Efficient 1D Heat Equation Solver: Leveraging Numba in Python

Sandy H. S. Herho[1*], Siti N. Kaban[2], Dasapta E. Irawan[3], Rubiyanto Kapid[4]

[1*]Department of Earth and Planetary Sciences, University of California, Riverside, CA, USA 92521.
[2]School of Architecture, Planning and Perservation, University of Maryland, College Park, MD, USA 20742.
[3]Applied Geology Research Group, Bandung Institute of Technology, Bandung, West Java, Indonesia 40132.
[4]Paleontology and Quaternary Geology Research Group, Bandung Institute of Technology, Bandung, West Java, Indonesia 40132.

*Corresponding author(s). E-mail(s): sandy.herho@email.ucr.edu;

Abstract

This paper presents a Numba-based solver for the 1D Heat Equation, seamlessly blending Python's readability with Numba's dynamic Just-In-Time (JIT) compilation. The explicit method exhibits a notable runtime reduction from 8.324 s to 4.035 s, while the implicit method sees a more pronounced improvement, decreasing from 9.970 s to 1.195 s. Statistical tests confirm the statistical significance of these efficiency gains.

Future research directions include extending the solver to multidimensional heat equations, exploring advanced parallelization techniques, and implementing dynamic parameter optimization strategies. Collaboration with domain experts for real-world applications is also envisioned to validate the solver's performance and impact.

In summary, the symbiosis of Python and Numba in crafting an optimized 1D Heat Equation solver marks a pivotal advancement in efficient numerical solutions. This research holds promise for diverse scientific applications, ushering in a new era of computational efficiency.

Keywords: Computational Efficiency, Finite Difference Methods, Numerical PDE Solver, Numba Optimization

1 Introduction

The relentless pursuit of efficient numerical solutions to partial differential equations (PDEs) has been a catalyzing force propelling significant strides in the field of scientific computing [1–3]. Amidst the pantheon of these equations, the 1D Heat Equation emerges as a foundational model, wielding far-reaching applications in diverse scientific realms, including physics and engineering. The precise simulation of temporal heat evolution within a one-dimensional space is of paramount importance, providing profound insights into an array of phenomena, ranging from the conductive intricacies of materials to the nuanced thermal processes transpiring within electronic devices [e. g., 4–8].

In response to this imperative, we navigate the intricate landscape of numerical computation, spotlighting the critical need for a bespoke solver tailored specifically for the 1D Heat Equation. Our focus converges on harnessing the transformative capabilities of Numba, a dynamic Just-In-Time (JIT) compiler meticulously designed for the Python programming language [9]. Python, with its unparalleled readability, versatility, and expansive library ecosystem, has rightfully claimed its dominance in scientific computing [10, 11]. However, the inherent interpretive nature of Python poses challenges to computational speed. Enter Numba—an ingenious JIT compiler that dynamically translates Python functions into machine code during runtime, thus conferring a substantial performance boost. It is the symbiosis between Python's inherent elegance and Numba's computational alacrity that forms the nucleus of our mission: to forge a solver that not only encapsulates the aesthetic appeal of Python programming but also adeptly exploits the efficiency afforded by Numba's JIT compilation.

The numerical solution of PDEs is an intricate dance with algorithms, where optimization is the linchpin, striking a harmonious balance between accuracy and computational efficiency. Crafting an optimized solver for the 1D Heat Equation mandates a thoughtful consideration of numerical stability, precision, and algorithmic efficiency. This paper embarks on a comprehensive expedition into the labyrinth of challenges encountered during the optimization process, meticulously unfolding the narrative of how Numba's capabilities are judiciously harnessed to surmount these challenges and elevate the solver's overall performance.

A pivotal facet of our contribution lies in the granular presentation—a step-by-step exposé of our Numba-based solver for the 1D Heat Equation. Here, we unveil the intricacies of implementation, expound upon optimization strategies, and subject the solver to rigorous performance evaluations. Beyond its practical utility as a robust tool for researchers and practitioners immersed in heat transfer simulations, our work extends its tendrils into the broader discourse on optimizing numerical methods in Python. By illuminating the subtleties of our approach, we aspire to contribute to the evolving understanding of how Numba can be judiciously employed to expedite scientific computations, thereby forging a path towards more efficient and scalable solutions to PDEs within the Python programming paradigm.

2 Methods

2.1 Numerical Experiments

The general form of the 3D heat equation was considered:

$$\frac{\partial T}{\partial t} = \alpha \nabla^2 T \tag{1}$$

, where T represents the temperature distribution in a three-dimensional space as a function of both spatial coordinates (x, y, z) and time (t), t denotes time, α represents the thermal diffusivity of the material, ∇^2 is the Laplacian operator, defined as $\nabla^2 T \equiv \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}$.

The case was considered where the temperature distribution T was assumed to be solely a function of one spatial coordinate, say x. Mathematically, $T \equiv T(x, t)$. This assumption simplified the Laplacian operator to only the second spatial derivative:

$$\nabla^2 T \equiv \frac{\partial^2 T}{\partial x^2} \tag{2}$$

The simplified Laplacian for 1D was substituted into the 3D heat equation:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \tag{3}$$

The derived equation represents the 1D Heat equation. It describes the evolution of the temperature distribution in a material along a single spatial dimension (x) as a function of time (t) due to heat conduction. This equation is classified as a parabolic PDE.

