Natural Quartic Spline

Rafael E. Banchs

INTRODUCTION

This report describes the natural quartic spline algorithm developed for the enhanced solution of the Time Harmonic Field Electric Logging problem. As it can be seen in [1], the enhanced method uses a differentiator of second order for the computation of one of the partial results. When a cubic spline is used, subsequent application of such a differentiator leads to partial results with discontinuous derivatives, affecting the quality of the final result. It is for that reason that a quartic spline algorithm was developed. The use of a spline of order 4 produces partial results with continues first order derivatives, which notoriously improves the quality of the final result [1].

THE NATURAL QUARTIC SPLINE

A spline function is a piecewise polynomial function in which the composing polynomials satisfy some continuity conditions [2]. In a quartic spline function, continuity conditions are imposed to the function itself and to the first, second and third order derivatives. Spline functions are better suited for interpolating data than polynomials because their lack of the wild oscillations that polynomials of high degree often present. Although natural cubic splines are considered the "best" interpolating functions; due to the reason exposed above, the development of a natural quartic spline was required by our particular application.

When using a spline function for interpolation purposes, the known data values are used as the spline knots; which are those points where the polynomial pieces are joined together. Figure 1 shows an example of how a spline function can be used for interpolating data points. In it, the values y_0, y_1, ... y_n correspond to the known data points, and their abscissa values t_0, t_1, ... t_n are the

knots of the spline S(x). As it can be seen, the spline function is composed by n functions $S_i(x)$ each of one is defined over the interval $[t_i,t_{(i+1)}]$.

In the case of a quartic spline, the functions $S_i(x)$ are given by polynomials of fourth degree whose coefficients must be determined. That accounts for a total of 5n unknowns, which can be computed by imposing some conditions over the functions $S_i(x)$. As it was mention before, for a quartic spline, these conditions are the continuity of the spline function S(x) and its derivatives of first, second and third order. These continuity conditions are imposed at the knots.

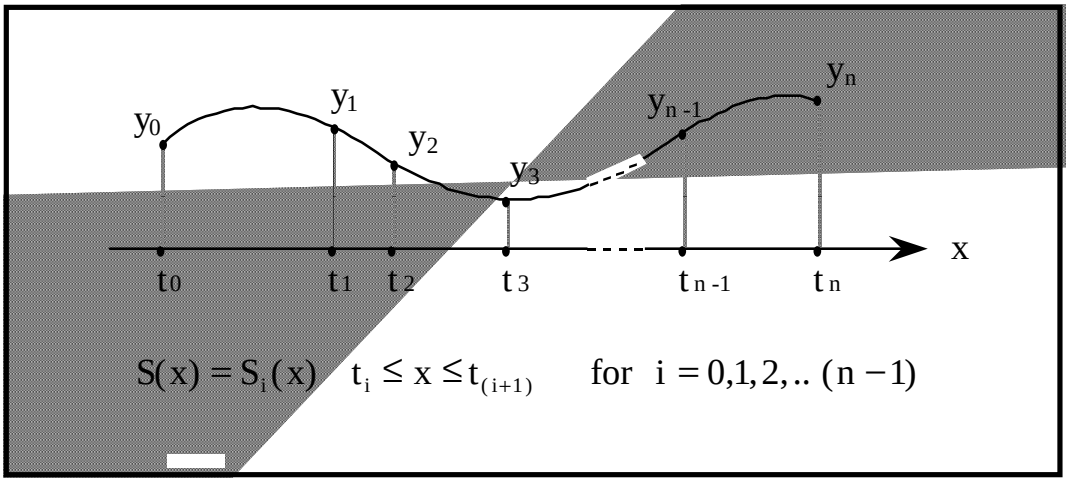

Figure 1: Interpolation by using Spline functions.

