INTERPOLATION METHODS FOR ANALYZING UNDER-SAMPLED IMPEDANCE DATA

Wissam Razouki

CSE 3802

INTRODUCTION

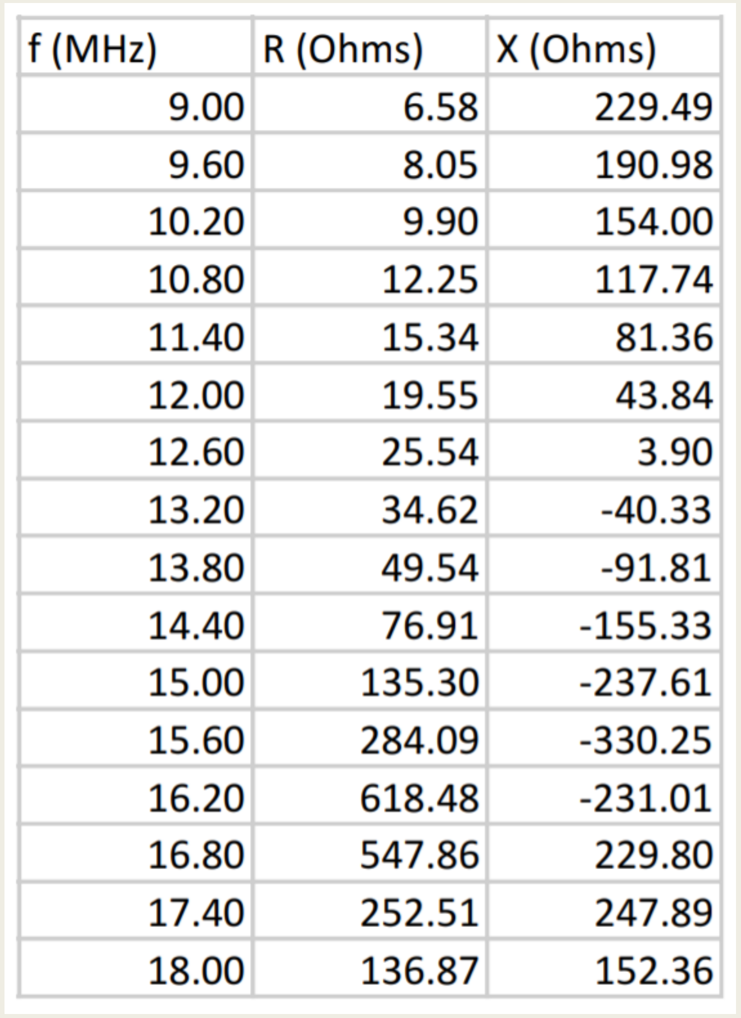
The Problem

The impedance data for an antenna is given as shown

- There is a total of 16 data points

Using different methods of interpolation, we must find:

1. The frequency f_1 where the antenna goes through its first resonance (Reactance $X = 0$ as $+X \to -X$)
2. The input resistance R_1 at f_1
3. The frequency f_2 where the antenna goes through an anti-resonance (Reactance $X = 0$ as $-X \to +X$)
4. The input resistance R_2 at f_2



f (MHz)	R (Ohms)	X (Ohms)
9.00	6.58	229.49
9.60	8.05	190.98
10.20	9.90	154.00
10.80	12.25	117.74
11.40	15.34	81.36
12.00	19.55	43.84
12.60	25.54	3.90
13.20	34.62	-40.33
13.80	49.54	-91.81
14.40	76.91	-155.33
15.00	135.30	-237.61
15.60	284.09	-330.25
16.20	618.48	-231.01
16.80	547.86	229.80
17.40	252.51	247.89
18.00	136.87	152.36

Methods Used

Reference method:

- MATLAB Cubic Spline Interpolation

Experimental methods:

- Newton Polynomial Interpolation
- Quadratic Spline Interpolation (with Clamped Boundary)

