# 3D Mountain Simulation with Flood Paths and Accumulated Water Volume

Matin Monshizadeh

August 2, 2024

**Abstract**

This project involves simulating a 3D terrain with flood paths using Python libraries such as NumPy, Matplotlib, and SciPy. By generating random 2D points and applying Delaunay triangulation, I modeled mountains using various terrain functions, including Gaussian peaks and valleys. Flood simulations were conducted from grid centers, tracking water flow across the terrain based on gradients, with boundary conditions enforced. The results demonstrate how water accumulates in lower elevations, such as valleys and flat areas, which is visualized through 3D terrain plots, flood paths, and a heatmap of accumulated water volume.

## 1 Introduction

### 1.1 Background

In the field of computer engineering, terrain modeling and simulation play a crucial role in understanding various environmental processes and phenomena. One significant application of terrain modeling is flood simulation, which helps predict and manage the impact of floods on different landscapes. Flood simulations are essential for urban planning, disaster management, and environmental conservation. They involve creating detailed models of terrain to simulate how water spreads across different surfaces, considering factors such as elevation, water volume, and terrain features.

### 1.2 Objective

The primary objective of this project is to develop a comprehensive simulation of flood propagation over a 3D terrain model. The project aims to achieve the following goals:

#### 1.2.1 Generate Realistic Terrain Models

Create 3D terrain models with various elevation profiles, such as Gaussian mountains, volcanic crater, and ridges, using synthetic data.

#### 1.2.2 Simulate Flood Paths

Implement a flood simulation algorithm that traces the paths of water flow from various starting points and determines the stopping points based on terrain features and gradients.

#### 1.2.3 Visualize Terrain and Flood Dynamics

Develop visualizations to represent the terrain, flood paths, and accumulated water volume, providing an intuitive understanding of how floods interact with different terrain features.

#### 1.2.4 Evaluate Water Accumulation

Analyze the total water volume accumulated in different grid cells and ensure that the simulation accurately reflects the expected rainfall distribution.

### 1.3 Motivation

Floods are highly destructive, causing significant damage to infrastructure, property, and lives. Predicting water flow and identifying flood-prone areas are vital for disaster management, urban planning, and environmental protection. This project simulates flood paths on 3D terrain to develop tools for predicting flood risks and improving prevention strategies. Such modeling is especially useful in areas with complex geography, where water flow is difficult to predict. The project's results can be applied to disaster preparedness, water resource management, and sustainable development, highlighting the importance of flood simulation in modern society.

## 2 Methodology

### 2.1 3D Reconstruction (Terrain Generation)

#### 2.1.1 Generating 2D Points

To begin with the terrain generation, we first created a set of random points in a 2D space. Specifically, we generated 400 random points uniformly distributed within the unit square $[\,0\,,\,1\,] \times [\,0\,,\,1\,]$. These points serve as the vertices for the terrain's underlying structure.

#### 2.1.2 Transforming 2D Points into 3D Terrain

To elevate these 2D points into a 3D terrain, we assigned a third dimension, $z$, to each point. This dimension represents elevation, which we derived using various mathematical functions. These functions are designed to create different terrain features, such as mountains, hills, and ridges.

We explored several mathematical functions to generate diverse terrain profiles:

- **Gaussian Mountain:** A single peak in the center of the terrain.

$$z = e^{-5\times\left((x-0.5)^2+(y-0.5)^2\right)}$$

- **Simple Hill (Circular Cone):** A conical hill centered in the terrain.

$$z = 1 - \sqrt{(x-0.5)^2 + (y-0.5)^2}$$

- **Multiple Peaks:** Two Gaussian peaks to simulate a mountainous region with multiple high points.

$$z = e^{-5\left((x-0.25)^2+(y-0.25)^2\right)} + e^{-5\left((x-0.75)^2+(y-0.75)^2\right)}$$

- **Ridges (Sine Waves):** Terrain characterized by sinusoidal ridges.

$$z = \sin(5\pi x)\sin(5\pi y)$$

- **Volcanic Crater:** A terrain with a central depression resembling a volcanic crater.

$$z = e^{-5r} - e^{-20r}$$

- **Plateau with Cliffs:** A flat plateau with steep cliffs around the edges.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$z = \tanh\left(10\left(0.5 - \left((x-0.5)^2 + (y-0.5)^2\right)\right)\right)$$

- **Mountain with a Valley:** A terrain function that simulates a mountain with a valley.

$$z = e^{-5\left((x-0.5)^2+(y-0.5)^2\right)} - 0.5 \cdot e^{-5\left((x-0.25)^2+(y-0.75)^2\right)}$$

#### 2.1.3 Triangulation and Mesh

After defining the elevation for each point, we used **Delaunay Triangulation** to connect these points into a mesh. This mesh structure allows for more accurate calculations of gradients and flood simulation. We computed the gradient at each mesh triangle to determine the direction of steepest descent, which is crucial for simulating how water would flow across the terrain.

