Numerical Methods and Analysis: Mid-Term Exam

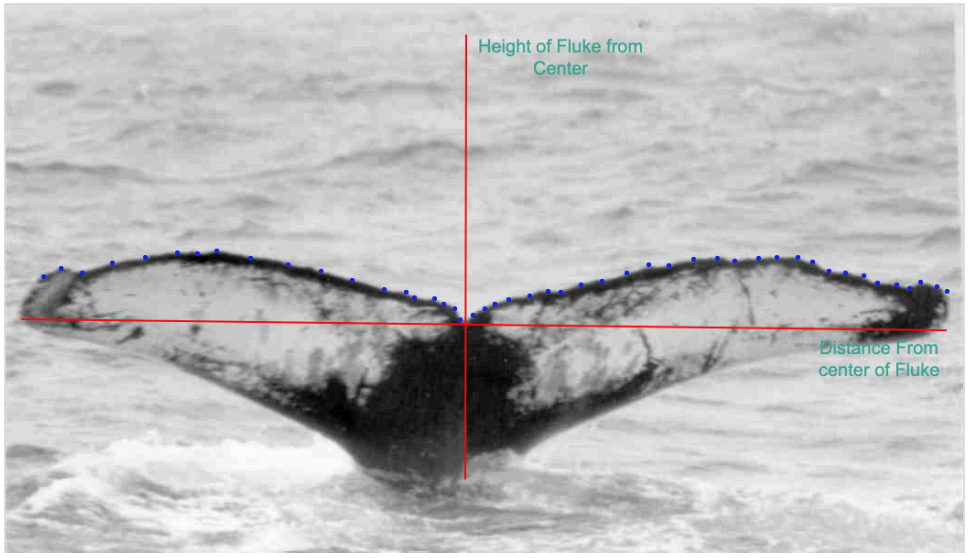
Nolan Dahlman

May 24, 2022

Analysis

The idea for this project is to be able to compare pictures of whale flukes to each other in order to properly identify a whale. The approach I will be taking is modeling a general function to the end of an individual whale's fluke that generalizes the shape of all whale flukes. For example this whale fluke from [Kaggle(2018)].

We can create a function that models the upper edge of the tail with the origin of the graph at the center of the two halves of the tail. Our graph of the data points will look like so

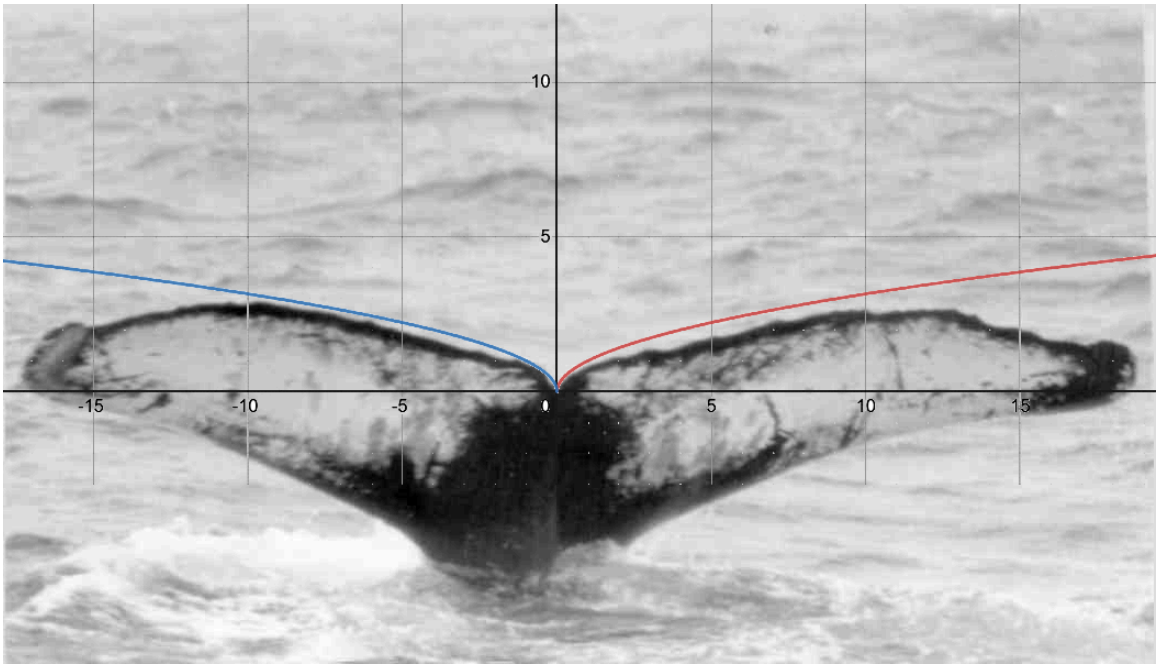


When doing this approximation it is important to note that these pictures of whale flukes will all be taken at different angles, as it may not be possible to get a picture straight on when the fluke is visible. To ensure that each whale fluke is accurately mapped to this function each time we will require that the distance from the center of fluke (x-axis) be the same in each photograph. As long as each halve of the tail is the same length on each side (which can be done with a photo editor) and the total length of the every fluke is the same on the x-axis, we are able to compare images and data points to evaluate whether or not the whale is identified.

Finding a Function that Approximates a General Whale Fluke

Our first step will be to approximate a function that roughly estimates the fluke of a whale. Here we look at the piece-wise function

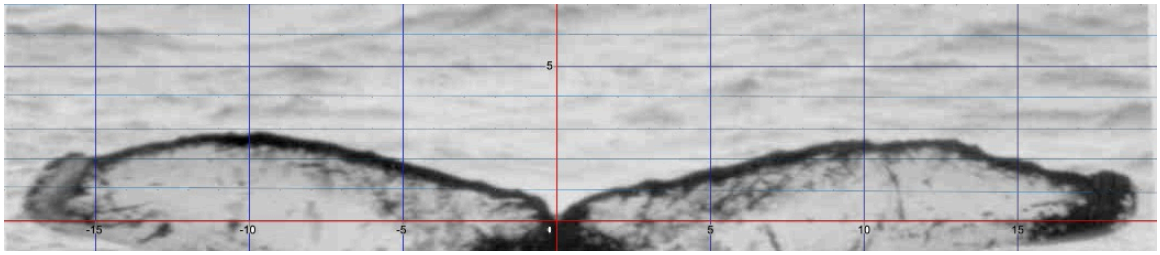
$$f(x) = \begin{cases} \sqrt{x} & \text{if } 0 \leq x \\ \sqrt{-x} & \text{if } x \leq 0 \end{cases} \tag{1}$$



Looking at a graph of this function we can see that this function estimates the fluke of a whale well.

Estimating Data Point of our Test Fluke

In order to estimate our data points for our whale fluke we will set it up on a coordinate plane and choose select points that we think will define the whale from other whales (Note: the coordinate system is similar to that of our base function approximation).



From this coordinate system we get the points

$$x = (-17, -16, -15, -14, -13, -12, -11, 10, 9, -8, -7, -6, -5, -4, -3, -2, -1, 0$$

$$, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18)$$

$$y = (.5, 2, 2, 2.33, 2, 5, 2.75, 2.75, 2.75, 2.6, 2.75, 2.33, 2.25, 1.9, 1.6, 1.4, 1.2, 1, 0, .75, 1, 1.3, 1.5, 1.6$$

$$, 1.75, 2.25, 2.25, 2.6, 2.6, 2.6, 2.6, 2.6, 2.25, 2, 1.6, 1.5, 1.75)$$

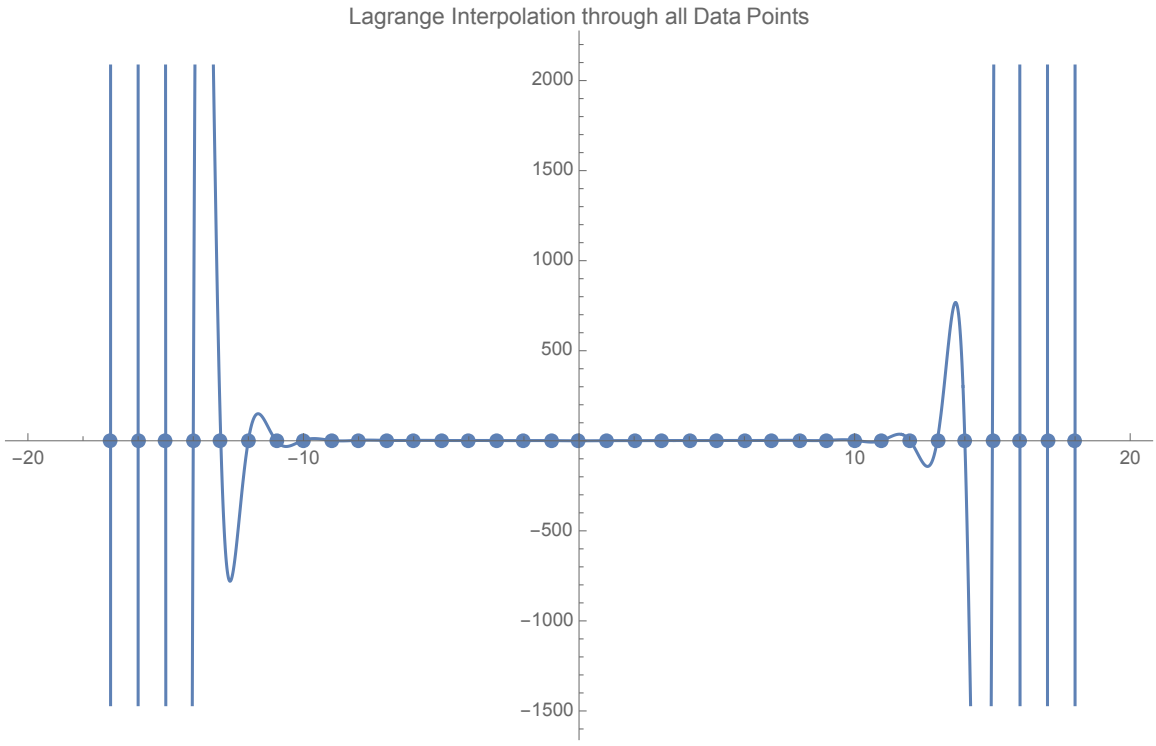
Using Approximation Methods to Model a Fluke

What we will do is use our data points that we have collected from our test fluke and make an interpolating polynomial that models the fluke for comparison to the original.

