Exploring Simple Spatial Models of River Flooding

New Mexico

Supercomputing Challenge

Final Report

April 11, 2021

Team 34

La Cueva High School

Team Members

Yana Outkin

Project Mentor

Darko Stefanovic

Teacher

Yolando Lozano

Table of Contents

Exploring Simple Spatial Models of River Floods

Executive Summary

River flooding can destroy farm fields and prevent crops from growing. It would be useful to have a model to predict in what places the Rio Grande and connecting irrigation channels would flood if there are heavy rains. I developed a computational model for water flow. The model displays flooded areas in a given region based on the elevation data and initial water level for a region. The model takes into account the impact of soil absorption, the velocity of the water, and slope between two areas where water is transferred. The region being modeled is represented as a grid of individual cells that each have a height value, and current water level which changes over time. Each cell is compared with the states of its 6 neighboring cells to determine the new resulting water level, and then all cells are updated synchronously. A hexagonal grid is used so that the center of each cell is equidistant to each of its neighbors.

Here is an example result obtained from the simulation. This model can predict specific places in which water may accumulate given rainfall.

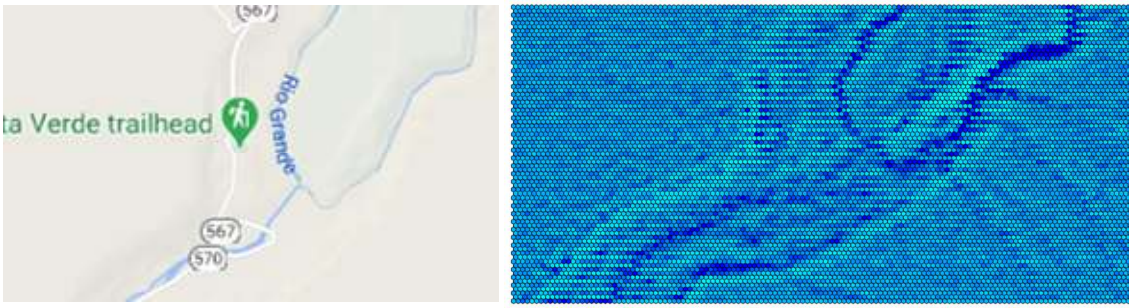

Figure 1: A result of the current model.

The model is intentionally simple. Many detailed models for water flow exist, but those tend to require extensive set-up and can be difficult to learn how to use correctly. An advantage of the current approach is that it is a simple model, runs fast even on a laptop, and uses publicly available GIS elevation data. The model's ease of use makes it useful for practical applications by farmers and city planners.

Introduction

Throughout the year I worked on two approaches to model flooding. The first approach uses a surface chemical reaction network simulator, and is described first in this paper. For the second and current approach I modified the simulator to perform numerical calculations on the water level of each cell - a synchronous cellular automata model. The results of these two models were compared to determine their effectiveness. The synchronous cellular automata model is more accurate in predicting what areas might flood, and faster as well. FEMA flood maps were used to validate the results of this more effective approach, and found it to be accurate in predicting flood-prone areas.

Surface Chemical Reaction Network (CRN) Model

Initially I used a surface chemical reaction network simulator because I was familiar with the surface CRN simulator that simulates chemical reactions on a surface. I was interested in seeing whether this framework could be used to simulate water flow.

The surface CRN simulator consists of 3 parts: the initial reactants on either a rectangular or hexagonal grid, a list of chemical reactions governing how the possible chemical species react, and a colormap that assigns a color to a specific chemical species for visualization. It is easy to see the progression of the chemical reactions on a surface with the colormap. The chemical reactions are either bimolecular, where two chemical species react to form two new chemical species, or unimolecular, where a species spontaneously reacts to form another chemical species.

I used a hexagonal grid with the “chemical species” being the water level and elevation of the specific cell. In the chemical equations, one unit of water was transferred between two cells on the hexagonal grid. I used the colormap to visualize how much water was in each cell during the reactions.

The chemical reactions are asynchronous. Each possible chemical reaction is scheduled a random time to react in a time distribution (Gillespie). The reactions are then placed in queue based on the scheduled times. The earliest scheduled reaction occurs first. Then the simulator goes through the queue, discarding reactions that have changed reactants due to the previous reactions that have already occured. The randomness of the scheduled times for the reactions is

useful to account for the uncertainty of water flow.

There are many interesting uses for this simulator, for example simulating forest fires where there are 3 types of “chemical species” - cells representing burned areas of forest, currently burning areas, and areas with not burned forest. I thought it would be interesting to see whether this framework could be applied to model flooding.

Chemical Reaction Network Framework

I generated a grid of the height and water levels for a region using Mathematica. The state of each cell on the hexagonal grid is a combination of the height and water level. Synthetic chemical reactions reflect changes in the cell's state where water is transferred.

Numerical calculations cannot be performed on the water levels using the chemical reaction networks. Instead one unit of water is transfered from a higher elevation to a lower one in a bimolecular reaction.

Each cell’s state is represented by {elevation_,’H’, water_}, for example 3H2, with elevation being a scaled value in feet, and w being the water level of the cell in inches. In each chemical reaction network, the initial reactants are two cells, and the products are the two cells where one unit of water has been transfered. The elevation for all cells remains constant, and is just used to calculate from which cell water flows.

