Option pricing with linear programming

István Ladjánszki

April 5, 2020

1 Option pricing with linear programming

This document summarises the code and results for the "Option priceing wiht lineat programming" project. The project were created for an Operations Research course that was helg in the 2020 fall semester at the Budapest Corvinus University. The code and the documentation were written by the author and of this document, István Ladjánszki and is distributed under the MIT licence. The code, the digested results, the source for this document and other supplemetary material can be found in the project repository on GitHub, at `https://github.com/ladjanszki/option_lp.git`.

1.1 Reproducibility

The development and run environment can be recreated by conda from the committed `environment.yml` file by invoking the

```
conda env create -f environment.yml
```

from the linux command line. For more infomration on this plese consult the conda manual. The run environment also requires the GLPK linear solver to be installed. The code was tested and all results were generated under `Ubuntu 18.04 LTS`. The code should work under Windows because of Pythons cross platform compatibility **but no test was done in this way**.

1.2 Code structure

The main files which have to be run are called `hw{1-4}.py` and all has the code for a given subsection in the Problem section. All helper code are stored in the `util.py` file. The main structure is the following. First a tree is generated with three stocks and cash as the zeroth asset. Based on this we have four assets in the problem. When building the tree the prices for

each asset generated from the parent node prices by the 1 equaiton. After building the tree is traversed in depth first way. Then an input file for the GLPK solver (glpsol under linux) is generated in the following way.

- The objective function has the starting portfolio amounts and have to be minimized.
- A self financing constraint added for every pair of world states between portfolio restructuring is possible.
- A non negativity constraint is added for every leaf node.
- A variable entry is added for every variable in the BOUNDS section

After the generation the GLPK solver is invoked and the output is redirected to a file. The output file is the parsed for a valid output of for the error message for unbounded problems. In case of an optimal solution the option price, the timing of the solver and other information is added to a summary file `report.csv`. The tables in the Problem section has these csv data as tables.

2 The problem

In the original problem we have a market which has three stocks and cash. The stocks are risky assets which means they have a prices in the different world states that depend on random variable ξ. The time evaluation of the i-th stock can be calulated by the equation below. We assume the price processes are normalized and discounted in a sense that cash always has a price of one.

$$S^i_{t+1} = 50 + 0.5 * S^i_t + \xi^i - \exp(-2.5) \tag{1}$$

The code in this repository gives the proce of an exchange option for different world settings and maturites. The exchange option gives the right for the owner to exchange one of the first stock to one of the second stock at maturity. In practice the payoff of these options are usually payed as the difference in cash. The starting price of all stocks equal to 100 and ξ is a lognormal random variable with mean 1 and standard deviation of 2.

2.1 One period different number of branches

In this section we used a one level tree for the stock prices. The tree had different number of leaves (100, 1000, 10000 and 100000) and we tried two different lognormal variables for stock price generation.i The results can be fount in the following table.

Option price	**Wall time [s]**	**Mean**	**Sigma**	**N Branch**
10.6220	0.2005	2	1	100
11.7820	0.3549	2	1	1000
11.9835	5.9615	2	1	10000
12.0706	923.1454	2	1	100000
12.1502	0.3272	0	2	100
12.1779	0.3892	0	2	1000
12.1820	6.2872	0	2	10000
12.1821	939.9210	0	2	100000

Table 1: One level tree

2.2 Two period with 100 branches

In this taks we priced the same exchange option. The stock price tree in this case was two levels deep and every node has 100 children. Based on this there are 10000 leaves in the tree. The option prices were calculated for two different generating lognormal distributions.

Option price	**Wall time [s]**	**Mean**	**Sigma**	**N Branch**
16.2196	7.3412	2	1	100×100
17.8773	7.6326	0	2	100×100

Table 2: Two level tree with two different lognormal variable

2.3 Three level tree with same number of branches on every level

Option price	Wall time [s]	Mean	Sigma	N Branch
N/A	0.2231	2	1	5
N/A	0.4714	2	1	10
13.4313	5.1865	2	1	20
15.5355	54.1588	2	1	30
16.6906	2302.9032	2	1	50
17.2658	19836.8210	2	1	70

Table 3: Three level tree with same branch number on every level

2.4 Three level tree with different number of branches on each level

In this case different number of branches have to be generated on every level. Besides that the portfolio can't be done on the first and the second day. The results can be found in the table below.

3 Usage

The entry point of the program is the $\text{option}_{\text{lp.py}}$ file. This shows the usage of the tree generator and the lp generator. After the file have been invoked a linear solver (testeg with glpsol under Ubuntu 18.04) have to be fed the generaqted input files All generated input files should be in the /inputs directory.

4 Testing

Testing can be carried out by invoking the test.py file This is not PyPi compliant testin only a validation of the working