CS166 - Final Project

December 17, 2021

This is a really bad assignment. I'm sorry. There was a bug in the simulation such that it wasn't preserving the volume of water on the grid during the update. I wasn't able to fix this before the deadline. So, there are a couple secions missing. Also, there is a divide by zero error in the theoretical approximation that I also didn't have time to fix.

1 Intro

For this assignment, I modeled rainfall and flooding in the Beauford watershed in North Carolina. My goal was to track water levels at a cluster of houses located on one of the hillsides during a heavy thunderstorm.

2 Simulation

2.1 Model

I modeled this scenario using a 2-dimensional cellular automata on an nxn grid. An nxn grid called the ground_level tracks the height of the soil at each grid space, which is static over the course of the simulation. The height of the ground at each space (i, j) is referred to as g_{ij}. An nxn grid called the water_depth tracks the amount of water in each space. These are both measured in meters.

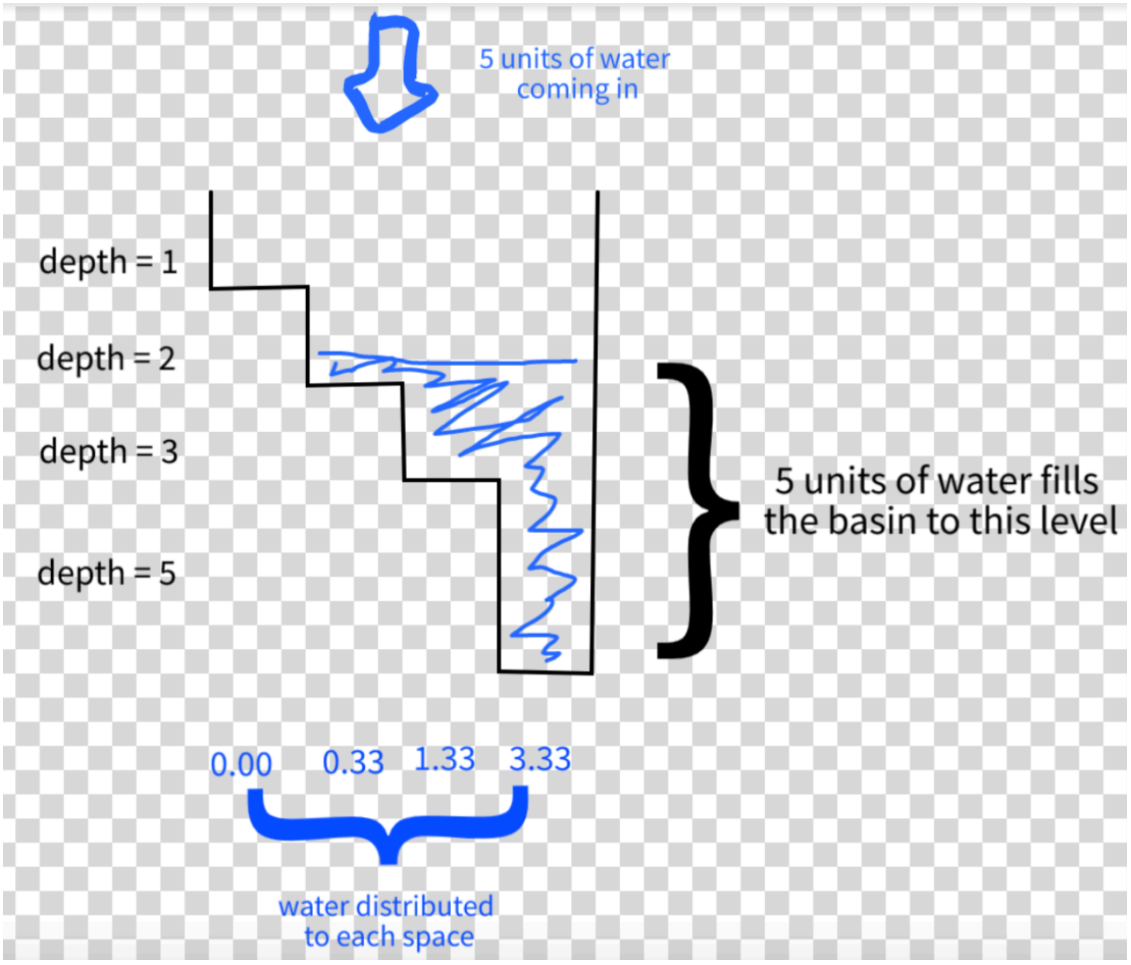
There are three main mechanics by which water moves around in the simulation: rainfall, inundation, and runoff. Inundation is when water is absorbed by the soil. Runoff is when water is transported downhill by gravity. The update for each of these steps is calculated in order for each grid space: first rainfall, then inundation, then runoff.