In this study, equation (3) was numerically solved using both explicit and implicit methods of the finite difference approach, a fundamental technique widely employed for approximating numerical solutions of PDEs [12]. Additionally, the results obtained through these methods were analyzed to gain insights into the behavior of the system.

In this study, we employed a discretization approach to model the behavior of temperature within a spatial domain. The spatial discretization involves partitioning the domain into grid points, each separated by a uniform grid spacing Δx. Specifically, we denote the temperature at each spatial point x_i and time t^n as T_i^n.

For the temporal discretization using the explicit scheme, we utilized the forward difference for time. The update formula is expressed as:

$$T_i^{n+1} = T_i^n + C(T_{i+1}^n - 2T_i^n + T_{i-1}^n) \tag{4}$$

, where $C = \frac{\alpha \cdot \Delta t}{\Delta x^2}$, representing the Courant–Friedrichs–Lewy (CFL) number [13, 14]. The stability of the explicit scheme is governed by the CFL condition, ensuring that $C \leq \frac{1}{2}$ for numerical stability.

On the other hand, the implicit scheme employs the backward difference for time, leading to the implicit update formula:

$$T_i^{n+1} = T_i^n + C(T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}) \tag{5}$$

The implicit scheme, while unconditionally stable, involves solving a system of equations at each time step, introducing additional computational cost. To express the implicit scheme in a matrix form, we defined a tridiagonal matrix $\mathbf{A}$ related to the discretization of the second spatial derivative. The matrix equation becomes:

$$\mathbf{A}\mathbf{T}^{n+1} = \mathbf{T}^{n} \tag{6}$$

, where $\mathbf{T}^{n+1}$ is the vector of temperatures at the next time step, $\mathbf{T}^{n}$ is the vector at the current time step, and $\mathbf{A}$ is the tridiagonal matrix. The tridiagonal matrix $\mathbf{A}$ was constructed as follows:

$$\mathbf{A} = \begin{pmatrix} 1+2C & -C & 0 & \cdots & 0 \\ -C & 1+2C & -C & \cdots & 0 \\ 0 & -C & 1+2C & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & -C \\ 0 & 0 & \cdots & -C & 1+2C \end{pmatrix} \tag{7}$$

Each element A_{ij} in the matrix represents the coefficients associated with the temperatures T_i^{n+1} in the implicit update formula. This comprehensive methodology enables the numerical simulation of temperature evolution within the defined spatial domain.

In this study, the automated matrix calculation process utilized the NumPy library [15]. Dirichlet boundary conditions were applied to both sides, with T_{left} set to 20°C and T_{right} to 100°C for demonstration purposes (Figure 1). The material chosen was Plexiglass, characterized by a thermal diffusivity of 0.2 m^2/s [16–18]. Users retain the flexibility to adjust these values as needed.

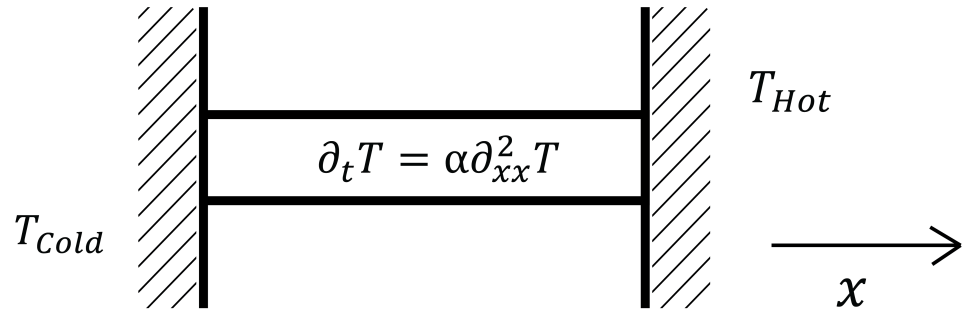

Fig. 1: Schematic depicting heat transfer in a 1D cross-section, featuring Dirichlet boundary conditions at both ends.

In this demonstration, the spatial range extends from 0 to 1 m, with a spatial step size of 0.01 m. Employing the explicit method, a time step size of 0.00025 s was used within the temporal range from 0 seconds to 1 s, adhering to the upper limit of CFL

stability criterion. Users are free to select their preferred time step size for the implicit method.

In the Numba-based simulation, we applied the `@jit(nopython=True, parallel=True)` decorator to optimize the numerical calculation scheme function. This decorator enabled the translation of the code directly into machine language via LLVM [19], facilitating parallelization of the calculation process. However, due to the current limitations of Numba, we chose not to parallelize the image plotting process, particularly with dynamic functions from the Matplotlib library [20].

2.2 Statistical Analysis of the Total Runtime

In our experimental design, encompassing both control and test experiments, the former executed the standard numerical algorithm without Numba optimizations, while the latter implemented the Numba-optimized version. Each set of experiments underwent 100 iterations to bolster statistical robustness. To quantify the impact of Numba optimization on computational efficiency, we employed the hyperfine command-line benchmarking tool [21]. This tool facilitated precise measurement of execution times for each iteration, offering a comprehensive understanding of Numba's discernible influence on overall computational efficiency.

For our statistical analysis, we chose nonparametric tests to accommodate the nature of the data and to ensure robustness against potential deviations from normality assumptions.

