

Graph grammars and Physics Informed Neural Networks for simulating of pollution propagation on Spitzbergen

Maciej Sikora^a, Albert Oliver-Serra^b, Leszek Siwik^a, Natalia Leszczyńska^c,
Tomasz Maciej Ciesielski^d, Eirik Valseth^{e,f}, Jacek Leszczyński^a, Anna Paszyńska^g,
Maciej Paszyński^a^{*}

^a AGH University of Krakow, Faculty of Computer Science, Al. Mickiewicza 30, Kraków, 30-059, Poland

^b University Institute of Intelligent Systems and Numeric Applications in Engineering, University of Las Palmas de Gran Canaria (ULPGC), Las Palmas de Gran Canaria, Spain

^c Medical University of Silesia-Katowice, Faculty of Medical Sciences, Katowice, Poland

^d The University Centre in Svalbard, Longyearbyen, Norway

^e Norwegian University of Life Sciences, Ås, Norway

^f Simula Research Laboratory, Oslo, Norway

^g Jagiellonian University, Faculty of Physics, Astronomy and Applied Computer Science, Łojasiewicza 11, Kraków, 30-348, Poland

ARTICLE INFO

Keywords:

Graph grammar model
Physics Informed Neural Networks
Pollution propagation simulations
Advection-diffusion equations
Pollution modeling

ABSTRACT

In this paper, we present two computational methods for performing simulations of pollution propagation described by advection-diffusion equations. The first method employs graph grammars to describe the generation process of the computational mesh used in simulations with the meshless solver of the three-dimensional finite element method. The graph transformation rules express the three-dimensional Rivara longest-edge refinement algorithm. This solver is used for an exemplary application: performing three-dimensional simulations of pollution generation by the recently closed coal-burning power plant and the new diesel power plant, the capital of Spitzbergen. The second computational code is based on the Physics Informed Neural Networks method. It is used to calculate the dissipation of the pollution along the valley in which the city of Longyearbyen is located. We discuss the instantiation and execution of the PINN method using Google Colab implementation. There are four novelties of our paper. First, we show a lower computational cost of the proposed graph grammar model in comparison with the mesh transformations over the computational mesh. Second, we discuss the benefits and limitations of the PINN implementation of the non-stationary advection-diffusion model with respect to finite element method solvers. Third, we introduce the PINN code for non-stationary thermal inversion simulations. Fourth, using our computer simulations, we estimate the influence of the pollution from power plants on the Spitzbergen inhabitants.

1. Introduction

The subject of our article is the presentation of two computational methods related to simulations of pollution propagation using advection-diffusion equations. The first approach's novelty lies in using the original model of graph grammars, a set of rules describing transformations of the graph describing the computational grid. Based on the generated computational grid, the simulation of pollution propagation is carried out using the finite element method, a matrix-free solver, and the assembly of the matrices for the iterative solver from local element matrices. For the second approach, we introduce the original implementation of the Physics Informed Neural Networks method in Google Colab, coupled with the results of the finite element method

solver, enabling the training of a neural network predicting pollution propagation in the long-term range based on the phenomenon of thermal inversion. In our simulations, we apply the concepts to a real-world scenario: pollution propagation in the city of Longyearbyen, the capital of Spitzbergen, where air pollution was generated by a coal-burning power plant recently replaced by the diesel engine power plant. This practical application underscores the relevance and importance of our research.

For the simulations of the pollution propagation from the power plants, we employ a finite element solver for the advection-diffusion equation [1] stabilized with the Streamlined-Upwind-Petrov-Galerkin (SUPG) method [2].

^{*} Corresponding author.

E-mail address: paszynsk@agh.edu.pl (M. Paszyński).

<https://doi.org/10.1016/j.asoc.2025.113394>

Received 12 September 2024; Received in revised form 22 May 2025; Accepted 30 May 2025

Available online 16 July 2025

1568-4946/© 2025 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

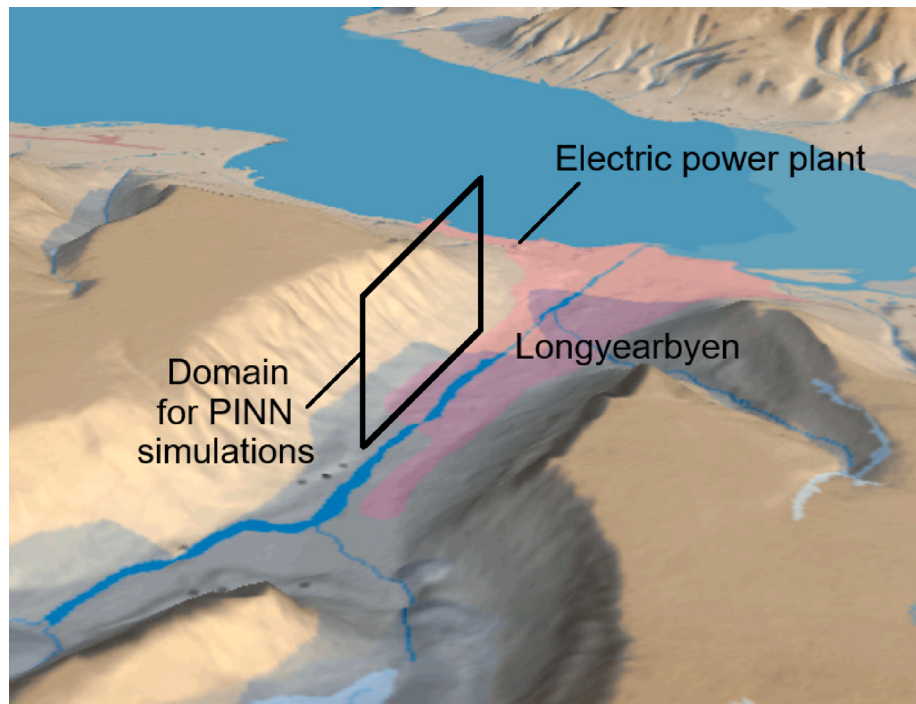


Fig. 1. The town of Longyearbyen at Spitzbergen, the location of the recently closed coal-burning power plant, was replaced by the diesel power plant. The computational domain for finite element method solver involves the entire domain. The computational domain for PINN simulations of thermal inversion involves the black rhombus.

The computational mesh for the simulations comes from a new graph grammar-based mesh generator, implemented in Julia, for a sequence of mesh refinements built with tetrahedral finite elements. Our graph grammar model expresses the three-dimensional version of the longest-edge refinement algorithm. The longest-edge refinement algorithm has been initially proposed for two-dimensional grids by Cecilia Rivara [3,4]. The graph grammar-based mesh refinements for two-dimensional grids have been employed and discussed in [1], and in [5–8] with the hanging nodes version. The Rivara’s longest edge refinement algorithm refines a mesh by iteratively bisecting the longest edges of tetrahedral elements. The algorithm ensures that the resulting mesh remains conforming, meaning all the elements share complete edges with their neighbors, and there are no hanging nodes (defined as the nodes located on broken edges, having one large unbroken neighbor, and two small neighbors). The presence of the hanging nodes is unwanted due to the complexity of processing. The longest edge refinement algorithm prevents the presence of the hanging nodes. The longest edge bisection strategy employed by the algorithm helps prevent the formation of excessively distorted or elongated elements, which could degrade numerical accuracy in simulations.

The topography of the Longyearbyen area has been built using the Global Multi-Resolution Topography (GMRT) synthesis,¹ i.e., a multi-resolution compilation of edited multibeam sonar data collected by scientists and institutions worldwide [9].

Finally, to enhance the modeling of the thermal inversion phenomena, we implemented and applied the Physics Informed Neural Networks (PINN) approach [10]. The PINN simulations of the thermal inversion presented in this paper concern the two-dimensional domain, defined along the valley where the town of Longyearbyen is located (see Fig. 1).

The extraordinary success of Deep Learning (DL) algorithms in various scientific fields [11–13] over the last decade has recently led to the exploration of the possible applications of (deep) neural networks (NN) for solving partial differential equations (PDEs). The

exponential growth of interest in these techniques started with the PINN [14]. PINNs have been successfully applied to solve a wide range of problems, from fluid mechanics [15,16], in particular Navier–Stokes equations [17–19], wave propagation [20,21], phase-field modeling [22], biomechanics [23,24], quantum mechanics [25], electrical engineering [26], problems with point singularities [27], uncertainty qualification [28], dynamic systems [29,30], or inverse problems [31–33], among others. In this paper, we use the PINN approach to model the thermal inversion phenomena in the town of Longyearbyen at Spitzbergen.

Our method focuses on the modeling of the pollution propagating from the power plant. We assume uniform pollution propagation. In the future work we may apply the Takagi–Sugeno fuzzy system method [34–38] and neural networks and multi-robotic systems [39–42] to model the control of the power plant.

The rest of the paper is structured as follows: Section 2 summarizes the novelties of the paper. In Section 3, we define the three-dimensional graph-grammar model expressing Rivara’s longest edge refinement algorithm. Section 4 describes the traditional way of implementing Rivara’s longest edge algorithm by performing mesh transformations. The goal of this section is to illustrate the benefits of the graph-grammar approach. Section 5 discusses the computational model developed to simulate pollution propagation from the power plant chimney. In particular, in Section 5.1, we describe the mesh generation process using the graph-grammar code; Section 5.2 describes the strong form, whereas Section 5.3 introduces the corresponding weak form of the advection-diffusion equation. Section 5.4 presents the stabilized SUPG formulation, and finally, Section 5.5 summarizes numerical experiments and obtained results. In Section 6, we introduce the PINN concept for simulations of the thermal inversion phenomenon, where we cover, in particular, the loss function employed and the sketch of the applied training algorithm. This is followed by a presentation of the software implementation of the PINN in Section 7 and its numerical results in Sections 7.6 and 7.7. Section 8 introduces the IGA-ADS solver, using the higher-order and continuity finite element method and linear computational cost solver. The goal of this section is to compare the quality and cost of the IGA-ADS solution with the

¹ <https://www.gmrt.org/>.

PINNs' results. Section 9 discusses the influence of pollution propagation on the Longyearbyen inhabitants. Finally, we conclude the paper in Section 10.

2. Novelty of the research

Physics Informed Neural Networks are of particular interest to the soft computing community. The research concerns the design of PINNs for solving difficult physics problems, as well as optimizing the neural network architecture, increasing the accuracy of the obtained solution and applying the PINN method to solve multi-criteria optimization problems. In [43] the variable transformation method is combined with PINNs to improve the approximability and accuracy of the solutions obtained by neural networks. The problem of the optimal architecture of the neural networks for PINNs has been also investigated by using soft computing methods such as evolutionary algorithms [44]. The problem of multi-objective optimization of different loss functions in PINNs has been addressed in [45] by introducing the adaptive weighting method. The training of PINNs for the stationary two and three-dimensional convection–diffusion–reaction problems is addressed in [46]. In our work, we employ the time-dependent two-dimensional advection–diffusion equations. In this sense, the complexity of our work is similar to [46], where we have “replaced” the third dimension with the time axis. This shows the elegance of the PINN methodology, where the time variable can be treated in the same way as the space variable by the introduction of the space–time domain. The soft computing community employs the Physics Informed Neural Networks to solve different challenging problems, such as the highly non-linear problems of gas-lifted oil wells in [47], or multi-body collision problems for full-scale train collisions [48]. The PINNs are often mixed with knowledge-based computing, as presented in [49]. The problem of pollution propagation prediction can be addressed with neural networks trained on existing data, as presented in [50]. In our paper, we address this problem by employing the Physics Informed Neural Networks for the pollution propagation at Spitsbergen, with the initial condition determined from the results of the finite element method simulations with graph-grammar-based mesh generation.

2.1. List of open problems

1. *There is no simulation investigation of the pollution generated by the coal-based power plant and the diesel engine power plants in Longyearbyen at Spitzbergen with Physics Informed Neural Networks and Finite Element Method.* No one has ever surveyed the Longyearbyen inhabitants concerning their living conditions due to the pollution generated by power plants during the Arctic night. The Spitzbergen area is a subject of measurements concerning the climate change investigation [51]. The horizontal temperature profiles and their estimated changes following the climate change was analyzed in [52]. We are not aware of a previous work using computer simulation to investigate the pollution propagation from power plants in the Longyearbyen area of Spitzbergen.
2. *There is no graph grammar-based formulation of the three-dimensional longest edge refinement algorithm, where the cost of identification of single mesh element is $\mathcal{O}(1)$.* The computational mesh can be represented by a large graph. The mesh refinements can be expressed as graph transformations, called graph-grammar productions. Each graph-grammar production consists of the left-hand side and the right-hand side graphs. Application of graph grammar production to a graph consist in finding in this graph a subgraph isomorphic with the left-hand-side graph and replacing it with the right-hand-side graph. The graph-grammar productions can be applied simultaneously in different places of the large graph representing the mesh. The main computational cost of processing graph grammar production is related to the

identification of the sub-graph of the large graph representing the whole mesh. This identified sub-graph must be isomorphic with the left-hand-side graph of the production. In general, the problem of finding a sub-graph of the large graph isomorphic to the desired graph is NP-complete [53]. The longest-edge refinement algorithm has been proposed for triangular 2D meshes by Cecilia Rivara [3,4] using traditional definitions of mesh and elements. This algorithm employs the longest-edge bisection path (LEPP). The latest modification of the LEPP algorithm incorporating boundary surface constraints and node-size considerations is described in [54]. The paper [1] presents the reformulation of the LEPP algorithms by a graph-grammar model. The graph-grammar expression of the longest-edge refinement algorithm allows for the simultaneous processing of several longest-edge bisection paths, contrary to the traditional LEPP algorithms (see [1]). The graph-grammar versions of the LEPP algorithm have also been introduced in [55]. These models [1,55] use the graph representation of the mesh, where pointers identify elements by their interior nodes, the element interiors are connected with vertices, vertices are connected with edges, edges form element faces, and faces form elements. The identification of a single tetrahedral mesh element in such a configuration of the mesh requires browsing edges connected to vertices. Such a model is highly inefficient and makes the implementation computationally expensive. This is because the tetrahedron's vertices may have an arbitrary number of connected edges in the computational mesh. The identification cost of a single mesh element in [1,55] is $\mathcal{O}(E)$, where E is the maximum number of edges adjacent to vertices in the graph representation of the mesh. Summing up, there is a need to design a graph-grammar model where the cost of identification of the sub-graph of the computational mesh isomorphic with mesh element is $\mathcal{O}(1)$.

3. *There is no comparison of the two-dimensional non-stationary advection–diffusion simulations of thermal inversion with Physics Informed Neural Networks (PINNs) and higher-order Finite Element Method (FEM) solver (isogeometric analysis solver).* In our previous work [56], we compared stationary advection–diffusion simulations (the Eriksson–Johnson model problem [57]) with PINNs and Variational PINNs and higher-order FEM solvers. The comparison between PINNs and FEM for the 2D Poisson problem, 1D time-dependent Allan–Cahn, and 1D semilinear Schrödinger equations is also presented in [58]. However, the non-stationary PINN simulations of the 2D time-dependent advection–diffusion applied for the thermal inversion simulations have not been compared to FEM solvers yet.

2.2. List of main scientific contributions

1. *We perform the simulation investigations of the pollution generated by the coal-based power plant and the diesel engine power plants in the city of Longyearbyen at Spitzbergen using Physics Informed Neural Networks and Finite Element Method with graph-grammar-based mesh generation.* We compare the impact of both power plants on the living quality of Longyearbyen inhabitants. For the wind direction and intensity, as well as vertical profiles of the temperature, we refer to the High-Resolution Operational Forecasts dataset obtained from the National Science Foundation [59] and the Global Wind Atlas [60]. Based on the data, we estimate the average wind speed at the Longyearbyen location as 4 m/s (see Fig. 2) directed from the coast towards the valley where the city is located (see Fig. 3). The wind velocity reaches maximum values in February and minimum values in June (see Fig. 5). The mean velocity profile at the height of 10 m is presented in Fig. 4. The average monthly wind velocity profiles at different times of the day are presented in Fig. 5. The mean wind speed index is a base index representing the average wind

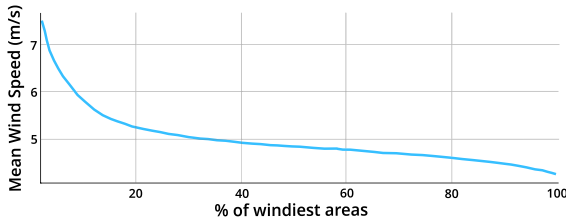


Fig. 2. A mean wind velocity profile for the region of Longyearbyen.

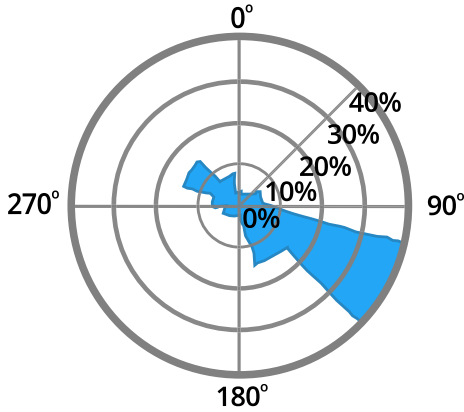


Fig. 3. A mean wind velocity direction for the region of Longyearbyen.

speed values at 10 m height. For the pollution propagation simulation we select the average wind velocity of 4 m/s, and the average wind is directed inside the valley (see Fig. 3).

2. We have proposed an efficient, longest-edge refinement graph-grammar-based algorithm for three-dimensional meshes for the first time. It allows the identification of a sub-graph representing tetrahedral element of the large graph representing the computational mesh in $\mathcal{O}(1)$ computational cost. This is because our graph grammar identifies element nodes using connections from element interiors directly to element edges. Each edge is connected to only one pair of vertices. Our graph-grammar rules successfully describe the longest-edge mesh refinements over such the mesh representation and show how to transform the graph to maintain the mesh's graph representation, allowing for low cost $\mathcal{O}(1)$ identification of tetrahedrons.
3. We present the first comparison between time-dependent two-dimensional advection-diffusion thermal inversion simulations performed with Physics Informed Neural Networks (PINNs) and higher-order and continuity finite element method solvers (isogeometric analysis (IGA)).

