MIP clearing for FX

Maciej Reich,Anthony Bordier

September 22, 2020

Contents

1 Introduction

This document outlines the changes made to the MIP matching engine to support FX clearing. The approach builds on top of the regular MIP engine, but modifies it quite deeply.

The overall idea is to pick a benchmark currency (USD throughout here, but could be anything, including a made-up token or a basket of other currencies), replicate all currency crosses to USD/X crosses and apply clearing. Internally, the solver tracks and maximises USD values of orders. Additional logic is added to make sure the USD quantities do not exceed the initial desired currency quantities, expressed in asset currency. Some approximations are applied, but the method is guaranteed never to overfill orders; some very small underfills may occur though.

2 Basic approach outline

First, it is actually easy to express the problem in current MIP if an extra simplification is made: all orders are entered with USD quantities. For an order to buy a cross X/Y, we replicate this into orders to sell USD/X and buy USD/Y, in quantity equal to the USD quantity of the original order.

The condition of the order would be:

$$p(USD/Y) \div p(USD/X) \leq \text{limit price}$$

Taking natural logarithm of both sides yields

$$\log\left(p(USD/Y)\right) - \log\left(p(USD/X)\right) \leq \log\left(\text{limit price}\right)$$

This is now a linear condition, but replacing price with log(price). However, everything else continues working as expected.

It is important to apply the log transform in the right place of the pipeline:

- Construct orders on original pairs (with USD order quantities)
- Replace prices with their natural logarithm
- Replicate orders to pairs with USD being the asset.
- Run matching
- De-replicate orders to originally-requested tokens
- Convert log-prices to prices via exponentiation

As it turns out, volume maximisation, surplus minimisation and price premium maximisation continue to work as expected.

Final price disambiguation is interesting. At present, we minimise

$$\sum_i \|p_{\text{new}} - p_{\text{old}}\|_2^2$$

If we replace the prices with log(price) we end up with

$$\sum_i \|\log(p_{\text{new}}) - \log(p_{\text{old}})\|_2^2 = \sum_i \|\log(p_{\text{new}}/p_{\text{old}})\|_2^2$$

which also has a natural interpretation, in minimising the percentage difference between the old and new prices. Therefore this step can also remain as-is.

This approach suffers from the drawback that amounts are entered as USD. If a user wants to buy 1 BTC, he would need to enter a USD-equivalent amount. If the BTC price moves during the auction and the order fills fully, the user could end up with more or less than 1 BTC, depending on the price fluctuation.

3 Allowing asset currency-denominated orders

It is possible to enable users to enter order quantities in asset currency, and also guarantee no overfilling. The method is approximate: it never allows for overfilling, but it does make it possible to underfill very slightly, by e.g. up to 0.5% in extreme circumstances. This can possibly be further rectified as a post-processing step.

Suppose we have an order to buy some quantity of BTC/USD. We then have

$$\text{Quantity (BTC)} = \text{Quantity (USD)} \div p(\text{BTC/USD})$$

Assuming Quantity (BTC) is fixed and Quantity (USD) is the solution variable, we can expand as follows:

$$\begin{aligned}\text{Quantity (USD)} &\leq \text{Quantity (BTC)} \div p(\text{USD/BTC}) \\ &= \text{Quantity (BTC)} \times \exp\{-\log(p(\text{USD/BTC}))\}\end{aligned}$$

This is exact so far. The log/exp transformations are included, since the solver does not track the prices, and only their logarithms.

While we cannot exponentiate inside the MIP solver, we can do a 1st order Taylor expansion around $\log(p(\text{BTC/USD}_0))$, where the subscript $_0$ means throughout a value known before the start of the auction (e.g. last price of the cross, or best guess for the new price). This gives

$$\text{Quantity (USD)} \leq \text{Quantity (BTC)} \times p(\text{BTC/USD})_0 \times (1 - \log(p(\text{USD/BTC})) + \log(p(\text{USD/BTC})_0)$$

These are now all quantities and operations known to the MIP solver. In essence we approximated division as follows:

$$1/x = \exp(-\log(x)) \sim 1/x_0\,(1 - \log(x) + \log(x_0) + \cdots)$$

The question would be how good an approximation this this.

Conservatism First of all, the approximation is in fact conservative. This is good news, as it can never lead to a situation where an order overfills. We have in fact

$$1/x_0\,(1 - \log(x) + \log(x_0)) \leq 1/x$$

so this approximation could lead to *underfilling*, but never overfilling. Since overfilling would be unacceptable, this is perhaps OK.

Approximation quality Rewriting the above approximation as

$$1/x = 1/x_0(1 - \log(x/x_0) + \cdots)$$

the quadratic error is in fact, from Taylor expansion,

$$0.5\log(x/x_0)^2 \approx 0.5\left(\frac{x}{x_0} - 1\right)^2$$

where the term in the bracket on the right is the percentage difference between x and x_0.

