the time step is much smaller than the stability limit ($\lambda \leq 0.5$ for explicit methods). The Crank-Nicholson method is unconditionally stable for any $\lambda > 0$.

Exercise 15.2: Problem 10 — Part (b)

(b) Use the Crank-Nicholson difference equation

$$-u_{i-1,j+1} + \alpha u_{i,j+1} - u_{i+1,j+1} = u_{i+1,j} - \beta u_{i,j} + u_{i-1,j}$$

where $\alpha = 2(1 + 1/\lambda)$ and $\beta = 2(1 - 1/\lambda)$, to find the system of equations for $u_{1,1}$, $u_{2,1}$ and $u_{3,1}$.

Step**Step 1: Compute α and β**

With $\lambda = 0.01$:

$$\alpha = 2 \left(1 + \frac{1}{\lambda} \right) = 2 \left(1 + \frac{1}{0.01} \right) = 2(1 + 100) = \boxed{202}$$

$$\beta = 2 \left(1 - \frac{1}{\lambda} \right) = 2(1 - 100) = 2(-99) = \boxed{-198}$$

Note: On the RHS of the difference equation, we have $-\beta u_{i,j}$, which becomes $-(-198)u_{i,j} = +198u_{i,j}$.

Step**Step 2: Initial Conditions**

At $t = 0$, $u(x, 0) = \sin(\pi x)$ with grid points $x_i = i \cdot h = 0.5i$:

$$u_{0,0} = \sin(0) = 0$$

$$u_{1,0} = \sin(0.5\pi) = 1$$

$$u_{2,0} = \sin(\pi) = 0$$

$$u_{3,0} = \sin(1.5\pi) = -1$$

$$u_{4,0} = \sin(2\pi) = 0$$

Boundary conditions: $u_{0,j} = 0$ and $u_{4,j} = 0$ for all j .

Step**Step 3: Write Equations for $j = 0$ (First Time Step)**

The Crank-Nicholson equation becomes:

$$-u_{i-1,j+1} + 202u_{i,j+1} - u_{i+1,j+1} = u_{i+1,j} + 198u_{i,j} + u_{i-1,j}$$

For $i = 1$:

$$\begin{aligned} -u_{0,1} + 202u_{1,1} - u_{2,1} &= u_{2,0} + 198u_{1,0} + u_{0,0} \\ -0 + 202u_{1,1} - u_{2,1} &= 0 + 198(1) + 0 \\ \Rightarrow 202u_{1,1} - u_{2,1} &= 198 \end{aligned}$$

For $i = 2$:

$$\begin{aligned} -u_{1,1} + 202u_{2,1} - u_{3,1} &= u_{3,0} + 198u_{2,0} + u_{1,0} \\ -u_{1,1} + 202u_{2,1} - u_{3,1} &= -1 + 0 + 1 \\ \Rightarrow -u_{1,1} + 202u_{2,1} - u_{3,1} &= 0 \end{aligned}$$

For $i = 3$:

$$\begin{aligned} -u_{2,1} + 202u_{3,1} - u_{4,1} &= u_{4,0} + 198u_{3,0} + u_{2,0} \\ -u_{2,1} + 202u_{3,1} - 0 &= 0 + 198(-1) + 0 \\ \Rightarrow -u_{2,1} + 202u_{3,1} &= -198 \end{aligned}$$

Results**System of Equations:**

$$\begin{aligned} 202u_{1,1} - u_{2,1} &= 198 \\ -u_{1,1} + 202u_{2,1} - u_{3,1} &= 0 \\ -u_{2,1} + 202u_{3,1} &= -198 \end{aligned}$$

In matrix form:

$$\begin{pmatrix} 202 & -1 & 0 \\ -1 & 202 & -1 \\ 0 & -1 & 202 \end{pmatrix} \begin{pmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \end{pmatrix} = \begin{pmatrix} 198 \\ 0 \\ -198 \end{pmatrix}$$

This is a **tridiagonal system** which can be solved efficiently using the Thomas algorithm.

Exercise 15.2: Problem 10 — Part (c)

(c) Solve the system of three equations without the aid of a computer program.

Step**Using Symmetry**

From the initial condition $u(x, 0) = \sin(\pi x)$, we observe the antisymmetry property:

$$\sin(\pi(2 - x)) = \sin(2\pi - \pi x) = -\sin(\pi x)$$

This antisymmetry about $x = 1$ is preserved by the heat equation. Therefore:

- $u_{1,j} = -u_{3,j}$ for all j (antisymmetric points)
- $u_{2,j} = 0$ for all j (center point on axis of antisymmetry)

Step**Solving Using Symmetry**

Let $u_{1,1} = -u_{3,1}$ and $u_{2,1} = 0$.

