# HW4 Report

Kangqi Yu 121090735

November 24, 2023

## 1 Introduction

In this report, I am trying to detect the existence of arbitrage opportunity by trading the five currencies provided in the data table using linear programming. In detail, I am trying to detect some arbitrages opportunity in two, three, four and five currencies respectively. The result is there is no pair of two arbitrage, 3/10 of pairs of three have arbitrage opportunity, 4/5 of pairs of four have arbitrage opportunity, and pair of five has arbitrage opportunity. The more members in an arbitrage pool, the higher probability to have an arbitrage opportunity.

## 2 Data

There are two axises for the data table, one is the transfer out currency, the other is the transfer in currency. There is a float in the intersection of them. The data table is shown below:

| | To: US Dollar | Pound | FFranc | D-Mark | Yen |
|---|---|---|---|---|---|
| From: US Dollar | | 0.6390 | 5.3712 | 1.5712 | 98.8901 |
| Pound | 1.5648 | | 8.4304 | 2.4590 | 154.7733 |
| FFranc | 0.1856 | 0.1186 | | 0.2921 | 18.4122 |
| D-Mark | 0.6361 | 0.4063 | 3.4233 | | 62.9400 |
| Yen | 0.01011 | 0.00645 | 0.05431 | 0.01588 | |

Figure 1: Data Table

# 3 Model

## 3.1 Optimization Problem

- **Decision**: a set of spot currency transactions.
- **Objective**: maximize the net amount of first currency.
- **Constraints**: The final net amount of each currency must be nonnegative.

## 3.2 Mathematical Setting

Assume there is n currencies in this model.

- **Index**: let $i = 1, 2, ..., n$ represent n currencies.
- **Transactions**: let $x_{ij}$ represent the amount of currency $i$ transfered to currency $j$.
- **Exchange rate**: let $r_{ij}$ represent the exchange rate from currency $i$ to currency $j$.
- **Final net**: $f_k$ = final net amount of currency $k$.

## 3.3 Optimization Formulation

$$f_i = \sum_{j \neq 1} r_{ji} x_{ji} - \sum_{j \neq 1} x_{ij}, i = 1, 2, ..., n$$

$$\max \{f_1\}$$

$$s.t. f_i, x_{ij} \geq 0, i, j = 1, 2, ..., n, f_1 \leq 1$$

If $f_1^* = 1$, there is an arbitrage opportunity, otherwise there is no arbitrage opportunity.

More interesting thing is that we already have known that $r_{ii} = 1$, so the above optimization formulation can be simplified as:

$$f_i = \sum r_{ji} x_{ji} - \sum x_{ij}, i = 1, 2, ..., n$$

$$\max \{f_1\}$$

$$s.t. f_i, x_{ij} \geq 0, i, j = 1, 2, ..., n, f_1 \leq 1$$

### 3.4 Method of solution

In this report, I use cvx in MATLAB to solve the optimization problem. The code is in the Appendix.

## 4 Empirical results

I denote (US, Pound, FFranc, D-Mark, Yen) to be (1, 2, 3, 4, 5), by solving above linear programming problem to see the optimal value. If it is 1, we can get that there is an arbitrage opportunity. If not, there is no arbitrage opportunity. The results are shown below:

- **Pair of two**: There is no arbitrage opportunity.
- **Pair of three**: (2, 3, 4), (2, 3, 5), (3, 4, 5) have arbitrage opportunity.
- **Pair of four**: (2 ,3, 4, 5), (1, 3, 4, 5), (1, 2, 3, 5), (1, 2, 3, 4) have arbitrage opportunity.
- **Pair of five**: (1, 2, 3, 4, 5) has arbitrage opportunity.

## 5 Discussions

As we see from the result, the rate of having arbitrage opportunity in the pair of two is 0, the rate of having arbitrage opportunity in the pair of three is 3/10, the rate of having arbitrage opportunity in the pair of four is 4/5, the rate of having arbitrage opportunity in the pair of five is 1. The above results show the more currencies wait for choosing, the high probability of having arbitrage opportunity, which is consitent to the less constraints in a optimization problem, the more feasible solutions are there. But, this model also has a limitation that we do not consider the transaction cost in this model, so these arbitrage opportunity may not be real arbitrage opportunity in the reality. So, we can take some transaction cost into consideration to refine the model.