**Delaunay Triangulation**

Delaunay triangulation is a method of connecting a set of points in a way that no point lies inside the circumcircle of any triangle in the triangulation. This property maximizes the minimum angle of all the triangles, avoiding the creation of very narrow or "skinny" triangles, which leads to a more evenly distributed and aesthetically pleasing mesh.

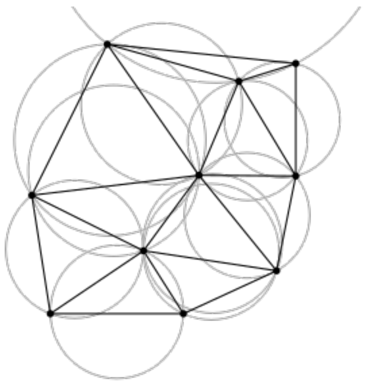


Figure 1: Delaunay Triangulation

In this project, the set of random 2D points generated earlier was triangulated using Delaunay triangulation. The triangulation process transforms these points into a connected mesh of non-overlapping triangles. Each triangle is formed by connecting three points such that the overall mesh covers the entire region defined by the points.

**Why Use Delaunay Triangulation?**

Delaunay triangulation is ideal for terrain generation and flood simulation because:

- **Even triangle shapes:** It avoids creating very thin triangles, which can distort the representation of the surface and lead to inaccuracies in gradient and flow calculations.
- **Robust for irregular point distributions:** It works well even when the points are randomly distributed, as in this project. This makes it flexible for modeling natural terrains.
- **Optimized for nearest neighbor calculations:** Since water flows along the path of steepest descent, Delaunay triangulation ensures that the connections between points reflect the local terrain structure accurately.

**How it Works**

Given a set of n points in 2D space, Delaunay triangulation forms triangles by following these steps:

- **Find the Circumcircle Condition:**. For each set of three points, it calculates the circumcircle (the circle that passes through all three points). A triangle is formed if no other points from the set lie within this circumcircle.
- **Maximize Minimum Angles:** The algorithm ensures that the smallest angles of all the triangles are as large as possible, which avoids sliver-like triangles.

In our project, Delaunay triangulation was performed using the Delaunay class from the scipy.spatial library. This method automatically constructs a triangular mesh from the given set of 2D points. The output is a set of triangles that define the relationships between the points. These triangles are then used for:

- **Visualizing the terrain:** The triangles are used to display the 3D terrain surface.
- **Calculating water flow:** The gradients of the terrain are calculated on these triangles, enabling the simulation of how water flows across the surface.

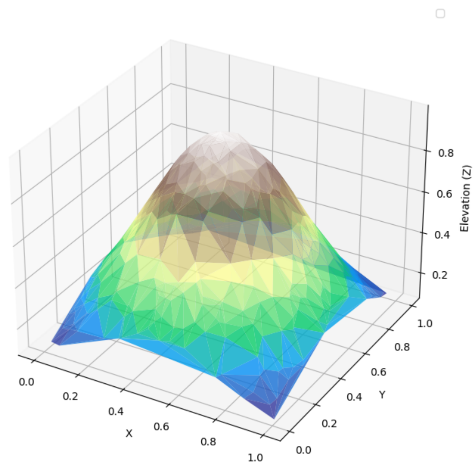


Figure 2: 3D Reconstruction of a Gaussian Mountain

## 2.2 Grid Overlay

### 2.2.1 Grid Definition

The grid overlay helps in systematically breaking down the terrain into discrete cells, allowing for precise simulation of flood paths and accurate visualization of accumulated water volumes. Each grid cell serves as a starting point for simulating flood paths and tracking water distribution.

### 2.2.2 Grid Creation

We utilized the np.meshgrid function to create a grid of x and y coordinates that covers the entire simulation area. This grid is defined with a cell size of 0.1 units.

The elevation for each grid point is computed using the same mountain function applied to the grid coordinates.

### 2.2.3 Flood Path Simulation

For each grid cell, the center is chosen as the starting point for the flood simulation.

The flood path is simulated from each starting point using the simulate_flood_path function. This function calculates the path by moving in the direction of the gradient while checking for boundary conditions.

### 2.2.4 Water Volume Accumulation

An initial water volume grid is created to keep track of the accumulated water in each cell. The rain volume per cell is set to 10 liters.

