Target Coverage & Result

Weiliang Zhao

January 28, 2023

1 Unit Test

1.1 Target Coverage

Unit testing is a method of verifying the functionality of individual units of code in isolation from the larger system. By isolating and testing specific functions, it is possible to identify and correct bugs or errors within the code, as well as detect any changes that may lead to simulation failures. Additionally, unit tests can detect incorrect input parameters and outputs, which can be identified through the reporting of failure messages and the output of specific parameter names and values. This level of granularity in testing allows for a more thorough assessment of the software's functionality and helps to ensure the integrity of the simulation results.

1.2 Result

The unit tests were implemented using scaffolding to verify the correctness of the input parameters and expected outputs. When an input parameter is changed outside of the specified range or an output does not match the expected value, the unit test will fail and return an error message indicating the specific input parameter or output that caused the failure. For example, if the firefighter location is set outside of the simulation map, the unit test will fail and return an error message stating that the firefighter is out of the simulation range, along with the incorrect location value.
However, the unit test is not able to report the location of the incorrect cell when the state of cells changes within the simulation, such as a cell transitioning from burnt to burning or flammable during the simulation. This limitation in the unit test could potentially lead to gaps and omissions in the sensitivity test.

2 Validation Test

2.1 Target Coverage

The validation tests for my software are designed to ensure that the output of the simulation matches the expected results based on the input parameters. However, due to the complex nature of fire simulations and the wide range of variables that can affect the outcome, it is possible that certain scenarios or edge cases may not be fully covered by the validation tests. Additionally, the limited number of test cases and examples used in the validation process may also impact the overall level of validation and the ability of the tests to identify potential errors or inaccuracies in the simulation results. As such, In the test result the software do not need achieve 100% accurate.

2.2 Result

The simulation results of the software are highly consistent with actual data, as it accurately models the spread of the fire and exhibits a high degree of alignment with key points in the real-world data. Based on this comparison data, it can be inferred that the software's testing has effectively reflected the code's accuracy. However, as previously mentioned, the data set used for this test did not include any boundary or extreme conditions. Therefore, in order to obtain a more reliable conclusion, it is necessary to conduct further testing using a larger set of test data, including edge cases and extreme scenarios, to fully evaluate the software's performance.

3 Sensitivity Test

3.1 Target Coverage

The sensitivity test for my software aims to evaluate the software's performance and behavior under varying input parameters and conditions. The target coverage for the sensitivity test includes a comprehensive range of input parameters, such as different map sizes, flammability levels, and firefighter deployment locations. Additionally, the sensitivity test will also include a variety of edge cases and extreme conditions, such as high-density urban areas, large-scale wildfires, and emergency response scenarios. The goal of the sensitivity test is to identify any potential issues or discrepancies in the simulation results and ensure that the software can accurately and effectively handle a wide range of scenarios. This is achieved by carefully selecting test cases to cover the different scenarios and variations of input parameters.

3.2 Result

The results of the sensitivity test for my software did not meet the expected outcome. Despite performing 100 random tests, the data obtained from these tests exhibited significant variability in terms of software execution time due to the random nature of the test cases selected. As a result, it was not possible to observe potential patterns or the impact of different data on the software performance from the 100 test cases. Further investigation and refinement of the sensitivity test methodology may be necessary to accurately analyze the software's performance under various scenarios.