Project 1 : Not-A-Knot Cubic Spline Interpolation

Sujal Dave (200254193)
MAE,
North Carolina State University
Email: sdave@ncsu.edu

***Abstract*—Interpolation and data fitting are fundamental in scientific computing in order to turn the data points into meaningful functions that can later be used to deduce important parameters and understand trends. Interpolations are used in order to fit the discrete points exactly in a polynomial which can then be differentiated or integrated. Data fitting of discrete points is usually done by minimizing the error using Least-squares and fitting one polynomial through the set of points. A piecewise cubic polynomial interpolation approach is used in this manuscript. The interpolated polynomial using this method is smooth, twice differentiable and leads to a decreased error on increasing the number of points, unlike the polynomial interpolation where the errors and oscillations of the interpolant increases. The program is written in FORTRAN computer language. The Runge function is tested for a set of 5, 9, 17 and 33 equidistant points. The logarithm of error is plotted against the logarithm of length of interval which can be seen to decrease with increasing points as expected. The program is later extended to a problem of parametric cubic spline with repeated abscissae and ordinates and a circle of uni radius is interpolated using 9 points.**

I. Introduction

Data approximations are one of the most important tools in scientific computing because of their wide use in post-processing the results obtained from the simulations and make the discrete data into meaningful curves. The data that is obtained experimentally is almost always susceptible to the conditions in which the experiment is being conducted. Hence, a number of times the experiment is carried out and that gives a lot of such data in discrete points which needs to be processed in order to know what everything adds up to. Data fitting is then used to minimize the error between the points and estimate a linear or polynomial fit using Least-Squares so that the data can now be either validated or used to further process like, use differentiation or integration in order to look at the hidden parameters. Interpolation is another technique for post processing the discrete points where one estimates a polynomial which pass through the points exactly so that the curve can be used to find slope by differentiating or integrate to find the area under the curve. There are a number of methods with which one can estimate the polynomial passing through a set of data points. The major ones are Polynomial Interpolation and Piecewise Polynomial Interpolation.

A. Polynomial Interpolation

Assume that one needs to compute an interpolant of degree n, that interpolates (n+1) data points $\{(x_i, y_i), i = 0, 1...n\}$. Let the interpolant be denoted as

$$v(x) = \sum_{i=0}^{n} c_i B_i(x) = c_o B_0(x) + c_1 B_1(x)..... + c_n B_n(x) \quad (1)$$

where $\{c_i\}$ are the unknown coefficients or parameters determined from the known data or function and $\{B_i\}$ are the predetermined basis functions. This will give us the (n+1) conditions that the interpolating function needs to satisfy, in the matrix form is:

$$\begin{pmatrix} B_0(x_0) & B_1(x_0) & . & B_n(x_0) \\ B_0(x_1) & B_1(x_1) & . & B_n(x_1) \\ . & . & . & . \\ B_0(x_n) & B_1(x_n) & . & B_n(x_n) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ . \\ c_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ . \\ y_n \end{pmatrix} \quad (2)$$

For the Basis functions we have a number of choices based upon the simplicity of the use and the ease of construction and evaluation.

1) Monomial Interpolation: In this case, the Basis functions are assumed as follows and the resultant matrix B obtained is known as the Vandermonde matrix.

$$v(x) = p(x) = p_n(x) = \sum_{i=o}^{n} c_i x^i = c_0 + c_1 x + ... + c_n x^n \quad (3)$$

$$B = \begin{pmatrix} 1 & x_0 & . & (x_0)^n \\ 1 & x_1 & . & (x_1)^n \\ . & . & . & . \\ 1 & x_n & . & (x_n)^n \end{pmatrix} \quad (4)$$

The matrix B here, is very ill conditioned and hence for large n, the estimated coefficients c are prone to inaccuracies and thus suffer stability difficulties.

