Time-slices based training of Physics-Informed Neural Networks for 3D Non-Stationary Thermal Inversion Simulations

Maciej Sikora[1][0009−0006−4465−2395],
Maciej Paszyński[1][0000−0001−7766−6052]

[1]AGH University of Krakow, Poland
maciejsikora@agh.edu.pl, paszynsk@agh.edu.pl

Abstract. During the day, the temperature in urban areas rises with height, leading to a negative vertical temperature gradient that causes the warm air near the ground to rise. While daytime convection facilitates mixing, nocturnal radiative cooling creates thermal inversions that trap emissions near the surface. This paper models this transition using time-dependent, three-dimensional advection-diffusion equations solved via Physics-Informed Neural Networks (PINNs). We propose a sequential training strategy over a four-dimensional space-time domain, partitioning the temporal axis into discrete segments to ensure convergence. In this framework, the terminal state of each segment provides the initial conditions for the subsequent slice, maintaining physical continuity. Our findings demonstrate that this PINN-based approach effectively captures the accumulation of ground-level pollutants during inversion events, achieving high numerical accuracy with significantly lower computational overhead compared to traditional grid-based Eulerian solvers.

Keywords: · Physics-Informed Neural Networks · Non-stationary advection-diffusion problem · Atmospheric simulations

1 Introduction

Air pollution in urban valleys, such as the Kraków area, is a significant health risk factor [15]. Modeling the stagnation of these pollutants is essential to mitigate negative effects on human health [7]. During the night, radiative cooling of the ground alters the vertical temperature profile, resulting in a thermal inversion. This meteorological phenomenon traps particulate-laden cloud vapor near the surface, preventing advective dispersion and leading to hazardous accumulations of pollution, as illustrated during a typical November night in Kraków (Figure 1). Accurately modeling this phenomenon requires capturing the underlying 3D advection-diffusion dynamics over extended periods.

Simulating such non-stationary 3D thermal inversions is computationally demanding. While traditional numerical methods—such as the isogeometric finite element method have been employed in two dimensions, their extension into

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_5

3D time-dependent spaces incurs substantial computational overhead. Physics-Informed Neural Networks (PINNs) [13,4] offer a promising mesh-free alternative. However, standard PINNs trained continuously over long, four-dimensional space-time domains frequently suffer from gradient pathologies and error propagation, which impede the stable simulation of transient atmospheric behavior. Although some time-decomposition algorithms exist, formulating a robust, accessible implementation that effectively segments 3D space-time while strictly maintaining physical continuity remains a notable gap in contemporary literature.

To address this challenge, we introduce a novel open-source PINN framework specifically designed for simulating time-dependent 3D thermal inversion phenomena. Our contribution significantly advances standard monolithic PINN capabilities by implementing a sequentially partitioned 4D space-time training strategy. We train modular neural networks on discrete temporal slices, explicitly coupling adjacent segments to preserve continuity and rigidly enforcing local initial conditions. A final neural network then projects these piecemeal solutions into a unified global approximation. This method curtails optimization stiffness over long time horizons, successfully extending earlier 2D thermal inversion models [9,14] into fully three-dimensional environments. We wrap this methodology in a user-friendly, cloud-compatible Google Colab toolkit that simplifies residual and boundary-condition configurations, eschewing the verbose boilerplate of traditional implementations.

The primary objective of this paper is to demonstrate the efficacy and robustness of our time-sliced PINN architecture in capturing the complex dynamics of 3D nocturnal thermal inversions. By modeling the advection-diffusion of pollution particles driven by characteristic vertical temperature profiles, we validate our proposed framework's ability to accurately and efficiently reproduce ground-level pollution stagnation.

2 Training algorithm for Three-dimensional time dependent Physics-Informed Neural Networks

We divide the time interval $[0, T]$ into N_{seg} time segments. We use the segment index $k \in \{1, \ldots, N_{seg}\}$ for the time segments, where each represents the time interval $\Delta t_k = [t_{k-1}, t_k]$. We create N_{seg} independent neural networks, where $\hat{u}^{(k)}$ denotes the neural network trained on the k-th slice. For each time slice, we create a mesh of collocation points in the spatial domain Ω for the training of the related neural network inside the slice. For the first segment ($k = 1$), we include the given initial condition. Once the training is finished, we extract the solution at the final time interval point, and we start the training for the next segment using the continuity condition. Specifically, the initial condition for slice $k > 1$ is $\hat{u}^{(k)}(\mathbf{x}, t_{k-1}) = \hat{u}^{(k-1)}(\mathbf{x}, t_{k-1})$. Having finished the training for all the segments, we create the final neural network and smooth the family of time-slice solutions by training a single final projection solution.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_5

Fig. 1: The photo of the pollution near the ground in the Kraków valley in November 2025 (photo by Maciej Paszyński).

