Electricity markets – Three islands connected with three cables

Table of Contents

1 - Problem Statement

In the problem we have selected there are three islands (representing three different markets) connected with three transmission cables, capable of transferring a certain amount of the exceeding energy to the other islands.
This problem started as an extension of the electricity market problem introduced in class, but evolved into a more advanced and defined one, due to the different possible interactions between the 3 markets.
In this report we're going to show our reasoning and how we used python to approach the problem and develop a solution.
We aim to analyse how the markets behave (i.e. how respective prices change) over changes in the capacity of the 3 cables.

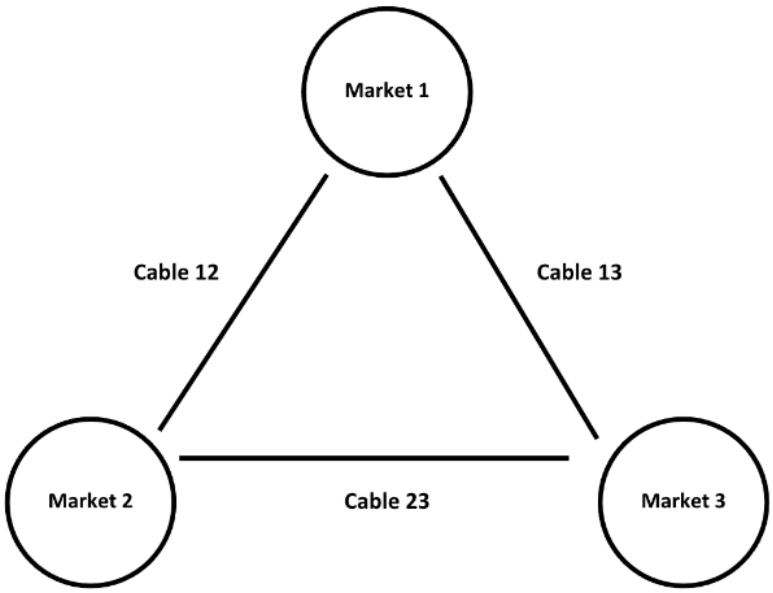

Figure 1 – Graphical representation of the three islands problem.

2 - Assumptions

The structure of the 3-markets environment is represented in **Figure 1**. The element which defines the markets, and their relations are:

- p_: price for the specified market;
- cap_{12}: capacity of the cable connecting Market 1 and Market 2;
- cap_{13}: capacity of the cable connecting Market 1 and Market 3;
- cap_{23}: capacity of the cable connecting Market 2 and Market 3;
- ES_: excess of supply for the specified market;
- d_level_, d_el_: respectively demand level and demand elasticity for the specified market;
- s_level_, s_el_: respectively supply level and supply elasticity for the specified market;

Our analysis is based on the assumption that:

- Each market has fixed and equal demand levels and elasticities;
- Each market has fixed supply elasticity set to 0;
- Each market has a different supply level;
- Cable capacities vary all together (i.e. if cap_{12} is set to 0.5, also cap_{13} and cap_{23} are set the same).

The last assumption is used to simplify the graphical representation, but our model and function implementation work also setting different capacities for different markets.

3 - Theoretical aspects

Depending on the Capacity level, on the supply levels and on prices relationships we are going to face 13 different scenarios, within 3 different market structures:

1. 3 separated markets (i.e. 3 different prices)
2. 1 unified market and 1 separated market (i.e. 2 different prices)
3. 1 unified market (i.e. 1 price)

We're going to focus on the 3 main cases separately. From now on we are going to use the acronym Cap to refer to the capacity level and specify which capacity when necessary.