Example Chemical Reaction Equation

9H2 + 8H7 -> 9H1 + 8H8 (4)

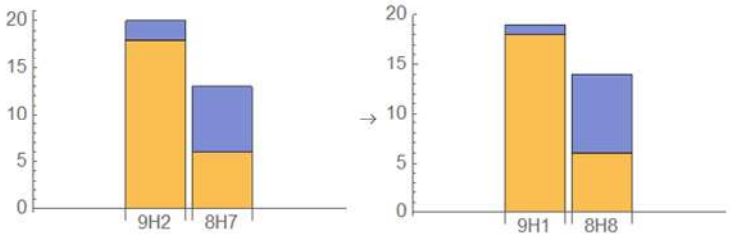

Figure 2: An example chemical reaction to transfer one unit of water.

The equation shown above is an example of a chemical reaction equation. In this surface CRN approach there are thousands of these equations due to all the possible combinations of two cells - two places with specific elevation and water levels. In this example the two neighboring cells are 9H2 and 8H7. In this example, the scaled difference in height values is 1 ft, and in water values 1 inch. 9H2 is a cell, with a elevation value of 9ft and a water value of 2 inches. 8H7 is another reactant cell with states being 8ft as the elevation, and 7 inches as the water level. One

unit of water is transferred from a higher elevation (9H2 = 9ft + 2 inches = 110 inches) to a lower elevation (8H7 = 8ft + 7 inches = 103 inches). The resulting states are 9H(2-1) and 8H(7+1). 9H1 and 8H8. The elevation values for each cell remains the same, and is just used in calculating water flow.

The following chemical reaction is seen above in Figure 2.

The rate constant (4) to the right of the equation represents the velocity of the water given the slope between the two cells, 4 inches per second.

Velocity is calculated using this equation:

$$v = \frac{\sqrt[3]{depth^2}.\sqrt{s}}{n}$$

where depth is the depth of the water, and s is the slope between the two cells (Cirbus and Podhoranyi). n is the Manning coefficient regarding soil properties, and was set to 1 for this approach. I didn't find specific soil data regarding water infiltration to use to calculate the Manning coefficient. I did find the total depth of the soil across a region but that was not useful for the calculations.

Scaling Elevations in CRN Model

The chemical equations must account for all the possibilities of the combination of two cell's states. The number of chemical reaction networks increases quadratically with a wider range of elevations and water levels. The total number of equations is the total possible heights, h, squared, times the total water levels, w, squared. Total chemical reaction networks = h^2 * w^2. To limit the amount of equations, the elevations and water levels are on a scale of 10. There being a range of 10 for possible elevation, and a range of 10 for water levels, there is a total of 10,000 chemical reaction networks. A greater range of elevation values would increase the number of chemical reactions needed, and make the simulation more precise, but it would run too slowly.

There is a compromise between the speed of the simulation and accuracy when using chemical reaction networks. A larger region has a greater range of elevations, a greater difference between the minimum and maximum elevation. As a result, each of the 10 scaled heights have to account for a greater range of elevation values. For example, in a region with a difference of 100 ft between the minimum and maximum elevations, each scaled height would represent a differ-

ence of 10 ft. As a result, the chemical reactions would not be able to tell the difference between 31 and 37 ft, for example, as both would have a scaled height of 3 (10ft = 1 scaled height), and there would be no water flow between the two cells as the elevations would be considered equal. For a larger region with a greater range of heights, this approach would decrease in accuracy due to scaling the heights. If the heights were not scaled, for example having 70 heights for 70ft total difference in elevation, there would be too many chemical reactions to account for all the possible states of each cell, and the simulation would not run in decent time.

This approach presents an interesting ending point for the simulation. When the water level is considered flooded, greater than the highest possible water level, 10 inches, there are no chemical reaction equations for the cell to react with. Therefore the simulation ends when a cell is flooded, or keeps alternating between two states (diffusion). Diffusion is used to model two cells that have equal scaled elevation levels and a water difference of 1 inch, where the water keeps alternating between the cells.

Velocity is accounted for in the rate constants. Velocity is directly proportional to the water level and slope between two cells. The resulting velocity, calculated using Mathematica (discussed in next section), is placed as the rate constant of the reaction. The rate of a reaction determines how fast the reaction will occur. Therefore a reaction with a greater rate constant (greater velocity) will occur more quickly. The addition of velocity using the rate constant makes the simulation more precise as water flows more quickly between two cells with a greater slope.

Water is conserved in the reactions, as it is only transferred among cells.

Generating Chemical Equations to Represent Water Flow

I used Mathematica to generate the elevation values for a region using the GeoElevation function. Next I assigned a specific water level to all the items in elevation data, creating a corresponding 2D array with the rainfall water level. The rainfall across the region was set to 3 inches simulating heavy rainfall. Using these 2 arrays with the elevation and water levels, I used Mathematica to format all the equations.

To format the chemical reactions all the possible combination of two possible neighboring cells were generated. Next, those combinations were evaluated to produce the products, where one unit of water was transferred from a higher to lower elevation. The scaled elevation was taken into account. The reactants and products were formatted and then exported to a text file to be used in the simulator.

I learned how to effectively use several Mathematica notebooks with connected variables. For example, in one notebook I entered in two coordinates for a region to export the elevation data, and then used the scaled elevation difference calculated in the first notebook to format the chemical reaction equations in another notebook.

I also learned how to use functions on every item in a list to create another list. For example I used the 2D array of all the two possible combinations between two cells to calculate the corresponding velocities. Then I used another function to format the two lists together into a chemical reaction equation format.

