Forest Fire Expansion Modelling Simulation Report

# Introduction

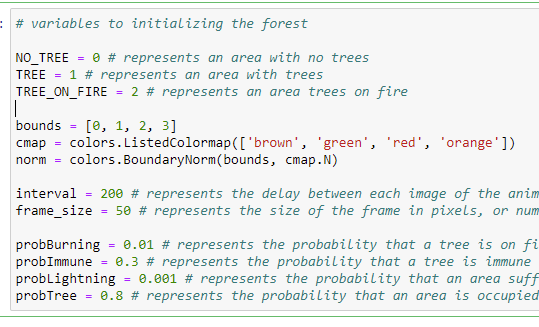
The goal of this report is to describe how the simulation of a forest fire expansion model was implemented using sequential and parallel processing methods. For the purpose of the simulation, the whole forest was treated as a grid of cells with values that either represent the absence of trees, the presence of non-burning trees or trees that are on fire. This phenomena is used for cellular automation simulations (Shiflet & Shiflet, 2006), for which modelling a forest fire exemplifies. The possibility of fire at any position expanding increases due to external influences like lightning strikes on the forest site and closeness to neighbouring trees that are on fire, depending on the immunity status of the tree.

# Initializing and Expanding the Fire

The simulation was implemented using the Python programming language, and some of its libraries namely, Numpy, Numba, Time, Random, and Matplotlib as shown in the Table below.

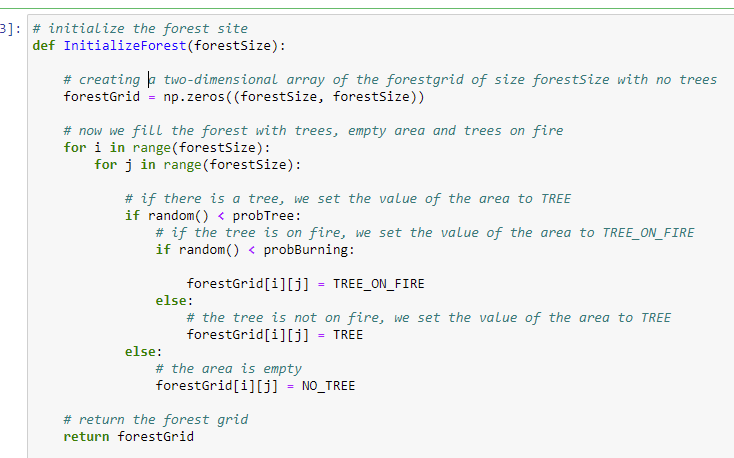
|  |  |
| --- | --- |
| Python libraries and tools used | Purpose |
| Numpy | Used for initializing and extending the boundaries of the forest grids. |
| Numba | Used for compiling and parallelizing the model functions. |
| Time | Used for capturing time used for a single iteration of the forest fire expansion |
| Random | Used to generate random values between 0 and 1 to test the probabilities. |
| Matplotlib | Used for plotting and creating the animation of the forest grids. |
| Jupyter Notebook | Interactive web-based platform used to write the models and display the forest grid animations. |

To initialize and expand the fire on the forest, a grid of zeros (or forest with no trees) were created with Numpy. After which the forest was looped through and the following conditions were applied at every stage.



The simulation of the forest fire being a Monte-Carlo simulation, (Gentle, 2003) as a result of the probabilities having an element of chance, random generator provided by the python was used to compare against the constant probabilities when filling the forest Grid.

