Fitting the term structure of interest rates: the practical implementation of cubic spline methodology

Rod Pienaar
Corporate & Investment Banking Division
Deutsche Bank AG, London

Moorad Choudhry*
Centre for Mathematical Trading and Finance
City University Business School, London

* moorad@yieldcurve.com

INTRODUCTION

Fitting the term structure of interest rates

The term structure of interest rates defines the set of spot or zero-coupon rates that exist in a debt capital market, of default-free bonds, distinguished only by their term to maturity. In practice the term structure is defined as the array of discount factors on the same maturity term. Extracting the term structure from market interest rates has been the focus of extensive research, reflecting its importance in the field of finance.

The term structure is used by market practitioners for valuation purposes and by central banks for forecasting purposes. The accurate fitting of the term structure is vital to the smooth functioning of the market. A number of approaches have been proposed with which to undertake this, and the method chosen is governed by the user's requirements. Practitioners desire an approach that is accessible, straightforward to implement and as accurate as possible. In general there are two classes of curve fitting techniques; the *parametric* methods, so-called because they attempt to model the yield curve using a parametric function; and the *spline* methods.[1] Parametric methods include the Nelson-Siegel model and a modification of this proposed by Svensson (1994, 1995), as well as models described by Wiseman (1994) and Bjork and Christensen (1997).[2] James and Webber (2000) suggest that these methods produce a satisfactory overall shape for the term structure but are suitable only where good accuracy is not required.[3] Market practitioners instead generally prefer an approach that gives a reasonable trade-off between accuracy and ease of implementation, an issue we explore in this article.

The cubic spline process presents no conceptual problems, and is an approximation of the market discount function. McCulloch (1975) uses cubic splines and Beim (1992) states that this approach performs at least as satisfactorily as other methods.[4] Although the basic approach can lead to unrealistic shapes for the forward curve (for example, see Vasicek and Fong (1982) and their suggested improvement on the approach using exponential splines), it is an accessible method and one that gives reasonable accuracy for the spot rate curve. Adams and Van Deventer (1994) illustrate using the technique to obtain maximum smoothness for forward curves (and an extension to *quartic* splines), while the basic technique has been improved as described by Fisher, Nychka and Zervos (1995), Waggoner (1997) and Anderson and Sleath (1999). These references are considered later.

Splines are a non-parametric polynomial interpolation method.[5] There is more than one way of fitting them. The simplest method is an ordinary least squares regression spline, but this approach produces wildly oscillating curves. The more satisfactory is a smoothing splines method. We consider the basic approach and how to implement it in this article.

[1] Parametric models are also known as *parsimonious* models.

[2] Nelson, C., Siegel, A., "Parsimonious modeling of the Yield Curve", *Journal of Business* 60, no 4 (1987), pp.473-489. Svensson, L., "Estimating Forward Interest Rates with the Extended Nelson and Siegel Method", *Sveriges Riksbank Quarterly Review* 3, (1995). Wiseman, J., "The Exponential Yield Curve Model", *JPMorgan European Fixed Income Research*, 1994. Bjork, T., Christensen, B., "Interest Rate Dynamics and Consistent Forward Rate Curves", *University of Aarhus Working Paper*, 1997, pp.1-38

[3] James, J., Webber, N., *Interest Rate Modelling*, Wiley 2000, page 444

[4] McCulloch, J., "The Tax-Adjusted Yield Curve", *Journal of Finance* 30, 1975, pp. 811-830. Beim, D., "Term Structure and the Non-Cash Value in Bonds", *First Boston Working Paper series*, 1992

[5] A spline originally referred to a tool used by draughtsmen or carpenters for drawing smooth curves.

Cubic Splines

Fitting a discount function

In mathematics a spline is a piecewise polynomial function, made up of individual polynomial sections or segments that are joined together at (user-selected) points known as *knot points*. Splines used in term structure modelling are generally made up with cubic polynomials, and the reason for cubic polynomials, as opposed to polynomials of order say, two or five, is explained in straightforward fashion by de la Grandville (2001).[6] A cubic spline is a function of order three, and a piecewise cubic polynomial that is twice differentiable at each knot point. At each knot point the slope and curvature of the curve on either side must match. We employ the cubic spline approach to fit a smooth curve to bond prices (yields) given by the term discount factors.

A polynomial of sufficiently high order may be used to approximate to varying degrees of accuracy any continuous function, which is why a polynomial approximation of a yield curve may be attempted. For example James and Webber (2000) state that given a set of *m* points with distinct values, a Lagrange polynomial of degree *m* will pass through every point.[7] However the fit can be very wild with extreme behaviour at the long end. We will demonstrate how a cubic spline approximation can be used to obtain better results.

This paper provides a discussion of piecewise cubic spline interpolation methodology and its application to the term structure. The authors' intent is to provide a comprehensive and accessible approach to cubic spline interpolation for implementation by practitioners. At the end of this paper the reader should have a full understanding of how cubic splines are calculated and the implications of using piecewise cubic spline interpolation methods. In addition the user can employ the approach shown to implement the methodology for their own applications, including constructing spot and forward yield curves from market-determined interest rates. We recommend a cubic spline technique because this ensures that the curve passes through all the selected (market determined) node points. This enables practitioners to fit a yield curve to observed market rates (Libor or bond yields) reasonably accurately and produces a satisfactory zero coupon curve under most circumstances.

Our starting point is a set of zero curve tenors (or discount factors) obtained from a collection of market instruments such as cash deposits, futures, swaps or coupon bonds. We therefore have a set of tenor points and their respective zero rates (or discount factors). The mathematics of cubic splines is straightforward but we assume a basic understanding of calculus and a familiarity with solving simultaneous linear equations by substitution. An account of the methods analysed in this paper is given in Burden and Faires (1997), which has very accessible text on cubic spline interpolation.[8]

[6] De la Grandville, O., *Bond Pricing and Portfolio Analysis*, MIT Press 2001, pp.248-252
[7] Op. cit., pp. 430-432
[8] Burden, R., Faires, D., *Numerical Analysis*, Brooks/Cole 1997

Background on cubic splines

When fitting a curve by interpolating between nodes or tenor points, the user must consider conflicting issues. There is a need to balance between simplicity and correctness, and hence a trade off between ease of use and the accuracy of the result. In certain cases practitioners will accept a lower degree of accuracy at the nodes, in favour of smoothness across the curve. In the cubic spline approach the primary aim is smoothness. In an attempt to create a smooth and accurate measurement at the nodes however, we may be confronted by oscillation in the curve. Although linear interpolation is a reasonable calculation method, interest rate markets are not linear environments made up of coupled straight lines. The point between two tenors cannot be accurately estimated using a straight line.

Although there are a number of alternative methods available to the practitioner, a reasonable approach is to stick with the concept of piecewise interpolation but to abandon the use of straight lines. The reason that we do not depart from piecewise interpolation, is that this method of curve smoothing provides accuracy at the nodes because each piecewise function touches a node. Accuracy at the nodes can be an important consideration when a pricing methodology based on the elimination of arbitrage is employed. Thus we continue with piecewise fitting, but instead of applying a linear fitting technique, we apply a cubic polynomial to each piece of the interpolation. Cubic splines provide a great deal of flexibility in creating a continuous smooth curve both between and at tenor points.[9]

CUBIC SPLINE METHODOLOGY

We assume that the practitioner has already calculated a set of nodes using a yield curve construction technique such as bootstrapping. A zero curve is then fitted using the cubic spline methodology by interpolating between nodes using individual cubic polynomials. Each polynomial has its own parameters but are constructed in such a way that their ends touch each node at the start and end of the polynomial. The set of splines, which touch at the nodes, therefore form a continuous curve. Our objective is to produce a continuous curve, joining market observed rates as smoothly as possible, which is the most straightforward means by which we can deduce meaningful data on the correct interest rate term structure in the market.

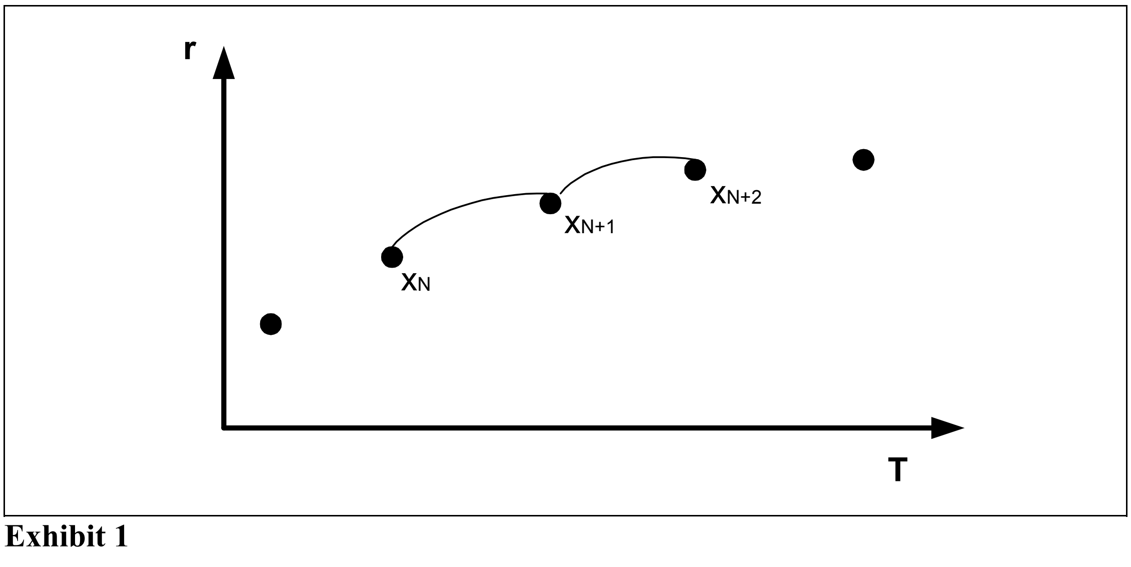

Exhibit 1

[9] See the earlier footnote for a word on the origin of the use of the term "spline".

