NUMERICAL SIMULATION OF HEAT TRANSFER DYNAMICS IN SMART RESIDENTIAL ENVIRONMENTS USING FINITE DIFFERENCE METHODS

ISMOILJONOV H.

Student Fergana State University, ismoiljonovhasanboy0@mail.com

Abstract: This article investigates the numerical simulation of two-dimensional heat transfer processes within a room modeled after a smart home environment. The governing equation is the classical parabolic heat diffusion partial differential equation $\frac{\partial U}{\partial t} = a^2 \nabla^2 U$*, discretized over a 30×50 spatial grid using the Forward Euler finite difference scheme. Dirichlet boundary conditions represent three physically distinct zones: a radiator as the primary heat source, a window as the principal heat sink, and a door modeling ambient ventilation exchange. The computational model is implemented using HTML5 Canvas, CSS3, and JavaScript, enabling real-time interactive visualization. The influence of the thermal diffusivity coefficient on solution stability is analyzed, and the role of stochastic perturbations in achieving physical realism is discussed. Results confirm that the discrete model reproduces the expected steady-state thermal profile consistent with theoretical predictions.*

Keywords: heat diffusion equation, finite difference method, Forward Euler scheme, Dirichlet boundary conditions, 2D Laplacian, Canvas API, smart home thermal modeling, thermal diffusivity, stochastic perturbation, temperature field visualization

Introduction. The automation of indoor climate control represents one of the most practically significant applications of applied mathematics in the modern built environment. Smart home systems — designed to monitor and regulate temperature, humidity, and air quality — require not merely hardware sensors and control logic, but a sound mathematical foundation describing the underlying physical phenomena. Without an accurate model of how thermal energy propagates through a bounded spatial domain, any control strategy remains heuristic at best. The present work addresses this requirement by constructing a two-dimensional numerical simulation of room-scale heat transfer, grounded in classical mathematical physics. The heat diffusion equation, a second-order parabolic partial differential equation (PDE), serves as the governing model. Its discrete approximation via the Forward Euler finite difference scheme on a structured rectangular grid provides a computationally tractable

framework amenable to real-time browser-based execution. The motivation for implementing the simulation in a standard web technology stack — HTML5, CSS3, and JavaScript — rather than a dedicated scientific computing environment is deliberate. Such an implementation ensures broad accessibility, requires no specialized software installation, and demonstrates that sophisticated physical modeling can be embedded within lightweight interactive applications. This is especially relevant for educational contexts and rapid engineering prototyping. The article is structured as follows: Section 2 establishes the mathematical basis of the heat equation and its two-dimensional formulation. Section 3 details the discretization procedure and stability analysis. Section 4 describes the boundary conditions and their physical justification. Section 5 covers the software architecture and rendering pipeline. Section 6 discusses the stochastic noise model. Section 7 presents and interprets the simulation results. Section 8 concludes with directions for future extension.

Mathematical Model: The Heat Diffusion Equation. The physical process of heat conduction in a homogeneous isotropic medium is governed by the Fourier heat equation. In the general three-dimensional case, this takes the form of a parabolic partial differential equation relating the rate of change of temperature at a point to the spatial curvature of the temperature field. For the two-dimensional case — appropriate when modeling a horizontal cross-section of a room or treating the vertical dimension as uniform — the governing equation is:

$$\frac{\partial U}{\partial t} = a^2 \cdot \left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right) = a^2 \cdot \nabla^2 U \qquad (1)$$

Equation 1. Two-dimensional heat diffusion equation

Here $U(x, y, t)$ denotes the temperature field as a function of the two spatial coordinates x and y and time t. The scalar coefficient a^2 is the thermal diffusivity of the medium, defined as $a^2 = \frac{k}{\rho \cdot c_p}$, where k is the thermal conductivity $[W/(m \cdot K)]$, ρ is the mass density [kg/m³], and c_p is the specific heat capacity $[J/(kg \cdot K)]$. Higher values of a^2 indicate a material through which heat propagates more rapidly.

The operator $\nabla^2 U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}$, known as the Laplacian of U, has a transparent physical interpretation: it measures how much the temperature at a given point differs from the local spatial average of its neighbors. When $\nabla^2 U > 0$, the point is cooler than its surroundings and absorbs heat; when $\nabla^2 U < 0$, it is warmer and releases heat. This local diffusive exchange mechanism, accumulated across the entire domain, produces the macroscopic spreading of thermal energy over time.

Equation (1) belongs to the class of linear parabolic PDEs. Its linearity implies that the superposition principle holds: the total temperature field resulting from multiple boundary sources can be decomposed into contributions from each source independently. This property is exploited in the present model, where the radiator, window, and central sensor simultaneously impose distinct boundary temperatures on different portions of the domain.