I also used Mathematica to create a colormap, assigning a color to a cell's state to visualize the reactions progress. With a range of 10 scaled elevation values, and 10 water values in inches, there is a total of 100 possible states for a cell. Each of the 100 possible states of a cell with a specific elevation and water level is assigned an RGB color. The greater the water level, the more blue in the RGB color.

Results: Surface CRN Approach

Here is an example of one of the simulations using the surface CRN simulator. On the left is a 3D elevation plot depicting the terrain, and on the right is the results of the simulation. This is a small region in Hidalgo County, which the Rio Grande flows through, that flooded in 2010.

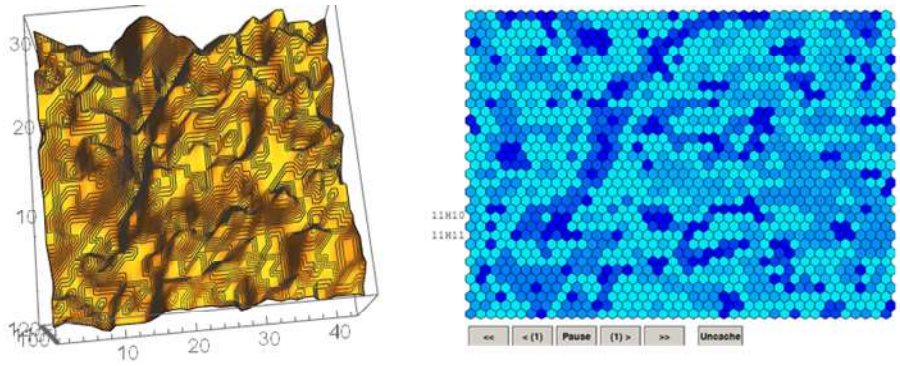

Figure 3: 3D elevation plot and results of surface CRN approach in a region in Hidalgo County.

The simulation accurately predicts that some of the rainfall water will flow into the river, while the rest will accumulate on the right of the river, possibly in arroyos. Turquoise represents no water, while dark blue represents 10 inches of water. The floodplain on the right of the river contains approximately 3 - 4 inches of water.

Water is spread evenly across the terrain to the right of the river, indicating flooding:

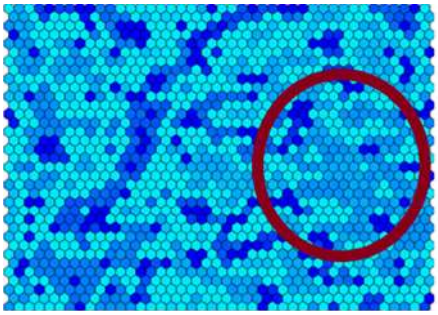

Figure 4: Water spreading across floodplain in a region in Hidalgo County.

The total elevation difference for this floodplain is 30ft, and it is expected that with 3 inches of rainfall the floodplain will flood.

Dark blue cells that are not a part of the river or neighboring channels are considered flooded. Bright turquoise cells contain no water. From this simulation, the floodplain on the right with 3 - 4 inches of water can be considered flooded. The dark blue patches to the right of the river might be arroyos, or channels in which the water has flowed in.

The simulation took some time to run due to processing the 10,000 chemical equations. For a larger region this simulation would become less precise due to the scaled heights. I decided to modify the simulator to create a custom model for water flow. I modified simulator to perform numerical calculations on the water levels of each cell, and take into account the real elevation data in feet without scaling. This approach would run much faster and be more precise as the heights would no longer need to be scaled. The model would also be able to account for more factors that affect water flow that the chemical reactions are not able, such as soil absorption.

Synchronous Cellular Automata Model

The model performs numerical calculations on the water level of each cell given the neighbors' heights and water levels. The changing water levels are visualized on a grid, where darker blues represent more water. The water levels for the cells are all updated synchronously. The model accounts for soil-absorption as well as the velocity of the water. To make the results more accurate, only a small percentage of water is transferred after each update. This creates a more continuous water flow across the region seen in real flooding.

Model Implementation

I modified the surface chemical reaction network simulator by Samuel Clamons (Clamons et al. 2020) because it had a lot of the basic framework I needed for my current model. Mainly I

needed to visualize the changing water levels, read in the elevation data for a region, and have a synchronous update rule that accounted for the neighboring cells' states. I used the pygame package for visualization, and built-in readers to read in the elevation and water grids as text files. I also used the colormap readers to assign a color for each specific water level for visualization.

Instead of deleting files from the simulator that I did not need with the chemical reaction equations, I copy and pasted the files that I did need into a blank project. First I used the readers to read in the elevation and water levels as text files. Since the original simulator was designed to read in one grid, I modified it to read in two grids. I found the files where the initial grid was initialized and used, and then followed the pattern to read in another grid. It was interesting to see organization of the files and how they were connected. For example, one file read in the entire text file, and another used specific phrases that indicated the start and end of a grid to parse the main text file.

I also discarded reading in the chemical reaction equations.

Update Rule

Next I added a file with the update rule. All cells used this one synchronous update rule instead of an update rule that fit the chemical reaction framework.

The update rule takes into account the neighbors' height and water levels to numerically calculate the resulting water-level in the center cell. Each center cell's water-level changes based on the water and height levels of the 6 neighboring cells (hexagonal grid). The update rule functions as a synchronous cellular automata. Soil absorption and velocity are taken into account.

Velocity is calculated using the slope between two cells and the water depth of the cell from which the water is transferred. The greater velocity of the water being transferred between two cells, the more water that is transferred.