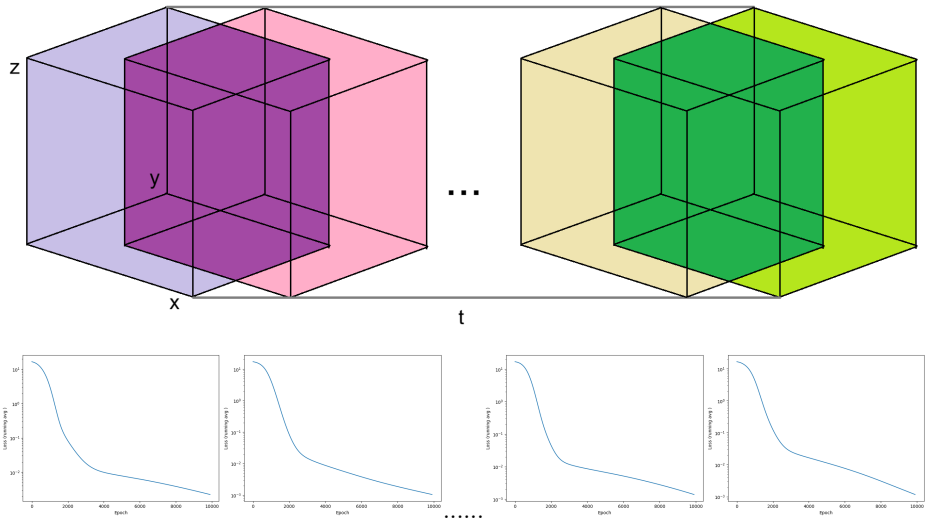

Fig. 2: Time segments for transient simulation.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_5

3 Physics Informed Neural Network for transient problems on the example of three-dimensional heat transfer problem

Let us consider a strong form of the exemplary transient PDE, the heat transfer problem. Find $\hat{u} \in C^3(\Omega \times [0,T])$ for $\mathbf{x} = (x,y,z)^T \in \Omega = [0,1]^3$, $t \in [0,T]$ such that

$$\underbrace{\partial_t \hat{u}(\mathbf{x},t;\theta)}_{\text{time evolution}} \underbrace{-\epsilon \Delta \hat{u}(\mathbf{x},t;\theta)}_{\text{diffusion term}} = \underbrace{f(\mathbf{x},t)}_{\text{forcing}}, \quad (\mathbf{x},t) \in \Omega \times [0,T], \tag{1}$$

With the initial condition $\hat{u}(\mathbf{x},0;\theta) = u_0(\mathbf{x})$ and zero Dirichlet boundary condition $\hat{u}(\mathbf{x},t;\theta) = 0$ for $\mathbf{x} \in \partial\Omega, t \in (0,T)$. In the PINN approach, the neural network $\hat{u}(\mathbf{x},t;\theta)$ represents the solution approximation, namely,

$$\hat{u}(\mathbf{x},t;\theta) = (\mathcal{A}_L \circ \sigma \circ \mathcal{A}_{L-1} \circ \cdots \circ \sigma \circ \mathcal{A}_1)(\mathbf{v}), \tag{2}$$

where $\mathbf{v} = [x,y,z,t]^T$ is the input vector, $\mathcal{A}_i(\mathbf{z}) = \mathbf{W}_i\mathbf{z} + \mathbf{b}_i$ represents the affine transformation of the i-th layer, and σ is the activation function (sigmoid), which, as we have shown in [8], is the best choice for PINN. We define the total loss function to optimize the network weights and biases as $\mathcal{L}_{total} = \omega_r \mathcal{L}_{res} + \omega_b \mathcal{L}_{bc}$, where ω_r and ω_b are the respective weights. The residual loss $\mathcal{L}_{res}$ is evaluated over N_p collocation points $\{(\mathbf{x}_j,t_j)\}_{j=1}^{N_p}$:

$$\mathcal{L}_{res} = \frac{1}{N_p} \sum_{j=1}^{N_p} \left| \mathcal{F}(\hat{u}(\mathbf{x}_j,t_j;\theta)) \right|^2, \tag{3}$$

where $\mathcal{F}$ is the PDE operator defined in the strong form, $\mathcal{F}(\hat{u}) = \partial_t \hat{u} - \epsilon \Delta \hat{u} - f$. We also define the zero Dirichlet boundary condition loss, computed over N_{bc} boundary points $\{(\mathbf{x}_j,t_j)\}_{j=1}^{N_{bc}}$:

$$\mathcal{L}_{bc} = \frac{1}{N_{bc}} \sum_{j=1}^{N_{bc}} \left| \hat{u}(\mathbf{x}_j,t_j;\theta) \right|^2. \tag{4}$$