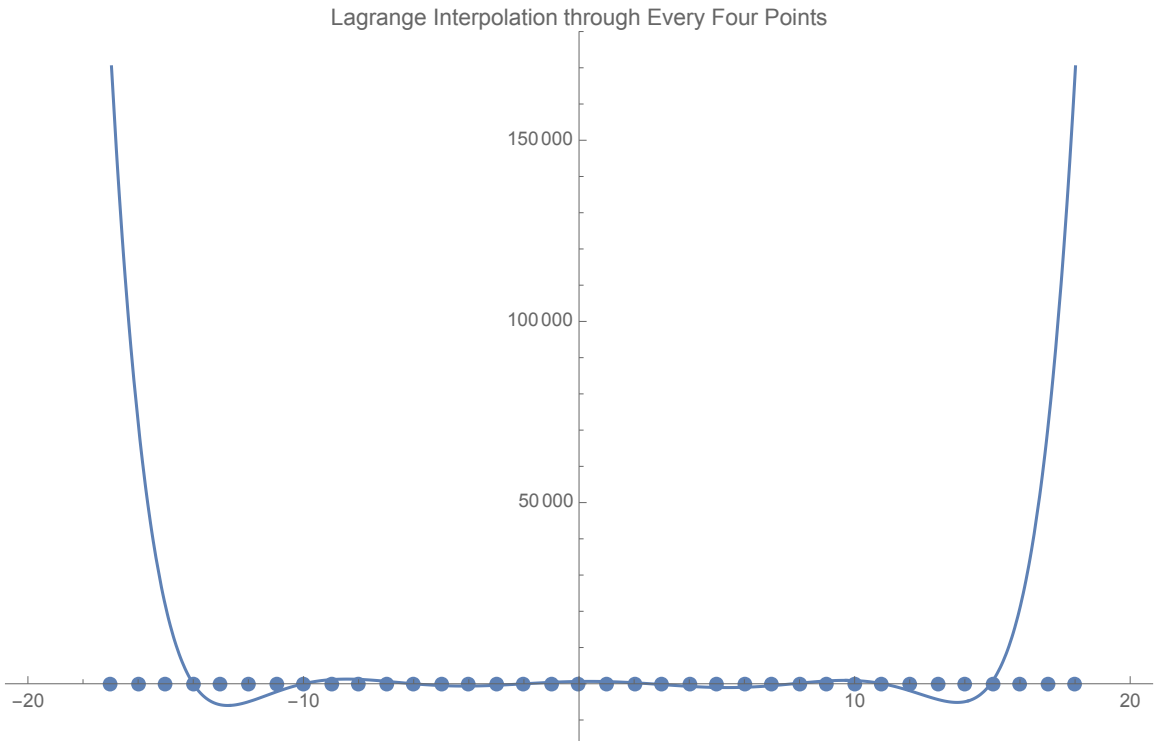
Because we have used a base piece-wise function as the general fluke for all our whales we can compare our interpolation with the general and hopefully get a numeric "signature" that we can use to identify each whale and show if a whale does not meet the present "signatures" of the data base, showing a new whale discovery.

Lagrange Interpolation [Epperson(2013)]

The Lagrange interpolations only requires the data points to make an interpolating polynomial we set up a divided difference table, fill out the table and get the polynomial.



As we can see from the graph of the polynomial this doesn't model the test whale's fluke very well. This is because the polynomial is of too high degree. However, if we use less data points we can get something that models the fluke better. We will could choose areas of data that seem to be unique to the whale. Consider the far right side we have a sort of bobby-pin structure, and something similar of the left. we will also use data point in the center to maintain the shape of the fluke.

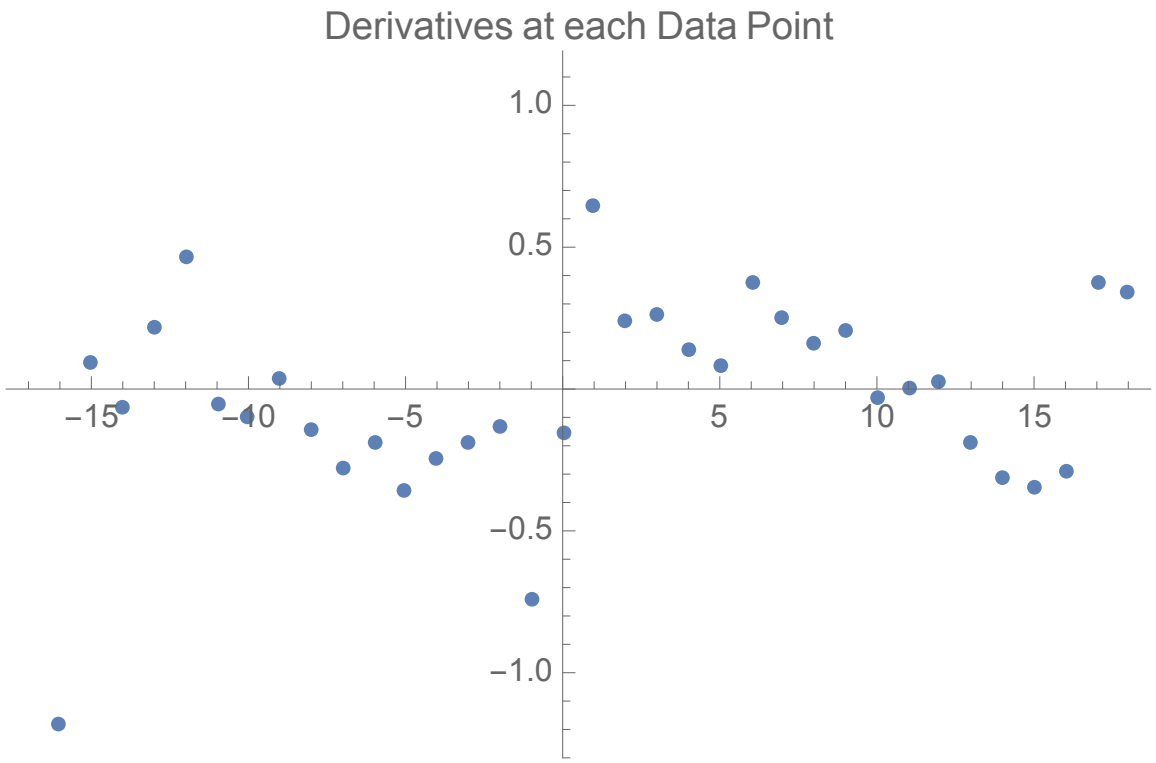


Notice that we have a better model of the test whale here however, because we used less data points we were not able to capture the amount of detail that we would hope when we are talking about a large amount of different whales. In fact, this method probably would only work slightly better than the base piece-wise function we used as a general model for all whales.

Numerical Differentiation [Epperson(2013)]

Numerical differentiation is a method of finding the derivatives at each point. This could be deemed useful for this project because if the whale has very intense detail at the end of it's fluke, the rate

of change may help with identifying unique aspects of a particular whale's fluke. An advantage of using this method is that you can use as many data points as you feel comfortable and get a numerical "signature" from Numerical differentiation. Whenever you come upon the same whale they will have the same or similar numerical differentiation signature. For example with the data points we have collected this whale's signature is

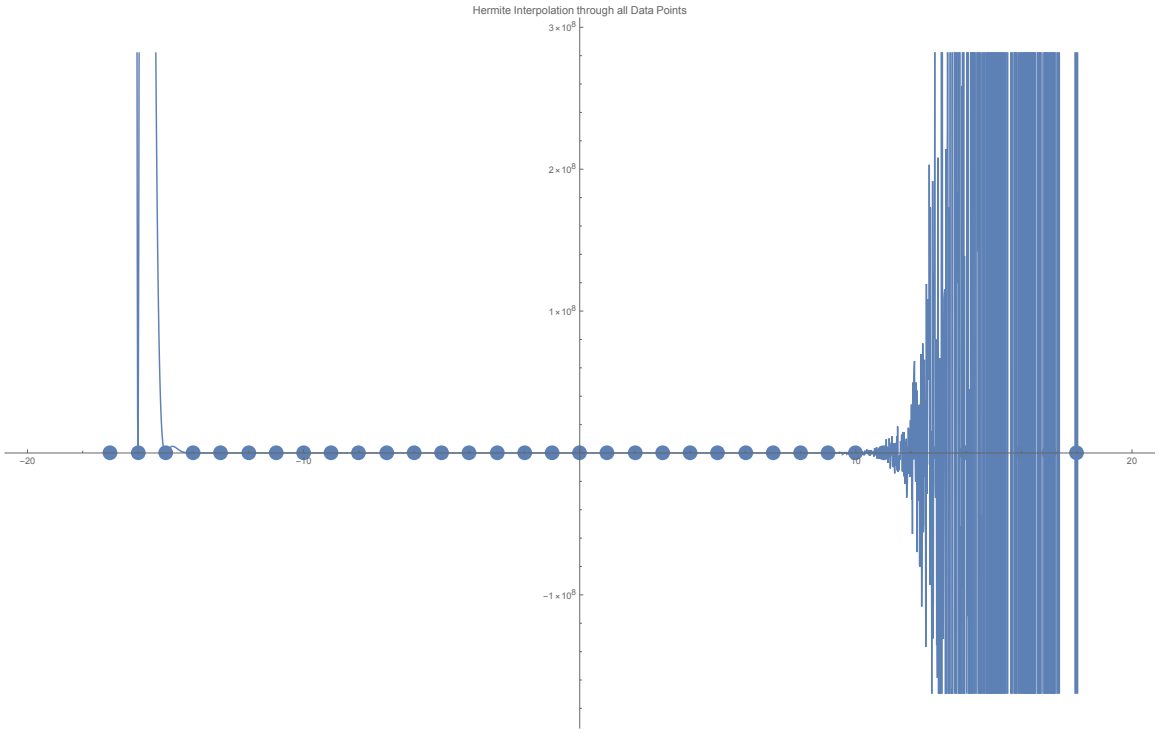


Note that for this approach it is important that you use the same coordinate system for all the whales. The interval of the fluke should be the same length for all photos of all whales.