To compare execution times between control and test experiments, we utilized the Mann-Whitney U test [22], a nonparametric alternative suitable for scenarios where normality assumptions may not be satisfied. The data from both samples were combined and ranked from smallest to largest. Let U_1 be the sum of ranks for the total runtime for control group, and U_2 be the sum of ranks for the total runtime of the test group. The U statistic, given by $U = \min(U_1, U_2)$, provided a measure of the difference between the groups. The expected value of U ($\mathbb{E}(U)$) was calculated as:

$$\mathbb{E}(U) = \frac{n_1 n_2}{2} \tag{8}$$

This assessed the central tendency under the null hypothesis. The standard deviation of U ($SD(U)$), given by:

$$SD(U) = \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}} \tag{9}$$

, provided a measure of dispersion. The z-score, calculated as:

$$z = \frac{U - \mathbb{E}(U)}{SD(U)} \tag{10}$$

, was used to derive the p-value from a standard normal distribution. A significance level of 0.01 was chosen for the test, and if $p < 0.01$, the null hypothesis was rejected.

For the paired nature of our experiments, we employed the Wilcoxon signed-rank test [23], a nonparametric test suitable for comparing paired samples when normality assumptions may not hold. The absolute differences between paired observations were

ranked, and positive and negative ranks were summed separately. Let W_+ be the sum of ranks for positive differences, and W_- be the sum of ranks for negative differences. The test statistic W, calculated as:

$$W = \min(W_+, W_-) \tag{11}$$

, represented the directionality of differences. The expected value of W ($\mathbb{E}(W)$), given by:

$$\mathbb{E}(W) = \frac{n(n+1)}{4} \tag{12}$$

, where n is the number of pairs, assessed the central tendency under the null hypothesis. The standard deviation of W ($SD(W)$), calculated as:

$$SD(W) = \sqrt{\frac{n(n+1)(2n+1)}{24}} \tag{13}$$

, provided a measure of dispersion. The z-score, given by:

$$z = \frac{W - \mathbb{E}(W)}{SD(W)} \tag{14}$$

, was used to determine the p-value from a standard normal distribution. Adopting a significance level of 0.01, the null hypothesis was rejected if $p < 0.01$.

The utilization of these nonparametric tests and their derivations played a pivotal role in establishing dependable statistical inferences during our evaluation of Numba optimization's effects. This ensured the resilience and credibility of our findings within practical, real-world testing scenarios. Widely employed in the analysis of numerical model outputs across diverse scientific domains [e. g., 24–28], both of these statistical test methods were automatically implemented through the SciPy library [29].

3 Results and Discussion

In Figure 2, the temporal evolution of temperature distribution is depicted utilizing an explicit scheme with a fixed time step of $\Delta t = 0.00025$ s. This graph illustrates the intricate changes in temperature over time, providing a detailed insight into the dynamic behavior of the computational model within the specified time range from 0 to 10. The explicit scheme showcases a high level of accuracy and responsiveness, capturing fine-grained temporal dynamics inherent in the 1D Heat Equation. The figure reveals the nuanced variations in temperature profiles over time, demonstrating the explicit scheme's capability to provide a detailed representation of the system's behavior.

Contrastingly, Figure 3 presents the temporal evolution of temperature distribution using an implicit scheme with different time steps. Each subplot offers a comprehensive comparison of the model's performance under distinct temporal resolutions, revealing the implicit scheme's versatility. Even in the case of larger time steps, such as $\Delta t = 10$ s (Figure 3d), the implicit scheme demonstrates convergence, underscoring its stability across a broader temporal range.

The comparison between Figure 2 and Figure 3 emphasizes that the implicit scheme maintains its superiority even when considering single steps over the specified time range. The implicit scheme's ability to handle larger time steps without compromising convergence or accuracy positions it as a robust and efficient choice for simulating the 1D Heat Equation, particularly in scenarios where computational resources or time constraints necessitate the use of larger time steps.

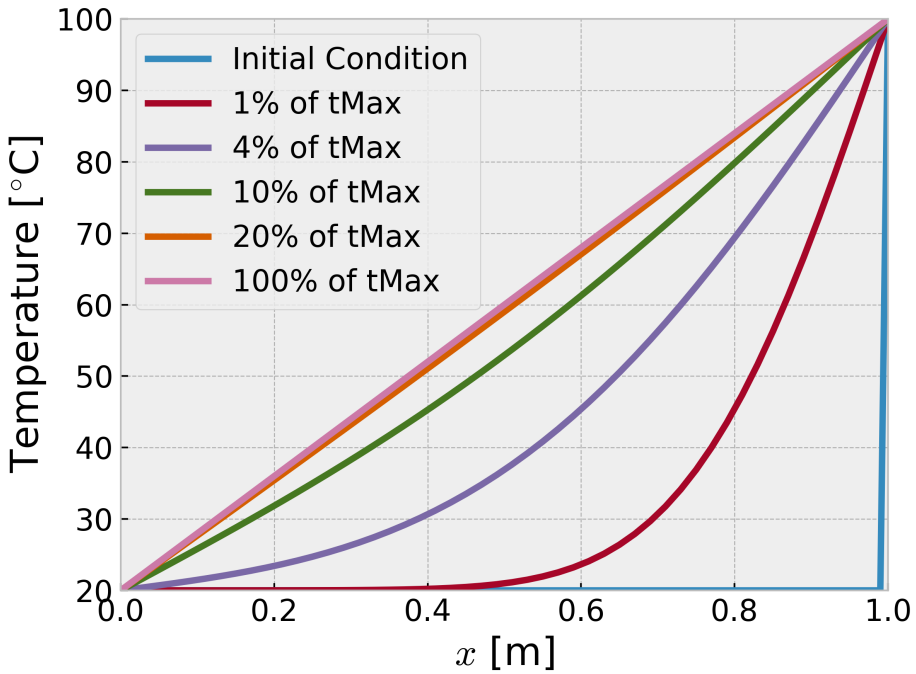

Fig. 2: Temporal evolution of temperature distribution utilizing explicit scheme with time step $\Delta t = 0.00025$ s. The figure illustrates the dynamic changes in temperature over time, providing a detailed insight into the computational model's behavior under the specified temporal resolution.