The initial conditions are enforced strongly by modifying the output of the neural network.

```
# Enforce strong initial condition
if self.previous_pinn is None:
  # First segment
  init_val = initial_condition_first_segment(x, y, z)
else:
  # Subsequent segments: use previous PINN at self.t_start
  t_init = torch.full_like(t, self.t_start)
  init_val = f(self.previous_pinn, x, y, z, t_init).detach()
logits = logits * (t - self.t_start)**2 + torch.exp(-1000*(t-
    self.t_start)**2) * init_val
return logits
```

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_5

4 Physics Informed Neural Network for transient atmospheric problems modeled with advection-diffusion equations

Let us consider the strong form of the transient advection-diffusion problem. Find $\hat{u} \in C^3(\Omega \times [0,T])$ for $\mathbf{x} \in \Omega = [0,1]^3$, $t \in [0,T]$ such that

$$\underbrace{\partial_t \hat{u}(\mathbf{x},t;\theta)}_{\text{time evolution}} \underbrace{-\epsilon \Delta \hat{u}(\mathbf{x},t;\theta)}_{\text{diffusion term}} + \underbrace{\mathbf{b}(\mathbf{x},t) \cdot \nabla \hat{u}(\mathbf{x},t;\theta)}_{\text{advection term}} = \underbrace{f(\mathbf{x},t)}_{\text{forcing}}, \quad (\mathbf{x},t) \in \Omega \times [0,T], \tag{5}$$

with the initial condition $\hat{u}(\mathbf{x},0;\theta) = u_0(\mathbf{x})$ and the zero-Neumann boundary condition $\frac{\partial \hat{u}(\mathbf{x},t;\theta)}{\partial n} = 0$ for $\mathbf{x} \in \partial\Omega, t \in [0,T]$. As in the previous section, the neural network $\hat{u}(\mathbf{x},t;\theta)$ approximates the solution and is structured with identical affine transformations and sigmoid activation functions. Optimization determines the network's weights and biases θ by minimizing the total loss function:

$$\mathcal{L}_{total} = \omega_r \mathcal{L}_{res} + \omega_b \mathcal{L}_{bc} + \omega_i \mathcal{L}_{init}. \tag{6}$$

The residual loss $\mathcal{L}_{res}$ is evaluated over N_p collocation points $\{(\mathbf{x}_j, t_j)\}_{j=1}^{N_p}$:

$$\mathcal{L}_{res} = \frac{1}{N_p} \sum_{j=1}^{N_p} \left| \mathcal{F}(\hat{u}(\mathbf{x}_j, t_j; \theta)) \right|^2, \tag{7}$$

where $\mathcal{F}$ is the PDE operator corresponding to the strong form, $\mathcal{F}(\hat{u}) = \partial_t \hat{u} - \epsilon \Delta \hat{u} + \mathbf{b} \cdot \nabla \hat{u} - f$. We define the Neumann boundary condition loss over N_{bc} boundary points $\{(\mathbf{x}_j, t_j)\}_{j=1}^{N_{bc}}$:

$$\mathcal{L}_{bc} = \frac{1}{N_{bc}} \sum_{j=1}^{N_{bc}} \left| \frac{\partial \hat{u}(\mathbf{x}_j, t_j; \theta)}{\partial n} \right|^2. \tag{8}$$

Unlike the strong initial-condition enforcement employed in Section 3 via structural output modifications, here we formulate a separate initial-condition loss term $\mathcal{L}_{init}$. This soft constraint approach affords the optimizer additional flexibility to balance the complex interplay between advective transport, diffusion, and the initial state. This flexibility is particularly beneficial for stabilizing training when dealing with sharp initial distribution gradients ubiquitous to pollution concentration fields. The initial condition loss is computed over N_{init} initial points $\{(\mathbf{x}_j, 0)\}_{j=1}^{N_{init}}$:

$$\mathcal{L}_{init} = \frac{1}{N_{init}} \sum_{j=1}^{N_{init}} \left| \hat{u}(\mathbf{x}_j, 0; \theta) - u_0(\mathbf{x}_j) \right|^2. \tag{9}$$

5 Setting up the code

Our code is available at[1]

[1] https://colab.research.google.com/drive/1T_J7jYslg9LI0iCjh6IuEVqOk2nz09SX

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_5

The model parameters are fundamentally grouped by their distinct operational roles:

- **Domain Parameters**
  - `LENGTH`: Defines the dimension L of the spatial domain uniformly across all three space axes (x, y, z), yielding $\Omega = [0, L]^3$.
  - `TOTAL_TIME`: Defines the temporal dimension T, establishing the total space-time domain $\Omega \times [0, T]$. Setting this explicitly disambiguates the time parameter from spatial lengths.
- **Sampling Parameters**
  - `NUMBER_OF_SEGMENTS`: Corresponds to N_{seg}, partitioning the temporal axis into discrete sequential slices for consecutive model training.
  - `N_POINTS`: Designates the number of points sampled per individual axis per time slice. A strictly 3D spatial plus temporal sampling scheme systematically selects uniformly distributed collocation points equally along the x, y, z, and t axes within each respective segment.
  - `N_POINTS_PLOT`: Specifies the evaluation sampling resolution used specifically for rendering the output diagnostics and 3D plots.
- **Loss Weights**
  - `WEIGHT_RESIDUAL`, `WEIGHT_INITIAL`, `WEIGHT_BOUNDARY`: These constants respectively map to ω_r, ω_i, and ω_b, dynamically scaling the relative importance of the PDE residual, initial-condition coupling, and Neumann boundary constraints.
- **Architecture Parameters**
  - `LAYERS`, `NEURONS_PER_LAYER`: Define the depth and width topology of the hidden layers. An identical neural architecture is initialized for each temporal segment.
- **Optimization Parameters**
  - `EPOCHS`, `LEARNING_RATE`: Determine the training duration (evaluations per segment) and the gradient descent step size governed by the optimization algorithm.

The default setup of the parameters for our simulations is as follows:

```
LENGTH = 1. # Domain size across spatial axes (x, y, z).
TOTAL_TIME = 1. # Domain size along the temporal axis.
NUMBER_OF_SEGMENTS = 10  # Selected number of discrete time
    slices.
N_POINTS = 10  # Number of collocation points per individual
    axis within a segment.
# Number of probing points per axis for plotting the solution
N_POINTS_PLOT = N_POINTS * NUMBER_OF_SEGMENTS * 10
WEIGHT_RESIDUAL = 1.0  # Loss coefficient for the PDE
    operator
WEIGHT_INITIAL = 10.0  # Loss coefficient establishing
    sequential segment coupling
WEIGHT_BOUNDARY = 0.1  # Loss coefficient for boundary
    conditions
```

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_5

```
LAYERS = 2  # Neural network hidden depth
NEURONS_PER_LAYER = 50  # Hidden units per layer
EPOCHS = 5000  # Optimization iterations (multiplied by the
    number of segments)
LEARNING_RATE = 0.002
```

These default values were chosen based on practical experiments:

- `LENGTH` and `TOTAL_TIME` are set to 1.0 to keep the domain normalized, which simply makes the gradients easier to scale.
- `NUMBER_OF_SEGMENTS` is set to 10 so the neural network doesn't have to learn the entire long time-frame all at once, which usually causes training to fail.
- `N_POINTS` is 10 because a $10 \times 10 \times 10 \times 10$ grid (for x, y, z, and t) gives $10,000$ points per slice. This provides enough data for the network to learn without running out of memory.
- The loss weights are intentionally unbalanced. Setting `WEIGHT_INITIAL` to 10.0 forces each new time slice to strictly match the end of the previous one. Setting `WEIGHT_BOUNDARY` to 0.1 keeps boundary constraints from interfering too much with the main advection-diffusion physics.
- A small network (`LAYERS=2`, `NEURONS_PER_LAYER=50`) trained for 5000 `EPOCHS` per slice is just expressive enough to capture the physics without overfitting to the training points.

6 Implementation of the heat transfer simulation

In this example, we aim to model the heat transfer simulation over a three-dimensional cube-shaped domain.

In the pure heat transfer equations in strong form, we look for the temperature scalar field $\hat{u}(\mathbf{x}, t; \theta) \in \mathbb{R}$ such that:

$$\partial_t \hat{u}(\mathbf{x}, t; \theta) - \epsilon \Delta \hat{u}(\mathbf{x}, t; \theta) = f(\mathbf{x}, t), \quad (\mathbf{x}, t) \in \Omega \times (0, T], \tag{10}$$

subject to zero Dirichlet boundary conditions and an initial state u_0. The residual loss $\mathcal{L}_{res}$ is symmetrically minimized dynamically across continuous domain collocation points. This translates to a PyTorch representation (abstracting boilerplate data handlers):

```
def residual_loss(self, pinn: PINN):
    x, y, z, t = get_interior_points(self.x_domain, self.
        y_domain, self.z_domain, self.t_domain, self.n_points
        )
    loss = (dfdt(pinn, x, y, z, t)
            - self.epsilon * (dfdx(pinn, x, y, z, t, order=2)
                              + dfdy(pinn, x, y, z, t, order
                                  =2)
                              + dfdz(pinn, x, y, z, t, order
                                  =2))
            - self.source(x, y, z, t))
    return loss.pow(2).mean()
```

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_5

We add the definitions of the `Kx`, `Ky`, and `bx`, `by` variables into the `Loss` class.