TESTING METHOD AND SOLUTIONS

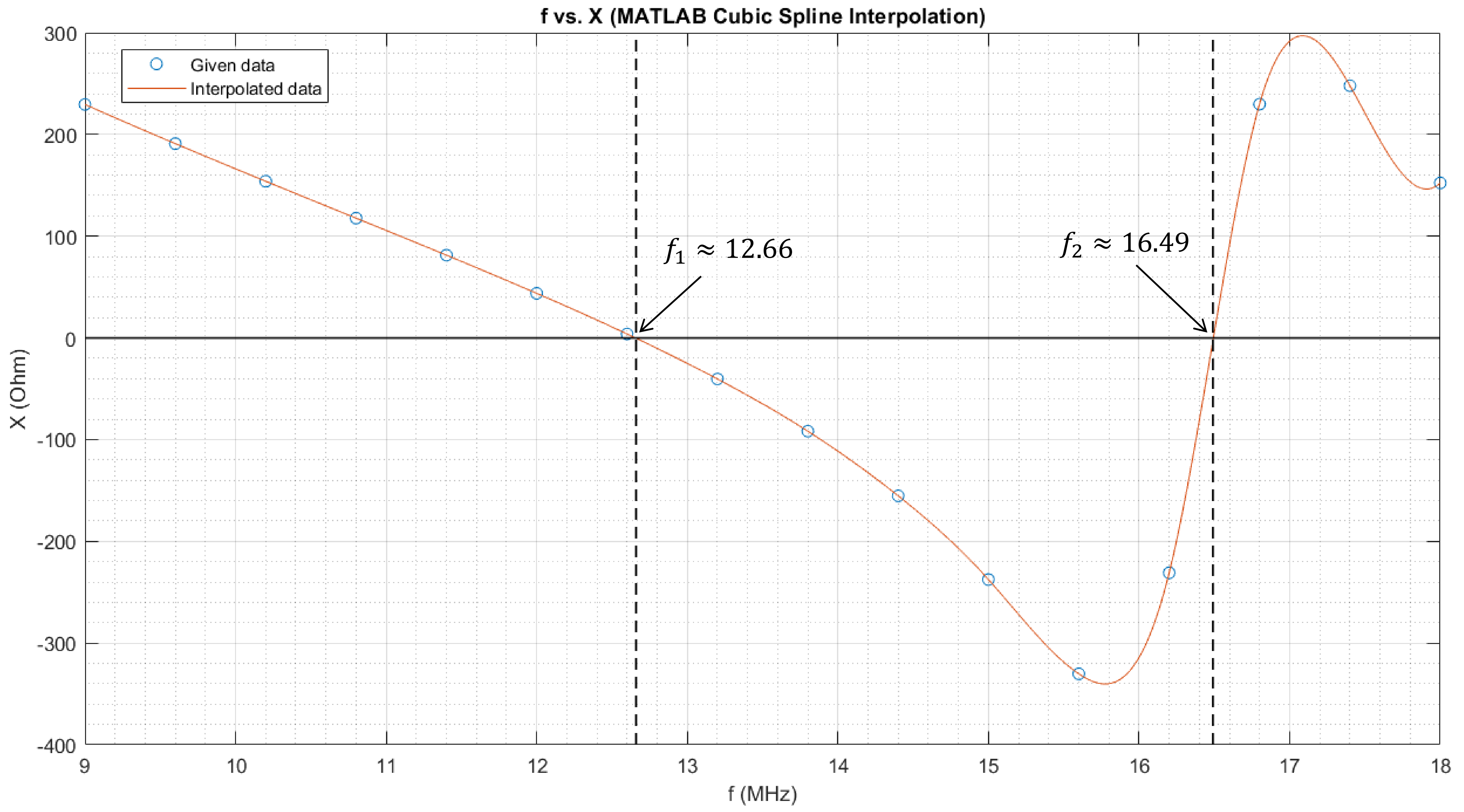
The Reference Method

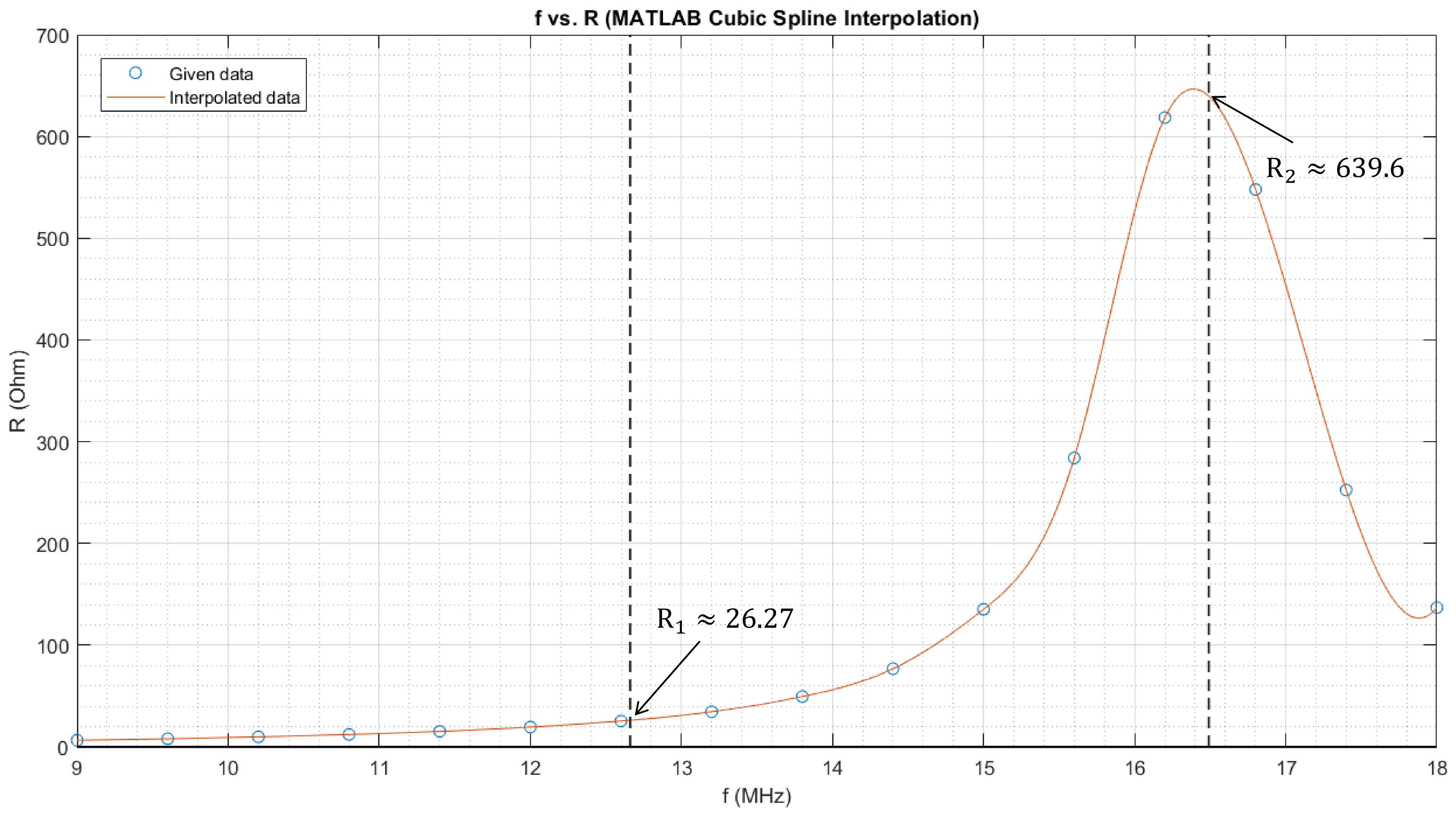
Since the exact solutions to the problem are not given, we need a way to empirically evaluate and compare our experimental methods.

An alternative way to do this is by using a **reference method**:

- MATLAB provides the built-in function `interp1(…)` for interpolating over one-dimensional data using various methods
- In order to obtain maximum possible accuracy, we must interpolate over the **entire dataset** using **cubic splines**
  - *First, we interpolate f vs. X and find the f values*
  - *Next, we interpolate f vs. R and find the R values at the f values found previously*

The resulting plots for “f vs. X” and “f vs. R” (with solutions marked) are shown below


f vs. X (MATLAB Cubic Spline Interpolation)
Given data
Interpolated data
$f_1 \approx 12.66$
$f_2 \approx 16.49$
X (Ohm)
f (MHz)
300
200
100
0
-100
-200
-300
-400
9
10
11
12
13
14
15
16
17
18


f vs. R (MATLAB Cubic Spline Interpolation)
Given data
Interpolated data
$R_2 \approx 639.6$
$R_1 \approx 26.27$
R (Ohm)
f (MHz)
0
100
200
300
400
500
600
700
9
10
11
12
13
14
15
16
17
18

The Reference Solutions

Given the results above, we define our **reference** solutions as

1. $f_1 \approx 12.66\ MHz$
2. $R_1 \approx 26.27\ \Omega$
3. $f_2 \approx 16.49\ MHz$
4. $R_2 \approx 639.60\ \Omega$

We can use these values to compare our experimental solutions for **Newton polynomial** and **quadratic spline interpolation**

- Of course, we are assuming some degree of uncertainty as these values are not the exact solutions, but they are the best we can come up with