Thus, for example, assuming the price of a USD/X cross moved by 1% during an auction, the error in this approximation will be $0.5 \times (10^{-2})^2 = 5 \times 10^{-5}$. For a very large move within one batch of 10%, that translates to $0.5 \times (10^{-1})^2 = 5 \times 10^{-3} = 0.5\%$. Given in John's studies the improvement from MIP was at least of order 10–30%, this is not a big sacrifice, even in case of extreme currency movements.

If more accuracy is needed, two approaches are possible:

- MIP can be re-ran, replacing previous batch price with the values from the first MIP run. This way, the change in currency will be much smaller, and the approximation much better.
- Alternatively, the output of the MIP problem could be used as the starting point to solve the full non-linear optimisation problem. The latter would generally be a more faithful representation, but would normally struggle to find good solutions. However, by feeding it the MIP solution as the starting point, it would be possible to achieve very good convergence, and likely very quickly too.

In theory, this approximation, in very anomalous situations, could lead to unoptimal crosses. For example, suppose optimally the price should make a large move to include a small order in the possible order crosses. However, the large price move would also lead to a larger approximation error in the above derivation, and overall lower volume. This is quite academic though, and should not cause issues in practice. Even then, everyone would get a fair price, and volume would be only a fraction of a percent away from optimal.

Restriction on max price movement It is necessary for 3 reasons to introduce a maximum price movement for atomic instruments (in this case, USD/X pairs):

1. As before, token prices must be bounded for MIP to work. Note in this case, atomic token prices are often actually negative, since we are dealing with log prices. This is fine though. The upper and lower bounds can also be negative, correspondingly.
2. Further, internally, the MIP problem solves for USD quantities, with the additional constraint applied as a correction. But the correction quality deteriorates for larger FX moves, therefore the move must be limited. As discussed above, for a 10% move, the approximation error is only about 0.5%, but for e.g. a 50% FX move, the error would be 25%. 10% seems a good compromise.
3. Finally, for extremely large moves, the approximation actually would break down. If log(new_price / old_price) is more than 1 (corresponding to a 270% move), the quantity bound actually becomes negative, making the problem infeasible (as well as nonsensical).

4 Code

Code implementing these changes is in the GitHub repo on the `fx_pair_replication_native_ccy` branch. Tests demonstrating its use can be found in file `$repo/tests/test_mip_solver_fx.py`, in particular `test_clear_orders_mip_multi_fx`. It should be easy to adapt the existing csv parsing/saving mechanism to also support csv test cases, again due to the shortness of time I didn't do it.

Functions translating from the raw output of the MIP problem to the individual order fills were rewritten, as the existing ones got very complicated, mostly with features not needed for currency matching. I did not have the time to write a direct test for it, but it is implicitly tested in `test_random_fx_orders` routine, which constructs random orderbooks and checks whether the results make sense (e.g. all atomic tokens clear to 0 etc.).

5 Miscellaneous details

5.1 Accuracy and tick size

Since now the solver tracks log prices, not outright prices, solver accuracy applies to log prices instead. This is equivalent to a fixed *relative* error on prices. For example, a tolerance of 1e-4 (very achievable) for a 10,000 BTC price leads to an accuracy of 1 USD on price — which seems reasonable. Raising accuracy to 1e-6 (tenuous but with care should be fine) leads to an accuracy of 0.01 USD on the same BTC price.

At the same time "cheap" crosses, such as for example ETH/BTC (currently 0.0205) would have an accuracy of about 0.000002 BTC, and 1e-6 an accuracy of 0.00000002. Due to shortness of time, I haven't implemented an adaptive rounding scheme, but this should be an easy project for any quant who takes over my work.

5.2 Integration of orders, replication, pre-processing and MIP solver

In previous work, the MIP solver was more modular. Replication was purely a pre-processing step, for example. The changes described in this document require a much close integration of the different parts. For example, the MIP solver needs to understand what the different tokens are, and how to perform FX conversions (as opposed to treating the tokens as fungible, opaque objects).

5.3 Fill distribution

For simplicity, time priority was not included in fill distribution. It should be trivial to include, but requires careful attention, for which I ran out of time.

5.4 Final price disambiguation

Final price disambiguation now depends on quantities filled, as well as knowing which orders filled or did not fill. This means that quantities need to be passed on to the final disambiguation function.

This is because now quantities are constrained by a term including the log(price) terms. It is possible that for a particular match to occur, the price needs to occupy a particular part of the solution space.

