ECE 445

SENIOR DESIGN LABORATORY

PROJECT PROPOSAL

Display of Ordinary Differential Equation

Team #5

KEJIA HU
(kejiahu2@illinois.edu)
ZHUOHAO LI
(zhuohao5@illinois.edu)
QIANHE YE
(qianhey2@illinois.edu)
QIRONG XIA
(qirongx2@illinois.edu)

Advisor: Prof. Pavel Loskot

March 6, 2024

Contents

1 Introduction

1.1 Problem

In many scientific and engineering fields, understanding the behavior of complex systems often requires analyzing interactions between multiple variables over time and space. Three-dimensional visualization offers superior expressiveness, effectiveness, and appropriateness in representing data information. [1] However, visualizing such systems in traditional 2D formats can be limiting, especially when dealing with dynamic phenomena or spatial dimensions. To overcome this limitation, there is a growing need for innovative visualization tools that can represent these systems in three dimensions, providing a more intuitive understanding of their behavior. Additionally, for problems involving spatial dimensions, a 3D visualization can illustrate how variables change not just over time but also across different points in space.

Modern software capabilities include sophisticated 3D visualization tools that not only enhance data comprehension but also facilitate interactive exploration and analysis. [2] However, screen-based 3D visualization is confined to the dimensions of the display device, limiting the viewer's ability to perceive objects from various angles and distances as they would in real-life settings. It may also struggle to accurately convey the scale of objects, leading to misconceptions or misinterpretations of their size and proportions. Thus, by developing a real-time 3D visualization platform, researchers can observe how variables evolve over time and space from different angles.

The primary objective of this project is to design and implement a portable and user-friendly 3D real-time visualization system capable of accurately representing the solution of time-varying 2D differential equations. The surface will be colored by projecting an image from the top. An example solution of the differential equation is illustrated in Appendix B. Our system will provide researchers with a comprehensive tool for visualizing and analyzing complex systems' behavior. Moreover, such a tool could extend its application from mathematics to various disciplines, including physics, biology, engineering, and environmental science.

1.2 Solution

To effectively address the challenges outlined in the problem background, our solution will also incorporate advanced features for enhanced user engagement and analytical capabilities. Our solution aims to create a 3D real-time visualization of a time-varying 2D Differential Equation function. This visualization platform will dynamically represent the changing behavior of the function over time. Additionally, we will project the color to the surface from either the top or below, adding another dimension of visual interpretation. Our design has four subsystems. The User Interface subsystem will get the Differential Equation from the user and transmit the data to the Control system. The control subsystem will get the Differential Equation solution from the User Interface subsystem and then convert the information to languages that the mechanical devices can understand. Our Mechanical system adjusts the height of the stick based on the signal transmitted from the Control subsystem. Additionally, this module will use a canvas to cover the top of the grid-like sticks to create a smooth visualization surface for displaying the solution. Finally, we will design a Coloring Subsystem which will project images to the canvas either from the top or below.

1.3 Visual Aid

Based on the solution we propose, a pictorial representation of our project is shown in Fig.1.