EXPERIMENTAL RESULTS AND ANALYSIS

Newton Polynomial

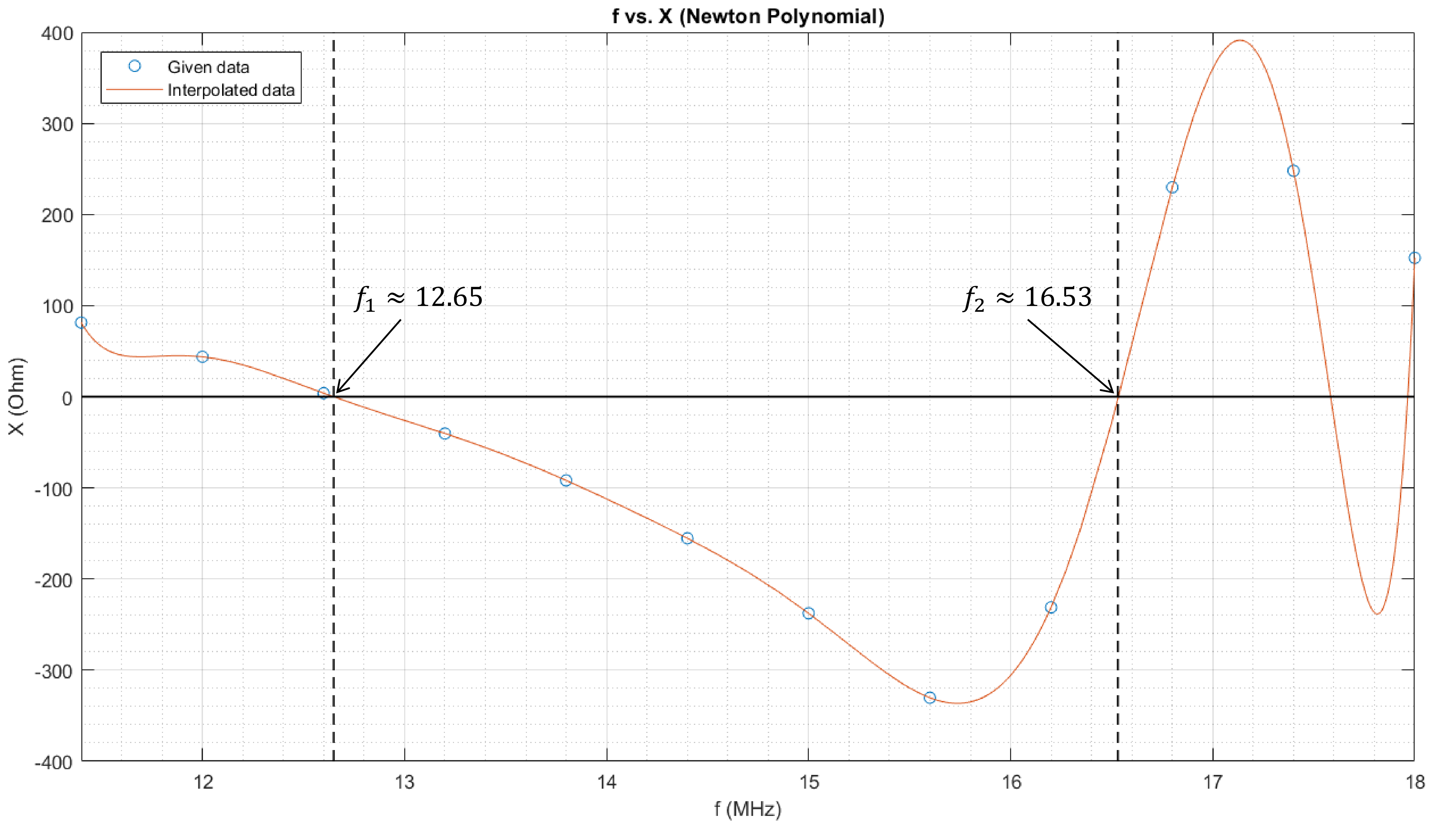
Newton’s method of interpolation falls under the “interpolating polynomial” family

- It requires the use of divided differences for calculating the coefficients of the polynomial
  - *This can result in a lot of terms in the code for many data points*

