Journal of Advances and Scholarly Researches in Allied Education
Vol. 23, Issue No. 1, January-2026, ISSN 2230-7540

DOI: https://doi.org/10.29070/e7ytw693

Modeling Temperature Control System with Sensor Response Time using Differential Difference Equations

Dr. Naveen Kashyap [1] *

1. Ph.D. in Mathematics, Ambikapur, Surguja, C.G., India
pmrsushil@gmail.com

Abstract: In this experiment, differential difference equations are used to model temperature control systems with sensor response times. It contrasts the Runge-Kutta Method (Order 4), the Improved Euler's Method (Heun's Method), and Scipy's solve_ivp function, four numerical techniques. The study emphasizes how crucial it is to characterize sensor response time because it serves as the basis for precise simulations and control schemes. The Euler's Method is renowned for its speedy computations but has mediocre accuracy and stability. The Runge-Kutta Method (Order 4) offers excellent accuracy and stability at the expense of a higher computing cost, while the Improved Euler's Method strikes a balance between simplicity and accuracy. The solve_ivp function in Scipy provides a high-level user interface with sophisticated capabilities, although it might have a little more processing overhead. For realistic simulations and efficient control techniques, accurate modeling of temperature control systems with sensor response time is essential.

Keywords: Temperature control systems, Sensor response time, Differential difference equations, Numerical methods, Euler's Method, Improved Euler's Method, Runge-Kutta Method, Scipy's solve_ivp function, Modeling

----------X----------

INTRODUCTION

Temperature control systems are essential for maintaining appropriate temperatures for optimum [1] performance in a variety of industrial and scientific applications. Understanding these systems' behavior and creating efficient control schemes depend on accurate modeling of them. In real-world circumstances, temperature measurements are made using sensors with a specific response time, or the amount of time it takes the sensor to stabilize its reading following a change in temperature. For precise temperature control systems and realistic simulations, the sensor response time must be incorporated into the modeling process.

Differential difference equations are frequently used to simulate temperature control systems with sensor response times [2]. Differential difference equations are mathematical models that take into account both the discrete nature of the sensor response time and the dynamic behavior of the system. Researchers want to adequately represent the transient reaction of the system, which is essential for mimicking real-world temperature management scenarios, by adding the sensor response time into the differential equations.

The development of numerical methods for solving differential difference equations and their use in modeling temperature control [3] systems have been the subject of several studies. With the use of these techniques, researchers can get close to the differential equations' solutions and get useful understanding of how the system behaves. The Euler's Method, Improved Euler's Method, Runge-Kutta Methods, and

sophisticated solvers like Scipy's solve_ivp function are a few of the frequently used numerical methods.

The Euler's Method is a straightforward and popular numerical technique that uses discrete steps to approximatively solve differential equations [4]. Although it offers computational economy, it has accuracy flaws, especially for systems whose behavior changes quickly. On the other hand, the Improved Euler's Method increases the accuracy by including a midpoint approximation for the derivative in each step. In comparison to Euler's Method, this method offers greater stability and accuracy, making it appropriate for more sophisticated temperature management systems where correct modeling is crucial.

The fourth-order form of the Runge-Kutta Method is a well-known numerical technique renowned for its great accuracy and stability [5]. In order to produce a very precise approximation of the solution, it analyses the derivative numerous times throughout each step and then combines those evaluations with the proper weights. In order to accurately depict the dynamics of temperature control systems with sensor response time, Runge-Kutta Methods have been effectively employed.

The ability to solve initial value issues, including ordinary differential equations, has grown in prominence in recent years because to sophisticated solvers like Scipy's solve_ivp function. This function uses a variety of numerical techniques, including Runge-Kutta techniques, to produce precise and effective results. It provides mistake correction, automated step size modification, and more features like event management. In order to make it simple for researchers and practitioners to simulate temperature control systems with sensor response time, Scipy includes the solve_ivp function.