Rainfall is the main stochastic element of this simulation. I choose a simple mechanism to model. Each grid space on the map is assigned a number r_{ij} between 0 and 1. At each step, with probability r_{ij}, r_{amount} units of water are added to the space. The r_{amount} parameter is constant across the model. This allows for modeling disparate rainfall in different regions of the simulation, but it also keeps it simple and efficient.

To model inundation, the simulation tracks the total amount of water i_{ij} that has been absorbed by each grid space. This is initialized to zero for all spaces. I chose two parameters — i_{max} and i_{rate} — for the simulation. The i_{max} parameter represents the maximum amount of water that each space can absorb, and i_{rate} represents the speed at which water is absorbed by each space. On each step, each space (i, j) can absorb at maximum $i_{rate}(i_{max} - i_{ij})$ units of water. It will absorb water until either it reaches this threshold or there is no remaining water above ground in the space.

The simulation models runoff by working from each grid space and trying to use the water available in the space to balance the water levels between it and adjacent cells. At each step, an amount of water (called the excess water) is removed from each grid space (i, j). Then, that water is distributed among the 9-space Moore neighborhood of the cell.

The way the excess water is distributed is analogous to pouring water into nine connected bins with bottoms at different heights. The bin with the lowest bottom will start to fill up first, followed by the second lowest bottom, etc. This diagram illustrates how rainfall can be distributed in this way, although in the simulation there would always be nine bins rather than 4.

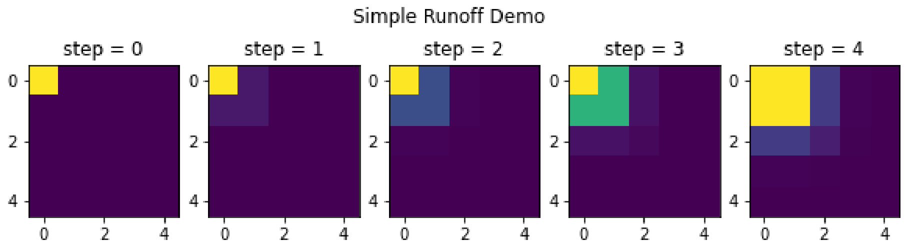

5 units of water coming in
depth = 1
depth = 2
depth = 3
depth = 5
5 units of water fills the basin to this level
0.00
0.33
1.33
3.33
water distributed to each space

Part or all of the excess water can be distributed back into the grid square (i, j). Once the excess water is removed from this space, its water level is recalculated, and it is treated like any of its neighbors for the purpose of distributing the excess water.

The amount of water that is removed from each space (i, j) is determined by how much higher the water level is at that space than the surrounding spaces. The amount is the difference between the water level at (i, j) and the lowest water level among cells in the Moore neighborhood of (i, j).

2.2 Demo

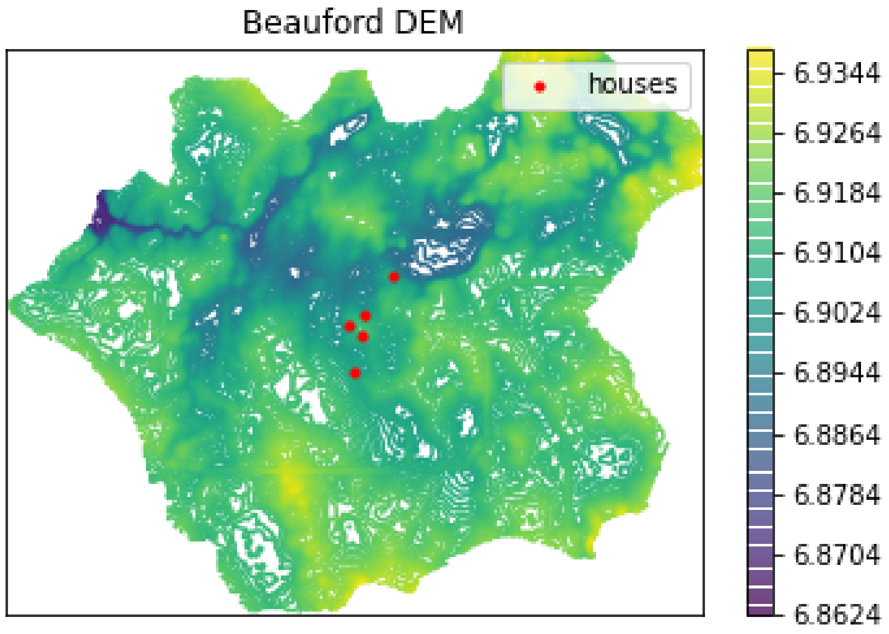
To verify that the simulation was working and illustrate the runoff rules, I simulated the process of a small column of water collapsing onto a flat plate. An illustration of this process is below. As you can see, the column quickly collapses, distributing the water across the plate. Small wavelets continue for a few steps but quickly dissipate as well.