For each simulated flood path, the stopping point is determined, and the water volume is added to the corresponding grid cell.

### 2.2.5 Visualization

The grid lines are plotted on the 3D terrain to provide a reference for the flood paths. This is achieved by plotting grid lines and flood paths on the terrain using matplotlib.

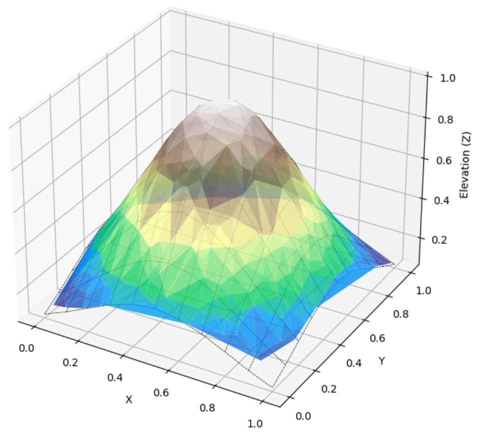


Figure 3: Grid Overlay of a Gaussian Mountain

## 2.3 Flood Simulation

In this project, the simulation of water flow paths over a terrain is driven by the **gradients** of the elevation at each point. The methodology is structured as follows:

### 2.3.1 Gradient Calculation

To simulate water flow over the terrain, the gradient of the elevation at each point must be determined. The gradient represents the slope of the terrain, and it is crucial for determining the direction of water flow.

For each triangular facet formed by the Delaunay triangulation, the elevation at the three vertices is known. A plane is fit to these points, and the gradient is calculated as the vector normal to this plane. This process involves solving the following system of equations to obtain the coefficients of the plane equation:

$$A \cdot p = z$$

Where $A$ is a matrix containing the x, y coordinates of the vertices and a column of ones, $p$ is the vector of plane coefficients, and $z$ is the vector of z-values (elevations) at the vertices. The first two components of $p$ represent the gradient in the x and y directions, respectively. These gradient vectors are used to simulate the water flow on the terrain.

### 2.3.2 Simulating Water Flow

Water flow is simulated by calculating a path starting from the center of each grid cell. At each point, the water flows downhill, following the direction of the negative gradient. The path is iteratively updated by moving the current point in the direction of the gradient for a small step size. The simulation stops if:

- The water reaches a flat area (where the gradient is near zero).
- The boundary of the simulation area is reached.

### 2.3.3 Boundary Handling

Boundary conditions are enforced to prevent the water flow from going outside the defined simulation area. If a calculated path attempts to leave the boundary, the simulation for that particular path is halted.

#### 2.3.4 Accumulation of Water Volume

To simulate rainfall and water accumulation, a uniform amount of rainfall is assigned to each grid cell. As the water flows along its calculated path, it accumulates in the final stopping point. This is tracked on a water volume grid, where the total water collected in each cell is visualized as a heatmap.

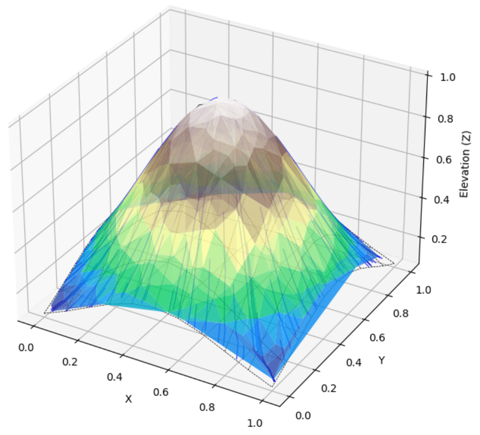


Figure 4: Flood Simulation on a Gaussian Mountain

## 2.4 Tools and Libraries

### 2.4.1 NumPy

In this project, NumPy was used for generating random 2D points, creating elevation values (z-values) for the terrain, and performing numerical operations on arrays such as calculating gradients and water volumes. The lstsq function from NumPy was employed for solving linear least squares problems to calculate gradients in the terrain.