2) Lagrange Interpolation: Here, the Basis is selected in a Lagrangian manner and the resulting B matrix is an identity matrix

$$\begin{aligned} p_n(x) &= \sum_{i=0}^{n} y_i L_i(x) \\ L_i(x) &= \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \end{aligned} \quad (5)$$

$$B = \begin{pmatrix} 1 & 0 & . & 0 \\ 0 & 1 & . & 0 \\ . & . & . & 0 \\ 0 & 0 & . & 1 \end{pmatrix} = \boldsymbol{I} \quad (6)$$

The Lagrangian interpolation works well in practice but for very large n, it has issues with rounding error.

Figure 1 shows the polynomial interpolation for the Runge function and how we can see oscillations for increased number of points:

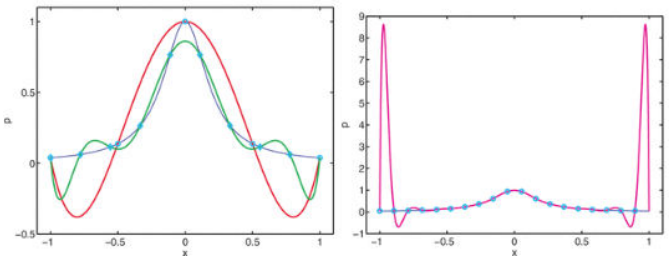

Fig. 1. Polynomial interpolation of Runge function with n=4,9 (left) and n=19 (right)

B. Piecewise Polynomial Interpolation

In this approach, we divide the interval [a,b] in to subintervals, $a = t_0 < t_1 < .... < t_r = b$ where, t_i are the breakpoints or knots. Then we construct the interpolant,

$$v(x) = s_i(x) \ , \ t_i \leq x \leq t_{i+1} \ , \ i = 0, 1... \ r-1 \tag{7}$$

where, $s_i(x)$ is a polynomial of low degree m (piecewise cubic, m=3).

As before, the polynomial v(x) must satisfy the interpolation conditions and a global smoothness property as well.

II. Description of Methodology

Here, we use the approach of the Piecewise Cubic Spline Interpolation with the global smoothness property of a Not-A-Knot condition.

Lets consider (n+1) points which form (n) sub-intervals over the given set of discrete points. Each sub-interval estimates a cubic polynomial passing through it which will be given by:

$$\begin{aligned} s_i(x) \ = \ & a_i + b_i(x - x_i) + c_i(x - x_i)^2 \\ & + d_i(x - x_i)^3 \ , \ x_i \leq x \leq x_{i+1} \end{aligned} \tag{8}$$

There are (4n) constants which will require (4n) set of equations. The interpolant must pass through each and every given data points. This requirement provides 2 equations per interval hence (2n) equations in total. Cubic splines are twice differentiable and hence that give more 2(n-1) equations. The two remaining equations are obtained from the global smoothness of not-a-knot condition.

A. Formulation of Cubic Spline

In order to interpolate a cubic spline in each sub-interval, the following steps and formula need to used:

Taking the first and second derivatives for Equation 8, we have

$$s_i^{'}(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2 \tag{9}$$

$$s_i^{''}(x) = 2c_i + 6d_i(x - x_i) \tag{10}$$

Let h denote the length of each subinterval, then

$$h_i = x_{i+1} - x_i \ , \ i = 0, 1, ...n - 1 \tag{11}$$

The first set of conditions for each interval is:

$$s_i(x) = f(x_i) \ \rightarrow a(x_i) = f(x_i) \tag{12}$$

$$s_i(x_{i+1}) = f(x_{i+1}) \ \rightarrow a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 \tag{13}$$

$$\therefore \ b_i + c_i h_i + d_i h_i = f[x_i, x_{i+1}] \ , \ i = 0, 1..n-2 \tag{14}$$

where, $f[x_i, x_{i+1}]$ is the Newton divided difference.

