LCH.CLEARNET

ForexClear Implied Volatility Smile Interpolation

Methodology Document

Version 1.1

March 2016

Disclaimer

This document is intended solely as information for clearing members of LCH.Clearnet Limited or others who are interested in the services carried out by LCH.Clearnet Limited. It includes a summary of the services provided by LCH.Clearnet Ltd and is not a binding commercial offer or invitation to enter a contract or other legally enforceable agreement. Although all reasonable care has been taken in the preparation of this document LCH.Clearnet Ltd disclaims all liability for the accuracy, sufficiency, completeness of both its contents or the information forming the basis of the document or for any reliance placed on the document by any person whatsoever, and, so far as permitted by law, no responsibility or liability is accepted in relation thereto, and for the avoidance of doubt, no person has any right or remedy (whether by way of a claim for contribution or otherwise) in tort (including negligence), for misrepresentation (whether negligent or otherwise, and whether made prior to, and/or in this undertaking) or otherwise as a result of the information provided in this document. This document may contain statements of opinion of LCH.Clearnet Ltd or its officers or employees and should not be relied upon by clearing members or other any other persons. The information contained in the booklet should not to be construed as a technical specification. All copyright and other intellectual property rights contained within and made available in this document remain vested in LCH.Clearnet Ltd.

Contents

1. Introduction

A component of pricing an FX Option is the interpolation along a volatility surface to retrieve the correct implied volatility used as an input to the Garman-Kohlhagen pricing formula. In time-space, ForexClear will be using linear interpolation along the variance, which leaves only the interpolation along volatility smiles to be addressed[1].

ForexClear will be receiving market data from member banks in the form of ATM, RR10, RR25, FLY10, FLY25 across a range of tenors from ON to 2Y. These quotations will then be converted into a volatility surface whereby each smile is defined in delta space. Each volatility smile will then be mapped to a log-moneyness axis (axis consistent across all tenors) and it is this form of the volatility surface that the pricing, and therefore interpolation will be applied.

As there exists a large number of interpolation methodologies that could be employed to calculate the correct implied volatility given a set of market data, this document will consider some of the (well-known) alternative interpolation methodologies that are applicable within a CCP context. The three models considered are:

1. Linear interpolation
2. Natural cubic spline interpolation
3. Constrained cubic spline interpolation

Each model discussed is assessed against the following desired properties for an interpolation method:

1. Continuous first derivative
2. No risk of negative implied volatilities
3. Minimal risk of “overshooting”
4. Simple and computationally fast implementation

Based on our adoption criteria, we ultimately conclude that the constrained cubic spline methodology provides a highly robust and tractable interpolation method.

2. Interpolation Methods

2.1 Linear Interpolation

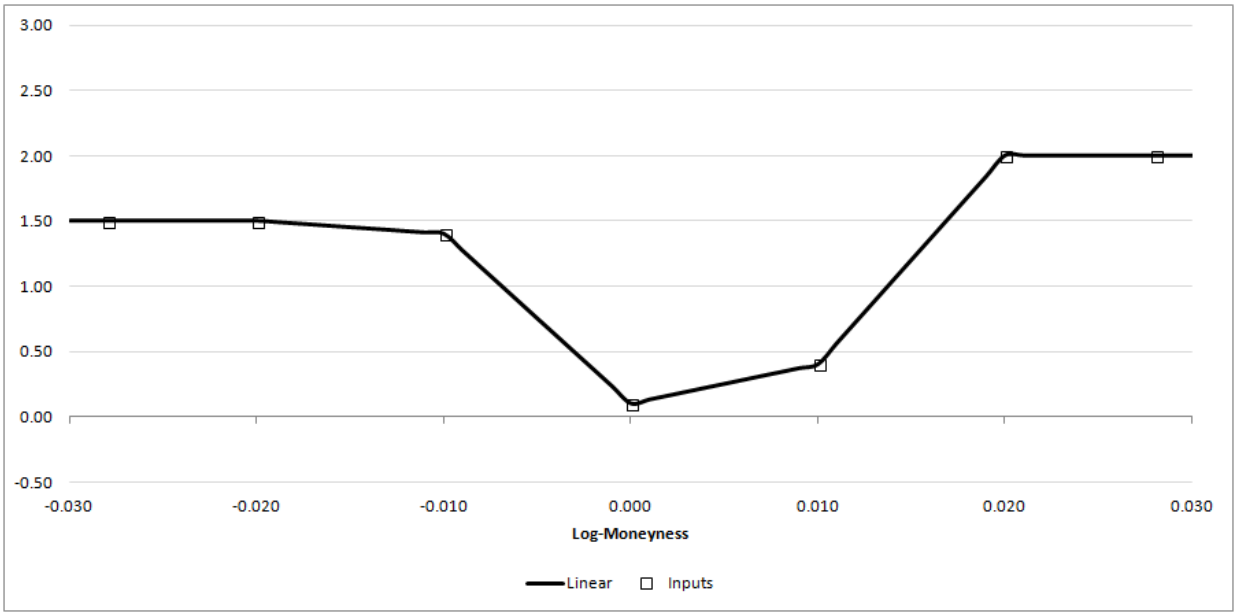
As shown in Figure 1 below, linear interpolation is the simplest interpolation method discussed in this document and computationally is the quickest interpolation method. Linear interpolation ensures that the interpolated curve goes through all of the quoted points (“inputs”) and there is no risk of oscillations occurring.