The respective boundary constraints are constructed by mapping evaluation targets linearly along the six faces of the 3D hypercube:

```
def boundary_loss(self, pinn: PINN):
    faces = get_boundary_points(self.x_domain, self.y_domain,
        self.z_domain, self.t_domain, self.n_points)
    loss = sum(f(pinn, x, y, z, t).pow(2).mean() for x, y, z,
        t in faces)
    return loss
```

and the initial state

```
def initial_condition_first_segment(x: torch.Tensor, y: torch
    .Tensor, z: torch.Tensor, t: torch.Tensor = None, pinn:
    PINN = None) -> torch.Tensor:
  r = torch.sqrt((x-0.5)**2 + (y-0.5)**2  + (z-0.5)**2)
  res = -100* (r < 0.25)
  return res
```

For the next segments, we will return the solution obtained from the previous segment

```
def initial_condition_next_segment(x: torch.Tensor, y: torch.
    Tensor, z: torch.Tensor, t: torch.Tensor = None, pinn:
    PINN = None) -> torch.Tensor:
   return f(pinn, x, y, z, t)
```

7 Implementation of the thermal inversion simulation

In this example, we aim to model the thermal inversion in a three dimensional domain using the advection-diffusion equations. The field solution u represents the cloud vapor mixed with pollution particles. The initial configuration is assumed to be zero concentration field. The source represents the evaporation of the fog particles from the ground, that mixed with the pollution particles. The movement of the cloud particles is enforced by the vertical thermal gradient. Following [1], we define $\frac{\partial T}{\partial z} = -2$ close to the ground ($z < 0.5$), and $\frac{\partial T}{\partial z} = 2$ for the higher regions of the domain ($z > 0.5$). In the transient advection-diffusion problem governing thermal inversion, we evaluate the scalar field equivalent utilizing spatial domains. Given $\Omega \ni \mathbf{x} \to \hat{u}(\mathbf{x}, t; \theta) \in \mathbb{R}$, we have:

$$\partial_t \hat{u}(\mathbf{x}, t; \theta) + \mathbf{b}(\mathbf{x}, t) \cdot \nabla \hat{u}(\mathbf{x}, t; \theta) - \epsilon \Delta \hat{u}(\mathbf{x}, t; \theta) = f(\mathbf{x}, t), \quad (\mathbf{x}, t) \in \Omega \times (0, T], \tag{11}$$

subject to zero Neumann boundary conditions ($\nabla \hat{u} \cdot n = 0$) and a zero initial state u_0.

```
def residual_loss(self, pinn: PINN):
    x, y, z, t = get_interior_points(self.x_domain, self.
        y_domain, self.z_domain, self.t_domain, self.n_points
        )
```

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_5

```
    loss = (dfdt(pinn, x, y, z, t)
            + self.dTz(z, t) * dfdz(pinn, x, y, z, t)
            - self.Kx * dfdx(pinn, x, y, z, t, order=2)
            - self.Ky * dfdy(pinn, x, y, z, t, order=2)
            - self.Kz * dfdz(pinn, x, y, z, t, order=2)
            - self.source(x, y, z, t))
    return loss.pow(2).mean()
```

We add the definitions of the `Kx`, `Ky`, and `bx`, `by` variables to the `Loss` class. In the thermal inversion simulations, the vertical diffusion is assumed to be one hundred times stronger than the horizontal diffusion; namely, `Kx=Ky=0.01`, and `Kz=1.0`.

We set up the zero Neumann boundary condition,

```
def boundary_loss(self, pinn: PINN):
    faces = get_boundary_points(self.x_domain, self.y_domain,
         self.z_domain, self.t_domain, self.n_points)
    loss = (dfdy(pinn, *faces[0]).pow(2).mean() + dfdy(pinn,
        *faces[1]).pow(2).mean() +
            dfdx(pinn, *faces[2]).pow(2).mean() + dfdx(pinn,
                *faces[3]).pow(2).mean() +
            dfdz(pinn, *faces[4]).pow(2).mean() + dfdz(pinn,
                *faces[5]).pow(2).mean())
    return loss
```

and zero initial state

```
def initial_condition_first_segment(x: torch.Tensor, y: torch
    .Tensor, z: torch.Tensor, t: torch.Tensor = None, pinn:
    PINN = None) -> torch.Tensor:
    return x*0
```

8 Numerical results

At this stage of development, our framework serves primarily as a qualitative proof of concept demonstrating the viability of time-sliced PINNs for tracking complex 3D atmospheric dynamics. While rigorous quantitative benchmark comparisons, exhaustive convergence metrics, and formal sensitivity analyses are slated for future studies, the current visual results successfully demonstrate the foundational capacity of our sequentially integrated models to capture macroscopic physical behaviors without succumbing to the optimization pathologies typical of monolithic PINN training over long time horizons.