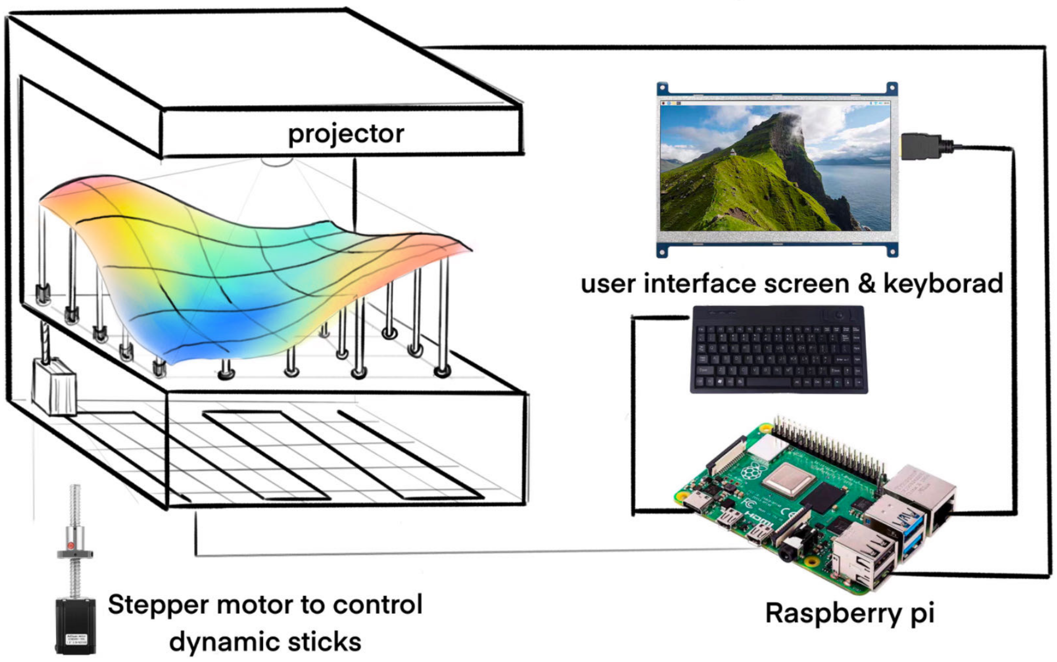

projector
user interface screen & keyborad
Stepper motor to control dynamic sticks
Raspberry pi

Figure 1: A pictorial representation of the device

1.4 High-Level Requirement List

- The device should provide a user-friendly interface that enables users to input differential equations easily and view the status of computations and solutions on the screen clearly.
- The 3D visualization system must update dynamically and respond to changes in the differential equation solution in real time, ensuring that users can observe the behavior of the system as it evolves over time without significant delays or lags.
- The entire system should be robust and reliable, capable of maintaining stable operation over extended periods. It should withstand environmental factors and variations in input conditions without compromising the accuracy or functionality of the visualization.

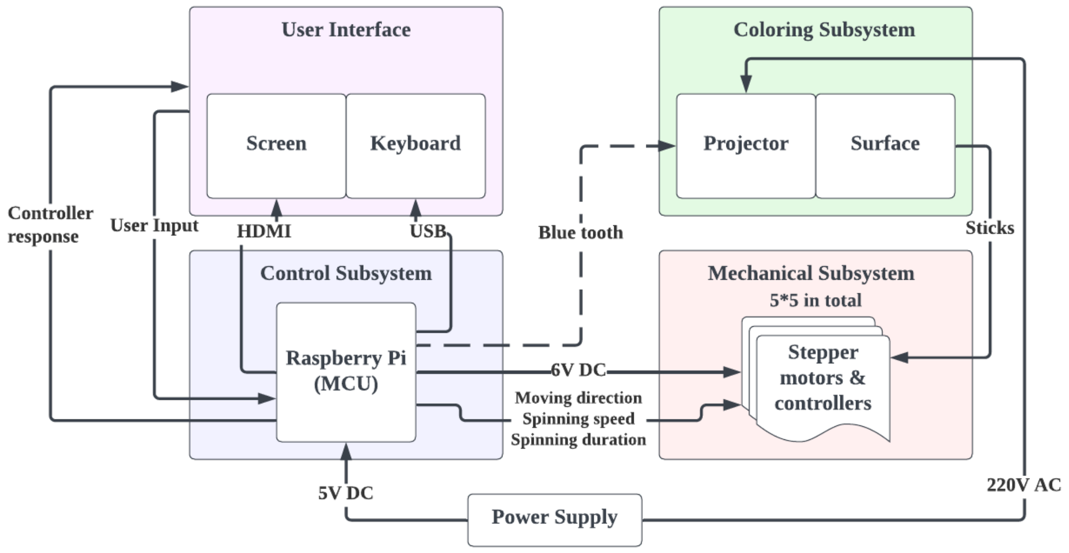


Figure 2: A pictorial representation of the device

2 Design Requirements

2.1 Block Diagram

The block diagram of the device is shown in Fig. 2.

2.2 Subsystem Overview

In this section, we give overviews of each subsystem, discussing the responsibilities and functions of them.

2.2.1 User Interface Subsystem

The User Interface (UI) Subsystem serves as the interface between the user and the visualization system. It receives input Differential Equations from the user, calculates the solution for them, and transmits the data to the Control Subsystem for visualization. Firstly, we need a Graphical User Interface (GUI) which contains components such as text input fields, buttons, sliders, and a keyboard for user interaction, allowing users to input Differential Equations through a graphical interface. Secondly, once the user inputs the Differential Equations, the module utilizes Raspberry Pi to compute the solution. Thirdly, after computing the solution, the Raspberry Pi transmits the data, including the time-varying solutions of the Differential Equations variables, to the Control Subsystem for visualization. Lastly, the Raspberry Pi will provide feedback on the screen regarding the status of the computation, such as progress indicators or error messages in case of invalid input or computation failures.