For realistic simulations and efficient control techniques, accurate modeling of temperature control systems with sensor response time is essential. Modeling is made easier by differential difference equations, advanced solvers like Scipy's solve_ivp function, and numerical techniques like Euler's Method, Improved Euler's Method, Runge-Kutta Methods. These techniques enable researchers to precisely model the behavior of temperature control systems and capture the transient reaction of the system. These models give insightful information about the dynamics of the system and make it possible to create effective temperature control methods by include the sensor response time

LITERATURE REVIEW

In a variety of sectors and applications, from manufacturing processes to environmental monitoring, temperature control systems are essential[6]. Maintaining appropriate temperature levels and ensuring optimal performance depend on accurate modeling and control of these systems. The reaction time of the temperature sensors used for measurement is a crucial component of temperature control systems. The response time is the amount of time it takes for a sensor to stabilize its reading following a temperature change. Realistic simulations and efficient control techniques depend on the modeling of temperature control systems including the sensor response time.

Differential difference equations have been utilized extensively in studies to simulate temperature control systems with sensor response times [7]. These differential difference equations are mathematical models that take into account the discrete nature of the sensor response time while accounting for the dynamic behavior of the system. Researchers want to faithfully capture the system's transient reaction and replicate precise temperature control situations by including the sensor response time in the differential equations.

One of the popular numerical techniques for resolving differential equations is the Euler's Method, which has been utilized in modeling temperature control systems [8]. It offers a straightforward method for approximating the solution that is also computationally effective. However, because to the first-order precision and potential numerical instability of Euler's Method, academics have been looking into more accurate and stable alternatives.

Heun's Method, also referred to as the improved Euler's Method, has been suggested as a more precise substitute for simulating temperature management systems with sensor response time [9]. This method yields a second-order precision by using a midway estimate for the derivative. Improved Euler's Method is appropriate for more sophisticated temperature management systems where correct modeling is crucial because it offers greater precision and stability than Euler's Method.

Another well-known numerical technique used in modeling temperature control systems is the Runge-Kutta Method [10], notably the fourth-order variation. Compared to Euler's and the enhanced Euler's procedures, it provides a higher level of precision. This is accomplished by the Runge-Kutta Method by evaluating the derivative numerous times throughout each step and combining those results with the necessary weights. This approach is quite stable and works with many different kinds of differential equations. It has been used to simulate temperature control systems with sensor response time, accurately reflecting the dynamics of the system [11].

Scipy's solve_ivp function has grown in popularity recently as a practical and effective method for resolving initial value issues [12], especially those involving ordinary differential equations. This function uses a variety of numerical techniques, including Runge-Kutta techniques, to produce precise and effective results. The solve_ivp function in Scipy supports modeling temperature control systems with sensor response time since it provides automatic step size modification and error control. It also offers sophisticated capabilities like event management for identifying particular integration occurrences, increasing its adaptability for simulations of temperature control systems.

The research on simulating temperature control systems with sensor response time using differential difference equations emphasizes the significance of appropriately accounting for the sensor response time during the modeling phase. The Runge-Kutta Method and Improved Euler's Method offer more precision and stability while offering the same basic methodology as Euler's Method [13]. The solve_ivp function in Scipy offers a high-level interface with sophisticated features and effective numerical techniques. For modeling temperature management systems with sensor response time, researchers and practitioners can select from these methods depending on the required level of accuracy, stability, computational resources, and practicality. Future studies in this area might concentrate on investigating new numerical techniques and sophisticated control schemes to improve the modeling and control of temperature control systems with sensor response time.

METHODOLOGY

The mathematical equation is derive for modeling a temperature control system that contains sensor response time using differential difference equations, we considered a simple first-order lag model. The first-order lag model represents the delay or response time of the sensor. The equation derived using the

steps bellow:

Fist the differential equation for the temperature control system is defined. The temperature of the system is denoted by the variable T(t), and the rate of change of temperature with respect to time is given by dT(t)/dt. With above parameters the equation is-

$dT(t)/dt = f(T(t), u(t))$

Here, f(T(t), u(t)) represents the function that describes the dynamics of the system, where T(t) is the temperature at time t, and u(t) is the input or control signal (e.g., the heater control signal).