Figure 1. Conceptual illustration of heat diffusion stages from source to equilibrium

Finite Difference Discretization and Stability Analysis. To enable numerical computation, the continuous domain must be replaced by a finite set of discrete points. The room plane is partitioned into a structured rectangular grid of ROWS = 30 rows and COLS = 50 columns, yielding $N = 1500$ cells in total. Each cell is identified by integer indices (r, c), where $r \in \{0, \dots, 29\}$ and $c \in \{0, \dots, 49\}$, and stores a scalar temperature value grid[r][c] representing the spatially averaged temperature over the cell.

The spatial second derivatives in the Laplacian are approximated by the standard second-order central finite difference scheme. For a uniform grid with spacing Δx in the horizontal direction and Δy in the vertical direction, the discrete Laplacian at cell (r, c) is:

$$L[r][c] = U[r-1][c] + U[r+1][c] + U[r][c-1] + U[r][c+1] - 4 \cdot U[r][c] \quad (2)$$

Equation 2. Discrete 2D Laplacian operator (five-point stencil)

This is the standard five-point finite difference stencil, which approximates $\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}$ with second-order accuracy $O(\Delta x^2 + \Delta y^2)$. For time integration, the Forward Euler (explicit) method is employed. Denoting the superscript n as the time step index, the update rule is:

$$U[r][c]^{n+1} = U[r][c]^{n} + a^2 \cdot L[r][c]^{n} \quad (3)$$

Equation 3. Forward Euler time-stepping scheme

This explicit scheme computes the temperature at the next time step entirely from values at the current step. The key advantage is simplicity of implementation: no linear system needs to be solved, and each cell update is independent of others at the same time level. The computational cost per time step scales as $O(N)$, which for $N = 1500$ cells is negligible on modern hardware.

The primary limitation of the explicit scheme is its conditional stability. The von Neumann stability condition for the 2D heat equation on a uniform square grid ($\Delta x = \Delta y$) requires:

$$a^2 \cdot \frac{\Delta t}{\Delta x^2} \leq \frac{1}{4} \quad (4)$$

Equation 4. Stability condition for the explicit 2D scheme

In the present implementation, the thermal diffusivity coefficient is set to $a^2 = 0.22$ in normalized units consistent with the pixel grid dimensions. To maintain stability and accelerate convergence toward the quasi-steady state, diffusionStep() is invoked 10 times per animation frame, each call performing 5 inner Euler iterations — thus executing 50 discrete time steps between consecutive rendered frames. This multi-step approach efficiently advances the thermal field while respecting the stability bound.

Boundary Conditions and Physical Justification. The heat diffusion equation requires boundary conditions specifying the temperature (or its flux) at the edges and

interior fixed points of the domain. The present model exclusively employs Dirichlet conditions — that is, the temperature is prescribed to a fixed value — at all boundary zones. Three distinct boundary regions are defined, each representing a physically identifiable feature of a furnished room.

The first boundary condition models the radiator, positioned along the upper wall of the room. It spans columns 12 through 38 of row 0, with the same values imposed also on row 1 to represent the finite spatial extent of the heating element. The radiator temperature is user-controlled through sensor S1 and has a default value of 28°C. Enforcing the condition on two rows provides a thermal influence depth into the domain, rather than a mathematically infinitesimally thin boundary source.

The second boundary condition represents the window on the left wall, spanning rows 6 through 22 of column 0. The window is the primary heat sink and is controlled by sensor S2 (default 16°C). Column 1 of the same row range is set to S2 + 1.5°C — slightly warmer than the window surface itself — to represent the thermal gradient observed in the thin air layer immediately adjacent to a cold glass surface, where convective effects begin to moderate the sharp temperature contrast.

The third boundary condition is applied to the door region at the bottom of the domain, spanning columns 19 through 30 of row 29. The door is fixed at 17°C, representing the ventilating effect of air circulation through a doorway connecting to an adjacent conditioned space. Additionally, the entire right wall (column 49) is set to 18°C for all rows, modeling the slightly warmer surface of an insulated exterior wall receiving indirect solar gain or adjacent to a heated space. Sensor S3, occupying rows 17–19 and columns 34–36, can be interpreted as a secondary device such as a fan-coil unit or supplemental heater placed in the interior of the room.

The function applyBoundaries() is invoked at the end of every Euler iteration, immediately after computing the new grid values. This ensures that the boundary constraints are re-imposed at each time step and cannot be eroded by the diffusion process — a numerically critical requirement, since without this re-enforcement the

Dirichlet conditions would effectively become Neumann (flux-free) conditions after the first iteration.