Summing up, there are the following novelty points in our paper.

The first theoretical point is that the graph-grammar model that we proposed is simple to implement and it has a low computational cost. Alternative traditional implementation using the computational mesh requires twelve mesh transformations, and the execution of these transformations requires expensive identifications of the left-hand sides, namely the full tetrahedron. We will show that the computational cost of the identification of the full tetrahedron can be estimated as $E_1 + E_2 + E_3 + E_4$, where E_i stands for the number of mesh edges adjacent to i th vertices of the tetrahedron, and for complex computational meshes this number can be large. On the other hand, the cost of identification of the left-hand side sub-graph in our graph grammar model is $\mathcal{O}(1)$. This is because the whole topology is connected through faces and edges, not through the vertices, and there is no need to identify the common edges.

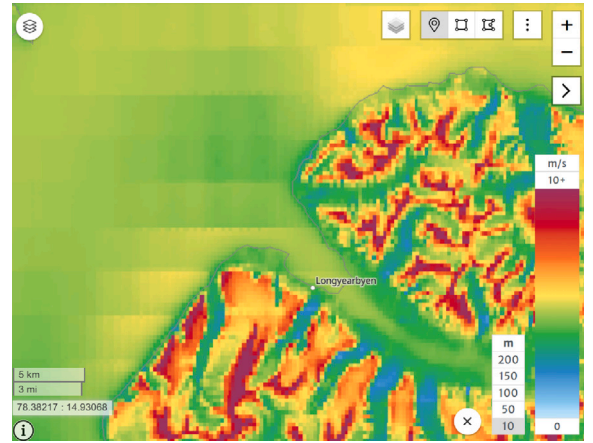


Fig. 4. A mean wind velocity profile for the region of Longyearbyen at the height of 10 m.

The second theoretical point of our paper is to compare the Physics Informed Neural Network (PINNs) implementation of the advection-diffusion model simulation of the pollution process with the higher-order and continuity finite element method implementation (isogeometric analysis (IGA)) using linear computational cost alternating direction solver [61,62] (ADS). The IGA-ADS simulations require large number of time steps when using the explicit time integration scheme to fulfill the CFD condition [63], or special stabilization method due to the advection dominating diffusion, leading to the numerical instability, even if using unconditionally stable time integration scheme. In PINNs, there are no time steps, the problem is solved in the space-time domain, the temporal axis is discrete, and there are no stabilization issues inherited from the finite element method. In this sense, our paper generalizes the results of [56] into a non-stationary advection-diffusion case. We will also compare the execution times and the profiles of the solutions.

The third novelty of the paper is the introduction of the Physics Informed Neural Network code for non-stationary thermal inversion simulations implemented in Google Collab. We present novel simulation investigation of the influence of the pollution generated by two power plants at Spitzbergen, the recently closed coal-burning power plant [64] and the novel power plant employing diesel engines. From our computer simulations, we estimate the amount of pollution and discuss its influence on the health of the Longyearbyen at Spitzbergen. We also augment our paper with data obtained from interviews performed with Longyearbyen inhabitants.

3. Graph grammar for the longest edge refinements of three-dimensional tetrahedral elements

Mesh refinement lies in subdividing an element of a mesh to obtain a finer mesh. During mesh refinement, the original nodes are not removed, and the topology of the original mesh is preserved. This is different from re-meshing the domain with smaller-sized elements. Longest-edge refinement (see Fig. 6) can be expressed mathematically as the bisection of a simplex:

$$(q = \{p_1, p_2, \dots, p_n, p_{n+1}\}) \in \mathcal{R}^{n+1}. \quad (1)$$

If the distance between p_k and p_m is the maximum distance of the simplex, then a new point is created such that $p = (p_k + p_m)/2$ and the two new simplices are created such as:

$$q_1 = \{p_1, p_2, \dots, p_{k-1}, p, p_{k+1}, \dots, p_m, \dots, p_{n+1}\} \quad (2)$$

$$q_2 = \{p_1, p_2, \dots, p_k, \dots, p_{m-1}, p, p_{m+1}, \dots, p_{n+1}\} \quad (3)$$

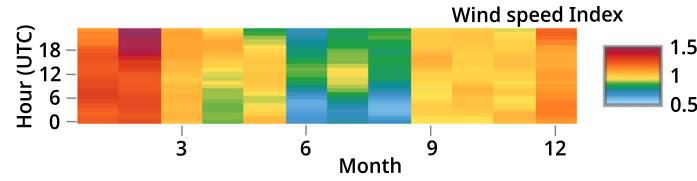


Fig. 5. An average wind speed index for particular times of a day, for particular months.

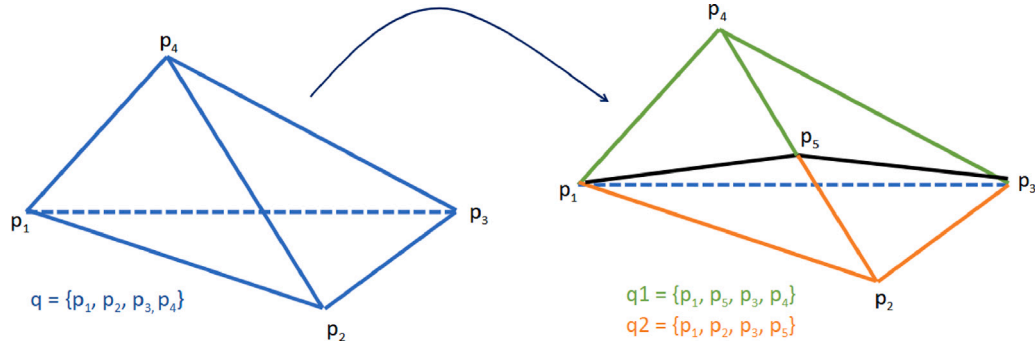


Fig. 6. The tetrahedral elements with four vertices p_1, p_2, p_3, p_4 . Its longest edge is located between vertices p_2 and p_4 . The longest edge refinement breaks the longest edge and introduces new vertex p_5 .

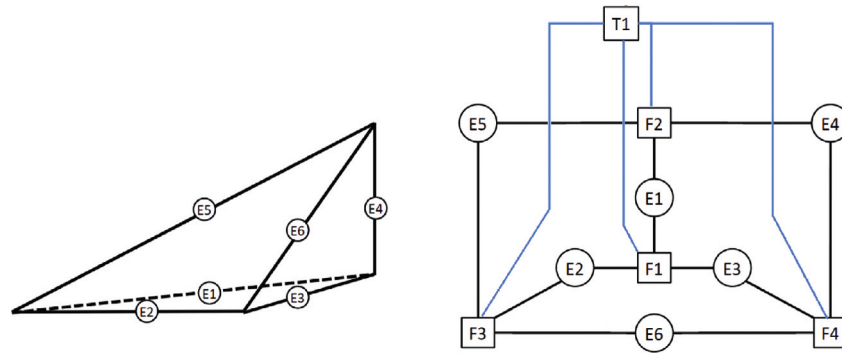


Fig. 7. Graph representation of a 3D tetrahedron denoted by T_1 . Its edges are denoted by E_1, E_2, E_3, E_4, E_5 . Its faces are denoted by F_1, F_2, F_3, F_4 .

Mathematically, the longest edge can be determined in any dimension. Geometrically, the longest edge refinement generates a new point in the middle of the longest edge, generating two new elements. The three-dimensional tetrahedral element is represented as a graph, with vertices representing interiors, edges (see Fig. 7) and faces (see Fig. 8) of the tetrahedron. We need five graph grammar productions **P0**, **P1**, **P2**, **P3**, and **P4** to express the tetrahedral mesh refinement. The graph grammar production **P0** marks tetrahedrons selected by the user for refinement. The other productions **P1**, **P2**, **P3**, and **P4** are employed to remove hanging nodes and broken faces on the rest of the graph representation of the computational mesh. Each of the productions bisects the tetrahedron into two new tetrahedral elements. The graph representation of the tetrahedral element has the following attributes:

- Attributes of vertex T representing the whole tetrahedron:
 - R : The triangle is marked to be refined
- Attributes of vertex E representing a single edge:
 - LE : The edge is one of the longest-edges
 - BR : The edge is broken
 - AE : The edge is located on the boundary (1 if is a boundary, 2 if is interior)
 - x, y, z : Coordinates of the edge (middle point)
 - IP : Pointer to the initial point

– FP : Pointer to the final point

- Attributes of vertex F representing a single face:

– BRF : The face is broken

In the following subsections, we focus on the computational tools developed for the numerical simulations of pollution propagation in Longyearbyen. In particular, we introduce a novel graph-grammar model for generating the computational mesh employed for the simulations.

3.1. Graph-grammar production P1

The first graph-grammar production denotes the case when the tetrahedral has no broken edges. The production's left-hand side is denoted in Fig. 9. The right-hand side for the graph-grammar production **P1** as well as for all the other productions **P2**, **P3**, and **P4** are presented in Fig. 10.

We have the following predicates of applicability of the graph-grammar production **P1** (i.e., conditions that must be fulfilled if the graph-grammar production can be executed):
 (NOT BR_1 AND LE_1) AND (R_1 OR ANY(BR_j)) AND NOT (BRF_1 OR BRF_2) AND NOT ANY(BR_j AND LE_j) AND NOT ANY(NOT BR_j AND LE_j AND LESS(E_1, E_j)),

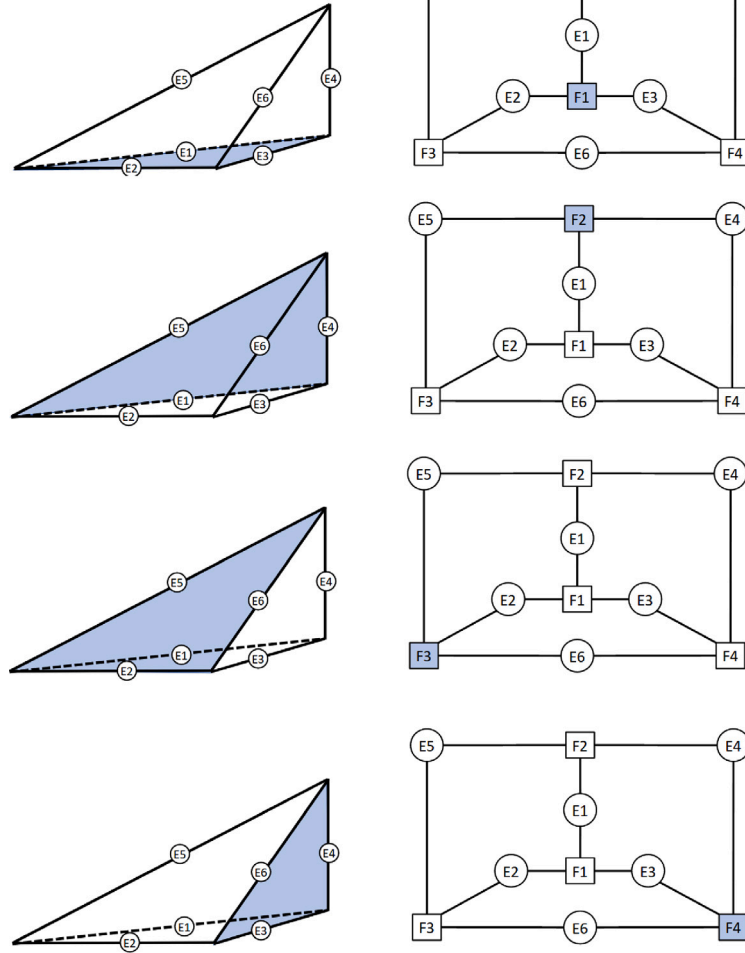


Fig. 8. Mapping the tetrahedral faces F_1, F_2, F_3, F_4 into the vertices of a graph.

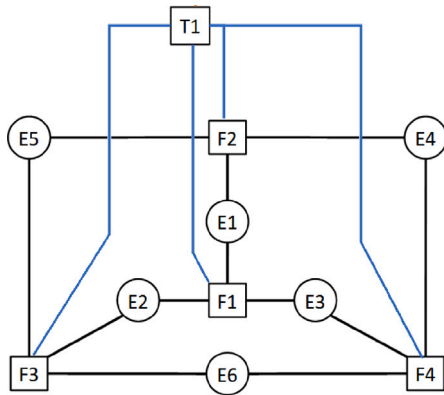


Fig. 9. Left-hand side of the graph-grammar production **P1** representing unbroken single tetrahedron.

The first component (NOT BR1 AND LE1) checks if the first edge is not broken and if it is the longest edge. The second component ($\text{R1 OR ANY(BR}_j\text{)}$) checks if the tetrahedron has been marked to be refined or already has some broken edges (is non-conforming) and, therefore, must be broken. The third component $\text{NOT (BRF1 OR BRF2)}$ checks if face1 or face2 are broken, and edge1 is not broken. In this case, the tetrahedron cannot be bisected by edge1. The fourth component $\text{NOT (BR}_j\text{ OR LE}_j\text{)}$ checks if any other edge is broken, and it is also the

longest edge. In this case, the longest edge is prioritized to be broken (edge1 will not be broken; we would rather break edge j). Finally, the fifth component $\text{NOT ANY (NOT BR}_j\text{ AND LE}_j\text{ AND LESS(E1, E}_j\text{))}$ checks if any other non-broken edge is also the longest edge.

3.2. Graph-grammar production **P2**

The second graph-grammar production denotes the case when there is one broken edge of the tetrahedral, but there are no broken faces. The left-hand side of the production is shown in Fig. 11, whereas the right-hand side for the graph-grammar production **P2** is presented in Fig. 10.

The following predicates of applicability of the graph-grammar production **P2** exist:

$(\text{LE1}) \text{ AND NOT (BRF1 OR BRF2) AND NOT ANY (BR}_j\text{ AND LE}_j\text{ AND LESS(E1, E}_j\text{))}$

The first component (LE1) checks if edge1 is the longest edge, so it should be broken. The second component $\text{NOT (BRF1 OR BRF2)}$ checks if face1 or face2 are broken. In this case, the production **P3** is the right production to apply. The third component $\text{NOT ANY (BR}_j\text{ AND LE}_j\text{ AND LESS(E1, E}_j\text{))}$ checks if any other broken edge is also denoted as the longest edge. In this case, we will break edge1 only if it is the longest one.

3.3. Graph-grammar production **P3**

The third graph-grammar production denotes the case when there is one broken edge of the tetrahedral and one adjacent broken face.

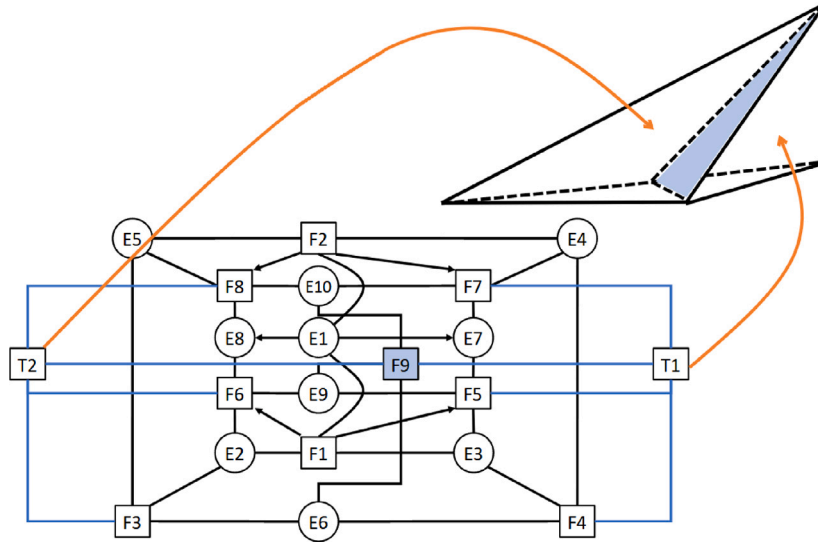


Fig. 10. The right-hand side of the graph-grammar production **P1**, **P2**, **P3**, and **P4** (same for all productions). It contains the edges of the original single tetrahedral $E_1, E_2, E_3, E_4, E_5, E_6$ and the newly created edges E_7, E_8, E_9, E_{10} . It also contains the faces of the original unbroken tetrahedral F_1, F_2, F_3, F_4 and the newly created faces F_5, F_6, F_7, F_8, F_9 . The two newly created tetrahedrons are denoted by T_2, T_3 .