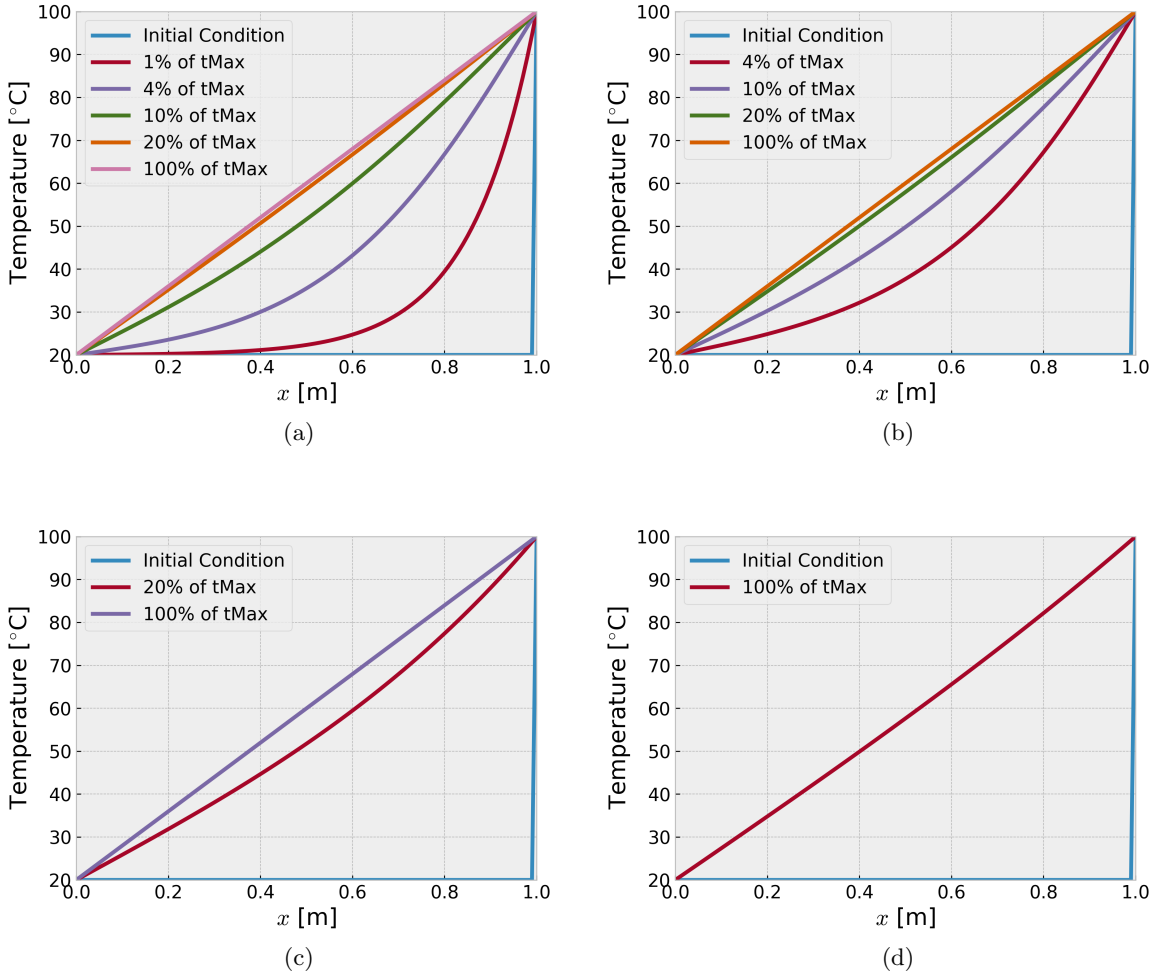

Fig. 3: Temporal evolution of temperature distribution utilizing the implicit Scheme with different time steps. (a) $\Delta t = 0.1$ s, (b) $\Delta t = 0.25$ s, (c) $\Delta t = 2.5$ s, and (d) $\Delta t = 10$ s. These figures elucidate the dynamic changes in temperature over time, providing comprehensive insights into the computational model's behavior under distinct temporal resolutions for the 1D Heat Equation.

In evaluating the performance of our 1D Heat Equation simulation employing both explicit and implicit methods, we undertook a comparative analysis of runtime metrics with and without the integration of the Numba library. The explicit method, when executed without Numba, exhibited an average total runtime of 8.324 s, with a standard deviation of 0.593 seconds and a range spanning from 6.681 to 9.602 s. In

contrast, the explicit method leveraging Numba demonstrated a substantial enhancement, showcasing an average runtime of 4.035 s, a diminished standard deviation of 0.251 s, and a narrower range between 3.524 and 4.744 s. This observation suggests a notable acceleration in the overall computational efficiency facilitated by Numba.