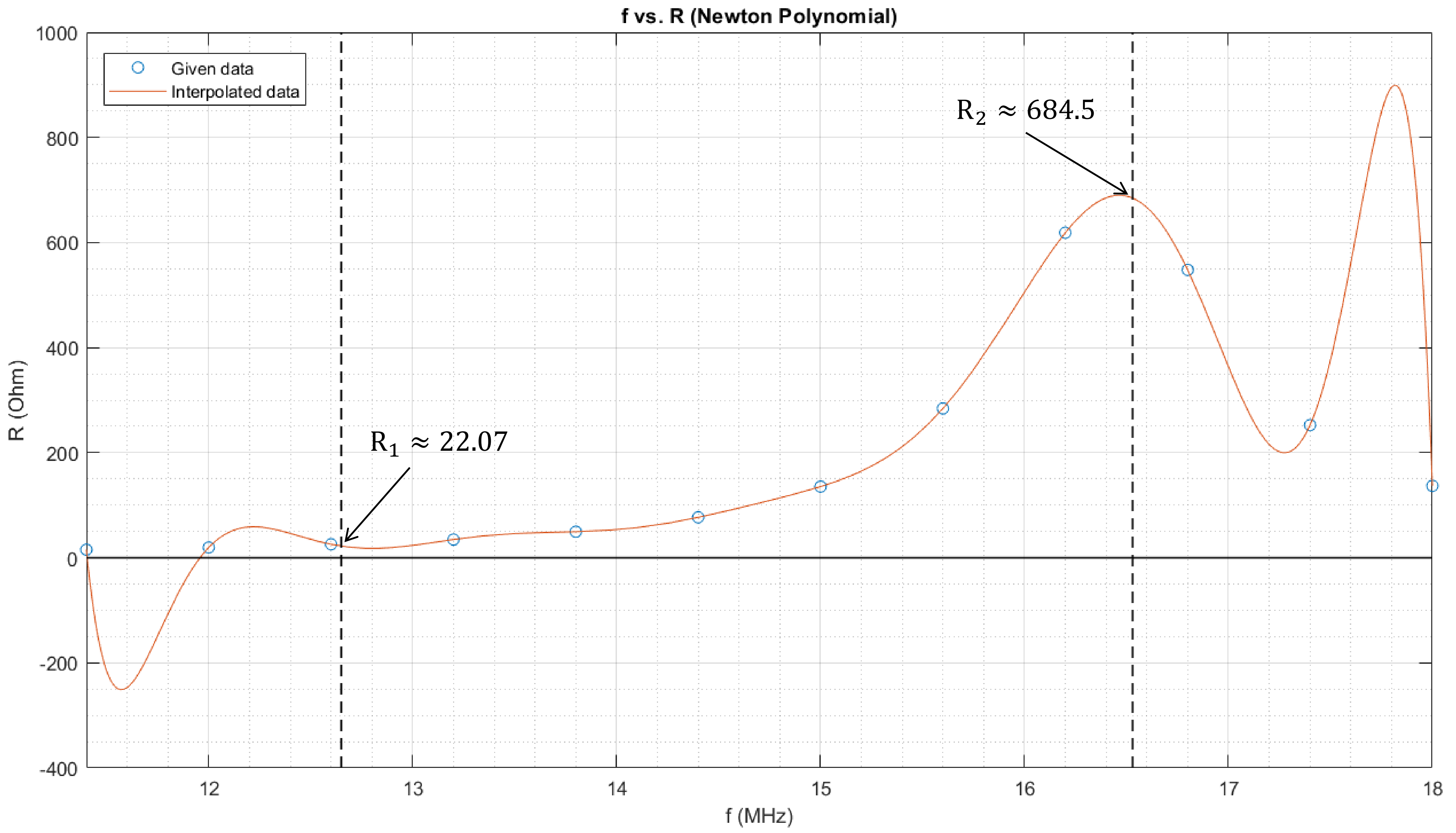
Since we only need to find the frequency and input resistance where the reactance goes from positive to negative and vice versa, we can choose to interpolate over only the necessary data points

- Choosing a small limit might cause loss of significance, while choosing a large limit might result in incorrect values as the polynomial becomes more complex
- For this method, we will interpolate over the data where $f \in [11.40,\ 18.00]$ (for a total of 12 data points)

The resulting plots for “f vs. X” and “f vs. R” (with solutions marked) are shown below



f vs. X (Newton Polynomial)
Given data
Interpolated data
$f_1 \approx 12.65$
$f_2 \approx 16.53$
X (Ohm)
f (MHz)
400
300
200
100
0
-100
-200
-300
-400
12
13
14
15
16
17
18


f vs. R (Newton Polynomial)
Given data
Interpolated data
$R_2 \approx 684.5$
$R_1 \approx 22.07$
R (Ohm)
f (MHz)
1000
800
600
400
200
0
-200
-400
12
13
14
15
16
17
18

