Rio Grande Cubic Spline Interpolation

Tyler Leibengood, Rabin Thapa, Bishal Lamichhane

October 2019

1 Introduction

This project involves choosing a random, continuous curve (that passes vertical line test) and interpolating its sampled data points to generate a function that approximates the curve. For the purpose of interpolating those points, **Cubic Spline Interpolation** is employed. From a set of n+1 data points [(x_i, y_i); i=0,1,2,...,n] cubic spline interpolation generates n cubic polynomials spanning the intervals $[x_i, x_{i+1}]$. The polynomials result to a piece-wise smooth, continuous function.

And the random curve to be approximated is chosen to be the Rio Grande border. **Rio Grande** is the river (hence continuous!) that serves as a natural boundary between Texas and Mexico (as shown in fig. 1).

2 Getting the points from the map

The list of curve points were gathered by opening the source image in Microsoft Paint. Roughly evenly spaced points in euclidian distance were taken. The software x metric, I called "right", and the software y metric, I called "down", components of each point were recorded using the cursor position indicator at the bottom of the Paint window. From these coordinates, the origin was repositioned and reflected so the positive y direction pointed up instead of down with the origin at the bottom left corner of the image. $x \to x$ and $y \to 500 - y$. Next we scaled the coordinates down by 100. $x \to \frac{x}{100}$ and $y \to \frac{y}{100}$. Finally, the points were translated left to be more near the origin. $x \to x - 1$. This provided the final data used for interpolation. The x and y points used for the interpolation are shown in Table 5 in the appendix.

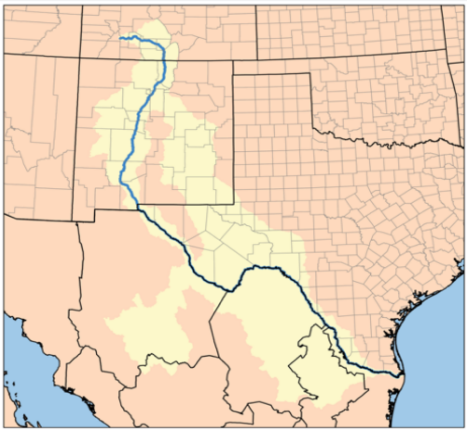

Figure 1: Map showing Rio Grande

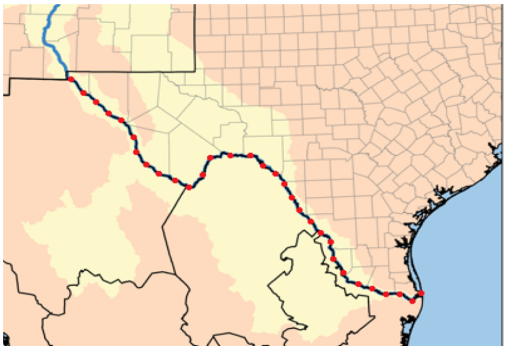

Figure 2: Portion of Rio Grande used for interpolation

3 Interpolating the points

For n cubic polynomials, there are 4n coefficients $[a_i, b_i, c_i, d_i$; i=0 to n-1] to be determined. So, we need 4n equations for those coefficients to have a unique solution.

- Equating the functional values at nodes with the data points gives us 2n equations
- Imposing the continuity of first derivative at nodes (except end points) gives us n-1 equations
- And, imposing the continuity of curvature at nodes (except end points) gives us n-1 equations
- Finally, using the natural form for cubic spline i.e. setting second derivatives at end points to be zero gives us 2 more equations

Upon solving for the coefficients using the above-mentioned conditions, we get a linear tridiagonal system which is then solved using a customized MATLAB code (attached in Appendix) to get the values of those coefficients.

Here is one example of formula for cubic splines using three points. Table 1 shows the coefficients and points used to get this formula. For $n = 2$ i.e. taking 3 data points, we get

the following spline:

$$S(x) = \begin{cases} 2.57 - 0.394(x - 0.50) - 0.044(x - 0.50)^3 & 0.5 \leq x \leq 2.13 \\ 1.74 - 0.741(x - 2.13) - 0.213(x - 2.13)^2 + 0.061(x - 2.13)^3 & 2.13 \leq x \leq 3.29 \end{cases}$$

j	xj	aj	bj	cj	dj
0	0.5	2.57	-0.39353	0	-0.043535
1	2.13	1.74	-0.74054	-0.21289	0.061174
2	3.29	0.69			

Table 1: Coefficients using three data points

Similarly, we have used six, nine, sixteen, and thirty one points as shown in Figure 5-Figure 10 to interpolate using natural cubic splines. The coefficients for these different cubic splines are included in the appendix(Table 3 - Table 5).