In Exhibit 1 we can see that two cubic polynomials which join at point x_{N+1} are used to form a continuous curve. However, it is also clear from the curves in figure 1 that the two polynomials do not result in a smooth curve. In order to have a smooth curve we need to establish "smoothing" criteria for each spline. To do this we must first ensure that the polynomials touch or join together at the nodes. Secondly we must ensure that where the polynomials touch that the curve is smooth. Finally we ensure that the curve is continuously differentiable, or in other words, the curve has a smooth rate of change at and between tenor points. The required criteria to meet these conditions are:

Requirement 1: the value of each polynomial is equal at tenor points;
Requirement 2: the first differential of each polynomial is equal at tenor points;
Requirement 3: the second differential of each polynomial is equal at tenor points; and
Requirement 4: the second differential of each polynomial is continuous between tenor points.

Considering a polynomial of the form $y = ax^3+bx^2+cx+d,$ the second differential $y'' = 6ax + 2b$ is a linear function and by its very definition is continuous between tenor points. The fourth requirement is therefore always met and this paper will not deal with this requirement in any further detail. The rest of this paper will refer to the first three requirements and how they are met at the nodes.

THE HYPOTHESIS

Assuming the final solution is unknown at this stage, it seems plausible that an almost infinite set of parameters *a, b* and *c* can be found which will result in all of our cubic spline requirements being met.

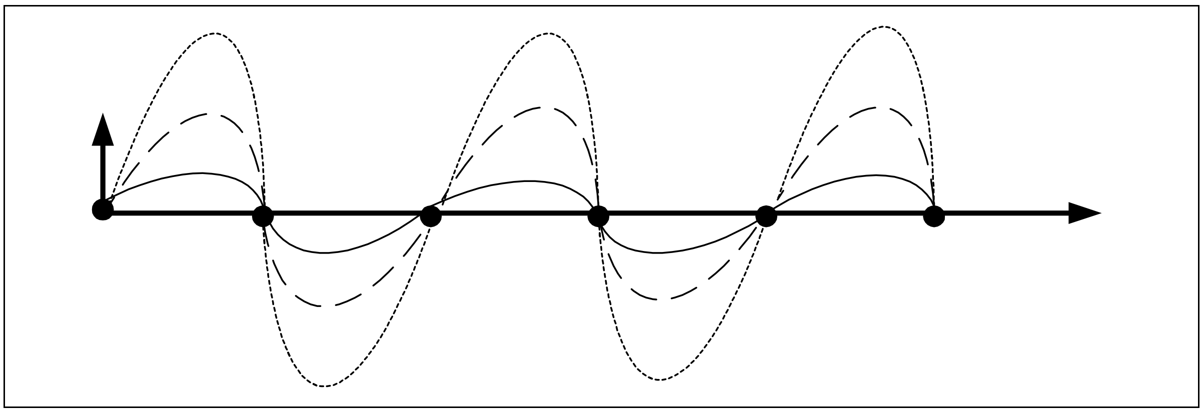

Exhibit 2

We observe in Exhibit 2 three imaginary curves, all of which would meet our requirements that the:

- first differential of each spline is equal at tenor points; and
- second differential of each spline is equal at tenor points.

Admittedly we have considered nodes that are sitting in a straight line but even where the nodes do not line up it may be possible to find a range of possible solutions. Taking this further, spline A and spline B as shown in Exhibit 3 are valid solutions yet it is intuitive, given our knowledge of interest rate markets, that A is likely to be more suitable for our purposes of yield curve interpolation.

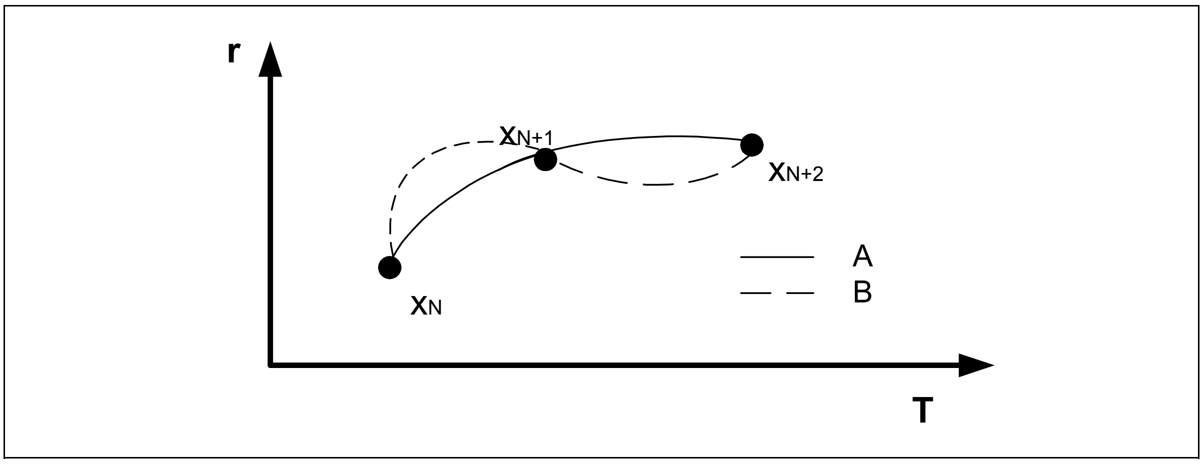

Exhibit 3

The issue to determine therefore, is, is there an infinite set of parameters each of which would meet our requirements for fitting the curve; or is it possible to determine a single solution? Of course our requirement is in a single solution; moreover, a solution that can be found quickly from any set of market rates.

PRACTICAL APPROACH

A working environment

By splitting the yield curve into individual node/tenor pairs, we may work with individual lines within each tenor. A cubic polynomial can then be added to each line to provide the cubic spline. For ease of illustration, we take this one step further and imagine an alternative horizontal axis. This is referred to as 'capital' X as shown in Exhibit 4. Assume that between each node pair that this horizontal axis X runs from 0 (at x_N) to $x_{N+1} - x_N$ (at x_{N+1}).

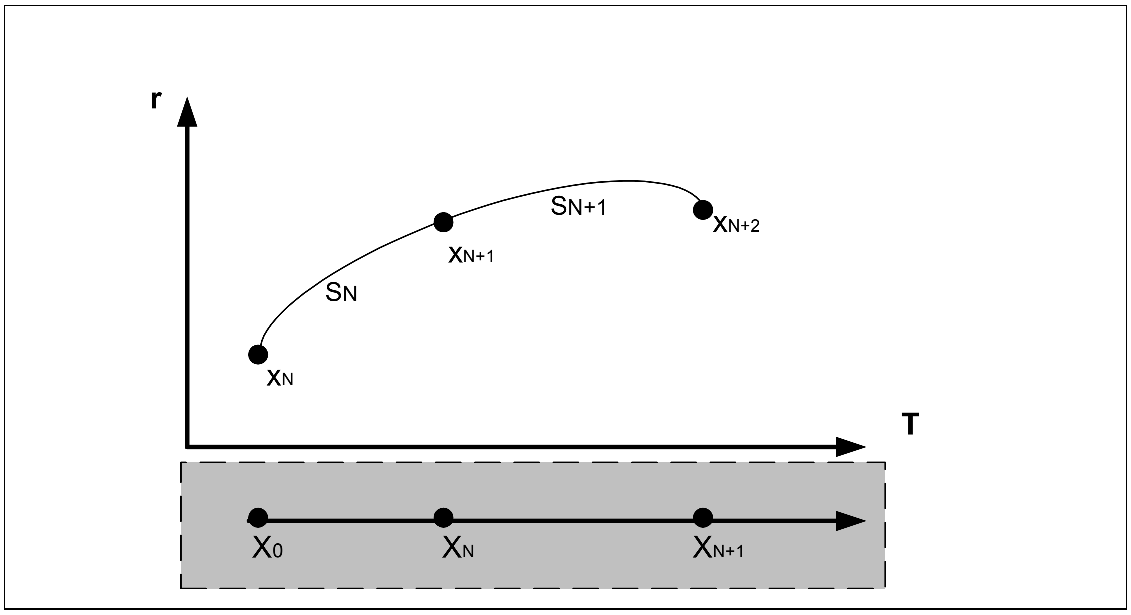

Exhibit 4

In Exhibit 4 the *X* axis is a calculated value determined from the *x* axis. The points x_N and x_{N+1} are isolated for spline S_N. It is then assumed that X_0 equals zero at x_N and stretches to X_N which equals $(x_{N+1}-x_N)$ on the *X* axis. If these lines are fully isolated then a cubic polynomial, of the form $y = aX^3 + bX^2 + cX + d$, can be constructed to touch the points x_N and x_{N+1}.

The first requirement

In order for the polynomial to touch the nodes then a cubic polynomial must be constructed so that at point X_0 the polynomial provides a results that is equal to y_N. This is very easy to achieve. Since *X* is equal to zero at its starting point, the polynomial takes the following form:

$$y_N = a_N 0^3 + b_N 0^2 + c_N 0 + d_N$$
$$y_N = d_N$$

So as long as d_N is equal to y_N then our polynomial will touch the node at X_0.

