Modeling California's Wildfires

Introduction

California is known for its devastating wildfires, occupying all of the top ten slots for costliest wildfires in the United States, each of which caused between 1 and 11 billion dollars in damages[1]. There are many factors that attribute to this cost, including residential damage, loss of life, fire suppression, and damage to the forest. The simulation you will be working with will focus mainly on how initial conditions affect the area burned in a forest environment.

Your team has been provided with a very basic wildfire simulation, and are tasked with adding extra parameters to make the model more realistic. These parameters include burn chance, forest density, and wind conditions. Each of these parameters will change how your model performs. Once you are satisfied with your changes to the initial model, you will generate several graphs to help you and your colleagues understand how these parameters affect the model. If you have extra time, there are several small bonus assignments at the end of this document.

Getting Familiar with the Simulation

Begin by opening the program "Wildfire.py" in Spyder, or your IDE of choice. Take a good look at each of the functions, but pay special attention to the Simulation Constants located near the top of the file. These will help you when adding new functionality to the model, and during your graphing phase. Once you are comfortable with the program go ahead and run it.

The simulation should run in a pop out window, and then close after it is finished. This is because matplotlib is currently using the "TkAgg" backend (shown on line 9). To get a print out of the simulation steps in the console, use the "inline" backend (shown on line 10) instead. This will help you get a better understanding of how your changes are affecting the model at each step. When using the "inline" backend, you can easily step through the simulation printout using the Pg Up and Pg Dn keys on your keyboard. Once you feel comfortable with using the simulation, begin the next section.

Modeling a Wildfire

[1] https://www.iii.org/fact-statistic/facts-statistics-wildfires

Now that you have a solid understanding for how the simulation works, and how to use it, you will begin modifying the program to add the additional functionalities explained below. Before you get started, since you will be heavily modifying the base program, it is highly recommended that you save to separate files early and often. Having a backlog of save files with different versions of the program will help prevent accidental loss of information, and will more easily allow you to restore your program to a working state when necessary. The following parameters can be added in any order, but the following the plan below is the recommended.

The first functionality you should add to the program is burn chance. This is the likelihood that any given tree will burn when one of its neighbors is burning. To do this, first define a function called burn_chance_happens that takes no parameters, and returns a Boolean value. One way to implement this is to use numpy's randint function to get integer values between 0 and 100, and return True when this value is less than the simulation constant BURN_CHANCE, and false otherwise. Once this function is working properly, use this function in the spread(x, y) function to alter the return value. When you are done, spread(x, y) should return true when the tree at position (x,y) is on fire and burn chance happens. If you have implemented your new function correctly, then when you run the program, the fire should spread in a more random pattern. This is to simulate the fire vulnerability of different types of trees, for example, Ash, Beech, and Redwood trees are generally very fire resistant, while Cedars, Cypress, and Juniper trees are very susceptible to fires due to their leaf density and shape. In this example, Ash trees would have a low BURN_CHANCE, somewhere between 50 and 65, while Cedar trees would have a high BURN_CHANCE, somewhere between 80 and 95.

Your second addition to this model should be forest density. This is meant to allow the program to simulate a wider range of forests since not all forests are densely populated. FOREST_DENSITY is the percentage (0 – 100) of live trees in the forest. To do this, you will need to modify the existing logic in function set_forest_state to set certain tiles to have FLOOR_STATE based on the FOREST_DENSITY simulation parameter. This can be done using numpy's randint function in a similar manner as was done with burn_chance_happens.

The final required addition to the model is wind. Environmental factors can be crucial in the strength, speed, and duration of a wildfire. Specifically, high winds can help fuel a wildfire by increasing the oxygen available to the fire, and can push wildfires along the direction of the wind by spreading embers (spotting) and heat down wind. To add this functionality to your model, you will need to create a new function that modifies the burn chance based on the direction of the wind, treat spread in the direction of the fire as additive, all other directions as subtractive.

Graphing Wildfire Data

When you begin graphing, you will have to run several simulations (one at each new parameter value), to properly show how changing model parameters affects the model. At this point, you will not want to display the results of each simulation in your console. It is highly recommended that you set the simulation constants DISPLAY and SHOW_STATS to False, otherwise your program will try to show you the results of each simulation step, and this will drastically increase runtime. Also, be sure that the global constants are set to the correct initial values as described in the "Wildfire Initial Conditions" file.

The last step for your model is to show how it reacts as we change the parameters we've added. To do this you will need to use the run_simulation function which runs the simulation and returns the number of live trees before and after that simulation is run. Using these counts you can get the number of acres burned to graph against your simulation parameter values. For example, to collect the data on the BURN_CHANCE Variable, you will have to run the simulation for each value of BURN_CHANCE (0-100), and collect the number of acres burned from each simulation for each value of BURN_CHANCE. You will then build a matplotlib plot using acres burned as the y data, and burn chance as the x data. You will need to follow this procedure for each parameter you've added to the plot, this includes burn chance, forest density, wind, and any other parameters you decide to add.

Presentation

Include in your final presentation a description of your code, the underlying model assumptions, the sensitivity testing, and the results with the given parameters as described below.

1. Prepare a summary presentation of the project that contains the following:
    a. A title, list of team members, and date of the contest
    b. A summary of the project results including the following sections:
        i. Description of the final main code along with a code listing
        ii. A summary of the model assumptions and how the model simplifies the systems that it represents

iii. A summary of the model runs made to derive the values of the parameters that met the objective of the modeling exercise. The summary should embed graphs and/or table to illustrate the model results.

iv. A summary of the sensitivity testing that was completed along with their results using appropriate graphs and/or tables.

2. If you undertake any of the extra projects, provide additional slides that provide the relevant summary, code changes, graphs, and tables as appropriate as a separated section for each of the bonus projects.

Helpful Notes:

- The "run_simulation" function returns the number of acres of living trees before the simulation, and the number of acres of living trees after the simulation has run.
- Assigning a value to a global constant within the [if __name__ == "__main__"] block will run the simulation with that global constant set to your new value.
- Each "tree object" in the simulation represents a square meter of land that is covered in trees, so the whole "forest" of 64x64 tree objects is a little more than an acre of land.
- Keep in mind that Python is sensitive to indentation, so each line of code that belongs to the same block should have the same indentation as the rest of the lines in the block.
- To produce your models, you should only need to interact with the Simulation Constants and the "run_simulation" function.

Recommendations:

- Restore the global constants to their initial value after you get the data you want from the simulation.
- Give your plots good titles and axis labels. Not only does this make your data look more professional, but it also makes it easier to understand what your plot is trying to show.

Bonus

- Generate a model relating wildfire cost to the number of acres burned using the US Wildfire Data at the top of the file.
- Create a function that modifies the burn chance based on the ambient temperature.
- Create a function that modifies the burn chance based on local humidity.

- Think about why the Acres Burned vs Wind Severity model has a negative trend even though an increase in wind should increase burn temperature, and fire spread rate. Hint: has to do with the simulation boundaries.
- Create a function that allows you to input the starting positions for each fire during runtime.