The first and the second derivative conditions give,

$$s_i^{'}(x_{i+1}) = s_{i+1}^{'}(x_{i+1}) \rightarrow b_i + 2c_i h_i + 3d_i h_i^2 = b_{i+1} \ , \ i = 0, 1...n-2 \tag{15}$$

$$s_i^{''}(x_{i+1}) = s_{i+1}^{''}(x_{i+1}) \rightarrow c_i + 6d_i h_i = c_{i+1} \ , \ i = 0, 1...n-2 \tag{16}$$

Hence, the coefficients b_i and d_i can be written as:

$$b_i = f[x_i, x_{i+1}] \ - \frac{h_i}{3}(c_{i+1} + 2c_i), \ i = 0, 1, ...n-1 \tag{17}$$

$$d_i = \frac{c_{i+1} - c_i}{3h_i} \tag{18}$$

Here, we can see that once we have all the c_i coefficients determined, we can find the rest with simple calculations as above. This means the problem is now reduced to solving for the coefficients $c_i$from the equations for i=0,1,..n-1

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_i = r_i \tag{19}$$

where,

$$r_i = \ 3(f[x_i, x_{i+1}] \ - \ f[x_{i-1}, x_i]) \tag{20}$$

B. Not-A-Knot Condition

In order to close this system of equations, we need two more equations that we will get from the global smoothness criteria.

The not-a-knot condition states that the third derivative at the first two and the last two intervals are equal, which gives us-

$$d_0 = d_1 \quad and \quad d_{n-1} = d_{n-2} \tag{21}$$

Solving for c_0 *and* c_n, we obtain the following:

$$c_0 = \ \frac{(h_1 + h_0)c_1 - h_0 c_2}{h_1} \tag{22}$$

$$c_n = \ \frac{(h_{n-1} + h_{n-2})c_{n-1} - h_{n-1}c_{n-2}}{h_{n-2}} \tag{23}$$

Hence the first and last equation in (19) will be as follows :

$$\frac{(h_1 + h_0)(2h_1 + h_0)}{h_1}c_1 + \frac{(h_1 - h_0)(h_1 + h_0)}{h_1}c_2 = r_1 \tag{24}$$

$$\begin{aligned} & \frac{(h_{-n-2} + h_{n-1})(h_{-n-2} - h_{n-1})}{h_{n-2}}c_{n-2} \\ & + \frac{(2h_{n-2} + h_{n-1})(h_{n-1} + h_{n-2})}{h_{n-2}}c_{n-1} = r_{n-1} \end{aligned} \tag{25}$$

Using the equations (24), (19) and (25), we can form a tridiagonal solver to solve for the coefficients c_i and then the subsequent coefficients can be found out from the equations which are given by (12), (17) and (18).

C. Interpolation Error

The interpolation error for the interpolated polynomial P(x) is given by

$$e = \int_{I_i} (P_i(x) - f(x))dx \tag{26}$$

In order to evaluate this integral, we make use of the Gauss Quadrature scheme. Since, the estimated polynomial is cubic, we can use 4 points to exactly determine the integral. The Gauss point formula and the Gauss points and weights are as listed in the Table 1.

$$\int_a^b f(x)dx = \sum_{i=1}^{N} w_i f(\xi_i) \tag{27}$$

TABLE I
GAUSS POINTS AND WEIGHTS

Gauss Points (ξ_i)	Weights (w_i)
-0.8611363116	0.3478548451
-0.3399810436	0.6521451549
0.3399810436	0.6521451549
0.8611363116	0.3478548451

In the present study, the L2 norm of the error is calculated and then the logarithm of the error norm is plotted against the logarithm of length of the interval. Subsequently, using least squares a linear fit is created to approximate the order of accuracy from the slope of the line.

$$\|e\|_2 = \sqrt{\sum_{i=1}^{\#\ of\ subintervals} \int_{I_i} (P_i(x) - f(x))^2 dx} \tag{28}$$

$$\log(\|e\|_2) = c + \alpha \log(h) \tag{29}$$

where, α gives the approximate order of accuracy for the approach.

D. Parametric Cubic Spline

Parametric cubic splines are constructed in a very similar to that of the cubic splines. Earlier, we estimated a single third order polynomial on an interval, whereas here, we estimate two third-order polynomials.