In order for the polynomial to touch the second node, the node at point x_{N+1}, then the polynomial must take the following form at point X_N:

$$y_{N+1} = a_N (x_{N+1} - x_N)^3 + b_N (x_{N+1} - x_N)^2 + c_N (x_{N+1} - x_N) + d_N$$

OR

$$\boxed{d_{N+1} = a_N X_N{}^3 + b_N X_N{}^2 + c_N X_N + d_N} \qquad (1)$$

where: $X_N = x_{N+1} - x_N$

It is worth noting that at this point in our process we do not know what the values of *a, b* or *c* are. These will be derived below from our other requirements.

<u>The second requirement</u>

To meet the second requirement of a cubic spline, the first differential y_N' must equal the first differential y_{N+1}' at the tenor point x_{N+1}.

In other words at node x_{N+1}:

$$3a_N X_N{}^2 + 2b_N X_N + c_N = 3a_{N+1} X_{N+1}{}^2 + 2b_{N+1} X_{N+1} + c_{N+1} \tag{2}$$

We know from our conditional working environment that at node x_{N+1} for function y_N' that $X = (x_{N+1} - x_N)$. We also know from the same assumption that $X = 0$ at the start of the next polynomial, i.e. for function y_{N+1}'. Therefore:

$$3a_{N+1} 0^2 + 2b_{N+1} 0 + c_{N+1} = 3a_N X_N{}^2 + 2b_N X_N + c_N$$

so that

$$\boxed{c_{N+1} = 3a_N X_N{}^2 + 2b_N X_N + c_N} \tag{3}$$

<u>Third requirement</u>

To meet the third requirement of a cubic spline, the second differential y_N'' assessed at the point x_{N+1} should equal the second differential y_{N+1}''.

In other words at node x_{N+1}:

$$6a_N X_N + 2b_N = 6a_{N+1} X_{N+1} + 2b_{N+1}$$

We know from our conditions that at node x_{N+1} for function y_N'' that $X = (x_{N+1} - x_N)$. We also know from the same assumption that $X = 0$ for function y_{N+1}''. Therefore:

$$6a_N X_N + 2b_N = 6a_{N+1} 0 + 2b_{N+1}$$
$$6a_N X_N = 2b_{N+1} - 2b_N$$

$$\boxed{a_N = \frac{b_{N+1} - b_N}{3X_N}} \tag{4}$$

<u>Meeting all requirements simultaneously</u>

We now have equations which ensure that each of the requirement can be met. We now need a solution that will ensure that all requirements are met at the same time. By substitution a set of calculations can be performed which meet both requirements and reduce these equations down to a factor of parameter *b* only.

Using equation (4) as a substitute for *a* in equation (3) we obtain:

$$c_{N+1} = 3a_N X_N{}^2 + 2b_N X_N + c_N$$
$$c_{N+1} = \frac{3(b_{N+1} - b_N)}{3X_N} X_N{}^2 + 2b_N X_N + c_N$$
$$c_{N+1} = (b_{N+1} - b_N)X_N + 2b_N X_N + c_N$$
$$\boxed{c_{N+1} = X_N(b_{N+1} + b_N) + c_N} \qquad (5)$$

Using equation (4) as a substitute for *a* in equation (1) we get:

$$d_{N+1} = \frac{(b_{N+1} - b_N)}{3X_N} X_N{}^3 + b_N X_N{}^2 + c_N X_N + d_N$$
$$d_{N+1} = \frac{(b_{N+1} - b_N)}{3} X_N{}^2 + b_N X_N{}^2 + c_N X_N + d_N$$
$$c_N X_N = -\frac{(b_{N+1} - b_N)}{3} X_N{}^2 - b_N X_N{}^2 + d_{N+1} - d_N$$
$$\boxed{c_N = -X_N \frac{(b_{N+1} + 2b_N)}{3} + \frac{(d_{N+1} - d_N)}{X_N}} \qquad (6)$$

Taking this solution one step further we can substitute equation (6) into equation (5) as follows:

$$\frac{(d_{N+2} - d_{N+1})}{X_{N+1}} - X_{N+1}\frac{(b_{N+2} + 2b_{N+1})}{3} = X_N(b_{N+1} + b_N) - X_N\frac{(b_{N+1} + 2b_N)}{3} + \frac{(d_{N+1} - d_N)}{X_N}$$
$$-X_{N+1}(b_{N+2} + 2b_{N+1}) = 3X_N(b_{N+1} + b_N) - X_N(b_{N+1} + 2b_N) + 3\frac{(d_{N+1} - d_N)}{X_N} - 3\frac{(d_{N+2} - d_{N+1})}{X_{N+1}}$$
$$-X_{N+1}(b_{N+2} + 2b_{N+1}) = X_N(2b_{N+1} + b_N) + 3\frac{(d_{N+1} - d_N)}{X_N} - 3\frac{(d_{N+2} - d_{N+1})}{X_{N+1}}$$
$$X_{N+1}b_{N+2} = -X_N(2b_{N+1} + b_N) - 3\frac{(d_{N+1} - d_N)}{X_N} + 3\frac{(d_{N+2} - d_{N+1})}{X_{N+1}} - 2X_{N+1}b_{N+1}$$
$$\boxed{b_{N+2} = \frac{-2X_N b_{N+1} - X_N b_N - 2X_{N+1}b_{N+1} - 3\frac{(d_{N+1} - d_N)}{X_N} + 3\frac{(d_{N+2} - d_{N+1})}{X_{N+1}}}{X_{N+1}}} \qquad (7)$$

A unique solution

For clarity and ease of illustration, the results of these equations are set out as a table of related formulas shown below in Table 1.

X	y (*d*)	Using equation 4.3 we can derive ***a***	Using equation 4.6 we can derive ***b***	Using equation 4.5 we can derive ***c***
X_1	d_1	$\frac{b_2 - b_1}{3X_1}$	$\mathbf{b_0}$	$-X_1\frac{(b_2+2b_1)}{3}+\frac{(d_2-d_1)}{X_1}$
X_2	d_2	$\frac{b_3 - b_2}{3X_3}$	$\mathbf{b_1}$	$-X_2\frac{(b_3+2b_2)}{3}+\frac{(d_3-d_2)}{X_2}$
X_3	d_3	$\frac{b_4 - b_3}{3X_4}$	$\frac{-2X_1b_2-X_1b_1-2X_2b_2-3\frac{(d_2-d_1)}{X_1}+3\frac{(d_3-d_2)}{X_2}}{X_2}$	$-X_3\frac{(b_4+2b_3)}{3}+\frac{(d_4-d_3)}{X_3}$
…	…	…	...	...
X_{N-1}	d_{N-1}	$\frac{b_N - b_{N-1}}{3X_{N-1}}$	$\frac{-2X_{N-3}b_{N-2}-X_{N-3}b_{N-3}-2X_{N-2}b_{N-2}-3\frac{(d_{N-2}-d_{N-3})}{X_{N-3}}+3\frac{(d_{N-1}-d_{N-2})}{X_{N-2}}}{X_{N-2}}$	$-X_{N-1}\frac{(b_N+2b_{N-1})}{3}+\frac{(d_N-d_{N-1})}{X_{N-1}}$
X_N	d_N	N/A	$\frac{-2X_{N-2}b_{N-1}-X_{N-2}b_{N-2}-2X_{N-1}b_{N-1}-3\frac{(d_{N-1}-d_{N-2})}{X_{N-2}}+3\frac{(d_N-d_{N-1})}{X_{N-1}}}{X_{N-1}}$	N/A

Table 1

It is simple matter to determine the values of parameters *a*, *b*, *c* and *d* at each node *n* by using the formulas set out in the table. Each node (from $n > 2$) is directly or indirectly dependent on the values of previous parameters and can be determined from those previous parameters. This is an important result, and means that any errors in the calculation early on are replicated and magnified throughout the analysis. However, the first two occurrences of *b* (b_0 and b_1) do not have previous nodes from which to determine their values. In other words the only values for which we do not have solutions are those for b_0 and b_1.

Depending on the values assumed for b_0 and b_1, the result is *usually* an oscillating *b* and ever increasing $|b|$. This means that the slope of the spline gets steeper at each tenor as the absolute value of the first differential increases, so the slope of the curve oscillates.

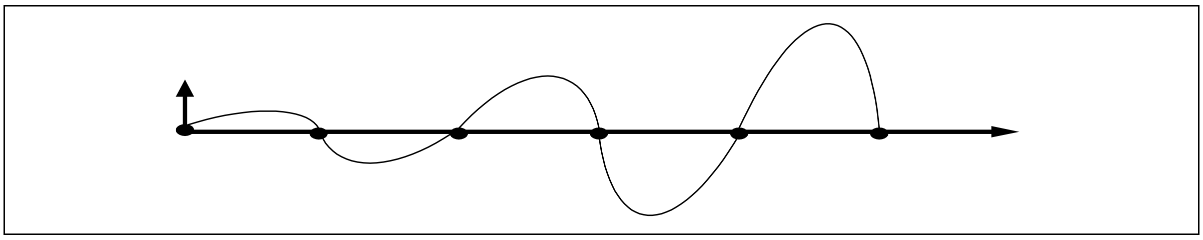

Exhibit 5

This systematic wave, shown in Exhibit 5 above, is clearly not the kind of behaviour that is commonly observed in a yield curve and should therefore not be modelled into the curve. Furthermore, we have no unique solution at this stage. An infinite number of values can be assigned to b_0 and b_1 and therefore an infinite number of solutions can be obtained (most of which exhibit the depicted oscillation effect). So this is still not what we seek.

We need an additional restriction that allows us to find a single solution and which eliminates the oscillation of the output. The restriction that we put in place is to set the second differential of the first spline y_0'' and last spline y_N'' equal to a constant. We will use a constant of zero for now, but we come back to this constant at a later stage. Creating this additional restriction means that we are left with only one unknown, parameter b_2. This is demonstrated, using the constant zero, in Table 2 below.