2.2.2 Control Subsystem

The Control Subsystem is in charge of the stick height adjustments. It translates solutions of Differential Equations received from the User Interface Subsystem into actionable control commands for the Mechanical Subsystem. This involves mapping the mathematical solutions to physical actions, such as the movement of the sticks. This subsystem includes the hardware setup, software development, and the integration of the two. For hardware setup, we will use Raspberry Pi as the central controller for the system. It will be connected to motors which will adjust the height of the stick and sensor which will give feedback on the height information to the Raspberry Pi. For software development, we will use Python to control the actuators based on received data and feedback from sensors. The Control Subsystem manages the accurate adjustment of stick height, serving as a vital connection between user commands, mathematical calculations, and actual movement in the system.

2.2.3 Mechanical Subsystem

The mechanical subsystem consists of rods that can move up and down dynamically in accordance with the displayed solution. A layer of rubber foil is secured to the top of each rod, ensuring the visualization remains seamless. Inputs from the control subsystem dictate the movements of this subsystem, which then executes these commands. Additionally, it provides feedback to the control subsystem for adjustments and fine-tuning. The system is designed to adhere precisely to the commands from the control system, aiming for high accuracy and precision in movements. Moreover, it's built to be sturdy, effectively reducing minor vibrations and environmental noise.

2.2.4 Coloring Subsystem

The Coloring Subsystem is on the top of the device and it gets the solution of the Differential Equations from the Raspberry Pi. Given the height data, the subsystem employs logic to assign colors based on the height of the sticks. For example, the highest sticks could be assigned to the color red, while the lowest points are assigned to the color blue. As the height of the sticks changes, the subsystem must dynamically adjust the projections in real time to reflect these changes. This is achieved by having a tiny projector connected to the Raspberry Pi, taking the time-varying solutions of the Differential Equations, and coloring the surface with the time and height of the sticks changing. We plan to implement this feature at the end since it needs to synchronize accurately with the mechanical part. The implementation of this subsystem will greatly enhance the 3D visualization and enable observers to easily discern patterns within the solution, leading to more informed insights.

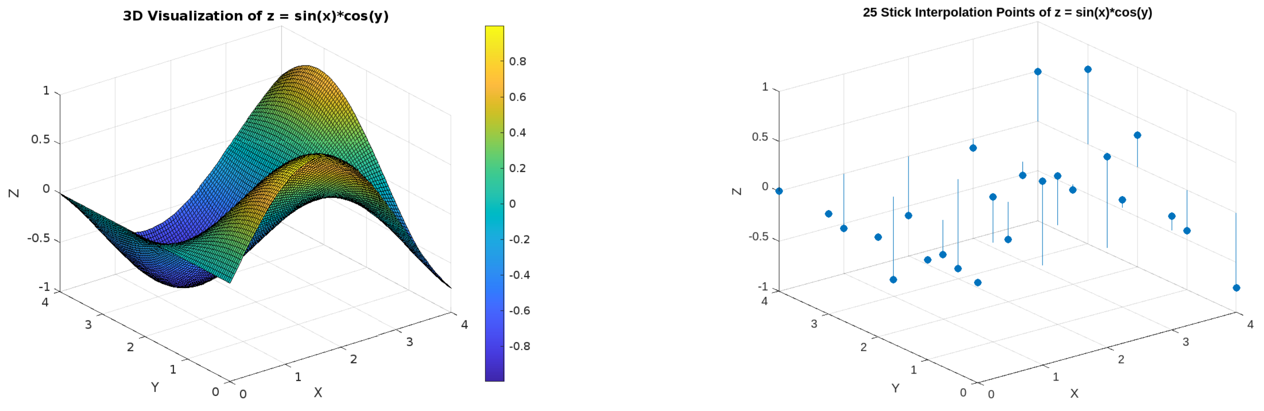
2.3 Subsystem Requirements

2.3.1 User Interface Subsystem

1. The GUI should be well-designed that includes components like text input fields, buttons, sliders for user interaction. This interface should be intuitive and user-friendly to allow users to easily input Differential Equations.
2. The subsystem should be designed for reliability, including robust error handling and validation of user inputs to prevent crashes and ensure accurate computation results.