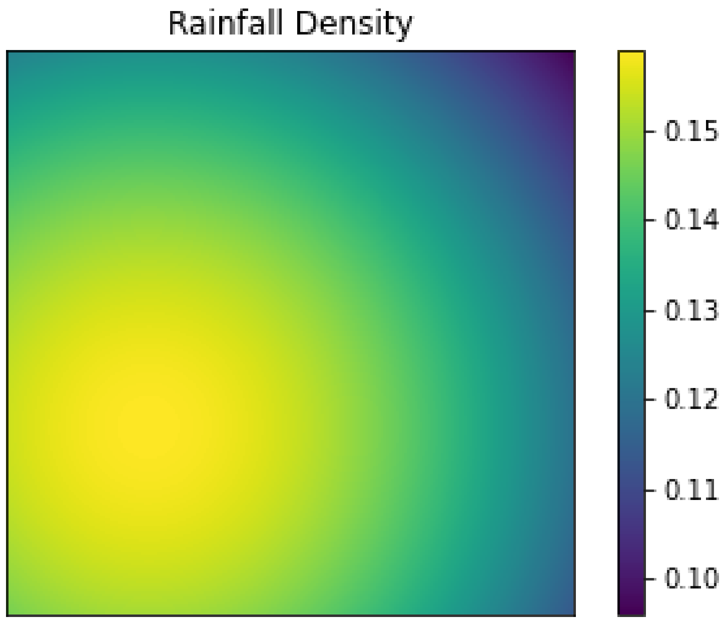
This was when I first realized that my simulation was not preserving water mass.

2.3 Parameters

The two most important parameters of the simulation are the ground height g_{ij} and the change of rainfall r_{ij} at each space. For the ground height, I used a dimensional elevation model (DEM) of the Beauford watershed in North Carolina. A map of this area is shown below. This is a contour plot of the logged elevation, which I found resulted in the most intuitive visualization. The red dots represent houses (I made up the locations).



I couldn't get good weather data for this region. So, I decided to try to model a rainstorm centered at the bottom left of the region. A plot of the r_{ij} values I used is shown below. I used a multivariate normal pdf to get this shape, but that was pretty arbitrary. I just wanted a shape that looked kind of like this.



There are a number of other parameters. To parametrize rainfall, there are r_{amount} and r_{ij}. I choose r_amount to be a small number (0.01). To parametrize inundation, there are i_{rate} and i_{max}. For i_{max}, I chose the value 0.04. So, each grid space can absorb up to 4 centimeters of rain. For i_{rate}, I chose 0.2. I had trouble finding research I could use to back up these values, so they are mostly based on guesses.

I chose to use an absorbing boundary condition for this simulation. At the edge of the grid, the unmodeled ground to the outside is assumed to be level with the height of the grid at the edge. Water runoff of the sides is then treated like water runoff into any other space. Water that leaves the grid in this way is removed from the simulation.

2.4 Outcome Variables

Floods cause a lot of damage if they get high enough to reach valuables, and they cause a lot of damage if valuables remain flooded for more than a short period of time. Otherwise, they are generally harmless. So, for this system, I chose to track the amount of time that the flood water submersed key areas of the map. If the floods waters failed to reach these regions, that would be considered especially good. If they are only there briefly, that is a bad but acceptable outcome.

I chose to study the levels of water reached at the house locations in the watershed.

