ISSN 2616-5767 Vol.6, Issue 11: 120-126, DOI: 10.25236/AJETS.2023.061118

Application Technology of Atmospheric Dispersion Models Based on FastAPI+Vue

Junqiao Chen[1]

[1]*College of Nuclear Technology and Automation Engineering, Chengdu University of Technology, Dongsanlu, Erxianqiao, Chengdu, Sichuan, 610059, China*

Abstract: *The application of atmospheric dispersion models for predicting and analyzing environmental pollution incidents has become a major trend in air pollution forecasting. This includes Gaussian models, Lagrangian models, CFD, and others. Atmospheric dispersion models are typically deployed on PC desktop platforms such as Windows and MAC on personal/public computers. Their input/output data usually rely on local file inputs, which is not conducive to the convenient use of the models.This article combines new framework technologies in the computer field with model theory. Specifically, it uses the Python-based FastAPI backend framework to deploy algorithm script modules and Vue to design the input and output UI interfaces, achieving the implementation of the algorithm on a web client platform. Additionally, by leveraging GIS technology, the output data results of the atmospheric model are structurally optimized and rendered. The rendered results have reference value for the development of emergency response plans by relevant departments in the event of accidents.*

Keywords: *Atmospheric dispersion model, Lagrangian, WebGIS, FastAPI.Vue, Python*

1. Introduction

In the past two decades, atmospheric dispersion models have been applied in many atmospheric pollution prediction systems. The pollutants in the atmosphere undergo four main processes, including transport, dispersion, migration, and transformation. A complete prediction process includes the chemical transformation of pollutant source terms, pollutant transport and dispersion modules, and pollutant dry and wet deposition, among others. Based on dispersion principles, key factors of concern, and applicable scenarios, various radioactive nuclide dispersion analysis systems have been developed to meet different environmental scenarios and usage requirements. Five types of models have been established, including Gaussian models, Lagrangian models, Eulerian models, CFD models, and Lagrangian-Eulerian nested models.Among these models, the "CALPUFF" model is one that was developed and maintained by Sigma Corporation with support from the U.S[1].Environmental Protection Agency. The CALPUFF model is based on the Lagrangian model principle and is one of the recommended atmospheric dispersion simulation models in China.

FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.7+ based on standard Python type hints[2].Using Python FastAPI, which boasts a robust ecosystem of dependencies for algorithmic tasks, we transcribed the CALPUFF model into Python code. This code serves as the backend for an application system related to atmospheric dispersion modeling. Additionally, we employed Vue framework technologies to achieve dynamic visualization of dispersion results.

2. Lagrangian Plume Model

2.1 Overview

The Lagrangian plume model is employed to describe the distribution of isotopes, using a coordinate system that moves with the plume centroid to depict the concentration and its variations (such as RODOS system)[3][4]. Its most significant distinction from the Gaussian method lies in its capability to continuously track the movement and changes in dispersing substances. It overcomes the limitations of the Gaussian plume model, providing a more realistic simulation of the spatiotemporal distribution of airborne radioactive materials in complex conditions.

The Lagrangian plume model, in particular, is a relatively simple and flexible atmospheric dispersion

ISSN 2616-5767 Vol.6, Issue 11: 120-126, DOI: 10.25236/AJETS.2023.061118

model for radioactive nuclides. It conceptualizes released pollutants as discrete plumes, simulating the transport process for each plume center. Each plume remains stationary for a specific time interval, and the concentration is calculated based on the plume's fixed position during that moment. Subsequently, the plume continues to move in the next calculation time step, with its size and intensity evolving until the next sampling time when it stabilizes again. Within the fundamental time step, the concentration at a particular point is the sum of the average concentrations of all surrounding plumes during that sampling time.

The Lagrangian plume model equation is as follows:

$$C_i(x,y,z)=\frac{Q(i)}{(2\pi)^{3/2}\sigma_x\sigma_y\sigma_z}\cdot\exp(-\frac{1}{2}((\frac{y-y_c(i)}{\sigma_y})+(\frac{x-x_c(i)}{\sigma_x})^2))\cdot(\exp(-\frac{z-z_c(i))^2}{2\sigma^2{}_z})+\exp(-\frac{z+z_c(i))^2}{2\sigma^2{}_z})) \quad (1)$$

The difference from the Gaussian plume model lies in the critical aspect of constructing the Lagrangian plume model, which involves the development of a regional wind field with spatiotemporal characteristics. Enhancing the model's predictive accuracy relies on incorporating more meteorological input data. This is also the reason why the Lagrangian plume model can achieve medium to large-scale predictions. Internationally, there are well-established algorithms for Lagrangian plume models, such as the CALPUFF model recommended by the U.S. Environmental Protection Agency and the RIMPUFF model used in the European RODOS Nuclear Accident Emergency Decision Support System. These two models produced predictions for actual atmospheric dispersion experiments at the Kincaid Power Plant in the United States that were within the same order of magnitude and showed minimal differences. By comparing and validating against these two models, we can assess the accuracy of the plume model used in this module. The schematic diagram illustrating the principles of the Lagrangian plume model is shown in Figure 1.