Velocity is calculated using this equation:

$$v = \frac{\sqrt[3]{depth^2}.\sqrt{s}}{n}$$

where depth is the depth of the water, and s is the slope between the two cells (Cirbus and Podhoranyi). n is the Manning coefficient regarding how quickly water spreads over a surface, and was set to 1 for this approach. I didn't find specific soil data regarding water infiltration and

soil propertiesto use to calculate the Manning coefficient. I did find the total depth of the soil across a region but that was not useful for these calculations.

It is possible for water to leave the center cell and be transferred to all neighbors, and it is possible for water to flow into the center cell from all neighbors.

The soil absorption constant removes a certain percentage of water from each cell based on the soil type, and keeps removing a smaller amount after each update. The more water the soil absorbs, the greater the soil constant.

To make the simulation more precise, only a small percentage of the possible water that can be transfered is transfered in each update. This creates a more continuous water flow across the grid as seen in real flooding. Currently the model transfers 1% of the water; a lower percentage would result in a more accurate simulation but would take more time to run. The simulation reaches its stopping point when there is no visible change when visualizing the water levels.

I used one synchronous update rule for all cells so that the water levels were changed at the same time across the grid. In real flooding, water is flowing across the region.

Initializing the Simulation

I read two separate initialization grids for the elevation and water level as text files for a certain region into the modified simulator. I used Mathematica to generate an elevation grid from the GIS data for an area enclosed by two coordinates, using the GeoElevation function to obtain the elevation data. Then the elevation data was exported into a text file. The starting water level, simulating rainfall, is 1 inch across all cells. Using this data each cell was assigned a corresponding height and water level value.

For the update rule it is important to be able to check a cell's neighboring water and height levels. One function was already in the simulator to check the 6 hexagonal neighbors of the water-level grid. I did not find where it was initialized in the simulator, so I created my own function to find the 6 hexagonal neighbors of a cell for the second grid with the elevation data. I used this function when I called the update rule. I learned how to connect files in Python by importing classes and functions from different files. I realized the importance of organization on debugging by importing classes and functions, instead of having everything in one large file.

Units

The elevation data is in feet, while the water level is in inches. The initial starting water level for all cells is 1 inch of rainfall. This parameter can be changed for different regions.

Visualization

To visualize the changing water levels on a grid, water levels are assigned a corresponding color using a colormap. After the water levels for the cells have been updated, these water levels are visualized on a hexagonal grid. Darker blues represent more water, while a green/turqoise represents less water. For a greater range of colors I used Mathematica to generate a colormap with 300 possible values depending on the water level. I read in the colormap as a text file into the modified simulator.

Here is an example of a colormap with 4 values: 0, 1, 2, and 3 inches.

Figure 5: A colormap with 4 water level values.

The elevation and water grids are read in as strings, so in the update rule I converted the string values into floating point numbers. The colormap is also read in as strings, so trying to display integers would not work. A color was assigned to a string water level value. The updated water levels were not rounded, and were converted into strings only during visualization.

Sanity Check

I tested the update rule on some simple surfaces to check that the results of the simulations were reasonable. One surface was a slanted slope with a lower elevation towards the bottom, and the initial water level was equal across the grid. Here is the result of the simulation:

Figure 6: Result of the synchronous cellular automata model on slanted slope with equal initial water level.

As expected, water flows to the bottom of the slanted grid.

On another surface the elevation values were kept equal and the initial water level was a slope across the grid a slope. As expected, the water leveled out:

Figure 7: Result of the synchronous cellular automata model with a slanted initial water level and equal elevation values.

Results: Synchronous Cellular Automata Model

Here is a simulation of the intersection of the Rio Grande and Rio Pueblo de Taos in a trail in Taos. The Rio Grande flows from North to South, and the Rio Pueblo de Taos River merges with the Rio Grande from the northeast. The area of this region is approximately 2 by 2 miles.

Figure 8: Google maps region of intersection of Rio Grande and Rio Pueblo de Taos river.

The result of the simulation without the impact of soil absorption:

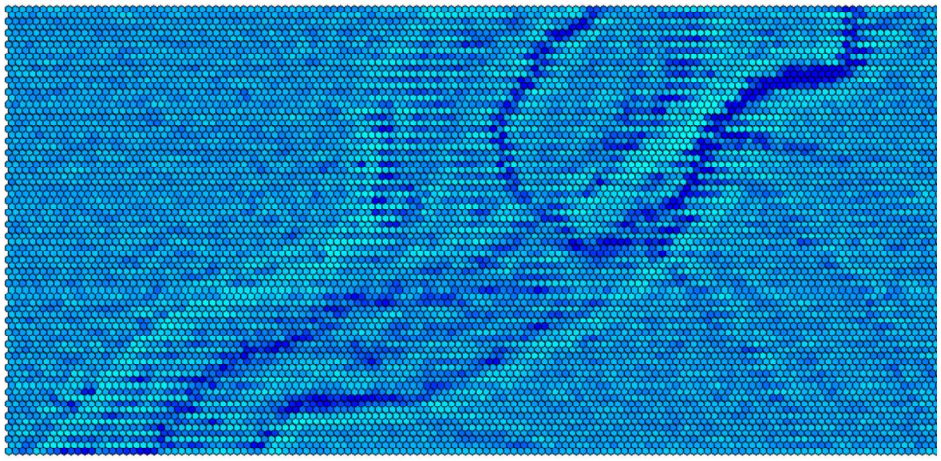

Figure 9: Results of synchronous cellular automata model where the Rio Grande and Rio Pueblo de Taos river intersect without the impact of soil absorption.