3.1 - Case 1: Three separated markets

When there are three separated markets, the full capacity is used on all cables, two cables with electricity flowing into the market with the highest price, two cables with electricity flowing out of the market with the highest prices. The cable with the price at the mid-level receives maximum from the market with the lowest price and gives the maximum possible to the island with the highest price.
That leads to 6 different combinations given by the different relationship price can have:

- $p_1 > p_2 > p_3$, full capacity used to transport the maximum energy possible from Market 2 and Market 3 to Market 1;
- $p_1 > p_3 > p_2$, full capacity used to transport the maximum energy possible from Market 2 and Market 3 to Market 1;

- $p_2 > p_1 > p_3$, full capacity used to transport the maximum energy possible from Market 1 and Market 3 (via market 1) to Market 2;
- $p_2 > p_3 > p_1$, full capacity used to transport the maximum energy possible from Market 1 (via market 3) and Market 3 to Market 2;
- $p_3 > p_2 > p_1$, full capacity used to transport the maximum energy possible from Market 1 and Market 2 to Market 3;
- $p_3 > p_1 > p_2$, full capacity used to transport the maximum energy possible from Market 1 and Market 2 to Market 3.

Let us analyse the first case $p_1 > p_2 > p_3$ (the other cases behave analogously).

The equilibrium prices are reached when:

$$ES_1(p_1) + cap_{12} + cap_{13} = 0$$

$$ES_2(p_2) - cap_{12} + cap_{13} = 0$$

$$ES_3(p_3) - cap_{12} - ca_{13} = 0$$

which requires solution prices to be:

$$p^*_1 > p^*_2 > p^*_3$$

Where p^* indicates the equilibrium price for the scenario.

3.2 - Case 2: One separate market, one integrated market

When the excess of supply in a specific market is less than the cable capacity within that market and another market (formally, the absolute value of the excess of supply must be less than the Cap), all the excess supply flows to the market with the higher price, leading to unification of the

two other markets. When the latter condition is not met for the remaining market, we are presented with the case with one separate market and one integrated market.
Assume for example the separated market is island 1, then the 2 resulting markets are characterized by the following prices:

- p_1 - price on Market 1
- p_{23} - price on Market 2-3

In total there are 6 combinations in the one separate market and one integrated market case , since the price in the separated market (p_1 in the previous example) can be both grater and lower than the price in the unified market.
We analyse the scenarios in which we have Market 1 and Market 2-3. Note that the same reasoning is to be applied to the other 4 scenarios

First scenario: $p_1 > p_{23}$

Full capacities cap_{12} and cap_{13} are used to transport from Market 2 and Market 3 to Market 1.

The equilibrium prices are reached when:

$$ES1(p_1) + cap_{12} + cap_{13} = 0$$

$$ES_2(p_{23}) + ES_3(p_{23}) - cap_{12} - cap_{13} = 0$$

Which requires:

$$p^*_1 > p^*_{23}$$

We also must ensure that cap_{23} is large enough to allows Market 2 and Market 3 to unify:

$$|ES_2(p^*_{23}) - cap_{12}| < cap_{23}$$

and

$$|ES_3(p^*_{23}) - cap_{13}| < cap_{23}$$

Second Scenario: $p_1 < p_{23}$

Full capacities cap_{12} and cap_{13} are used to transport from Market 1 to Market 2-3.
The equilibrium prices are reached when:

$$ES_1 (p_1) - cap_{12} - cap_{13} = 0$$

$$ES_2(p_{23}) + ES_3(p_{23}) + cap_{12} + cap_{13} = 0$$

Which requires:

$$p^*_1 < p^*_{23}$$

Again, we must ensure that cap_{23} is large enough to allows Market 2 and Market 3 to unify:

$$| ES_2 (p^*_{23}) + cap_{12}| < cap_{23}$$

and

$$|ES_3(p^*_{23}) - cap_{13}| < cap_{23} .$$

The remaining 4 scenarios, as anticipated, follow the same pattern of the explained ones, so we avoid reporting them.

3.3 - Case 3: One unified market

When the cable capacities connecting each market is enough to allow the full excess of supplies to stream through each cable, we end in a situation with a single market denoted by a common price p* .
We have just one scenario in this case and the equilibrium price is reached by setting:

$$ES_1(p^*)+ES_2(p^*)+ ES_3(p^*) =0$$

Which requires

$$| ES_1(p^*)| < cap_{12} + cap_{13}$$

$$| ES_2(p*)| < cap_{12} + \text{cap}_{23}$$

$$| ES_3(p^*)| < cap_{13} + \text{cap}_{23}$$

to allow each market to unify with the others.