### 2.4.2 Matplotlib

Matplotlib was used to visualize the 3D terrain and flood paths. imshow was used to display a heatmap of the accumulated water volumes across the grid cells

### 2.4.3 SciPy

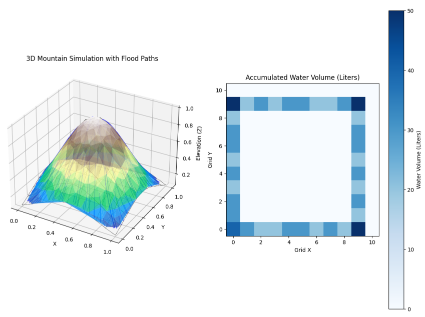
In this project, SciPy's Delaunay function was used to perform Delaunay triangulation of the 2D points. This triangulation was crucial for determining the mesh over which the flood simulation was performed

## 3 Results

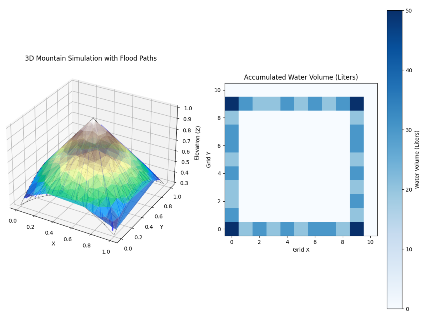
The figure demonstrates the results of simulating flood paths on a 3D mountainous terrain and visualizing the accumulated water volume in a grid representation.

- **Left: 3D Mountain Simulation with Flood Paths**
  The 3D plot depicts a randomly generated mountainous terrain. The flood paths (represented as colored lines) trace the water flow from the mountain's peak downwards, following the gradient of the terrain. Water generally flows from higher elevations toward lower areas, simulating how rainfall would naturally flow down the mountain.

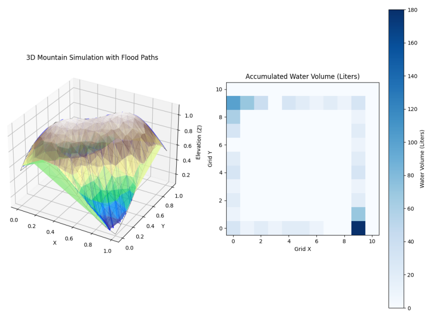
- **Right: Accumulated Water Volume Heatmap**
  The 10 × 10 grid illustrates the distribution of accumulated water volume (in liters) across the terrain. Each cell corresponds to a section of the terrain, and the intensity of the color represents the amount of water accumulated in that region. The water tends to accumulate mostly at the boundaries of the grid, indicating that water flows down from the central peak and pools around the edges. Darker shades of blue indicate areas with higher water volume, while lighter shades represent regions with less water accumulation.



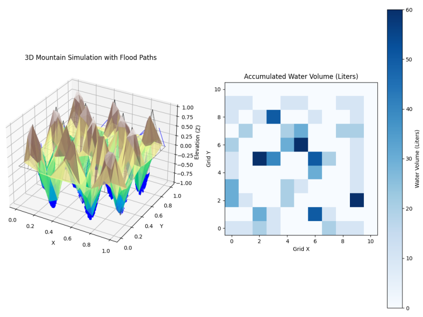
(a) Gaussian Mountain



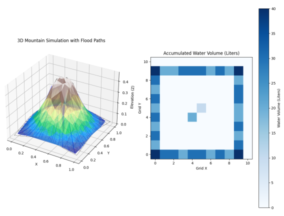
(b) Simple Hill (Circular Cone)



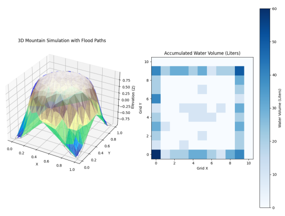
(c) Multiple Peaks



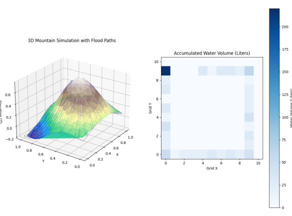
(d) Ridges (Sine Waves)



(e) Volcanic Crater



(f) Plateau with Cliffs



(g) Mountain with a Valley

Figure 5: Results of flood simulation and water volume visualization on 3D terrain.

## 4 Discussion

In this project, I successfully simulated flood paths and water volume accumulation on a 3D terrain generated using a variety of mathematical functions. The results showed that the terrain's geometry significantly influenced the flood behavior, with water naturally accumulating in lower elevations as expected. The grid-based approach and the use of Delaunay triangulation allowed for effective simulation of water flow, and the visualization techniques provided a clear representation of flood dynamics and water accumulation.

While the current approach achieved the project objectives, alternative methods could further enhance the realism and accuracy of both the terrain generation and flood simulation. One potential improvement is the use of **Perlin noise** for terrain generation. Unlike the mountain functions used in this project, Perlin noise would allow for more complex, naturally occurring terrains by introducing randomness and small-scale features that resemble real-world landscapes. This could result in more intricate flood paths and more realistic water flow behavior.