X	**y** (***d***)	Using equation 4.3 we can derive ***a***	Using equation 4.6 we can derive ***b***	Using equation 4.5 we can derive ***c***
X_1	d_1	$\frac{b_2 - b_1}{3X_1}$	**0**	$-X_1\frac{(b_2+2b_1)}{3}+\frac{(d_2-d_1)}{X_1}$
X_2	d_2	$\frac{b_3 - b_2}{3X_3}$	***The only parameter left to solve for is b_1***	$-X_2\frac{(b_3+2b_2)}{3}+\frac{(d_3-d_2)}{X_2}$
X_3	d_3	$\frac{b_4 - b_3}{3X_4}$	$\frac{-2X_1b_2 - X_1b_1 - 2X_2b_2 - 3\frac{(d_2-d_1)}{X_1} + 3\frac{(d_3-d_2)}{X_2}}{X_2}$	$-X_3\frac{(b_4+2b_3)}{3}+\frac{(d_4-d_3)}{X_3}$
…	…	…	...	...
X_{N-1}	d_{N-1}	$\frac{b_N - b_{N-1}}{3X_{N-1}}$	$\frac{-2X_{N-3}b_{N-2} - X_{N-3}b_{N-3} - 2X_{N-2}b_{N-2} - 3\frac{(d_{N-2}-d_{N-3})}{X_{N-3}} + 3\frac{(d_{N-1}-d_{N-2})}{X_{N-2}}}{X_{N-2}}$	$-X_{N-1}\frac{(b_N+2b_{N-1})}{3}+\frac{(d_N-d_{N-1})}{X_{N-1}}$
X_N	d_N	N/A	**0**	N/A

Table 2

If we find a value for b_2 that results in a final value of zero for b_N then we have a single solution and this solution should eliminate the oscillation shown above. We can determine this solution using two different methods:

- iteration; or
- Gaussian Elimination of a tri-diagonal matrix.

Before we consider each of these solution techniques we consider first the requirement of a boundary condition in order to obtain a unique solution for a cubic spline. In our discussion above we ordained a boundary condition of $b_0 = b_N = 0$. In practise two boundary conditions have become widely accepted:

1. **Natural spline**
   In a natural spline the second differential at x_0 and x_N is set to zero.
   In other words $y_0'' = y_N'' = 0$.

2. **Clamped spline**

   In a clamped spline the first differential of the function that produced the nodes and the first derivative of the spline are set equal. In other words $y_{0'} = f(x_0)'$ and $y_{N'} = f(x_N)'$. It is immediately apparent when we construct a yield curve that we do not have a function that can be used to replicate the nodes. The first differential of this function is therefore not available. A reasonable approximation can be used based on the slope of the linear interpolation function between the first two and the last two nodes.

   Although this provides a reasonable approximation in most circumstances it is not always an appropriate measure. An incorrect choice of boundary values could result in spurious and oscillating results at the short and/or long end of the curve.

An example using the same input data but different (albeit rather extreme) boundary values is shown below in Exhibit 6. The natural boundary uses values zero and zero. In the clamped boundary we have used –50 and –50 as boundary values. Although these boundary values are extreme, they do illustrate the effect that inappropriate boundary values can have on spline results.

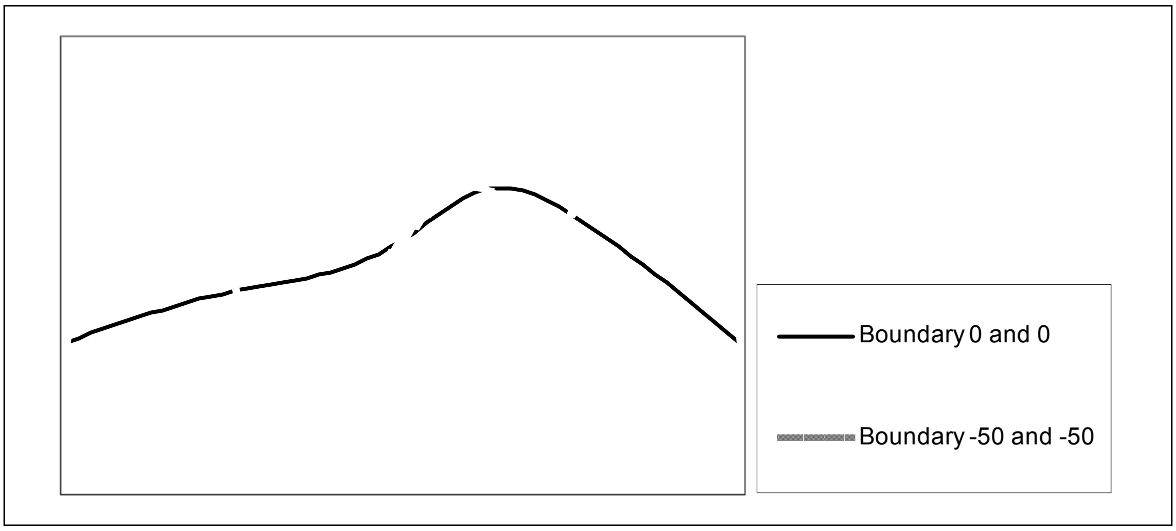

Exhibit 6

These results are not unexpected. Readers may question the practical difference between having a natural boundary condition against having a boundary condition that is obviously inappropriate. Both approaches may lead to oscillation and an incorrect result. The sole practical difference is that where we set our own boundary value, however inappropriate, the extent of the error is under our own control. For this reason users may prefer this approach.

<u>The solution</u>

We now consider each approach to obtaining the solution.

<u>Iterative solution</u>

A solution for b_1 can be obtained by iteration. This "trial-and-error" style approach is straightforward to understand but is not without its limitations.

When a cubic spline solution is solved by iteration for a single parameter, the degree of accuracy required is very high. In test solutions the authors found that a higher degree of accuracy was required for a higher number of nodes. A calculation for fifteen nodes or more required the solution to be accurate to at least eight decimal places. Even apparently negligible differences in decimal accuracy can result in strange spline parameters and in turn produce the same oscillation observed above when no boundary values were set. This is particularly evident at the long end of the curve as the error becomes compounded by previous inaccuracies, thus leading to yield curves of limited practical application when anything longer than the medium-term maturity range is modelled.

A fictional set of numbers have been used to demonstrate this point in Table 3 below. The "Date" column holds the maturity dates for each rate, while the "Rate" column is of course the set of interest rates for each particular term to maturity.

Date	**Rate (d)**	**parameter a**	**parameter b**	**parameter c**
1-Jan-00	6.000	- 0.00001228	0.00000000	0.00544212
7-Jan-00	6.030	0.00000351	- 0.00022106	0.00411577
31-Jan-00	6.050	- 0.00000019	0.00003181	- 0.00042615
1-Apr-00	6.100	- 0.00000001	- 0.00000235	0.00137086
1-Jul-00	6.200	0.00000002	- 0.00000426	0.00076898
1-Oct-00	6.250	- 0.00000001	0.00000117	0.00048462
1-Jan-01	6.300	0.00000000	- 0.00000042	0.00055340
1-Jul-01	6.400	- 0.00000000	0.00000083	0.00062739
1-Jan-02	6.520	0.00000000	- 0.00000126	0.00054853
1-Jan-03	6.610	- 0.00000000	0.00000004	0.00010301
1-Jan-05	6.700	0.00000000	0.00000000	0.00013362
1-Jan-06	6.750	- 0.00000000	0.00000003	0.00014328
1-Jan-07	6.800	0.00000000	- 0.00000010	0.00011518
1-Jan-10	6.900	- 0.00000000	0.00000014	0.00015545
1-Jan-11	6.960	0.00000000	- 0.00000020	0.00013152
1-Jan-12	7.000	- 0.00000000	0.00000023	0.00014041
1-Jan-14	7.100	0.00000000	- 0.00000047	- 0.00003778
1-Jan-15	7.050	- 0.00000000	0.00000013	- 0.00016286
1-Jan-20	7.000	0.00000000	- 0.00000004	0.00000616
1-Jan-25	6.950	- 0.00000000	0.00000002	- 0.00002600
1-Jan-30	6.950		0.00000000	

Table 3

This data is illustrated graphically at Exhibit 7 below.