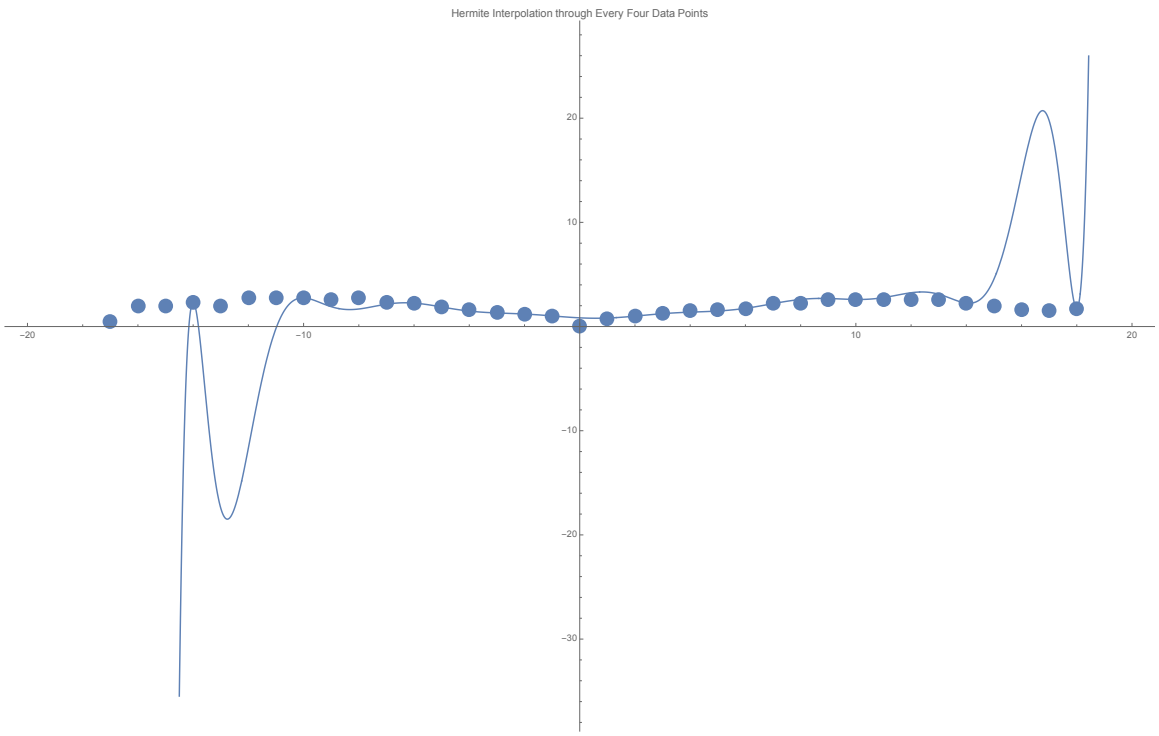
Hermite Interpolation [Epperson(2013)]

Similar to the Lagrange Interpolating Polynomial, the Hermite uses the given data points to approximate the model. However, Hermite also uses our numerical differentiation to use the derivative with each ordered pair. This can be deemed useful, especially when modeling certain sections of the fluke because these derivative help include the points and their apparent rate of change.

Unfortunately, the Hermite interpolating polynomial has the same fault as the Lagrange. With double the amount of data as the Lagrange, the Hermite also does not model the test fluke very well.



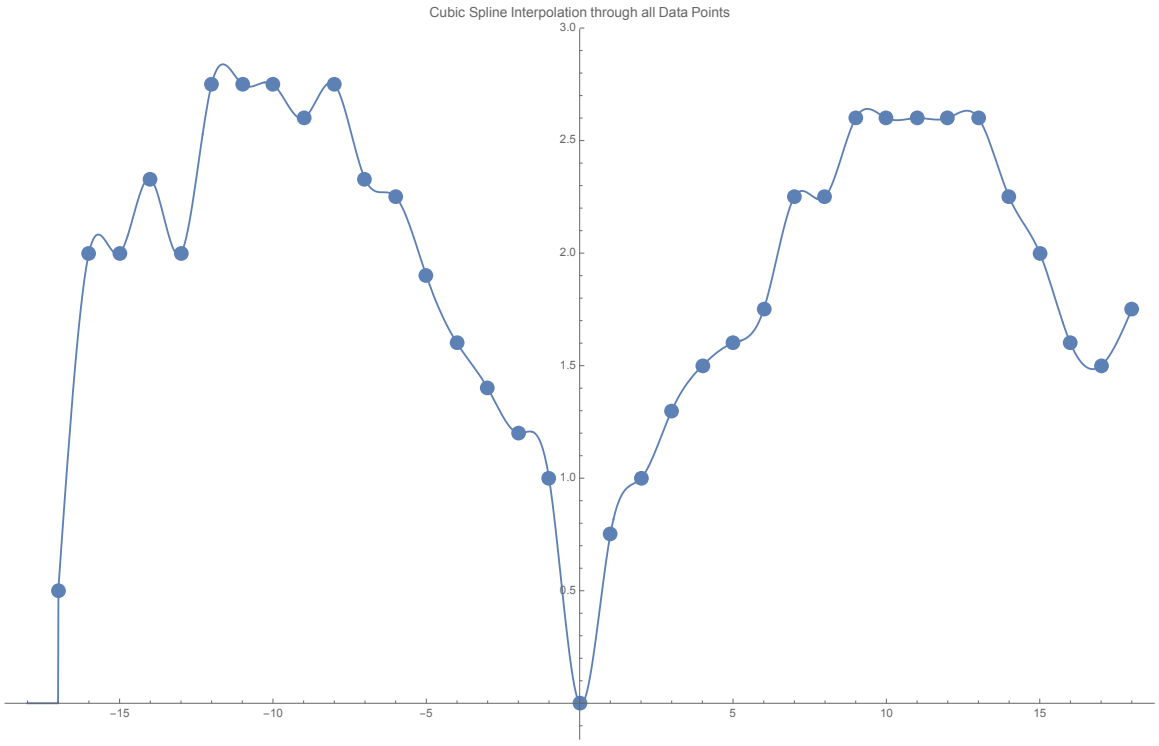
So similar to the Lagrange we can reduce the number of points to get a better model of the fluke.



Maybe this is a better approximation than the Lagrange but it still doesn't help us when we are looking at thousands of different whale flukes.

Cubic Spline Interpolation [Epperson(2013)]

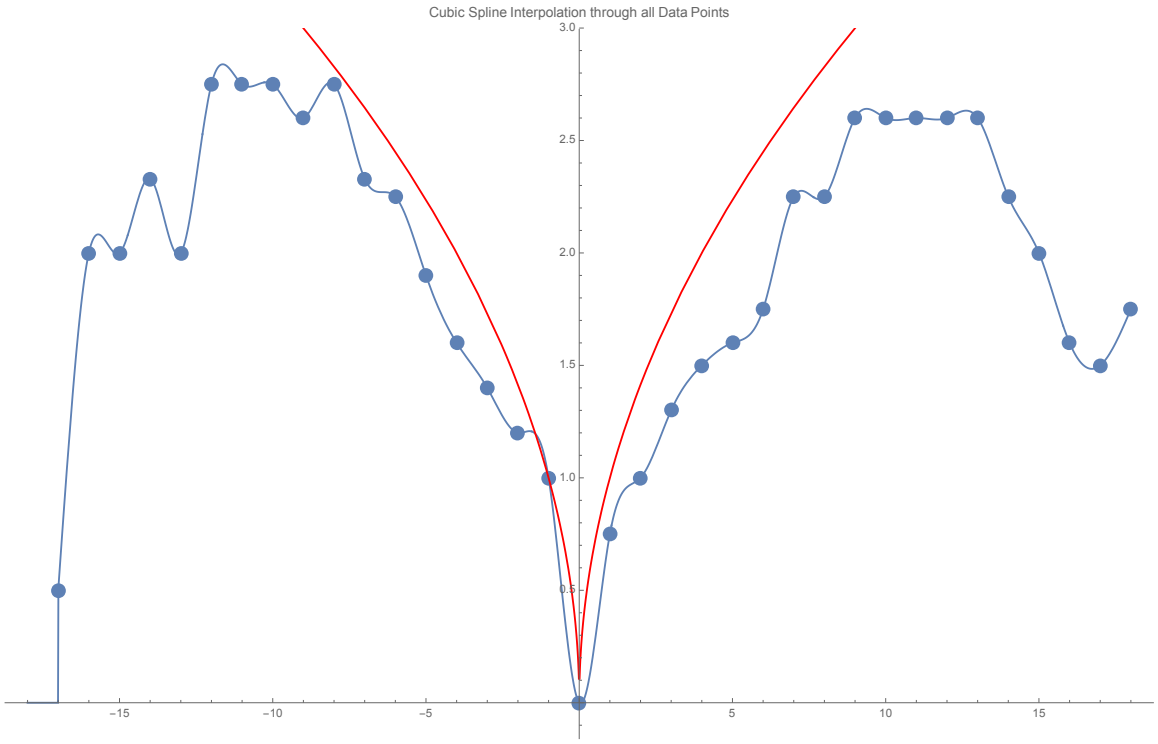
Cubic Spline interpolation is unique because it is a piece-wise function that's individual intervals map from point to point. Giving us a good approximation and model for them. This is a Cubic Spline for the data we estimated.



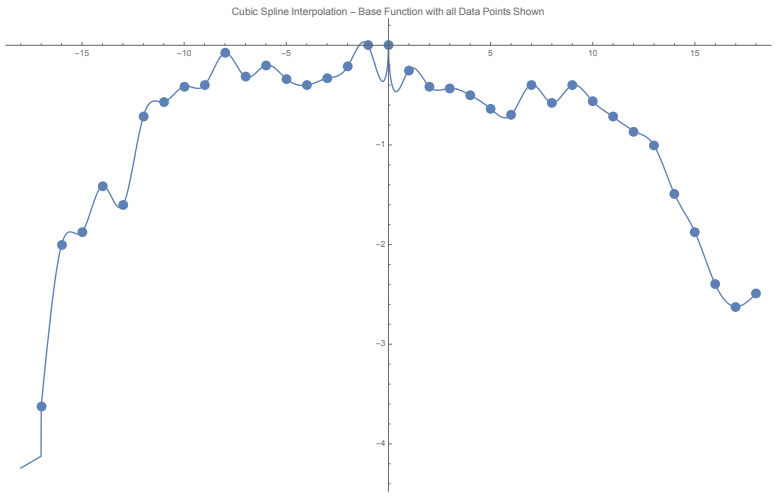
As we can see this is definitely the best model for our test fluke. We can even add or take away data points and test to see if the function approximates the points well. But how can we use this interpolation method to determine whether a whale is new or already in the data base?