Quadratic Spline

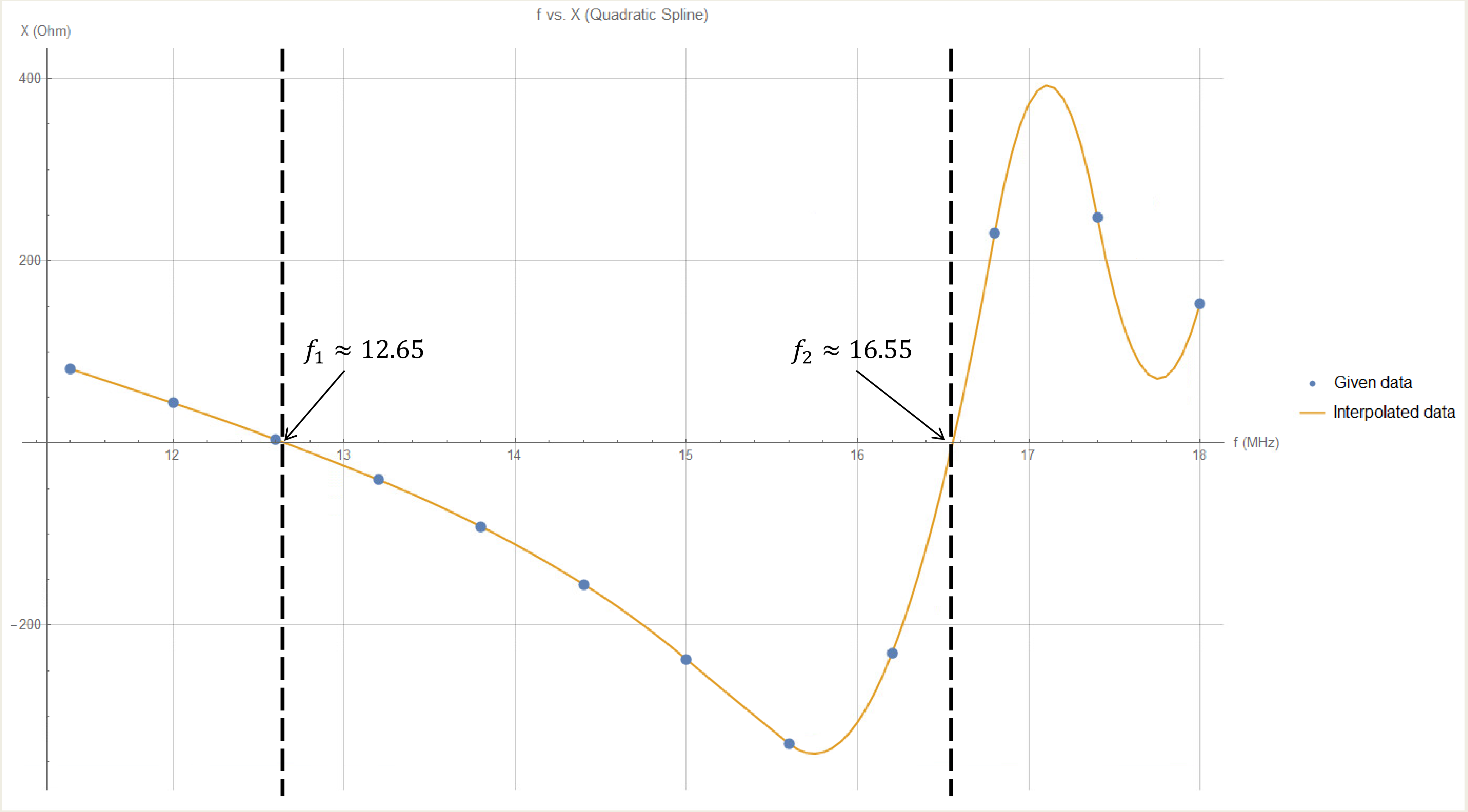
Quadratic splines fall under the “spline” family