Linear interpolation fails to capture any curvature, the error arising from lack of curvature can be minimised by increasing the number of input points, however ForexClear market data restrictions dictate only 5 points along each volatility smile.

Another disadvantage of linear interpolation is that it has discontinuous first derivatives.

[1] Flat extrapolation beyond the 10 and 90 deltas will be applied to minimise the potential for arbitrage violations in the extremities of the wings.

Figure 1 - Example of Linear Interpolation



2.2 Natural Cubic Spline Interpolation

A natural cubic spline curve is a piecewise cubic curve with continuous second derivative. The natural cubic spline interpolation2 consists of constructing n-1 segments ("splines") of the smile between the n input points using third degree polynomials. Each spline is constructed using the following constraints:

1. Each spline must pass through 2 consecutive points
2. For each input point (except 1st and last) the first and second derivatives must be continuous
3. For the first and last input point, the first derivative must be equal to 0 (flat extrapolation assumed).

Given N input points$(x_0, y_0), \dots, (x_i, y_i), \dots, (x_N, y_N)$, natural cubic spline interpolation can be defined as follows:

Each spline can is a polynomial of the third degree,

$$f_i(x) = a_i x + b_i x + c_i x^2 + d_i x^3 \quad (1)$$

Such that each curve passes through the input points, ie

$$f_i(x_i) = f_{i+1}(x_i) = y_i \quad (2)$$

Both the first and second order derivatives of the functions either side of an input point are equal,

[2] A formal definition of the natural cubic spline interpolation method can be found at: http://www.geos.ed.ac.uk/~yliu23/docs/lect_spline.pdf

$$f_i{}'(x_i) = f'_{i+1}(x_i) \quad (3)$$

$$f_i{}''(x_i) = f''_{i+1}(x_i) \quad (4)$$

The first derivative of the first and last points of the curve is set to 0 (flat extrapolation)

$$f'_1(x_0) = f'_n(x_n) = 0 \quad (5)$$

Using equations (1)-(5) as constraints, we can construct a linear system of equations such that the solution for each *i* returns the coefficients required in equation (1).

The natural cubic spline method addresses some of the limitations of the linear interpolation by the use of third degree polynomials for each spline, producing a construction that is twice differentiable and also exhibits some degree of convexity.

There is a risk of oscillations when using the natural cubic spline as an interpolation method. Additionally, the natural cubic spline interpolation may result in the effect of "overshooting" in which turning points occur in between input points which could result in negative implied volatilities. Figures 2 and 3 below show the behaviour of the natural cubic spline under various conditions.