From equation (1): $202u_{1,1} - u_{2,1} = 198$

$$202u_{1,1} - 0 = 198 \implies u_{1,1} = \frac{198}{202} = \frac{99}{101}$$

Therefore:

$$u_{3,1} = -u_{1,1} = -\frac{99}{101}$$

Step**Verification**

Check equation (2): $-u_{1,1} + 202u_{2,1} - u_{3,1} = 0$

$$-\frac{99}{101} + 202(0) - \left(-\frac{99}{101}\right) = -\frac{99}{101} + \frac{99}{101} = 0 \quad \checkmark$$

Check equation (3): $-u_{2,1} + 202u_{3,1} = -198$

$$-0 + 202\left(-\frac{99}{101}\right) = -\frac{202 \times 99}{101} = -\frac{2 \times 99}{1} = -198 \quad \checkmark$$

Results

Solutions:

$$u_{1,1} = \frac{99}{101} \approx [0.9802]$$

$$u_{2,1} = [0]$$

$$u_{3,1} = -\frac{99}{101} \approx [-0.9802]$$

Python Output

FULL TIME EVOLUTION (Crank-Nicholson)

Solution at selected times (coarse grid h=0.5) :

t	x=0.00	x=0.50	x=1.00	x=1.50	x=2.00
0.000	0.0000	1.0000	0.0000	-1.0000	-0.0000
0.010	0.0000	0.9802	0.0000	-0.9802	0.0000
0.050	0.0000	0.9048	0.0000	-0.9048	0.0000
0.100	0.0000	0.8187	0.0000	-0.8187	0.0000
0.150	0.0000	0.7408	0.0000	-0.7408	0.0000
0.200	0.0000	0.6703	0.0000	-0.6703	0.0000
0.300	0.0000	0.5488	0.0000	-0.5488	0.0000

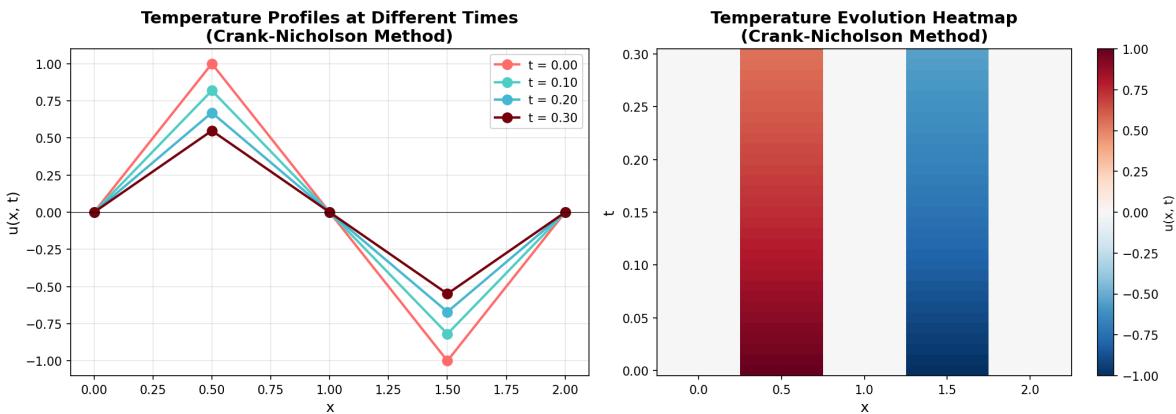


Figure 2: Left: Temperature profiles at different times showing antisymmetric decay. Right: Heatmap of the full space-time evolution.

Exercise 15.2: Problem 12 (25 pts)

Use the difference equation

$$u_{i,j+1} = \lambda u_{i+1,j} + (1 - 2\lambda)u_{i,j} + \lambda u_{i-1,j}$$

to approximate the solution of the boundary-value problem

$$\begin{aligned}\frac{\partial^2 u}{\partial x^2} &= \frac{\partial u}{\partial t}, \quad 0 < x < 1, \quad 0 < t < 1 \\ u(0, t) &= 0, \quad u(1, t) = 0, \quad 0 \leq t \leq 1 \\ u(x, 0) &= x(1 - x), \quad 0 \leq x \leq 1\end{aligned}$$

Step

Explicit FTCS Method

The **Forward-Time, Centered-Space (FTCS)** method is the simplest finite difference scheme for the heat equation:

- **Forward in time:** Uses $u_{i,j+1} - u_{i,j}$ for $\partial u / \partial t$
- **Centered in space:** Uses $u_{i+1,j} - 2u_{i,j} + u_{i-1,j}$ for $\partial^2 u / \partial x^2$
- **Explicit:** New values computed directly from old values (no linear system to solve)
- **First-order in time:** $O(\Delta t)$ temporal accuracy
- **Second-order in space:** $O(h^2)$ spatial accuracy

Step

Stability Analysis

The explicit method is **conditionally stable**. Using von Neumann stability analysis:

$$\text{Stability requires: } \lambda = \frac{k\Delta t}{h^2} \leq \frac{1}{2}$$

Physical interpretation: Information cannot propagate faster than one grid cell per time step.

Condition	Behavior	Physical Meaning
$\lambda < 0.5$	Stable, smooth	Normal diffusion
$\lambda = 0.5$	Stable, exact averaging	Pure neighbor averaging
$\lambda > 0.5$	Unstable, oscillations	Unphysical amplification

Step**Comparison: Explicit vs Crank-Nicholson**

Property	Explicit (FTCS)	Crank-Nicholson
Time accuracy	$O(\Delta t)$	$O(\Delta t^2)$
Space accuracy	$O(h^2)$	$O(h^2)$
Stability	$\lambda \leq 0.5$ required	Unconditionally stable
Computation	Direct formula	Solve tridiagonal system
Implementation	Very simple	Moderate complexity
Best for	Quick prototyping	Production simulations

Use $n = 5$ and $m = 50$. Solve this using Python.

Step**Step 1: Grid Parameters**

Spatial discretization: With $n = 5$ divisions over $[0, 1]$:

$$h = \frac{1}{n} = \frac{1}{5} = 0.2$$

Temporal discretization: With $m = 50$ divisions over $[0, 1]$:

$$\Delta t = \frac{1}{m} = \frac{1}{50} = 0.02$$

Stability parameter: For the heat equation $u_t = u_{xx}$ (diffusivity $k = 1$):

$$\lambda = \frac{k\Delta t}{h^2} = \frac{1 \times 0.02}{(0.2)^2} = \frac{0.02}{0.04} = \boxed{0.5}$$

Stability: The explicit method is stable when $\lambda \leq 0.5$. We are exactly at the stability limit!

Step**Step 2: Initial and Boundary Conditions**

Initial condition at $t = 0$: $u(x, 0) = x(1 - x)$

i	0	1	2	3	4	5
x_i	0	0.2	0.4	0.6	0.8	1.0
$u_{i,0}$	0	0.16	0.24	0.24	0.16	0

Boundary conditions: $u_{0,j} = 0$ and $u_{5,j} = 0$ for all j .

Step

Step 3: Explicit Update Formula

With $\lambda = 0.5$, the difference equation simplifies dramatically:

$$u_{i,j+1} = 0.5 \cdot u_{i+1,j} + (1 - 1) \cdot u_{i,j} + 0.5 \cdot u_{i-1,j}$$

$$u_{i,j+1} = \frac{1}{2}(u_{i+1,j} + u_{i-1,j})$$

This is simply the **average of the neighboring values!** The center point's current value has zero weight.

Python Implementation

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # Parameters
5 L = 1.0; T_final = 1.0
6 n = 5; m = 50
7 h = L / n          # h = 0.2
8 dt = T_final / m # dt = 0.02
9 lam = dt / h**2   # lambda = 0.5
10
11 # Grid
12 x = np.linspace(0, L, n+1)
13 t = np.linspace(0, T_final, m+1)
14
15 # Initialize solution
16 u = np.zeros((m+1, n+1))
17 u[0, :] = x * (1 - x) # Initial condition
18
19 # Time stepping (explicit method)
20 for j in range(m):
21     for i in range(1, n):
22         u[j+1, i] = lam*u[j, i+1] + (1-2*lam)*u[j, i] + lam*u[j, i-1]
23
24 # Print results
25 print("Solution at selected times:")
26 for j in [0, 10, 20, 30, 40, 50]:
27     print(f"t={t[j]:.2f}: ", [f"{u[j, i]:.4f}" for i in range(n+1)])

```

Python Output

