Binomial Model Tree in MATLAB

Author: Mozahid Islam

Date: 01/11/24

Institution: Asian University of Bangladesh

Table of Contents

1. Introduction

The binomial model tree is a popular computational tool often used in financial modeling, particularly in options pricing, to simulate the movement of asset prices over time. This project aimed to create a binomial model tree in MATLAB, allowing for dynamic calculations and real-time visualization of the tree structure in MATLAB's console.

This report details the implementation steps, challenges, results, and potential enhancements of the project, highlighting the learning outcomes and insights gained.

2. Objectives

The main objectives of this project were as follows:

1. **Implement a Binomial Model Tree:** Construct a tree structure using MATLAB to represent upward and downward movements at each time step.
2. **Dynamic Visualization**: Provide real-time updates of the matrix in the MATLAB console, displaying the intermediate states of the binomial tree.
3. **Final Output Display**: Show the completed binomial tree matrix in the console for analysis after all calculations are done.

3. Methodology

3.1 Project Setup and Initial Parameters

The project was set up in MATLAB, with tests conducted on both MATLAB Desktop and MATLAB Online platforms. The following parameters were defined as input to the function:

- **Initial Value**: The starting value of the tree.
- **Steps**: Number of discrete time steps.
- **Rate of Change**: Factor controlling the upward and downward movements at each step.

The model was constructed using a matrix with the dimensions based on the number of time steps:

- **Rows**: 2 * steps + 1
- **Columns**: steps + 1

3.2 Binomial Model Tree Calculation

The binomial tree was calculated using a matrix where each node represented a possible value at a given time step. The function calculated each node as follows:

- **Upward Movement**: Each node moving up is calculated using the formula

  node * exp(rate_of_change).

- **Downward Movement**: Each node moving down is calculated using

  node * exp(-rate_of_change).

For each time step, the matrix was populated with calculated values based on previous nodes in the tree. MATLAB's exp() function was used to apply exponential growth and decay factors.

3.3 Dynamic Console Display

A key feature was displaying the binomial tree dynamically in the MATLAB console. Each state of the matrix was shown in real time, simulating live updates. To achieve this, the function:

- Used disp() to print each step's matrix.
- Included a brief pause() function to control update speed.
- Applied drawnow to force the immediate display of the current matrix state.

4. Results

Upon running the function, the binomial tree was dynamically built in the MATLAB console, displaying each intermediate state until completion. After the calculations were complete, the final state of the binomial tree matrix was shown, providing a complete view of all possible outcomes after the last time step.

Sample Final Output:

Column 1	Column 2	Column 3	...
Value	Up	Up	...
Value	Down	Down	...

This output showed the probability of each path within the tree structure, which can be used for further analysis or visualization.

5. Challenges and Solutions

5.1 Console Display in MATLAB Online

In MATLAB Online, the console behavior differs from MATLAB Desktop. Clearing the console with clc prevented real-time updates, causing the dynamic display to lag or not update correctly.

Solution:

- Removed clc to prevent clearing the console buffer.
- Used drawnow to ensure immediate display of updates.

5.2 Visual Clarity of Real-Time Updates

Displaying each intermediate step led to cluttered output in the console, which made tracking the tree structure challenging.

Solution:

- Adjusted the pause duration (pause(0.05)) to control the update speed.
- Increased spacing in the console output for better readability.

6. Possible Enhancements

1. **Graphical Visualization**: Implementing a graphical representation of the tree using MATLAB's plot or imagesc function could provide a clearer visualization of the tree structure.
2. **User Interface**: Developing a GUI using MATLAB App Designer would allow users to interact with the tree, adjust parameters, and view results in a more intuitive manner.
3. **Extended Models**: Extending the model to support different types of probabilistic models, such as trinomial trees, or different applications in areas beyond finance, like genetic probability simulations.

7. Conclusion

This project successfully implemented a binomial model tree in MATLAB, with a dynamic, real-time console display. The function built a flexible matrix structure that simulated possible upward and downward movements for each time step, displaying both intermediate and final results.

This experience deepened my understanding of probability modeling, matrix manipulation, and MATLAB's capabilities for real-time visualization.

8. Appendix: Code Implementation

MATLAB Code for Binomial Model Tree

```
function B = binomial_model(initial_value, steps, rate_of_change)
    total_columns = steps + 1;
    total_rows = 2 * steps + 1;
    B = zeros(total_rows, total_columns);
    middle_row = steps + 1;
    B(middle_row, 1) = initial_value;
    for i = 1:steps
        for j = 0:i
            B(middle_row - j, i + 1) = B(middle_row - j + 1, i) * exp(rate_of_change);
            B(middle_row + j, i + 1) = B(middle_row + j - 1, i) * exp(-rate_of_change);
        end
        disp(['Binomial Model Tree at Step ', num2str(i), ':']);
        disp(B);
        drawnow;
        pause(0.05);
    end
    disp('Final Binomial Model Tree:');
    disp(B);
end
B = binomial_model(10, 10, 0.15);
```

This structure gives a comprehensive overview of the **Binomial Model Tree in MATLAB** project, detailing each step from design to implementation, with explanations of challenges and future possibilities. This format also serves as a basis for further improvements and documentation.