4 - Python Implementation

The main function we developed is *three_islands*, taking as input:

- d_level, d_el, s_level, s_el for the 3 markets
- cap_{12}, cap_{23} and cap_{13}

The function gives as output 3 arrays containing the equilibrium price for each market in each of the 13 scenarios, and a *whichcase* array.
Each value of *whichcase* can be either:

- 1 if the requirements for the scenario are met;
- 0 if the requirements for the scenario are not met.

Every time *three_islands* is used, *whichcase* array must include just a single value of 1 and twelve values of 0.
Before analysing *three_islands* in depth, we briefly explain the functions underneath it:

- *demand,* it provides the demand function for a specified price, demand level and demand elasticity.
- *supply,* it provides the supply function for a specified price, demand level and demand elasticity.
- *excess_supply*, compute the excess of supply, using *demand* and *supply. The addterm* component is used to add/subtract the cap level in *three_islands.*
- *common_excess_supply_two*, compute the common excess of supply in a 2-unified-market situation. It includes the element presented above for *excess_supply*.
- *common_excess_supply_three,* compute the common excess of supply in a 3-unified-market situation. It includes the element presented above for *excess_supply*. It doesn't inlcude an *addterm* component.

Here's the step that the function aims to achieve:

1. *Variables initialization:* The function sets up zero arrays for *whichcase*, *price1*, *price2* and *price3*. These arrays will be used to store the state of the market scenarios (feasible = 1 /not feasible = 0) and the prices for each market.

2. *Computing the prices*: In each step of the if-else statement, the function starts with the calculation of the price for the 3 markets, optimizing one function among *excess_supply, common_excess_supply_two* and *common_excess_supply_three*. The optimization function used is *opt.bisect* from the scipy library. This function uses a numerical method to find the roots of a

continuous function. It achieves this by starting at two points (in our case 10^{-8} and 10), iteratively narrowing down the interval to approximate the roots' location.

3. *Computing excesses of supply*: If required by the if statement, excesses of supply are computed.

4. *Checking feasibility of the scenario*: for each scenario, the function checks if the requirements are met. If the requirements are met, it's assigned a value of 1 to *whichcase* (0 otherwise).

5. *Going on:* If the requirements are not met, then the function continue with the next scenario.

6. *Return*: In the end, the function returns 4 arrays containing *whichcase* binary values and prices.

Moreover, we implemented a for loop to try every capacity level and plot the behaviour of each market's price respect to the cap (as mentioned before, the three capacity levels vary together).

5 - Results

After fine-tuning the *three_island* function, we incorporate it into a for loop to show how the price changes for each market due to the change in capacity constraints.
We display the percentage of capacity on the x axes and the price on the y axis.

Having assumed that each market has a different supply level, we were expecting to observe an initial situation with three separate markets, where the distances in the initial price levels depended on the spread between the supply levels. Then, we were expecting, for a certain level of capacity, the joining of the two nearest markets (in term of supply level) and, eventually, the fusion of the compound-market and the separated one into a unified single market.

Indeed, we anticipated that the capacities of the cables needed to merge the markets were inverse-proportional to the spread in supply levels.

We tried different setting for the supply levels of the 3 markets, resulting in the plots reported below.

Figure 2 shows how the structure of the overall economic environment changes in relation to the changes of the cable capacities.