Figure 2 - Example of Natural Cubic Spline Interpolation

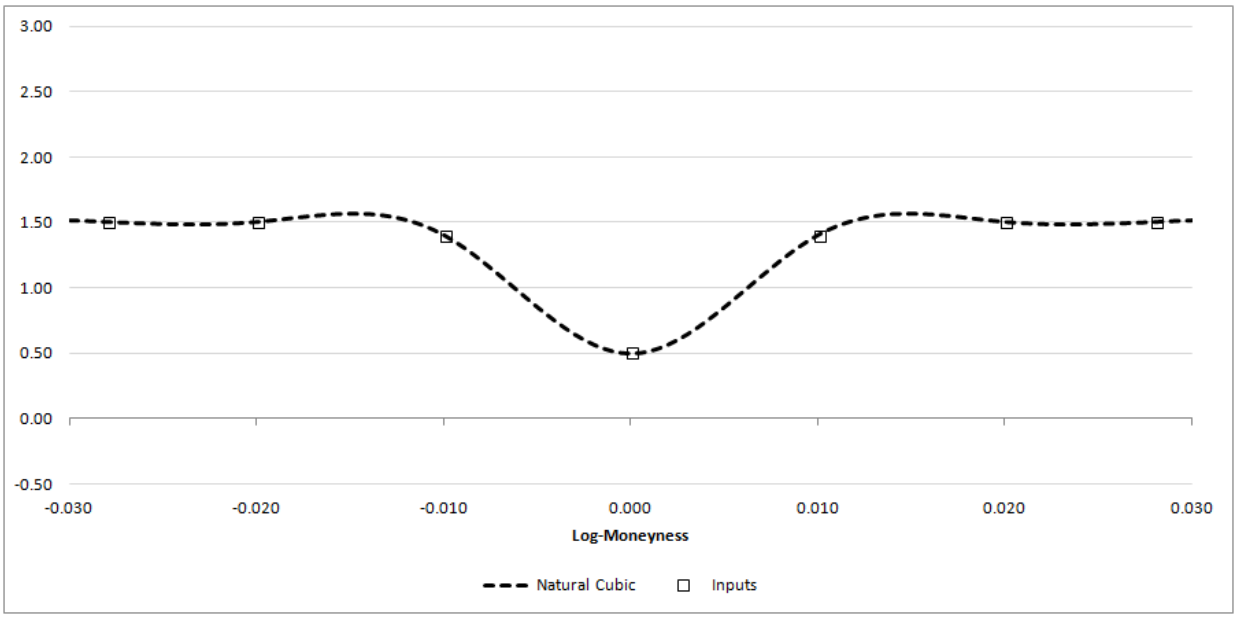
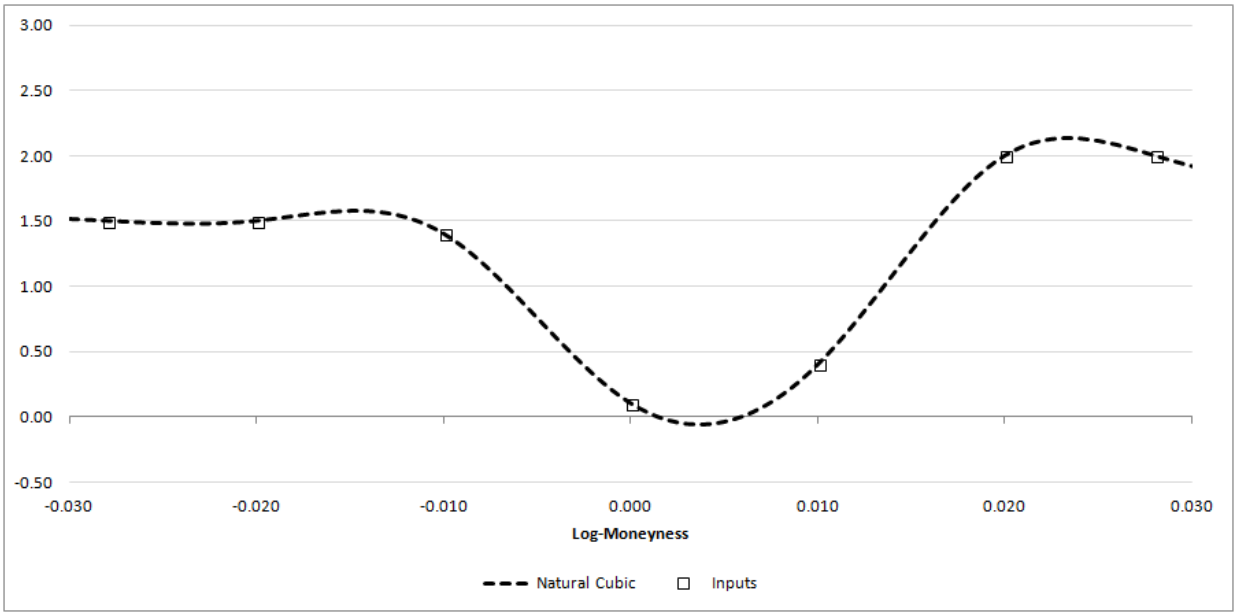