The pure, isotropic heat transfer simulations model an initial localized high-temperature sphere radiating into an otherwise cool surrounding medium. Representative snapshots of the central cross-section are presented in Figure 3. As expected physically, the heat propagates outward symmetrically and diminishes in peak intensity smoothly as time progresses, validating the correct temporal transfer mechanics of our methodology.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_5

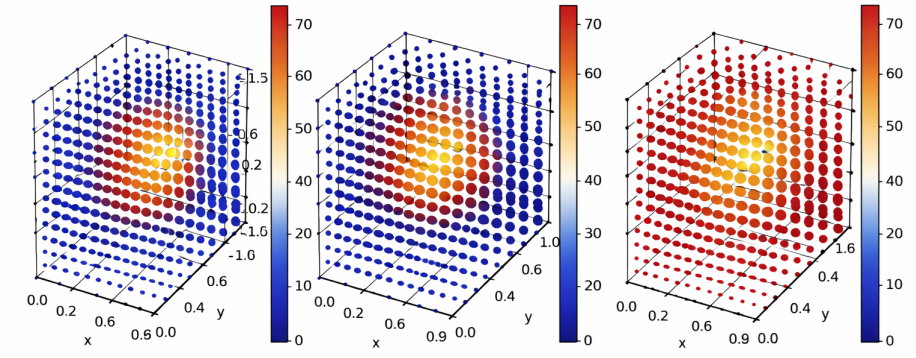

Fig. 3: Heat transfer simulation. Snapshots from the cross-section of the domain illustrating uniform outward diffusion over time.

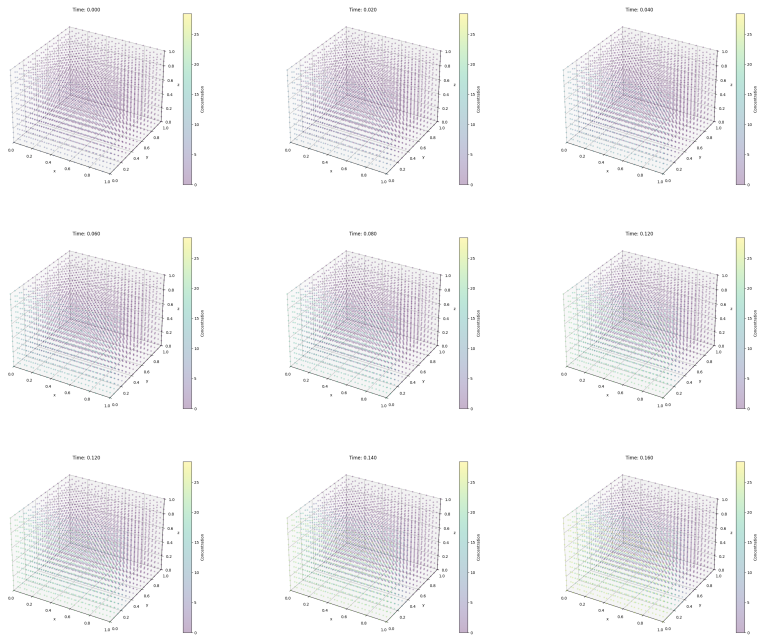

Fig. 4: Thermal inversion simulation. 3D view of the pollution concentration field. The ground is on the back side of the box. View from the top.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_5

For the non-stationary thermal inversion scenarios, visual outputs are divided into three-dimensional and cross-sectional representations. The full three-dimensional volumetric snapshots of the concentration field are presented accurately in Figure 4. This specific figure renders a top-down view of the domain, where the localized ground level resides along the rear face of the bounding box. To afford better visibility into the interior gradient behavior longitudinally, Figure 5 explicitly isolates planar cross-sections of the domain.

In these thermal inversion simulations, the computational domain vividly captures how cloud vapor evaporating from the ground accumulates and structurally stagnates tightly at low altitudes. This macroscopic stagnation is strictly forced by the prescribed bidirectional vertical temperature gradient directing the governing advection limits. Ultimately, these qualitative results successfully reproduce the macroscopic atmospheric stagnation phenomenon frequently observed in topographical basins like the Kraków area (as shown empirically in Figure 6). The simulated numerical advection correctly demonstrates that morning evaporation, carrying dense pollution particles, remains structurally trapped near the ground surface due to nocturnal radiative temperature inversions.