```
STEP 4: First Few Time Steps (Manual)
-----
j = 0 -> j = 1 (t = 0.00 -> t = 0.02):
u_{1,1} = 0.5 x (0.2400 + 0.0000) = 0.1200
u_{2,1} = 0.5 x (0.2400 + 0.1600) = 0.2000
u_{3,1} = 0.5 x (0.1600 + 0.2400) = 0.2000
u_{4,1} = 0.5 x (0.0000 + 0.2400) = 0.1200

j = 1 -> j = 2 (t = 0.02 -> t = 0.04):
u_{1,2} = 0.5 x (0.2000 + 0.0000) = 0.1000
u_{2,2} = 0.5 x (0.2000 + 0.1200) = 0.1600
u_{3,2} = 0.5 x (0.1200 + 0.2000) = 0.1600
u_{4,2} = 0.5 x (0.0000 + 0.2000) = 0.1000

NUMERICAL RESULTS
=====
t | x=0.0 | x=0.2 | x=0.4 | x=0.6 | x=0.8 | x=1.0 |
-----
0.00 | 0.0000 | 0.1600 | 0.2400 | 0.2400 | 0.1600 | 0.0000 |
0.20 | 0.0000 | 0.0182 | 0.0295 | 0.0295 | 0.0182 | 0.0000 |
0.40 | 0.0000 | 0.0022 | 0.0035 | 0.0035 | 0.0022 | 0.0000 |
0.60 | 0.0000 | 0.0003 | 0.0004 | 0.0004 | 0.0003 | 0.0000 |
0.80 | 0.0000 | 0.0000 | 0.0001 | 0.0001 | 0.0000 | 0.0000 |
1.00 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
```

Results

Observations:

1. **Symmetry:** The solution maintains symmetry about $x = 0.5$ throughout, reflecting the symmetric initial condition.