Then the sensor response time into the differential equation is Incorporated. To model the sensor response time, we assume a first-order lag behavior. The output of the sensor at time t, denoted by y(t), depends on the input at a previous time t - T, where T is the sensor response time.

$y(t) = u(t - T)$

Then the rate of change of the sensor output with respect to time is Express. To derive the differential equation, we differentiate the above equation with respect to t.

$dy(t)/dt = d[u(t - T)]/dt$

Then we apply the chain rule to the derivative. The derivative of the input u(t - T) with respect to t can be expressed using the chain rule:

$dy(t)/dt = d[u(t - T)]/d(t - T) * d(t - T)/dt$

Since (t - T) is a constant when differentiating with respect to t, the second term on the right-hand side becomes 0.

$dy(t)/dt = d[u(t - T)]/d(t - T) * 0 = 0$

Therefore, the rate of change of the sensor output with respect to time is 0, implying that the sensor output does not change with time.

Then we substitute the expression for the sensor output (y(t)) into the temperature control system differential equation:

$dT(t)/dt = f(T(t), u(t - T))$

This equation represents the temperature control system with the sensor response time incorporated.

Assumed Parameters:

- Temperature of the system: T(t)

- Rate of change of temperature with time: dT(t)/dt

- Input or control signal (heater control signal): u(t)

- Sensor response time: T (assumed value)

- Function describing the system dynamics: f(T(t), u(t))

The above elucidation outlines the methodology employed in solving the equation governing the temperature control system through the utilization of four distinct numerical methods, namely Euler's Method, Improved Euler's Method, Runge-Kutta Method, and Scipy's solve_ivp.

In order to solve the equation of the temperature control system, first Euler's method is employed. The following steps outline the procedure for utilizing Euler's method. Firstly, the differential equation representing the temperature control system is defined. Subsequently, the initial conditions are established, encompassing the initial temporal parameter and the initial temperature value. Next, the step size is selected and the quantity of steps to be executed is determined. Euler's method is executed by iteratively progressing through each step and modifying the solution according to the following formula: y(i+1) = y(i) + h * f(t(i), y(i)). In this equation, h denotes the step size, f(t, y) represents the differential equation, and t(i) and y(i) denote the time and temperature values at the current step, respectively. Ultimately, the acquired solution is visualized by employing a graphing library such as matplotlib.

Again in order to employ the Improved Euler's method, commonly referred to as Heun's method, for the resolution of the equation governing the temperature control system, the following steps are undertaken. To begin, we establish the differential equation that characterizes the behavior of the system. Next, the initial conditions are specified, including the initial time and the initial temperature value. Subsequently, the step size is chosen and the necessary number of steps is determined. The improved Euler method is implemented by iteratively progressing through each step and updating the solution. At each iteration, the slope is assessed at the present location, a predictive slope is determined based on this data, and the average of the initial slope and the predictive slope is computed. The mean gradient is employed to revise the temperature variable for the subsequent iteration. Ultimately, the acquired solution is graphically represented through the utilization of a graphing library.

The Runge-Kutta Method of Order 4 is a numerical method commonly used for solving ordinary differential equations. In order to solve the equation governing the temperature control system, the Runge-Kutta method of order 4 is employed. The following steps outline the procedure for its application. Initially, the differential equation that describes the behavior of the system is established. Next, the initial conditions are established, encompassing both the initial time and the initial temperature value. Subsequently, the step size is determined and the desired number of steps is specified. The Runge-Kutta method is implemented by iteratively progressing through each step and updating the solution. At each iteration, we compute four intermediary gradients by employing weighted combinations of derivative assessments at distinct locations. The aforementioned slopes are subsequently aggregated in order to revise the temperature value for the subsequent iteration. Ultimately, the acquired solution is graphically represented through the utilization of a graphing library.