Software Architecture and Rendering Pipeline. The simulation is realized as a single-page web application composed of three source files following the conventional separation-of-concerns pattern: index.html defines the semantic structure, style.css specifies the visual presentation, and script.js contains all computational and rendering logic.

The HTML document declares a 900×550 pixel Canvas element, which serves as the primary rendering surface. The interface is divided into three panels: a top header bar displaying the application title, a system status indicator (simulating IoT device online state), and a real-time digital clock; a left sidebar housing the three sensor control cards and two opacity sliders; and the main room section occupying the right portion of the viewport and containing the canvas together with a real-time statistics panel below it.

The stylesheet employs a dark-mode palette defined through CSS custom properties (variables) in the :root selector, enabling consistent theming throughout the interface. Color tokens include: --bg-deep (#020617) for the primary background, --bg-panel (#0f1e2e) for control panels, and --sky (#38bdf8) as the primary accent color. Sensor identities are encoded chromatically: red (#ef4444) for the radiator (S1), blue (#3b82f6) for the window (S2), and green (#22c55e) for the central device (S3). The Space Mono monospace font is used for numerical readouts to align decimal points, while Syne is applied to labels and headings for typographic contrast.

The JavaScript module is organized into four functional groups. The first group handles state initialization: declaring grid dimensions (ROWS, COLS), cell dimensions in pixels (cellW, cellH), the opacity control variables (roomOpacity, heatOpacity), and the simulation interval handle (simInterval). The second group manages grid operations: initGrid() allocates the 30×50 array and fills all cells with the ambient initial temperature of 20°C, then calls applyBoundaries(). The third group implements the computational core: diffusionStep() executes the discrete Euler update of Equation

(3), and addNoise() injects the stochastic perturbation. The fourth group governs rendering: draw() composes the canvas frame by sequentially blending the background room image, the heat map overlay, the grid lines, the directional arrows, the sensor markers, and the boundary labels.

Figure 3. General user interface of the simulation application

The getColor(temp) function implements the temperature-to-color mapping using a piecewise linear interpolation across four color stages. The normalized ratio r = (temp − 14)/18 maps the temperature interval [14°C, 32°C] to [0, 1]. The four color transitions are: Blue to Cyan (r ∈ [0, 0.25]), Cyan to Green (r ∈ [0.25, 0.5]), Green to Yellow (r ∈ [0.5, 0.75]), and Yellow to Red (r ∈ [0.75, 1.0]). This "blue-cool to red-warm" palette conforms to the convention used in infrared thermal imaging and is intuitively interpretable by users without specialized training.

Stochastic Perturbation and Physical Realism. A purely deterministic numerical solution of the heat equation converges to a spatially smooth, stationary temperature field in finite time. While mathematically correct given the idealized boundary conditions, such a solution appears visually static and fails to capture the minor turbulent fluctuations characteristic of real indoor thermal environments — arising from convective air currents, local variations in wall emissivity, and imprecision in sensor readings.

To address this, the addNoise() function introduces a small stochastic perturbation to the interior grid values at each animation frame. The update applied to every interior cell is:

$$U[r][c] \leftarrow U[r][c] + (rand - 0.5) \cdot 0.08 \quad (5)$$

Equation 5. Stochastic noise injection formula

Here rand ~ Uniform(0,1) is drawn independently for each cell. The net effect is a zero-mean additive perturbation uniformly distributed on [−0.04, +0.04] °C. This amplitude is approximately two orders of magnitude smaller than the dominant temperature gradients in the domain (12–13°C between radiator and window), ensuring that the noise does not meaningfully distort the solution accuracy while providing sufficient visual variation to prevent the rendered heat map from appearing frozen.

An important implementation detail is that noise is applied exclusively to interior cells — the boundary cells (rows 0, 29 and columns 0, 49, as well as the fixed sensor zones) are excluded. This is correct because their temperatures are immediately overwritten by applyBoundaries() in the next iteration; injecting noise there would have no persistent effect and would merely introduce unnecessary computation.

Simulation Results and Physical Interpretation. Upon initialization, the entire grid is set to the uniform ambient temperature U_0 = 20°C. After applyBoundaries() enforces the Dirichlet conditions, the radiator zone (row 0–1, columns 12–38) is set to 28°C and the window cells (rows 6–22, column 0–1) are set to 16°C and 17.5°C respectively. A single preliminary diffusion step is executed prior to rendering the initial frame, so the user observes a state where thermal influence has already begun propagating.