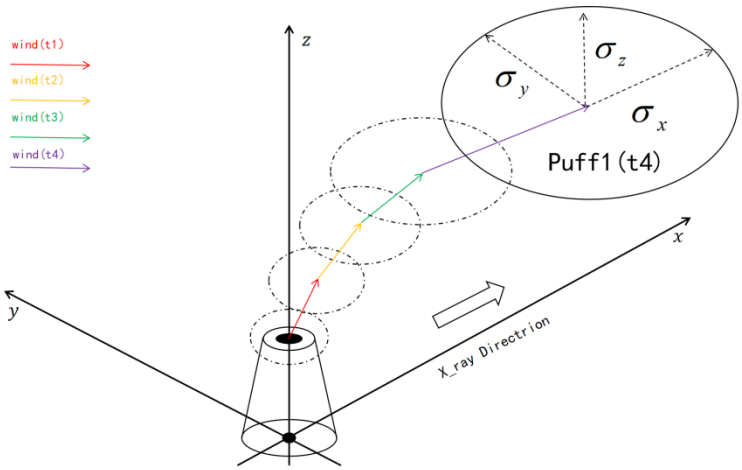

Figure 1: Schematic of the Lagrangian Plume Model Principles

In predictive models, dispersion parameters are positioned in the denominator, and they have a significant impact on algorithm accuracy. Therefore, the crucial step is how to select dispersion parameters by considering the topography, terrain, and meteorological characteristics of different key areas. Atmospheric dispersion parameters characterize the ability of pollutants to disperse and the range of their dispersion in a random turbulent field. They are represented in three component directions as σx, σy, and σz, and they exhibit spatiotemporal variations, particularly with varying values along the downwind distance x from the source. To obtain atmospheric dispersion parameters, it is essential to determine atmospheric stability. The internationally recognized method for classifying atmospheric stability is the Pasquill classification system, which has been revised domestically and referred to as the Revised Pasquill Classification System [5]. It categorizes atmospheric stability into six levels: Strongly Unstable, Unstable, Weakly Unstable, Neutral, Slightly Stable, and Stable, denoted as A, B, C, D, E, and F, respectively. When determining the stability level, the first step involves calculating the solar zenith angle by using equation as followings:

$$h_0=\arcsin[\sin\varphi\sin\delta+\cos\varphi\cos\delta\cos(15t+\lambda-300)] \quad (2)$$

φ and λ represent the geographical latitude and longitude, θ is the solar zenith angle, which can be retrieved based on the month and time, and t is the observation time.

ISSN 2616-5767 Vol.6, Issue 11: 120-126, DOI: 10.25236/AJETS.2023.061118

Finally, the dispersion parameters are determined by the Pasquill classification method, and you can obtain the Pasquill dispersion parameters by referring to the P-G empirical curve as shown in Figure 2.

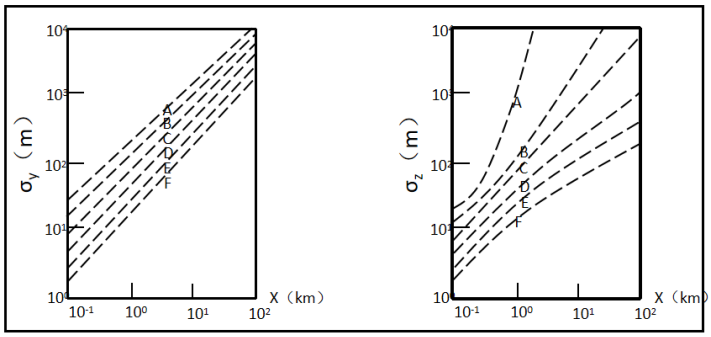

Figure 2: P-G Curve

2.2 MATLAB Simulation

When conducting research on numerical simulation models, we often use MATLAB to validate the effectiveness of the model. Writing MATLAB code allows us to clearly define the input and output data formats of the model, providing a bridge for implementing the model in other computer programming languages such as C++ and Python. MATLAB also offers GUI plugins for users, theoretically allowing us to present our final results using MATLAB. However, this approach clearly goes against the original intention of utilizing web platform frameworks, which we will discuss later in this paper regarding the advantages of web platforms over MATLAB. Nevertheless, we leverage MATLAB's powerful numerical simulation capabilities for the initial design of our model algorithm, ensuring that the final results meet our expectations.