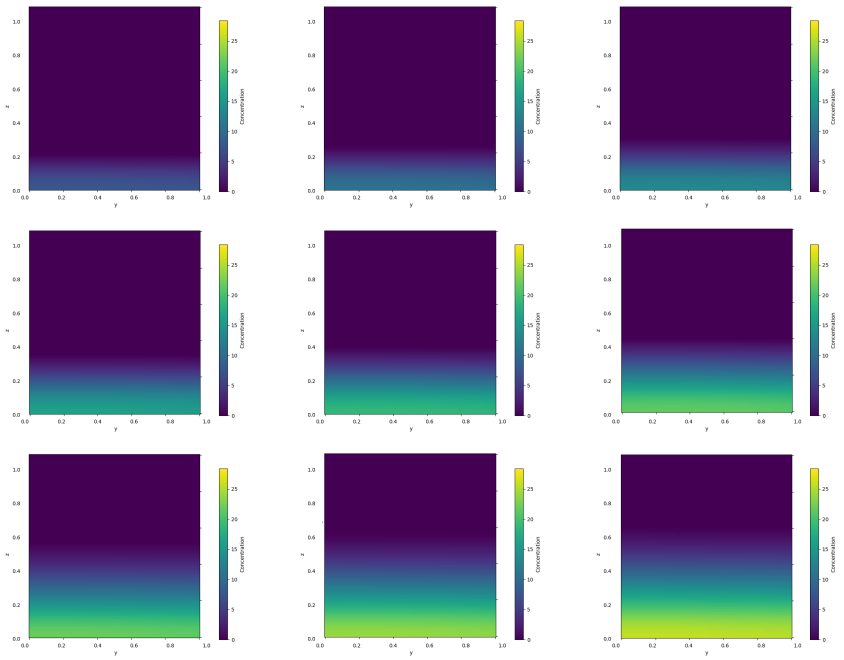

Fig. 5: Thermal inversion simulation. Snapshots from the cross-section of the domain

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_5

Fig. 6: The photo of the pollution near the ground in the Kraków valley in December 2025 (photo by Maciej Paszyński).

9 Conclusions

This study presents a functional proof-of-concept for using Physics-Informed Neural Networks (PINNs) to solve 3D, time-dependent advection-diffusion problems. By using a time-slicing approach—which breaks the simulation into shorter, connected time segments before smoothing the final results—we avoid many of the training failures that usually occur when PINNs try to handle long spans of time all at once.

Our initial simulations effectively illustrate how thermal inversions prevent cloud vapor and trapped pollution particles from dispersing upwards. Although the current model successfully captures this core stagnation behavior, it remains a foundational step. Further physical validation, formal benchmarking, and rigorous sensitivity testing will be needed before it can be used as a fully reliable predictive tool.

Looking ahead, we plan to expand this framework to simulate active methods for clearing local smog. Specifically, we will model the generation and propagation of localized shock waves—similar to those produced by anti-hail cannons—to test if they can theoretically break up these trapped pollution layers [10,17]. We also plan to compare the Physics Informed Neural Network solvers to isogeometric finite element method solvers [16] or *hp*-adaptive finite element method [11,12,5]. We may also consider the application of inverse solvers [3,6,2] for a better fit of the model parameters.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_5

Acknowledgments

This work has been supported by the National Science Centre, Poland grant no. 2025/57/B/ST6/00058.