2.3.2 Control Subsystem

1. The Raspberry Pi needs to connect to 25 stepper motors, each of them given a stable 6V voltage, allowing it to achieve 500 RPM (Revolution Per Minute) rotation rate. A Matlab simulation of 3D visualization and 25-point interpolation is shown in Fig 3.



(a) 3D visualization (b) 25 points interpolation

Figure 3: Matlab simulation

2. The time interval of different height signals given to the stepper motors should not exceed 1 second.
3. The control system is responsible for synchronizing the surface coloring and the movement of the sticks. We can leverage parallel computing in the two cores on the Raspberry Pi. That said, every time the microcontroller sends the signals to the sticks, it should also send the generated coloring image to the projector simultaneously.

2.3.3 Mechanical Subsystem

1. The stick and the connection of which with the motor must be robust to support the canvas without noticeable wobbling during and after the motion.

2. The motors should act accurately and nearly identically according to the signal input from the control subsystem.
3. The canvas needs to be elastic and resistant to tearing, while reflecting the position of different sticks accurately.

2.3.4 Coloring Subsystem

1. The coloring subsystem consists of a tiny projector connected to the Raspberry Pi. The changing frequency of the coloring surface should be consistent with the movement of the surface, which is supposed to be more than 1Hz.
2. The projected images should be able to cover just about all 0.5m*0.5m areas of the surface.
3. The color coding needs to be accurate and timely based on the user input.

2.4 Tolerance Analysis

2.4.1 Timely responses of the motor

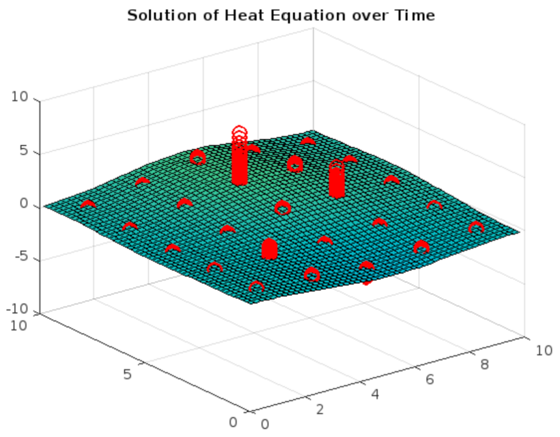
Each dynamic stick in our project is powered by a motor, making the motor's responsiveness to the microcontroller a crucial factor in the project's visual effectiveness. We employed stepper motor linear actuators for the precise vertical movement of the sticks. In Appendix B, we explored the detailed simulation of a heat wave function solver. The sticks are designed to move vertically within a 16 cm range, allowing us to adjust the time-varying function to a visually optimal spectrum. By identifying the highest and the lowest values of the function, we calculate the median to serve as the horizon for each display.

Given that our control over the sticks' movement is limited to directing their motion through the motor's positive and negation poles, and lacking a viable method to change their speed, we simulated the motion of the sticks in any direction as uniform linear motion. However, according to the moving trajectory of sticks shown in Figure 4, the stick's moving distance varies across different frames given the same time interval. Consequently, we propose two major constraints for determining the frame rate of the device:

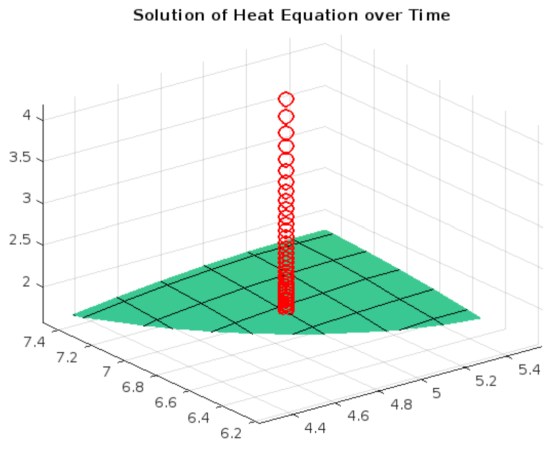
1. The maximal stick's achievable moving velocity.
2. The maximal time interval that the microcontroller dispatches the motion commands to the motor.

A reasonable breakdown of the working period of the motor consists of (1) the screw performing rotation, and (2) waiting for the next signal sent from the microcontroller.