We use fictitious atmospheric pollution events to simulate the impact of accidents. In a 2D plane, we assume the center point as the location of the accident and establish a Cartesian coordinate system. The source continuously releases pollutants into the air, with the wind direction at the source assumed to be southeast at 135 degrees. The results obtained are shown in Figure 3. We calculate the maximum concentration of pollutants accumulated in the ground-level air after a continuous release of pollutants for a certain period.

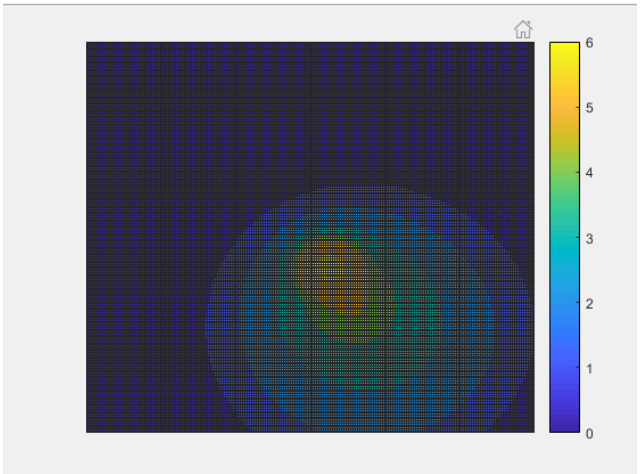

Figure 3: Simulation by MATLAB

During actual accidents, we can compare real data with simulated data. Additionally, particle dispersion experiments can be conducted in the real operational scenarios of the model. Although this experiment entails higher costs, it provides relatively accurate model parameters, especially dispersion parameters, for fixed locations. The dispersion parameters vary for coordinates at different distances from the source point, and calibrating these parameters can lead to more reliable results and increased confidence in the model.

ISSN 2616-5767 Vol.6, Issue 11: 120-126, DOI: 10.25236/AJETS.2023.061118

3. FastAPI

FastAPI is a modern, high-performance Python web framework used for building web applications with ease and speed. It's designed to be fast, simple to use, and provide automatic interactive documentation, making it a popular choice for building APIs. FastAPI leverages Python type hints and automatic data validation to simplify the development process and ensure robust, predictable APIs. It also offers asynchronous support, making it suitable for handling concurrent requests efficiently. FastAPI's automatic documentation generation, based on OpenAPI and JSON Schema, helps developers understand and interact with APIs more easily. Overall, FastAPI is an excellent choice for developing fast, reliable, and well-documented web applications and APIs in Python.

3.1 Model flow

Based on the algorithm principles of the Lagrangian plume model, we can design a program flowchart for the radioactive substance atmospheric dispersion model, as shown in Figure 4.

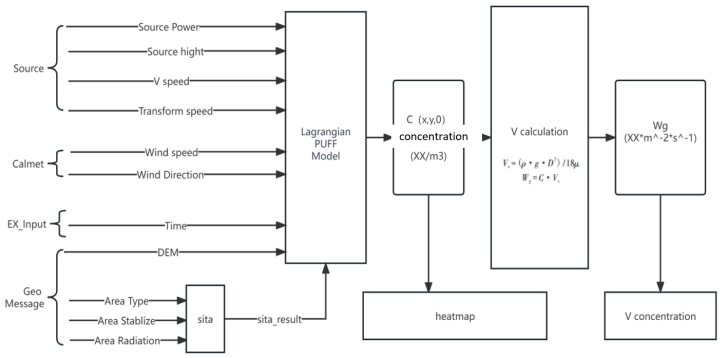

Figure 4: Lagrangian Model flow

The first step involves source term input, where source term parameters include the composition of the source term, source term height, source term intensity, release time, and the source term coordinates required for integration into the WEBGIS system. We achieve this by establishing an "Atmospheric Dispersion Incident Data Table" in the system, allowing us to predefine and manage source term parameters in a unified format.