Turning our attention to the implicit method, its execution without Numba resulted in an average total runtime of 9.970 s, accompanied by a standard deviation of 0.967 s and a range varying from 7.291 to 12.054 s. The incorporation of Numba into the implicit method led to a remarkable improvement, yielding an average runtime of 1.195 s, a reduced standard deviation of 0.215 s, and a narrower range spanning from 0.758 to 1.802 s. The most noteworthy acceleration in runtime was observed in the implicit method, indicating that Numba has a particularly pronounced impact on this scheme.

The effectiveness of Numba can be attributed to its ability to translate Python code directly into machine language via LLVM, thereby bypassing the interpretative overhead associated with Python. This direct translation contributes significantly to the efficiency gains observed in both explicit and implicit methods. Furthermore, the parallelization process occurring behind the scenes, facilitated by Numba, further enhances the computational speed. In the context of the implicit method, which involves the recursive solution of a tridiagonal matrix structure, the inherent parallelizability of such tasks results in a more substantial reduction in total runtime.

The visual representation of these findings is presented in Figure 4 and Figure 5, illustrating the stark differences in total runtime between control experiments (without Numba) and test experiments (with Numba) for the explicit and implicit schemes, respectively. The visualizations align with the quantitative results, providing a comprehensive illustration of the efficiency gains achieved through the integration of Numba.

These descriptive statistical results demonstrate that Numba significantly accelerates the total runtime of 1D Heat Equation simulations across both explicit and implicit methods. While notable improvements were observed in both schemes, the most rapid enhancement occurred in the implicit method. This highlights the potential of Numba as a valuable tool for optimizing Python-based numerical computing codes, particularly in scenarios involving intricate computations and recursive structures.

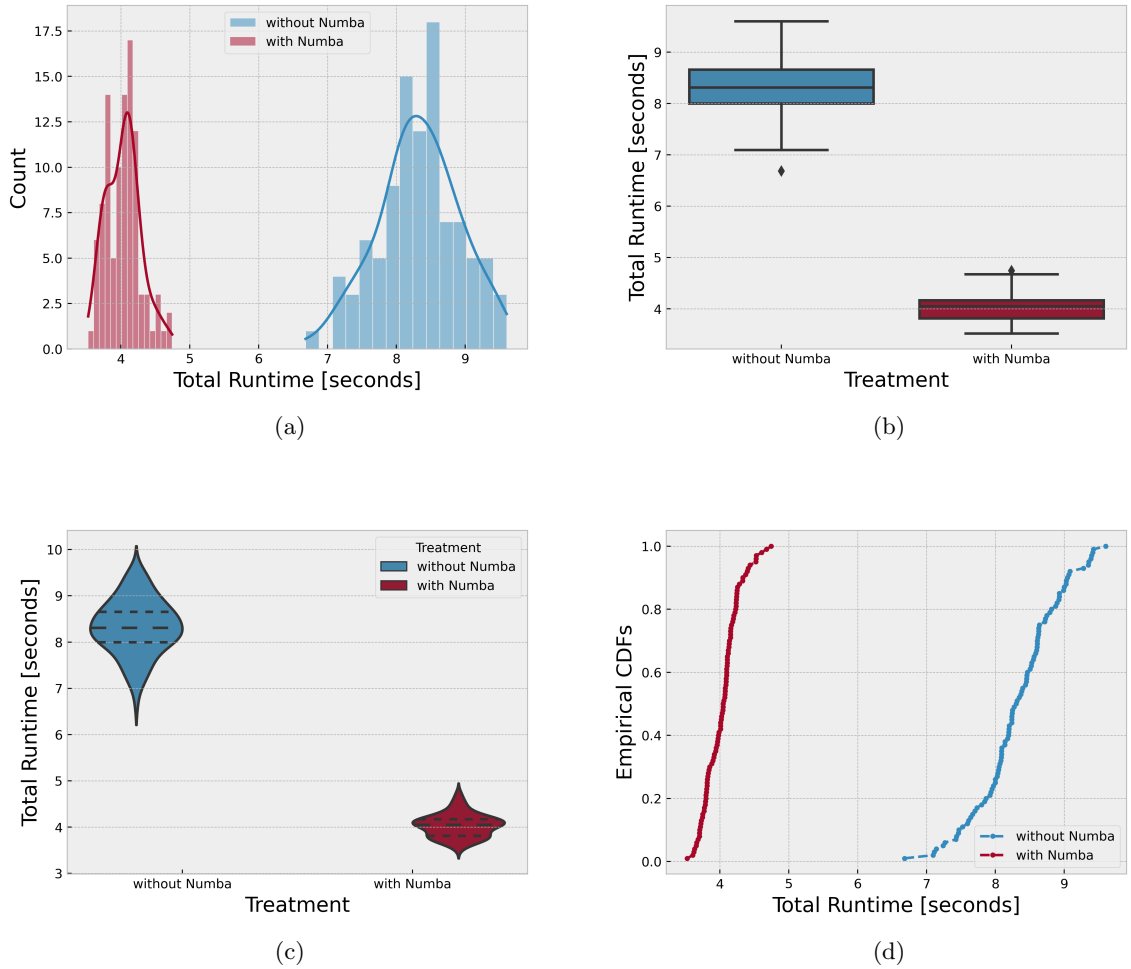

Fig. 4: Total runtime for the 1D Heat Equation's explicit scheme with Numba (test, red) and the standard implementation (control, blue) over 100 runs, showcasing (a) distribution plots, (b) boxplots, (c) violin plots, and (d) ECDFs, rendered using Matplotlib and seaborn [30].