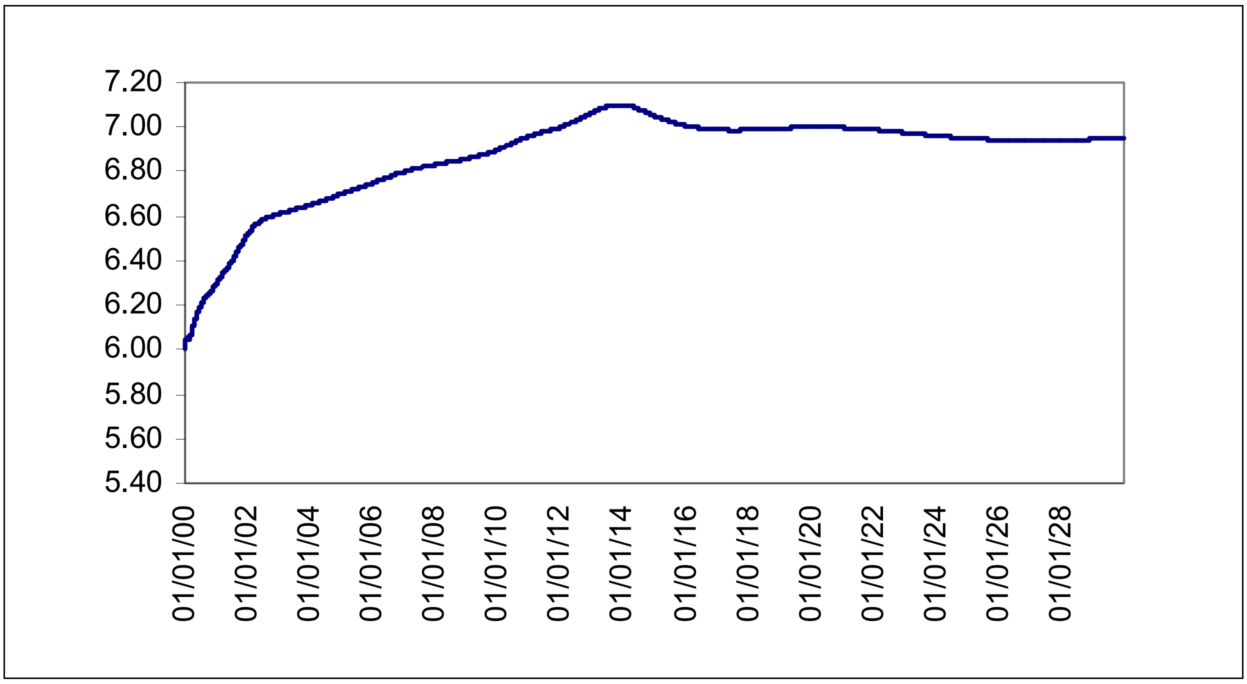

Exhibit7

In Table 3 above, an accuracy of eight decimal places is shown but in fact a much higher level (over 15 decimal places) of accuracy was required to calculate the results. When we adjust the level of accuracy, just on parameter b_1, to seven decimal places the results are significantly flawed, as shown in Table 4 below.[10]

Date	**Rate (d)**	**parameter a**	**parameter b**	**parameter c**
1-Jan-00	6.000	-0.00001228	0.00000000	0.00544210
7-Jan-00	6.030	0.00000351	- 0.00022105	0.00411580
31-Jan-00	6.050	-0.00000019	0.00003179	-0.00042640
1-Apr-00	6.100	-0.00000001	-0.00000230	0.00137252
1-Jul-00	6.200	0.00000002	-0.00000442	0.00076105
1-Oct-00	6.250	-0.00000002	0.00000174	0.00051482
1-Jan-01	6.300	0.00000002	-0.00000255	0.00044055
1-Jul-01	6.400	-0.00000006	0.00000695	0.00123776
1-Jan-02	6.520	0.00000008	-0.00002345	-0.00179846
1-Jan-03	6.610	-0.00000011	0.00006372	0.01289764
1-Jan-05	6.700	0.00000103	-0.00017986	-0.07200383
1-Jan-06	6.750	-0.00000419	0.00095266	0.21006837
1-Jan-07	6.800	0.00000395	-0.00363079	-0.76744773
1-Jan-10	6.900	-0.00006704	0.00936251	5.51451411
1-Jan-11	6.960	0.00028391	-0.06404843	-14.44584982
1-Jan-12	7.000	-0.00043548	0.24683078	52.26970709
1-Jan-14	7.100	0.00407923	-0.70817417	-284.97230573
1-Jan-15	7.050	-0.00230683	3.75858533	828.42777079
1-Jan-20	7.000	0.00741195	-8.87822401	-8,520.0324431
1-Jan-25	6.950	-0.02736125	31.74664902	33,260.580061
1-Jan-30	6.950		-118.13828171	

[10] The results were calculated using the "Goal Seek" function on Microsoft Excel.

Table 4

It can be seen that within the long dates, parameter *b* starts to oscillate and grow in an exponential manner. A graphical representation of the rates as a result of this flawed data is shown at Exhibit 8 below. Note that the oscillation error is highly pronounced.

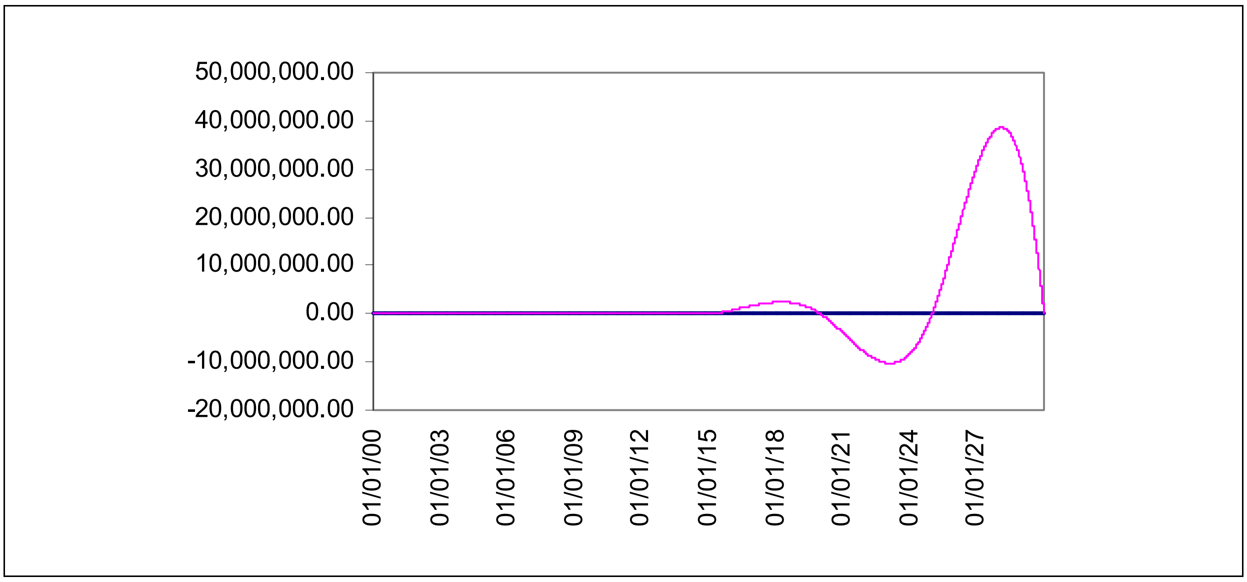

Exhibit 8

The degree of accuracy obtained through iteration is dependent on the starting point for the first calculation and the number of iterations allowed as a maximum. There is no way of ensuring that the required degree of accuracy will be obtained without undertaking very high magnitude (and process intensive) calculations in the iterative algorithm. Without the comfort of extensive manual review of the results by a person with a clear understanding of the calculation and its implications, we do not recommend the use of the iteration approach to derive a solution.

<u>Solving for a system of linear equations by elimination</u>

We now consider again equation (7) derived above, and re-arrange it slightly as (8).

$$X_{N+1}b_{N+2} + 2(X_N + X_{N+1})b_{N+1} + X_N b_N = -3\frac{(d_{N+1}-d_N)}{X_N} + 3\frac{(d_{N+2}-d_{N+1})}{X_{N+1}} \qquad (8)$$

It can be seen that all parameters *X* and *d* can be obtained by reference to values that are already known at the nodes. These are in fact node (or time-to-maturity) dependent constants. In other words we have a system of linear equations from node 1 to *N*. Readers will know that simultaneous linear equations can be solved by substitution. This method of solving linear equation can be applied to larger sets of linear equations, although we require increased processing power.

The system of equations can be represented in a N-2 by N+1 matrix as follows:

X_0	$2(X_0+X_1)$	X_1					$-3(\frac{(d_1-d_0)}{X_0}-\frac{(d_2-d_1)}{X_1})$
	X_1	$2(X_1+X_2)$	X_2				$-3(\frac{(d_2-d_1)}{X_1}-\frac{(d_3-d_2)}{X_2})$
		…	…	…			…
			…	…	…		…
				X_{N-2}	$2(X_{N-2}+X_{N-1})$	X_{N-1}	$-3(\frac{(d_{N-1}-d_{N-2})}{X_{N-2}}-\frac{(d_N-d_{N-1})}{X_{N-1}})$

In essence, if you look at the parameters *b* for which we are attempting to solve, this can be laid over the above matrix as follows:

b_0	b_1	b_2				
	b_1	b_2	b_3			
		…	…	…		
			…	…	…	
				b_{N-2}	b_{N-1}	b_N

In other words we are looking for a set of values for b_0 to b_N that will solve the linear system for each and every node *N*.

Our basic limitation imposed above is not lifted. We set b_0 and b_N equal to 0 in order to apply the natural boundary condition. We can then substitute our solution for equation/row 1 into equation/row 2. We perform a similar continuous set of substitutions until we have a solution for b_{N-1}. This solution can then be substituted backward through the solved equations to obtain a solution for b_1.

A matrix of this form, that is, an upper and lower triangular quadrant for which no value is required (observed by the grey shaded area) is also known as a *tri-diagonal matrix*. More advanced methods of solving matrices (and in particular tri-diagonal types) are available. It is outside the scope of this article to cover these methods in detail; interested readers may wish to consult Burden and Faires (1997).[11] For the purposes of illustration however, we have prepared a simple example solution for a small matrix of values, and this appears as an Appendix to this article.

The same values used for the iterative solution were processed using the elimination solution. The results and their illustrative chart are set out in Table 5 and Exhibit 9 respectively below.

[11] Op. cit.