The RPM of the stepper motor shaft is directly related to the linear speed of the external nut (and consequently the stick attached to it) in a proportional manner. The screw lead is the distance the nut moves parallel to the screw axis when the screw makes one complete



(a) The trajectories of 25 sticks



(b) The trajectory of a single stick

Figure 4: The trajectories of sticks

revolution, with unit mm/rev. The formula to calculate the linear speed V_{linear} (in unit mm/min) based on the motor's RPM and the screw's lead L is:

$$V_{linear} = RPM \cdot L \tag{1}$$

Given that the screw lead is 2mm/rev, the linear speed of the stick is:

$$V_{linear} = 500RPM \cdot 2mm/rev = 1000mm/min \approx 1.667cm/s$$

Therefore, given that the maximum travel distance per work cycle is set to 1.5 cm, the minimum duration of a work cycle must be $t_{motor} = \frac{1.5cm}{1.667cm/s} = 0.899s$. Since the time taken by the microcontroller to transmit signals to the motor is negligible relative to the movement latency of the actuators, a time interval of 1 second is sufficient to guarantee the device's full operational capability.

2.4.2 Synchronization between the sticks and the projection

The synchronization between the movement of the sticks in the Mechanical Subsystem and the projection of colors in the Coloring Subsystem is of great importance to 3D visualization. Inconsistency or delay in this synchronization could lead to inaccurate and confusing visualization of the differential equation solution. Therefore, we need to ensure that the movement of the sticks and the projection of colors occur within an acceptable time. Assume that the mechanical movement of the sticks takes t_m seconds to complete, and the coloring subsystem requires t_c seconds to project the colors onto the surface. Through testing and adjusting the RPM of the sticks, we try to minimize the time difference $|t_m - t_c|$ to less than 0.5 seconds.

2.4.3 Accuracy of the numerical solutions

Many of the Differential Equations do not have an explicit solution. Hence, we need to use a numerical method to get an approximate solution. To ensure the solution solved by a numerical method is similar to the true solution, we need to perform a tolerance analysis on the difference between the two.

Let's denote:

- $u(x_i, y_j, t_k)$ as the true solution of the Differential Equation at each grid point.
- $u_{\text{num}}(x_i, y_j, t_k)$ as the solution obtained numerically in Python at each grid point.

We can measure the difference between the true solution and the numerical solution using a suitable metric such as the L2-norm. The error at each grid point (x_i, y_j, t_k) can be calculated using the absolute difference between the true solution and the numerical solution:

$$e_{i,j,k} = |u(x_i, y_j, t_k) - u_{\text{num}}(x_i, y_j, t_k)| \tag{2}$$

Then, the L^2 norm of the error over the entire grid can be approximated by summing the squared errors at each grid point and taking the square root:

$$\|e\|_2 \approx \sqrt{\sum_{i,j,k} e_{i,j,k}^2} \tag{3}$$

where the sum is taken over all grid points (x_i, y_j, t_k).

When we implement the Differential Equation solving algorithm, we need to calculate the error difference based on some Differential Equations that we know the true solutions. We need to ensure the $\|e\|_2 \leq T_{\text{tol}}$, where $T_{\text{tol}} = 10^{-2}$.

3 Ethics & Safety

In the development and use of any product, safety should always be the top priority. To ensure "the safety, health, and welfare of the public" as outlined in IEEE's ethical guidelines[3], we should strictly adhere to relevant regulations throughout the research and development phases, as well as inform users about the proper usage and to communicate the potential risks with misuse.

Furthermore, to be forthright and grounded in reality when making claims or estimates based on the data at hand[3], we should make it clear that the visualized differential equation solution is an approximation. Despite our efforts to closely simulate real-world scenarios, users should understand that it is not a substitute for real solutions.

The essence of design lies in simplifying life and enhancing work efficiency. It's crucial for society to aim for respect, inclusivity, fairness, and balance, guaranteeing that everyone can access the necessary tools and resources for a rewarding life, free from discrimination related to race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or expression[3]. At our core, we are committed to enabling individuals to engage with differential equations in a more direct way while learning.

References

[1] J. Wood, S. Kirschenbauer, J. Döllner, A. Lopes, and L. Bodum, “Using 3d in visualization,” in *Exploring geovisualization*, Elsevier, 2005, pp. 293–312.

[2] A. R. Teyseyre and M. R. Campo, “An overview of 3d software visualization,” *IEEE transactions on visualization and computer graphics*, vol. 15, no. 1, pp. 87–105, 2008.

[3] IEEE. “Ieee code of ethics.” (2020), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 03/05/2024).

[4] Matlab. “Simple heat equation solver.” 2024-03-03. (2023), [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/59916-simple-heat-equation-solver.