6 Extension to FX swap

6.1 Problem intuition

By definition, the MIP only deals with constraints such as inequalities on a linear combination of variables. In the FX case, the price constraint inequalities structurally apply to products or ratios of FX spot or forward prices, in order to define constraints on cross FX prices, such as:

$$L \leq S_{btc/eth} \leq H \iff L \leq \frac{S_{btc/usd}}{S_{eth/usd}} \leq H$$

where L is a lower (or bid) price, and H is a higher (or ask) price, and S means spot price, for a given currency pair.

Therefore these constraints have to be transformed into inequalities on linear combinations of (possibly a function of) spots or forward. This is done by taking the logs of the LHS and RHS of the initial inequality terms, such as:

$$\log(L) \leq \log(\frac{S_{btc/usd}}{S_{eth/usd}}) \leq \log(H)$$

.

It transforms a multiplicative problem in the native space into an additive problem in the log space, such as:

$$\log(L) \leq \log(S_{btc/usd}) - \log(S_{eth/usd}) \leq \log(H)$$

.

As a result, the FX MIP operates exclusively on linear systems of logarithms of price variables, NOT on linear systems of price variables.

Which means that if we have to handle an inequality on a linear combination of 2 variables in the native space, such as:

$$L \leq F - S \leq H$$

we can't do that directly, as there exist no variables α, β, γ and δ such that

$$L \leq F - S \leq H \iff \alpha \cdot \log(L) \leq \beta \cdot \log(F) - \gamma \cdot \log(S) \leq \delta \cdot \log(H)$$

.

If we define F to be the forward price and S to be the spot price for a given currency pair, then:

$$Swap = F - S$$

is by definition a spot starting swap for this currency pair. From what we've seen above, the MIP can't directly handle constraints on the swap price as a linear combination of a forward and a spot.

However, we can rewrite the constraint:

$$L \leq F - S \leq H$$

as:

$$L + S \leq F \leq H + S$$

Which is:

$$\begin{cases} F \leq H + S \\ L + S \leq F \end{cases}$$

And we can rewrite the first inequality:

$$F \leq H + S$$

Which is equivalent to:

$$\exists s \in \mathbb{R} \colon \begin{cases} F \leq H + s \\ s \leq S \end{cases}$$

And composing with the log function:

$$\exists s \in \mathbb{R} \colon \begin{cases} \log(F) \leq \log(H + s) \\ \log(s) \leq \log(S) \end{cases}$$

Similarly, we can rewrite the second inequality:

$$L + S \leq F$$

As:

$$\exists s \in \mathbb{R} \colon \begin{cases} L + s \leq F \\ S \leq s \end{cases}$$

And composing with the log function:

$$\exists s \in \mathbb{R} \colon \begin{cases} \log(L + s) \leq \log(F) \\ \log(S) \leq \log(s) \end{cases}$$

Hence, in both cases, we have replaced an inequation on 2 native variables by a system of 2 inequations on 1 log variable each, which the MIP can handle, as we are going to see in the next section.

6.2 Problem formalisation

Let's assume we have a set of decreasing bid prices $L_i, i \in \{0, m\}$ (resp ask price $U_i, i \in \{0, n\}$) define some interval for a given swap, so that:

$$L_0 \leq L_1 ... \leq L_m \leq Swap \leq U_0 \leq U_1 ... \leq U_n$$

Which is, from the definition of the swap price:

$$L_0 \leq L_1 ... \leq L_m \leq F - S \leq U_0 \leq U_1 ... \leq U_n$$

Which is equivalent to:

$$L_0 + S \leq L_1 + S ... \leq L_m + S \leq F \leq U_0 + S \leq U_1 + S ... \leq U_n + S$$

We've seen in the previous section that we could rewrite each of the inequalities above on 2 native variables as a system of 2 inequalities on 1 log-variable. However the parameterisation of this system of 2 equations with the real index number s (which is a constant in each of the systems) is by definition continuous, and for the sake of integrating it in the MIP constraints, we need to discretize it with a finite set of $s_k, k \in \{0, K\}$.

Let's assume that $S \in [S_{min}, S_{max}] \equiv \bigsqcup_{k=0}^{K-1} [s_k, s_{k+1}]$, where $s_0 = S_{min}$ and re $s_K = S_{max}$

6.2.1 Problem formalisation - higher boundary

From here, we can discretised equation accordingly:

$$\exists k \in \{0, K-1\} \colon \left\{ \begin{array}{l} \log(F) \leq \log(H_j + s_k) \\ \log(s_k) \leq \log(S) \end{array} \right.$$

Which is equivalent, in a MIP disjunctive formulation, to the following system:

1. $\left\{ \begin{array}{l} \log(F) - \log(H_j + s_k) + B_{ij} \cdot (1 - b_{ijk}) \leq 0 \\ \log(s_k) - \log(S) + D_{ij} \cdot (1 - b_{ijk}) \leq 0 \end{array} \right. \iff \left\{ \begin{array}{l} F \leq H_j + s_k \\ s_k \leq S \end{array} \right.$