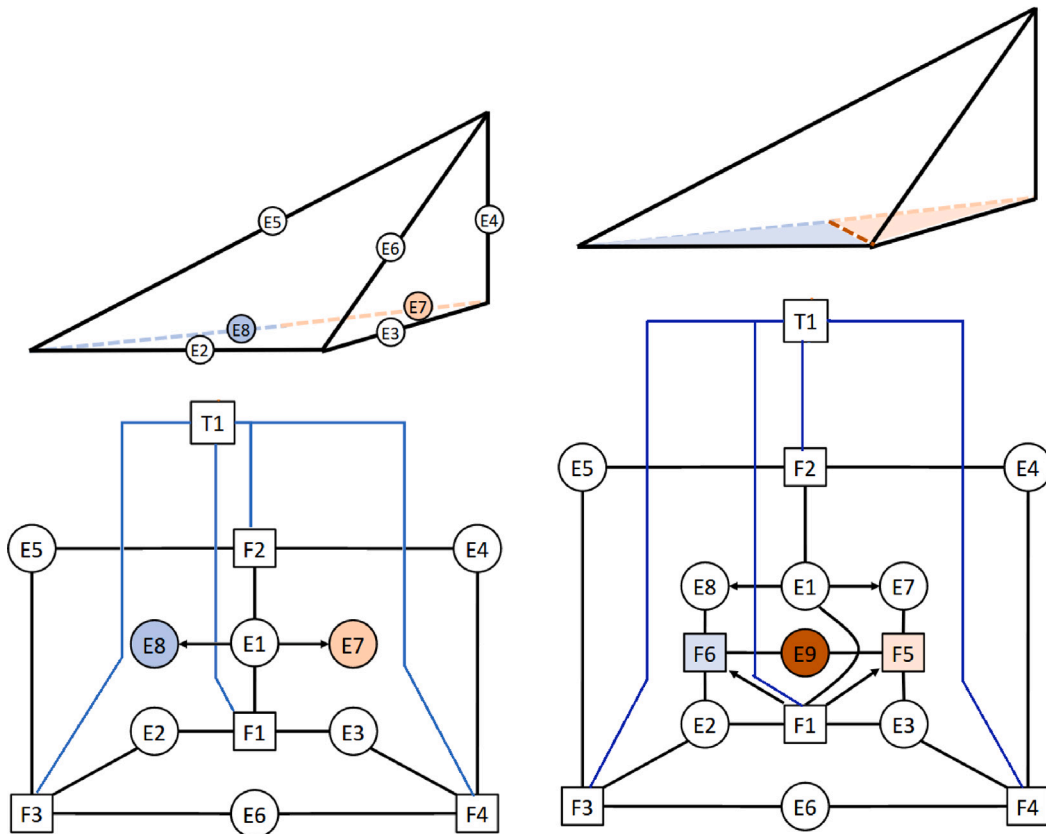


Fig. 11. Left-hand side of the graph-grammar production **P2** (left panel) and **P3** (right panel). They are applied to break tetrahedrons with broken edge (production **P2**) or broken face (production **P3**). They transform the graph into the tetrahedral broken into to tetrahedrons, presented in Fig. 10.

The production's left-hand side is shown in Fig. 11 and the right-hand side in Fig. 10. The following predicates of applicability of the graph-grammar production **P3** exist:

(LE1) AND (BRF1 AND NOT BRF2) AND NOT ANY(BR_j AND LE_j AND LESS(E1, E_j))

The first component of the predicate of applicability (LE1) checks if edge1 is the longest edge, so it should be broken. The second component (BRF1 AND NOT BRF2) checks if face1 is not broken and face2 is broken. In this case, the right production to apply is the production **P2**. The third component NOT ANY(BR_j AND LE_j AND

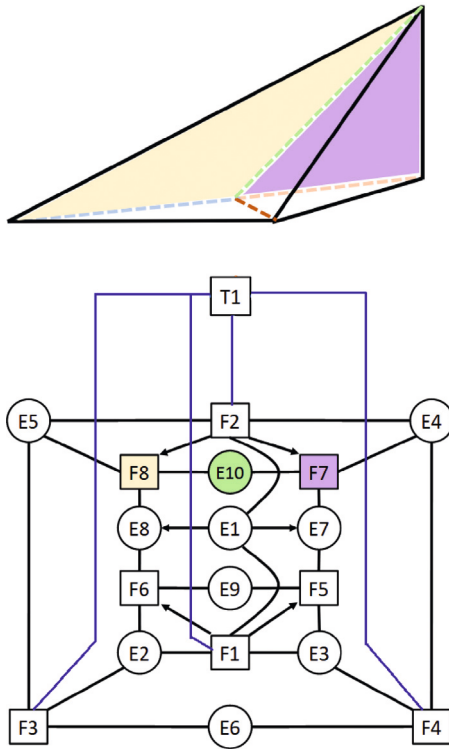


Fig. 12. Left-hand side of the graph-grammar production **P4**. It is applied to break the tetrahedron with two broken faces. They transform the graph into the tetrahedron broken into to tetrahedrons, presented in Fig. 10.

$\text{LESS}(E1, E_j))$ checks if any other broken edge is also denoted as the longest edge. In this case, we will break edge1 only if it is the longest edge.

3.4. Graph-grammar production **P4**

The fourth graph-grammar production denotes the case when there is one broken edge of the tetrahedron and two adjacent broken faces. The left-hand side of the graph-grammar production is shown in Fig. 12 and the right-hand side in Fig. 10.

There are the following predicates of applicability of the graph-grammar production **P4**:

$(\text{LE1}) \text{ AND NOT ANY}(\text{BR}_j \text{ AND } \text{LE}_j \text{ AND } \text{LESS}(E1, E_j))$

The first component of the predicate of applicability (**LE1**) checks if edge1 is the longest edge, and, therefore, it should be broken. The second component $\text{NOT ANY}(\text{BR}_j \text{ AND } \text{LE}_j \text{ AND } \text{LESS}(E1, E_j))$ checks if any other broken edge is also denoted as the longest edge. In this case, we must use the comparison operator to ensure that edge1 is the longest.

3.5. Control diagram for graph grammar

The diagram of controlling the execution of the graph grammar production is presented in Fig. 14. It starts with the execution of a sequence of production **P0** that the user controls and it marks tetrahedron for refinements. Next, the productions **P1**, **P2**, **P3**, **P4** are executed to remove the hanging nodes.

4. Transformations over computational mesh

In this section, we show how to express the three-dimensional longest-edge refinement algorithm by mesh transformation over the traditional representation of the computational mesh. This kind of

graph transformation is employed in traditional mesh generators using the Rivara algorithm. We will use this example to compare the computational cost of our graph grammar model with the traditional approach. In this approach, we have the computational mesh with tetrahedrons. Each tetrahedron is identified by its interior (denoted by I), and there are links from the interior to the vertices, and the vertices are connected by edges (denoted by E), see Table 1.

4.1. Mesh transformation rules

In order to implement Rivara's longest edge refinement algorithm, we introduce mesh transformation (**MT1**), presented in Table 1. It breaks a tetrahedral element along the longest edge. It breaks the edge into two new edges, breaks the face into two new faces. It introduces a new internal face, and it breaks an interior into two new interiors. The mesh transformations are isomorphic; that is why we assume we can apply the same transformation for tetrahedron oriented in different ways. All the other mesh transformations break tetrahedral elements along the longest edge to remove the hanging edges (to eliminate the situation when a tetrahedral element has a broken face). All the tetrahedrons are broken along the longest edge to ensure the proper proportions of the created computational mesh. We have 12 mesh transformations, listed in Tables 1–3. The transformations (**MT2**) and (**MT3**) consider the longest edge of a tetrahedral element with one face already broken and break the tetrahedron along the longest edge. The transformations (**MT4**), (**MT5**), (**MT6**), (**MT7**) and (**MT8**) consider the longest edge of a tetrahedron with two faces already broken, and break the tetrahedron along the longest edge. The transformations (**MT9**), (**MT10**), (**MT11**), and (**MT12**) consider the longest edge of a tetrahedron with three faces already broken, and break the tetrahedron along the longest edge.

4.2. Comparison of the longest-edge refinements with graph-grammar productions and mesh transformations

The execution of Rivara's longest-edge refinement algorithm requires identification of the tetrahedron to be broken as requested by the user or by the algorithm removing the hanging nodes. Each tetrahedron is identified by its interior, which is connected to the four vertices. There are also pointers from the vertices to the edges of the tetrahedron. Identification of these edges, connecting the vertices of the tetrahedron, is expensive since there can be an arbitrarily large number of edges connected to a vertex. The computational cost of identification of the common edges can be estimated as $\mathcal{O}(E_1 + E_2 + E_3 + E_4)$ where E_1 stands for the number of edges connected to the first vertex, E_2 denotes the number of edges connected to the second vertex, one of them that we look for, connecting the second vertex with the first vertex. Similarly, the number of edges assigned to the third and fourth vertex is E_3 and E_4 .

This computational cost of the identification of the tetrahedral element can be reduced down to $\mathcal{O}(1)$ by employing the graph representation of the mesh, where there the tetrahedral interior is connected directly to the faces, the faces are connected to the edges. We simply do not travel through the vertices. The vertices are represented as the edges in this graph. Thus, the process of identification of the tetrahedron is straightforward since we travel from interior to faces in $\mathcal{O}(1)$, from faces to edges in $\mathcal{O}(1)$, and then we have only one graph edge (representing mesh vertex) between each pair of edges. We do not need to identify the common edges. The cost of identification of the tetrahedron is thus $\mathcal{O}(1)$.

5. Simulation of the pollution propagation from the power plants in Longyearbyen using the finite element method

This section describes our finite element method simulations of the advection-diffusion model of pollution propagation from the power plant chimney.

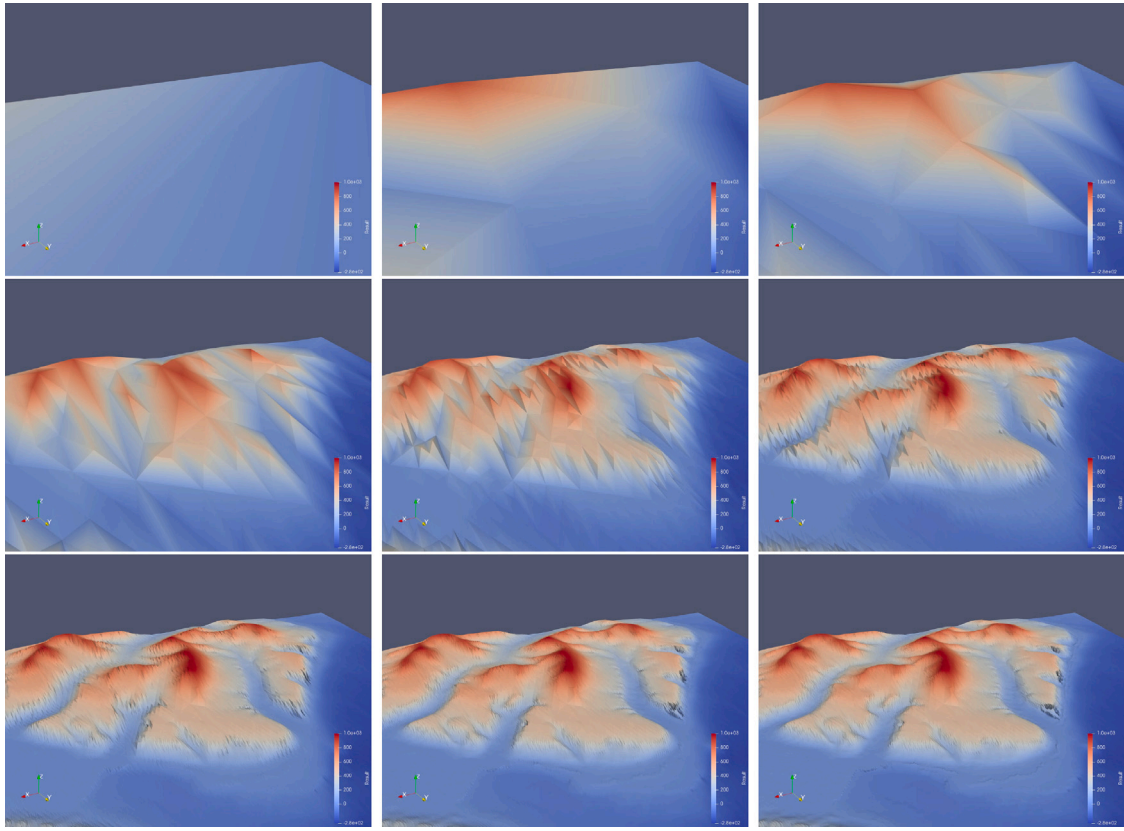


Fig. 13. A sequence of mesh refinements performed by the longest-edge refinement algorithm to generate the topography of Spitzbergen.

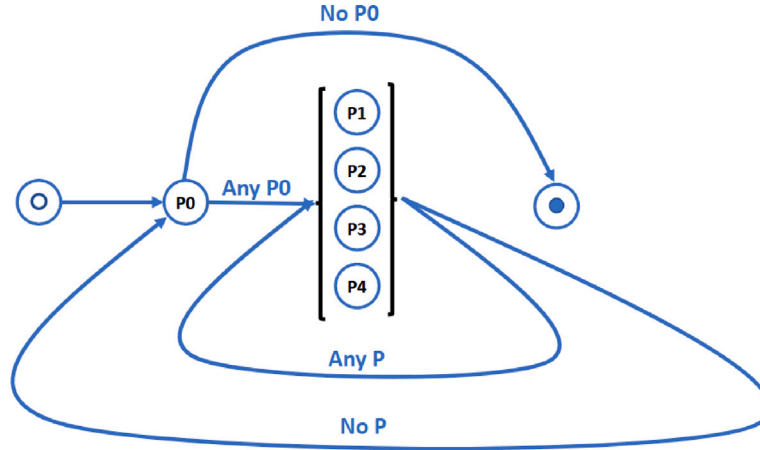


Fig. 14. Diagram controlling the execution of graph-grammar productions **P1**–**P4**. First, all required production **P0** are executed. It is followed by the simultaneous executions of productions **P1**, **P2**, **P3**, **P4** as many times as possible. When it is no longer possible, we do not have hanging nodes anymore, we go back to the execution of production **P0**. If this is no longer required, we stop the refinements process.

5.1. Mesh generation for Spitzbergen topography

We employ the graph grammar described in Section 3 for the generation of the computational mesh with triangular elements covering the topography of the Longyearbyen area, based on the GMRT data [9]. An exemplary sequence of generated meshes is presented in Fig. 13. In this Figure, we plot the cross-section of the tetrahedral mesh with the approximation of the terrain's topography. We list the number of generated nodes in Table 4. The total generation time was equal to 82 s. The automatically refined mesh from the graph-grammar algorithm has been manually modified to add a chimney representing the power plant. For an overview of this mesh, see Figs. 15, 16 and 17.

5.2. Strong form of the advection-diffusion-reaction equations

We use the advection-diffusion equation to model the transport of pollutants:

$$\frac{\partial u}{\partial t} + \beta \cdot \nabla u - \nabla \cdot (\epsilon \nabla u) = f, \quad (4)$$

where $u(x, y, z, t)$ is the pollutant concentration field; $\beta(x, y, z, t) = (\beta_x(x, y, z, t), \beta_y(x, y, z, t), \beta_z(x, y, z, t))$ the wind velocity vector field, and ϵ the diffusion coefficient. We discretize (4) in time by introducing time steps $0 = t_0 < t_1 < t_2 < \dots < t_N = T$ and the Crank-Nicholson finite

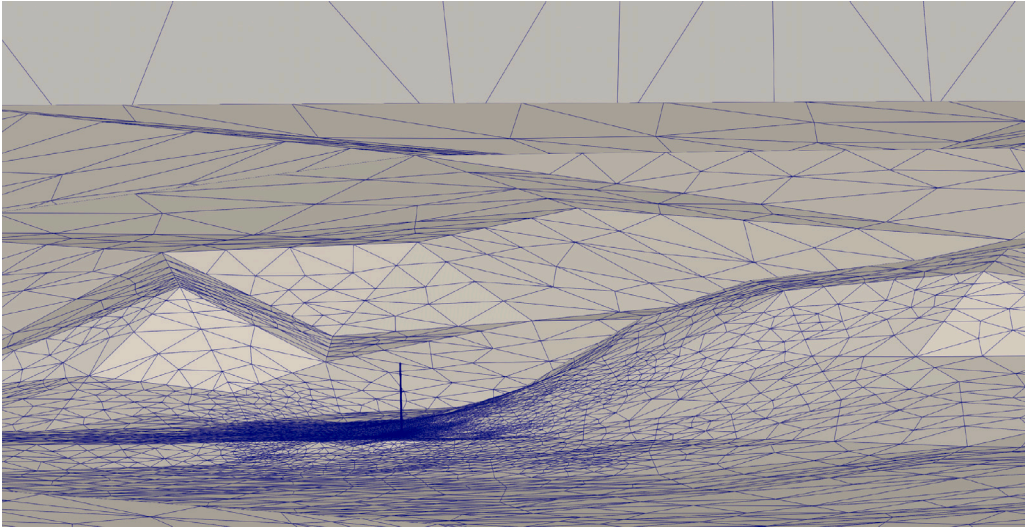


Fig. 15. A computational mesh covering the terrain with the chimney.

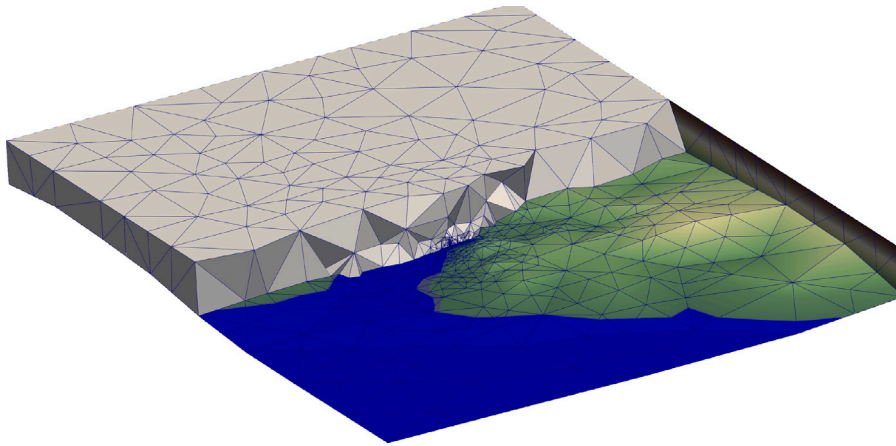


Fig. 16. A cross-section of the 3D computational mesh at the chimney's location.