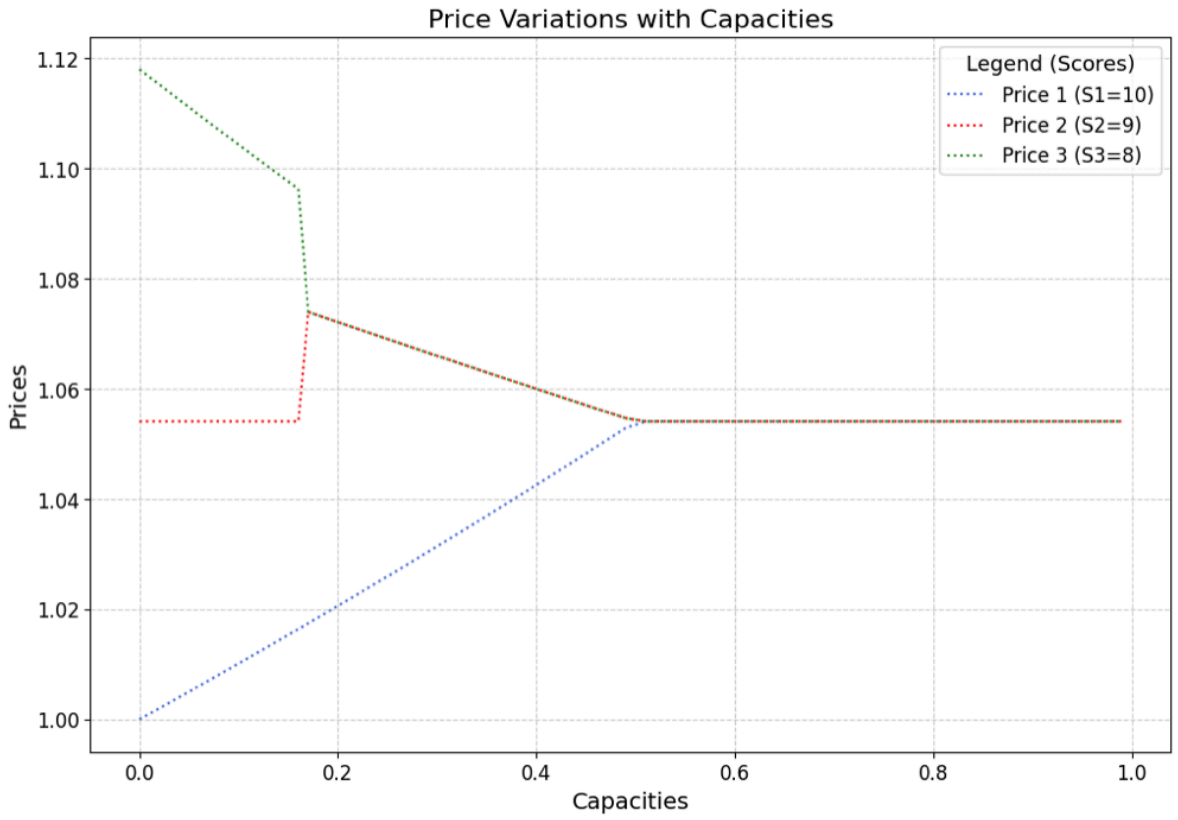

Figure 2 – Markets behaviour for initial supply levels.

Initially, the three markets have different prices. We can observe how p_3 decreases, since Market 3 is the one with the lower supply level, thus the one which would benefit the more (from customer perspective) from an increase in supply due to electricity stream through the cables. P_2 remains constant until the merge of Market 2 and 3 (around cap = 0.18) then p_{23} decreases to slowly reaching a merging point between Market 2-3 and Market 1 (around cap = 0.5).

p_1 increases until the merging point of Market 1, since Market 1 is denoted by the highest level of supply, so reducing the excess of supply (by streaming energy through the cables) result in a reduction of the price.

Note again that we assumed the demand level to be fixed and constant along the 3 markets, so just the changing in supply level influences the equilibrium price (also changing in capacity levels influence indirectly – trough the influence in the supply levels – the prices).

Moreover, we tested the validity of the function by switching the hierarchy order of the supply level between the Market:

- Original order: *s_level1 > s_level2 > s_level3*
- New order: *s_level3 > s_leve11 > s_level2 **(Figure 3)***

In the **Figure 3** we can notice how the merging points and the prices behave similarly when the order of the "supply hierarchy" is changed.