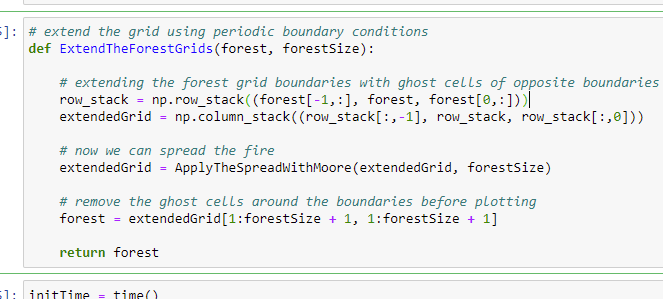
## Conditions for site initialization

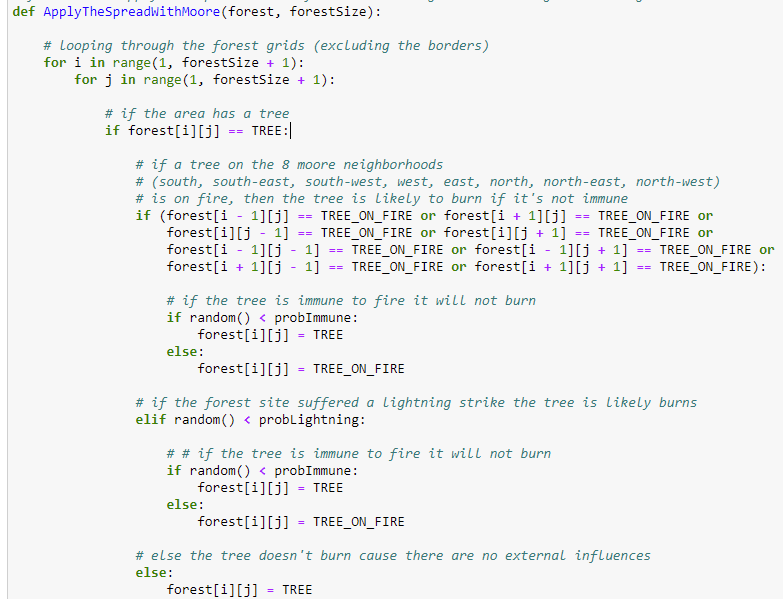


## Conditions for Fire Expansion

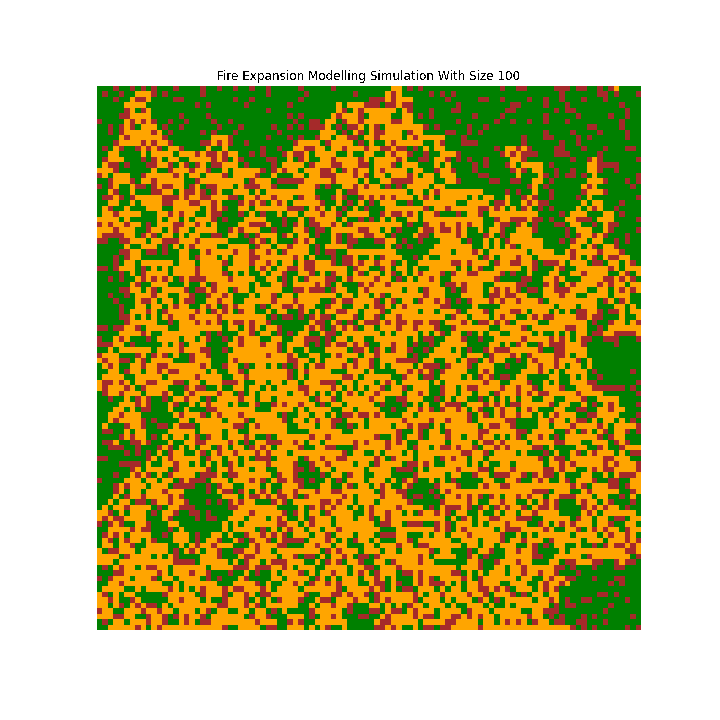
Once the grid is initialized, the forest boundaries are extended using periodic boundary conditions that works by creating invisible areas around the forest boundaries. It was done using the row\_stack and column\_stack functions provided by Numpy.

The essence of extending the boundaries is to mitigate boundary effect at the forest borders. And to help check for Moore neighbours of cells at the boundaries. The invisible areas are removed after each iteration.





# Result and Evaluation



The parallelization of the functions was achieved using Numba, as it provides just in time compilation of the functions. However, to improve execution time, Numba recommends to execute the functions one time before trying it again, so that it compiles it in the first trial, and makes significant difference in the execution time.

|  |  |  |
| --- | --- | --- |
| Forest Grid Size | Sequential | Parallel |
| 100 X 100 | 0.064 | 0.064 |
| 400 X 400 | 1.091 | 0.906 |
| 800 X 800 | 2.987 | 2.948 |
| 1000 X 1000 | 5.225 | 4.853 |
| 1200 X 1200 | 8.252 | 6.735 |
| 2000 X 2000 | 25.955 | 18.359 |

# Conclusion

I believe I have been able to show how parallelization reduces execution time. However, in the future, I hope to try applying my knowledge from this simulation in other domain areas.

# Bibliography

Gentle, J. E. (2003) *Random number generation and monte carlo methods*, Second edition. London; New York: Springer.

Shiflet, A. B. & Shiflet, G. W. (2006) *Introduction to computational science: Modeling and simulation for the sciences*. Princeton; Oxford: Princeton University Press.