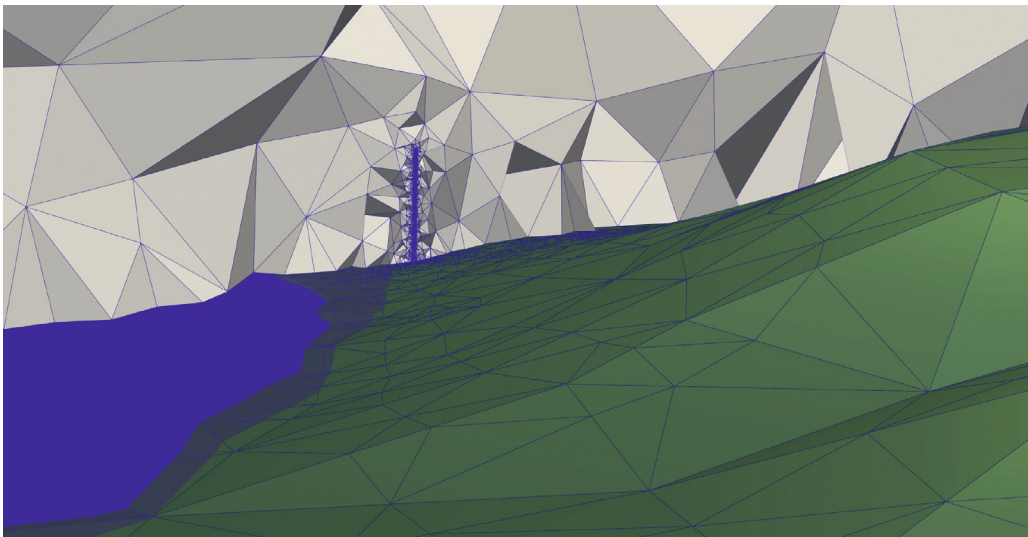


Fig. 17. A cross-section of the 3D computational mesh at the chimney's location.

Table 1
Transformations over traditional computational mesh.

Name	Mesh transformation
(MT1)	
(MT2)	
(MT3)	
(MT4)	

difference scheme in time:

$$\frac{u^{t+1} - u^t}{\Delta t} + \beta \cdot \nabla \frac{u^{t+1} + u^t}{2} - \nabla \cdot \left(\epsilon \nabla \frac{u^{t+1} + u^t}{2} \right) + c \frac{u^{t+1} + u^t}{2} = f^t \quad (5)$$

5.3. Weak form of the advection-diffusion-reaction equations

To apply the finite element method, we introduce the weak formulation of (4) to find $u \in V = H^1(\Omega)$ such that:

$$\frac{u^{t+1} - u^t}{\Delta t} + \frac{b(u^t, v) + b(u^{t+1}, v)}{2} = l(v) \quad \forall v \in V \quad (6)$$

where:

$$b(u, v) = (\beta \cdot \nabla u, v)_{\Omega} - (\epsilon \nabla u, \nabla v)_{\Omega} + (\epsilon n \cdot \nabla u, v)_{\Gamma} + (cu, v)_{\Omega} \quad (7)$$

Table 2
Transformations over traditional computational mesh.

Name	Mesh transformation
(MT5)	
(MT6)	
(MT7)	
(MT8)	

$$l(v) = (f, v)_{\Omega} \quad (8)$$

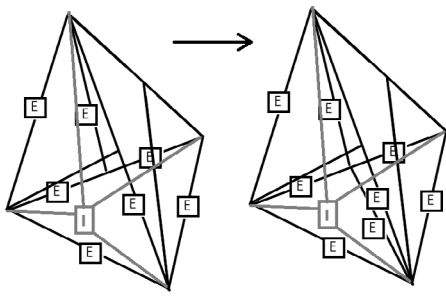
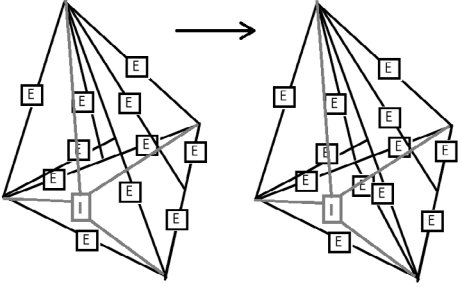
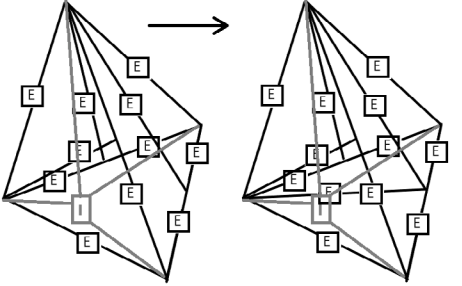
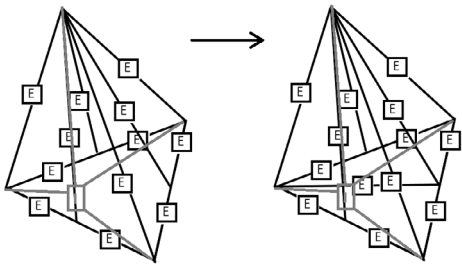
where we utilize inner product notation: $(u, v)_{\Omega} = \int_{\Omega} u v dx dy dz$, and $(u, v)_{\Gamma} = \int_{\Gamma} u v ds$ denotes the L^2 scalar product on Ω , $\Gamma = \partial\Omega$, and $n = (n_x, n_y, n_z)$ is the vector normal to Γ .

5.4. Streamline-Upwind Petrov–Galerkin method

For advection-diffusion equations, the standard Bubnov–Galerkin finite element method is known to be numerically unstable for coarse

Table 3

Transformations over traditional computational mesh.

Name	Mesh transformation
(MT9)	
(MT10)	
(MT11)	
(MT12)	

meshes. To make it numerically stable, we apply the Streamline-Upwind Petrov–Galerkin (SUPG) method [2]. Starting with the Bubnov–Galerkin discretization, we seek for $u_h \in V_h \subset V$ such that:

$$\left(\frac{u_h^{t+1} - u_h^t}{\Delta t}, v_h \right) + \frac{b(u_h^t, v_h) + b(u_h^{t+1}, v_h)}{2} = l(v_h) \forall v_h \in V_h \subset V, \quad (9)$$

where V_h is span by polynomial functions introduced by the tetrahedral finite elements. The SUPG method modifies then the weak form to stabilize the formulation:

$$b(u_h^{t+1}, v_h) + \sum_K (R(u_h^{t+1}), \tau \beta \cdot \nabla v_h)_K = l(v_h) + \sum_K (f, \tau \beta \cdot \nabla v_h)_K \quad \forall v \in V, \quad (10)$$

Table 4

The number of mesh nodes on a generated sequence of triangular element meshes approximating the topography of the Svalbard area, using the 3D longest-edge refinement graph grammar.

Iteration #1	Number of nodes	Iteration #2	Number of nodes
1	4	11	8723
2	9	12	16352
3	17	13	29135
4	37	14	49619
5	89	15	83745
6	200	16	144882
7	445	17	258620
8	984	18	440682
9	2093	19	749160
10	4355	20	1572864

where $R(u_h^{t+1}) = \beta \cdot \nabla u_h^{t+1} + \epsilon \Delta u_h^{t+1}$, and $\tau^{-1} = \beta \cdot \left(\frac{1}{h_K^x}, \frac{1}{h_K^y}, \frac{1}{h_K^z} \right) + 3p^2 \epsilon \frac{1}{h_K^{x^2} + h_K^{y^2} + h_K^{z^2}}$, and h_K^x, h_K^y, h_K^z denote three dimensions of an element K . Thus, we have:

$$b_{SUPG}(u_h^{t+1}, v_h) = l_{SUPG}(v_h) \quad \forall v_h \in V_h, \quad (11)$$

$$\begin{aligned} b_{SUPG}(u_h^{t+1}, v_h) = & \beta_x \left(\frac{\partial u_h^{t+1}}{\partial x}, v_h \right)_\Omega + \beta_y \left(\frac{\partial u_h^{t+1}}{\partial y}, v_h \right)_\Omega \\ & + \beta_z \left(\frac{\partial u_h^{t+1}}{\partial z}, v_h \right)_\Omega + \\ & \epsilon \left(\frac{\partial u_h^{t+1}}{\partial x}, \frac{\partial v_h}{\partial x} \right)_\Omega + \epsilon \left(\frac{\partial u_h^{t+1}}{\partial y}, \frac{\partial v_h}{\partial y} \right)_\Omega \\ & + \epsilon \left(\frac{\partial u_h^{t+1}}{\partial z}, \frac{\partial v_h}{\partial z} \right)_\Omega + \\ & (cu_h, v_h)_\Omega - \left(\epsilon \frac{\partial u_h^{t+1}}{\partial x} n_x, v_h \right)_\Gamma - \\ & \left(\epsilon \frac{\partial u_h^{t+1}}{\partial y} n_y, v_h \right)_\Gamma - \left(\epsilon \frac{\partial u_h^{t+1}}{\partial z} n_z, v_h \right)_\Gamma + \\ & \left(\beta_x \frac{\partial u_h^{t+1}}{\partial x} + \beta_y \frac{\partial u_h^{t+1}}{\partial y} + \beta_z \frac{\partial u_h^{t+1}}{\partial z} + \epsilon \Delta u_h^{t+1}, \right. \\ & \left. \left(\frac{1}{h_x} + 3\epsilon \frac{p^2}{h_K^{x^2} + h_K^{y^2}} \right)^{-1} \beta_x \frac{\partial v_h}{\partial x} + \beta_y \frac{\partial v_h}{\partial y} + \beta_z \frac{\partial v_h}{\partial z} \right)_\Omega \\ l_{SUPG}(v_h) = & (f, v_h)_\Omega + \left(f, \left(\frac{1}{h_x} + 3\epsilon \frac{p^2}{h_K^{x^2} + h_K^{y^2}} \right)^{-1} \right. \\ & \left. \times \left(\beta_x \frac{\partial v_h}{\partial x} + \beta_y \frac{\partial v_h}{\partial y} + \beta_z \frac{\partial v_h}{\partial z} \right) \right)_\Omega. \end{aligned}$$

We incorporate the implicit Crank–Nicholson method into the finite element setup:

$$\left(\frac{u^{t+1} - u^t}{\Delta t}, w_h \right)_\Omega + b_{SUPG} \left(\frac{u_h^t + u_h^{t+1}}{2}, v_h \right) = l_{SUPG}(v_h) \quad \forall v_h \in V_h, \quad (12)$$

$$\begin{aligned} & (u^{t+1}, w_h)_\Omega + \frac{\Delta t}{2} b_{SUPG}(u_h^{t+1}, v_h) \\ & = (u^t, w_h)_\Omega + \frac{\Delta t}{2} b_{SUPG}(u_h^t, v_h) + l_{SUPG}(v_h) \\ & \quad \forall v_h \in V_h. \end{aligned}$$

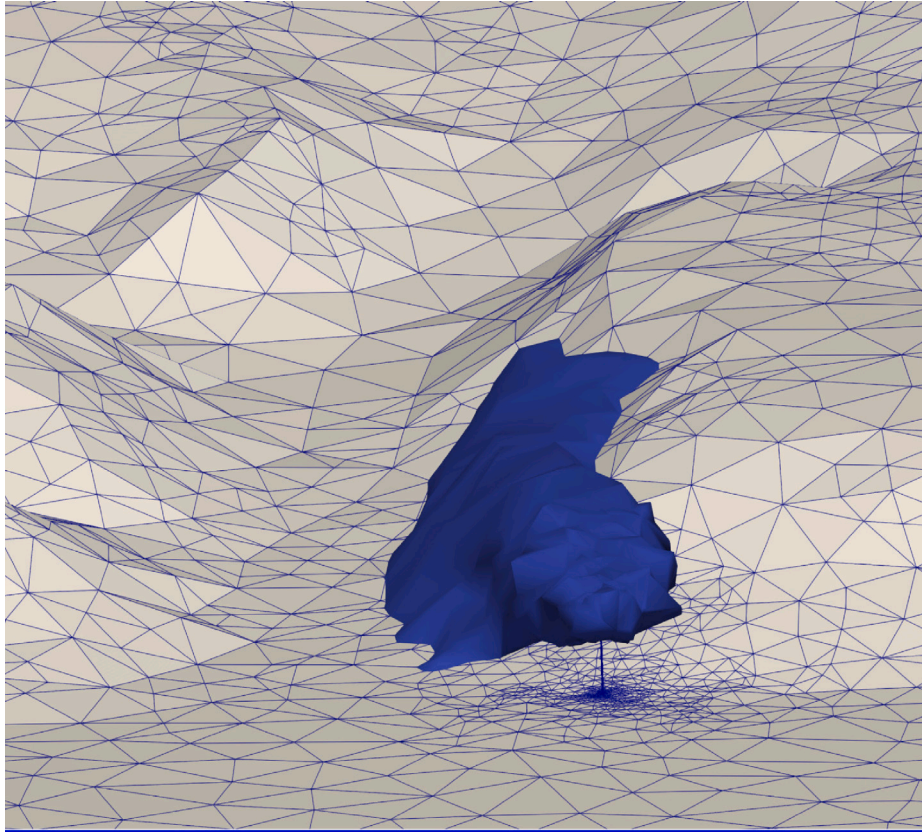


Fig. 18. The front view of the smoke propagated from the chimney into the valley after two hours of power plant operation.

The element matrices and right-hand-side vectors are discretized to obtain the local systems over each element, with matrices and right-hand-sides:

$$\begin{bmatrix} (\psi_1, \psi_1) & \cdots & (\psi_1, \psi_{15}) \\ \vdots & \ddots & \vdots \\ (\psi_{15}, \psi_1) & \cdots & (\psi_{15}, \psi_{15}) \end{bmatrix} \begin{bmatrix} u_1^t \\ \vdots \\ u_{15}^t \end{bmatrix} + \frac{\Delta t}{2} * \begin{bmatrix} b_{SUG}^K(\psi_1, \psi_1) & \cdots & b_{SUG}^K(\psi_1, \psi_{15}) \\ \vdots & \ddots & \vdots \\ b_{SUG}^K(\psi_{15}, \psi_1) & \cdots & b_{SUG}^K(\psi_{15}, \psi_{15}) \end{bmatrix} \begin{bmatrix} u_1^t \\ \vdots \\ u_{15}^t \end{bmatrix} = \begin{bmatrix} l_{SUG}^K(\psi_1) \\ \vdots \\ l_{SUG}^K(\psi_{15}) \end{bmatrix} \quad (13)$$

The resulting local systems are submitted to the matrix-free GMRES iterative solver.

5.5. Numerical results

We simulated the pollution propagation with the source located on the top of the chimney, assuming the average wind direction and velocity for the winter season. As illustrated in Figs. 18–20, the pollution propagates into the valley where Longyearbyen is located. Nine hours after the chimney starts producing the pollution, the whole valley is filled with pollution.

6. Simulation of thermal inversion using Physics Informed Neural Networks

Having the pollution propagated into the Longyearbyen valley, we can now investigate the thermal inversion phenomena during the summer and winter periods to check if the accumulated pollution dissipates or stays in the valley. This section discusses modeling the thermal inversion effect with the Physics Informed Neural Networks [14,15]. The PINN codes used for thermal inversion simulations are available at <https://colab.research.google.com/drive/>

15WDZZV36v2qmvU_vq0Ter0RvKxwrYs9 and https://colab.research.google.com/drive/1Ta29ihEOX6rWhDozK_7u89Ev0A3dX8sz

As the initial state for the simulation, we consider the pollution propagated into the valley by the chimney as directed by the wind. This pollution concentration is based on the finite element method solver. The vertical temperature profile effect is obtained by introducing the advection field as the temperature gradient.

Thermal inversion, also known as temperature inversion, is a meteorological phenomenon where the typical (decreasing with height) temperature gradient of the atmosphere is reversed (increasing with height). Typically, the temperature decreases with altitude, meaning the air is cooler higher up. However, during a thermal inversion, a layer of cooler air becomes trapped near the ground by warmer air above it. The trapped cold air also traps pollutants near the ground, leading to poor air quality. Thermal inversions are more common in valleys, where the topography limits air circulation. Inversions are also more likely to occur during the winter, especially during clear nights when the ground cools rapidly.

Thermal inversions are a common and significant phenomenon during the Arctic night due to the extreme and prolonged cold conditions that characterize this region. The Arctic night refers to the period during the winter months when the Sun does not rise above the horizon for an extended period, resulting in continuous darkness. During the thermal inversion phenomenon, the temperature increases with altitude instead of decreasing. Inversions are typical in winter when the low layers of the atmosphere are cooled by a cold surface covered with snow and ice while the higher layers remain warmer.

Thermal inversions during the Arctic night are a natural consequence of the region's extreme and prolonged cold conditions. They result in very stable and cold air near the surface, with warmer air above, and can persist for long periods.

We model the thermal inversion by introducing the vertical temperature profiles specific to winter and summer seasons in the region of the Town of Longyearbyen at Spitzbergen.

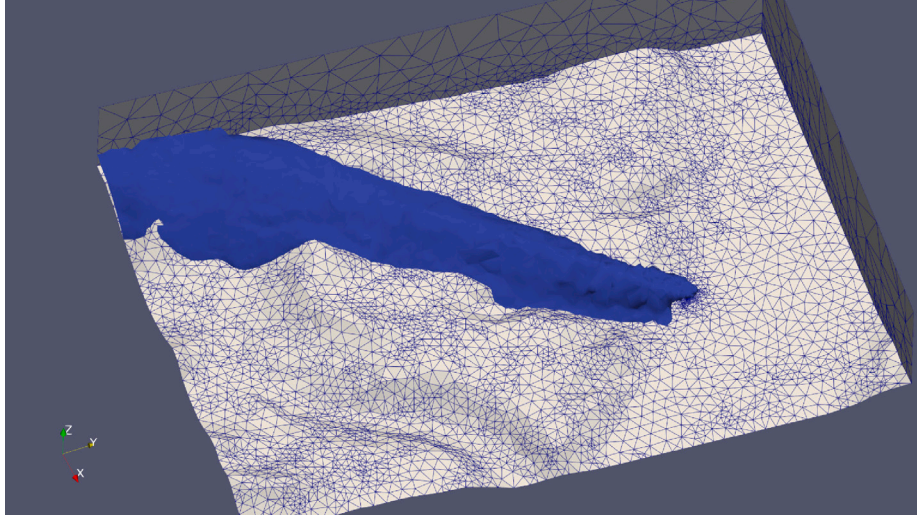


Fig. 19. The top view of the smoke propagated from the chimney into the valley after 9 h of power plant operation.