1) Approach: Since we have repeated values of abscissae and ordinates (as in case of the circle), for a closed curve(parametric) cubic spline, we introduce a third variable which takes care of the repeated nature of the data. [1]

Lets parameterize the (x,y) coordinates with a variable, say, t. Now, we represent the splines equations in terms of t, as follows:

$$\begin{aligned}&(t_0, x_0)(t_1, x_1)(t_2, x_2).....(t_n, x_n)\\ &(t_0, y_0)(t_1, y_1)(t_2, y_2).....(t_n, y_n)\end{aligned} \tag{30}$$

where, $t_0 < t_1 < ..... < t_n$ is a partition of the interval which is unique and not repeated and given by say,

$$t_i = \frac{i}{n}\ ,\ i = 0, 1, ...n \tag{31}$$

Then, the interpolation conditions are:

$$\begin{aligned}&X(t_i) = x_i \quad , \quad Y(t_i) = y_i \quad , \quad i = 0, 1, ...n\\ &C = (X(t_i), Y(t_i) = y_i)\end{aligned} \tag{32}$$

where, C is the new parameterized curve. [2]

III. RESULTS

A. Runge Function

The program for the cubic splines was written in FORTRAN and the plots were visualized in TecPlot.

The following plots show the Runge function using the not-a-knot piecewise cubic polynomial interpolation.

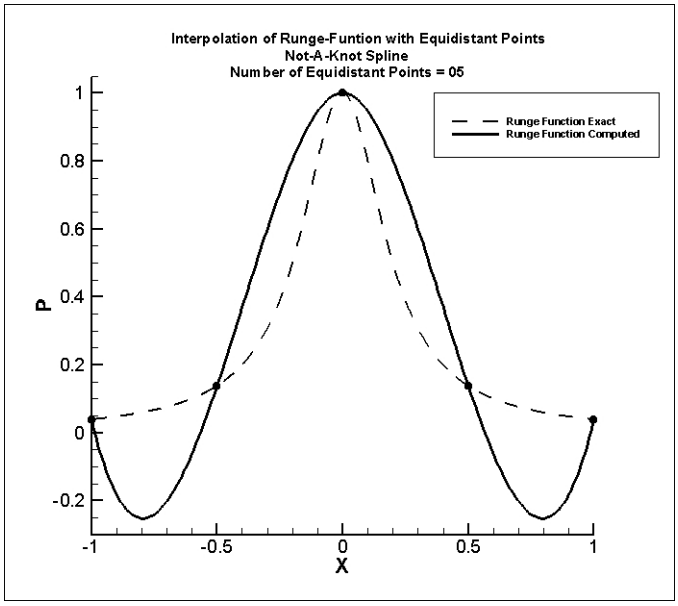

Fig. 2. N=5

We can see that for the case of N=5 equidistant points, the polynomial is almost as seen in the Figure 1. As we start increasing the number of points, the polynomial in this case shows a reduced error and the approximation is lot better than the schemed showed in Figure 1. In all the graphs, the discrete points chosen were equidistant.

The error for the different interval sizes were calculated and are as presented in Table 2.

TABLE II
ERROR AT DIFFERENT INTERVALS

Interval Size (h)	Error
0.5	0.194825
0.25	1.28065E-02
0.125	4.8498E-04
6.25E-02	4.4396E-05

In order to get the hidden details from the error values, we can find the L2 norm for the error as discussed above and after taking the logarithm for the same, we can plot it against logarithm of the interval size.