With soil absorption:

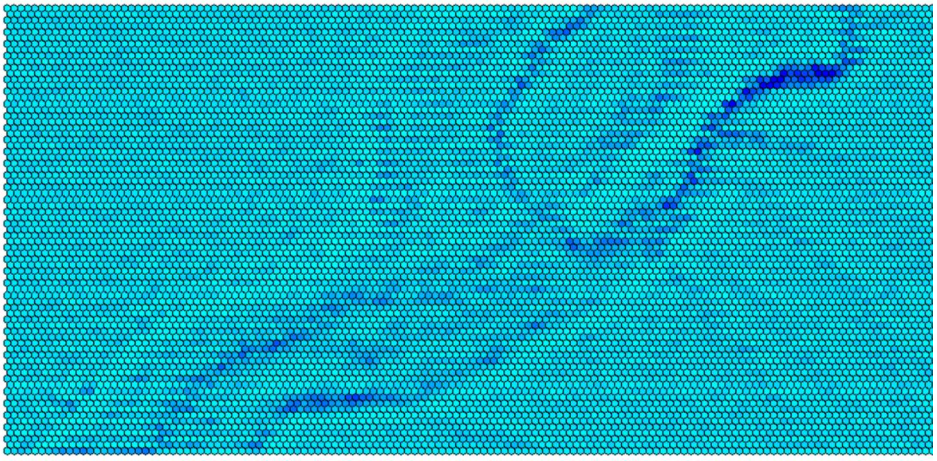

Figure 10: Results of synchronous cellular automata model where the Rio Grande and Rio Pueblo de Taos river intersect accounting for the impact of soil absorption.

In both simulations water is accumulating in the Rio Pueblo de Taos river and in the center of the Rio Grande. Based on the model this region will not flood with 1 inch of rainfall as the water will flow into the river and the rest of the water will be absorbed by the soil.

To the east of the river several small streams might form.

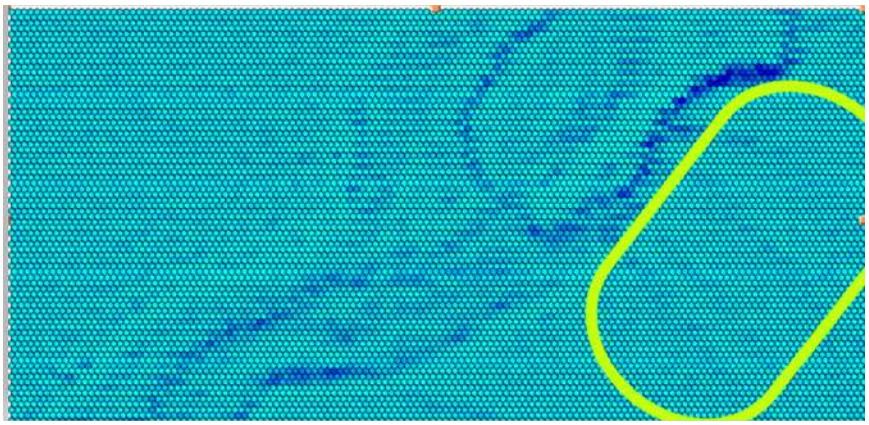

Figure 11: Formation of small streams to the east of the intersection of the Rio Grande and Rio Pueblo de Taos river.

The update rule can quickly update the water levels of the cells across a grid. It is faster, and has a greater possible resolution, than the surface CRN approach. I compared the two models to determine which model is more accurate in predicting the places water will accumulate given rainfall.

Comparison of Two Models

I compared the results of the current model to the surface chemical reaction network (CRN) approach to determine which model has greater accuracy in predicting in which specific regions water will accumulate. The current model is more accurate, and faster, in predicting which specific places will be flooded.

Rio Grande and Rio Pueblo de Taos River

Using the surface chemical reaction network (CRN) approach where each cell's state was a symbol representing both the water level and elevation, water would be transferred in the form of synthetic chemical reactions.

Due to the many possible states of each cell, the elevation values were scaled and as a result, there was less accuracy for a larger region due to the greater difference between the minimum and maximum heights.

Here is the result of running the surface CRN model on the same region with the Rio Grande and the Rio Pueblo de Taos river:

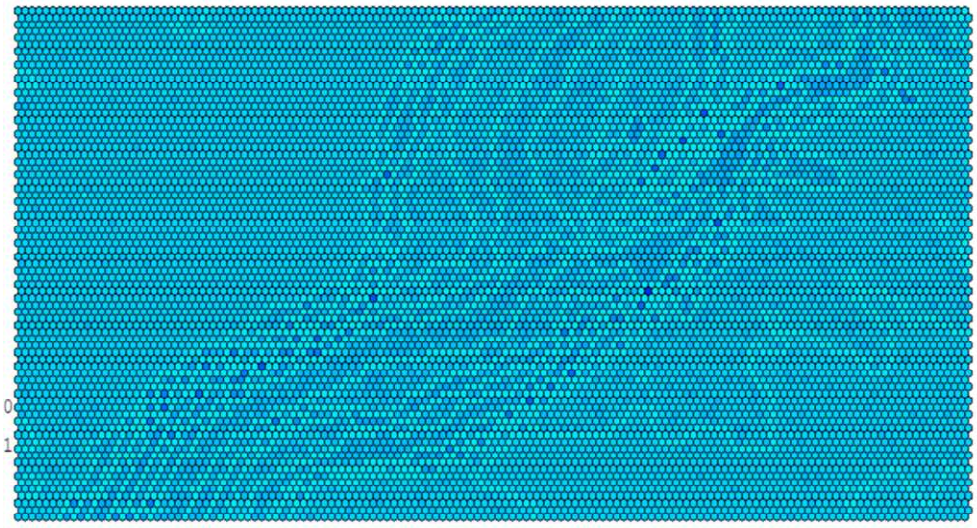

Figure 12: Results of surface CRN model on the intersection of the Rio Grande and Rio Pueblo de Taos river.