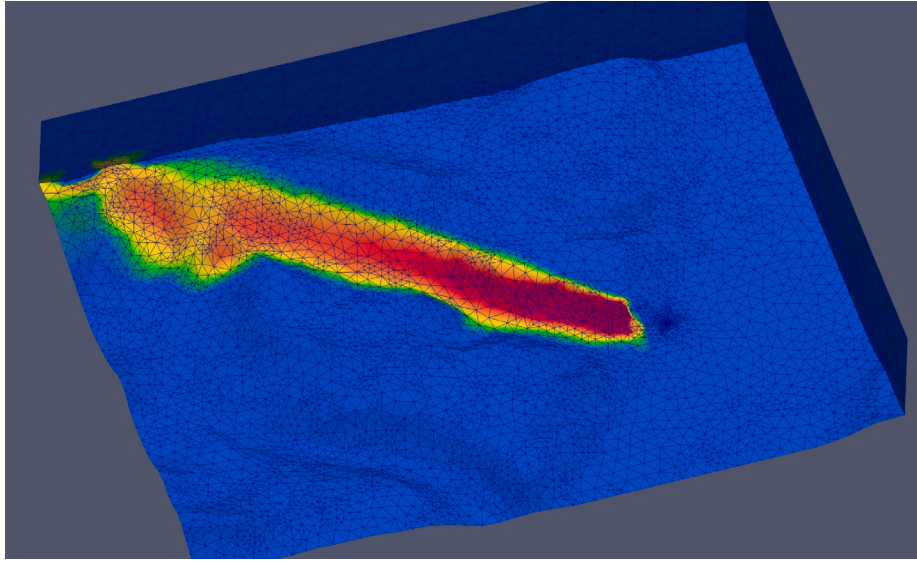


Fig. 20. The concentration of the pollution near the ground after 9 h of working of the power plant.

We also assume that the horizontal diffusion coefficient $K_x = 0.1$ is stronger than the vertical diffusion coefficient $K_y = 0.01$. We focus on advection-diffusion equations in the strong form. We seek the pollution concentration field $[0, 1]^2 \times [0, 1] \ni (x, y, t) \rightarrow u(x, y, t) \in \mathcal{R}$

$$\frac{\partial u(x, y, t)}{\partial t} + (b(x, y, t) \cdot \nabla) u(x, y, t) - \nabla \cdot (K \nabla u(x, y, t)) = 0, \quad (14)$$

$$(x, y, t) \in \Omega \times (0, T]$$

$$\frac{\partial u(x, y, t)}{\partial n} = 0, \quad (x, y, t) \in \partial\Omega \times (0, T] \quad (15)$$

$$u(x, y, 0) = u_0(x, y), \quad (x, y, t) \in \Omega \times 0 \quad (16)$$

This PDE translates into

$$\frac{\partial u(x, y, t)}{\partial t} + \frac{\partial T(y)}{\partial y} \frac{\partial u(x, y, t)}{\partial y} - 0.1 \frac{\partial u(x, y, t)}{\partial x^2} - 0.01 \frac{\partial u(x, y, t)}{\partial y^2} = 0, \quad (17)$$

$$(x, y, t) \in \Omega \times (0, T]$$

$$\frac{\partial u(x, y, t)}{\partial n} = 0, \quad (x, y, t) \in \partial\Omega \times (0, T] \quad (18)$$

$$u(x, y, 0) = u_0(x, y), \quad (x, y, t) \in \Omega \times 0 \quad (19)$$

Neural networks are composed of interconnected layers of nodes, or neurons, designed to process and learn from data. The architecture of

a typical neural network is shown in Fig. 21. The input to the neural network are the spatial point coordinates (x, y) and the time moment t . The output from the neural network is the concentration velocity field $u(x, y, t)$ that is later automatically differentiated in space and time and employed in the loss functions modeling the initial conditions, the boundary conditions, and the residual loss. In PINN, the neural network represents the solution,

$$u(x, y, t) = PINN(x, y, t) = A_n \sigma(A_{n-1} \sigma(\dots \sigma(A_1[x, y, t] + B_1) \dots + B_{n-1}) + B_n) \quad (20)$$

where A_i are matrices representing neural network layers, B_i represent bias vectors, and σ is the sigmoid activation function. We define the loss function as the residual of the PDE:

$$L_{residual}(x, y, t) = \left(\frac{\partial PINN(x, y, t)}{\partial t} + \frac{\partial T(y)}{\partial y} \frac{\partial PINN(x, y, t)}{\partial y} - 0.1 \frac{\partial PINN(x, y, t)}{\partial x^2} - 0.01 \frac{\partial PINN(x, y, t)}{\partial y^2} - f \right)^2 \quad (21)$$

$$L_{residual}(x, y, t) = \left(\frac{\partial u}{\partial t} + \frac{\partial T}{\partial y} \frac{\partial u}{\partial y} - K_x \frac{\partial u}{\partial x^2} - K_y \frac{\partial u}{\partial y^2} - f \right)^2 \quad (22)$$

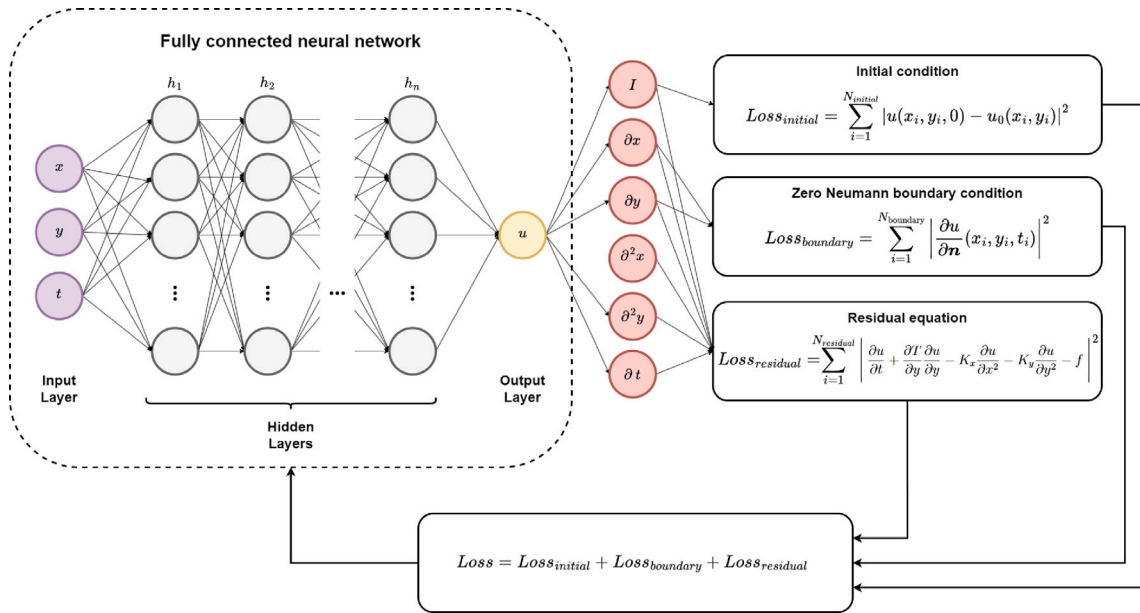


Fig. 21. The structure of the Physics Informed Neural Network for modeling of time-dependent advection-diffusion equations.

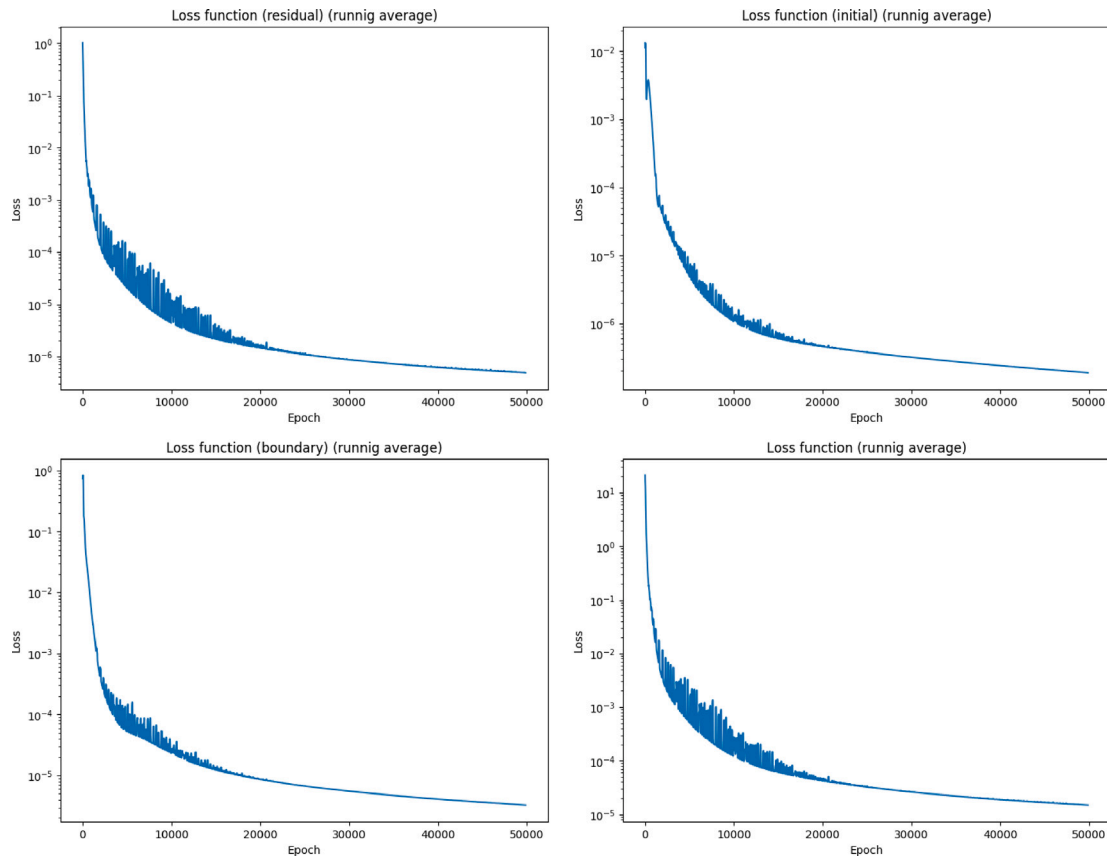


Fig. 22. Svalbard summer. The temperature decreases in vertical direction close to the ground. The convergence of residual, initial, boundary, and total loss functions.

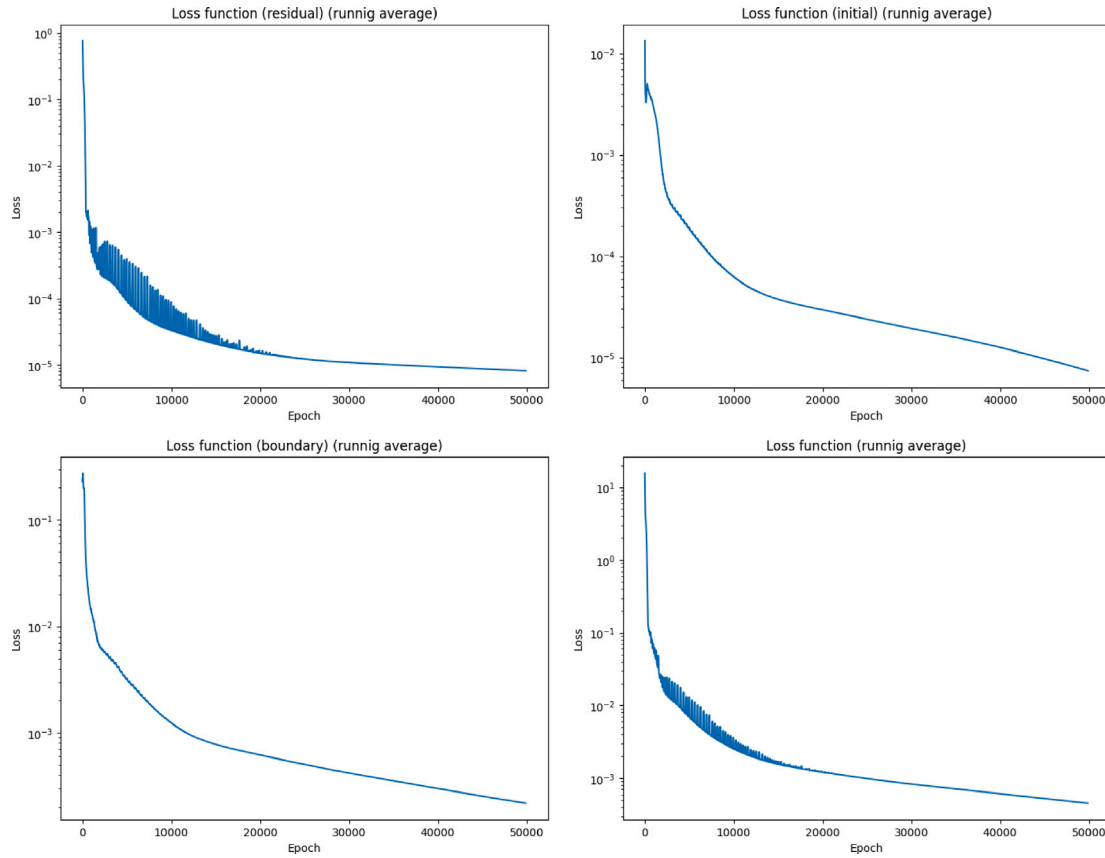


Fig. 23. Svalbard winter. The temperature increases in vertical direction close to the ground. The convergence of residual, initial, boundary, and total loss functions.

We also define the loss for training the initial condition as the residual of the initial condition:

$$L_{Initial}(x, y, 0) = (PINN(x, y, 0) - u_0(x, y))^2 \quad (23)$$

as well as the loss of the residual of the boundary condition:

$$L_{boundary}(x, y, t) = f \left(\frac{\partial PINN(x, y, t)}{\partial n} - 0 \right)^2 \quad (24)$$

We employ the Adam optimizer (Adaptive Moment Estimation) [65] for training. The general idea of the Adam algorithm is to average the gradients from several past iterations, converging towards global minima and avoiding local minima.

7. The structure of the code

7.1. Colab implementation

The simulation code may be downloaded from <https://github.com/pmaczuga/pinn-notebooks> and executed in Google Colab in fully automatic mode.

7.2. Parameters

There are the following model parameters that the user can define:

- **LENGTH, TOTAL_TIME.** The code works in the space-time domain, where the training is performed by selecting point along x , y and t axes. The **LENGTH** parameter defines the dimension of the domain along x and y axes. The domain dimension is $[0, \text{LENGTH}] \times [0, \text{LENGTH}] \times [0, \text{TOTAL_TIME}]$. The **TOTAL_TIME** parameter defines the length of the space-time domain along the t axis. It is the total time of the transient phenomena we want to simulate.

- **N_POINTS.** This parameter defines the number of points used for training. By default, the points are selected randomly along x , y , and t axes. It is easily possible to extend the code to support different numbers of points or different distributions of points along different axes of the coordinate system.
- **N_POINTS_PLOT.** This parameter defines the number of points used to probe the solution and plot the output plots after the training.
- **WEIGHT_RESIDUAL, WEIGHT_INITIAL, WEIGHT_BOUNDARY.** These parameters define the weights for the training of residual, initial condition, and boundary condition loss functions.
- **LAYERS, NEURONS_PER_LAYER.** These parameters define the neural network by providing the number of layers and number of neurons per neural network layer.
- **EPOCHS, and LEARNING_RATE** provide a number of epochs and the training rate for the training procedure.

During the training, we used the following global parameter values:

```
LENGTH = 1.
TOTAL_TIME = 1.
N_POINTS = 15
N_POINTS_PLOT = 150
WEIGHT_RESIDUAL = 20.0
WEIGHT_INITIAL = 1.0
WEIGHT_BOUNDARY = 10.0
LAYERS = 2
NEURONS_PER_LAYER = 600
EPOCHS = 30_000
LEARNING_RATE = 0.002
```


7.3. PINN class

The PINN class defines the functionality for a simple neural network accepting three features as input: the values of (x, y, t) and returning a single output, namely the value of the solution $u(x, y, t)$. We provide the following features:

- The `f` routine compute the values of the approximate solution at point (x, y, t) .
- The routines `dfdt`, `dfdx`, `dfdy` compute the derivatives of the approximate solution at point (x, y, t) with respect to either x , y , or t using the PyTorch autograd method.

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

7.4. Processing initial and boundary conditions

Since the training is performed in the space–time domain $[0, \text{LENGTH}] \times [0, \text{LENGTH}] \times [0, \text{TOTAL_TIME}]$, we provide in

- `get_interior_points` the functionality to identify the points from the training of the residual loss, in
- `get_initial_points` the functionality to identify points for the training of the initial loss, and in
- `get_boundary_points` the functionality for training the boundary loss.