Date	Rate (d)	parameter a	parameter b	parameter c
1-Jan-00	6.000	-0.00001228	0.00000000	0.00544212
7-Jan-00	6.030	0.00000351	-0.00022106	0.00411577
31-Jan-00	6.050	-0.00000019	0.00003181	-0.00042615
1-Apr-00	6.100	-0.00000001	-0.00000235	0.00137086
1-Jul-00	6.200	0.00000002	-0.00000426	0.00076898
1-Oct-00	6.250	-0.00000001	0.00000117	0.00048462
1-Jan-01	6.300	0.00000000	-0.00000042	0.00055340
1-Jul-01	6.400	-0.00000000	0.00000083	0.00062739
1-Jan-02	6.520	0.00000000	-0.00000126	0.00054853
1-Jan-03	6.610	-0.00000000	0.00000004	0.00010301
1-Jan-05	6.700	0.00000000	0.00000000	0.00013362
1-Jan-06	6.750	-0.00000000	0.00000003	0.00014328
1-Jan-07	6.800	0.00000000	-0.00000010	0.00011518
1-Jan-10	6.900	-0.00000000	0.00000014	0.00015545
1-Jan-11	6.960	0.00000000	-0.00000020	0.00013151
1-Jan-12	7.000	-0.00000000	0.00000023	0.00014041
1-Jan-14	7.100	0.00000000	-0.00000047	-0.00003779
1-Jan-15	7.050	-0.00000000	0.00000013	-0.00016284
1-Jan-20	7.000	0.00000000	-0.00000004	0.00000594
1-Jan-25	6.950	-0.00000000	0.00000002	-0.00002515
1-Jan-30	6.950		0.00000000	

Table 5

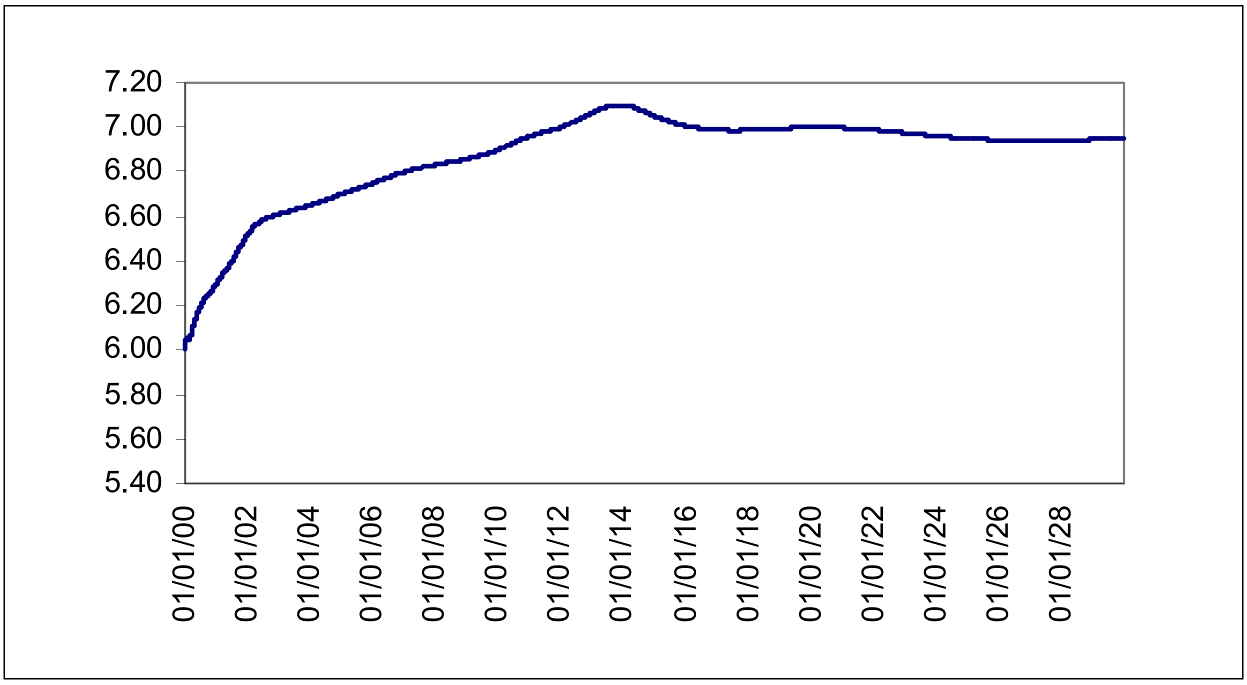

Exhibit 9

On first observation these values appear to be identical to those obtained using the iterative solution. In fact even at the highest level of accuracy possible in our iterative solution we notice a difference in the values for parameter **c** when we look at the dates 1 Jan 2014 onwards (which appear in the grey boxes in Table 5 above). Although this is not apparent in the chart, the results in the table where numbers appear with greater accuracy, show these and other small differences not shown in Exhibit 9.

Based on these results we conclude that the technique of solving for a system of linear equations is superior to an iterative solution. This is because:

- no starting point for the calculation needs to be determined by the user or the system;
- the accuracy of the solution is not dependent on the number of iterative calculations performed; and
- the results do not need the same degree of review to assess their accuracy.

This is not to say that this method is flawless. Even a tri-diagonal methodology is reliant on the degree of precision applied in its calculation. Modern computing hardware and software has limitations in the size or length of the floating point numbers that it can process. However if programmed with care, a typical application can deal with significantly large numbers.

EMPIRICAL PROOF OF PRECISION

In our cubic spline application (CUBED^3) we have chosen C++ as the programming language and we have used the C++ 'long double' variable type to store and process our values. A long double is usually anything between a 74 and 128 bit place holder, depending on the compiler and the system on which the calculations are performed. Applying some basic binary mathematics and allowing 1 bit for sign storage we can calculate:

2^{71} = 2,361,183,241,434,820,000,000

This should be sufficient to provide an adequate level of accuracy for most cubic spline calculations required of a zero curve application.[12] To test this we have performed empirical testing to corroborate our conclusion using a completely fictitious set of data that was designed to provide an extreme testing environment and data that is more sensitive to calculation anomalies than any likely to occur in real life.[13] Our fake input values were chosen to include:

- a large number of nodes (over 100);
- high oscillations at various points in the curve; and
- various points of flat data.

[12] This assurance is based on the fact that a typical yield curve application very, very rarely has more than 30 nodes. Any application where there are large node numbers may require higher levels of accuracy.

[13] In other words we use interest rate values that are extreme and unlikely to be observed in a yield curve in practice. Bond traders would be amused if one morning they discovered that the bond redemption yield curve looked anything like Exhibit 10.

A large number of tenors was chosen to compound any rounding errors that might occur as part of the elimination multiplier. Oscillation at various points in the curve are used to set up waves that can continue when they subsequently flow into areas of flat data and which would highlight errors, if they occur. Flat sections of the curve are used so that any errors become highly visible.

A graph of this extreme test data is set out in Exhibit 10 below:

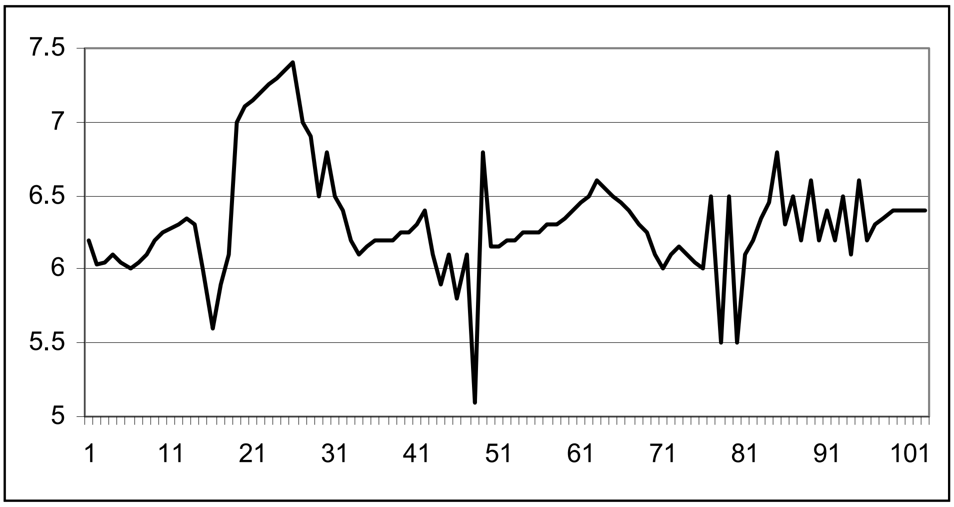

Exhibit 10

The resulting smooth graph after the cubic spline parameter have been calculated and applied looks as follows:

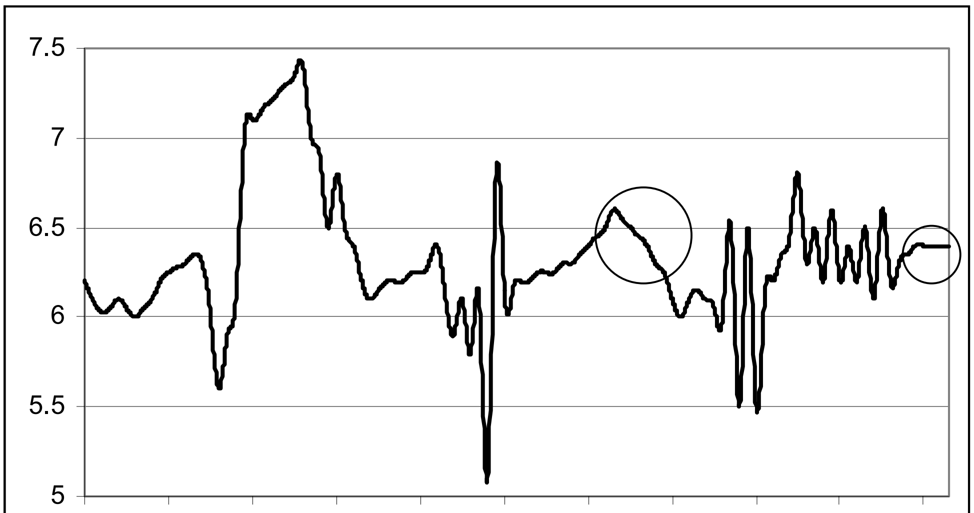

Exhibit 11

Two areas on the graph with relatively flat or consistent data values have been highlighted in Exhibit 11 as potential areas where calculation error may be observed. These areas of the graph are isolated and shown at Exhibit 12 and Exhibit 13 below.