2.5 Implementation

Performance was a major concern with this simulation. It is extremely computationally intensive to run for large numbers of timesteps, and I wanted to be able to experiment with multiple interventions at a reasonable speed. So, I used NumPy for the less computationally intensive parts of the simulations, and I used jit-compiled operations on Jax arrays for the expensive parts. I avoided normal Python as much as possible. Although the simulation is still slow, I was able to cut down the compute time significantly by using these libraries and a number of clever tricks to speed up various parts of the computation.

3 Theoretical Approximation

3.1 Modeling with ODEs

In order to create a theoretical approximation to this model, I decided to use a system of ordinary differential equations to model rainfall and runoff. This model has more simplifying assumptions than the simulation (for one, it neglects inundation), but it serves a good reality check. If the theoretical and simulated results differ too broadly, it could suggest that there is an error in the code.

Recall that I am modeling an $n \times n$ grid. For the purposes of the analytical model, I map each tuple (i, j) for $0 \leq i, j \leq n$ to a unique integer p on the interval $[0, n^2]$. So, some arbitrary attribute x_{ij} which might be might measured of a grid space (i, j) can be referred to as x_p. This makes it makes it possible to use column vectors to describe attributes of all the grid spaces.

In order to simplify considerations related to the relative heights of different objects, I define the runoff adjacency matrix A as follows. It is an $n^2 \times n^2$ matrix where the entry (k, l) is a 1 if the grid space k is both adjacent to the grid space j and has a lower soil elevation, and a 0 otherwise. So, for each row k of the matrix, the 1 entries denote the grid cells from which k collects water runoff. I call the space k a child of the space j (and j the parent of k) if $A_{i,j} = 1$.

Let E_p denote the excess water at a space p—the water that will be removed as runoff in the next step. Let I_p denote the water that will be received as runoff. Let R_p denote the water that will be received through rainfall. For convenience, let each of these quantities without subscript refer to the corresponding vector across all grid spaces p. I model the change in the water depth w'_p at a grid space p using the following differential equation.

$$w'_p = \alpha R_p + \beta(I_p - E_p)$$

Or, equivalently,

$$\nabla w = \alpha R + \beta(I - E)$$

Here, α and β are scalar parameters which control the relative importance of rainfall and runoff in the model. I assume that $\alpha + \beta = 1$ and that $0 \leq \alpha, \beta \leq 1$.

3.2 Deriving an expression for ∇w

For simplicity, I decided to set the water speed s as constant across the grid. The amount of water E_p that will be removed from a space p will be equal to the speed of the water and the amount of water in the space. I set the constraint $0 < s < 1$ for physical realism. So,

$$E = sw$$

I assume that outflowing water from each space will evenly distributed among child cells, i.e., which are adjacent in the grid and have lower soil elevation. Let O_p denote the amount of water that is distributed to each of the runoff adjacent cells. Let prod denote the elementwise product and inv denote the elementwise multipicative inverse. Then,

$$O = \text{prod}\big(E, \text{inv}(A^T\vec{1})\big)$$

This expression counts the number of children each node p has and divides the excess water E_p by that amount.

It is straitforward to derive an expression for I in terms of O. For each space p we simply sum O across parents of p.

$$I = AO$$

Finally, we derive an expression for R. Recall that r_p is the chance of rainfall at the gridspace p, and r_{amount} is the amount of rain that occurs. Let $\vec{r}$ denote the vector of the r_p for all p. Then the expected rainfall per timestep R is given by the following scalar/vector product.

$$R = r_{amount}\vec{r}$$

We can now restate the above differential equation in terms of the parameters s, α and β and known quantities of the system.

$$\nabla w = \alpha r_{amount}\vec{r} + \beta\Big(\text{Aprod}\big(sw, \text{inv}(A^T\vec{1})\big) - sw\Big)$$

Simplify, using the fact that $\beta = 1 - \alpha$,

$$\nabla w = \alpha r_{amount}\vec{r} + (1-\alpha)s\Big(A\text{prod}\big(w, \text{inv}(A^T\vec{1})\big) - w\Big)$$

3.3 Steady States

To determine the steady states of the system, we can solve for values of α, s and w for which $\nabla w = 0$.

$$0 = \alpha r_{amount}\vec{r} + (1-\alpha)s\Big(\text{Aprod}\big(w, \text{inv}(A^T\vec{1})\big) - w\Big)$$

Note that $0 \neq (1-\alpha)s$.

$$0 = \frac{\alpha}{(1-\alpha)s} r_{amount}\vec{r} + A\text{prod}\big(w, \text{inv}(A^T\vec{1})\big) - w$$