## 6 Appendix

Pair of two:

```
E = [1 0.6390 5.3712 1.5712 98.8901;
1.5648 1 8.4304 2.4590 154.7733;
```

```
0.1856  0.1186  1  0.2921  18.4122;
0.6361  0.4063  3.4233  1  62.9400;
0.01011  0.00645  0.05431  0.01588  1];

for i = 1:5
    for j = i + 1:5
        cvx_begin
            cvx_quiet(true);
            variables x(2, 2)
            maximize sum(E([i, j], i).*x(:, 1)) - sum(x(1, :))
            subject to
                sum(E([i, j], i).*x(:, 1)) - sum(x(1, :)) <=
                    1;
                sum(E([i, j], i).*x(:, 1)) - sum(x(1, :)) >=
                    0;
                sum(E([i, j], j).*x(:, 2)) - sum(x(2, :)) >=
                    0;
                x >= 0;
        cvx_end
        comb = [i j];
        disp(comb)
        disp(cvx_optval);
    end
end
```

Pair of three:

```
E = [1  0.6390  5.3712  1.5712  98.8901;
1.5648  1  8.4304  2.4590  154.7733;
0.1856  0.1186  1  0.2921  18.4122;
0.6361  0.4063  3.4233  1  62.9400;
0.01011  0.00645  0.05431  0.01588  1];
```

```
for i = 1:5
    for j = (i + 1):5
        for m = (j + 1):5
            cvx_begin
                cvx_quiet(true);
                variables x(3, 3)
                maximize sum(E([i, j, m], i).*x(:, 1)) -
                   sum(x(1, :))
                subject to
                    sum(E([i, j, m], i).*x(:, 1)) - sum(x(1,
                       :)) <= 1;
                    sum(E([i, j, m], i).*x(:, 1)) - sum(x(1,
                       :)) >= 0;
                    sum(E([i, j, m], j).*x(:, 2)) - sum(x(2,
                       :)) >= 0;
                    sum(E([i, j, m], m).*x(:, 3)) - sum(x(3,
                       :)) >= 0;
                    x >= 0;
            cvx_end
            comb = [i j, m];
            disp(comb);
            disp(cvx_optval);
        end
    end
end
```

Pair of four:

```
E = [1 0.6390 5.3712 1.5712 98.8901;
1.5648 1 8.4304 2.4590 154.7733;
0.1856 0.1186 1 0.2921 18.4122;
```

```
0.6361  0.4063  3.4233  1  62.9400;
0.01011  0.00645  0.05431  0.01588  1];

for i = 1:5
    for j = (i + 1):5
        for m = (j + 1):5
            for n = (m + 1):5
                cvx_begin
                    cvx_quiet(true);
                    variables x(4, 4)
                    maximize sum(E([i, j, m, n], i).*x(:, 1))
                      - sum(x(1, :))
                    subject to
                        sum(E([i, j, m, n], i).*x(:, 1)) -
                          sum(x(1, :)) <= 1;
                        sum(E([i, j, m, n], i).*x(:, 1)) -
                          sum(x(1, :)) >= 0;
                        sum(E([i, j, m, n], j).*x(:, 2)) -
                          sum(x(2, :)) >= 0;
                        sum(E([i, j, m, n], m).*x(:, 3)) -
                          sum(x(3, :)) >= 0;
                        sum(E([i, j, m, n], n).*x(:, 4)) -
                          sum(x(4, :)) >= 0;
                        x >= 0;
                cvx_end
                comb = [i j, m, n];
                disp(comb);
                disp(cvx_optval);
            end
        end
    end
```

```
end
```

Pair of five:

```
E = [1 0.6390 5.3712 1.5712 98.8901;
1.5648 1 8.4304 2.4590 154.7733;
0.1856 0.1186 1 0.2921 18.4122;
0.6361 0.4063 3.4233 1 62.9400;
0.01011 0.00645 0.05431 0.01588 1];

cvx_begin
cvx_quiet(true);
variables x(5, 5)
maximize sum(E(:, 1).*x(:, 1)) - sum(x(1, :))
subject to
    sum(E(:, 1).*x(:, 1)) - sum(x(1, :)) <= 1;
    sum(E(:, 1).*x(:, 1)) - sum(x(1, :)) >= 0;
    sum(E(:, 2).*x(:, 2)) - sum(x(2, :)) >= 0;
    sum(E(:, 3).*x(:, 3)) - sum(x(3, :)) >= 0;
    sum(E(:, 4).*x(:, 4)) - sum(x(4, :)) >= 0;
    sum(E(:, 5).*x(:, 5)) - sum(x(5, :)) >= 0;
    x >= 0;
cvx_end
disp(cvx_optval);
```