Figure 3 - Example of when Natural Cubic Spline could produce Negative Implied Volatilities



2.3 Constrained Cubic Spline Interpolation

After analysing the weaknesses of both the linear and cubic spline interpolation we can see that whilst both the convexity and differentiability of the natural cubic spline are desirable, the "overshooting" and oscillation effects are not. However a solution that retains the positive properties, whilst limiting the downsides of the natural cubic spline, would be optimal.

To address this, the constrained cubic spline interpolation[3] relaxes the constraint that the second derivative must be continuous whilst adding a constraint that ensures that each spline along the volatility smile is monotonic; this additional property ensures no negative implied volatilities and also no oscillations.

The constrained cubic spline interpolation uses the same method of the natural cubic spline, with the following constraints:

1. If the vertical difference between an input point and any of its adjacent points is 0, the first derivative of the smile at that point is set to 0.
    - This effectively causes the two input points that have the same y-value to act as turning points, therefore causing the smile to become horizontal at that point which eradicates the risk of negative implied volatilities.
2. If an input point has a lower vertical distance than both of its adjacent input points, the first derivative of the smile at that point is set to 0.
    - This causes the input point in question to become a turning point of the volatility smile, which removes the risk of oscillations and overshooting the interpolation – which could potentially cause negative implied volatilities.

[3] http://www.korf.co.uk/spline.pdf

The constrained cubic spline methodology is a modification to the natural cubic spline outlined in 2.2

Given a collection of input points $(x_0, y_0), \dots, (x_i, y_i), \dots, (x_n, y_n)$ we can use equations (1)-(3) & (5) defined in 2.2, as well as the additional constraint that the first derivative at each input can be specified,

$$f_i'(x_i) = f_{i+1}'(x_i) = f'(x_i) \tag{6}$$

To solve equations (1)-(6) we must calculate the gradient at each point using,

$$f'(x_i) = 2\left(\frac{x_{i+1}-x_i}{y_{i+1}-y_i} + \frac{x_i-x_{i-1}}{y_i-y_i-1}\right)^{-1} \tag{7}$$

For each input point, if either adjacent points are higher (or lower) or one of the adjacent points has the same vertical distance, we set the gradient to 0,

$$(y_{i+1} - y_i)(y_i - y_{i-1}) \leq 0 \Rightarrow f'(x_i) = 0 \tag{8}$$

The gradient at each end point is defined separately,

$$f_1'(x_0) = \frac{3(y_1-y_0)}{2(x_1-x_0)} - \frac{f'(x_1)}{2} \tag{9}$$

$$f_n'(x_n) = \frac{3(y_n-y_{n-1})}{2(x_n-x_{n-1})} - \frac{f'(x_{n-1})}{2} \tag{10}$$

Using the above formulas for the first derivative, we can calculate the second derivative at the 2 inputs points of each spline,

$$f_i''(x_{i-1}) = -\frac{2[f_i'(x_i)+2f_i'(x_{i-1})]}{(x_i-x_{i-1})} + \frac{6(y_i-y_{i-1})}{(x_i-x_{i-1})^2} \tag{11}$$

$$f_i''(x_i) = \frac{2[2f_i'(x_i)+f_i'(x_{i-1})]}{(x_i-x_{i-1})} - \frac{6(y_i-y_{i-1})}{(x_i-x_{i-1})^2} \tag{12}$$