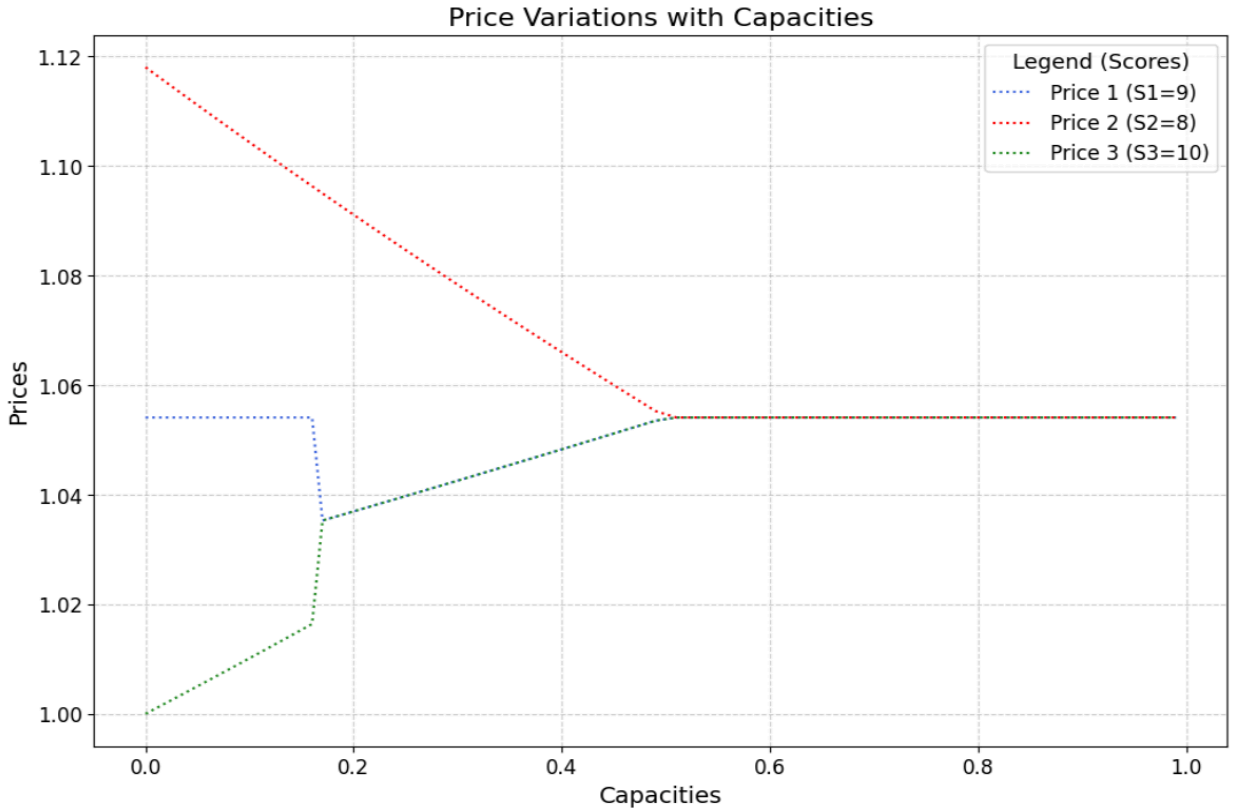

Figure 3 – Markets behaviour switching supply hierarchy.

Furthermore, we tried to change the supply levels, maintaining the initial hierarchy (*s_level1* > *s_level2* > *s_level3*).

Figure 4 was obtained reducing s_level2 from 9 to 8.5, resulting in a reduction in the capacity needed to merge Market 2 and 3, along with a small increase in the capacity needed to merge Market 2-3 with Market 1.