The Cubic Spline "Signature"

When we are looking at our original base piece-wise function in comparison to the Cubic spline we can use the difference in points to determine a unique set of data points.



If we look at the error that comes from our Cubic Spline approximation from the base function we get



This is a representation of the difference of our cubic spline minus the base function. The error associated with each is as follows. A negative error implies that the fluke is smaller than the base function at point and vice versa. You may very well compare the data points from each picture

to each other, but with the graph it gives you the ability to check the spaces between the data points. Seeing how those match up may be the key to determining whether a whale is new or not.

Results

When looking at all of our interpolation methods we have been left with one that stands out from them all. The Cubic Spline represents our test fluke well. The best thing about the Cubic Spline, in this case, is that we can add as many data points to our function and still get a good interpolation of the data. This is advantageous because we won't know how many unique portions of the fluke a whale will have until we have a picture. This way, with more pictures of the same whale we can add points to our Spline approximation and theoretically have a very good model of the fluke for any given whale we come across.

Further, with the Cubic Spline Signature method we can determine whether or not a whale is new by comparing our current signatures of individual whale flukes to any given picture and see how they compare. This gives us the opportunity to compare even the tiniest differences to determine whether or not the whale is new.

References

[Kaggle(2018)] Kaggle. Humpback whale identification challenge, 2018. URL https://www.kaggle.com/competitions/whale-categorization-playground/overview.

[Epperson(2013)] James F Epperson. *An introduction to numerical methods and analysis*. John Wiley & Sons, 2013.

Midterm Exam

In[15]:=
```
img = 
```


In[49]:=
```
f[x_] := Piecewise[{
    {Sqrt[-x], x ≤ 0},
    {Sqrt[x], 0 ≤ x}
   }];
```

In[235]:=
```
xi = {-17, -16, -15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5, -4,
  -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18}
fxi = {.5, 2, 2, 2.33, 2, 2.75, 2.75, 2.75, 2.6, 2.75, 2.33, 2.25,
  1.9, 1.6, 1.4, 1.2, 1, 0, .75, 1, 1.3, 1.5, 1.6, 1.75, 2.25,
  2.25, 2.6, 2.6, 2.6, 2.6, 2.6, 2.25, 2, 1.6, 1.5, 1.75}
```

Out[235]=
```
{-17, -16, -15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5, -4, -3,
 -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18}
```

Out[236]=
```
{0.5, 2, 2, 2.33, 2, 2.75, 2.75, 2.75, 2.6, 2.75, 2.33,
 2.25, 1.9, 1.6, 1.4, 1.2, 1, 0, 0.75, 1, 1.3, 1.5, 1.6, 1.75,
 2.25, 2.25, 2.6, 2.6, 2.6, 2.6, 2.6, 2.25, 2, 1.6, 1.5, 1.75}
```

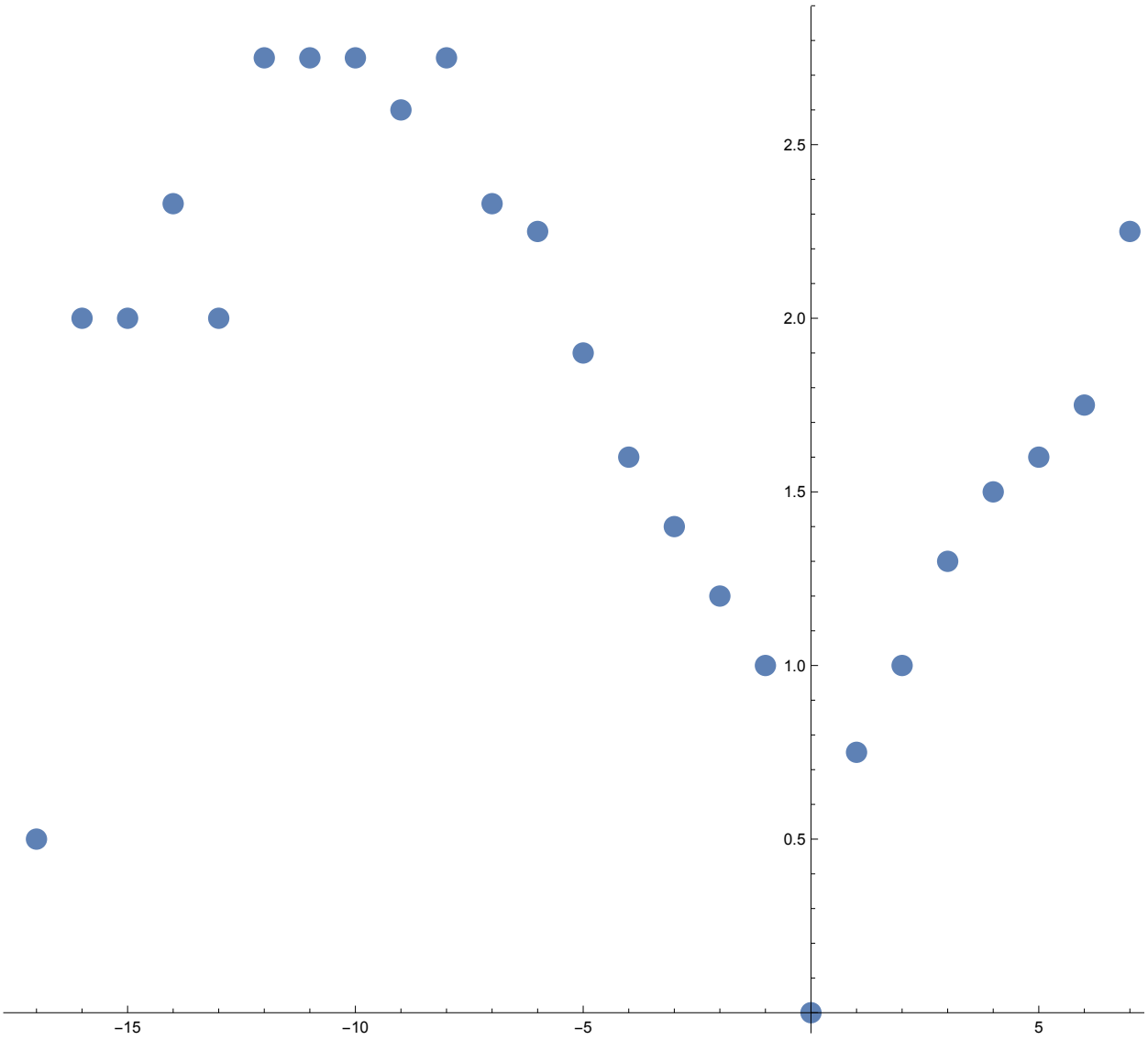
In[603]:=
```
n = Length[xi]
l = 4;
m = Length[xi] / l;
OrderedPairs = Table[{}, n];
For[ i = 1, i ≤ n, i++,
 OrderedPairs[[i]] = {xi[[i]], fxi[[i]]}
]
OrderedPairs
```

Out[603]=
```
36
```

Out[608]=
```
{{-17, 0.5}, {-16, 2}, {-15, 2}, {-14, 2.33}, {-13, 2}, {-12, 2.75}, {-11, 2.75},
 {-10, 2.75}, {-9, 2.6}, {-8, 2.75}, {-7, 2.33}, {-6, 2.25}, {-5, 1.9}, {-4, 1.6},
 {-3, 1.4}, {-2, 1.2}, {-1, 1}, {0, 0}, {1, 0.75}, {2, 1}, {3, 1.3}, {4, 1.5},
 {5, 1.6}, {6, 1.75}, {7, 2.25}, {8, 2.25}, {9, 2.6}, {10, 2.6}, {11, 2.6},
 {12, 2.6}, {13, 2.6}, {14, 2.25}, {15, 2}, {16, 1.6}, {17, 1.5}, {18, 1.75}}
```

In[218]:= `ListPlot[OrderedPairs]`

Out[218]=



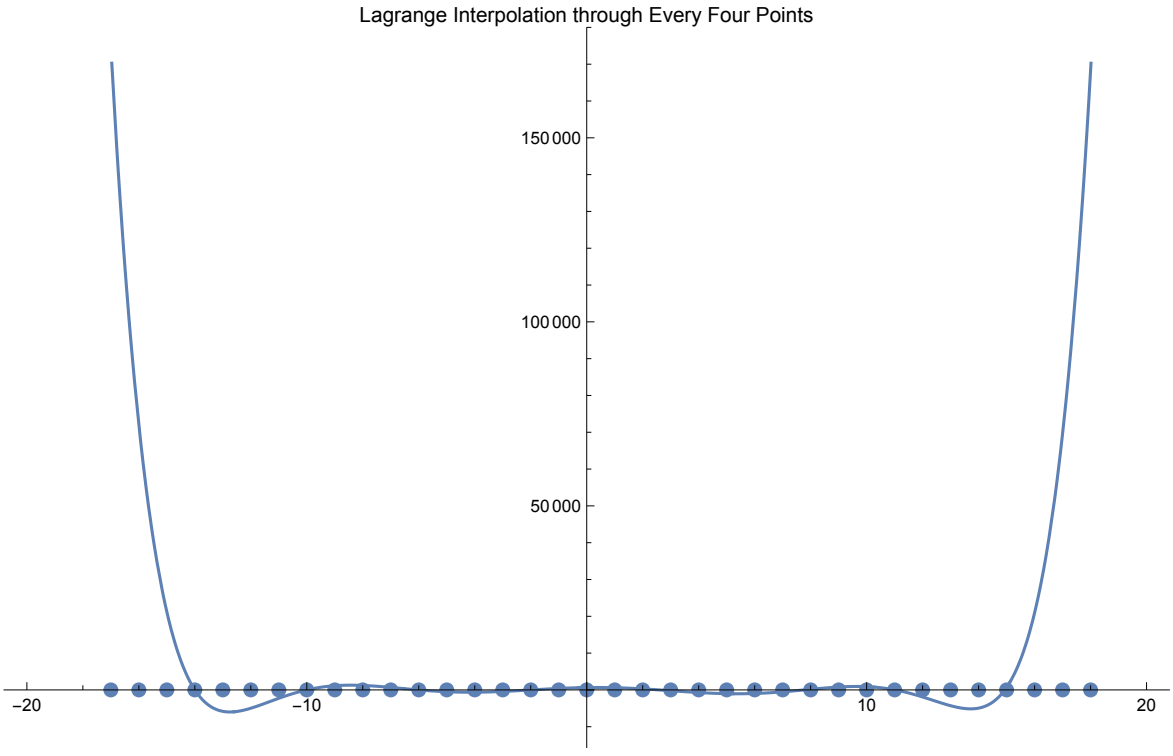
Lagrange Interpolation:

In[609]:=
```
MatrixForm[DivDif = Table[{}, m, m]];
For[i = 1, i ≤ m, i++, DivDif[[i, 1]] = fxi[[l * i]]] ;
MatrixForm[DivDif];
For[j = 2, j ≤ m, j++,
 For[i = j, i ≤ m, i++, DivDif[[i, j]] =
   (DivDif[[i, j - 1]] - DivDif[[i - 1, j - 1]]) /
    (xi[[i]] - xi[[i - (j - 1)]])]]
MatrixForm[DivDif];
Lag[x_] := Sum[DivDif[[i, i]] * Product[(x - xi[[l * j]]), {j, 1, i - 1}], {i, 1, m}]
Simplify[Lag[x]]
```

Out[615]=
$$636.79 + 109.47\,x - 143.206\,x^2 - 9.65522\,x^3 + 4.48253\,x^4 + 0.140194\,x^5 - 0.0423056\,x^6 - 0.000501984\,x^7 + 0.000116071\,x^8$$

In[617]:=
```
Show[
 Plot[Lag[x], {x, -20, 20},
  PlotLabel → "Lagrange Interpolation through Every Four Points"],
 ListPlot[OrderedPairs]
]
```

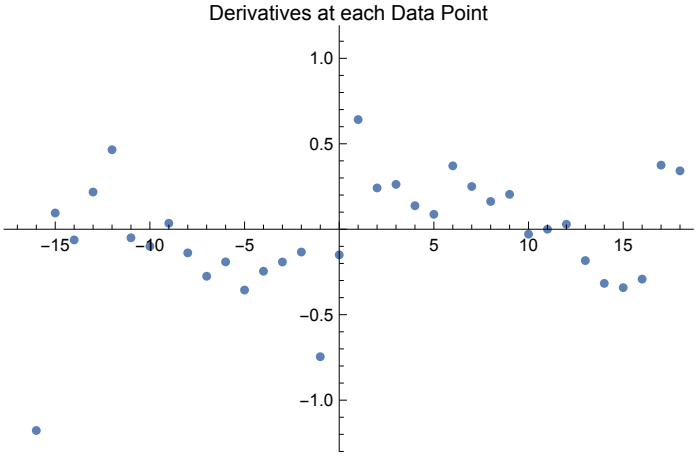
Out[617]=



Numerical Differentiation

In[661]:=
```
h = xi[[2]] - xi[[1]];
derivfxi = Table[{}, n];
For[i = 3, i ≤ n - 2, i++,
 derivfxi[[i]] =
  (fxi[[i - 2]] - 8 fxi[[i - 1]] +
     8 fxi[[i + 1]] - fxi[[i + 2]]) / (12 h)]
For[i = 1, i ≤ 2, i++,
 derivfxi[[i]] =
  (-25 fxi[[i]] + 48 fxi[[i + 1]] - 36 fxi[[i + 2]] +
     16 fxi[[i + 3]] - 3 fxi[[i + 4]]) (1 / (12 h))]
For[i = n, i ≥ n - 1, i--,
 derivfxi[[i]] =
  (-25 fxi[[i]] + 48 fxi[[i - 1]] - 36 fxi[[i - 2]] +
     16 fxi[[i - 3]] - 3 fxi[[i - 4]]) (1 / (12 (-h)))]
derivfxi;
OrderedPairs2 = Table[{}, n];
For[ i = 1, i ≤ n, i++,
  OrderedPairs2[[i]] = {xi[[i]], derivfxi[[i]]}
 ];
OrderedPairs2;
ListPlot[OrderedPairs2, PlotLabel → "Derivatives at each Data Point"]
```

Out[670]=



Hermite Interpolation

In[690]:=
```
q = 4;
w = 2 * Length[xi] / q;
MatrixForm[DivDif2 = Table[{}, w, w + 1]];
For[i = 1, i ≤ w / 2, i++,
 DivDif2[[(2 i), 1]] = xi[[q * i]]]
For[i = 1, i ≤ w / 2, i++,
 DivDif2[[(2 i) - 1, 1]] = xi[[q * i]]]

MatrixForm[DivDif2];

For[i = 1, i ≤ w / 2, i++,
 DivDif2[[(2 i), 2]] = fxi[[q * i]]]
For[i = 1, i ≤ w / 2, i++,
 DivDif2[[(2 i) - 1, 2]] = fxi[[q * i]]]

MatrixForm[DivDif2];

For[ j = 2, j ≤ 2, j++,
 For[i = j, i ≤ w, i++, DivDif2[[i, j + 1]] = (DivDif2[[i, j]] - DivDif2[[i - 1, j]]) /
     (DivDif2[[i, j - 1]] - DivDif2[[i - 1, j - 1]])
]]
MatrixForm[DivDif2];
```

Power: Infinite expression $\frac{1}{0}$ encountered.

Infinity: Indeterminate expression 0. ComplexInfinity encountered.

Power: Infinite expression $\frac{1}{0}$ encountered.

Infinity: Indeterminate expression 0. ComplexInfinity encountered.

Power: Infinite expression $\frac{1}{0}$ encountered.

General: Further output of Power::infy will be suppressed during this calculation.

Infinity: Indeterminate expression 0. ComplexInfinity encountered.

General: Further output of Infinity::indet will be suppressed during this calculation.

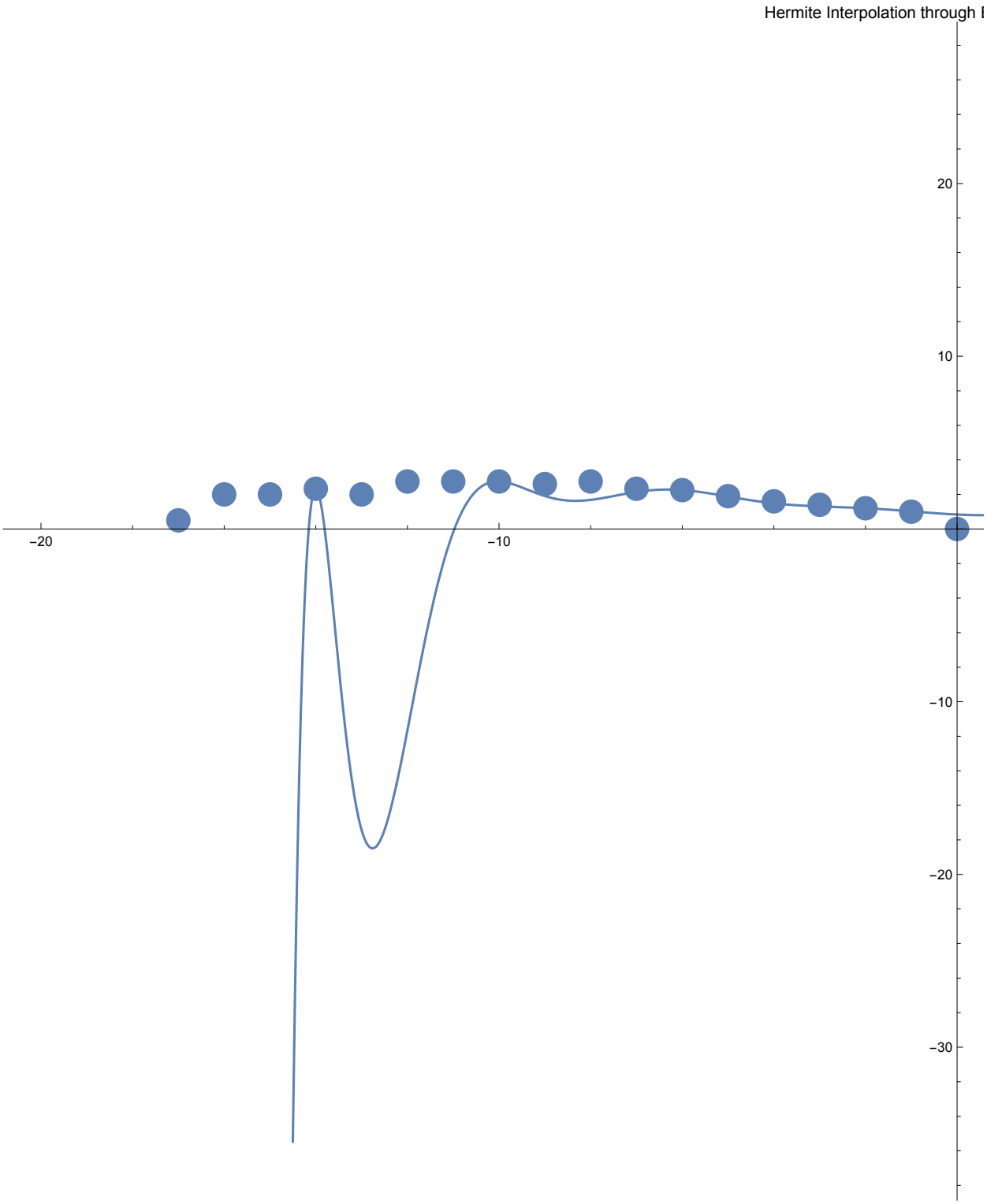
In[701]:=
```
For[i = 1, i ≤ w / 2, i++,
 DivDif2[[(2 i), 3]] = derivfxi[[q * i]]]
MatrixForm[DivDif2];
For[ j = 3, j ≤ w, j++,
 For[i = j, i ≤ w, i++, DivDif2[[i, j + 1]] = (DivDif2[[i, j]] - DivDif2[[i - 1, j]]) /
    (DivDif2[[i, j - (j - 1)]] - DivDif2[[i - (j - 1), j - (j - 1)]])]]
MatrixForm[DivDif2];
```