In the second step, geographical and meteorological data are accessed and preprocessed. Initially, the algorithm's accuracy and scope are determined based on the release time input in the first step. Using this accuracy and scope, we generate a grid of points to hold the algorithm results and retrieve geographical and meteorological information within the temporal and spatial scales from the database. Subsequently, we calculate the dispersion parameters for each plume through dispersion parameter algorithms. The goal of this step is to obtain the central coordinates of multiple radioactive substance plumes. At each plume's central coordinate point, various data layers, including meteorological data, geographical information, and dispersion parameters, are derived through the aforementioned preprocessing. These data layers ultimately serve as input for the core algorithm of the Lagrangian plume model.

In the third step, the algorithm undergoes iterative calculations. According to the algorithm model principles, we can calculate the concentration impact of any central plume on any point within the algorithm model's scope. Summing the contributions of all plumes results in the concentration prediction for that point. The location information of this point, along with surface air radioactive isotope concentration values, will be used for heat map rendering in WEBGIS and will also serve as metadata for accident risk prediction and consequence assessment functions.

3.2 Model Interface

Interface design is a standardized step in the program implementation of algorithm models. We can separate the algorithm's input, output, and algorithm core design. By maintaining the interface and the core, we achieve the maintenance of the algorithm. Additionally, the interface documentation serves as the fundamental basis for database design. Based on the data required for the three algorithm steps mentioned above, we can design the database data tables. Due to the numerous data tables involved in this algorithm, we will use meteorological data and its interface as an example.

ISSN 2616-5767 Vol.6, Issue 11: 120-126, DOI: 10.25236/AJETS.2023.061118

The meteorological interface includes data required by the algorithm such as wind speed, wind direction, precipitation, temperature, cloud cover, etc. The interface is as shown in Table 1.

Table 1: Climate Model Interface.

Real Name	Data Type	Interface Name	Description
Avg Wind Speed in 5min	(float)	getWind_S	Data from the nearest station.
Avg Wind Direction in 5min	(int)	getWind_D	Data from the nearest station.
Precipitation	(float)	getRain	Daily precipitation.
Temperature	(int)	getTemp	Unit :℃
cloud cover	(int)	getCloud	Using type of “x%”

To ensure the smooth operation of the project in the background, we have designed interfaces for every parameter in the database. For example, parameters like source concentration and source height in the source term parameters, as well as DEM elevation in the geographical information parameters, latitude and longitude coordinates, and so on.The objective is to decouple parameters from the algorithm. In other words, the algorithm flow depicted in Figure 3 runs within the Python script's main function, but all intermediate and input parameters used by the algorithm are accessed via interfaces (even for fixed parameters, which are separately defined in static files). This approach allows data to be modified flexibly without altering the model's algorithm files. Such a setup helps avoid many manual errors and is also highly convenient for our ongoing maintenance of both data and algorithms, reducing the frequency of bug occurrences.

3.3 Database

A database is a repository organized, stored, and managed according to a data structure.Each database has one or more distinct APIs for creating, accessing, managing, searching, and replicating the stored data. Data can also be stored in files, but reading and writing data in files is relatively slow.

Therefore, we now use Relational Database Management Systems (RDBMS) to store and manage large volumes of data. A relational database is built on the foundation of the relational model and employs mathematical concepts and methods such as set algebra to handle data in the database.

The characteristics of an RDBMS (Relational Database Management System) are as follows: Data is presented in the form of tables. Each row represents various record names.Each column corresponds to the data domain for the record name. Many rows and columns form a table.Several tables make up a database.

MySQL is currently one of the widely used relational databases. For the atmospheric dispersion model, it is crucial to constantly update the meteorological data table to ensure that the data within the model remains self-updated. After establishing the local database, it is necessary to develop a program service for data validation and updating. This service should be configured to update the data at regular intervals, such as every hour (or as per the specified time precision, like 5 minutes). This is a key aspect that sets the atmospheric dispersion model system apart from conventional systems.

4. Vue (GIS)

In order to enable users to conveniently select input parameters, utilize the algorithm model, and retrieve algorithm results, all while displaying and interacting with this information within a GIS system, we need to build a client-side application software. The web frontend can run on various browsers, including those on PC and mobile devices. Frontend technology is generally divided into frontend design and frontend development. Frontend design is typically related to the visual design of the website, whereas frontend development involves the implementation of the website's frontend code, including fundamental components such as HTML, CSS, and JavaScript.

During frontend development, when structuring the webpage, HTML defines the elements, CSS positions and styles these elements, and JavaScript is used to implement corresponding effects and interactivity.