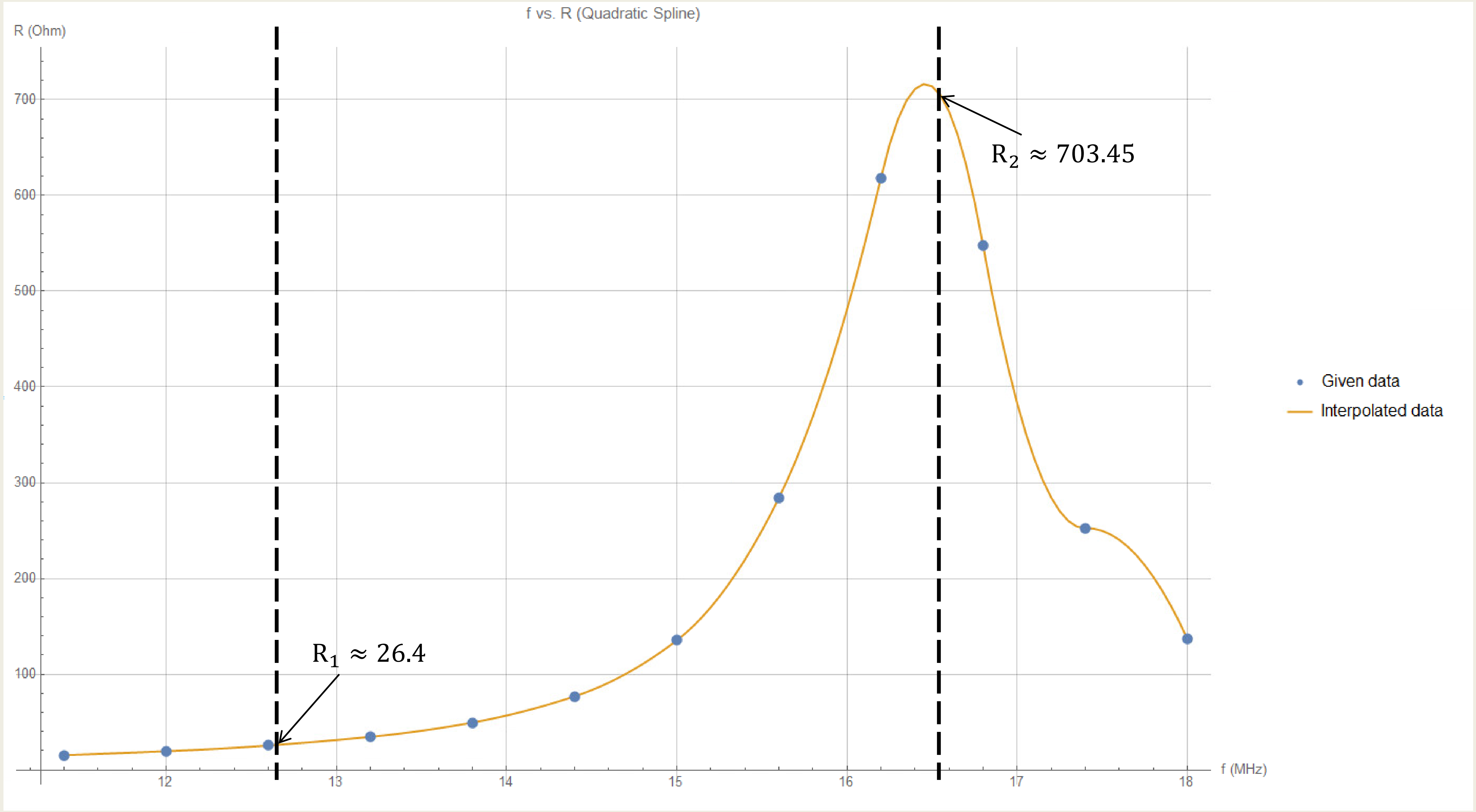
- It uses a set of piecewise functions to build a less complex polynomial, resulting in similar results while avoiding Runge’s phenomenon of higher order polynomials

In order to do a proper comparison, we should use the same limits as before

- For this method, we will interpolate over the data where $f \in [11.40,\ 18.00]$ (for a total of 12 data points)
  - *This was implemented in Mathematica using a* ***clamped boundary*** *(MATLAB does not handle piecewise functions well)*

The resulting plots for “f vs. X” and “f vs. R” (with solutions marked) are shown below

