Piecewise Flat Interpolation of Overlapping Commodity Forward Prices

Jake C. Fowler

THIS DOCUMENT IS CURRENTLY WORK IN PROGRESS

Abstract

This paper presents an algorithm for the piecewise flat interpolation of the linear commodity instruments: forwards, swaps and futures. First the problem is formulated as a system of linear equations. This system is not guaranteed to have a unique solution, so a pseudoinverse is used to find find the least squares solution. Next a case where this solution does not give the desired results is presented. A modification of the original algorithm is described which adds a vector from the nullspace of the linear system matrix to the least squares solution. This nullspace vector is chosen to minimise the Euclidean distance to a target vector which is itself derived from market prices. Finally there is a discussion on future research into how this algorithm can be improved.

1 Introduction and Terminology

This document presents an algorithm for the piecewise flat interpolation of commodity forwards, futures and swap contracts. It is applicable for linear commodity contracts contingent on either a continuous delivery of physical commodity, financial payoff based on the average of an index throughout the contract delivery period. The rest of this document will refer to just *forward* prices and curves, even though the same methodology can be applied to futures and swap instruments as well. Also the document refers to the *delivery period* of the contracts, in place of *delivery or fixing period* where *fixing period* is the set of time periods from which the average price of an index is used to calculate the payoff of a swap contract.

2 Bootstrapping Definition

The process of bootstrapping is to transform a set of forward prices into a piecewise flat forward curve consistent with these prices. In the typical case, the delivery periods of the input forward contracts will be overlapping, so boostrapping can been seen as an algorithm for producing a set of forward prices for contiguous delivery periods, consistent with the input forward prices. In this

wasy, bootstrapping should in the most general case be seen as an interpolation method.

3 Uses For Bootstrapping

One example of the use of a bootstrapped curve is for it to be loaded into an ETRM (Energy Trading Risk Management) system for the valuation of a portfolio of vanilla trades. The bootstrapping creates an unambiguous price for each delivery period in the lowest granularity which a commodity trades, which has practical benefits over just using the raw forward prices of traded contracts.

A piecewise flat bootstrapped curve can also be used for the conservative valuation of a physical asset with embedded optionality.

4 No-Arbitrage Forward Price Condition

This section describes the no-arbitrage relationship between the set of prices of forward contracts for a consecutive delivery period and price of a single lower granularity contract which spans the whole delivery period of these. It is this relationship which forms the basis of the boostrapping algorithm.

Denote:

- F as a forward price of the *big* contract. The notation does not contain the concept of the time when this is observed, but it can just be assumed that the current valuation date is earlier than t_s the start of the delivery period.
- f_i as the forward price for the ith *small* contract a higher granularity period than F has, with $i = 1 \dots n$
- D_i as the discount factor from the settlement date of contract f_i to the valuation date. We assume that the delivery period of f_i is short enough for the whole contract to be settled on a single date.
- w_i is the weighting of the delivery period of contract i, relative to the delivery period that forward price F corresponds to.

To show the no-arbitrage relationship, construct a portfolio which is long one unit of the *big* contract and short w_i units of each *small* contract. This portfolio is essentially flat, does not require and upfront cash, and so should have zero PV. If the PV were positive then a long positive would present an arbitrage, as would a negative PV presents an arbigrate from taking a short position in the portfolio. Expressing the PV as the discounted cash flow from take the portfolio to settlement:

$$\sum_{i=1}^{n} (F - f_i) w_i D_i = 0 \tag{1}$$

Which can be rearranged to:

$$F = \frac{\sum_{i=1}^{n} f_i w_i D_i}{\sum_{i=1}^{n} w_i D_i} \tag{2}$$

This is the no-arbigrate relationship between the *big* contract and it's associated *small* contracts, and can be seen as a weighted average.

5 Bootstrapping Algorithm

5.1 Shaping The Forward Curve

The application of shaping to a forward curve is typically done with power and gas prices where there is strong seasonality, the forward prices are at a too low granularity to show this seasonality, and a spline algorithm cannot be trusted to give the correct shape.

In this paper the application of shaping is done in two different ways: with spreads and ratios. In both cases the application of shaping can be expressed as a linear equations and appended to the same linear system which describes the forward price constraints. In this way the solution to the linear system can jointly boostrap the forward prices and apply shaping.

5.2 Linear System

The above forward price and shaping constraints can be written in matrix form as:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{3}$$

Here element i of vector $\mathbf{x}$ equals f_i, and the joint bootstrapping and shaping problem involves solving for $\mathbf{x}$. However, it is unlikely that the above system can be solved to a unique solution as matrix $\mathbf{A}$ will not necessarily be square. As such, the vector of bootstrapped prices $\hat{\mathbf{x}}$ is found as the least squares solution:

$$\hat{\mathbf{x}} = \mathbf{A}^{+}\mathbf{b} \tag{4}$$

Where matrix $\mathbf{A}^{+}$ is the pseudoinverse (otherwise knowne as the Moore-Penrose inverse) of $\mathbf{A}$.