And using the first and second derivatives we can define the coefficients for each spline *i* that we use inf_i,

$$d_i = \frac{f_i''(x_i)-f_i''(x_{i-1})}{6(x_i-x_{i-1})} \tag{13}$$

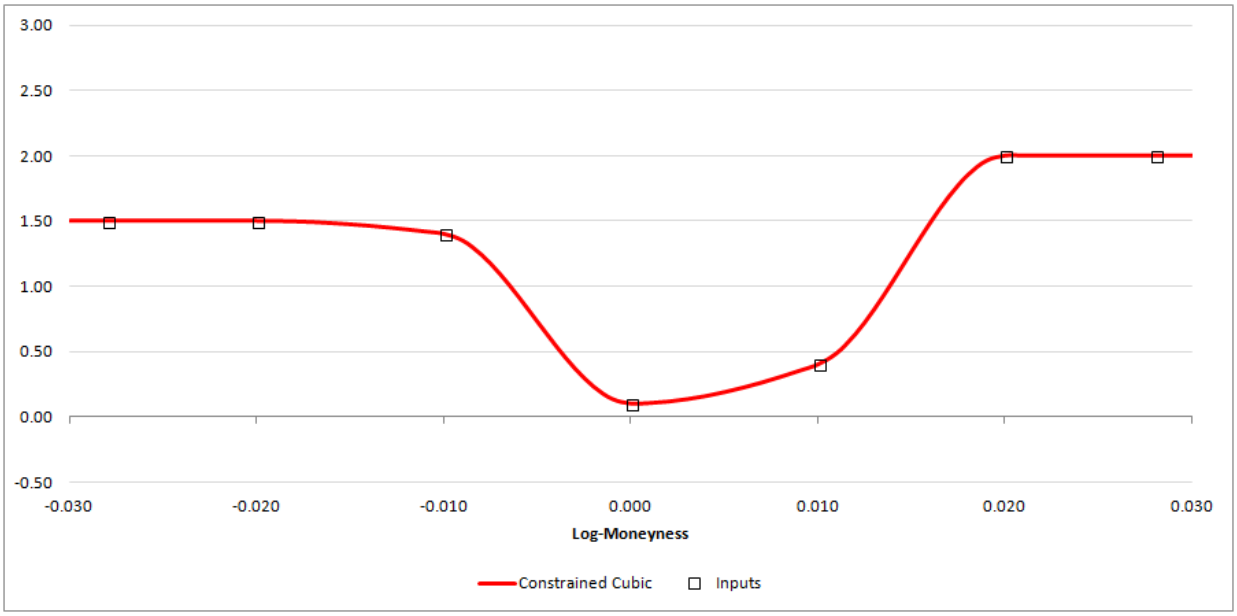
$$c_i = \frac{x_i f_i''(x_{i-1})-x_{i-1}f_i''(x_i)}{2(x_i-x_{i-1})} \tag{14}$$

$$b_i = \frac{y_i-y_{i-1}-c_i(x_i^2-x_{i-1}^2)-d_i(x_i^3-x_{i-1}^3)}{(x_i-x_{i-1})} \tag{15}$$

$$a_i = y_{i-1} - b_i x_{i-1} - c_i x_i^2 - d_i x_{i-1}^3 \tag{16}$$

Figure 4 has the same input points as Figure 3, however due to the additional constraints the interpolation does not "overshoot" between the 50 and 75 delta points and as a result does not produce negative volatilities. We can also see from Figure 4 that the convexity of this smile remains.

Figure 4 - Example of Constrained Cubic Spline exhibiting no overshooting



3. Optimisation to Constrained Cubic Spline

One of the downsides to the constrained cubic spline interpolation methodology is the time taken to interpolate along a set of input points. Although the computational time is not as high as that of a natural cubic spline interpolation, constrained cubic spline in its native form is not as efficient as standard linear interpolation. Time constraints are a key factor in deciding interpolation methodology for Initial Margin calculations, due to the necessity for Real Time Registration.