This model does not include the effect of soil absorption so water also accumulates both on the land as well as in the river.

In this model water is more equally spread out in the region even though it should accumulate in more specific places. This is because the heights have to be scaled to limit the number of synthetic reactions between the states of two cells. Each scaled height has a difference of tens of feet, possibly hundreds of feet, and this approach would not be able to distinguish the difference between specific elevations. As a result, water may not flow across a region with a small difference in elevation because the scaled elevation values would be the same. This accounts for the more even distribution of water across the region.

On the left is the surface CRN model where water is more equally distributed, and on the right is the result of the current model that performs numerical calculations on the water level of each cell given the neighbor's heights and water levels.

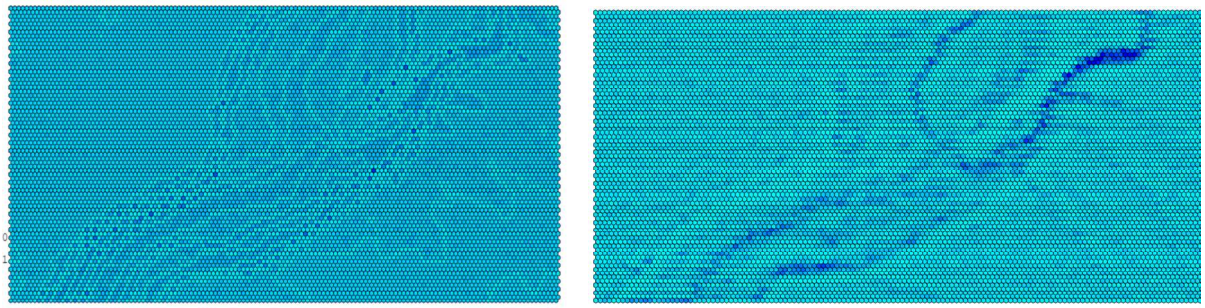

Figure 13: Side-by-side comparison of the results of the surface CRN model (left) and the synchronous cellular automata approach (right) where the Rio Grande and Rio Pueblo de Taos river intersect.

Given a larger region the modified simulator, the current model is more accurate and faster in predicting in what places water will accumulate.

Validating Results

I used FEMA flood zone maps to validate the results of the current synchronous cellular automata model. There wasn't adequate data for New Mexico, so I used the flood maps available for Florida. Florida has many rivers, and properties along the rivers have a high chance of becoming flooded. As Florida has more vegetation and greater humidity, I increased the soil constant for this simulation because the soil in Florida can absorb more water than the soil in New Mexico.

Here is a map of the Indian River County in Florida:

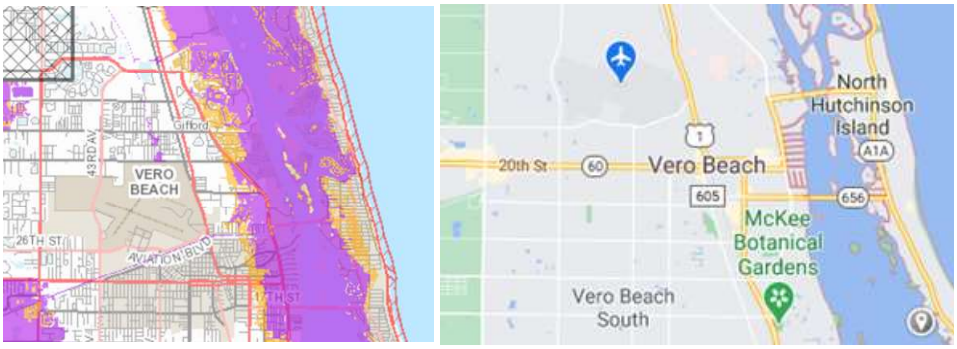

Figure 14: FEMA flood map and Google Map of a section of the Indian River County in Florida intersection.

The purple and pink regions have a greater chance of flooding with a .2% annual risk based on the FEMA data.

I ran my model on a section of the Indian river near Vero Beach where houses have a greater risk of being flooded.

Here is the result of the simulation on the right, and a zoomed-in section of the flood map:

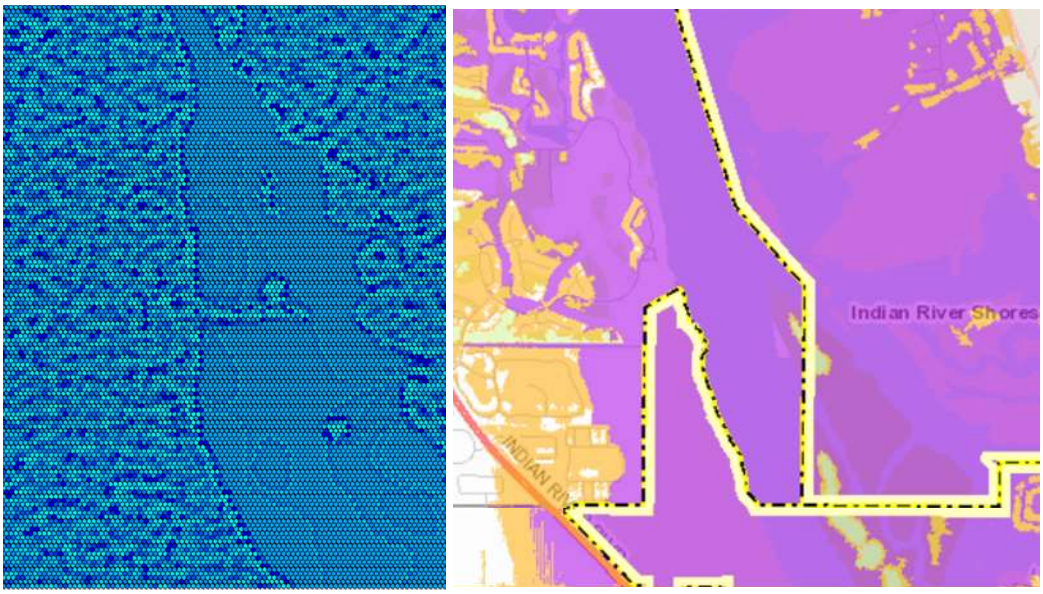

Figure 15: Result of simulation using the current model on a region in Indian River county, Florida, and the corresponding FEMA flood map.

There are several similarities between the two which are circled:

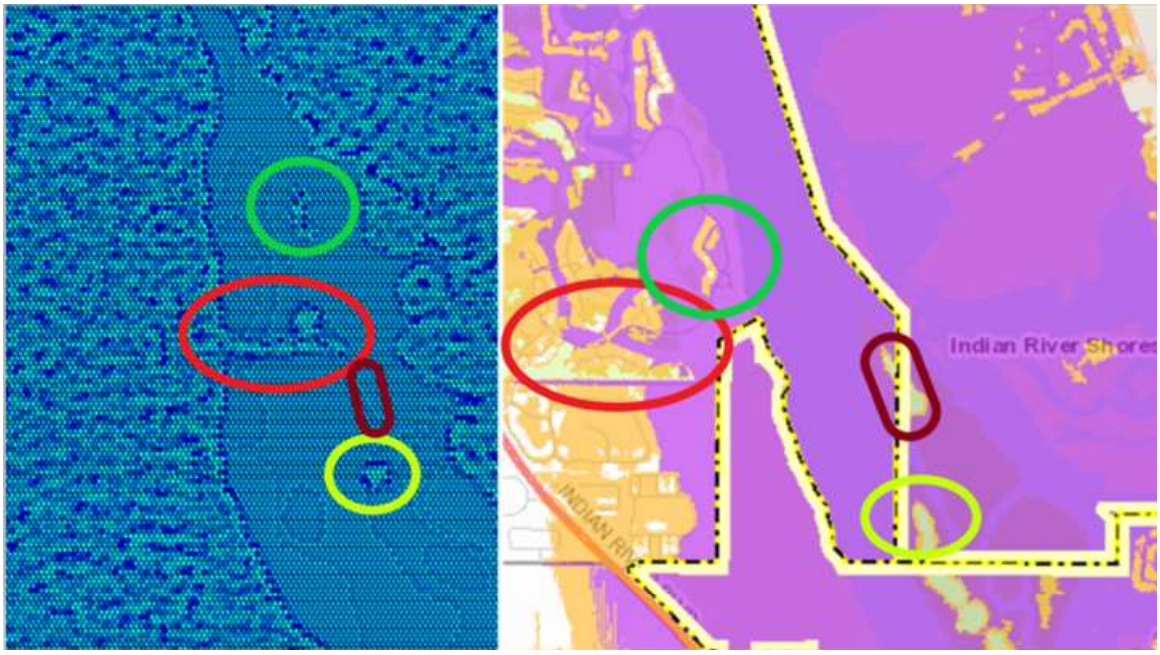

Figure 16: Similarities between the results of the simulation and the FEMA flood map on a region in Indian River County.

Much of the water flows into the river and the area surrounding the river.

Here is a similar area close to the Indian River in Florida which is likely to flood.

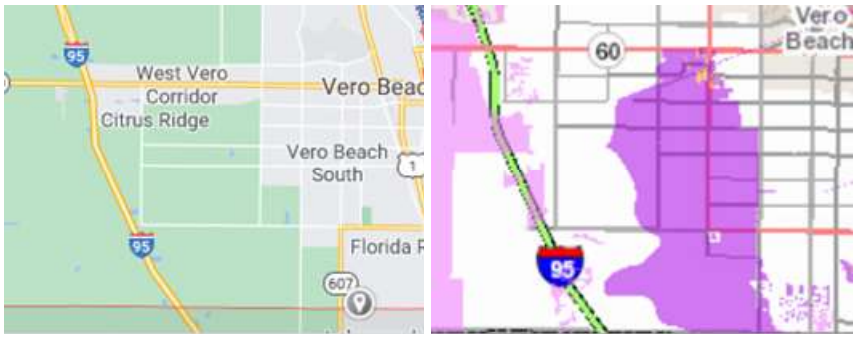

Figure 17: FEMA flood map and Google Map of Vero Beach in Florida.

Even with soil absorption, much of the water remains scattered throughout the region, indicating flooding.