2. **Decay:** The temperature decays exponentially toward zero (the boundary values).
3. **Maximum:** The peak temperature is always at the center ($x = 0.5$).
4. **Stability:** With $\lambda = 0.5$ at the stability limit, the method remains stable and provides smooth results.

Energy Decay:

t	Total Energy $\int_0^1 u^2 dx$
0.00	0.033280
0.20	0.000480
0.40	0.000007

The energy decays by approximately three orders of magnitude every 0.2 time units.

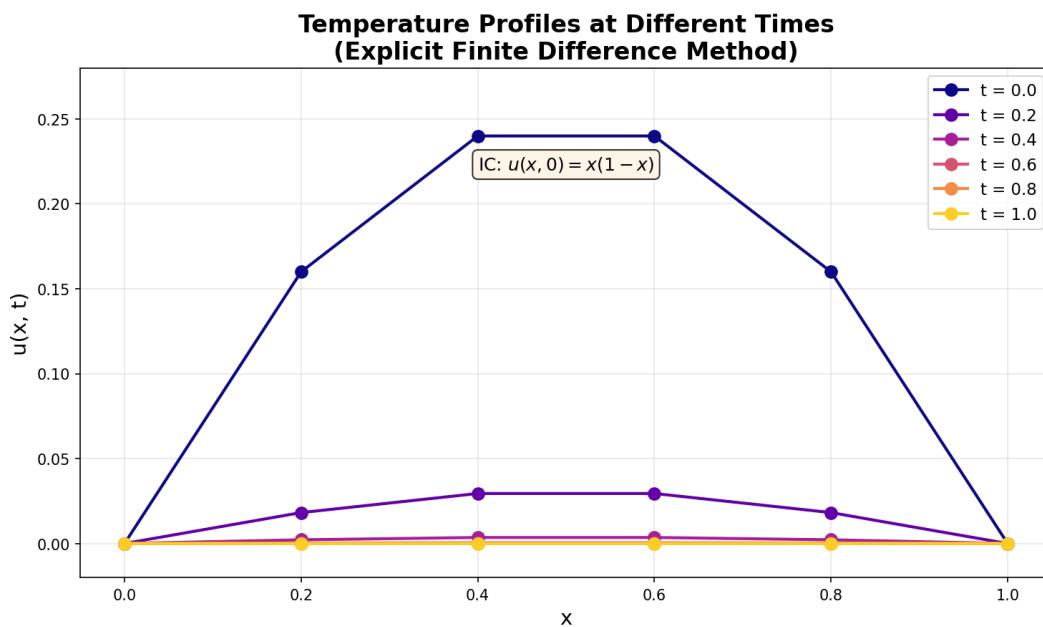


Figure 3: Temperature profiles at different times showing exponential decay from the initial parabolic distribution $u(x, 0) = x(1 - x)$.

Heat Equation Solution Surface
 $u_{xx} = u_t$ with $u(x,0) = x(1-x)$

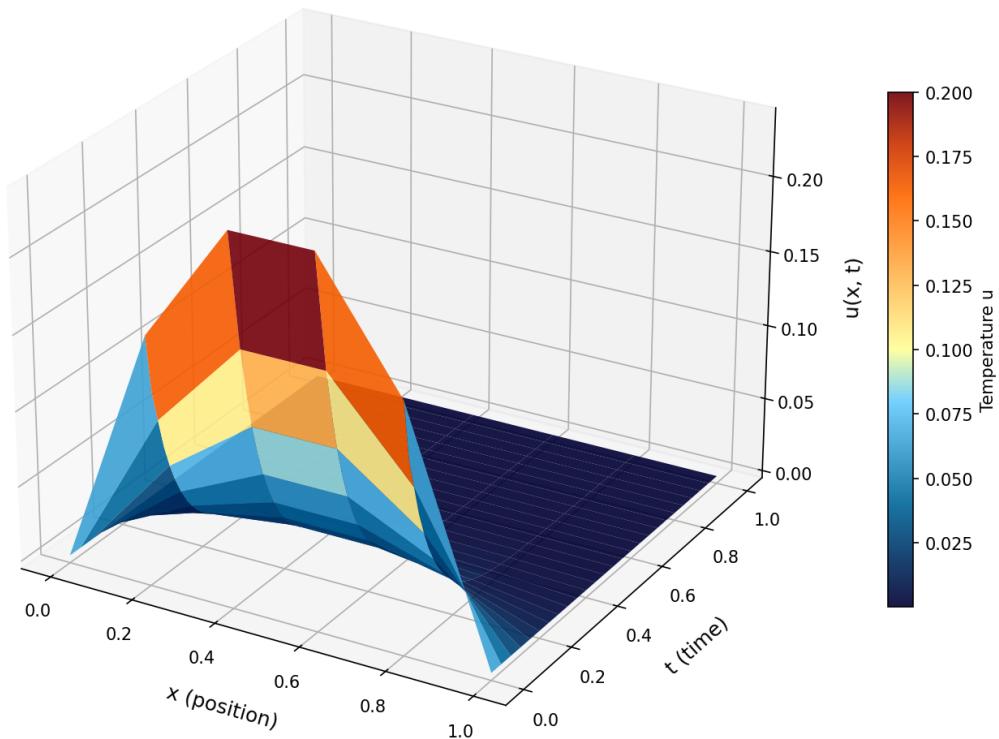


Figure 4: 3D surface plot of the heat equation solution $u(x, t)$ showing the complete spatiotemporal evolution.

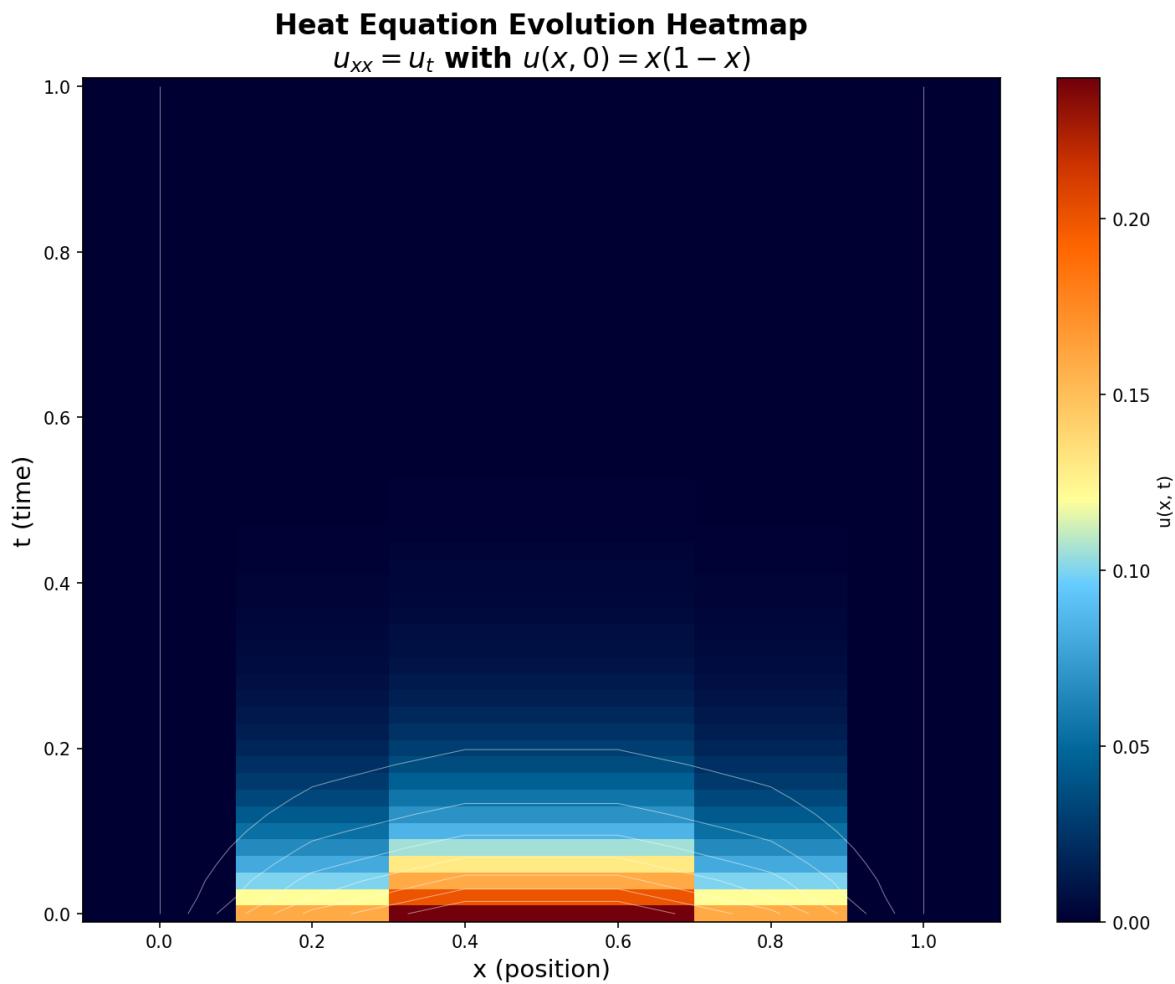


Figure 5: Heatmap representation of the temperature evolution. The rapid decay from the initial condition is clearly visible.

Python Output

