WIP Urban Wind Flow Modeling with Physics-Informed Neural Networks (PINNs)

- Author: Andres Roncal

- Institution: IAAC: AI 2023-24

- Date: [Add the date]

Abstract

This paper presents the development of a web-based platform for urban wind flow modeling using Physics-Informed Neural Networks (PINNs). By integrating real-time data from NOAA, OpenWeatherMap, and OpenStreetMap, the project aims to simulate wind flow in urban environments. The methodology includes stages of data collection, PINN model development,and frontend and backend integration. The platform leverages NVIDIA Modulus and PyTorch to implement neural networks, offering a tool to optimize building designs and improve thermal performance. The results demonstrate the potential of PINNs in urban wind flow simulations and highlight areas for future research and development in sustainable urban design.

Importance of Urban Wind Flow Modeling

Wind flow modeling in urban environments is needed for understanding and optimizing the interactions between wind patterns and structures. The ability to simulate wind flow pedestrian comfort, reducing energy consumption, and mitigating the urban heat island effect. By accurately modeling wind patterns, designers can design buildings and public spaces that use natural ventilation, lowering reliance on mechanical cooling systems. Traditional wind modeling approaches often are paid and not easily accessible. So tools such as PINNs are needed to provide an open source option.

PINNs

Physics-Informed Neural Networks (PINNs) represent the field of computational fluid dynamics (CFD), offering an approach that embeds physical laws directly into the neural network training process. PINNs incorporate partial differential equations (PDEs) such as the Navier-Stokes equations, ensuring that the predictions align with established physical principles. This integration of physics-based constraints enhances the model's accuracy and reliability, particularly in complex and dynamic environments like urban wind flow simulations. In this project, PINNs are utilized to model the patterns of wind flow within urban landscapes by leveraging real-time data inputs. By incorporating wind conditions and urban geometry into the neural network, the project aims to create a simulation tool that adapts to changing environmental factors, providing insights for designers.

Objectives

The project is an open-source, web-based platform that implements the use of neural networks, to simulate urban wind flow. This platform is designed to integrate real-time data from various sources, such as NOAA, OpenWeatherMa, and OpenStreetMap for detailed 3D building models. The platform will provide a tool for analyzing and optimizing wind flow in urban environments. The project also aims to foster a collaborative environment by inviting contributions from urban planners, architects, and designers, enhancing the platform's functionality and adaptability. Ultimately, the project seeks to contribute to sustainable urban development by improving pedestrian comfort and reducing energy consumption through more effective wind management strategies.

Data

The project integrates three primary data sources to construct a simulation of urban wind flow:

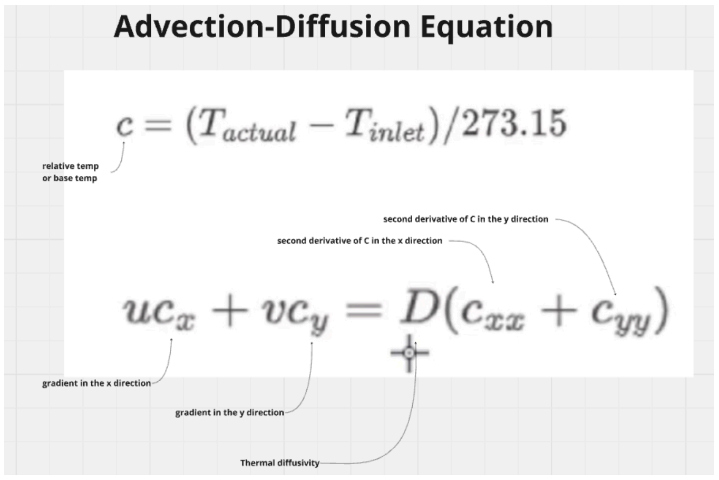
- NOAA Historical Data: Provides historical wind data accessed through APIs, forming the baseline for understanding long-term wind patterns and trends.
- OpenWeatherMap (OWM): Real-time wind condition data from OWM is dynamically fetched through specific API calls.
- OpenStreetMap (OSM): Detailed 3D building models using the Overpass API, enabling the extraction of building footprints data within a defined geographic radius. These geometries are processed and converted into a format compatible with the Nvidia Modulus Sym framework, crucial for accurate simulations of wind interactions with urban infrastructure.

The data collection process faced challenges, such as ensuring the synchronization of real-time data with the simulation engine and aligning the geometric data with the simulation grid. The transformation of OSM data into usable 3D geometries involved parsing and simplifying the data to fit the neural network architecture, maintaining computational efficiency while capturing essential features of the urban landscape.

Model

The collected data is integrated into a pipeline that feeds into the PINNs. This model combines the wind data, updates from OpenWeatherMap, and 3D building models from OSM to produce a simulation of urban wind flow.