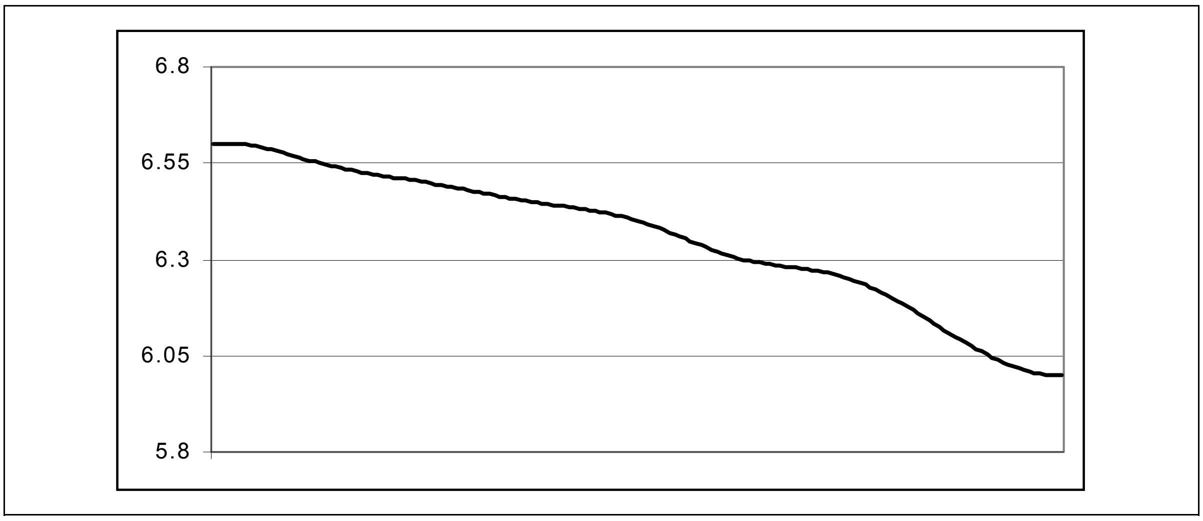

Exhibit 12

In the first area we observe some oscillation. However, this is not oscillation as a result of calculation errors. This is a smoothing effect that is required to meet the requirements of a cubic spline and to ensure a smooth curve. The data between points 63 and 71 is consistently downward sloping but the data then slopes upward again at point 72. The curve starts to "adapt" at an earlier stage in order to facilitate this change in direction. Therefore this behaviour is unavoidable, but under most applications for the spot curve does not present a material problem.

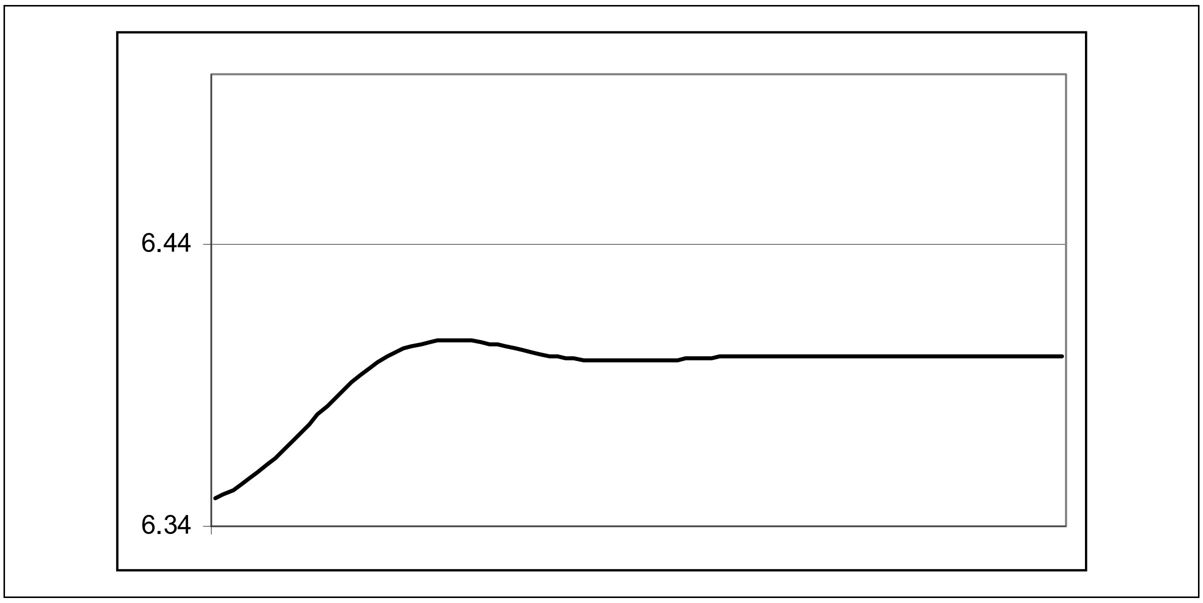

Exhibit 13

The second area of the curve provides another typical cubic spline example as the curve "adapts" to its new parameters. Once again this is a natural spline phenomenon and not an error in the calculated values.

Empirical data does not prove beyond a doubt that a cubic spline method, applied using an appropriate solution technique and precise software, will always produce accurate results. Nonetheless we believe that it is reasonable to assume from the test data set out above that the cubic spline methodology, used in conjunction with appropriate calculation tools, provides accurate zero curve results in most fixed income market conditions.

A LOOK AT FORWARD RATES

Previous literature has highlighted the use of the cubic spline approach to model forward curves, and its limitations. Certainly a cubic spline discussion would be incomplete without a look at its application to forward rates. We will use our empirical data to highlight typical forward rate behaviour under the cubic spline technique. Our sample data does not reflect actual market conditions and is an extreme data set, to say the least. However, it does highlight a point with regards to forward rates that can often be observed sometimes under normal market conditions. To this end we isolate the last sub-set of the data, as shown above, and plot the forward rates for that data set.

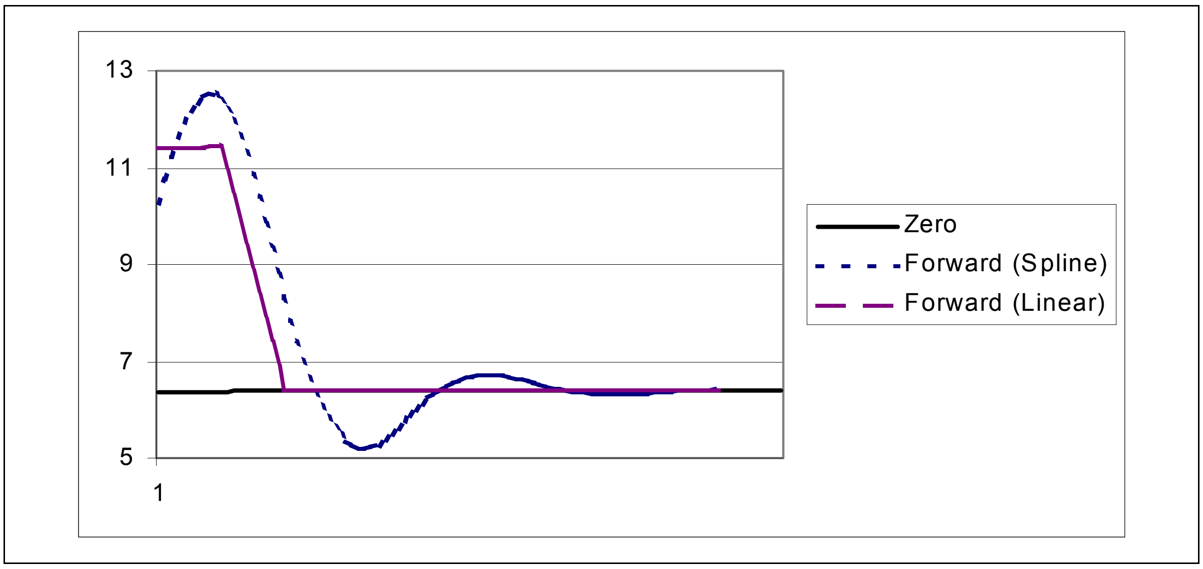

Exhibit 14

From data that was interpolated using the linear method versus data interpolated using the cubic spline, a comparison of forwards shows how the forwards in a cubic spline environment can oscillate. As expected, the relatively minor oscillations observed first in the zero rates curve are compounded excessively in the forward rate calculation. The linear interpolation approach, shown for comparison purposes at Exhibit X, eliminates much of the oscillation but of course is not a smooth curve, which is as undesirable. The user is confronted with the need to balance the conflicting requirements; a trade-off is called for and for most practical applications the cubic spline approach and its smoothing results is preferred. It remains important however that the user reviews cubic spline data by looking at both the zero and forward rates.

Using the actual United Kingdom 10-year zero curve for 2 January 2000, the forward rates have been calculated using cubic spline and linear interpolation and compared in Exhibit 15 and Exhibit 16 respectively below. There is no observed reason to favour the latter approach over the former.