In addition to been continuous, the spline function S(x) must also attain the values of the data points y_0, y_1, ... y_n at the knots. These two conditions can be written as follows:

$$S(t_i) = y_i \qquad \text{for } i = 0,1,2,...n \tag{1.a}$$

$$S_i(t_{i+1}) = S_{i+1}(t_{i+1}) \qquad \text{for } i = 0,1,2,...n-2 \tag{1.b}$$

or, what is totally equivalent:

$$\left.\begin{aligned} S_i(t_i) &= y_i \\ S_i(t_{i+1}) &= y_{i+1} \end{aligned}\right\} \qquad \text{for } i = 0,1,2,...n-1 \tag{2}$$

And the continuity conditions over the derivatives are given by:

$$S_i'(t_{i+1}) = S_{i+1}'(t_{i+1}) \qquad \text{for } i = 0,1,2,...n-2 \tag{3}$$

$$S_i''(t_{i+1}) = S_{i+1}''(t_{i+1}) \qquad \text{for } i = 0,1,2,...n-2 \tag{4}$$

$$S_i'''(t_{i+1}) = S_{i+1}'''(t_{i+1}) \qquad \text{for } i = 0,1,2,...n-2 \tag{5}$$

As it can be verified, a total of 5n-3 independent equations are defined by the conditions imposed in (2), (3), (4) and (5). So, there are still three missing equations in order to complete the required system of equations. These equations can be obtained by imposing conditions to some of the derivatives of S(x) at the extreme knots t_0 and t_n.

A common choice is to make zero the highest continuous derivative in those points. When such a conditions is imposed, the resulting spline function is called a natural spline [2]. By doing that we obtain the following additional pair of equations:

$$S_0'''(t_0) = S_{n-1}'''(t_n) = 0 \tag{6}$$

Finally, the last equation will be provided by conditioning the second order derivative at t_n. As it will be seen latter, these has been done with the intention of gaining a little more control over the behavior of the spline function at large abscissa values. This is very important in our particular application because the data points to be interpolated are spaced in a logarithmic fashion. In this way, consecutive data points with larger abscissa values are farther from each other than consecutive data points with smaller abscissas. This increases the chances of the interpolating spline function to oscillate in the region of large abscissa values. At this moment, for simplicity, the second derivative at t_n will be equated to zero.

$$S_{n-1}''(t_n) = 0 \tag{7}$$

Then, a set of 5n equations, given in (2), (3), (4), (5), (6) and (7), with 5n unknowns have been defined. By solving it, the coefficients of the n functions $S_i(x)$ will be obtained and therefore the natural quartic spline function S(x) will be determined.

SIMPLIFICATION OF THE SYSTEM OF EQUATIONS

Although the system of equations have been already defined in the previous section, some algebra can be used in order to simplify it and improve the efficiency, from a computational point of view, of the final natural quartic spline algorithm.

First of all, let us define the set of variables z_i for i = 0,1,2...n as the values of the third order derivative of the spline function S(x) at the knots. That is:

$$z_i = S'''(t_i) \qquad \text{for } i = 0,1,2...n \tag{8}$$

Then, let us define the third derivative of the function $S_i(x)$ as follows:

$$S_i'''(x) = \frac{z_{i+1}}{h_i}(x - t_i) + \frac{z_i}{h_i}(t_{i+1} - x) \qquad \text{for } i = 0,1,2...n-1 \tag{9}$$

where h_i is given by $t_{i+1} - t_i$.

As it can be seen, the derivatives defined in (9) satisfy the continuity condition in (5). This can be easily verified by noticing that $S_i'''(t_i) = z_i$ and $S_i'''(t_{i+1}) = z_{i+1}$.

Expressions for the functions $S_i(x)$ can be obtained by integrating (9) three times. By doing so, gathering coefficients and trickily choosing the integration constants, the following expression is obtained:

$$S_i(x) = \frac{z_{i+1}}{24\,h_i}(x - t_i)^4 - \frac{z_i}{24\,h_i}(t_{i+1} - x)^4 + C_i(x - t_i)(t_{i+1} - x) + D_i(x - t_i) + E_i(t_{i+1} - x) \qquad \text{for } i = 0,1,2...n-1 \tag{10}$$

where the C_i's, D_i's and E_i's are coefficients to be determined in terms of the z_i's.

