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 Python's cross platform compatibility but no test was done in this way. The raw output for the code (e.g. generated lp files and GLPK solver output files are not committed to the repository since this is bad practice. All input and ouput can be reproduced by runing the code. In case of a 1000000 variable problem the generated lp file was a few hundred megabites in size and should not be part of a git repository.

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. If instead of the option price N/A presented it means the solver did not find the optimal solution.

2 Testing

Testing can be carried out by invoking the test.py file This is not PyPi compliant testing only a validation for the code. By running the file a tree is built for a CRR tree example and one for the example we had in the lecture. The results should be 33.3333 for the CRR case and 0.09 for the lecture test.

3 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.

3.1 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

The exchange option can not be sold for less than the optimized value since this is the cheapest self financing portfolio we can hedge the contingent claim with.

3.2 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

In this case the price of the contingent claim is higher than in the case of the one level tree. This is due to the fact that the deeper the tree the more "tailed" the price sitribution and we buy more rights with the same option.

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

In this case a three level deep tree have been generated. In the table below we can see the number of branches on each level at the last column of the table. As it can be seen from the table the time to solution grows exponentially with the number of branches which matches the expectations. The GLPK solver can be invoked with millions of variables. In our case the 100 × 100 × 100 case approximetely 4 million variables and the solver seems working. After this the time to solution limited the tests with bigger systems.

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
17.2658	19836.8210	2	1	100

Table 3: Three level tree same branches

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

In this case different number of branches (n_1, n_2, n_3) have to be generated on every level of the tree. Besides that portfolio restructuring can't be done on

the first and the second day. The results can be found in the table below for different n_i values. It can be seen that the price of the derivative increases as the number of the leaf nodes increases. The same argument can be stated here as in the second example about the more rights we buy with the mode number of leaf nodes.

Option price	**Wall time [s]**	**Mean**	**Sigma**	n_1	n_2	n_3
12.5849	1.2439	2	1	5	25	25
14.4631	18.069	2	1	5	50	50
13.5782	2.1034	2	1	10	20	20
12.8102	4.3088	2	1	10	25	25
11.4226	1.0791	2	1	10	30	10
14.3374	7.4782	2	1	10	30	30
14.6844	18.176	2	1	10	30	50
13.4860	9.8637	2	1	10	50	20
15.4675	23.596	2	1	10	50	30
13.0064	8.8087	2	1	20	10	50
14.3000	6.1029	2	1	20	20	20
14.6453	9.3920	2	1	20	20	25
12.1698	9.5032	2	1	20	25	20
14.4986	20.743	2	1	20	25	30
16.2050	53.584	2	1	20	25	50
13.8388	14.617	2	1	20	30	20
14.3226	20.127	2	1	20	30	25
15.3628	64.593	2	1	20	50	25
14.9632	80.658	2	1	20	50	30
16.3901	235.43	2	1	20	50	50
13.8580	3.6767	2	1	25	10	25
13.5660	5.4483	2	1	25	10	30
15.3462	13.205	2	1	25	10	50
15.2305	9.8623	2	1	25	20	20
15.0448	14.995	2	1	25	20	25
15.1830	19.045	2	1	25	20	30
16.2601	56.921	2	1	25	20	50
14.4837	16.391	2	1	25	25	20
14.8456	24.741	2	1	25	25	25
15.1199	32.788	2	1	25	25	30
15.8294	83.410	2	1	25	25	50
12.9405	6.1926	2	1	25	30	10
15.3801	23.322	2	1	25	30	20

14.4122	35.430	2	1	25	30	25
15.9571	48.750	2	1	25	30	30
15.9255	139.40	2	1	25	30	50
12.8200	17.749	2	1	25	50	10
15.3769	66.909	2	1	25	50	20
15.9113	97.199	2	1	25	50	25
15.2363	150.31	2	1	25	50	30
16.4178	378.08	2	1	25	50	50
13.4538	5.1909	2	1	30	10	25
13.6721	7.5660	2	1	30	10	30
14.1814	18.859	2	1	30	10	50
13.4409	14.134	2	1	30	20	20
12.8951	22.294	2	1	30	20	25
13.7375	27.878	2	1	30	20	30
15.9628	91.921	2	1	30	20	50
14.2243	22.748	2	1	30	25	20
14.4052	33.862	2	1	30	25	25
14.3239	45.546	2	1	30	25	30
15.1957	126.02	2	1	30	25	50
11.6226	8.2678	2	1	30	30	10
14.4814	32.180	2	1	30	30	20
15.4985	50.943	2	1	30	30	25
15.7484	77.562	2	1	30	30	30
15.7980	197.44	2	1	30	30	50
13.5920	26.625	2	1	30	50	10
14.8138	93.541	2	1	30	50	20
15.4964	152.84	2	1	30	50	25
15.5123	218.45	2	1	30	50	30
16.5364	618.51	2	1	30	50	50
11.1502	4.7117	2	1	50	5	30
12.5836	9.3704	2	1	50	10	20
15.2938	22.079	2	1	50	10	30
15.3378	56.709	2	1	50	10	50
14.4756	39.382	2	1	50	20	20
14.3157	68.940	2	1	50	20	25
14.6614	84.981	2	1	50	20	30
15.9737	241.66	2	1	50	20	50
15.7410	64.054	2	1	50	25	20
14.7997	100.74	2	1	50	25	25
16.0441	132.71	2	1	50	25	30

16.5137	376.48	2	1	50	25	50
12.6584	24.295	2	1	50	30	10
14.9843	96.912	2	1	50	30	20
14.7288	161.24	2	1	50	30	25
15.5946	214.81	2	1	50	30	30
16.6490	565.14	2	1	50	30	50
13.8552	74.657	2	1	50	50	10
15.8309	287.23	2	1	50	50	20
16.0927	456.80	2	1	50	50	25
16.0134	669.16	2	1	50	50	30
16.3224	1705.4	2	1	50	50	50

Table 4: Three level tree different branches