- Real-Time: The real-time data from OpenWeatherMap enhances the model's responsiveness, allowing it to adapt to current conditions.
- 3D Geometry: The 3D building models from OSM serve as the physical environment within which the wind simulations are conducted. These models, processed and simplified as needed, provide the structural context necessary for accurate wind flow simulations.
- PINN Integration: It utilizes the combined data to simulate wind behavior in urban environments. The network is trained to adhere to the principles of fluid dynamics, specifically the Navier-Stokes and Advection-Diffusion equations. This ensures that the simulations are both physically accurate and responsive to real-time data inputs.



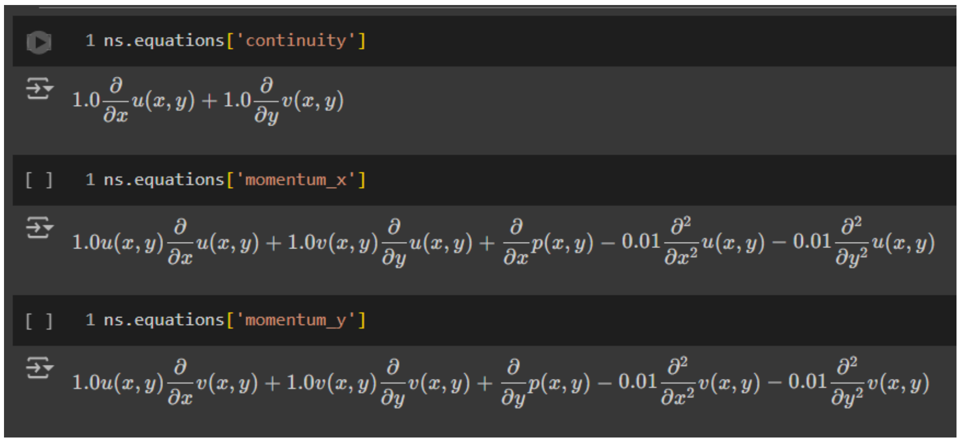
Advection-Diffusion Equation

$$c = (T_{actual} - T_{inlet})/273.15$$

$$uc_x + vc_y = D(c_{xx} + c_{yy})$$

relative temp
or base temp
second derivative of C in the y direction
second derivative of C in the x direction
gradient in the x direction
gradient in the y direction
Thermal diffusivity

Navier Stokes



```
ns.equations['continuity']
```

$$1.0\frac{\partial}{\partial x}u(x,y) + 1.0\frac{\partial}{\partial y}v(x,y)$$

```
ns.equations['momentum_x']
```

$$1.0u(x,y)\frac{\partial}{\partial x}u(x,y) + 1.0v(x,y)\frac{\partial}{\partial y}u(x,y) + \frac{\partial}{\partial x}p(x,y) - 0.01\frac{\partial^2}{\partial x^2}u(x,y) - 0.01\frac{\partial^2}{\partial y^2}u(x,y)$$

```
ns.equations['momentum_y']
```

$$1.0u(x,y)\frac{\partial}{\partial x}v(x,y) + 1.0v(x,y)\frac{\partial}{\partial y}v(x,y) + \frac{\partial}{\partial y}p(x,y) - 0.01\frac{\partial^2}{\partial x^2}v(x,y) - 0.01\frac{\partial^2}{\partial y^2}v(x,y)$$

- Boundary and Initial Conditions: The PINNs are configured with boundary conditions derived from the data. These conditions govern the entry, exit, and flow of wind through the urban landscape, ensuring that the simulations are grounded in real-world physics.
- Domain Constraints and Monitoring: The model incorporates constraints that maintain physical plausibility, such as ensuring continuity and momentum conservation. Monitors are placed at critical points within the simulation to track key metrics, providing real-time feedback on the accuracy and performance of the model.
- Output and Visualization: The final output includes detailed visualizations of wind speed, direction, and pressure across the urban landscape. These visualizations are essential for assessing pedestrian comfort and optimizing urban design.

Equations and Model Development

Navier-Stokes Equations:

The Navier-Stokes equations are governing the motion of fluid substances like air and water. In the context of urban wind flow modeling, these equations simulate how air moves around buildings and through city streets. The equations describe how the velocity field of the fluid evolves over time, considering factors like viscosity, pressure, and external forces.

The Navier-Stokes equations are embedded within the PINNs, ensuring that the predictions generated by the neural network adhere to the fundamental laws of fluid dynamics. This allows the model to simulate wind interactions with urban geometry, such as the acceleration of wind around corners or the formation of vortices in the wake of tall buildings.