It is important to note for later discussion in this paper that the above solution will be *least squares* in two different senses. Firstly, should matrix $\mathbf{A}$ have rank less than m, as would be the case where there is some redundancy in the input contracts, $\hat{\mathbf{b}}$ will not necessarily be equal to the vector of input contracts $\mathbf{b}$. It will however but the vector in the column space of $\mathbf{A}$ which has the lowest sum of element-wise squared difference to $\mathbf{b}$.

Secondly, should matrix $\mathbf{A}$ have rank less than it's number of columns, as well typically be the case in practice, the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ will have

infinite solutions, and $\hat{\mathbf{x}}$ will be chosen as the smallest solution, i.e. the one with the lowest some of squared elements. At first it seems that this second *least squares* property is a desirable one, given we are looking for a piecewise flat output curve. Think of the case of bootstrapping a single contract to the forward prices of the periods within it's delivery period. A vector with the same price for all sub-periods as the single input contract price will be also equal to the vector of prices with lowest sum of squared elements. The section below however shows that the least squares solution is not necessarily a good one for the bootstrapping problem, and introduces an adjustment which improves the results.

5.3 The Problem of Partially Overlapping Contracts

The bootstrapping algorithm presented in the previous section has some cases, where it does not perform well. Take an example of two input contracts, the first with delivery covering the first three periods: t_1, t_2, t_3. The second contract has delivery covering two periods: t_3, t_4. If the prices of these two contracts is both 10.0, then the vector of bootstrapped price is calculated as $\begin{bmatrix}8.0 & 8.0 & 14.0 & 6.0\end{bmatrix}^T$. Intuitively this result comes as a surprise, one would expect that the bootstrapped price of all four periods will equal 10.0, the price of both input contracts.

It can easily be verified that these prices average back to the input contracts over just the delivery periods of each input forward. The next check that these results are not erroneous is to calculate the euclidean length of the boostrapped price vector and compare it to the length of the intuitively expeted flat price vector of 4 10's. The actual bootstrapped vector is of length 360.0, which is indeed lower than the length of flat price vector of 400.0 Where matrix $\mathbf{A}$ is underdetermined, and hence has infinite solutions, we know that the bootstrapper will return the single solution with the smallest length. Although the above check does not confirm that there has not been a mistake (i.e. a smaller solution vector exists), it does bring into question whether the criteria of returning the price vector of lowest length is indeed desirable.

The weighting matrix is as follows:

$$\mathbf{A} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 0.5 & 0.5 \end{bmatrix} \tag{5}$$

With pseudo-inverse:

$$\mathbf{A}^+ = \begin{bmatrix} 1.2 & -0.4 \\ 1.2 & -0.4 \\ 0.6 & 0.8 \\ -0.6 & 1.2 \end{bmatrix} \tag{6}$$

To verify that the problem seen above will also be the case with an arbitrary forward price of the input contracts equal to c, we can calculate the derived bootstrapped prices algebraically:

$$\mathbf{x} = \mathbf{A}^{+}\mathbf{b} = \begin{bmatrix} 0.8c \\ 0.8c \\ 1.4c \\ 0.6c \end{bmatrix} \tag{7}$$

This confirms that whatever the prices of the two input forward contracts, the bootstrapped contracts will not be constant and equal to the input contract price.

The above example is contrived and simplified for ease of explaination. Cases seen with real market data being bootstrapped with a similar structure, where two forward contract one another, but neither is completely overlapped by the other. In such cases the input forward prices are not all equal, so the identification of the boostrapper results being problem is not the same as the above. example. The problem seen with real market data is that sections of the bootstrapped curve vary wildly from the input prices.

Such cases are referred to in this rest of this paper as *partially overlapping contracts* and an examples seen in real market data when a contract for a one week delivery partially overlaps with a month.

As such, *partially overlapping contracts* should not merely be thought of as edge cases, and the bootstrapping algorithm needs to be made robust to such cases.

5.4 Introduction of Nullspace Component to Solution

The previous section has shown an example where the least squares solution does not give one we desire, due to a deviation from the prices of the input contracts where partially overlapping contracts are present. To rectify this, we can note that in the problematic scenario, and in the typical case of running the bootstrapping algorithm, the rank of matrix $\mathbf{A}$ will be less than it's number of columns n, the number of boostrapped prices. In such cases there will be infinite solutions, and a general form of these solutions found using the fact that any vector in the nullspace of $\mathbf{A}$ can be added to any particular solution.