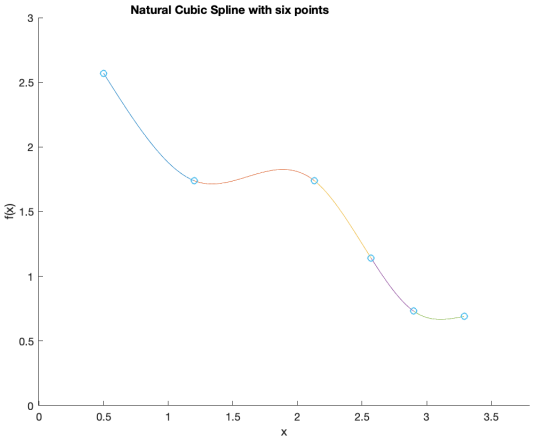

Figure 3:

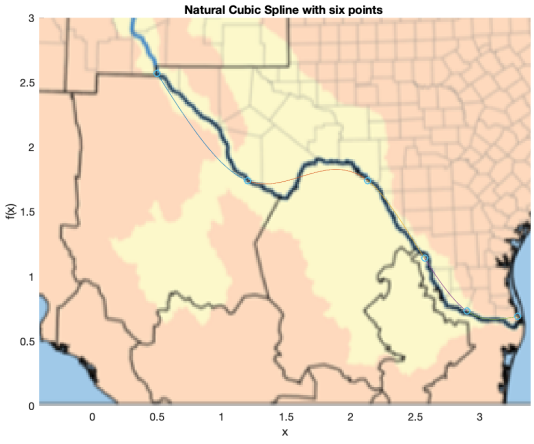

Figure 4:

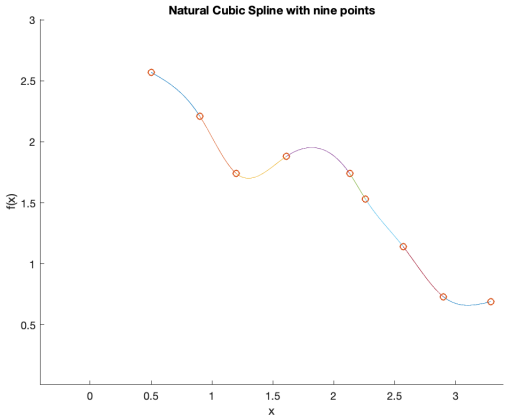

Figure 5:

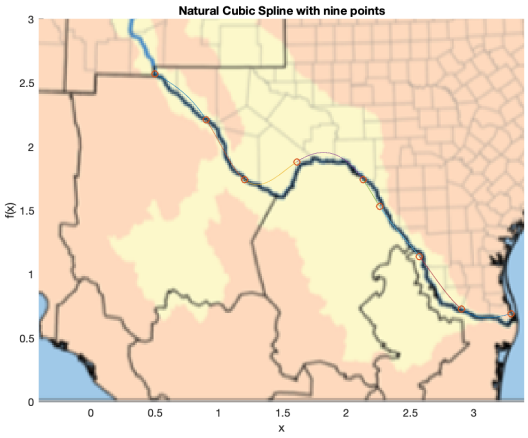

Figure 6:

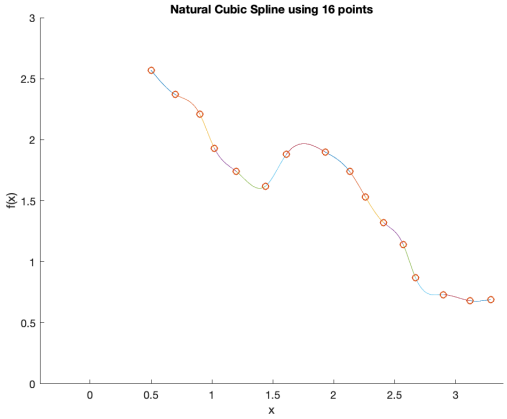

Figure 7:

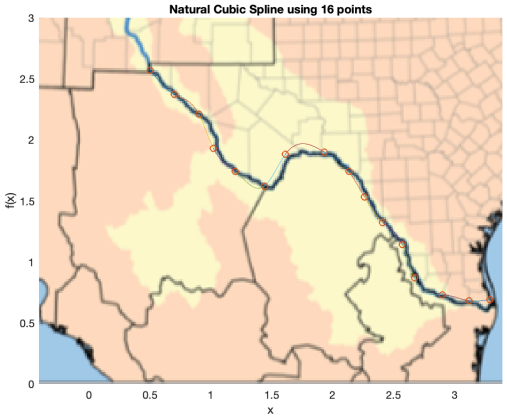

Figure 8:

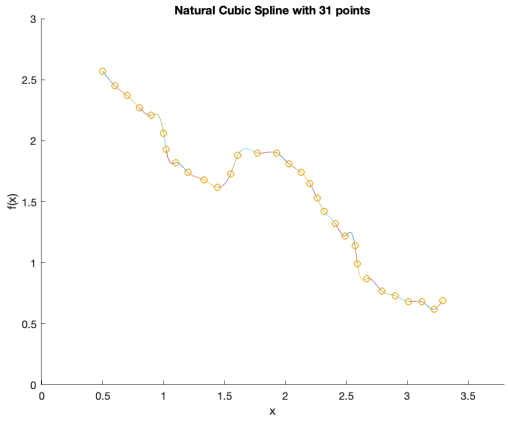

Figure 9:

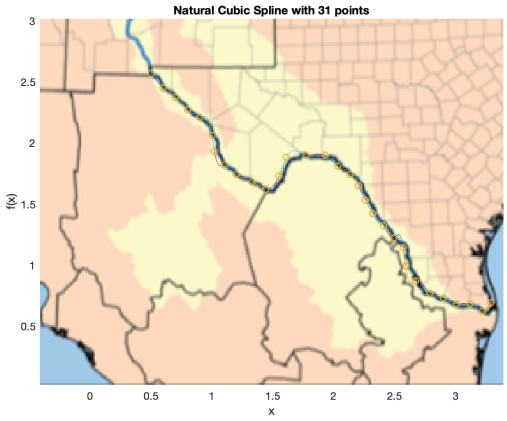

Figure 10:

j	xj	aj	bj	cj	dj
0	0.5	2.57	-1.5818	0	0.80836
1	1.2	1.74	-0.39352	1.6976	-1.3704
2	2.13	1.74	-0.7917	-2.1257	1.877
3	2.57	1.14	-1.5722	0.35189	1.9618
4	2.9	0.73	-0.69902	2.2941	-1.9607
5	3.29	0.69			

Table 2: Coefficients using six data points

4 Conclusion

Different numbers (3, 6, 9, 16, 31) of data points were taken from the map and they were used to generate the interpolating cubic splines. The coefficients of those splines were calculated using MATLAB code. Then, those splines were plotted on top of the actual curve. It can be seen clearly that taking more points results in splines that are better approximation of the curve.

Appendices

A Coefficient tables

j	xj	aj	bj	cj	dj
0	0.5	2.57	-0.5687	0	-2.0706
1	0.9	2.21	-1.5626	-2.4847	8.2373
2	1.2	1.74	-0.82938	4.9288	-5.0564
3	1.61	1.88	0.66233	-1.2905	-0.96335
4	2.13	1.74	-1.4613	-2.7933	12.369
5	2.26	1.53	-1.5605	2.0304	-3.4029
6	2.57	1.14	-1.2827	-1.1343	3.8071
7	2.9	0.73	-0.78758	2.6347	-2.2518
8	3.29	0.69			

Table 3: Coefficients using nine data points

j	xj	aj	bj	cj	dj
0	0.5	2.57	-1.2153	0	5.3826
1	0.7	2.37	-0.56939	3.2296	-21.913
2	0.9	2.21	-1.9071	-9.9183	53.056
3	1.02	1.93	-1.9955	9.1817	-21.998
4	1.2	1.74	-0.82832	-2.6972	16.939
5	1.44	1.62	0.80398	9.4985	-30.772
6	1.61	1.88	1.3655	-6.1953	6.6353
7	1.93	1.9	-0.56108	0.17462	-6.8462
8	2.13	1.74	-1.3128	-3.9331	12.349
9	2.26	1.53	-1.7093	0.88283	7.8614
10	2.41	1.32	-0.91381	4.4205	-35.878
11	2.57	1.14	-2.2547	-12.801	83.473
12	2.67	0.87	-2.3106	12.241	-21.05
13	2.9	0.73	-0.020312	-2.2833	6.1025
14	3.12	0.68	-0.13887	1.7444	-3.4203
15	3.29	0.69			