It is possible to drastically reduce the performance of constrained cubic spline interpolation by pre-computing the co-efficients a, b, c, d for each spline. For each volatility smile of 5 input points, we create 4 splines and therefore need to calculate 16 co-efficients (4 splines * 4 constants).

By pre-computing each of the co-efficients, we can then reduce the process of interpolation using the constrained cubic spline to simply identify which spline the point lies along and then performing a single algebraic equation.

3.1 Performance Comparison

As noted in the introduction, speed of computation is one of the key factors when deciding which interpolation method is to be performed. To compare each method and assess the impact of pre-computing the constants a test whereby X interpolations along a smile of 5 input points was performed and the the time taken each method takes to complete was recorded. Results for this test can be seen in Table 1.

Table 1 - Time taken (in seconds) to perform the number of interpolations for different interpolation methods

Number of interpolations performed	Linear Interpolation	Constrained Cubic Interpolation	Constrained Cubic with co-efficients pre-computed
100,000	0.02	0.04	0.01
250,000	0.05	0.1	0.04
1,000,000	0.17	0.43	0.16

Another key factor considered when deciding which methodology to use for pricing is the accuracy of the interpolation method. With that in mind, comparison tests were performed to assess the number of implied volatility smile arbitrages found when using the same input points but different interpolation methods (linear and constrained cubic spline).

Table 2 outlines results of the percentage of volatility smiles arbitrages found under each interpolation method for all ForexClear historical data from January 2001 to November 2014. The tolerance is representative of the spread that would need to be crossed in the market to execute a butterfly or call spread. Therefore in order to monetize a potential arbitrage in the market one would need to factor into account this tolerance.

Table 2 - Percentage of Arbitrages when performing either linear or constrained cubic interpolation

	AUDUSD		EURCHF		EURGBP		EURJPY	
Tolerance	Linear	Cubic	Linear	Cubic	Linear	Cubic	Linear	Cubic
0	99.24%	86.61%	99.80%	96.06%	99.84%	89.36%	99.74%	98.23%
0.0001	78.67%	0.25%	83.06%	2.76%	92.33%	0.32%	85.21%	7.76%
0.0002	67.38%	0.00%	70.53%	0.70%	85.41%	0.00%	75.57%	1.32%
0.0003	60.31%	0.00%	62.47%	0.04%	79.08%	0.00%	68.24%	0.25%
0.0005	49.46%	0.00%	50.28%	0.00%	69.20%	0.00%	58.23%	0.01%
	EURUSD		**GBPUSD**		**USDCHF**		**USDJPY**	
Tolerance	Linear	Cubic	Linear	Cubic	Linear	Cubic	Linear	Cubic
0	99.76%	94.77%	99.82%	95.91%	99.78%	93.56%	99.92%	97.58%
0.0001	88.86%	3.13%	89.56%	3.50%	89.29%	0.60%	89.47%	2.52%
0.0002	80.85%	0.00%	81.00%	0.00%	78.92%	0.00%	80.15%	0.00%
0.0003	74.93%	0.00%	74.13%	0.00%	72.23%	0.00%	72.41%	0.00%
0.0005	66.64%	0.00%	64.39%	0.00%	61.76%	0.00%	61.52%	0.00%

4. Conclusions

Based on our adoption criteria, it is clear to see that the constrained cubic spline methodology offers the best performance for interpolation along a volatility surface, compared to both linear interpolation and natural cubic spline interpolation.

Figure 5 and Figure 6 below show all three interpolation types mapped on to the same graph and can be used to visually highlight the key differences between the three models.

Figure 5 highlights that linear interpolation does not exhibit enough convexity, particularly around the ATM point. Figure 6 highlights that the natural cubic spline interpolation can sometimes result in overshooting, which increases the likelihood of arbitrages and negative implied volatilities.

The constrained cubic spline method effectively incorporates the advantages of both models whilst eliminating the highlighted concerns. Additionally, the constrained cubic spline interpolation reduces the number of implied volatility smile arbitrages and when enhanced by pre-computing coefficients, can be performed in around the same time as linear interpolation. Constrained cubic spline interpolation does not require calibration, performs efficiently given the smile input data, and remains computationally simple as opposed to a more complex model, eg. SABR. For the purposes of smile interpolation within the context of CCP valuation and margining, we conclude that the constrained cubic spline methodology is fit-for-purpose.