If the solution which is closest to $\mathbf{0}$ is not the good, then we need to define a new vector $\mathbf{x}^{target}$, the solution which should be closest to will be choosen. In the case of the problematic example where all input contracts have the same price, then $\mathbf{x}^{target}$ with all elements equal to this price seems like the obvious choice. See the section below for a discussion on choosing $\mathbf{x}^{target}$.

Defined $\mathbf{K}$ as a matrix with columns containing the orthonormal basis of the nullspace of $\mathbf{A}$, and error vector as $\mathbf{e} = \mathbf{x}^{target} - \mathbf{x}$. We want to find a linear combination of the columns of $\mathbf{K}$ which can be added to $\mathbf{x}$ to move it closest to $\mathbf{x}^{target}$. This corresponds to find the least-squares solution of $\mathbf{c}$ to the following system:

$$\mathbf{Kc} = \mathbf{e} \tag{8}$$

The least-squares solution can be found be solving the Normal Equations:

$$\mathbf{K}^T\mathbf{Kc} = \mathbf{K}^T\mathbf{e} \tag{9}$$

As $\mathbf{K}$ is orthonormal $\mathbf{K}^T\mathbf{K} = \mathbf{I}$, hence $\mathbf{c}$ is calculated as:

$$\mathbf{c} = \mathbf{K}^T\mathbf{e} \tag{10}$$

Or, substituting for $\mathbf{e}$ and multiplying out the $\mathbf{K}^T$ term:

$$\mathbf{c} = \mathbf{K}^T\mathbf{x}^{target} - \mathbf{K}^T\mathbf{x} \tag{11}$$

$\mathbf{x}$ is in the Row Space of $\mathbf{A}$ (see Strang (1998)), hence orthogonal to any vector in the nullspace of $\mathbf{A}$, so $\mathbf{K}^T\mathbf{x} = \mathbf{0}$ and the expression for $\mathbf{c}$ can be simplified:

$$\mathbf{c} = \mathbf{K}^T\mathbf{x}^{target} \tag{12}$$

Using this result, our nullspace adjusted boostrapped prices can be written as

$$\mathbf{x}^* = \mathbf{x} + \mathbf{Kc} \tag{13}$$

Substituting for $\mathbf{x}$ and $\mathbf{c}$:

$$\mathbf{x}^* = \mathbf{A}^+\mathbf{b} + \mathbf{K}\mathbf{K}^T\mathbf{x}^{target} \tag{14}$$

From a practical implementation perspective it is worth noting that the nullspace basis can be found from the SVD (Singular Value Decomposition) of $\mathbf{A}$ [1]. Performing an SVD on $\mathbf{A}$ as an intermediate step to calculating $\mathbf{A}^+$, which represents a convenience for the implementation.

5.5 Example of Using Nullspace Component

This section use the input data as TODO, but includes the nullspace component presented in equation xyz as an example and how this solves the problematic case. As before before input contracts have price c, and we will set the target vector as follows:

$$\mathbf{x}^{target} = \begin{bmatrix} c \\ c \\ c \\ c \end{bmatrix} \tag{15}$$

[1] For the SVD $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, the nullspace of $\mathbf{A}$ can be found from the columns of $\mathbf{V}$ corresponding to the column (or row) indices of the zero singular values along the diagonal of $\mathbf{\Sigma}$.

And the nullspace projection matrix (to do check it is this) is:

$$\mathbf{K}\mathbf{K}^T = \begin{bmatrix} 0.6 & -0.4 & -0.2 & 0.2 \\ -0.4 & 0.6 & -0.2 & 0.2 \\ -0.2 & -0.2 & 0.4 & -0.4 \\ 0.2 & 0.2 & -0.4 & 0.4 \end{bmatrix} \tag{16}$$

Substitution this into equation xyz:

$$\begin{aligned} \mathbf{x}^* &= \mathbf{A}^+\mathbf{b} + \mathbf{K}\mathbf{K}^T\mathbf{x}^{target} \\ &= \begin{bmatrix} 0.8c \\ 0.8c \\ 1.4c \\ 0.6c \end{bmatrix} + \begin{bmatrix} 0.6 & -0.4 & -0.2 & 0.2 \\ -0.4 & 0.6 & -0.2 & 0.2 \\ -0.2 & -0.2 & 0.4 & -0.4 \\ 0.2 & 0.2 & -0.4 & 0.4 \end{bmatrix} \begin{bmatrix} c \\ c \\ c \\ c \end{bmatrix} \\ &= \begin{bmatrix} 0.8c \\ 0.8c \\ 1.4c \\ 0.6c \end{bmatrix} + \begin{bmatrix} 0.2c \\ 0.2c \\ -0.4c \\ 0.4c \end{bmatrix} \\ &= \begin{bmatrix} c \\ c \\ c \\ c \end{bmatrix} \end{aligned} \tag{17}$$

We can see from this example that including the nullspace term has the desired effect of resulting in the bootstrapped prices to equal the input forward prices. The above example is of course contrived and does not look like a real use of bootstrapping. However, experiments have been conducted on more realistic inputs to the boostrapper, which have shown that in the case of partially overlapping contracts the inclusion of the nullspace term has improved the results by moving the calculated bootstrapped prices closer to the input forward prices. In these cases, without the nullspace term included the bootstrapped prices deviated wildly from the input contract prices in a way which was clearly not a good interpolation.