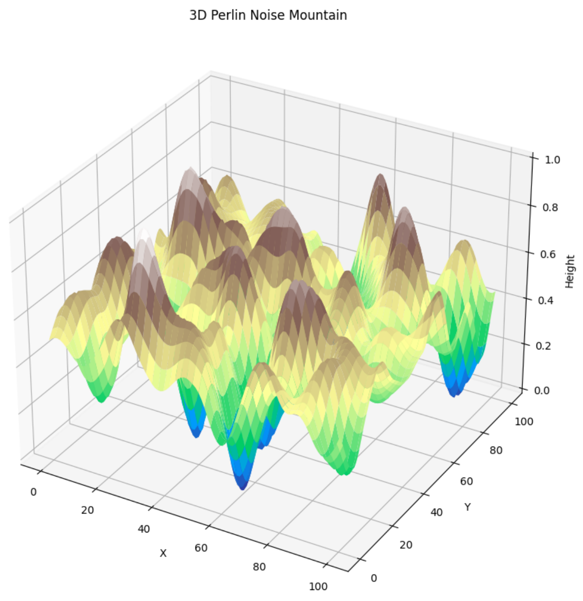


Figure 6: Terrain generated with Perlin noise

Additionally, for **3D reconstruction of terrain**, **Poisson surface reconstruction** offers an advanced method that could provide smoother and more accurate surfaces from scattered points. While the Delaunay triangulation used here was effective, Poisson surface reconstruction could be more appropriate for handling noisy data or real-world applications where terrain data is gathered from sensors or satellite images.

For better visualization, tools like **Open3D** or **OpenGL** could be employed. These libraries are designed for high-performance 3D visualization and would enable more interactive and detailed visual outputs. Additionally, software like **Blender** could be used to create more photorealistic renderings of the terrain and flood dynamics. These advanced visualization tools would be especially useful for larger or more complex terrains, where the current 2D and 3D plots might not capture the full detail and scope of the simulation.

Another promising direction for future work is the use of **3D reconstruction from multiple views**. This technique, which generates 3D models from 2D images taken from different perspectives, could provide a real-world basis for flood simulations by converting actual terrain into 3D models. By applying this method, simulations could be run on real geographical areas, making the results more applicable for real-life flood predictions and environmental modeling.

Despite the strengths of the current approach, there are limitations. The terrain models, while effective, are relatively simple and may not fully capture the complexity of natural landscapes. Additionally, the resolution of the grid used for water volume accumulation could be increased to improve precision, although this would come at the cost of greater computational requirements.

In future work, exploring the alternative methods mentioned above could significantly enhance the project. By implementing more complex terrain generation algorithms, adopting advanced reconstruction techniques, and utilizing more powerful visualization tools, the simulation can be made more realistic and applicable to a wider range of scenarios. Additionally, testing this simulation on real-world terrain data would further validate its effectiveness and open up possibilities for practical applications in flood risk assessment and environmental planning.

## 5 Conclusion

This project successfully achieved its objectives of generating realistic 3D terrain models, simulating flood paths, visualizing terrain and flood dynamics, and evaluating water accumulation. By using a combination of random point generation, Delaunay triangulation, and gradient-based water flow simulations, we were able to model diverse terrain types, including mountains, valleys, and ridges, as well as simulate the behavior of water flow over these terrains.

The flood path simulations accurately reflected the terrain's topography, with water flowing toward lower elevations as expected. The use of heatmaps to visualize water accumulation provided clear insights into how rainfall concentrated in certain areas, particularly at the base of slopes and in valleys. The grid overlay, coupled with boundary checks, helped ensure that the simulations remained within the defined terrain, while handling boundary conditions appropriately.

Despite the success of the overall approach, certain challenges were encountered, particularly in handling boundary conditions during flood path simulation and ensuring that water volume calculations aligned with the expected rainfall distribution. However, these were addressed through refinement of the flood path algorithm and grid-based water volume accumulation, leading to results that accurately captured water distribution across the terrain.

For future work, there is potential to enhance the realism of the terrain model by using more sophisticated methods such as Perlin noise or Poisson surface reconstruction. These approaches could produce more detailed and natural terrain features. Additionally, integrating advanced tools like Open3D, OpenGL, or Blender could significantly improve visualization, allowing for more complex 3D representations and animations of flood dynamics. Another direction could involve exploring 3D reconstruction from multiple views to simulate real-world terrains more accurately.

In conclusion, this project demonstrated a successful integration of terrain modeling, flood simulation, and visualization techniques. It provides a foundation for further research and development in areas such as flood risk assessment, urban planning, and environmental modeling, where accurate terrain and flood dynamics simulations are critical.