Figure 5 - Comparison of Different Interpolation Methods for Symmetrical Smile

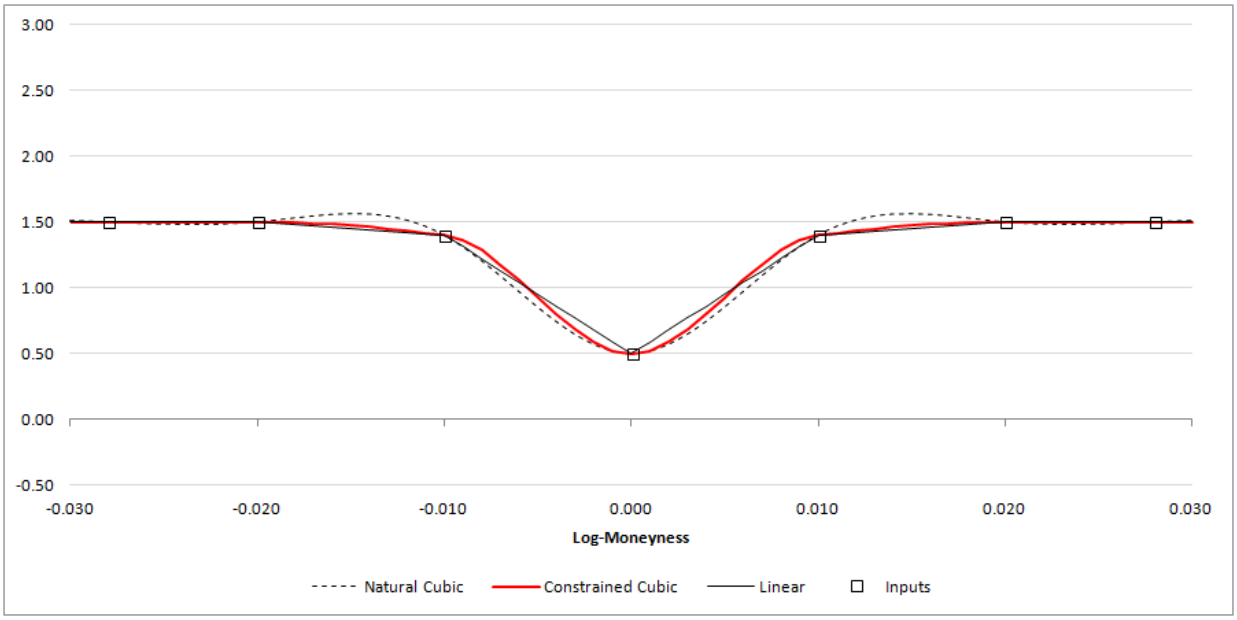
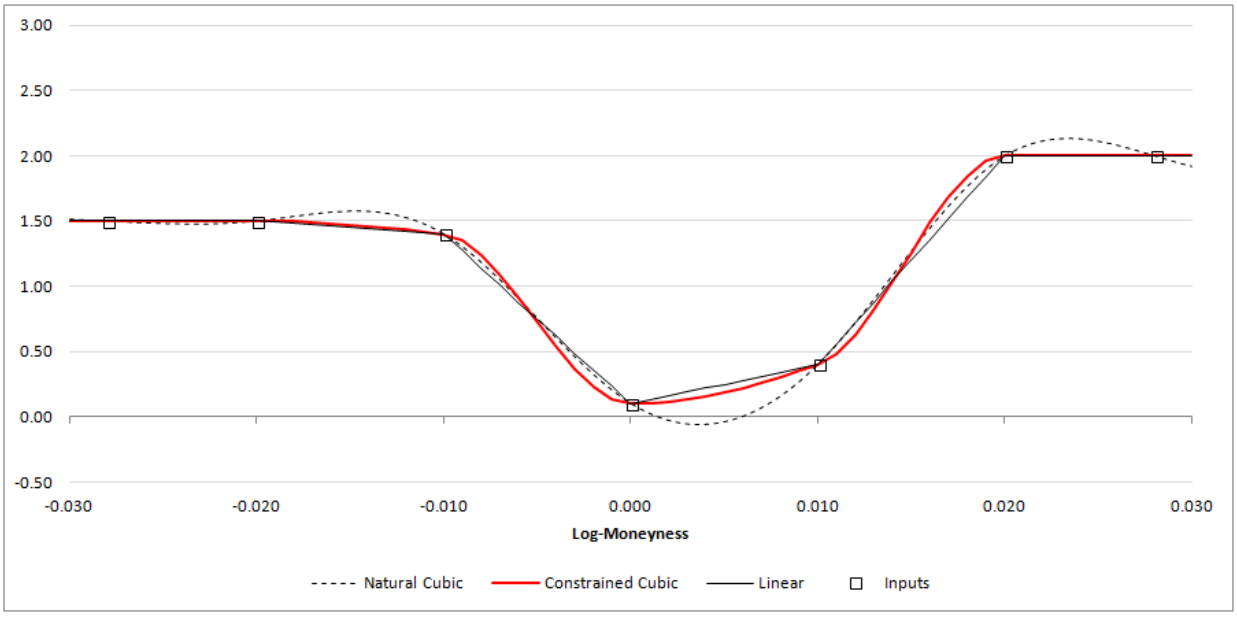