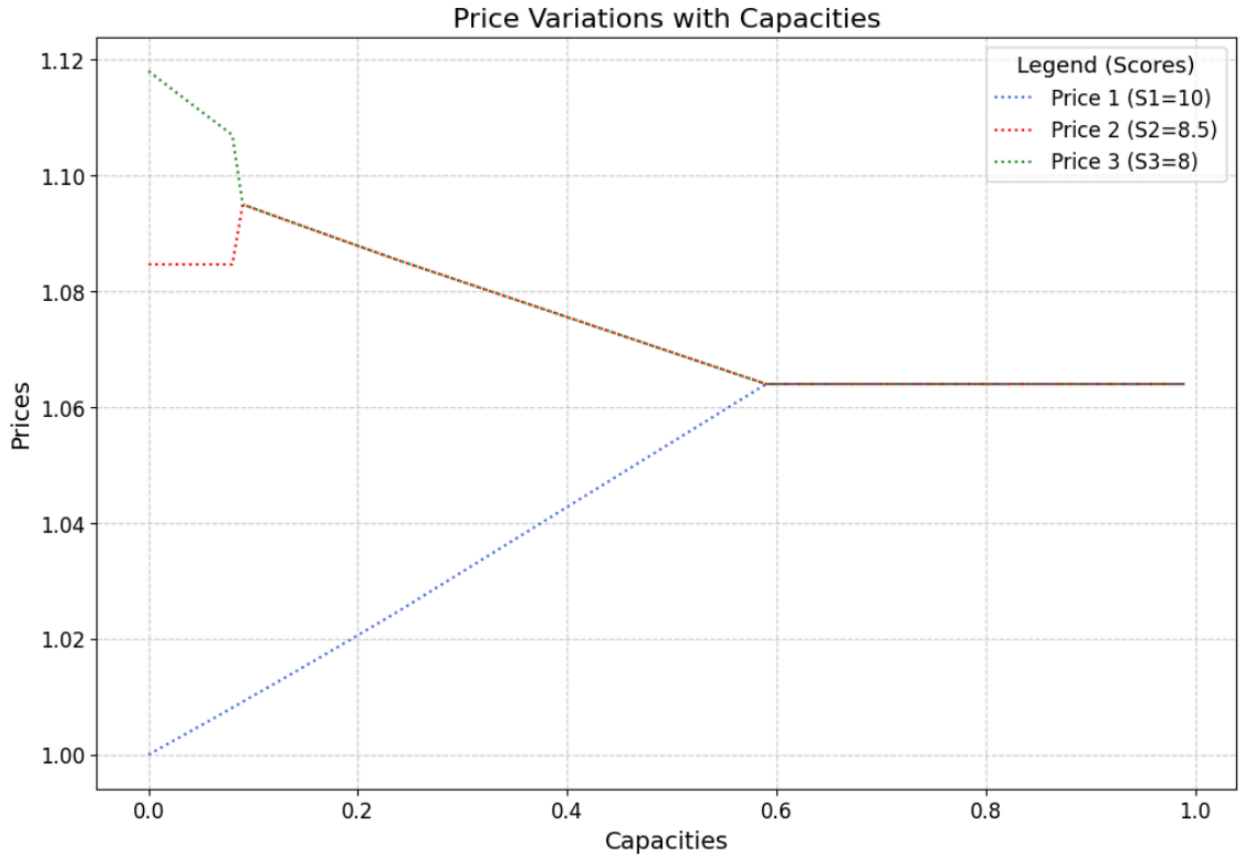

Figure 4 – Markets behaviours after a reduction in spread between supply 2 and supply 3.

Analogously, **Figure 5**, generated by increasing the s-level1 from 10 to 10.5, shows an increase in the capacity level needed to merge Market 1 with Market 2-3.