References

1. U.S. Standard Atmosphere vs. Altitude, Engineering Toolbox (2003), https://www.engineeringtoolbox.com/standard-atmosphere-d_604.html
2. Barabasz, B., Gajda-Zagórska, E., Migórski, S., Paszyński, M., Schaefer, R., Smołka, M.: A hybrid algorithm for solving inverse problems in elasticity. Int. J. Appl. Math. Comput. Sci. **24**(4), 865–886 (Dec 2014). https://doi.org/10.2478/amcs-2014-0064, https://doi.org/10.2478/amcs-2014-0064
3. Barabasz, B., Migórski, S., Schaefer, R., Paszyński, M.: Multi-deme, twin adaptive strategy hp-hgs. Inverse Problems in Science and Engineering **19**(1), 3–16 (2011). https://doi.org/10.1080/17415977.2010.531477
4. Cai, S., Mao, Z., Wang, Z., Yin, M., Karniadakis, G.E.: Physics-informed neural networks (PINNs) for fluid mechanics: A review. Acta Mechanica Sinica **37**(12), 1727–1738 (2021)
5. Demkowicz, L., Gatto, P., Kurtz, J., Paszyński, M., Rachowicz, W., Bleszyński, E., Bleszyński, M., Hamilton, M., Champlin, C., Pardo, D.: Modeling of bone conduction of sound in the human head using hp-finite elements: Code design and verification. Computer Methods in Applied Mechanics and Engineering **200**(21), 1757–1773 (2011). https://doi.org/https://doi.org/10.1016/j.cma.2011.02.001, https://www.sciencedirect.com/science/article/pii/S0045782511000363
6. Gajda-Zagórska, E., Schaefer, R., Smołka, M., Paszyński, M., Pardo, D.: A hybrid method for inversion of 3d dc resistivity logging measurements. Natural computing **14**(3), 355–374
7. Guarnieri, M., Balmes, J.R.: Outdoor air pollution and asthma. The Lancet **383**(9928), 1581–1592 (2014). https://doi.org/https://doi.org/10.1016/S0140-6736(14)60617-6
8. Maczuga, P., Paszyński, M.: Influence of activation functions on the convergence of physics-informed neural networks for 1d wave equation. In: Mikyška, J., de Mulatier, C., Paszynski, M., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M. (eds.) Computational Science – ICCS 2023. pp. 74–88. Springer Nature Switzerland, Cham (2023)
9. Maczuga, P., Sikora, M., Służalec, T., Szubert, M., Sztangret, Ł., Szeliga, D., Łoś, M., Dzwinel, W., Pingali, K., Paszyński, M.: Physics Informed Neural Network Code for 2d Transient Problems (PINN-2DT) compatible with Google Colab. In: Lees, M.H., Cai, W., Cheong, S.A., Su, Y., Abramson, D., Dongarra, J.J., Sloot, P.M.A. (eds.) Computational Science – ICCS 2025. pp. 177–191. Springer Nature Switzerland, Cham (2025)
10. Misan, K., Kozieja, M., Łoś, M., Gryboś, D., Leszczyński, J., Maczuga, P., Woźniak, M., Serra, A.O., Paszyński, M.: The first scientiffic evidence for the hail cannon. In: Mikyška, J., de Mulatier, C., Paszynski, M., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M. (eds.) Computational Science – ICCS 2023. pp. 177–190. Springer Nature Switzerland, Cham (2023)

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_5

11. Pardo, D., Demkowicz, L., Torres-Verdín, C., Paszynski, M.: A self-adaptive goal-oriented hp-finite element method with electromagnetic applications. part ii: Electrodynamics. Computer Methods in Applied Mechanics and Engineering **196**(37), 3585–3597 (2007). https://doi.org/https://doi.org/10.1016/j.cma.2006.10.016, https://www.sciencedirect.com/science/article/pii/S0045782507000898, special Issue Honoring the 80th Birthday of Professor Ivo Babuška
12. Paszyński, M.: On the parallelization of self-adaptive hp-finite element methods part ii. partitioning communication agglomeration mapping (pcam) analysis. Fundamenta Informaticae **93**(4), 435–457 (2009)
13. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics **378**, 686–707 (2019)
14. Sikora, M., Oliver-Serra, A., Siwik, L., Leszczyńska, N., Ciesielski, T.M., Valseth, E., Leszczyński, J., Paszyńska, A., Paszyński, M.: Graph grammars and Physics Informed Neural Networks for simulating of pollution propagation on Spitzbergen. Applied Soft Computing **182**, 113394 (2025). https://doi.org/https://doi.org/10.1016/j.asoc.2025.113394, https://www.sciencedirect.com/science/article/pii/S1568494625007057
15. Traczyk, P., Gruszecka-Kosowska, A.: The condition of air pollution in Kraków, Poland, in 2005–2020, with health risk assessment. International Journal of Environmental Research and Public Health **17**(17) (2020). https://doi.org/10.3390/ijerph17176063, https://www.mdpi.com/1660-4601/17/17/6063
16. Woźniak, M., Łoś, M., Paszyński, M., Dalcin, L., Calo, V.M.: Parallel fast isogeometric solvers for explicit dynamics. Computing and Informatics **36**(2), 423–448 (Jun 2017), https://www.cai.sk/ojs/index.php/cai/article/view/2017_2_423
17. Łoś, M., Siwik, L., Woźniak, M., Gryboś, D., Maczuga, P., Oliver-Serra, A., Leszczyński, J., Paszyński, M.: Shock waves generators: From prevention of hail storms to reduction of the smog in urban areas — experimental verification and numerical simulations. Journal of Computational Science **77**, 102238 (2024). https://doi.org/https://doi.org/10.1016/j.jocs.2024.102238, https://www.sciencedirect.com/science/article/pii/S1877750324000310

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_5