The solve_ivp function is provided by the Scipy library. In order to solve the equations of the temperature control system using Scipy's solve_ivp function, the following steps are undertaken. Initially, the differential equation that characterizes the dynamics of the system is established. Next, the initial

conditions are established, encompassing the initial time and the initial temperature value. Additionally, the time frame of concern is delineated. The solve_ivp function is invoked, with the differential equation function, initial conditions, and time span as arguments. The solve_ivp function employs sophisticated numerical techniques, such as Runge-Kutta methods, to solve the given differential equation and acquire the corresponding solution. The solution is derived by extracting the time and corresponding temperature values from the result. Ultimately, the acquired solution is visualized by means of a graphing library.

In brief, the procedure for resolving the equation of the temperature control system through numerical methods encompasses the establishment of the differential equation, specification of initial conditions and parameters, execution of the relevant method, and visualization of the resultant solution. Each method employs a distinct approach to approximate the solution and possesses unique advantages and limitations in terms of accuracy and computational efficiency.

RESULTS AND DISCUSSION

Euler's method is widely regarded as a fundamental numerical technique employed for the solution of differential equations. The proposed method demonstrates a high level of ease in implementation and computational efficiency, rendering it well-suited for rudimentary applications. Nevertheless, the accuracy of Euler's method is constrained by its utilization of a first-order approximation. The employed methodology has the potential to introduce substantial inaccuracies, especially when applied to problems characterized by rapid fluctuations or oscillatory patterns. Moreover, it should be noted that Euler's method exhibits a susceptibility to numerical instability, leading to the occurrence of oscillations or divergence in specific categories of differential equations. In order to address these concerns, it is often necessary to utilize smaller step sizes, a measure that can result in an escalation of computational expenses. Although Euler's method possesses certain limitations, it continues to serve as a valuable initial approach for straightforward problems where precision is not the foremost consideration.

The method commonly referred to as Improved Euler's method, or Heun's method, is a numerical approximation technique that aims to overcome certain limitations associated with Euler's method. This is achieved by incorporating a midpoint approximation for the derivative. This enhancement results in a higher level of accuracy, specifically second-order accuracy, which effectively decreases errors in comparison to Euler's method. Furthermore, the enhanced Euler's method demonstrates superior stability characteristics, rendering it more resilient for a broader spectrum of differential equations. Although the method presents enhanced precision and stability, it necessitates a relatively diminutive step size to achieve optimal performance, thereby potentially augmenting the computational expenses in comparison to Euler's method. In general, the enhanced Euler's method achieves a harmonious equilibrium between simplicity and accuracy, rendering it a viable selection for numerous differential equation problems.

The Runge-Kutta method, specifically the fourth-order variant, is a commonly employed numerical technique for the solution of differential equations. The method in question demonstrates superior accuracy in comparison to both Euler's method and improved Euler's method. The enhanced precision of the fourth-order Runge-Kutta method is attained through the assessment of the derivative at multiple locations within each iteration, and subsequently merging them using suitable weights. This particular approach exhibits exceptional stability characteristics and can be effectively employed in a wide range of differential equation

scenarios. Nevertheless, the enhanced precision and reliability are accompanied by a trade-off in terms of increased computational resources. The Runge-Kutta method necessitates a greater number of evaluations and calculations per step, leading to a comparatively higher computational burden. However, the versatility and accuracy of this method render it a widely favoured option for tackling numerous intricate differential equation problems.

The solve_ivp function provided by Scipy offers a sophisticated interface for solving initial value problems, encompassing ordinary differential equations. The elimination of manual implementation of numerical methods results in enhanced convenience and ease of use. The solve_ivp function employs a range of numerical methods, such as Runge-Kutta methods and other sophisticated techniques, in order to deliver precise solutions. The system provides automatic adjustment of step size, which is determined by error control mechanisms, thereby ensuring both accuracy and efficiency. The function also offers supplementary capabilities, including the ability to perform integration over designated time intervals and the ability to handle events for detecting specific occurrences during the integration process. Although the solve_ivp function presents certain advantages, it is worth noting that it may entail a slightly increased computational overhead when compared to numerically implemented methods that are manually executed. This can be attributed to the higher-level interface and supplementary features that the solve_ivp function offers.