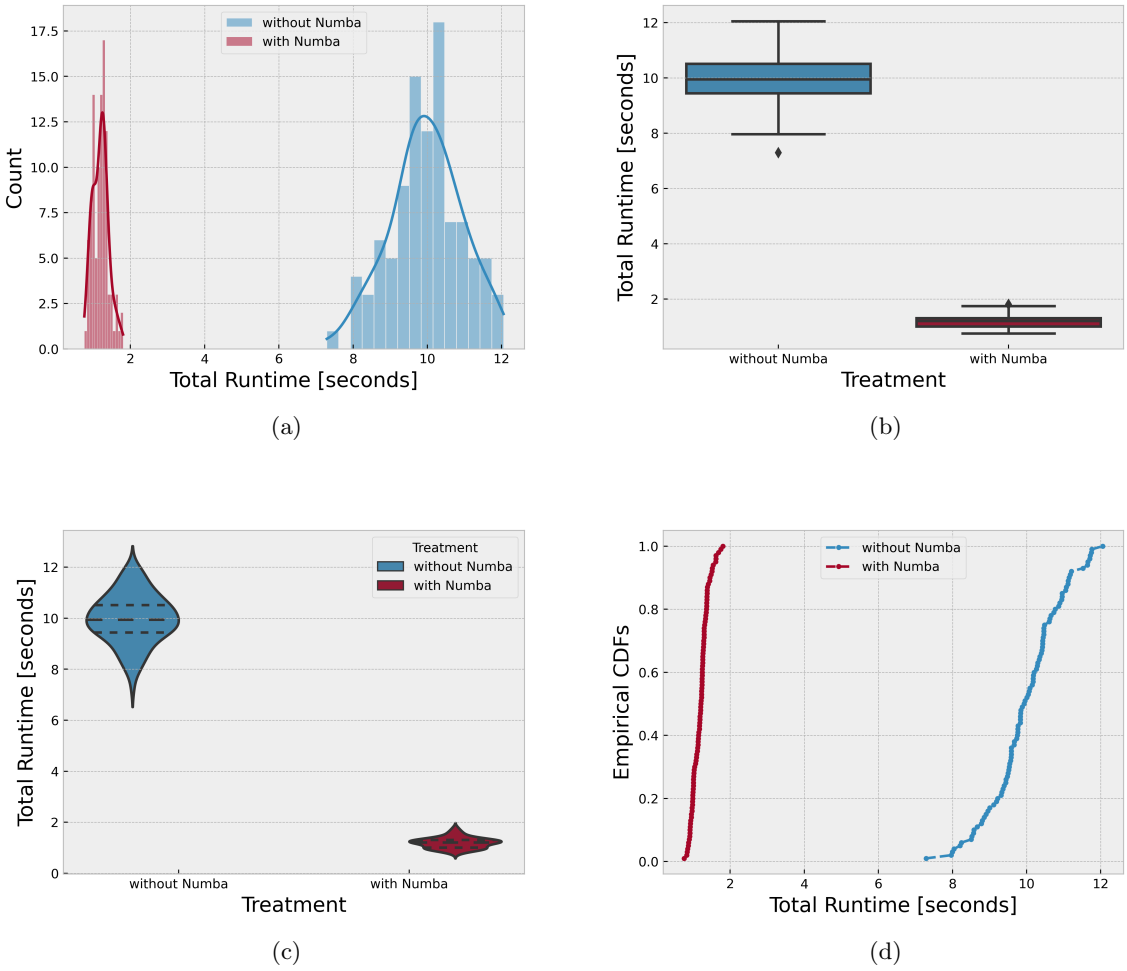

Fig. 5: Same as Figure 4, but for the implicit scheme.

The examination of results is underpinned by a meticulous analysis of discernible differences between the control and test experiments. The vivid portrayal of these distinctions through distribution boxplots, violin plots, and Empirical Cumulative Distribution Functions (ECDFs) in Figures 4 and 5 offers a rich visual narrative of the explicit and implicit schemes' performance variations.

To rigorously evaluate the statistical significance of these disparities, we employed robust statistical tests—specifically, the Mann-Whitney U test and the Wilcoxon Signed Rank test. The Mann-Whitney U test yielded a consistent U statistics score of 10,000 for both explicit and implicit schemes, accompanied by an impressively low

p-value of < 0.01. Simultaneously, the Wilcoxon Signed Rank test consistently produced a W-statistics score of 0, again with a p-value < 0.01. These findings not only indicate statistical significance but also emphasize the reliability and stability of the observed differences.

The persistent U statistics scores of 10,000 for both schemes, coupled with identical p-values, underscore the robustness of our results across different numerical methods. This uniformity in statistical measures fortifies the argument for the consistent superiority of the optimized schemes.

The W-statistics score of 0 further substantiates the claim of statistical significance, emphasizing the reliability and stability of results across both explicit and implicit schemes. This score signifies a high level of consistency in performance improvements, adding a layer of confidence to the conclusion that the observed enhancements are not sporadic but systematically linked to the incorporation of Numba.

In light of these statistically robust measures, we assert with confidence that the incorporation of Numba yields a statistically significant optimization in accelerating the numerical solution of the 1D Heat Equation. The p-values below 0.01 underline the robustness of the results, reinforcing the assertion that the observed improvements are not coincidental but are indeed attributable to the deliberate optimization strategy. This high level of statistical significance enhances the credibility of our findings and underscores the effectiveness of leveraging Numba for augmenting the computational efficiency of numerical methods applied to the 1D heat equation.