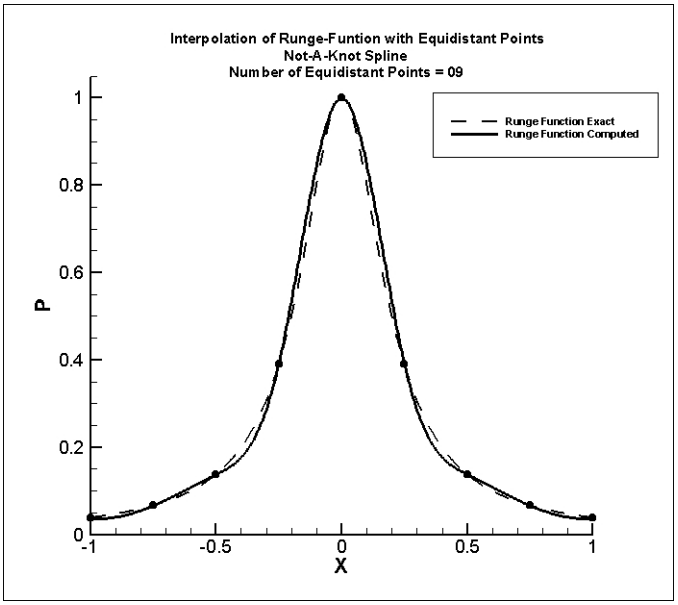

Fig. 3. N=9

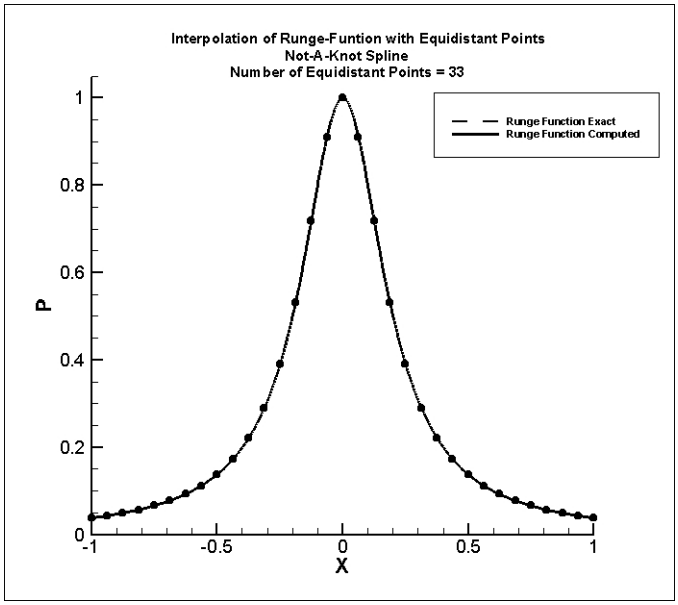

Fig. 5. N=33

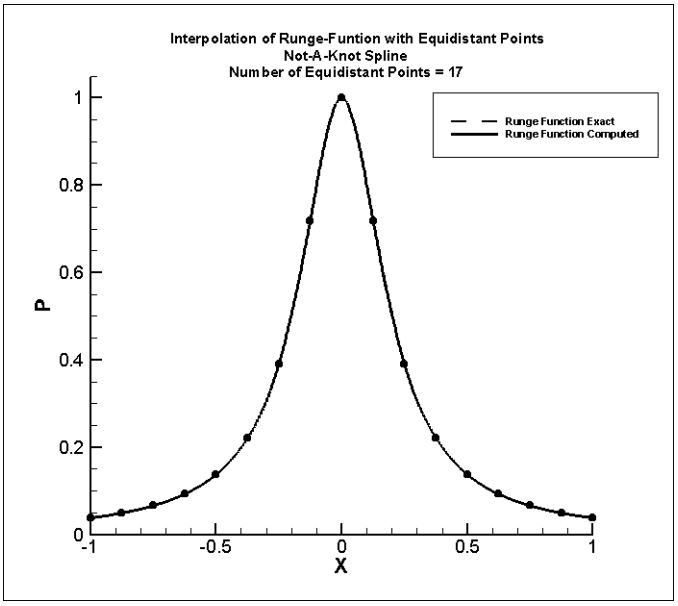

Fig. 4. N=17

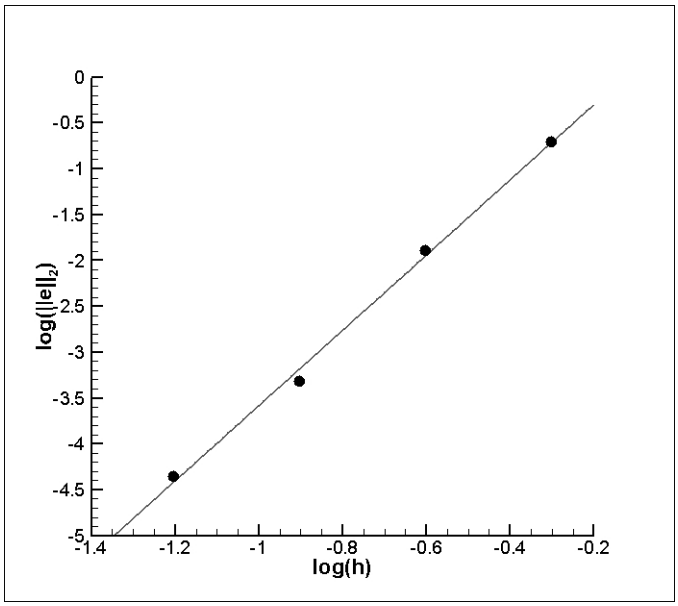

Fig. 6. Log(error) vs Log(h)

Figure 6 shows the points along with the Linear fit which is done using the Least Squares method.

The linear fit is obtained by solving the normal equations, given by:

$$A^T Ax \;=\; A^T b \tag{33}$$

Upon calculating the slope for the equation, from the formula in equation (29), we arrive at

$$\alpha \;=\; 3.6$$

In reality, the value of α should be equal to 4 because the not-a-knot method is 4th order accurate. We get a slightly lesser number due to the round off errors, Gauss quadrature error and the error arising from the methods of convergence used for solving the linear system of equations.

B. Circle

Using the approach and equations of the parametric spline as listed in the section above, the circle formed is shown in Figure 7.

The circle is very smooth with very slight discontinuity at the point (1,0) which is barely visible to the naked eye but this discontinuity can be removed by taking two or more points