Integration of Real-Time Data:

The primary benefit of using real-time data is that it enables the simulation to reflect current weather conditions.

The model is designed to fetch and incorporate real-time wind conditions from OpenWeatherMap. This data is then processed and integrated into the simulation pipeline, ensuring that the wind flow model is continuously updated with the latest environmental conditions.

Methodology

The methodology follows a structured approach, progressing through Requirements, Planning, Design, Development, Testing, and Deployment phases. This process ensures that each aspect of the project is systematically addressed, from initial concept to final implementation.

Paragraph 2: Detailed Phases

1. Requirements: The project began with defining the core requirements for developing a web application capable of simulating wind flow using Physics-Informed Neural Networks (PINNs). This phase involved identifying key datasets from Ladybug Tools, NOAA, and OpenWeatherMap. User inputs for building geometry, real-time data integration, and visualization features were also outlined.
2. Planning: The phase involved setting up the development environment, selecting Vue.js for the frontend, Flask/FastAPI for the backend, and pyTorch and tensorFlow for initial PINN testing. The project's folder structure was established, and the repository was set up to support collaborative development and version control.
3. Design: During the phase, the system architecture was planned, including the integration of the frontend, backend, and the PINN model. This phase focused on creating the layout and components for the Vue.js application, defining API endpoints for data handling, and detailing the architecture required for the PINN model.
4. Development: The development phase involved implementing the Vue.js application to handle user input and provide visualizations of the wind simulation. The backend API was developed using Flask/FastAPI, and the PINN model was integrated and trained with the collected data. The model was deployed as a REST API, making it accessible for real-time simulations.
5. Testing: Comprehensive testing was conducted to ensure the reliability and accuracy of the frontend, backend, and PINN components. Unit tests were performed to verify the functionality of each module, and integration tests ensured that the components worked seamlessly together.
6. Deployment: The final phase involved deploying the backend on a cloud platform and hosting the frontend on the web. This ensured that the platform was accessible to users, providing them with the tools to simulate and visualize urban wind flow in real-time.

PINN Model Development

The model was developed using PyTorch for its flexibility in prototyping and NVIDIA Modulus for its advanced capabilities in integrating physical laws with neural networks. These tools were chosen to ensure the model could accurately simulate fluid dynamics in urban environments. The development process involved iteratively testing and refining the model. The Navier-Stokes and Advection-Diffusion equations were implemented within the neural network to simulate urban wind flow, with real-time data integration to ensure the model's responsiveness to changing environmental conditions. Challenges such as computational efficiency and data synchronization were addressed to achieve a reliable simulation tool.

The frontend of the app was built using Vue.js2, a progressive JavaScript framework known for its flexible, component-based architecture. Key integrations include Three.js for 3D rendering and Geolib for geographic calculations. Three.js facilitates the visualization of complex wind flow simulations, while Geolib supports the necessary geographic computations that underpin accurate wind modeling.

Example Code:

```
import * as THREE from 'three';

import { OrbitControls } from
'three/examples/jsm/controls/OrbitControls';

import { getDistance, getRhumbLineBearing } from 'geolib';
```

These libraries enable the rendering of urban landscapes and the precise calculation of distances and bearings, critical for simulating wind flow in 3D environments.

The user interface is structured around component-based design, which allows for modular and maintainable code. For example, the ‘`AppSidebar`’ component handles user input and displays real-time weather data, which is fetched via the OpenWeatherMap API. This data informs the wind flow simulations, allowing users to visualize how real-time weather conditions affect urban environments.

Example Code:

```
<template>

  <div class="sidebar">

    <h2>Controls</h2>

    <form @submit.prevent="updateCoordinates">

      <!-- User input fields -->

    </form>

    <div class="weather-data" v-if="typeof weather.main !== 'undefined'">

      <!-- Weather information display -->

    </div>

  </div>
```

```
</template>

<script>

export default {

  name: 'AppSidebar',

  props: {

    weather: Object,

  },

  methods: {

    updateCoordinates() {

      this.$emit('update-coordinates', { latitude: this.latitude,
longitude: this.longitude, radius: this.radius });

    },

    windDirection(deg) {

      const directions = ["N", "NNE", "NE", "ENE", "E", "ESE", "SE",
"SSE", "S", "SSW", "SW", "WSW", "W", "WNW", "NW", "NNW"];

      const index = Math.round(deg / 22.5) % 16;

      return directions[index];

    }

  }

};

</script>
```

This UI component structure allows users to interact with the simulation by adjusting geographical coordinates and radius, and seeing immediate visual feedback on how these adjustments affect the wind flow simulation.