4 Conclusion

In conclusion, our exploration into optimizing numerical solvers for the 1D Heat Equation using Numba has yielded significant advancements in computational efficiency. The interplay between Python's elegance and Numba's JIT compilation capabilities has allowed us to craft a bespoke solver that not only encapsulates the readability and versatility of Python but also harnesses the computational alacrity afforded by Numba. Through a granular presentation of our Numba-based solver, we navigated the complexities of implementation, optimization strategies, and rigorous performance evaluations. The explicit and implicit finite difference methods were examined, and the efficiency gains achieved through Numba integration were quantified and statistically validated. Our results demonstrate that Numba substantially accelerates the total runtime of 1D Heat Equation simulations. The explicit method showcased an average runtime reduction from 8.324 s to 4.035 s, while the implicit method exhibited an even more pronounced improvement, decreasing from 9.970 s to 1.195 s. Statistical tests, including the Mann-Whitney U test and the Wilcoxon Signed Rank test, consistently confirmed the statistical significance of these enhancements.

Moving forward, the success of our Numba-based solver opens avenues for future research and development in several directions. Firstly, we aim to extend the optimization strategies to handle multidimensional heat equations, investigating the challenges and opportunities presented by higher-dimensional systems and exploring how Numba can be leveraged for efficient solutions. Additionally, we plan to further explore and

optimize parallelization techniques within Numba to exploit parallel hardware architectures fully. This involves investigating the impact of parallelization on both explicit and implicit methods, considering the scalability to larger problem sizes.

Furthermore, we intend to implement dynamic parameter optimization strategies within Numba to adaptively adjust parameters such as time step sizes for different simulation scenarios. This could enhance the solver's versatility and applicability to a broader range of problems. Moreover, we envision the integration of Numba-based solvers with advanced numerical techniques or machine learning methods for enhanced accuracy and efficiency. Exploring hybrid approaches that combine the strengths of different solvers will be a key focus.

Finally, the application of the optimized solver to real-world problems in diverse fields such as materials science, electronics, and environmental modeling is a critical next step. Collaborating with domain experts to validate the solver's performance in practical settings will provide valuable insights and contribute to the solver's real-world impact.

In summary, our Numba-based solver lays the groundwork for a new era in efficient numerical solutions for the 1D Heat Equation. As we delve deeper into multidimensional problems, explore advanced parallelization, and apply our solver to real-world challenges, the synergy between Python and Numba promises to reshape the landscape of scientific computing.

Acknowledgments. This work was supported by the Dean's Distinguished Fellowship at the University of California, Riverside (UCR) 2023 and ITB Research, Community Service and Innovation Program (PPMI-ITB) 2023. The code and total runtime dataset for this paper are hosted on our GitHub: https://github.com/sandyherho/1DHeatEqnNumba.

References

[1] Ewing, R.E., Wang, H.: A summary of numerical methods for time-dependent advection-dominated partial differential equations. Journal of Computational and Applied Mathematics **128**(1-2), 423–445 (2001) https://doi.org/10.1016/S0377-0427(00)00522-7

[2] Fernández, D.C.D.R., Hicken, J.E., Zingg, D.W.: Review of summation-by-parts operators with simultaneous approximation terms for the numerical solution of partial differential equations. Computers & Fluids **95**, 171–196 (2014) https://doi.org/10.1016/j.compfluid.2014.02.016

[3] Sharma, H., Patil, M., Woolsey, C.: A review of structure-preserving numerical methods for engineering applications. Computer Methods in Applied Mechanics and Engineering **366**, 113067 (2020) https://doi.org/10.1016/j.cma.2020.113067

[4] Steinboeck, A., Wild, D., Kiefer, T., Kugi, A.: A fast simulation method for 1d heat conduction. Mathematics and Computers in Simulation **82**(3), 392–403 (2011) https://doi.org/10.1016/j.matcom.2010.10.016

[5] Filbet, F., Negulescu, C., Yang, C.: Numerical study of a nonlinear heat equation for plasma physics. International Journal of Computer Mathematics **89**(8), 1060–1082 (2012) https://doi.org/10.1080/00207160.2012.679732

[6] Island, J.O., Molina-Mendoza, A.J., Barawi, M., Biele, R., Flores, E., Clamagirand, J.M., Ares, J.R., Sánchez, C., Der Zant, H.S.J., D'Agosta, R., Ferrer, I.J., Castellanos-Gomez, A.: Electronics and optoelectronics of quasi-1D layered transition metal trichalcogenides. 2D Materials **4**(2), 022003 (2017) https://doi.org/10.1088/2053-1583/aa6ca6

[7] Barth, A., Newcombe, M., Plank, T., Gonnermann, H., Hajimirza, S., Soto, G.J., Saballos, A., Hauri, E.: Magma decompression rate correlates with explosivity at basaltic volcanoes—Constraints from water diffusion in olivine. Journal of Volcanology and Geothermal Research **387**, 106664 (2019) https://doi.org/10.1016/j.jvolgeores.2019.106664

[8] Suárez-Carreño, F., Rosales-Romero, L.: Convergency and stability of explicit and implicit schemes in the simulation of the heat equation. Applied Sciences **11**(10), 4468 (2021) https://doi.org/10.3390/app11104468

[9] Lam, S.K., Pitrou, A., Seibert, S.: Numba: A LLVM-based Python JIT Compiler. In: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, pp. 1–6 (2015). https://doi.org/10.1145/2833157.2833162