Figure 6 - Comparison of Different Interpolation Methods with Large Skew



Appendix A - Algorithm for Constrained Cubic Spline Interpolation with Pre-Computed Constants

Algorithm 1 Pre-Computation of Constrained Cubic Spline Co-efficients

// Pseudocode to calculate coefficients for use in constrained cubic spline interpolation, given N inputs points with indices 0 to N-1

for $i = 1$ **to** $N-2$ **do** // For each input point not an end point

  if $(y[i+1]-y[i]) * (y[i]-y[i-1]) \leq 0$ **then**

    $f'[i] \leftarrow 0$

  else

    $f'[i] \leftarrow 2/(\frac{x[i+1]-x[i]}{y[i+1]-y[i]} + \frac{x[i]-x[i-1]}{y[i]-y[i-1]})$

$f'[0] \leftarrow \frac{3(y[1]-y[0])}{2(x[1]-x[0])} - 0.5f'[1]$

$f'[N-1] \leftarrow \frac{3(y[N-1]-y[N-2])}{2(x[N-1]-x[N-2])} - 0.5f'[N-2]$

for $i = 0$ **to** $N-2$ **do**

  $f_i''[i] \leftarrow -2\frac{f'[i+1]+2f'[i]}{x[i+1]-x[i]} + 6\frac{y[i+1]-y[i]}{(x[i+1]-x[i])^2}$

  $f_i''[i+1] \leftarrow 2\frac{2f'[i+1]+f'[i]}{x[i+1]-x[i]} - 6\frac{y[i+1]-y[i]}{(x[i+1]-x[i])^2}$

  $d_i = \frac{1}{6}\frac{f_i''[i+1]-f_i''[i]}{x[i+1]-x[i]}$

  $c_i = 0.5\frac{x[i+1]f_i''[i]-x[i]f_i''[i+1]}{x[i+1]-x[i]}$

  $b_i = \frac{y[i+1]-y[i]-c_i(x[i+1]^2-x[i]^2)-d_i(x[i+1]^3-x[i]^3)}{x[i+1]-x[i]}$

  $a_i = y[i] - b_i x[i] - c_i x[i]^2 - d_i x[i]^3$

Algorithm 2 Interpolation using constrained cubic spline with co-efficients as input

// Pseudocode to perform constrained cubic spline interpolation, with a, b, c and d vectors as inputs, as well as the value to be interpolated on, xi

for $i = 0$ **to** $N-2$ **do** // For each spline

  if $(xi \leq x[i+1])and(xi \geq x[i])$ **then**

    $res \leftarrow a_i + b_i xi + c_i xi^2 + d_i xi^3$

if $(xi \leq x[0]))$ **then**

  $res \leftarrow y[0]$

if $(xi \geq x[N-1]))$ **then**

  $res \leftarrow y[N-1]$