In[705]:=
```
Herm[x_] :=
 Sum[DivDif2[[i, i + 1]] * Product[(x - DivDif2[[j, 1]]), {j, 1, i - 1}], {i, 1, w}]
Simplify[Herm[x]]
```

Out[706]=
$$0.835947 - 0.125614\,x + 0.0902594\,x^2 + 0.0255941\,x^3 - 0.00748951\,x^4 - 0.00188607\,x^5 + 0.000396979\,x^6 + 0.000056305\,x^7 - 0.0000105713\,x^8 - 7.52368 \times 10^{-7}\,x^9 + 1.48157 \times 10^{-7}\,x^{10} + 4.13804 \times 10^{-9}\,x^{11} - 1.10761 \times 10^{-9}\,x^{12} - 6.95182 \times 10^{-13}\,x^{13} + 4.14841 \times 10^{-12}\,x^{14} - 7.10704 \times 10^{-14}\,x^{15} - 6.09489 \times 10^{-15}\,x^{16} + 1.77298 \times 10^{-16}\,x^{17}$$

In[708]:=
```
Show[
 Plot[Herm[x], {x, -20, 20},
  PlotLabel → "Hermite Interpolation through Every Four Data Points"],
 ListPlot[OrderedPairs]
]
```

Out[708]=



Cubic Spline Interpolation

In[489]:=
```
d = Length[xi];
hi = Table[{}, Length[xi] - 1];
```

```
For[i = 1, i ≤ Length[xi] - 1, i++,
 hi[[i]] = xi[[i + 1]] - xi[[i]]]
hi
aMatrix = Table[{}, d - 2, d - 2];
MatrixForm[aMatrix];

For[ j = 1, j ≤ d - 2, j++,
 For[i = 1, i ≤ d - 2, i++,
  aMatrix[[i, j]] = 0
 ]
]

(*First diagonal (i)*)
For[ j = 2, j ≤ d - 2, j++,
 For[i = j - 1, i ≤ j - 1, i++,
  aMatrix[[i, j]] = hi[[i + 1]]
 ]
]

(*Middle diagonal (i)*)
For[ j = 1, j ≤ d - 2, j++,
 For[i = j, i ≤ j, i++,
  aMatrix[[i, j]] = (2 (hi[[i]] + hi[[i + 1]]))
 ]
]
(*Bottom Diagonal (i-1)*)
For[ j = 1, j ≤ d - 3, j++,
 For[i = j + 1, i ≤ j + 1, i++,
  aMatrix[[i, j]] = hi[[i]]
 ]
]


(*Matrix A to solve for cj*)
MatrixForm[aMatrix]

(*Matrix b to solve for cj's*)
bMatrix = Table[{}, d - 2, 1];
For[i = 2, i ≤ d - 1, i++,
 bMatrix[[i - 1]] =
  ((3 / hi[[i]]) (fxi[[i + 1]] - fxi[[i]])) - ((3 / hi[[i - 1]]) (fxi[[i]] - fxi[[i - 1]]))
]
bMatrix
```

```
(*Correct Matrix for b*)
MatrixForm[bMatrix]
cj = LinearSolve[aMatrix, bMatrix]
(*Do not redo this Function or else to many variables in cj*)
PrependTo[cj, 0];
AppendTo[cj, 0];
cj
(*Creating the bj Mat*)
bj = Table[{}, d - 1, 1];
MatrixForm[bj];
(*Filling bj matrix with proper values*)
For[i = 1, i ≤ d - 1, i++,
 bj[[i]] =
  ((fxi[[i + 1]] - fxi[[i]]) / hi[[i]]) - ((hi[[i]] / 3) (2 cj[[i]] + cj[[i + 1]]))
]
bj
(*Making a Matrix for dj*)
dj = Table[{}, d - 1, 1];
MatrixForm[dj]
For[i = 1, i ≤ d - 1, i++,
 dj[[i]] = (1 / (3 hi[[i]])) (cj[[i + 1]] - cj[[i]])
]
dj
```

Out[492]= {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}

Out[499]//MatrixForm=

$$\begin{pmatrix}
4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4
\end{pmatrix}$$

Out[502]= $\{-4.5, 0.99, -1.98, 3.24, -2.25, 0., -0.45, 0.9, -1.71, 1.02, -0.81,$
$0.15, 0.3, 6.66134 \times 10^{-16}, 0., -2.4, 5.25, -1.5, 0.15, -0.3, -0.3, 0.15,$
$1.05, -1.5, 1.05, -1.05, 0., 0., 0., -1.05, 0.3, -0.45, 0.9, 1.05\}$

Out[503]//MatrixForm=

$$\begin{pmatrix} -4.5 \\ 0.99 \\ -1.98 \\ 3.24 \\ -2.25 \\ 0. \\ -0.45 \\ 0.9 \\ -1.71 \\ 1.02 \\ -0.81 \\ 0.15 \\ 0.3 \\ 6.66134 \times 10^{-16} \\ 0. \\ -2.4 \\ 5.25 \\ -1.5 \\ 0.15 \\ -0.3 \\ -0.3 \\ 0.15 \\ 1.05 \\ -1.5 \\ 1.05 \\ -1.05 \\ 0. \\ 0. \\ 0. \\ -1.05 \\ 0.3 \\ -0.45 \\ 0.9 \\ 1.05 \end{pmatrix}$$

Out[504]= {-1.33483, 0.839334, -1.0325, 1.31067, -0.970197, 0.320114, -0.31026, 0.470927, -0.673448, 0.512864, -0.358008, 0.109169, 0.0713322, -0.0944979, 0.306659, -1.13214, 1.8219, -0.90546, 0.299942, -0.144307, -0.0227143, -0.0648359, 0.432058, -0.613396, 0.521527, -0.422712, 0.119322, -0.0545743, 0.0989756, -0.341328, 0.216337, -0.224019, 0.229738, 0.205065}

Out[507]= {0, -1.33483, 0.839334, -1.0325, 1.31067, -0.970197, 0.320114, -0.31026, 0.470927, -0.673448, 0.512864, -0.358008, 0.109169, 0.0713322, -0.0944979, 0.306659, -1.13214, 1.8219, -0.90546, 0.299942, -0.144307, -0.0227143, -0.0648359, 0.432058, -0.613396, 0.521527, -0.422712, 0.119322, -0.0545743, 0.0989756, -0.341328, 0.216337, -0.224019, 0.229738, 0.205065, 0}

Out[511]= {1.94494, 0.610111, 0.114611, -0.0785568, 0.199616, 0.540093, -0.109989, -0.100135, 0.0605312, -0.141989, -0.302573, -0.147718, -0.396557, -0.216056, -0.239221, -0.0270597, -0.85254, -0.16278, 0.75366, 0.148141, 0.303776, 0.136755, 0.0492046, 0.416427, 0.235088, 0.143219, 0.242034, -0.0613563, 0.003391, 0.0477923, -0.19456, -0.319552, -0.327234, -0.321514, 0.11329}

Out[513]//MatrixForm=

$$\begin{pmatrix} \{\} \\ \{\} \end{pmatrix}$$

Out[515]= {−0.444944, 0.724722, −0.623945, 0.781059, −0.760291, 0.430104, −0.210125, 0.260396, −0.381458, 0.395437, −0.290291, 0.155726, −0.0126123, −0.0552767, 0.133719, −0.4796, 0.98468, −0.90912, 0.401801, −0.148083, 0.0405309, −0.0140405, 0.165631, −0.348485, 0.378308, −0.314746, 0.180678, −0.0579653, 0.0511833, −0.146768, 0.185888, −0.146785, 0.151252, −0.00822431, −0.0683551}