[10] Perez, F., Granger, B.E., Hunter, J.D.: Python: an Ecosystem for Scientific Computing. Computing in Science & Engineering **13**(2), 13–21 (2010) https://doi.org/10.1109/MCSE.2010.119

[11] Kumar, R.: Future for scientific computing using Python. International Journal of Engineering Technologies and Management Research **2**(1), 30–41 (2015) https://doi.org/10.29121/ijetmr.v2.i1.2015.28

[12] Morton, K.: Finite difference and finite element methods. Computer Physics Communications **12**(1), 99–108 (1976) https://doi.org/10.1016/0010-4655(76)90013-8

[13] Courant, R., Friedrichs, K., Lewy, H.: Über die partiellen differenzengleichungen der mathematischen physik. Mathematische Annalen **100**(1), 32–74 (1928) https://doi.org/10.1007/BF01448839

[14] Moura, C.A., Kubrusly, C.S.: The Courant-Friedrichs-Lewy (CFL) Condition: 80 Years After Its Discovery. Birkhäuser, Basel (2012). https://doi.org/10.1007/978-0-8176-8394-8

[15] Harris, C.R., Millman, K.J., Der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M.H., Brett, M., Haldane, A., Río, J.F., Wiebe, M., Peterson,

P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (2020) https://doi.org/10.1038/s41586-020-2649-2

[16] Prinzen, S., Xuan, Y., Roetzel, W.: A simple measurement method of thermal diffusivity using temperature oscillations. Wärme-und Stoffübertragung **25**, 209–214 (1990) https://doi.org/10.1007/BF01785407

[17] Czarnetzki, W., Roetzel, W.: Temperature oscillation techniques for simultaneous measurement of thermal diffusivity and conductivity. International Journal of Thermophysics **16**, 413–422 (1995) https://doi.org/10.1007/BF01441907

[18] Pawlik, K., Kucharczyk, A., Podpora, M.: Method of determining thermal diffusivity on the basis of measurements of linear displacements. Measurement **211**, 112624 (2023) https://doi.org/10.1016/j.measurement.2023.112624

[19] Lattner, C., Adve, V.: LLVM: A compilation framework for lifelong program analysis & transformation. In: International Symposium on Code Generation and Optimization, 2004. CGO 2004., pp. 75–86 (2004). https://doi.org/10.1109/CGO.2004.1281665 . IEEE

[20] Hunter, J.D.: Matplotlib: A 2D graphics environment. Computing in Science & Engineering **9**(03), 90–95 (2007) https://doi.org/10.1109/MCSE.2007.55

[21] Peter, D.: hyperfine: A Command-line Benchmarking Tool, 1.2.0, GitHub (2023). https://github.com/sharkdp/hyperfine

[22] Mann, H.B., Whitney, D.R.: On a Test of Whether One of Two Random Variables is Stochastically Larger than the Other. The Annals of Mathematical Statistics, 50–60 (1947) https://doi.org/10.1214/aoms/1177730491

[23] Wilcoxon, F.: Probability Tables for Individual Comparisons by Ranking Methods. Biometrics **3**(3), 119–122 (1947) https://doi.org/10.2307/3001946

[24] Fuentes-Pérez, J.F., Quaresma, A.L., Pinheiro, A., Sanz-Ronda, F.J.: OpenFOAM vs FLOW-3D: A comparative study of vertical slot fishway modelling. Ecological Engineering **174**, 106446 (2022) https://doi.org/10.1016/j.ecoleng.2021.106446

[25] Gaur, S., Singh, R., Bandyopadhyay, A., Singh, R.: Diagnosis of GCM-RCM-driven rainfall patterns under changing climate through the robust selection of multi-model ensemble and sub-ensembles. Climatic Change **176**(2), 13 (2023) https://doi.org/10.1007/s10584-022-03475-z

[26] Newman, T., Borker, R., Aubiniere-Robb, L., Hendrickson, J., Choudhury, D., Halliday, I., Fenner, J., Narracott, A., Hose, D.R., Gosling, R., Gunn, J.P., Morris,

P.D.: Rapid virtual fractional flow reserve using 3D computational fluid dynamics. European Heart Journal-Digital Health, 028 (2023) https://doi.org/10.1093/ehjdh/ztad028

[27] Herho, S.H.S., Herho, K.E.P., Susanto, R.D.: Did hydroclimate conditions contribute to the political dynamics of Majapahit? A preliminary analysis. Geographica Pannonica **27**(3), 199–210 (2023) https://doi.org/10.5937/gp27-44682

[28] Uchikawa, H., Kin, T., Koizumi, S., Sato, K., Uchida, T., Takeda, Y., Koike, T., Kiyofuji, S., Yamashiro, S., Mukasa, A., Nobuhito, S.: Aneurysmal Inflow Rate Coefficient Predicts Ultra-early Rebleeding in Ruptured Intracranial Aneurysms: Preliminary Report of a Computational Fluid Dynamics Study. Neurologia medico-chirurgica **63**(10), 450–456 (2023) https://doi.org/10.2176/jns-nmc.2023-0003

[29] Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S.J., Wilson, M.B.J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, , Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., Mulbregt, P., Contributors, S...: SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods **17**(3), 261–272 (2020) https://doi.org/10.1038/s41592-019-0686-2

[30] Waskom, M.L.: seaborn: statistical data visualization. Journal of Open Source Software **6**(60), 3021 (2021) https://doi.org/10.21105/joss.03021