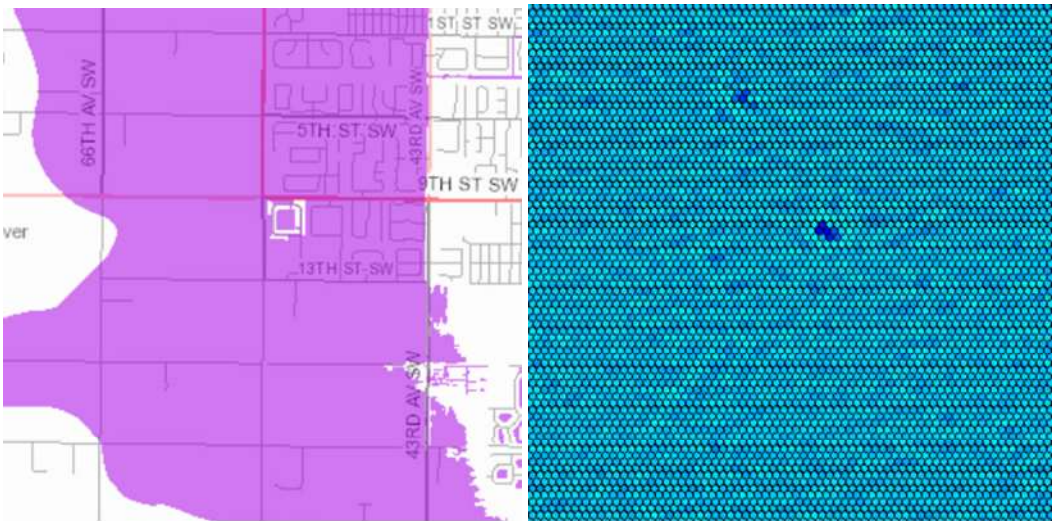

Figure 18: Results of current model on an area in Vero Beach, Florida.

Water flows into an area indicated by the flood map in the center of this region. This area is likely to flood, and the area surrounding this region (white on the flood map and bright turquoise in the simulation) is not.

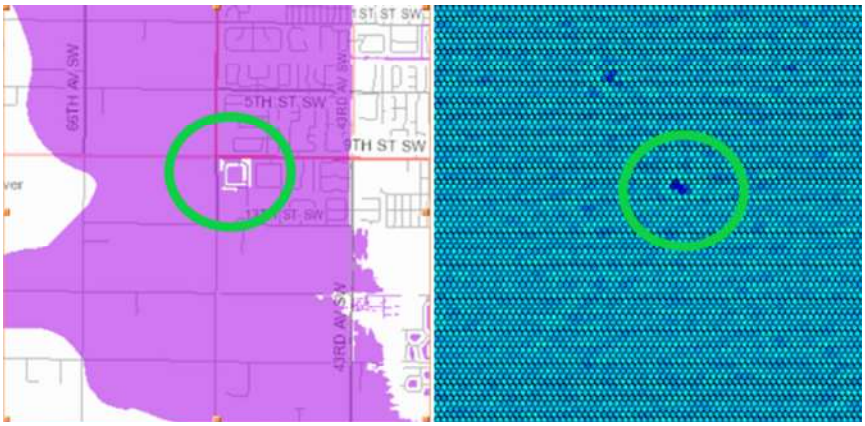

Figure 19: Similarities between the results of the simulation and the FEMA flood map in a region in Vero Beach.

For a region with an area of several miles squared, this model is effective in displaying the areas that are likely to flood.

Conclusions and Future Work

The synchronous cellular automata approach, resulting from modifying the simulator, is fast and effective in predicting flooding regions. It is interesting that a simple cellular automata approach accounting for mainly the velocity of the water being transferred, the slope between two cells, and the impact of soil absorption, can effectively predict possible flooding areas that are consistent with the FEMA flood maps. The surface CRN approach dealing with chemical reactions is much slower, and is limited in its ability to precisely represent distinct elevations for a larger region.

Future work includes comparing the results of the synchronous cellular automata model to high grade water modelling software such as the surface-water modelling system by the Army Corps of Engineers. The software is very precise, but it is difficult to learn how to use it properly. It would be interesting to add a continuous rainfall feature where water is added to the system at a constant rate simulating ongoing rainfall. It would also be useful to account for major highways and roads as water is going to flow faster over paved surfaces, as it is important to predict what specific roads are in danger of flooding.

Acknowledgements

I would like to thank my mentor Darko Stefanovic for the advice, conversations, and encouragement. I would also like to thank my sponsor teacher Mrs. Lozano for teaching me programming

in her computer science classes and her advice. I would like to thank the Supercomputing Project Manager Celia Einhorn for her help in navigating the challenge requirements and helping me organize my project submission.

References

Cirbus, Juraj & Michal Podhoranyi. (2013). Cellular Automata for the Flow Simulations on the Earth Surface, Optimization Computation Process. Applied Mathematics & Information Sciences. 7. 2149-2158. 10.12785/amis/070605.

Clamons Samuel, Qian Lulu and Winfree Erik. 2020 Programming and simulating chemical reaction networks on a surfaceJ. R. Soc. Interface.1720190790
http://doi.org/10.1098/rsif.2019.0790

Surface CRN Simulator: https://github.com/sclamons/surface_crns.git. Accessed 2 August 2020.

Douass, Souhaib. “Flood zones detection using a runoff model built on Hexagonal shape based cellular automata.” arXiv preprint arXiv:2007.10079 (2020).

Gillespie, Daniel T. “A general method for numerically simulating the stochastic time evolution of coupled chemical reactions.” Journal of computational physics 22.4 (1976): 403-434.

Vasić, Marko, David Soloveichik, and Sarfraz Khurshid. “CRN++: Molecular programming language.” Natural Computing (2020): 1-17.