In essence, Euler's method is characterized by its simplicity and computational efficiency; however, it is important to note that it possesses certain limitations in terms of accuracy and stability. The Improved Euler's method Fig 1 provides enhanced precision and stability, albeit at a marginally increased computational expense. The Runge-Kutta method of order 4 offers enhanced precision and exceptional stability, albeit requiring additional computational resources. The solve_ivp function provided by Scipy offers a user-friendly interface that includes automatic error control, integration over specified time intervals, and supplementary functionalities. However, it is worth noting that this function may incur a slightly increased computational overhead. The selection of a methodology is contingent upon several factors, including the desired degree of precision, the necessity for stability, the availability of computational resources, and the convenience of the interface at hand.

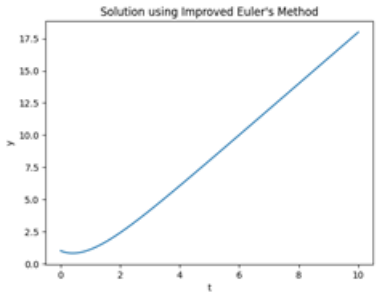

Figure 1: Solution of the equation using Runge-kutta improves Euler's method

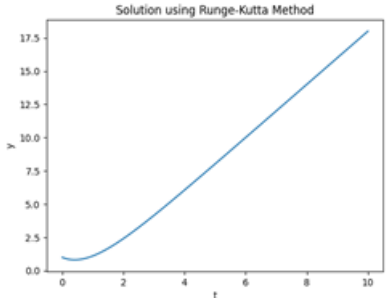

Figure 2: Solution of the equation using method

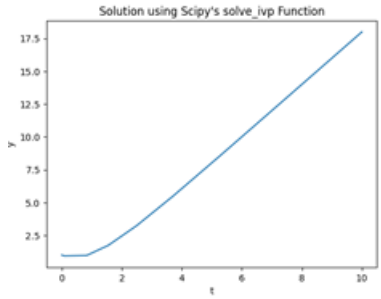

Figure 3: Solution of the equation Using Scipy's solve_ivp method

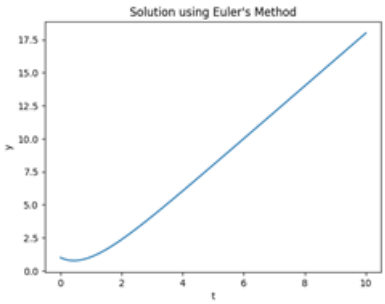

Figure 4: Solution of the equation Using Euler's method

Euler's technique, which offers computational efficiency but has limited precision and stability, is the simplest and most basic strategy. The enhanced Euler's Method (Heun's Method) Fig 4 offers greater precision and stability, however, a short step size is still required for optimal performance. The Range-Kutta Method Fig 2 (Order 4) offers great precision and stability but at a higher computational cost. Scipy's solve_ivp function Fig 3 gives a useful high-level interface with error control and more possibilities, although possibly having a little more computational work. The particular requirements of the temperature control system modeling, taking into account accuracy, stability, processing power, and practicality, define the method of choosing.

CONCLUSION AND FUTURE SCOPE

For modeling temperature management systems with sensor response time using differential-difference equations, we have examined four numerical approaches: Euler's Method, Improved Euler's Method (Heun's Method), Runge-Kutta Method (Order 4), and Scipy's solve_ivp function.

The most straightforward method, Euler's Method, offers simplicity and computational effectiveness. However, due to its low accuracy and stability issues, it is less ideal for complicated systems or those whose behavior is subject to fast change.