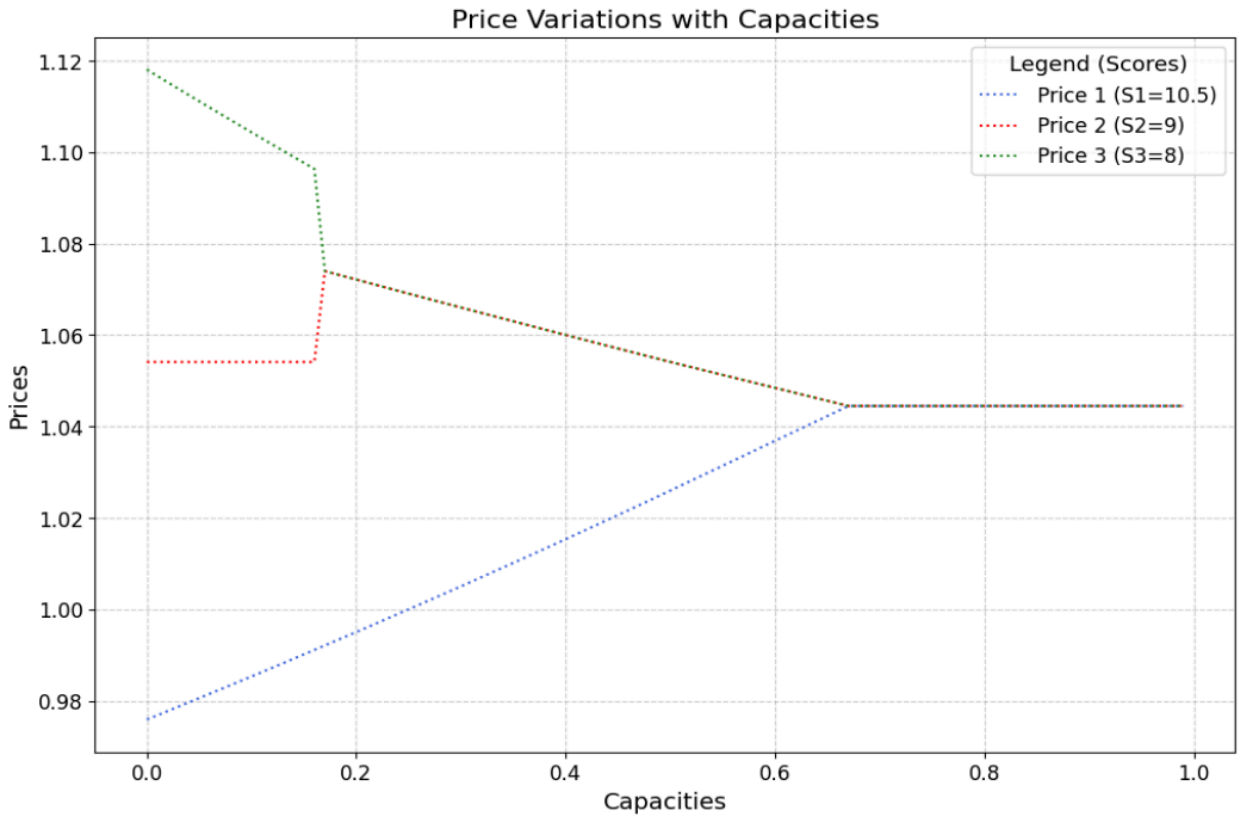

Figure 5 – Markets behaviours after an increase in spread between supply 1 and supply 2.

Notice that the various equilibrium prices slightly change in each plot, that is because for every supply level setting, the optimization calculus for finding the equilibrium price changes as well.

Appendix – *three_island*

In this appendix we report the code relative to *three_island* function.

Input

```
def three_islands(d_level1, d_el1, s_level1, s_el1, d_level2, d_el2, s_level2, s_el2, d_level3, d_el3, s_level3, s_el3, cap12, cap13, cap23):
    whichcase = np.zeros(13) #stores the "feasibility" of the solution given for each single scenario
    prices1 = np.zeros(13)
    prices2 = np.zeros(13)
    prices3 = np.zeros(13)
```

Figure 6 - Inputs of three_island function; creation of 4 element to store prices and whichcase indicator.

We already explained the inputs that *three_island* needs in the **Chapter 4**.
The elements *prices_* store the price level for the scenarios, while *whichcase* stores a binary result for each scenario: 1 if the solution (i.e. the equilibrium prices) satisfies the constraints for the scenario, 0 otherwise.

Now we are going to present how we code each market structure, taking the first scenario from each case as an example.
Note that the order of the cases is switched with respect to the one presented in the **Chapter 3** because we needed to code the function starting from the 3-island unified market case.

Case: One unified market

```
# Scenario 0: 1 unified market
prices1[0] = prices2[0] = prices3[0] = opt.bisect(
    common_excess_supply_three_island, 1e-8, 10,
    args=(d_level1, d_el1, s_level1, s_el1, d_level2, d_el2, s_level2, s_el2, d_level3, d_el3, s_level3, s_el3))

# Calculate excess supply for each island at the initial price
ES1 = excess_supply(prices1[0], d_level1, d_el1, s_level1, s_el1, 0)
ES2 = excess_supply(prices2[0], d_level2, d_el2, s_level2, s_el2, 0)
ES3 = excess_supply(prices3[0], d_level3, d_el3, s_level3, s_el3, 0)

# Check if initial excess supplies are within transmission capacities
if abs(ES1) < cap12 + cap13 and abs(ES2) < cap12 + cap23 and abs(ES3) < cap13 + cap23:
    whichcase[0] = 1
    return whichcase, prices1, prices2, prices3
```

Figure 7 - Case: One Unified Market.