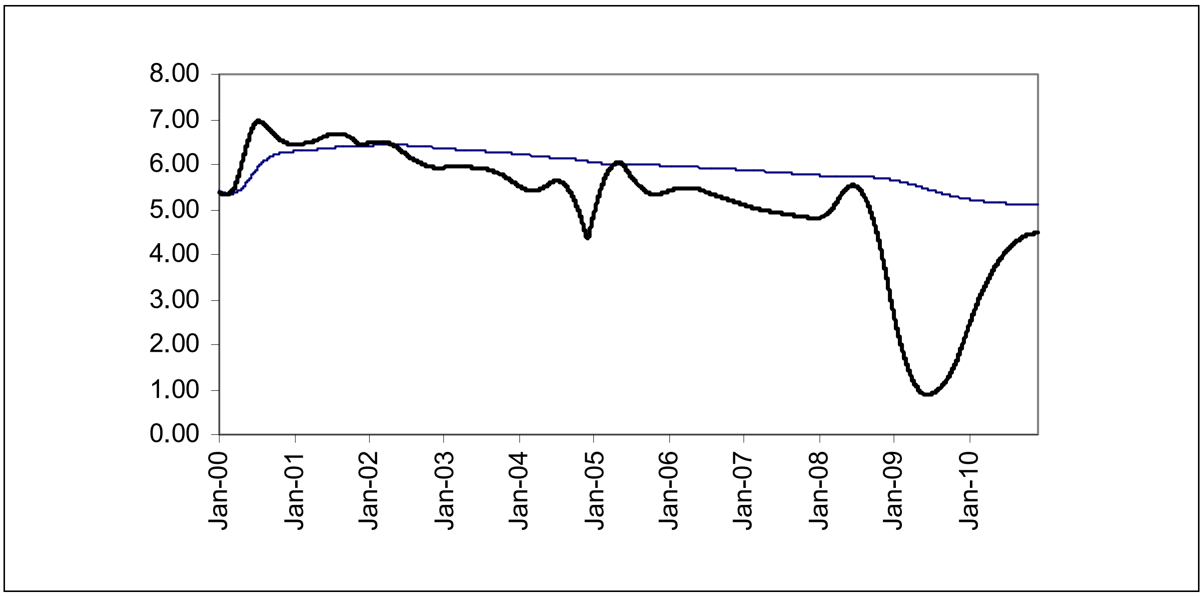

Exhibit 15

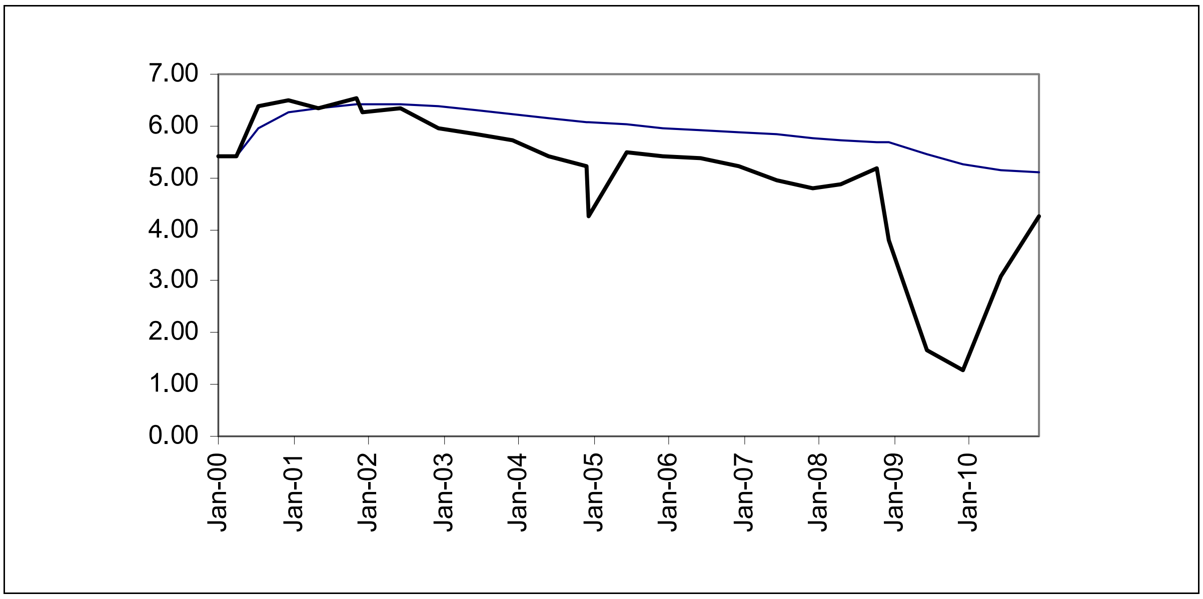

Exhibit 16

Improvements to the basic approach

As a result of the drawback when fitting the forward curve, the basic technique has been improved to remove the oscillation effect at longer maturities. As we saw from the test results presented earlier, the oscillation of a spline is partly a function of the number of nodes used. The paradox with this factor is that in practice, at very long maturities the forward (and also the spot) curve would be expected to be reasonably flat. To remove the oscillation, as described first by Fisher, Nychka and Zervos (1995), this involves the addition of a *roughness penalty* when minimising the sums of squares.[14] Waggoner (1997) introduced a *variable roughness penalty*, which enabled the approach to retain the flexibility at the short end and

[14] Fisher, M., Nychka, D., Zervos, D., "Fitting the Term Structure of Interest Rates with Smoothing Splines", Working Paper No. 95-1, *Finance and Economics Discussion Series*, Federal Reserve Board 1995

reduce oscillation at the long end.[15] Using the Waggoner approach enables users to retain the flexibility and ease of the cubic spline approach as well as a more realistic forward curve. Anderson and Sleath (1999) state that the advantage of the spline approach over parametric methods is that separate segments of the spline can be adjusted independently of each other.[16] The significance of this is that a change in market levels at one of the term structure will not affect significantly at other parts of the curve. This is a drawback of the parametric methods. Ironically Anderson and Sleath modify the Waggoner model in a way that would appear to incorporate elements of the parametric approach, and their results appear to improve on the earlier works.

CONCLUSION

The purpose of this paper has been to present an accessible account of how the cubic spline methodology of term structure estimation could be implemented by users involved in any area of the debt capital markets. The technique is straightforward and quick, and is valid for a number of applications, most of which are "normal" or conventional yield curves. For example users are recommended to use it when curves are positively sloping, or when the long end of the curve is not downward sloping. The existence of humps along the short or medium terms of the curve can cause excessive oscillation in the forward curve but the zero curve may still be used for valuation or relative value purposes.

Oscillation is a natural effect of the cubic spline methodology and its existence does not impair its effectiveness under many conditions. If observed rates produce very humped curves, the fitted zero-curve using cubic spline does not produce usable results. For policy making purposes, for example as used in central banks, and also for certain market valuation purposes, users require forward rates with minimal oscillation. In such cases however the Waggoner or Anderson-Sleath models will overcome this problem. We therefore recommend the cubic spline approach under most market conditions.

[15] Waggoner, D., "Spline Methods for Extracting Interest Rate Curves from Coupon Bond Prices", *Working Paper No. 97-10*, Federal Reserve Bank of Atlanta 1997

[16] Anderson, N., Sleath, J., "New Estimates of the UK Real and Nominal Yield Cuirves", *Bank of England Quarterly Bulletin*, November 1999, pp. 384-392

APPENDIX

Example matrix solution based on Gaussian elimination.

We will solve for the following values (where the values of *X* have already been calculated).

x	**X**	**y**
0.90	0.40	1.30
1.30	0.60	1.50
1.90	0.20	1.85
2.10	0.90	2.10
3.00	0.80	1.95
3.80	0.50	0.40
4.30		0.25

Firstly we construct our matrix as follows.

X_0	$2(X_{0+}X_1)$	X_1				$-3(\frac{(d_1-d_0)}{X_0}-\frac{(d_2-d_1)}{X_1})$
	X_1	$2(X_{1+}X_2)$	X_2			$-3(\frac{(d_2-d_1)}{X_1}-\frac{(d_3-d_2)}{X_2})$
			…	…	…	…
			X_{N-2}	$2(X_{N-2+}X_{N-1})$	X_{N-1}	$-3(\frac{(d_{N-1}-d_{N-2})}{X_{N-2}}-\frac{(d_N-d_N}{X_{N-1}}$

Where b_1 is set to zero this provides the values.

b1	**b2**	**b3**	**b4**	**b5**	**b6**	**b7**	
0.0	2.0	0.6					0.3
	0.6	1.6	0.2				2.0
		0.2	2.2	0.9			-4.3
			0.9	3.4	0.8		-5.3
				0.8	2.6	0.5	4.9

In turn we can substitute row 1 into row 2 to obtain.

b1	**b2**	**b3**	**b4**	**b5**	**b6**	**b7**	
0.0	2.0	0.6					0.3
	0.0	4.7	0.7				6.4
		0.2	2.2	0.9			-4.3
			0.9	3.4	0.8		-5.3
				0.8	2.6	0.5	4.9

Similar substitutions, and the fact that b_7 is constrained as zero, yield the matrix below.

b1	b2	b3	b4	b5	b6	b7	
0.0	2.0	0.6					0.3
	0.0	4.7	0.7				6.4
		0.0	51.4	21.3			-107.0
			0.0	172.9	45.7		-196.4
				0.0	516.2	0.0	1,258.0

This means that we can solve for b_6. Once we have a solution for $b_6$we can solve for b_5 and so on. As a final result we get the following values for parameter *b*.

b1	b2	b3	b4	b5	b6	b7
0.0	-0.338	1.544	-1.344	-1.780	2.437	0.0

Parameters *a* and *c* can be determined directly from the values of *b* above.