Improved In comparison to Euler's Method, Euler's Method (Heun's Method) offers greater accuracy and stability. It accomplishes this by approximating the derivative at each step's midway. Even if its performance has improved, for the best precision it still needs a modest step size.

A popular numerical approach noted for its great accuracy and superb stability is the Runge-Kutta approach (Order 4). It is a good option for more complicated temperature management systems since it provides greater precision compared to Euler's and Improved Euler's methods. However, because there are more derivative assessments, there is a higher computational expense.

A high-level interface for initial value problems, including ordinary differential equations, is provided by Scipy's solve_ivp function. Over certain time periods, it provides ease, error prevention, and integration. Internally, the function applies a number of numerical techniques, including Runge-Kutta techniques, to get precise results. However, compared to manually constructed methods, it might have a little bit more processing cost.

The method of choice is determined by the particular specifications of the modeling of the temperature control system. Euler's Method might be sufficient for situations that are straightforward or computationally cheap. Simpleness and precision are balanced by the improved Euler's Method. Despite having more demanding processing requirements, the Runge-Kutta Method (Order 4) is ideally suited for accurate modeling of complex systems. The solve_ivp function in Scipy offers a practical and adaptable solution for individuals looking for a high-level interface with sophisticated functionality.

In general, the choice of a numerical approach depends on the trade-off between the desired level of convenience, accuracy, stability, and processing resources. When deciding on the best approach for employing differential-difference equations to model temperature control systems with sensor response time, researchers and practitioners should take these considerations into account.

References

1. Luo, Jie, et al. "Battery thermal management systems (BTMs) based on phase change material (PCM): A comprehensive review." *Chemical Engineering Journal* 430 (2022): 132741.
2. Saqib, Muhammad, Ilyas Khan, and Sharidan Shafie. "Application of fractional differential equations to heat transfer in hybrid nanofluid: modeling and solution via integral transforms." *Advances in Difference Equations* 2019.1 (2019): 1-18.
3. Singh, Harendra, Devendra Kumar, and Dumitru Baleanu, eds. *Methods of mathematical modelling: fractional differential equations*. CRC Press, 2019.
4. Ijaz Khan, M., et al. "Mathematical and numerical model for the malaria transmission: Euler method scheme for a malarial model." *International Journal of Modern Physics B* 37.16 (2023): 2350158.

5. Simangunsong, Laurent, and Sudi Mungkasi. "Fourth order Runge-Kutta method for solving a mathematical model of the spread of HIV-AIDS." *AIP Conference Proceedings*. Vol. 2353. No. 1. AIP Publishing, 2021.

6. Freer, Robert, and Anthony V. Powell. "Realising the potential of thermoelectric technology: A Roadmap." *Journal of Materials Chemistry C* 8.2 (2020): 441-463.

7. Kumar, Ashish, et al. "Efficient stochastic model for operational availability optimization of cooling tower using metaheuristic algorithms." *IEEE Access* 10 (2022): 24659-24677.

8. Babanezhad, Meisam, et al. "Developing intelligent algorithm as a machine learning overview over the big data generated by Euler–Euler method to simulate bubble column reactor hydrodynamics." *ACS omega* 5.32 (2020): 20558-20566.

9. Mohanty, Jyoti Ranjan. "Temperature Dependent Magnetic Properties of Crystallographically Amorphous Tbco: An Atomistic Simulation Study."

10. Uçak, Kemal. "A novel model predictive Runge–Kutta neural network controller for nonlinear MIMO systems." *Neural Processing Letters* 51.2 (2020): 1789-1833.

11. Lin, Jiayuan, et al. "A review on recent progress, challenges and perspective of battery thermal management system." *International Journal of Heat and Mass Transfer* 167 (2021): 120834.

12. ODE, A. Single First-Order. "Appendix C SciPy's odeint Ordinary Differential Equation Solver."

13. Arora, Geeta, Varun Joshi, and Isa Sani Garki. "Developments in Runge–Kutta Method to Solve Ordinary Differential Equations." *Recent Advances in Mathematics for Engineering* (2020): 193-202.