By imposing the continuity and data fitting conditions defined in (2) to (10), the following expressions for the coefficients D_i's and E_i's are found:

$$D_i = -\frac{z_{i+1}}{24}h_i^2 + \frac{y_{i+1}}{h_i} \qquad \text{for } i = 0,1,2...n-1 \tag{11.a}$$

$$E_i = \frac{z_i}{24} h_i^2 + \frac{y_i}{h_i} \qquad \text{for } i = 0,1,2...n-1 \tag{11.b}$$

Now, by differentiating (10), the first derivative of $S_i(x)$ can be computed:

$$S_i'(x) = \frac{z_{i+1}}{6 h_i} (x - t_i)^3 + \frac{z_i}{6 h_i} (t_{i+1} - x)^3 + C_i (t_{i+1} + t_i - 2x) + D_i - E_i \qquad \text{for } i = 0,1,...n-1 \tag{12}$$

from where, by replacing (11) and imposing the continuity condition defined in (3), the following equation is obtained:

$$\frac{h_{i-1}^2}{24} z_{i-1} + \frac{(h_i^2 - h_{i-1}^2)}{8} z_i - \frac{h_i^2}{24} z_{i+1} + C_i h_i + C_{i-1} h_{i-1} = \Delta_{i-1} - \Delta_i \qquad \text{for } i = 1,2,...n-1 \tag{13}$$

with Δ_i given by:

$$\Delta_i = \frac{y_{i+1} - y_i}{t_{i+1} - t_i} = \frac{y_{i+1} - y_i}{h_i} \qquad \text{for } i = 0,1,2,...n-1 \tag{14}$$

Similarly, but by differentiating twice, the second derivative is obtained:

$$S_i''(x) = \frac{z_{i+1}}{2 h_i} (x - t_i)^2 - \frac{z_i}{2 h_i} (t_{i+1} - x)^2 - 2 C_i \qquad \text{for } i = 0,1,2,...n-1 \tag{15}$$

from where, by imposing the continuity condition defined in (4), the next expression follows:

$$C_{i-1} = C_i + \frac{(h_i + h_{i-1})}{4} z_i \qquad \text{for } i = 1,2,...n-1 \tag{16}$$

Notice, just for verification, that an additional differentiation on (15) leads to the initial definition given in (9).

Next, by replacing (16) into (13) and gathering terms, the following expression is obtained:

$$\frac{h_{i-1}^2}{3} z_{i-1} + \chi_i^2 z_i - \frac{h_i^2}{3} z_{i+1} + 8 \chi_i C_i = 8 (\Delta_{i-1} - \Delta_i) \qquad \text{for } i = 1,2,...n-1 \tag{17}$$

with χ_i given by:

$$\chi_i = h_i + h_{i-1} = t_{i+1} - t_{i-1} \qquad \text{for } i = 1,2,...n-1 \tag{18}$$

Observe that (17) and (16) constitutes a system of 2n-2 equations with 2n-2 unknowns, which are the z_i's for i = 1,2...n-1 and the C_i's for i = 0,1,2...n-2. That is because the conditions imposed in (6) and (7) define the values of z_0, z_n and C_{n-1}. Indeed, it follows from (6) that $z_0 = z_n = 0$; and, from (6), (7) and (15) that $C_{n-1} = 0$.

In this way, by solving (16) and (17) and replacing into (10) along with (11), the spline function S(x) is finally obtained. Nevertheless, the solution of (16) and (17) can be performed in a very efficient way by using a recursive algorithm, which will be developed in the next section.

RECURSIVE COMPUTATION OF THE COEFFICIENTS

The particular characteristics of the equations obtained in the previous section allow the use of a recursive procedure in their solution. The use of such a procedure increases enormously the efficiency of the algorithm because matrix inversion is avoided.