Table 4: Coefficients using 16 data points

j	xj	aj	bj	cj	dj
0	0.5	2.57	-1.3215	0	12.152
1	0.6	2.45	-0.95695	3.6457	-20.762
2	0.7	2.37	-0.85066	-2.5828	10.894
3	0.8	2.27	-1.0404	0.68542	37.186
4	0.9	2.21	0.21225	11.841	-289.64
5	1	2.06	-6.1086	-75.05	2774
6	1.02	1.93	-5.7818	91.392	-453.84
7	1.1	1.82	0.12721	-17.529	82.573
8	1.2	1.74	-0.90148	7.2425	-29.679
9	1.33	1.68	-0.52318	-4.3325	37.545
10	1.44	1.62	-0.11343	8.0574	18.77
11	1.55	1.73	2.3405	14.251	-193.23
12	1.61	1.88	1.9638	-20.53	56.485
13	1.77	1.9	-0.26781	6.5825	-30.679
14	1.93	1.9	-0.51758	-8.1436	43.194
15	2.03	1.81	-0.85048	4.8146	-33.099
16	2.13	1.74	-0.88051	-5.115	-9.6229
17	2.2	1.65	-1.7381	-7.1358	46.171
18	2.26	1.53	-2.0957	1.175	53.3
19	2.32	1.42	-1.3791	10.769	-86.573
20	2.41	1.32	-1.5444	-12.606	203.57
21	2.49	1.22	0.34726	36.251	-663.65
22	2.57	1.14	-6.5946	-123.03	3887.8
23	2.59	0.99	-6.8502	110.24	-542.09
24	2.67	0.87	0.38087	-19.856	81.148
25	2.79	0.77	-0.87901	9.3571	-42.472
26	2.9	0.73	-0.36218	-4.6587	34.718
27	3.01	0.68	-0.12683	6.7982	-51.32
28	3.12	0.68	-0.49415	-10.137	90.789
29	3.22	0.62	0.20204	17.099	-81.425
30	3.29	0.69			

Table 5: Coefficients using 31 data points

B MATLAB code

```
% Natural cubic spline interpolation
n = input('Enter n for (n+1) nodes, n:  ');
x = [0.5  0.9  1.2 1.61 2.13 2.26  2.57  2.9  3.29];
a = [2.57 2.21  1.74 1.88 1.74 1.53  1.14  0.73  0.69];

m = n - 1;
h = zeros(1,m+1);
for i = 0:m
  h(i+1) = x(i+2) - x(i+1);
end
xa = zeros(1,m+1);
for i = 1:m
  xa(i+1) = ...
      3.0*(a(i+2)*h(i)-a(i+1)*(x(i+2)-x(i))+a(i)*h(i+1))/(h(i+1)*h(i));
end
xl = zeros(1,n+1);
xu = zeros(1,n+1);
xz = zeros(1,n+1);
xl(1) = 1;
xu(1) = 0;
xz(1) = 0;
for i = 1:m
  xl(i+1) = 2*(x(i+2)-x(i))-h(i)*xu(i);
  xu(i+1) = h(i+1)/xl(i+1);
  xz(i+1) = (xa(i+1)-h(i)*xz(i))/xl(i+1);
end
xl(n+1) = 1;
xz(n+1) = 0;
b = zeros(1,n+1);
c = zeros(1,n+1);
```

```
30  d = zeros(1,n+1);
31  c(n+1) = xz(n+1);
32  for i = 0:m
33    j = m-i;
34    c(j+1) = xz(j+1)-xu(j+1)*c(j+2);
35    b(j+1) = (a(j+2)-a(j+1))/h(j+1) - h(j+1) * (c(j+2) + 2.0 * c(j+1)) / ...
          3.0;
36    d(j+1) = (c(j+2) - c(j+1)) / (3.0 * h(j+1));
37  end
38  fprintf('\nThe numbers x(0), ..., x(n) are:\n');
39  for i = 0:n
40    fprintf('   %5.4f', x(i+1));
41  end
42  fprintf('\n\nThe coefficients of the spline on the subintervals are:\n');
43  fprintf('     a(i)        b(i)          c(i)          d(i)\n');
44  for i = 0:m
45    fprintf('%11.8f  %11.8f  %11.8f  %11.8f ...
          \n',a(i+1),b(i+1),c(i+1),d(i+1));
46  end
47 hold on                          %% Graph Spline
48 for i = 1:n
49     xx = x(i):.01:x(i+1);
50     cubic=a(i)+b(i)*(xx-x(i))+c(i)*(xx-x(i)).^2+d(i)*(xx-x(i)).^3;
51     plot(xx,cubic)
52 end
53 plot(x,a,'o')
54 I=imread('Map.png');
55 h=image([-0.993,3.973],[5.007936,0.007936],I);
56 uistack(h,'bottom')
57 axis([0 3.5 0 3])
58 axis equal
59 hold off
60
61 %%
```

```
62 xlim
63 ylim
64 end
```