We can simplify the the central term using the diag($\vec{v}$) operator, which returns the diagonal matrix whose diagonal entries are the elements of v. The expression becomes,

$$0 = \frac{\alpha}{(1-\alpha)s} r_{amount}\vec{r} + A\text{diag}(\text{inv}(A^T\vec{1}))w - w$$

In other words, we need to solve the linear system,

$$s\Big(A\text{diag}(\text{inv}(A^T\vec{1})) - I\Big)w = -\frac{\alpha}{(1-\alpha)}r_{amount}\vec{r}$$

We can interpret this equation in an only slightly hand-wavy sense as saying that drainage (the left hand side) must balance rainfall (the right hand side).

We get a solution when both sides are zero. On the right hand side, this happens when either α or r_{amount} is zero. These mean the same thing: there is no rainfall. On the left hand side, this happens when $A\text{diag}(\text{inv}(A^T\vec{1}))$ is the identity, which only happens when A is the identity. The interpretation of this solution is that we get a steady state if there is no rain and no runoff; we just stay where we started. The left hand side is also zero when w is zero. The interpretation here is that if there is no rainfall and we don't start with any water, then we will never get any.

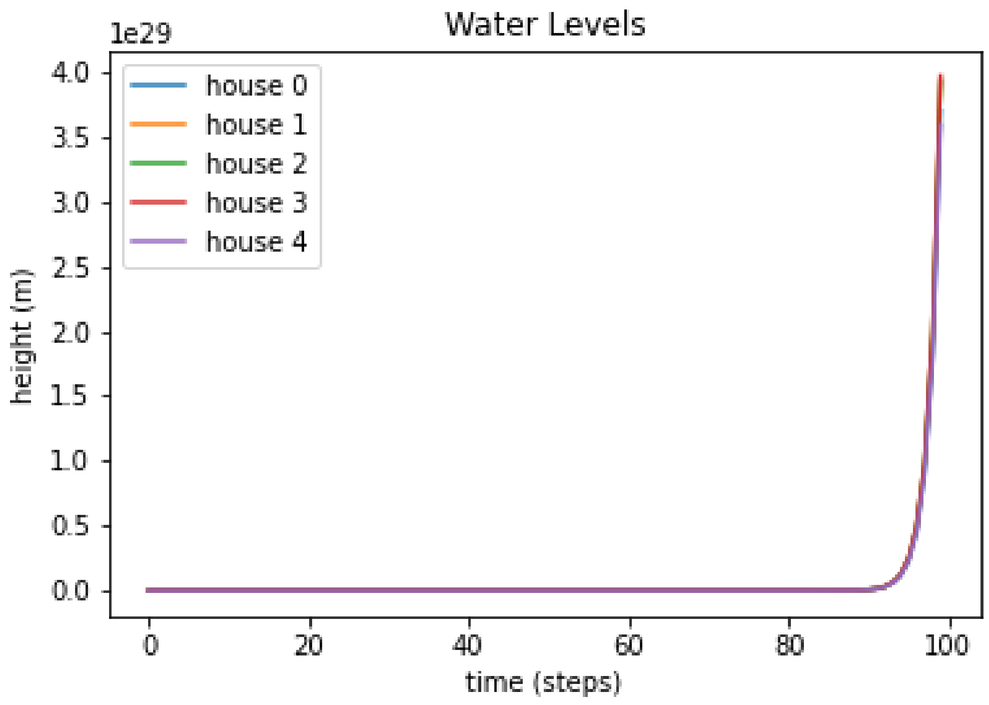
All of these make sense. This is evidence that the modeling and derivation were done correctly.

We also get a solution if the inverse exists. We can remove the constants and determine the steady state solution w.

$$w = \frac{\alpha}{(1-\alpha)s}\Big(A\text{diag}(\text{inv}(A^T\vec{1})) - I\Big)^{-1}r_{amount}\vec{r}$$

4 Simulation Results

The first time I ran the simulation, I tracked the amount of water at each house and plotted it. I got this:



Clearly, this doesn't make any sense. I can't figure out what's going wrong though.

5 Notes

This report is targeted at a team of engineers who are considering an intervention to reduce damage from flooding. Because the audience members would have technical backgrounds, I decided to use subscripts to describe most quantities because it is more concise and precise. I included enough material to satiate some of the curiosity they might have about the code.

The code is at https://github.com/leo-ware/waterflow