First of all, notice that (16) provides a recursive means for the computation of the C_i's. In fact, by iterating on (16) itself, the following formula for the C_i's results:

$$C_i = C_{n-1} + \sum_{k=i+1}^{n-1} \frac{z_k \chi_k}{4} \qquad \text{for } i = 0,1,2,...n-2 \tag{19}$$

where the χ_i's are the same as defined in (18), and $C_{n-1} = 0$ as it was determined in the previous section.

Then, by replacing (19) into (17) and gathering some terms, the next expression is obtained:

$$\frac{h_{i-1}^2}{3} z_{i-1} + \chi_i^2 z_i + (2 \chi_i \chi_{i+1} - \frac{h_i^2}{3}) z_{i+1} + 2 \chi_i \sum_{k=i+2}^{n-1} \chi_k z_k = 8 (\Delta_{i-1} - \Delta_i) \quad \text{for } i = 1,2,...n-1 \tag{20}$$

which, by recalling that $z_0 = z_n = 0$, can be rewritten in matrix form as:

$$\begin{bmatrix} \chi_1^2 & 2\chi_1\chi_2 - \frac{h_1^2}{3} & 2\chi_1\chi_3 & \dots & 2\chi_1\chi_{n-1} \\ \frac{h_1^2}{3} & \chi_2^2 & 2\chi_2\chi_3 - \frac{h_2^2}{3} & \dots & 2\chi_2\chi_{n-1} \\ 0 & \frac{h_2^2}{3} & \chi_3^2 & \dots & 2\chi_3\chi_{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & \chi_{n-1}^2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{n-1} \end{bmatrix} = 8 \begin{bmatrix} \Delta_0 - \Delta_1 \\ \Delta_1 - \Delta_2 \\ \Delta_2 - \Delta_3 \\ \vdots \\ \Delta_{n-2} - \Delta_{n-1} \end{bmatrix} \tag{21}$$

As it can be seen from (21), the system matrix can be easily transformed to upper triangular by performing some row operations.

In order to simplify the notation let us denote the matrix, the unknown vector and the constant vector in (21) as **M**, **z** and **v** respectively. Then, as in the standard notation, $m_{i,j}$ will make reference to the element located at row i and column j of **M**, and z_i and v_i to the element located at position i of **z** and **v** respectively. By doing so, (21) becomes:

$$\mathbf{M}\,\mathbf{z} = \mathbf{v} \tag{22.a}$$

or, alternatively:

$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & \dots & m_{1,n-1} \\ m_{2,1} & m_{2,2} & m_{2,3} & \dots & m_{2,n-1} \\ 0 & m_{3,2} & m_{3,3} & \dots & m_{3,n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & m_{n-1,n-1} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{n-1} \end{bmatrix} = 8 \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{n-1} \end{bmatrix} \tag{22.b}$$

Then, the matrix **M** can be transformed to upper triangular by applying the forward elimination recursion described as Algorithm 1. Notice that any row operation on **M** must be performed on **v** too, in order to maintain the validity of the equation. Also notice than in every step of the

recursion, where a row operation is performed, some of the $m_{i,j}$'s and v_i's are modified (on place computations are performed).

---

for $k = 1$ to $n-2$

  $\text{factor} = \dfrac{m_{k+1,k}}{m_{k,k}}$

  for $j = k+1$ to $n-1$

    $m_{k+1,j} = m_{k+1,j} - \text{factor}\ m_{k,j}$

  end for

  $m_{k+1,k} = 0$

  $v_{k+1} = v_{k+1} - \text{factor}\ v_k$

end for

Algorithm 1: Forward Elimination

After executing Algorithm 1, (22) is reduced to an upper triangular system of equations in which the values of the z_i's can be easily computed by backward recursion. Algorithm 2 presents a backward recursion that simultaneously computes the C_i's by making use of (16). On place computation is also used in Algorithm 2.