7.5. Loss functions

We provide interfaces for defining the loss functions inside the `Loss` class. Namely, we define the `residual_loss`, `initial_loss` and `boundary_loss`. Since the initial and boundary loss is universal, and residual loss is problem specific, we provide fixed implementations for the initial and boundary losses, assuming that the initial state is prescribed in the `initial_condition` routine and that the boundary conditions are zero Neumann. The code can be easily extended to support different boundary conditions.

```
class Loss:
...
    def residual_loss(self, pinn: PINN):
        x, y, t = get_interior_points \
            (self.x_domain, self.y_domain, \
             self.t_domain, self.n_points, pinn.device())
        loss = dfdt(pinn, x, y, t).to(device)
        - self.dTy(y, t)*dfdy(pinn, x, y, t).to(device)
        - self.Kx*dfdx(pinn, x, y, t, order=2).to(device)
        - self.Ky*dfdy(pinn, x, y, t, order=2).to(device)
        - self.source(y, t).to(device)
        return loss.pow(2).mean()

    def initial_loss(self, pinn: PINN):
        x, y, t = get_initial_points \
            (self.x_domain, self.y_domain, \
             self.t_domain, self.n_points, pinn.device())
        pinn_init = self.initial_condition(x, y)
        loss = f(pinn, x, y, t) - pinn_init
        return loss.pow(2).mean()

    def boundary_loss(self, pinn: PINN):
        down, up, left, right = get_boundary_points \
            (self.x_domain, self.y_domain, self.t_domain, \
             self.n_points, pinn.device())
        x_down, y_down, t_down = down
        x_up, y_up, t_up = up
        x_left, y_left, t_left = left
        x_right, y_right, t_right = right

        L_down = dfdy( pinn, x_down, y_down, t_down )
```

```
        L_up = dfdy( pinn, x_up, y_up, t_up )
        L_left = dfdx( pinn, x_left, y_left, t_left )
        L_right = dfdx( pinn, x_right, y_right, t_right )

        return L_down.pow(2).mean() + \
            L_up.pow(2).mean() + \
            L_left.pow(2).mean() + \
            L_right.pow(2).mean()
```

The initial condition is defined in the `initial_condition` routine, which returns a value of the initial condition at point $(x, y, 0)$.

```
# Initial condition
def initial_condition(x: torch.Tensor, y: torch.Tensor) \
    -> torch.Tensor:
...
    res = INITIAL POLLUTION DISTRIBUTION
        AS OBTAINED FROM FEM SOLVER
    return res
```

The minimization of the three losses, is the multi-objective optimization problem. The loss functions can be weighted $L = W_{\text{residual}}L_{\text{residual}} + W_{\text{initial}}L_{\text{initial}} + W_{\text{boundary}}L_{\text{boundary}}$ with the weights $(W_{\text{residual}}, W_{\text{initial}}, W_{\text{boundary}})$ selected automatically using the SoftAdapt algorithm [66].

The number of neurons and the number of layers in the PINNs can be estimated using the results of Jinchao Xu, showing the analogies between neural networks and linear and higher-order finite element methods [67,68]. The weights of the loss functions for the multi-objective optimization can be determined automatically using SoftAdapt algorithm [66].

7.6. Summer simulation

In this section, we present numerical results of the pollution dissipation computed for the vertical temperature profile during the summer day. In summer, temperature inversions are less common on the Spitzbergen than in winter. The temperature in the troposphere (lower layer of the atmosphere) usually decreases with altitude. The convergence of the loss functions is summarized in Fig. 22. The snapshots from the simulations are presented in Fig. 24. The pollution concentration units are dimensionless, and the goal of the simulation is to present the quantitative behavior of the pollution propagation with the temperature profile during the summer period. The pollution generated by the power plant dissipates due to the vertical temperature gradients.

7.7. Winter simulation

Now, we present numerical results of the pollution dissipation computed for the temperature profiles during the winter night. The convergence of the loss functions is summarized in Fig. 23. The snapshots from the simulations are presented in Fig. 25. The dimensionless pollution concentration units illustrate the quantitative behavior of the pollution propagation with the temperature profile from the Arctic night. Just like in urban environments, thermal inversions in the Arctic can trap pollutants. The absence of sunlight during the Arctic night leads to intense cooling of the Earth's surface. As the ground loses heat, the air directly above it also cools rapidly. The lack of solar heating during the Arctic night results in stable atmospheric conditions with minimal vertical air mixing. This stability allows the cold air to remain trapped near the surface. The simulation results show that the initial concentration of the pollution is trapped near the ground and it only dissipates through the borders of the domain. Constant source of the pollution, as from the power plants, will keep the pollution concentration high in the Longyearbyen valley.

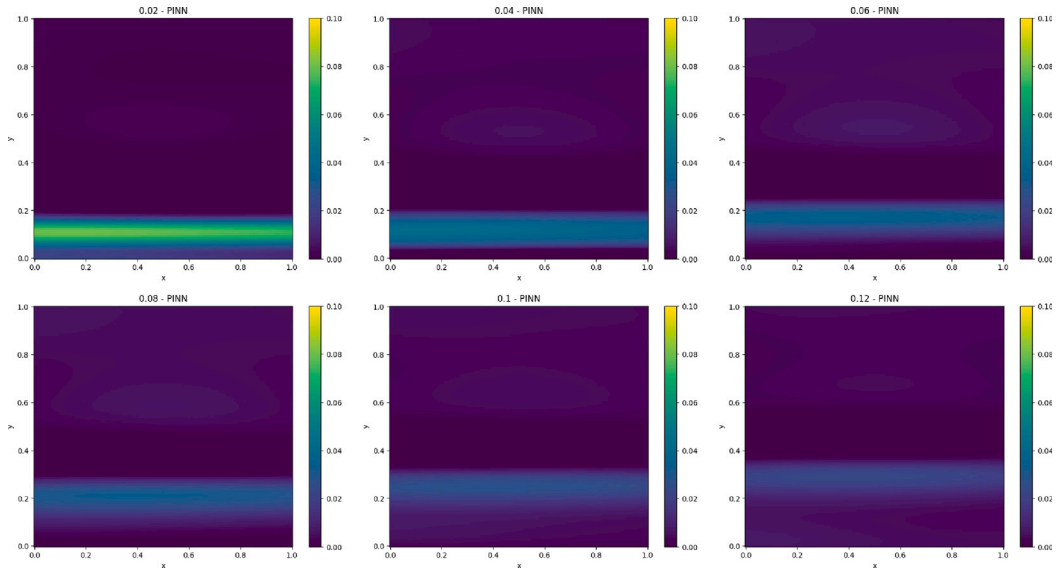


Fig. 24. Thermal inversion simulation for the Svalbard summer, where the temperature increases in the vertical direction. Due to the temperature gradient, the pollution moves up and dissipates.

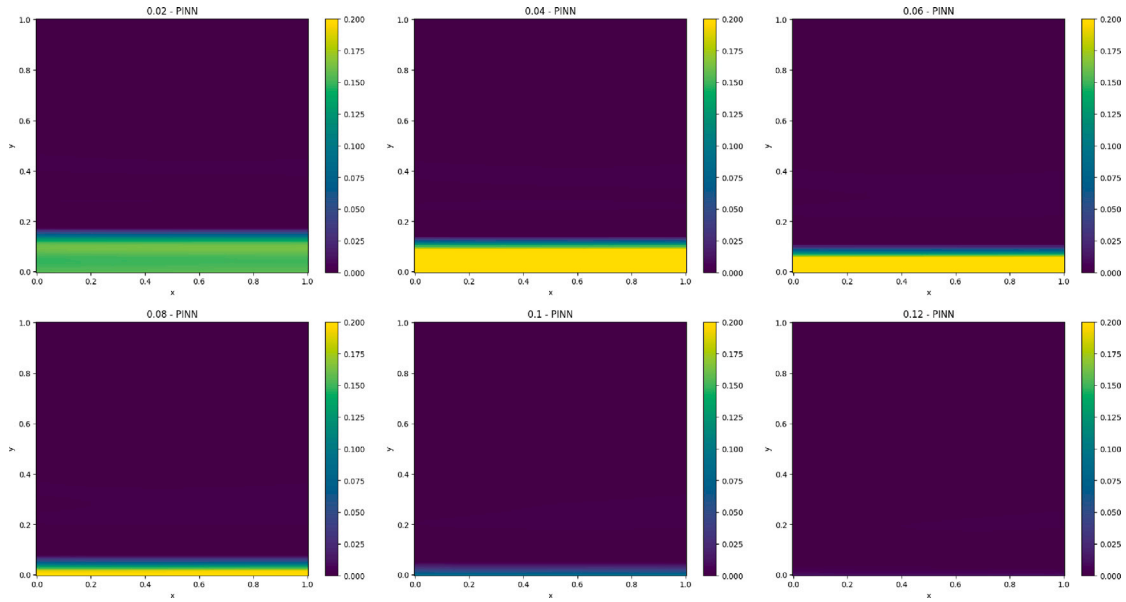


Fig. 25. Pollution concentration during the Svalbard winter where the temperature decreases in the vertical direction. The cold weather traps the pollution near the ground.

7.8. Notes on computational cost

The two-dimensional time-dependent PINN simulator execution on A100 Backend Google Compute Engine with Python 3 and GPU graphic card equipped with 83.48 GB of memory and 235.68 GB disc space takes around 15 min of computing time. This execution time is comparable with the execution of 200 times steps of the non-stationary three-dimensional graph-grammar-based finite element method solver on a laptop with 11th Gen Intel(R) Core(TM) i5-11500H @ 2.90 GHz, 2.92 GHz, and 32 GB of RAM, providing an estimate of 9 h of real-time pollution generation from a chimney. The finite element method simulation requires the development of the stabilized time-integration scheme, which, in our case, is the Crank–Nicolson method. The PINNs do not require the development of a stabilized time-integration scheme; the time-dependent problem is trained in the space–time domain.

8. Simulation of the thermal inversion with IGA-ADS code

We will compare our PINN solver to the higher-order and continuity isogeometric analysis (IGA) alternating direction (ADS) solver [61,62]. The choice of IGA-FEM follows from the fact that it delivers higher-order and continuity solutions, similar to the smooth neural network solution. Additionally, it delivers linear computational costs.

8.1. Derivation of the IGA-ADS solver

In the IGA-ADS approach, we first discretize in time, denoting $u_t(x, y) = u(x, y, t)$ and employ an explicit time integration scheme with time step size dt , namely

$$u_{t+1}(x, y) = u_t(x, y) + \frac{\partial T(y)}{\partial y} \frac{\partial u_t(x, y)}{\partial y} - 0.1 \frac{\partial u_t(x, y)}{\partial x^2} - 0.01 \frac{\partial u_t(x, y)}{\partial y^2}, \quad (25)$$

The weak formulation is obtained by testing with e.g. B-spline basis functions

$$\begin{aligned} (u_{t+1}(x, y), v(x, y)) = & (u_t(x, y), v(x, y)) + dt \left(\frac{\partial T(y)}{\partial y} \frac{\partial u(x, y)}{\partial y}, v(x, y) \right) \\ & + 0.1dt \left(\frac{\partial u(x, y)}{\partial x}, v(x, y) \right) + 0.01dt \left(\frac{\partial u(x, y)}{\partial y}, v(x, y) \right) \end{aligned} \quad (26)$$

We discretize with B-spline basis functions defined over the cube-shaped domain $\Omega = [0, 1]^3$

$$\begin{aligned} u^{t+1} &= \sum_{i=1, \dots, N_x; j=1, \dots, N_y} u_{ij}^{t+1} B_i^x B_j^y, \\ u^t &= \sum_{i=1, \dots, N_x; j=1, \dots, N_y} u_{ij}^t B_i^x B_j^y, \end{aligned} \quad (27)$$

and we test with B-spline basis functions

$$\begin{aligned} \sum_{ij} u_{ij}^{t+1} (B_i^x B_j^y, B_m^x B_n^y) &= \sum_{ij} u_{ij}^t (B_i^x B_j^y, B_m^x B_n^y) + \\ & dt \sum_{ij} u_{ij}^t \left(\frac{\partial T(y)}{\partial y} B_i^x \frac{\partial B_j^y}{\partial y}, B_m^x B_n^y \right) + \\ & 0.1dt \sum_{ij} u_{ij}^t \left(\frac{\partial B_i^x}{\partial x} B_j^y, \frac{\partial B_m^x}{\partial x} B_n^y \right) + \\ & 0.01dt \sum_{ij} u_{ij}^t \left(B_i^x \frac{\partial B_j^y}{\partial y}, B_m^x \frac{\partial B_n^y}{\partial y} \right) \end{aligned} \quad (28)$$

$m = 1, \dots, N_x; n = 1, \dots, N_y,$

where $(u, v) = \int_{\Omega} u(x, y) v(x, y) dx dy$. We separate directions

$$\begin{aligned} \sum_{ij} u_{ij}^{t+1} (B_i^x, B_m^x)_x (B_j^y, B_n^y)_y &= \sum_{ij} u_{ij}^t (B_i^x, B_m^x)_x (B_j^y, B_n^y)_y + \\ & dt \sum_{ij} u_{ij}^t (B_i^x, B_m^x)_x \frac{\partial T(y)}{\partial y} \left(\frac{\partial B_j^y}{\partial y}, B_n^y \right)_y + \\ & 0.1dt \sum_{ij} u_{ij}^t \left(\frac{\partial B_i^x}{\partial x}, \frac{\partial B_m^x}{\partial x} \right)_x (B_j^y, B_n^y)_y + \\ & 0.01dt \sum_{ij} u_{ij}^t (B_i^x, B_m^x)_x \left(\frac{\partial B_j^y}{\partial y}, \frac{\partial B_n^y}{\partial y} \right)_y \end{aligned} \quad (29)$$

$m = 1, \dots, N_x; n = 1, \dots, N_y,$

We introduce

$$\begin{aligned} \mathbf{M}_x &= \{(B_i^x, B_m^x)_x\}_{im} = \left\{ \int B_i^x B_m^x dx \right\}_{im}, \\ \mathbf{M}_y &= \{(B_j^y, B_n^y)_y\}_{jn} = \left\{ \int B_j^y B_n^y dy \right\}_{jn}, \\ \mathbf{A}_x &= \left\{ \left(\frac{\partial B_i^x}{\partial x}, B_m^x \right)_x \right\}_{im} = \left\{ \int \frac{\partial B_i^x}{\partial x} B_m^x dx \right\}_{im}, \\ \mathbf{A}_y &= \left\{ (B_j^y, \frac{\partial B_n^y}{\partial y})_y \right\}_{jn} = \left\{ \int B_j^y \frac{\partial B_n^y}{\partial y} dy \right\}_{jn}, \\ \mathbf{S}_x &= \left\{ \left(\frac{\partial B_i^x}{\partial x}, \frac{\partial B_m^x}{\partial x} \right)_x \right\}_{im} = \left\{ \int \frac{\partial B_i^x}{\partial x} \frac{\partial B_m^x}{\partial x} dx \right\}_{im}, \\ \mathbf{S}_y &= \left\{ \left(\frac{\partial B_j^y}{\partial y}, \frac{\partial B_n^y}{\partial y} \right)_y \right\}_{jn} = \left\{ \int \frac{\partial B_j^y}{\partial y} \frac{\partial B_n^y}{\partial y} dy \right\}_{jn} \end{aligned} \quad (30)$$

In general, the Kronecker product matrix $\mathcal{M} = \mathcal{A}_x \otimes \mathcal{B}_y$ over a 2D domain $\Omega = \Omega_x \times \Omega_y$ is defined as $\mathcal{M}_{ijmn} = \mathcal{A}_{xim} \mathcal{B}_{yjn}$. We rewrite

$$\begin{aligned} \mathbf{M}_x \otimes \mathbf{M}_y \mathbf{u}^{t+1} &= \mathbf{M}_x \otimes \mathbf{M}_y \mathbf{u}^t - dt \frac{\partial T(y)}{\partial y} \mathbf{M}_x \otimes \mathbf{A}_y \mathbf{u}^t + \\ & 0.1dt \mathbf{S}_x \otimes \mathbf{M}_y \mathbf{u}^t + 0.01dt \mathbf{M}_x \otimes \mathbf{S}_y \mathbf{u}^t \end{aligned} \quad (31)$$

We solve the problem (31) in a loop, with the time step size limited by the CFD condition. The solver employs the Kronecker product

Table 5
Comparison of PINN and IGA-ADS.

	PINN	IGA-ADS
Iterations	20,000 (training)	30,000 (time-steps)
Time	[0,1] (time interval)	$dt = 1e-5$ (time-step)
Resolution	15×15 points (training)	40×40 elements (mesh)
Approximation	2 layers with 600 neurons	quadratic C^1 B-splines
Machine	A100 google compute engine	11th Gen i5-1150H @ 2.90 HGz
Memory	83.48 GB RAM	32 GB RAM
Time	15 min	28 min

properties of the matrices, resulting in a linear computational cost of generation and solution of a single time step.

8.2. Comparison of PINNs with IGA-ADS

We have executed the IGA-ADS code for comparison with the PINN code. The snapshots from both simulations are presented in Fig. 26 for the summer simulations and in Fig. 27 for the winter simulations. We compare the parameters and the requirements of both simulations in Table 5. The execution of a single IGA-ADS simulation requires 30,000-time steps with $dt = 1e-5$. The computational domain is projected into $[0, 1]^2$ square, and the time interval is also scaled into $[0, 1]$. We employ a two-dimensional mesh with 40×40 finite elements and quadratic B-splines of C^1 continuity for discretization. The execution time takes around 28 min on a Laptop with 11th Gen Intel(R) Core(TM) i5-11500H with 2.92 GHz and 32 GB of RAM. We obtained the same quality results as the PINN code after 15 min of training on the A100 Backend Google Compute Engine with Python 3 and GPU graphic card equipped with 83.48 GB of memory.

9. Influence of the pollution propagation in Longyearbyen inhabitants

In this section, we investigate the influence of pollution propagation on the inhabitants. We compare the results of the survey with the conclusions from the simulation results.

9.1. Pollution propagation

By air pollution, we refer to the presence of potentially harmful and excessive quantities of such substances in the air that we breathe. The presence of these substances, known as pollutants, is natural to some extent, but special attention is given to man-made pollutants that pose a high risk to humans and, widely speaking, to all living organisms, climate, and the environment. Attention is particularly given when pollutant concentration is above certain thresholds where adverse effects are known to occur, though recent evidence has also been documented on the adverse health effects of air pollution at lower exposure levels [69].