```

ANALYTICAL SOLUTION COMPARISON
-----
Exact solution (Fourier series):
u(x,t) = Sum B_n sin(n*pi*x) exp(-n^2*pi^2*t)

where B_n = (2/L) integral_0^1 x(1-x) sin(n*pi*x) dx

For odd n: B_n = 8/(n^3*pi^3)
For even n: B_n = 0

Comparison at x = 0.5:
t      Numerical      Analytical      Error
-----
0.00    0.240000    0.250000    1.00e-02
0.20    0.029453    0.035841    6.39e-03
0.40    0.003538    0.004979    1.44e-03
0.60    0.000425    0.000692    2.67e-04
0.80    0.000051    0.000096    4.50e-05
1.00    0.000006    0.000013    7.22e-06

```

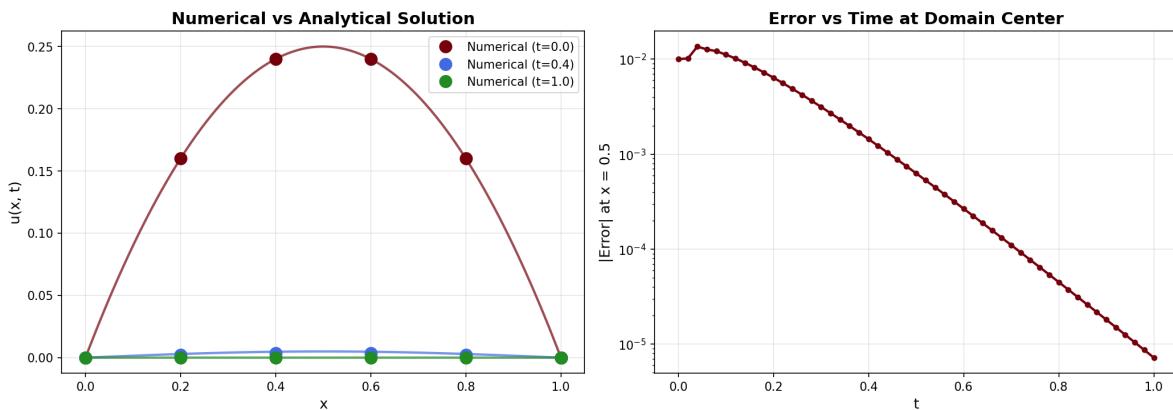


Figure 6: Left: Comparison of numerical (markers) and analytical (lines) solutions at different times. Right: Error decay at the center point $x = 0.5$.