During the early transient phase — approximately the first 5–10 animation frames — a hot plume develops from the radiator and spreads downward and laterally. Simultaneously, a cold region extends from the window into the interior of the room. The boundary between these zones exhibits the steepest temperature gradient and the most rapid color transitions in the visualization. The door region at the bottom introduces a weak cold influence from below, creating a secondary low-temperature zone in the lower-center portion of the domain.

After approximately 20–30 animation frames, the temperature field approaches a quasi-steady state. In this regime, the statistics panel consistently reports an average temperature of approximately 21–22°C, a maximum near 28°C (at the radiator boundary), a minimum near 16°C (at the window boundary), and a temperature

difference of 12–13°C. These values are physically consistent with the imposed boundary conditions and the domain geometry.

The interactive sensor controls allow the user to investigate parametric variations. Increasing S1 (radiator temperature) enlarges the hot zone and shifts the average temperature upward. Decreasing S2 (window temperature) intensifies the cold zone and increases the overall temperature differential. Adjusting S3 introduces a localized anomaly in the interior — modifiable between a secondary heat source and a cooling device — demonstrating the model's capacity to represent multi-device configurations.

Conclusion. This article has presented a complete mathematical and computational study of two-dimensional heat transfer simulation for smart home applications. The principal contributions can be summarized as follows. From a mathematical standpoint, the work demonstrates the effective application of the classical parabolic heat equation to a practically motivated domain, with physically grounded Dirichlet boundary conditions representing a radiator, a window, and a door. The derivation and justification of the discrete Forward Euler scheme, together with the explicit formulation of the von Neumann stability condition, provide a rigorous foundation for the computational approach. From an engineering standpoint, the implementation achieves real-time interactive simulation within a standard web browser without requiring any external libraries or installed software. The modular JavaScript architecture cleanly separates physical computation from user interface logic, facilitating future extension and maintenance. The opacity-controlled overlay of the room floor plan onto the heat map provides an intuitive spatial reference for interpreting thermal patterns. From a scientific standpoint, the quasi-steady state results — mean temperature of 21–22°C, maximum gradient of 12–13°C between the radiator and window — are quantitatively consistent with the imposed boundary conditions and with the expected behavior of the governing equation. The stochastic perturbation model successfully introduces physical realism without compromising numerical accuracy. Future extensions of this work could pursue several directions: replacing the

explicit Euler scheme with an unconditionally stable implicit method (Crank–Nicolson) to permit larger time steps and finer grids; incorporating a convection term to model forced or natural air circulation; extending the model to three spatial dimensions; integrating real-time sensor data via WebSocket connections from physical IoT devices; and harnessing GPU parallelism through WebGL compute shaders for high-resolution simulations applicable to building-scale thermal management.

References

1. Crank, J., & Nicolson, P. (1947). A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. Mathematical Proceedings of the Cambridge Philosophical Society, 43(1), 50–67.
2. LeVeque, R. J. (2007). Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems. Philadelphia: SIAM.
3. Cengel, Y. A., & Ghajar, A. J. (2014). Heat and Mass Transfer: Fundamentals and Applications, 5th ed. New York: McGraw-Hill Education.
4. Strikwerda, J. C. (2004). Finite Difference Schemes and Partial Differential Equations, 2nd ed. Philadelphia: SIAM.
5. Тихонов, А. Н., & Самарский, А. А. (1977). Уравнения математической физики. Москва: Наука.
6. Thomas, J. W. (1995). Numerical Partial Differential Equations: Finite Difference Methods. New York: Springer.
7. Smith, G. D. (1985). Numerical Solution of Partial Differential Equations: Finite Difference Methods, 3rd ed. Oxford: Oxford University Press.
8. Morton, K. W., & Mayers, D. F. (2005). Numerical Solution of Partial Differential Equations: An Introduction. Cambridge: Cambridge University Press.
9. Flanagan, D. (2020). JavaScript: The Definitive Guide, 7th ed. Sebastopol: O‘Reilly Media.
10. MDN Web Docs. (2024). Canvas API Reference. https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API
11. Incropera, F. P., DeWitt, D. P., Bergman, T. L., & Lavine, A. S. (2011). Fundamentals of Heat and Mass Transfer, 7th ed. Hoboken: John Wiley & Sons.
12. ISO 7730:2005. Ergonomics of the Thermal Environment — Analytical Determination and Interpretation of Thermal Comfort Using Calculation of the PMV and PPD Indices. Geneva: ISO.
13. Pozrikidis, C. (2011). Introduction to Finite and Spectral Element Methods Using MATLAB, 2nd ed. Boca Raton: CRC Press.
14. Fourier, J. B. J. (1822). Théorie analytique de la chaleur. Paris: Firmin Didot Père et Fils.