In[522]:=
```
Remove[Wild]
```

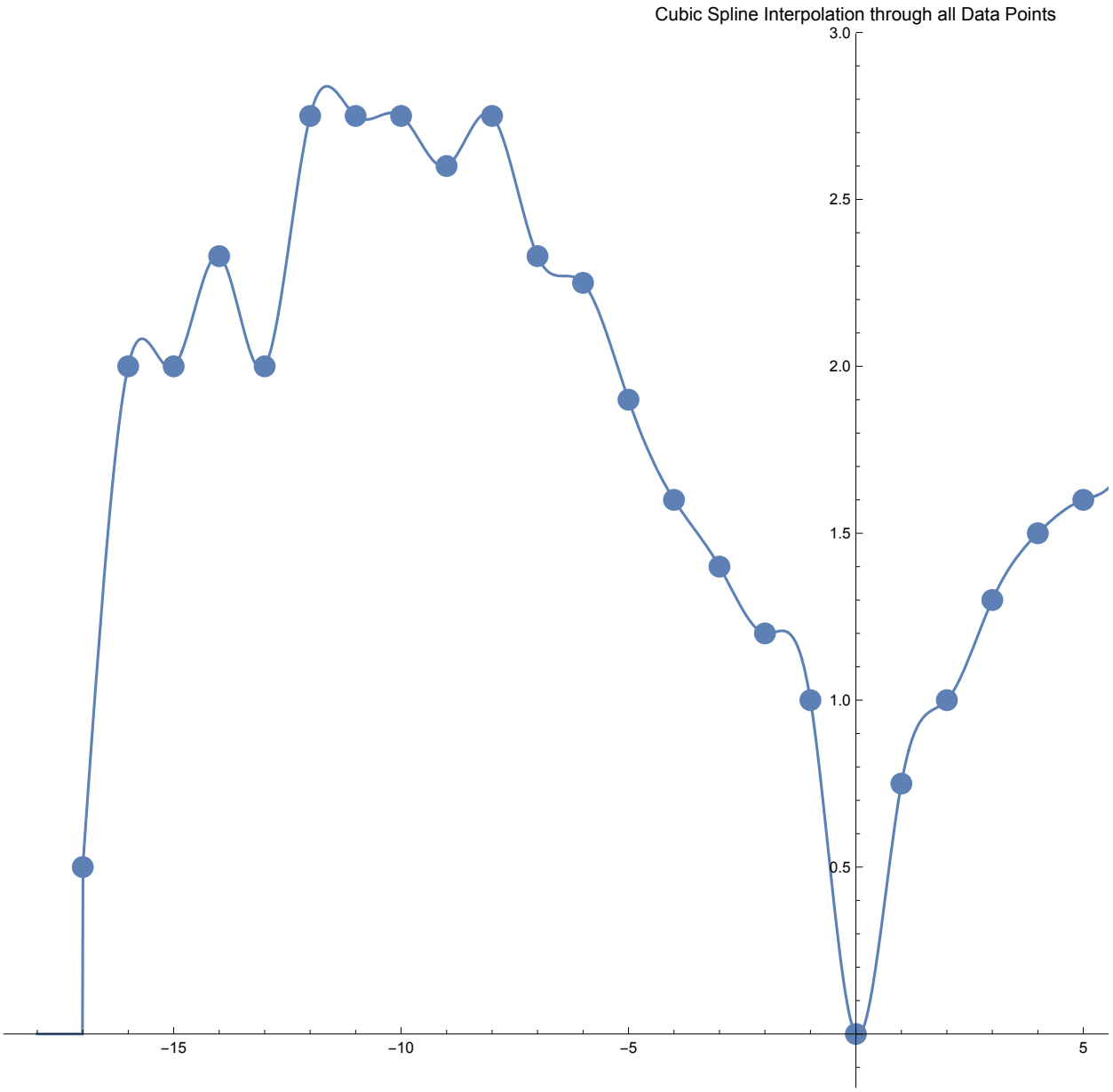
```
Wild[x_] := Piecewise[ {
   {fxi[[1]] + bj[[1]] (x - xi[[1]]) + cj[[1]] (x - xi[[1]])^2 +
     dj[[1]] (x - xi[[1]])^3, xi[[1]] ≤ x ≤ xi[[2]]},
   {fxi[[2]] + bj[[2]] (x - xi[[2]]) + cj[[2]] (x - xi[[2]])^2 +
     dj[[2]] (x - xi[[2]])^3, xi[[2]] ≤ x ≤ xi[[3]]},
   {fxi[[3]] + bj[[3]] (x - xi[[3]]) + cj[[3]] (x - xi[[3]])^2 +
     dj[[3]] (x - xi[[3]])^3, xi[[3]] ≤ x ≤ xi[[4]]},
   {fxi[[4]] + bj[[4]] (x - xi[[4]]) + cj[[4]] (x - xi[[4]])^2 +
     dj[[4]] (x - xi[[4]])^3, xi[[4]] ≤ x ≤ xi[[5]]},
```

```
{fxi[[5]] + bj[[5]] (x - xi[[5]]) + cj[[5]] (x - xi[[5]])^2 +
  dj[[5]] (x - xi[[5]])^3, xi[[5]] ≤ x ≤ xi[[6]]},
{fxi[[6]] + bj[[6]] (x - xi[[6]]) + cj[[6]] (x - xi[[6]])^2 +
  dj[[6]] (x - xi[[6]])^3, xi[[6]] ≤ x ≤ xi[[7]]},
{fxi[[7]] + bj[[7]] (x - xi[[7]]) + cj[[7]] (x - xi[[7]])^2 +
  dj[[7]] (x - xi[[7]])^3, xi[[7]] ≤ x ≤ xi[[8]]},
{fxi[[8]] + bj[[8]] (x - xi[[8]]) + cj[[8]] (x - xi[[8]])^2 +
  dj[[8]] (x - xi[[8]])^3, xi[[8]] ≤ x ≤ xi[[9]]},
{fxi[[9]] + bj[[9]] (x - xi[[9]]) + cj[[9]] (x - xi[[9]])^2 +
  dj[[9]] (x - xi[[9]])^3, xi[[9]] ≤ x ≤ xi[[10]]},
{fxi[[10]] + bj[[10]] (x - xi[[10]]) + cj[[10]] (x - xi[[10]])^2 +
  dj[[10]] (x - xi[[10]])^3, xi[[10]] ≤ x ≤ xi[[11]]},
{fxi[[11]] + bj[[11]] (x - xi[[11]]) + cj[[11]] (x - xi[[11]])^2 +
  dj[[11]] (x - xi[[11]])^3, xi[[11]] ≤ x ≤ xi[[12]]},
{fxi[[12]] + bj[[12]] (x - xi[[12]]) + cj[[12]] (x - xi[[12]])^2 +
  dj[[12]] (x - xi[[12]])^3, xi[[12]] ≤ x ≤ xi[[13]]},
{fxi[[13]] + bj[[13]] (x - xi[[13]]) + cj[[13]] (x - xi[[13]])^2 +
  dj[[13]] (x - xi[[13]])^3, xi[[13]] ≤ x ≤ xi[[14]]},
{fxi[[14]] + bj[[14]] (x - xi[[14]]) + cj[[14]] (x - xi[[14]])^2 +
  dj[[14]] (x - xi[[14]])^3, xi[[14]] ≤ x ≤ xi[[15]]},
{fxi[[15]] + bj[[15]] (x - xi[[15]]) + cj[[15]] (x - xi[[15]])^2 +
  dj[[15]] (x - xi[[15]])^3, xi[[15]] ≤ x ≤ xi[[16]]},
{fxi[[16]] + bj[[16]] (x - xi[[16]]) + cj[[16]] (x - xi[[16]])^2 +
  dj[[16]] (x - xi[[16]])^3, xi[[16]] ≤ x ≤ xi[[17]]},
{fxi[[17]] + bj[[17]] (x - xi[[17]]) + cj[[17]] (x - xi[[17]])^2 +
  dj[[17]] (x - xi[[17]])^3, xi[[17]] ≤ x ≤ xi[[18]]},
{fxi[[18]] + bj[[18]] (x - xi[[18]]) + cj[[18]] (x - xi[[18]])^2 +
  dj[[18]] (x - xi[[18]])^3, xi[[18]] ≤ x ≤ xi[[19]]},
{fxi[[19]] + bj[[19]] (x - xi[[19]]) + cj[[19]] (x - xi[[19]])^2 +
  dj[[19]] (x - xi[[19]])^3, xi[[19]] ≤ x ≤ xi[[20]]},
{fxi[[20]] + bj[[20]] (x - xi[[20]]) + cj[[20]] (x - xi[[20]])^2 +
  dj[[20]] (x - xi[[20]])^3, xi[[20]] ≤ x ≤ xi[[21]]},
{fxi[[21]] + bj[[21]] (x - xi[[21]]) + cj[[21]] (x - xi[[21]])^2 +
  dj[[21]] (x - xi[[21]])^3, xi[[21]] ≤ x ≤ xi[[22]]},
{fxi[[22]] + bj[[22]] (x - xi[[22]]) + cj[[22]] (x - xi[[22]])^2 +
  dj[[22]] (x - xi[[22]])^3, xi[[22]] ≤ x ≤ xi[[23]]},
{fxi[[23]] + bj[[23]] (x - xi[[23]]) + cj[[23]] (x - xi[[23]])^2 +
  dj[[23]] (x - xi[[23]])^3, xi[[23]] ≤ x ≤ xi[[24]]},
{fxi[[24]] + bj[[24]] (x - xi[[24]]) + cj[[24]] (x - xi[[24]])^2 +
  dj[[24]] (x - xi[[24]])^3, xi[[24]] ≤ x ≤ xi[[25]]},
{fxi[[25]] + bj[[25]] (x - xi[[25]]) + cj[[25]] (x - xi[[25]])^2 +
  dj[[25]] (x - xi[[25]])^3, xi[[25]] ≤ x ≤ xi[[26]]},
```

```
    {fxi[[26]] + bj[[26]] (x - xi[[26]]) + cj[[26]] (x - xi[[26]])^2 +
      dj[[26]] (x - xi[[26]])^3, xi[[26]] ≤ x ≤ xi[[27]]},
    {fxi[[27]] + bj[[27]] (x - xi[[27]]) + cj[[27]] (x - xi[[27]])^2 +
      dj[[27]] (x - xi[[27]])^3, xi[[27]] ≤ x ≤ xi[[28]]},
    {fxi[[28]] + bj[[28]] (x - xi[[28]]) + cj[[28]] (x - xi[[28]])^2 +
      dj[[28]] (x - xi[[28]])^3, xi[[28]] ≤ x ≤ xi[[29]]},
    {fxi[[29]] + bj[[29]] (x - xi[[29]]) + cj[[29]] (x - xi[[29]])^2 +
      dj[[29]] (x - xi[[29]])^3, xi[[29]] ≤ x ≤ xi[[30]]},
    {fxi[[30]] + bj[[30]] (x - xi[[30]]) + cj[[30]] (x - xi[[30]])^2 +
      dj[[30]] (x - xi[[30]])^3, xi[[30]] ≤ x ≤ xi[[31]]},
    {fxi[[31]] + bj[[31]] (x - xi[[31]]) + cj[[31]] (x - xi[[31]])^2 +
      dj[[31]] (x - xi[[31]])^3, xi[[31]] ≤ x ≤ xi[[32]]},
    {fxi[[32]] + bj[[32]] (x - xi[[32]]) + cj[[32]] (x - xi[[32]])^2 +
      dj[[32]] (x - xi[[32]])^3, xi[[32]] ≤ x ≤ xi[[33]]},
    {fxi[[33]] + bj[[33]] (x - xi[[33]]) + cj[[33]] (x - xi[[33]])^2 +
      dj[[33]] (x - xi[[33]])^3, xi[[33]] ≤ x ≤ xi[[34]]},
    {fxi[[34]] + bj[[34]] (x - xi[[34]]) + cj[[34]] (x - xi[[34]])^2 +
      dj[[34]] (x - xi[[34]])^3, xi[[34]] ≤ x ≤ xi[[35]]},
    {fxi[[35]] + bj[[35]] (x - xi[[35]]) + cj[[35]] (x - xi[[35]])^2 +
      dj[[35]] (x - xi[[35]])^3, xi[[35]] ≤ x ≤ xi[[36]]}
   }]
```