The prices are set all equal and computed using bisect optimization algorithm on *common_excess_supply_three_island* (mentioned in **Chapter 4**). Excesses of supply are computed using the common price found above.
whichcase stores 1 only if the absolute value of each ES is less than the sum of the capacities connecting the market with the others.

Case: One separate market, one integrated market

```
# Scenario 1: 2 markets, p1 > p23
prices1[1] = opt.bisect(excess_supply, 1e-8, 10, args=(d_level1, d_el1, s_level1, s_el1, +cap12 + cap13))
prices2[1] = prices3[1] = opt.bisect(
    common_excess_supply_two, 1e-8, 10,
    args=(d_level2, d_el2, s_level2, s_el2, d_level3, d_el3, s_level3, s_el3, -cap12 - cap13))

ES2 = excess_supply(prices2[1], d_level2, d_el2, s_level2, s_el2, +cap12)
ES3 = excess_supply(prices3[1], d_level3, d_el3, s_level3, s_el3, +cap13)

if abs(ES2) < cap23 and prices1[1] > prices2[1]:
    whichcase[1] = 1
    return whichcase, prices1, prices2, prices3

# Scenario 2: 2 markets, p1 < p23
prices1[2] = opt.bisect(excess_supply, 1e-8, 10, args=(d_level1, d_el1, s_level1, s_el1, -cap12 - cap13))
prices2[2] = prices3[2] = opt.bisect(
    common_excess_supply_two, 1e-8, 10,
    args=(d_level2, d_el2, s_level2, s_el2, d_level3, d_el3, s_level3, s_el3, +cap12 + cap13))

ES2 = excess_supply(prices2[2], d_level2, d_el2, s_level2, s_el2, -cap12)
ES3 = excess_supply(prices3[2], d_level3, d_el3, s_level3, s_el3, -cap13)

if abs(ES2) < cap23 and prices1[2] < prices2[2]:
    whichcase[2] = 1
    return whichcase, prices1, prices2, prices3
```

Figure 8 - Case: One separated market, one integrated market.

In Scenario 1 prices are computed separately for Market 1 and Market 2-3, using *opt.bisect* on *excess_supply* and *common_excess_supply_two* respectively. In *prices1* computation, we need to add the capacity brought from cable 12 and 13; meanwhile, we need to subtract the same values from the optimization for *prices2* and *prices3*.
In the end, we need to ensure that the absolute value of ES2 is lower than capacity constraint on cable 23 and that *prices1* is in fact bigger than *prices2* and *prices3*.
The same applies for *Scenario 2*, switching the signs of the cap and the sign of the inequality on the prices.

Case: Three separated markets

```
# Scenario number 7: 3 markets, p1 > p2 > p3
prices1[7] = opt.bisect(excess_supply, 1e-8, 10, args=(d_level1, d_el1, s_level1, s_el1, cap12 + cap13))
prices2[7] = opt.bisect(excess_supply, 1e-8, 10, args=(d_level2, d_el2, s_level2, s_el2, -cap12 + cap13))
prices3[7] = opt.bisect(excess_supply, 1e-8, 10, args=(d_level3, d_el3, s_level3, s_el3, -cap13 -cap12))

ES2 = excess_supply(prices2[7], d_level2, d_el2, s_level2, s_el2, 0)
ES3 = excess_supply(prices3[7], d_level3, d_el3, s_level3, s_el3, 0)

if prices3[7] < prices2[7] and prices2[7] < prices1[7]:
    whichcase[7] = 1
    return whichcase, prices1, prices2, prices3
```

Figure 9 - Case: Three separated market.

Each price is computed using *opt.bisect* on *excess_supply* adding the two caps to the market with the bigger price and subtracting the two caps from the market with the lower price. For what concerns the market in between, energy arrives from Market 3 and streams to Market 1.
To ensure the feasibility of the scenario, we need to check if *prices1* is greater than *prices2* and *prices2* is greater than *prices3* (using the equilibrium prices computed above).

Return

```
return np.zeros(13), np.zeros(13), np.zeros(13), np.zeros(13)  # returning dummy arrays instead of a string
```

Figure 10 – Return.

In the end the function returns 4 arrays with prices and *whichcase*.