We designed a Geographic Information System (GIS) based on the Vue framework. By utilizing the official API provided by Tianditu (a map service provider), we accessed map tiles containing basic geographic information from the map server (similar to how Google Maps operates). The map initially supports fundamental map operations. By incorporating the Tianditu API into the HTML file, we can display vector maps, regional imagery, place name labels, and other geographic and hydrological

ISSN 2616-5767 Vol.6, Issue 11: 120-126, DOI: 10.25236/AJETS.2023.061118

information on the webpage. Additionally, the GIS map offers features such as full-screen display, zoom in, zoom out, pan, mouse zoom, keyboard controls, previous view, next view, location tracking, map saving, map switching, map layer control, overview map (minimap), map scale, and more.

Within the expanded results of the atmospheric dispersion model, the use of map APIs supports the ability to click on map points to obtain prediction results for specific locations.

The GIS system displays the image base map in the form of a heatmap. Whether it's the atmosphere or rivers, the points on the heatmap include three pieces of information: longitude, latitude, and isotope concentration, all encapsulated in JSON format. We incorporate the Heat-Map open-source heatmap API into the HTML file, passing JSON format data containing the aforementioned three pieces of information. After rendering with Heat-Map, a heatmap is generated, presenting the predicted results of the radioactive pollution dispersion model to the users.

The heatmap overlays as a layer on top of the base map. Users can obtain information about the current point's location coordinates, locality, prediction time, isotope deposition concentration, environmental surface gamma radiation air absorption dose rate, and other details by clicking on points within the heatmap. This information is displayed in the risk warning window on the right side of the main interface.

5. Conclusions

This paper is based on technologies such as FastAPI, Vue, and databases and explores the Lagrangian plume model algorithm. The algorithm is simulated and implemented in MATLAB, and then rewritten as Python code. A backend service for atmospheric dispersion modeling and a frontend for displaying results have been developed. This lays the foundation for the application of the atmospheric dispersion model on new devices and platforms. The system architecture combines real-time data, ease of use, and upgradeability. It can be used to guide consequence assessments of accidents by predicting surface air pollutant concentration curves, enabling short, medium, and long-term assessments of regional pollution situations.

In practical applications, the system also needs to consider the following aspects:

1) Consider non-uniform release sources and sources involving complex chemical reactions.

2) Set sensitive points for facilities and cities of particular interest to output concentration change curves over a specific period.

3) Adjust the predicted curves based on locally measured data and provide feedback to calculate dispersion parameters.

4) Consider the dispersion and transport of radioactive substances from finite-duration release sources after the release has ended to improve the accuracy of medium and long-term predictions.

Furthermore, the atmospheric dispersion model script developed in this paper uses the Lagrangian plume model, which is suitable for different complex terrain and land use conditions (same as CFD[6] but much easier). It can predict surface concentration of atmospheric radioactive substances using real-time meteorological data. The predictive results can be utilized for various responses to atmospheric pollution accidents and consequence assessments, making it valuable for the development of emergency response platforms.

References

[1] Lu Yanyan. A numerical study of harmful gas dispersion based on CALPUFF model [J]. E3S Web of Conferences, 2020, Vol. 191: 3007

[2] Priya Bansal, Abdelkader Ouda. Study on Integration of FastAPI and Machine Learning for Continuous Authentication of Behavioral Biometrics [A]. 2022 International Symposium on Networks, Computers and Communications (ISNCC) [C], 2022

[3] Zhong S, Zhou L, Wang Z. Software for Environmental Impact Assessment of Air Pollution Dispersion Based on ArcGIS [C]//International Conference on Environmental Science and Information Application Technology. 2013.

[4] O. Pylypenko, R. Bezhenar, S. Kivva, P. Kopka, S. Potempski, H. Wojciechowicz, M. Zheleznyak. Application of the hydrological model chain of the RODOS decision support system for nuclear

ISSN 2616-5767 Vol.6, Issue 11: 120-126, DOI: 10.25236/AJETS.2023.061118

emergencies to the analysis of possible consequences of severe accident [J]. Annals of Nuclear Energy, 2023, Vol. 188: 109823
[5] Saunier O, Mathieu A, Didier D, et al. Using GAMMA dose rate monitoring with inverse modelling techniques to estimate the atmospheric release of a nuclear power plant accident: Application to the Fukushima case [J]. 2012.
[6] Vervacken, L., Camps, J., & Meyers, J. (2015). Dynamic dose assessment by Large Eddy Simulation of the near-range atmospheric dispersion. Journal of Radiological Protection, 35(1), 165-178.