Final Project

Finance 5350: Computational Financial Modeling

Due Date: December 18, 2020 at Midnight

Preliminary

Create a Python module in a file names `options.py`. A sample file is included in the folder for this project. You will add to this module as you progress through the final project. You will import it into various notebooks that will contain your answers to the problems below.

Part I: The European Binomial Option Pricing Model

Write a Jupyter notebook named `Part-One.ipynb` to solve the following problems. You should import the `options.py` module in the first code cell of the notebook as follows:

```
import options as opt
```

Problem 1

- **(a)** Complete the function `european_binomial_pricer` in the `options.py` module to implement the multiperiod European binomial option pricing model. This step is to be completed before you import the module into your notebook.
- **(b)** Verify that it works for both call and put options with $n = 1$ (i.e. a single period). Compare against a hand-written solution. Assume the following:
  - Let $S_0 = \$100$
  - Let $K = \$105$
  - Let $r = 8\%$
  - Let $T = 1$ year
  - Let $\delta = 0.0$ (i.e. no dividends)
  - Let $\sigma = 20\%$
- **(c)** Verify that it works for both call and put options with $n = 3$. Compare against a hand-written solution. Use the same parameters as above in **(b)**.
- **(d)** What happens if you set $n = 200$? Solve for both the call and put prices. **DO NOT** try to solve by hand! Again, use the parameter values from **(b)**.

Problem 2

- **(a)** Use the functions included in `options.py` to price the call and put option from **Problem 1** part **(b)** with the Black-Scholes option pricing model. See McDonald Chapter 12 for background on the Black-Scholes option pricing model.
- **(b)** Use the `european_binomial_pricer` function with the following values: $n = 20, 40, 60, 80, \ldots, 200$ (i.e. increment by 20). Compare to the Black-Scholes prices obtained above. Make a table to report the results. What can you say about the European Bimomial model relative to the Black-Scholes model? Discuss the convergence of the European Bimomial to the Black-Scholes model.

Part II: The American Binomial Option Pricing Model

Write a Jupyter notebook named `PartTwo.ipynb` to solve the following problems. You should import the `options.py` module in the first code cell of the notebook as follows:

```
import options as opt
```

Problem 1

- Using the functions `european_binomial_call` and `european_binomial_put` as starting points, implement the functions `american_binomial_call` and `american_binomial_put`. These functions should solve the optimal stopping problem implicit in the American option pricing problem. Write your solutions in the `options.py` module. This step is to be completed before you import the module for the problems below.

Problem 2

- **Set-up:** Let $S_0 = \$100$, $K = \$95$, $r = 8\%$ (continuously compounded), $\sigma = 30\%$, $\delta = 0$, $T = 1$ year, and $n = 3$.
- **(a)** Verify that the binomial option price for an American call option is \$18.283. Verify that there is never early exercise; hence a European call would have the same price. Compare your Python solution to a hand-written solution.
- **(b)** Show that the binomial option price for a European put option is \$5.979. Verify that put-call parity is satisfied.
- **(c)** Verify that the price of an American put is \$6.678.
- **(d)** Repeat each of the above for $n = 200$. How can you be sure there is never early exercise of the American call from part **(a)**? **DO NOT** attempt to solve this part by hand!

Problem 3

- Repeat the previous problem assuming that the stock pays a continuous dividend of 8% per year (continuously compounded).
- Calculate the prices of the American and European puts and calls.
- Which options are early-exercised? Explain your answer.

Part III: Simulating Binomial Trees

Write a Jupyter notebook named `PartThree.ipynb` to solve the following problems. Make sure to import the `options.py` module at the top of your notebook. You should know how to do that by now, so I won't belabor the point.

Problem 1

- Complete the function `binomial_path` in the `options.py` module that simulates a binomial path.
- This step is to be completed prior to being imported for the problem below.

Problem 2

- **Set-up:** Let $S_0 = \$100$, $r = 8\%$ (continuously compounded), $\sigma = 30\%$, $\delta = 5\%$, and $T = 1$ year.
- Set $n = 252$ (i.e. roughly the number of trading days per year so that each sub-period is a single day).
- Simulate a binomial path using your new function.
- Use the `Matplotlib.pyplot` function `plot` to make a plot of your simulated path. Label your axes appropriately.

Extra Credit

- **Set-up:** Let $S_0 = \$41.0$, $K = \$40.0$, $T = 1$ year, $\sigma = 30\%$ per annum, $r = 8\%$ per annum, $\delta = 0$.
- Price both European call and put options with the Black-Scholes model and the European Binomial model with $n = 200$ time steps.
- Now write a function that prices the European call and put via Monte Carlo simulation.
  - The solution should use your binomial path simulation to simulate $M = 10,000$ simulated paths through the tree.
  - This will give you M different terminal stock prices.
  - Get the corresponding option payoffs using the payoff function.
  - Take an average of the option payoffs with the Numpy method `np.mean`.
  - Discount this value to time zero and compare it with the Black-Scholes and European Binomial model prices.
  - Also calculate the standard error of the simulation with `np.std` and divide by `np.sqrt(M)`.
  - Repeat for $M = 25,000; 50,000; 75,000;$ and; $100,000$.
  - Make a table to report the data (both discounted mean and standard errors).