The system integration combined the frontend, backend, and PINN model into a single cohesive platform. Vue.js2 manages the user interface and rendering, while Flask/FastAPI handles data processing and API endpoints on the backend. The PINN model, developed using NVIDIA Modulus, processes real-time weather data fetched from OpenWeatherMap and simulates wind flow through urban environments.

Example Code:

```
methods: {

    async fetchWeatherData(latitude, longitude) {
      try {

        const response = await
fetch(`https://api.openweathermap.org/data/2.5/weather?lat=${latitude}&lo
n=${longitude}&units=metric&appid=4470f6319555f06d113186032c495792`);

        const data = await response.json();

        this.weather = data;

      } catch (error) {

        console.error('Failed to fetch weather data:', error);

      }

    },

}
```

This integration ensures the platform operates seamlessly, allowing for accurate and timely simulations based on user inputs and real-time weather data.

The project successfully developed a web-based platform using Vue.js, Three.js, and Geolib, integrated with a backend API powered by Flask/FastAPI, to simulate urban wind flow using Physics-Informed Neural Networks (PINNs). The platform effectively processes and visualizes real-time data, providing a responsive and accurate simulation environment for urban planners and researchers.

Future work could involve expanding the platform's capabilities, such as integrating additional environmental factors like temperature and humidity or enhancing the model's complexity with more advanced simulation models. Scaling the platform for broader use, including deploying it on larger cloud infrastructures, could improve accessibility and performance. Additionally, the platform has potential applications beyond urban wind flow modeling to include urban planning, disaster management, and renewable energy assessments.

References:

1. NVIDIA Modulus. (n.d.). *Physics-Informed Machine Learning with NVIDIA Modulus*. Retrieved from docs.nvidia.com
2. *Scientific Machine Learning Through Physics-Informed Neural Networks*. (2022). Journal of Computational Physics, 466, 110924. https://doi.org/10.1007/s10915-022-01939-z
3. *GPT-PINN: Generative Pre-Trained Physics-Informed Neural Networks*. (n.d.). Retrieved from arxiv.org
4. *Using Hybrid Physics-Informed Neural Networks for Complex Engineering Simulations*. (n.d.). Retrieved from arxiv.org
5. *Architectural Strategies for PINN Optimization*. (n.d.). Retrieved from docs.nvidia.com

6. *Scientific Machine Learning through PINNs*. (n.d.). Retrieved from physicsbaseddeeplearning.org
7. Zappitelli, R., Najjar, S. K., Bovo, R., Maqbool, T., & Kaeuffer, A. (2023). *Machine Learning for Pedestrian-Level Wind Comfort Analysis*. Buildings, 14(6), 1845. https://doi.org/10.3390/buildings14061845
8. *Deep learning to replace, improve, or aid CFD analysis in built environment applications: A review*. (2021). Building and Environment, 207, 108459. https://doi.org/10.1016/j.buildenv.2021.108459
9. *Evaluating the pedestrian level of service for varying trip purposes using machine learning algorithms*. (2024). Scientific Reports, 14, 53403. https://doi.org/10.1038/s41598-024-53403-7
10. NVIDIA Modulus. (n.d.). *NVIDIA Modulus - Advanced Topics*. Retrieved from courses.machinedecision.com
11. *Can Taichi play a role in CFD?* | *Taichi Docs*. (n.d.). Retrieved from docs.taichi-lang.org
12. *Table of Contents* | *NVIDIA Docs*. (n.d.). Retrieved from docs.nvidia.com
13. *Regression using LightGBM - GeeksforGeeks*. (n.d.). Retrieved from geeksforgeeks.org
14. *Random Forest Regression in Python - GeeksforGeeks*. (n.d.). Retrieved from geeksforgeeks.org
15. *Computer Fluid Dynamics Python at DuckDuckGo*. (n.d.). Retrieved from duckduckgo.com
16. *Deep Learning*. (n.d.). Retrieved from docs.nvidia.com
17. *Physics-Informed Machine Learning through Modulus* | *Physical Loss Code*. (n.d.). Retrieved from physicsbaseddeeplearning.org
18. *CFD Analysis in Built Environment Applications Using Deep Learning: A Review*. (2024). Journal of Building and Environment, 207, 108459. https://doi.org/10.1016/j.buildenv.2021.108459
19. *Evaluating the Pedestrian Level of Service for Varying Trip Purposes Using Machine Learning Algorithms*. (2024). Nature Communications, 14, 53403. https://doi.org/10.1038/s41598-024-53403-7
20. *Computer Fluid Dynamics Python at DuckDuckGo*. (n.d.). Retrieved from duckduckgo.com
21. *Deep Learning to Replace, Improve, or Aid CFD Analysis in Built Environment Applications: A Review*. (2024). Building and Environment, 207, 108459. https://doi.org/10.1016/j.buildenv.2021.108459