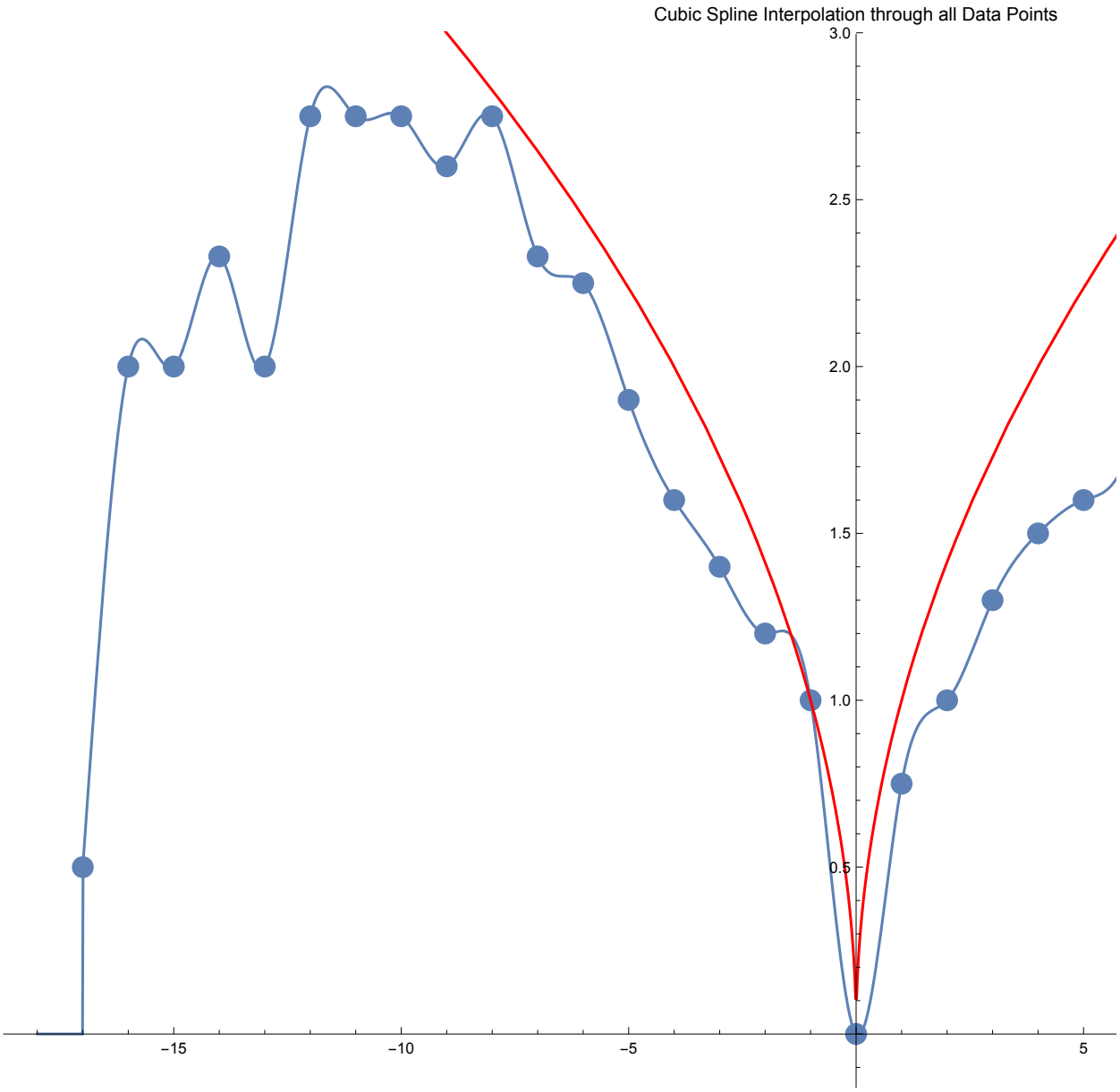
In[671]:=
```
Show[
 Plot[Wild[x], {x, -18, 18},
  PlotLabel → "Cubic Spline Interpolation through all Data Points"],
 ListPlot[OrderedPairs]
]
```

Out[671]=


Cubic Spline Interpolation through all Data Points
3.0
2.5
2.0
1.5
1.0
0.5
−15
−10
−5
5

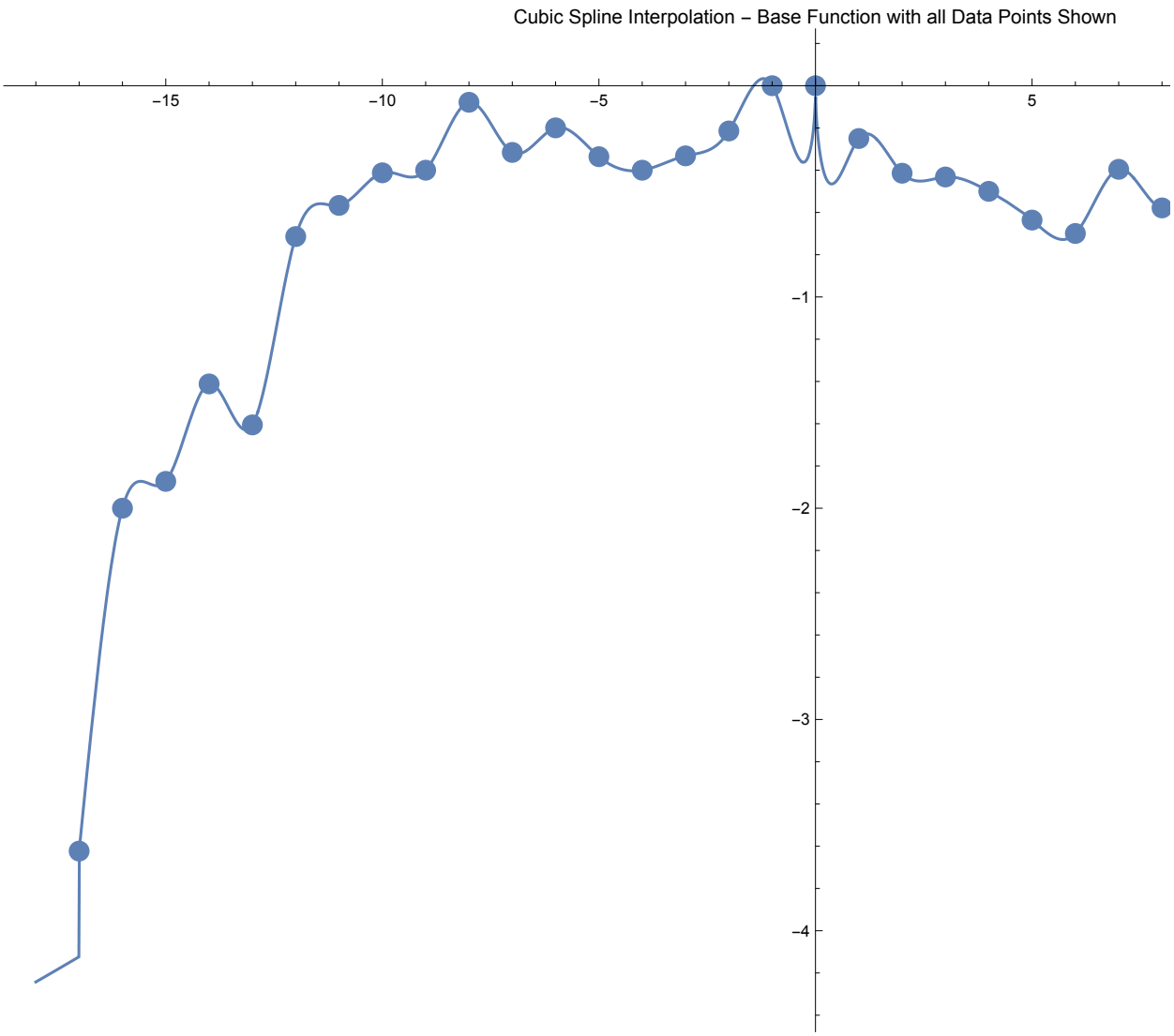
In[710]:=
```
Show[
 Plot[Wild[x], {x, -18, 18},
  PlotLabel → "Cubic Spline Interpolation through all Data Points"],
 ListPlot[OrderedPairs],
 Plot[f[x], {x, -18, 18}, PlotStyle → Red]
]
```

Out[710]=



In[729]:=
```
Show[
 Plot[Wild[x] - f[x], {x, -18, 18}, PlotLabel →
   "Cubic Spline Interpolation - Base Function with all Data Points Shown"],
 ListPlot[OrderedPairs3]
]
```

Out[729]=



In[716]:=
```
error = Table[{}, n];
For[i = 1, i ≤ n, i++,
 error[[i]] = Wild[xi[[i]]] - f[xi[[i]]]]
error
```

Out[718]=
```
{-3.62311, -2., -1.87298, -1.41166, -1.60555, -0.714102, -0.566625,
 -0.412278, -0.4, -0.0784271, -0.315751, -0.19949, -0.336068, -0.4,
 -0.332051, -0.214214, 0., -1.11022 × 10^-16, -0.25, -0.414214, -0.432051, -0.5,
 -0.636068, -0.69949, -0.395751, -0.578427, -0.4, -0.562278, -0.716625,
 -0.864102, -1.00555, -1.49166, -1.87298, -2.4, -2.62311, -2.49264}
```

In[719]:=
```
OrderedPairs3 = Table[{}, n];
For[ i = 1, i ≤ n, i++,
 OrderedPairs3[[i]] = {xi[[i]], error[[i]]}
]
OrderedPairs3
```

Out[721]=
$$\{\{-17, -3.62311\}, \{-16, -2.\}, \{-15, -1.87298\}, \{-14, -1.41166\}, \{-13, -1.60555\}, \{-12, -0.714102\}, \{-11, -0.566625\}, \{-10, -0.412278\}, \{-9, -0.4\}, \{-8, -0.0784271\}, \{-7, -0.315751\}, \{-6, -0.19949\}, \{-5, -0.336068\}, \{-4, -0.4\}, \{-3, -0.332051\}, \{-2, -0.214214\}, \{-1, 0.\}, \{0, -1.11022 \times 10^{-16}\}, \{1, -0.25\}, \{2, -0.414214\}, \{3, -0.432051\}, \{4, -0.5\}, \{5, -0.636068\}, \{6, -0.69949\}, \{7, -0.395751\}, \{8, -0.578427\}, \{9, -0.4\}, \{10, -0.562278\}, \{11, -0.716625\}, \{12, -0.864102\}, \{13, -1.00555\}, \{14, -1.49166\}, \{15, -1.87298\}, \{16, -2.4\}, \{17, -2.62311\}, \{18, -2.49264\}\}$$

In[725]:=
```
ListPlot[OrderedPairs3, PlotLabel → "Error From Spline Compared to Base Function"]
```

Out[725]=