really close to the starting/ending point which will ensure a proper polynomial fit even at the repeated points.

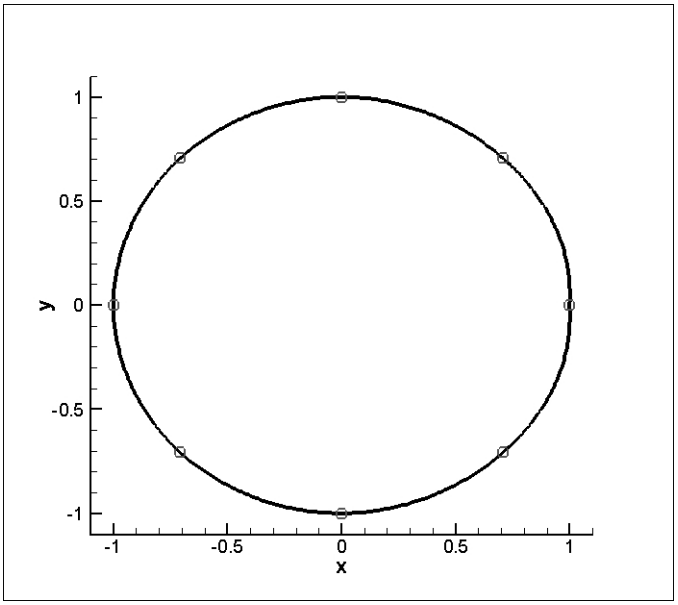

Fig. 7. Circle using Parametric Spline Interpolation

IV. Conclusion

From the current project for the course, the following conclusions have been made:

1) The program developed for Not-A-Knot cubic spline interpolation works accurately as seen from the results above.
2) Unlike polynomial interpolants with the Runge function, the cubic spline interpolant leads to an accurate representation of the function.
3) With increase in number of equally spaced sub-intervals, the cubic spline interpolant approaches the exact analytical function.
4) The linear fit for log($\|e\|_2$) vs log(h), (h=length of sub interval) was expected to have a slope of 4.
   However, the linear fit has a slope of 3.6. The deviation is due to the Gauss quadrature error and the ill-conditioning of the linear system of equations.

References

[1] J. H. Ahlberg, E. N. Nilson, and J. L. Walsh, *The Theory of Splines and Their Applications*, and others, Ed. Elsevier.
[2] U. M. Ascher and C. Greif, *First Course in Numerical Methods*, and others, Ed., 2011.