The WHO guideline for annual average ($PM_{2.5}$) concentration is not greater than $5 \mu g/m^3$, and one of the most recognized frameworks for measuring and expressing air quality, adopted by (many) other countries and organizations globally, i.e., the Air Quality Index (AQI) [70] used by the United States Environmental Protection Agency, distinguishes the following ($PM_{2.5}$) concentration levels and its impact on health conditions [71] (see Table 1):

- $0-10 \mu g/m^3$: air quality is considered satisfactory (or “Good” using AQI nomenclature), and air pollution poses little or no risk;
- $10.1-35.4 \mu g/m^3$: air quality is (“Moderate”), i.e., acceptable; however, some pollutants may pose moderate health concerns for a small number of people;

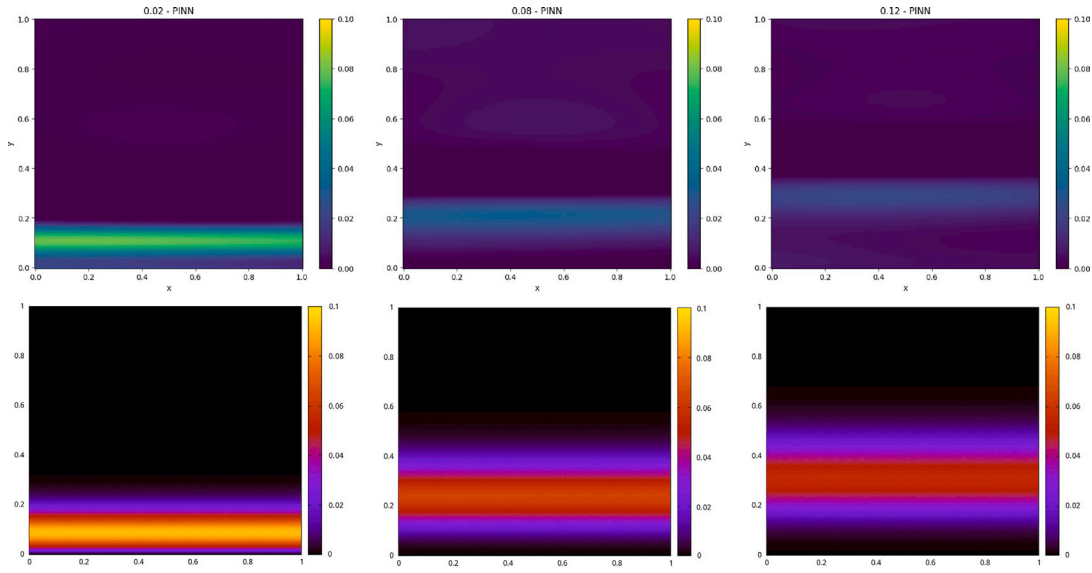


Fig. 26. Comparison of the PINN with IGA-ADS executed for the summer simulations.

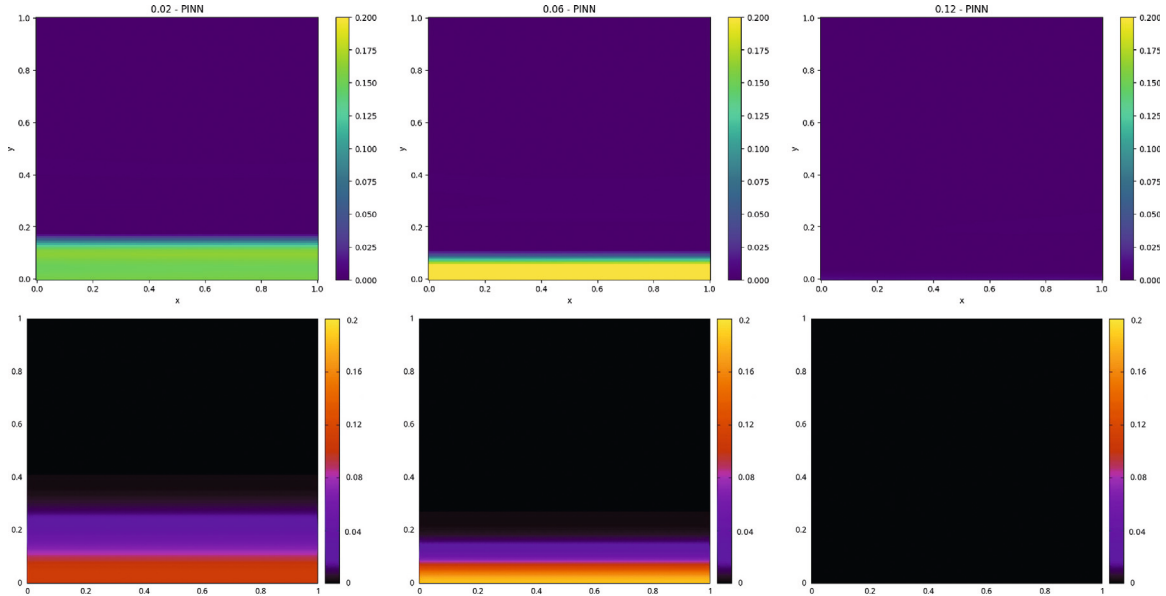


Fig. 27. Comparison of the PINN with IGA-ADS executed for the winter simulations.

- 35.5–55.4 $\mu\text{g}/\text{m}^3$: air quality is considered “Unhealthy for sensitive groups”, which means that members of sensitive groups (like children or elderly people [72]) may experience severe negative health effects, but the general population is less likely to be affected;
- 55.5–125.4 $\mu\text{g}/\text{m}^3$: air quality is considered “Unhealthy”, and everyone in the exposed population may experience negative health effects;
- 125.5–225.4 $\mu\text{g}/\text{m}^3$: air quality is considered “Very unhealthy”. It usually triggers health alerts since everyone in the exposed population may experience serious negative health issues;
- 225.5–500 $\mu\text{g}/\text{m}^3$: air quality is considered “Hazardous”, and the conditions are considered an emergency situation since the entire exposed population is likely to be strongly negatively affected.

When it comes to Norway, the $(\text{PM}_{2.5})$ concentration levels are well below the limits set by both the World Health Organization and the European Union and among the lowest in Europe. With an annual

Table 6

AQI categories.

AQI category	AQI values	$(\text{PM}_{2.5})$ [$\mu\text{g}/\text{m}^3$] (breakpoints)
Good	0–50	0.0–10.0
Moderate	51–100	10.1–35.4
Unhealthy for sensitive groups	101–150	35.5–55.4
Unhealthy	151–200	55.5–125.4
Very unhealthy	201–300	125.5–225.4
Hazardous	301–	225.5–

average $(\text{PM}_{2.5})$ concentration in the range 5 to 8 $\mu\text{g}/\text{m}^3$, while in the larger cities of Oslo and Bergen, it is slightly higher and is 7–8 $\mu\text{g}/\text{m}^3$ and 6–7 $\mu\text{g}/\text{m}^3$, respectively.

In this paper, we focused on the conditions in Longyearbyen, where the pollutants are trapped around the ground level during the Arctic nights. We compare the conditions that existed before the old power

plant was closed [64], where the pollutants were emitted by the coal-fired power plant, with the actual conditions, where the newest diesel engine power plant emits the pollutants.

9.2. Survey among inhabitants

It is interesting to see how Longyearbyen residents subjectively perceive their health and living conditions, especially in terms of air pollution. We conducted a survey among randomly selected residents of Longyearbyen [73]. The average age of respondents was 36, and 37% of them were male. Almost every participant claimed to have a good (63%) or very good (31%) health status. About one-third of the residents live in Svalbard less than one year and about 10% were Svalbard residents for more than ten years. Most of them (~79%) live in the city center of Longyearbyen. Nearly half of the respondents (~46%) work physically and have an active lifestyle; however, around 63% spend less than two hours outside per day. None of the respondents smoke cigarettes, however, 21% experienced respiratory issues, mostly during winter months and in the evening or at night. Around 15% reported having cough or dyspnoea, and 5% claimed that their health problems intensified since moving to Svalbard. So, the first (more qualitative than quantitative) conclusion is that around one-fifth of respondents reported respiratory issues, mostly during winter months and in the evening or at night, i.e., for the periods and seasons where the pollutants may not be easily transferred to other areas while being trapped around the ground level.

9.3. Conclusions from the computer simulations

The computer simulations of pollution propagation described in our article are only qualitative in nature but nevertheless allow us to draw the following conclusions. The graph-grammar-based simulations of the pollution propagation from the power plants show that due to the terrain and the direction and average speed of the wind, most of the generated pollution is dispersed in the valley space, up to a height of about 100 m (Figs. 18–20). The PINN simulations of the thermal inversion, they show that during the Arctic night the thermal inversion traps the pollutants near the surface.

Coal-burning power plant in a winter season generates 200–400 milligrams of PM_{2.5} pollutants for every kilogram of coal burned [74–76]. Coal combustion at the Spitzbergen power plant reached up to 3.4 tons of coal for each hour of power plant operation. As our simulations showed, due to the terrain and the direction and average speed of the wind, most of the pollution generated by the power plants is dispersed in the valley space, up to a height of about 100 m (Figs. 18–20). The area over which the pollutants disperse is about 100 km², multiplying by the height of 100 m yields a volume of 10⁷ m³ over which the PM_{2.5} particles burned in 4 h disperse (Figs. 19–20). Hence, in 4 h, the power plant burns 13,600 kg of coal, which gives an average of 13,600 · 300 = 4,080,000 mg of PM_{2.5}, which disperses in a volume of 10⁷ m³. This gives an average concentration of 2.45 milligrams [mg] of PM_{2.5} per cubic meter, which is equal to 2450 micrograms [μg] per cubic meter (which is considered Hazardous according to AQI Categories, see Table 6). The estimation of the PM_{2.5} pollution generated by the diesel engine power plant in a winter season is the following. A 1 Mega-Wat [MW] diesel engine power plant operating at full load for an hour (assuming the technology with 0.3 gram of PM_{2.5} emission per kilo-Wat-hour [g/kWh] [77]) generates power output of 1000 kilo-Wat [kW], which results in a PM_{2.5} emission that can be estimated as 1000 · 0.3 [g/kWh], which results in 300 gram of PM_{2.5} per hour [g/h]. The diesel power plant at Longyearbyen generates 11 Mega-Wat [MW] of power [78], which results in a total of 11 · 300 · 4 = 13,200 gram of PM_{2.5} per 4 h. Due to the topography and the wind condition, this PM_{2.5} spreads in the valley, contributing to an average of 13,200 · 10⁷ gram per m³ = 13,200,000,000 · 10⁷ microgram [μg] per m³ = 1320 μg per m³ (which is also considered Hazardous according to AQI Categories, see Table 1).

10. Conclusions

We presented an original model describing the production of graph grammars, transformation sequences of graphs representing a computational grid, expressing an algorithm for adapting a three-dimensional computational mesh that does not generate hanging nodes. The graph transformation rules model the Rivara algorithms. Its the idea is to break elements along the longest edges and propagate the refinement to adjacent elements to avoid hanging nodes in three-dimensional computational grids. The graph transformations were used to generate a computational grid for simulating pollution propagation from coal-fired and diesel engine power plants in Longyearbyen, Spitzbergen. We also introduce a computational code performing Physics Informed Neural Networks simulations of the pollution propagation. The PINNs are attractive alternatives for simulations carried out using the finite element method. They do not require a time integration scheme and do not generate stability problems encountered by time integration. On the other hand, successful training of the PINN model is a multi-objective optimization problem, and it requires guessing several model parameters, such as the number of layers of the neural network, the number of neurons, the training rate, and the loss function weights for training. Some analogies between neural networks and linear and higher-order finite element methods can be found in works of [67,68]. They enable us to estimate the size of the neural networks. The weights of the loss functions for the multi-objective optimization can be determined automatically using the SoftAdapt algorithm [66]. Nevertheless, the actual state-of-the-art PINNs enable, in the authors' opinion, the successful and efficient application of the two-dimensional PINN model in engineering applications. From the performed simulations, we can estimate that the coal-fired power plant for 4 h in a winter season generated a pollution of 2450 PM_{2.5} micrograms [μg] per cubic meter (which is considered Hazardous according to AQI Categories), and the diesel based power plant in a winter season generates 1320 PM_{2.5} micrograms [μg] per cubic meter (which is also considered Hazardous according to AQI Categories).

CRedit authorship contribution statement

Maciej Sikora: Visualization, Software, Investigation. **Albert Oliver-Serra:** Visualization, Validation, Software, Project administration, Methodology, Funding acquisition. **Leszek Siwik:** Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Data curation, Conceptualization. **Natalia Leszczyńska:** Writing – review & editing, Writing – original draft, Visualization, Validation, Investigation, Data curation. **Tomasz Maciej Ciesielski:** Supervision, Methodology, Investigation, Conceptualization. **Eirik Valseth:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Conceptualization. **Jacek Leszczyński:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Data curation, Conceptualization. **Anna Paszyńska:** Writing – review & editing, Writing – original draft, Project administration, Methodology, Formal analysis, Conceptualization. **Maciej Paszyński:** Writing – review & editing, Writing – original draft, Validation, Supervision, Software, Project administration, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The Authors gratefully acknowledge the support and assistance of The Polish Polar Station Hornsund for help with data collection.

The authors are grateful for support from the funds the Polish Ministry of Science and Higher Education assigned to AGH University of Krakow. The work is supported by “Excellence initiative - research university” for the AGH University of Krakow.

The work of Albert Oliver-Serra was supported by “Ayudas para la recualificación del sistema universitario español” grant funded by the ULPGC, the Ministry of Universities by Order UNI/501/2021 of 26 May, and the European Union-Next Generation EU Funds.

Data availability

Data will be made available on request.

References

- [1] K. Podsiadło, A.O. Serra, A. Paszyńska, R. Montenegro, I. Henriksen, M. Paszyński, K. Pingali, Parallel graph-grammar-based algorithm for the longest-edge refinement of triangular meshes and the pollution simulations in Lesser Poland area, *Eng. Comput.* 37 (2021) 3857–3880.
- [2] A.N. Brooks, T.J. Hughes, Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations, *Comput. Methods Appl. Mech. Engrg.* 32 (1) (1982) 199–259, [http://dx.doi.org/10.1016/0045-7825\(82\)90071-8](http://dx.doi.org/10.1016/0045-7825(82)90071-8), URL <https://www.sciencedirect.com/science/article/pii/0045782582900718>.
- [3] M.C. Rivara, Algorithms for refining triangular grids suitable for adaptive and multigrid techniques, *Int. J. Numer. Methods Eng.* 20 (4) (1984) 745–756.
- [4] M.C. Rivara, Mesh refinement processes based on the generalized bisection of simplices, *SIAM J. Numer. Anal.* 21 (3) (1984) 604–613.
- [5] A. Paszyńska, M. Paszyński, E. Grabska, Graph transformations for modeling hp-adaptive finite element method with triangular elements, in: *Proceedings of the 8th International Conference on Computational Science, Part III, ICCS '08*, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 604–613, http://dx.doi.org/10.1007/978-3-540-69389-5_68.
- [6] A. Paszyńska, M. Paszyński, E. Grabska, Graph transformations for modeling hp-adaptive finite element method with mixed triangular and rectangular elements, in: *Proceedings of the 9th International Conference on Computational Science*, in: *ICCS 2009*, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 875–884, http://dx.doi.org/10.1007/978-3-642-01973-9_97.
- [7] M. Paszyński, On the parallelization of self-adaptive hp-finite element methods part i. composite programmable graph grammar model, *Fund. Inform.* 93 (4) (2009) 411–434.
- [8] M. Paszyński, On the parallelization of self-adaptive hp-finite element methods part ii. partitioning communication agglomeration mapping (PCAM) analysis, *Fund. Inform.* 93 (4) (2009) 435–457.
- [9] W.B.F. Ryan, S.M. Carbotte, J.O. Coplan, S. O'Hara, A. Melkonian, R. Arko, R.A. Weissel, V. Ferrini, A. Goodwillie, F. Nitsche, J. Bonczkowski, R. Zemsky, Global multi-resolution topography synthesis, *Geochem. Geophys. Geosystems* 10 (3) (2009) <http://dx.doi.org/10.1029/2008GC002332>, arXiv:<https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2008GC002332>, URL <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2008GC002332>.
- [10] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.* 378 (2019) 686–707, <http://dx.doi.org/10.1016/j.jcp.2018.10.045>.
- [11] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, et al., Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, *IEEE Signal Process. Mag.* 29 (6) (2012) 82–97.
- [12] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, *Commun. ACM* 60 (6) (2017) 84–90.
- [13] M. Gheisari, G. Wang, M.Z.A. Bhuiyan, A survey on deep learning in big data, in: *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing, EUC, Vol. 2*, IEEE, 2017, pp. 173–180.
- [14] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.* 378 (2019) 686–707.
- [15] S. Cai, Z. Mao, Z. Wang, M. Yin, G.E. Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: A review, *Acta Mech. Sin.* 37 (12) (2021) 1727–1738.
- [16] Z. Mao, A.D. Jagtap, G.E. Karniadakis, Physics-informed neural networks for high-speed flows, *Comput. Methods Appl. Mech. Engrg.* 360 (2020) 112789.
- [17] J. Ling, A. Kurawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, *J. Fluid Mech.* 807 (2016) 155–166, <http://dx.doi.org/10.1017/jfm.2016.615>.
- [18] L. Sun, H. Gao, S. Pan, J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, *Comput. Methods Appl. Mech. Engrg.* 361 (2020) <http://dx.doi.org/10.1016/j.cma.2019.112732>.
- [19] N. Wandel, M. Weinmann, M. Neidlin, R. Klein, Spline-PINN: Approaching PDEs without data using fast, physics-informed Hermite-spline CNNs, *Proc. AAAI Conf. Artif. Intell.* 36 (8) (2022) 8529–8538, <http://dx.doi.org/10.1609/aaai.v36i8.20830>, URL <https://ojs.aaai.org/index.php/AAAI/article/view/20830>.
- [20] M. Rasht-Beheht, C. Huber, K. Shukla, G.E. Karniadakis, Physics-Informed Neural Networks (PINNs) for wave propagation and full waveform inversions, *J. Geophys. Res.: Solid Earth* 127 (5) (2022) e2021JB023120.
- [21] N. Geneva, N. Zabarar, Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks, *J. Comput. Phys.* 403 (2020) <http://dx.doi.org/10.1016/j.jcp.2019.109056>.
- [22] S. Goswami, C. Anitescu, S. Chakraborty, T. Rabczuk, Transfer learning enhanced physics informed neural network for phase-field modeling of fracture, *Theor. Appl. Fract. Mech.* 106 (2020) <http://dx.doi.org/10.1016/j.tafmec.2019.102447>.
- [23] M. Alber, A.B. Tepole, W.R. Cannon, S. De, S. Dura-Bernal, K. Garikipati, G. Karniadakis, W.W. Lytton, P. Perdikaris, L. Petzold, E. Kuhl, Integrating machine learning and multiscale modeling-perspectives, challenges, and opportunities in the biological biomedical, and behavioral sciences, *NPJ Digit. Med.* 2 (2019) <http://dx.doi.org/10.1038/s41746-019-0193-y>.
- [24] G. Kissas, Y. Yang, E. Hwuang, W.R. Witschey, J.A. Detre, P. Perdikaris, Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks, *Comput. Methods Appl. Mech. Engrg.* 358 (2020) <http://dx.doi.org/10.1016/j.cma.2019.112623>.
- [25] H. Jin, M. Mattheakis, P. Protopapas, Physics-informed neural networks for quantum eigenvalue problems, in: *2022 International Joint Conference on Neural Networks, IJCNN, 2022*, pp. 1–8, <http://dx.doi.org/10.1109/IJCNN55064.2022.9891944>.
- [26] R. Nellikkath, S. Chatzivasileiadis, Physics-informed neural networks for minimising worst-case violations in DC optimal power flow, in: *2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), 2021*, pp. 419–424, <http://dx.doi.org/10.1109/SmartGridComm51999.2021.9632308>.
- [27] X. Huang, H. Liu, B. Shi, Z. Wang, K. Yang, Y. Li, M. Wang, H. Chu, J. Zhou, F. Yu, B. Hua, B. Dong, L. Chen, A universal PINNs method for solving partial differential equations with a point source, in: *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI- 22, 2022*, pp. 3839–3846.
- [28] Y. Yang, P. Perdikaris, Adversarial uncertainty quantification in physics-informed neural networks, *J. Comput. Phys.* 394 (2019) 136–152, <http://dx.doi.org/10.1016/j.jcp.2019.05.027>.
- [29] F. Sun, Y. Liu, H. Sun, Physics-informed spline learning for nonlinear dynamics discovery, in: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI- 21, 2021*, pp. 2054–2061.
- [30] J. Kim, K. Lee, D. Lee, S.Y. Jin, N. Park, DPM: A novel training method for physics-informed neural networks in extrapolation, in: *35th AAAI Conference on Artificial Intelligence, AAAI 2021, 2021*, pp. 8146–8154.
- [31] Y. Chen, L. Lu, G.E. Karniadakis, L. Dal Negro, Physics-informed neural networks for inverse problems in nano-optics and metamaterials, *Opt. Express* 28 (8) (2020) 11618–11633.
- [32] S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs, *IMA J. Numer. Anal.* 42 (2) (2022) 981–1022.
- [33] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, S.G. Johnson, Physics-informed neural networks with hard constraints for inverse design, *SIAM J. Sci. Comput.* 43 (6) (2021) B1105–B1132, <http://dx.doi.org/10.1137/21M1397908>.
- [34] M.S. Aslam, S. Shamrooz, H. Bilal, Fuzzy PD-sliding mode control design for networked system with time delays, *Eur. J. Control* 78 (2024) 101020, <http://dx.doi.org/10.1016/j.ejcon.2024.101020>, URL <https://www.sciencedirect.com/science/article/pii/S0947358024000803>.
- [35] M.S. Aslam, H. Bilal, W. jer Chang, A. Yahya, I.A. Badruddin, S. Kamangar, M. Hussien, Indirect adaptive observer control (IAOC) design for truck-trailer model based on T-S fuzzy system with unknown nonlinear function, *Complex Intell. Syst.* 10 (2024) 7311–7331.
- [36] M.S. Aslam, H. Bilal, S. S.Band, P. Ghasemi, Modeling of nonlinear supply chain management with lead-times based on Takagi-Sugeno fuzzy control model, *Eng. Appl. Artif. Intell.* 133 (2024) 108131, <http://dx.doi.org/10.1016/j.engappai.2024.108131>, URL <https://www.sciencedirect.com/science/article/pii/S0952197624002896>.
- [37] H. Bilal, M.S. Aslam, Y. Tian, A. Yahya, B. Abu Izneid, Enhancing trajectory tracking and vibration control of flexible robots with hybrid fuzzy ADRC and input shaping, *IEEE Access* 12 (2024) 150574–150591, <http://dx.doi.org/10.1109/ACCESS.2024.3453944>.