5.6 Choosing The Target Vector

In the case where all input forward prices are the same, choosing $\mathbf{x}^{target}$ is easy, but what about the more realistic case when all prices are different? This is somewhat arbitrary, but intuition says that each element of $\mathbf{x}^{target}$ should be a function of the prices of all input contracts which span the delivery period each element of $\mathbf{x}^{target}$ represents the bootstrapped price of. Some experimentation has shown that taking the price of contract with minimum length of delivery period yields good results. In the case where there is more than one contract has the same minimum length, which in practice would be unusual, then the price of the contract which is delivers earliest should be used.

6 Future Work

6.1 Choice of Target Vector

More research into choice of the target vector can be done, including an analysis of both the intuitive and mathematical reasoning behind the logic for calculating $\mathbf{x}^{target}$ presented in this paper, as well as alternative strategies. The results of varying $\mathbf{x}^{target}$ should be analysed with real market data to assess usage in practice.

6.2 Combined Spline Interpolation and Bootstrapping

The whole approach to bootstrapping in this paper should also be compared to algorithms which perform both boostrapping and interpolation, such as Fowler and Benth et al. (2007). In the case of partially overalapping contracts, the inclusion of the nullspace component and target vector presented in this paper is used to impose a shape on the resulting boostrapped prices, but one could argue that using a spline is more suitable due to the smooth structure of a spline giving a more desirable shape. A piecewise flat curve could then be averaged back from the result of the spline, and the use of a bootstapper to create the input to another spline algorithm is no longer needed. Two downside of using a combined bootstrapping and interpolation algorithm can be seen. The first is that introduces additional complication in the case where only a piecewise flat curve is required from the bootstrapping. The second is that a higher order polynomial is required to ensure that the number of of unknowns (polynomial coefficients) is at least as large as the number of constraints in the linear system solved in the spline calculation. This is because the number of piecewise polynomials being solved for can vary when the input contracts are allowed to overlap. The higher the order of the piecewise polynomials, the more tendency it has to oscillate. The author has observed undesirable oscillations when using a 4th order polynomial, as used in Benth et al. (2007), when interpolating natural gas and power forward curves. In practice it is unlikely that a polynomial should change it's slope more than once when interpolating commodity forward curve data.

6.3 Efficient Handling of Collinear Rows

The handling of shaping constraints which are collinear with the forward price constraints (and prior shaping constraints) in an efficient manner is an area which requires more work. For example if the forward prices for Q1-23, Jan-23, and Feb-23 are all present, then the shaping ratio or spread from Q1-2023 to Mar-23 is redundant because the 3 prices already specify the Mar-23 forward price. Mathematically, the addition of the shaping constraint row to matrix $\mathbf{A}$ will not change the rank of $\mathbf{A}$. In such situations the shaping row will never be required and ideally the bootstrapping algorithm should detect this and not

append the shaping constraint row to $\mathbf{A}$. The calculation could involve fully recalculating the rank of $\mathbf{A}$ after each constraint row has been appended, or whether each new row is a linear combination of existing rows, but this could be expensive. There should be a more efficient algorithm which for each new row makes use of the calculations used to check previous rows. Some research into rank-revealing QR factorisations might yield an approach.

Similarly, checking whether forward price constraints are collinear with other forward price constraints could be desired, although the case for discarding redundant forward price constraints is less clear cut than it is for shaping constraints. The algorithm caller would have to be aware of the dependency of the results on the order in which the forward prices and shaping factors are fed in. To make the results deterministic, an ordering first step would be required.

Such handling of constraint collinearity would be required for a truly formulaic bootstrapping algorithm.

References

F. E. Benth, S. Koekebakker, and F. Ollmar. Extracting and applying smooth forward curves from average-based commodity contracts with seasonal variation. *The Journal of Derivatives*, 15:52–66, 2007.

J. C. Fowler. Tension spline algorithm for building commodity forward curves. URL `https://github.com/cmdty/curves/blob/master/docs/tension_spline/tension_spline.pdf`.

Gilbert Strang. *Introduction to Linear Algebra: 2nd Edition.* Wellesley-Cambridge Press, 1998.