2. $\left\{ \begin{array}{l} \log(F) - \log(H_j + s_k) + A_{ij} \cdot (1 - c_{ijk}) \geq 0 \\ \log(s_k) - \log(S) + C_{ij} \cdot (1 - c_{ijk}) \geq 0 \end{array} \right. \iff \left\{ \begin{array}{l} F \geq H_j + s_k \\ s_k \geq S \end{array} \right.$

3. $\left\{ \begin{array}{l} \log(F) - \log(H_j + s_k) + B_{ij} \cdot (1 - d_{ijk}) \leq 0 \\ \log(s_k) - \log(S) + C_{ij} \cdot (1 - d_{ijk}) \geq 0 \end{array} \right. \iff \left\{ \begin{array}{l} F \leq H_j + s_k \\ s_k \geq S \end{array} \right.$

4. $\left\{ \begin{array}{l} \log(F) - \log(H_j + s_k) + A_{ij} \cdot (1 - e_{ijk}) \geq 0 \\ \log(s_k) - \log(S) + D_{ij} \cdot (1 - e_{ijk}) \leq 0 \end{array} \right. \iff \left\{ \begin{array}{l} F \geq H_j + s_k \\ s_k \leq S \end{array} \right.$

5. $b_{ijk} + c_{ijk} + d_{ijk} + e_{ijk} = 1 \iff$ only one of the cases (1), (2), (3) or (4) is realised

6. $\left\{\begin{array}{l} \forall k, b_{ij} \geq b_{ijk} \\ \forall k, b_{ij} \leq \sum\limits_{k=0}^{K-1} b_{ijk} \\ \forall k, b_{ij} \leq 1 \end{array}\right. \iff b_{ij} = \max\limits_{k}(b_{ijk}) \iff \exists\, b_{ijk}$ such that (1) is verified

6.2.2 Problem formalisation - lower boundary

From here, we can discretised equation accordingly:

$$\exists k \in \{0, K-1\}\colon \left\{\begin{array}{l} \log(L_j + s_{k+1}) \leq \log(F) \\ \log(S) \leq \log(s_{k+1}) \end{array}\right.$$

Which is equivalent, in a MIP disjunctive formulation, to the following system:

1. $\left\{\begin{array}{l} -\log(F) + \log(L_j + s_{k+1}) + B_{ij} \cdot (1 - b_{ijk}) \leq 0 \\ -\log(s_{k+1}) + \log(S) + D_{ij} \cdot (1 - b_{ijk}) \leq 0 \end{array}\right. \iff \left\{\begin{array}{l} F \geq L_j + s_{k+1} \\ s_{k+1} \geq S \end{array}\right.$

2. $\left\{\begin{array}{l} -\log(F) + \log(L_j + s_{k+1}) + A_{ij} \cdot (1 - c_{ijk}) \geq 0 \\ -\log(s_{k+1}) + \log(S) + C_{ij} \cdot (1 - c_{ijk}) \geq 0 \end{array}\right. \iff \left\{\begin{array}{l} F \leq L_j + s_{k+1} \\ s_{k+1} \leq S \end{array}\right.$

3. $\left\{\begin{array}{l} -\log(F) + \log(L_j + s_{k+1}) + B_{ij} \cdot (1 - d_{ijk}) \leq 0 \\ -\log(s_{k+1}) + \log(S) + C_{ij} \cdot (1 - d_{ijk}) \geq 0 \end{array}\right. \iff \left\{\begin{array}{l} F \geq L_j + s_{k+1} \\ s_{k+1} \leq S \end{array}\right.$

4. $\left\{\begin{array}{l} -\log(F) + \log(L_j + s_{k+1}) + A_{ij} \cdot (1 - e_{ijk}) \geq 0 \\ -\log(s_{k+1}) + \log(S) + D_{ij} \cdot (1 - e_{ijk}) \leq 0 \end{array}\right. \iff \left\{\begin{array}{l} F \leq L_j + s_{k+1} \\ s_{k+1} \geq S \end{array}\right.$

5. $b_{ijk} + c_{ijk} + d_{ijk} + e_{ijk} = 1 \iff$ only one of the cases (1), (2), (3) or (4) is realised

6. $\left\{\begin{array}{l} \forall k, b_{ij} \geq b_{ijk} \\ \forall k, b_{ij} \leq \sum\limits_{k=0}^{K-1} b_{ijk} \\ \forall k, b_{ij} \leq 1 \end{array}\right. \iff b_{ij} = \max\limits_{k}(b_{ijk}) \iff \exists\, b_{ijk}$ such that (1) is verified