- [38] M.S. Aslam, H. Bilal, M. Hayajneh, Lqr-based PID controller with variable load tuned with data-driven methods for double inverted pendulum, *Data Anal. Mach. Learn. Soft Comput.* (28) (2024) 325–338.
- [39] H. Bilal, M.S. Obaidat, M. Shamrooz Aslam, J. Zhang, B. Yin, K. Mahmood, Online fault diagnosis of industrial robot using IoRT and hybrid deep learning techniques: An experimental approach, *IEEE Internet Things J.* 11 (19) (2024) 31422–31437, <http://dx.doi.org/10.1109/JIOT.2024.3418352>.
- [40] A. Kumar, S. Wang, A.M. Shaikh, H. Bilal, B. Lu, S. Song, Building on prior lightweight CNN model combined with LSTM-AM framework to guide fault detection in fixed-wing UAVs, *Int. J. Mach. Learn. Cybern.* 15 (2024) 4175–4191.
- [41] H. Bilal, F. Ahmed, M.S. Aslam, Q. Li, J. Hou, B. Yin, A blockchain-enabled approach for privacy-protected data sharing in internet of robotic things networks, *Hum.-Centric Comput. Inf. Sci.* 14 (71) (2024).
- [42] M. Shamrooz Aslam, H. Bilal, W.-J. Chang, A. Yahya, I. Anjum Badruddin, S. Kamangar, M. Hussien, Formation control of heterogeneous multi-agent systems under fixed and switching hierarchies, *IEEE Access* 12 (2024) 97868–97882, <http://dx.doi.org/10.1109/ACCESS.2024.3419815>.
- [43] J. Zheng, Y. Yang, VT-PINN: Variable transformation improves physics-informed neural networks for approximating partial differential equations, *Appl. Soft Comput.* 167 (2024) 112370, <http://dx.doi.org/10.1016/j.asoc.2024.112370>, URL <https://www.sciencedirect.com/science/article/pii/S156849462401144X>.
- [44] A. Kaplarević-Mališić, B. Andrijević, F. Bojović, S. an Nikolić, L. Krstić, B. Stojanović, M. Ivanović, Identifying optimal architectures of physics-informed neural networks by evolutionary strategy, *Appl. Soft Comput.* 146 (2023) 110646, <http://dx.doi.org/10.1016/j.asoc.2023.110646>, URL <https://www.sciencedirect.com/science/article/pii/S1568494623006646>.
- [45] F. Cao, X. Guo, X. Dong, D. Yuan, wbPINN: Weight balanced physics-informed neural networks for multi-objective learning, *Appl. Soft Comput.* 170 (2025) 112632, <http://dx.doi.org/10.1016/j.asoc.2024.112632>, URL <https://www.sciencedirect.com/science/article/pii/S1568494624014066>.
- [46] J. Pan, X. Xiao, L. Guo, X. Feng, A high resolution physics-informed neural networks for high-dimensional convection–diffusion–reaction equations, *Appl. Soft Comput.* 148 (2023) 110872, <http://dx.doi.org/10.1016/j.asoc.2023.110872>, URL <https://www.sciencedirect.com/science/article/pii/S1568494623008906>.
- [47] J.E. Kittelsen, E.A. Antonelo, E. Camponogara, L.S. Inslund, Physics-informed neural networks with skip connections for modeling and control of gas-lifted oil wells, *Appl. Soft Comput.* 158 (2024) 111603, <http://dx.doi.org/10.1016/j.asoc.2024.111603>, URL <https://www.sciencedirect.com/science/article/pii/S1568494624003776>.
- [48] Z. Tang, S. Dong, X. Yang, J. Zhang, Application of a parallel physics-informed neural network to solve the multi-body dynamic equations for full-scale train collisions, *Appl. Soft Comput.* 142 (2023) 110328, <http://dx.doi.org/10.1016/j.asoc.2023.110328>, URL <https://www.sciencedirect.com/science/article/pii/S1568494623003460>.
- [49] J. Chen, F. Zhu, Y. Han, D. Ren, Target temperature field prediction via a thermodynamic knowledge-based artificial neural network, *Appl. Soft Comput.* 174 (2025) 112972, <http://dx.doi.org/10.1016/j.asoc.2025.112972>, URL <https://www.sciencedirect.com/science/article/pii/S1568494625002832>.
- [50] K.K.R. Samal, K.S. Babu, S.K. Das, Multi-output spatio-temporal air pollution forecasting using neural network approach, *Appl. Soft Comput.* 126 (2022) 109316, <http://dx.doi.org/10.1016/j.asoc.2022.109316>, URL <https://www.sciencedirect.com/science/article/pii/S1568494622004914>.
- [51] J. Kuśmierczyk-Michulec, J. Baré, Climate change as observed through the IMS radionuclide station in Spitzbergen, *Sci. Rep.* 14 (1) (2024) 10906, <http://dx.doi.org/10.1038/s41598-024-59319-6>.
- [52] D. Mewes, C. Jacobi, Horizontal temperature fluxes in the arctic in CMIP5 model results analyzed with self-organizing maps, *Atmosphere* 11 (3) (2020) <http://dx.doi.org/10.3390/atmos11030251>, URL <https://www.mdpi.com/2073-4433/11/3/251>.
- [53] L.B.Q. Nguyen, I. Zelinka, V. Snasel, L.T.T. Nguyen, B. Vo, Subgraph mining in a large graph: A review, *WIREs Data Min. Knowl. Discov.* 12 (4) (2022) e1454, <http://dx.doi.org/10.1002/widm.1454>, [arXiv:https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1454](https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1454), URL <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1454>.
- [54] H. Huang, F. Yu, X. Jiang, J. Xie, Y. Du, J. Cao, J. Shan, H. Zhang, Z. Guan, Mesh deformation and adaptive refinement for multi-physics field, 2025, <http://dx.doi.org/10.2139/ssrn.5124132>, SSRN.
- [55] A. Mosialek, A. Szaflarski, R. Pych, M. Kisiel-Dorohinicki, M. Paszyński, A. Paszyńska, Graph-grammar based longest-edge refinement algorithm for three-dimensional optimally p refined meshes with tetrahedral elements, in: M. Paszyński, D. Kranzlmüller, V.V. Krzhizhanovskaya, J.J. Dongarra, P.M. Sloot (Eds.), *Computational Science – ICCS 2021*, Springer International Publishing, Cham, 2021, pp. 200–213.
- [56] M. Sikora, P. Krukowski, A. Paszyńska, M. Paszyński, Comparison of physics informed neural networks and finite element method solvers for advection-dominated diffusion problems, *J. Comput. Sci.* 81 (2024) 102340, <http://dx.doi.org/10.1016/j.jocs.2024.102340>, URL <https://www.sciencedirect.com/science/article/pii/S187750324001339>.
- [57] V. Calo, M. Łoś, Q. Deng, I. Muga, M. Paszyński, Isogeometric Residual Minimization Method (iGRM) with direction splitting preconditioner for stationary advection-dominated diffusion problems, *Comput. Methods Appl. Mech. Engrg.* 373 (2021) 113214, <http://dx.doi.org/10.1016/j.cma.2020.113214>, URL <https://www.sciencedirect.com/science/article/pii/S0045782520303996>.
- [58] T.G. Grossmann, U.J. Komorowska, J. Latz, C.-B. Schönlieb, Can physics-informed neural networks beat the finite element method? *IMA J. Appl. Math.* 89 (1) (2024) 143–174, <http://dx.doi.org/10.1093/imat/hxae011>, [arXiv:https://academic.oup.com/imat/article-pdf/89/1/143/58325885/hxae011.pdf](https://academic.oup.com/imat/article-pdf/89/1/143/58325885/hxae011.pdf).
- [59] European centre for medium-range weather forecasts: ECMWF IFS high resolution operational forecasts, 2016, URL <https://doi.org/10.5065/D68050ZV>, <https://rda.ucar.edu/datasets/d113001/>.
- [60] N.N. Davis, J. Badger, A.N. Hahmann, B.O. Hansen, N.G. Mortensen, M. Kelly, X.G. Larsén, B.T. Olsen, R. Floors, G. Liczcano, P. Casso, O. Lacave, A. Bosch, I. Bauwens, O.J. Knight, A.P. van Loon, R. Fox, T. Parvanyan, S.B.K. Hansen, D. Heathfield, M. Onninen, R. Drummond, The global wind atlas: A high-resolution dataset of climatologies and associated web-based application, *Bull. Am. Meteorol. Soc.* 104 (8) (2023) E1507 – E1525, <http://dx.doi.org/10.1175/BAMS-D-21-0075.1>, URL <https://journals.ametsoc.org/view/journals/bams/104/8/BAMS-D-21-0075.1.xml>.
- [61] K. Misan, M. Kozięja, M. Łoś, D. Gryboś, J. Leszczyński, P. Maczuga, M. Woźniak, A.O. Serra, M. Paszyński, The first scientific evidence for the hail cannon, in: J. Mikyška, C. de Mulatier, M. Paszyński, V.V. Krzhizhanovskaya, J.J. Dongarra, P.M. Sloot (Eds.), *Computational Science – ICCS 2023*, Springer Nature Switzerland, Cham, 2023, pp. 177–190.
- [62] M. Łoś, L. Siwik, M. Woźniak, D. Gryboś, P. Maczuga, A. Oliver-Serra, J. Leszczyński, M. Paszyński, Shock waves generators: From prevention of hail storms to reduction of the smog in urban areas — experimental verification and numerical simulations, *J. Comput. Sci.* 77 (2024) 102238, <http://dx.doi.org/10.1016/j.jocs.2024.102238>, URL <https://www.sciencedirect.com/science/article/pii/S187750324000310>.
- [63] R. Courant, K. Friedrichs, H. Lewy, Über die partiellen differenzengleichungen der mathematischen physik, *Math. Ann.* 100 (1928) 32–74, URL <https://api.semanticscholar.org/CorpusID:120760331>.
- [64] EnergiWatch, Nyheter om fornybar energi, 2024, URL <https://energiwatch.no/nyheter/fornybar/article16527415.ece> (Accessed 25 November 2024).
- [65] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- [66] A.A. Heydari, C.A. Thompson, A. Mehmood, SoftAdapt: Techniques for adaptive loss weighting of neural networks with multi-part loss functions, 2019, arXiv: [arXiv:1912.12355](https://arxiv.org/abs/1912.12355), URL <https://arxiv.org/abs/1912.12355>.
- [67] J. He, L. Li, J. Xu, C. Zheng, J. He, Relu deep neural networks and linear finite elements, *J. Comput. Math.* 38 (3) (2020) 502–527, <http://dx.doi.org/10.4208/jcm.1901-m2018-0160>.
- [68] J. He, J. Xu, Deep neural networks and finite elements of any order on arbitrary dimensions, 2024, arXiv:2312.14276, URL <https://arxiv.org/abs/2312.14276>.
- [69] R. Vilcassim, G.D. Thurston, Gaps and future directions in research on health effects of air pollution, *EBioMedicine* 93 (2023) <http://dx.doi.org/10.1016/j.ebiom.2023.104668>.
- [70] U.S. Environmental Protection Agency, Air data basic information, 2024, URL <https://www.epa.gov/outdoor-air-quality-data/air-data-basic-information> (Accessed 25 November 2024).
- [71] Environmental Protection Agency, Reconsideration of the national ambient air quality standards for particulate matter, 2023, URL <https://www.federalregister.gov/documents/2023/01/27/2023-00269/reconsideration-of-the-national-ambient-air-quality-standards-for-particulate-matter>. [cited 13 August 2024].
- [72] D. Vilcins, R.C. Christofferson, J.-H. Yoon, S.N. Nazli, P.D. Sly, S.A. Cormier, G. Shen, Updates in air pollution: Current research and future challenges, *Ann. Glob. Heal.* 90 (1) (2024) 176–194, <http://dx.doi.org/10.5334/aogh.4363>.
- [73] N. Leszczyńska, A. Nasiek, B. Chłapek, K. Krzywiecka, The impact of air pollution on the respiratory system issues among the inhabitants of Svalbard (Longyearbyen), in: *The 70th International Medical Congress of Silesia*, 2024.
- [74] L. Henneman, C. Choirat, I. Dedoussi, F. Dominici, J. Roberts, C. Zigler, Mortality risk from United States coal electricity generation, *Sci.* 382 (6673) (2023) 941–946, <http://dx.doi.org/10.1126/science.adf4915>, URL <https://www.science.org/doi/pdf/10.1126/science.adf4915>, URL <https://www.science.org/doi/abs/10.1126/science.adf4915>.
- [75] Z. Asif, Z. Chen, H. Wang, Y. Zhu, Update on air pollution control strategies for coal-fired power plants, *Clean Technol. Env. Policy* 24 (8) (2022) 2329–2347.
- [76] W. Wilczyńska-Michalik, J. Dańko, M. Michalik, Characteristics of particulate matter emitted from a coal-fired power plant, *Pol. J. Environ. Stud.* (29) (2020).
- [77] Y.-C. Lin, Y.-C. Li, K.T. Amesho, F.-C. Chou, P.-C. Cheng, Characterization and quantification of PM_{2.5} emissions and PAHs concentration in PM_{2.5} from the exhausts of diesel vehicles with various accumulated mileages, *Sci. Total Environ.* 660 (2019) 188–198, <http://dx.doi.org/10.1016/j.scitotenv.2019.01.007>, URL <https://www.sciencedirect.com/science/article/pii/S0048969719300075>.
- [78] D. Nikel, Svalbard: World's northernmost coal power plant closes, *Life Nor.* (2023) URL <https://www.lifeinnorway.net/northernmost-coal-power-plant-closes/>.