Appendix A Define and Solve 2D Time-Dependent PDE

Below is a pseudocode of how to obtain a partial differential equation (PDE) from the user and solve it using numerical methods.

Algorithm 1 Solving 2D Time-Dependent PDE $z(x, y, t)$

```
1: **Input:** User-provided PDE
2: **Prompt User Input:**
3: $pde\_input \leftarrow$ Get input from the user in the format $'a\frac{\partial^2 z}{\partial x^2} + b\frac{\partial^2 z}{\partial y^2} + c\frac{\partial z}{\partial t} = f(x, y, t)'$
4: **Parse Input:**
5: Parse $pde\_input$ to extract coefficients $a$, $b$, $c$ and the function $f(x, y, t)$
6: **Spatial and Temporal Domain:**
7: Define spatial and temporal domain by generating arrays $x\_values$, $y\_values$, $t\_values$
   using desired parameters
8: **Initialize Solution Array:**
9: Initialize array $z\_values$ of size $(\text{num\_points\_x}, \text{num\_points\_y}, \text{num\_points\_t})$ with ze-
   ros
10: **Boundary and Initial Conditions:**
11: Set initial conditions for $z(x, y, t)$ at $t = 0$ using input initial condition function
12: **Time-stepping Loop:**
13: **for** $n$ **in** $[0, \text{num\_points\_t} - 1]$ **do**
14:     **for** $i$ **in** $[1, \text{num\_points\_x} - 1]$ **do**
15:         **for** $j$ **in** $[1, \text{num\_points\_y} - 1]$ **do**
16:             **Implement Numerical Scheme:**
17:             $z\_values[i, j, n + 1] \leftarrow$ Update rule for $z(x, y, t)$ at $(x_i, y_j, t_{n+1})$
18:         **end for**
19:     **end for**
20: **end for**
21: **Output:** Solution of the PDE $z(x, y, t)$
```

Appendix B Matlab Animation of Heat Equation

2D time-dependent PDE appears a lot and has wide applications. One common PDE that exhibits time variation in two spatial dimensions is the heat equation. The heat equation describes how a quantity (such as temperature) changes over time in a given region, based on its spatial distribution and the rate at which it diffuses.

The heat equation in three dimensions (including time) is given by:

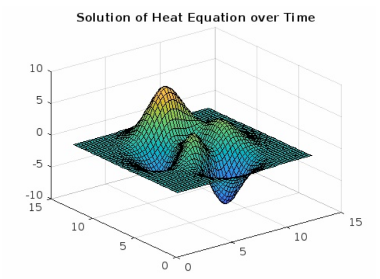
$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right)$$

Where:

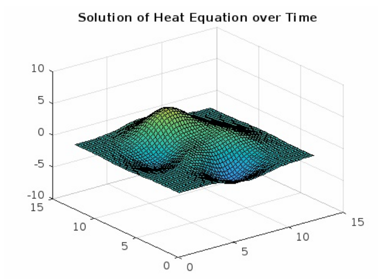
- $u(x, y, z, t)$ is the temperature (or another quantity) at position (x, y, z) and time t.
- α is the thermal diffusivity coefficient, a constant that characterizes the material's ability to conduct heat.
- $\frac{\partial u}{\partial t}$ denotes the partial derivative with respect to time.
- $\frac{\partial^2 u}{\partial x^2}$, $\frac{\partial^2 u}{\partial y^2}$, and $\frac{\partial^2 u}{\partial z^2}$ are the second partial derivatives with respect to x, y, and z respectively. These terms describe how the temperature changes along each spatial dimension.

This equation describes how the temperature $u(x, y, z, t)$ evolves over time due to the diffusion of heat in three dimensions. The spatial derivatives account for the change in temperature along the x, y, and z directions, while the time derivative describes how the temperature changes over time.

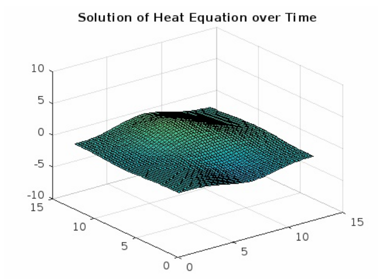
The animation depicting the solution of the heat equation in MATLAB [4], is illustrated in Fig 5.



(a) frame at t = 1



(b) frame at t = 12



(c) frame at t = 35

Figure 5: Animation of Heat Equation