In summary, the recursive procedure for computing the coefficients of the natural quartic spline S(x) must be implemented as follows:

1.- Initialization of the values of z_0 , z_n and C_{n-1}.
2.- Construction of the matrix system in (21) by using (20).
3.- Reduction of (21) to an upper triangular system by executing Algorithm 1.
4.- Computation of the C_i's and the z_i's by executing Algorithm 2.
5.- Computation of the D_i's and the E_i's by using (11).

6.- Determination of the $S_i(x)$'s by substituting the coefficients into (10).

$z_0 = 0;\quad z_n = 0;\quad C_{n-1} = 0;$
for $k = n-1$ to 1 step size -1
  $z_k = 8\, v_k$
  for $j = n-1$ to $k+1$ step size -1
    $z_k = z_k - m_{k,j}\ z_j$
  end for
  $z_k = \dfrac{z_k}{m_{k,k}}$
  $C_{k-1} = C_k + \dfrac{z_k\, \chi_k}{4}$
end for

Algorithm 2: Backward computation of the z_i's and C_i's.

NATURAL QUARTIC SPLINE WITH CONCAVITY APPROXIMATION

As it was mentioned before, in our particular application the data points to be interpolated are spaced logarithmicaly. Then, for large abscissa values, the spline function is more likely to oscillate. One way of reducing this problem is to choose an appropriate value for $S''_{n-1}(t_n)$ such that it approximates the concavity of the actual function being interpolated. In fact, the value of the concavity, or second derivative, at t_n can be properly approximated by finite differences. By doing so, condition (7) becomes:

$$S''_{n-1}(t_n) = \Psi_n = \frac{1}{t_n - t_{n-1}}\left(\frac{y_n - y_{n-1}}{t_n - t_{n-1}} - \frac{y_{n-1} - y_{n-2}}{t_{n-1} - t_{n-2}}\right) = \frac{1}{h_{n-1}}\left(\Delta_{n-1} - \Delta_{n-2}\right) \qquad (23)$$

which together with (6) and (15), defines the following new value for C_{n-1}:

$$C_{n-1} = -\frac{\Psi_n}{2} \tag{24}$$

The new value of C_{n-1} defined in (24) produces some little changes in the equations. Now, when (19) is replaced into (17) the next expression follows instead of (20):

$$8\ (\Delta_{i-1} - \Delta_i - \chi_i\ C_{n-1})$$
$$= \frac{h_{i-1}^2}{3}\ z_{i-1} + \chi_i^2\ z_i + (2\ \chi_i\ \chi_{i+1} - \frac{h_i^2}{3})\ z_{i+1} + 2\ \chi_i \sum_{k=i+2}^{n-1} \chi_k\ z_k \qquad \text{for } i = 1, 2, \ldots n-1 \tag{25}$$

As it can be seen from (25), the only component affected in (22) is the vector **v**, whose elements are now given by:

$$v_i = \Delta_{i-1} - \Delta_i - \chi_i\ C_{n-1} \qquad \text{for } i = 1, 2, \ldots n-1 \tag{26}$$

The rest of the procedure remains exactly the same as it was described in the previous section.

CONCLUSIONS

Although natural cubic splines are considered the “best” interpolating functions, the use of a differentiator of order two in the enhanced solution of the Time Harmonic Field Electric Logging problem make the use of a natural quartic spline more suitable. However, quartic spline algorithms are more complex and time consuming. Also, they are more likely to oscillate than cubic spline algorithms. This last problem can be substantially reduced by using the remaining free parameters in the set of equations for approximating better the behavior of the data. In this particular case, approximation of the concavity at large abscissa values was used.

REFERENCES

[1] Update Report #6: Enhanced Method for the Solution of the Potential Difference Integral.

[2] Cheney, W.; Kincaid, D. (1994), Numerical Mathematics and Computing. Brooks/Cole Publishing.