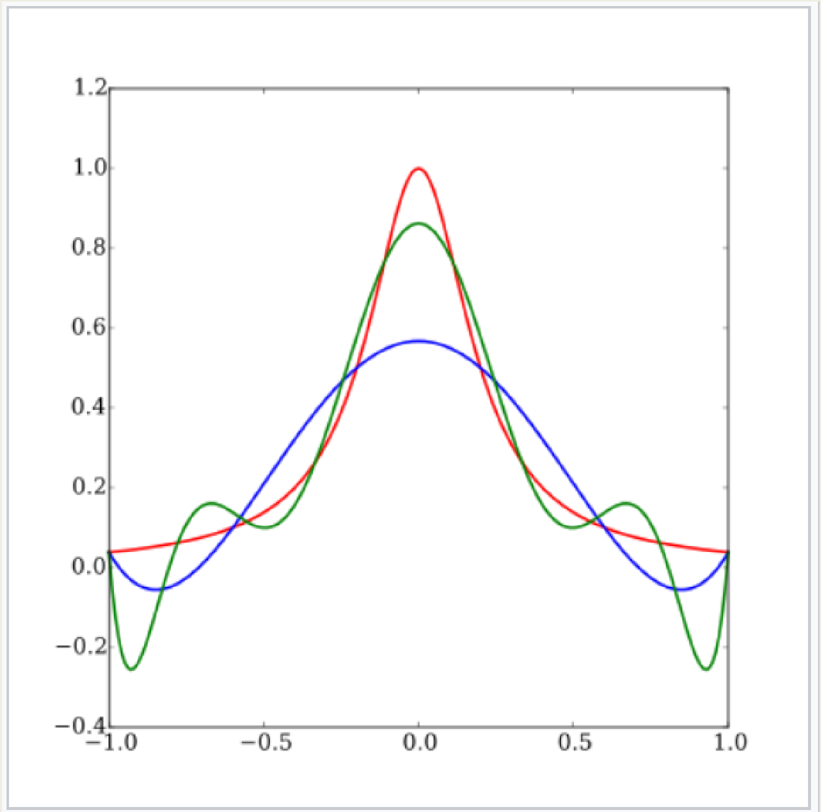

f vs. X (Quadratic Spline)
X (Ohm)
400
200
-200
12
13
14
15
16
17
18
f (MHz)
$f_1 \approx 12.65$
$f_2 \approx 16.55$
Given data
Interpolated data


f vs. R (Quadratic Spline)
R (Ohm)
700
600
500
400
300
200
100
12
13
14
15
16
17
18
f (MHz)
$R_2 \approx 703.45$
$R_1 \approx 26.4$
● Given data
Interpolated data

Graphical Analysis

Looking at the graphs above, we notice that the "waves" in the Newton polynomial are much greater at the endpoints compared to the quadratic spline

- This is a simple case of **Runge's phenomenon**, which shows that interpolating polynomials of a higher degree tend to oscillate more at the edges of the interval
- In this case, we are analyzing a 11th degree Newton polynomial vs. a 2nd degree quadratic spline polynomial



The red curve is the Runge function.
The blue curve is a 5th-order interpolating polynomial (using six equally spaced interpolating points).
The green curve is a 9th-order interpolating polynomial (using ten equally spaced interpolating points).

Runge's phenomenon (from Wikipedia)

Error Analysis

The formula for **percent error** is

$$\frac{|E-R|}{R} \times 100\%$$

Where E is the **experimental** value, and R is the **reference** value.

Using this on each of the solutions for each experimental method, we can come up with concrete results telling us which method was better in finding the necessary data (assuming our reference, MATLAB’s cubic spline interpolation, is most accurate).

Error Analysis Results

Newton Polynomial

Value	Percent Error
f_1	0.08%
R_1	15.99%
f_2	0.24%
R_2	7.02%
Average percent error for all values	5.83%

Quadratic Spline

Value	Percent Error
f_1	0.08%
R_1	0.49%
f_2	0.24%
R_2	9.98%
Average percent error for all values	2.70%

Risk, Challenges, and Sources of Error

It is impossible to know the exact solutions for these values. The best we can do is approximate using interpolation.

- Cubic splining (using all data points with the built-in MATLAB function) should **theoretically** provide the best possible results, thus we set it as our **reference** method and use it to evaluate our **experimental** methods

Due to the solutions being relatively close to each other for all methods, we can say with a **high degree of confidence** that both experimental methods provide fairly correct solutions.

Regardless, **sources of error** could be attributed to many things, including:

- Runge's phenomenon
- Loss of significance
- A limited data set
- Large variances in subsequent points (X goes from $-231.01\ \Omega$ to $229.80\ \Omega$, for example)

Further Analysis and Conclusions

According to these results, we can conclude that **quadratic splining is better suited** for this problem than Newton polynomial (*on average*).

- This is to be expected, as quadratic splining uses a lower order polynomial, resulting in less variation in the data

For a simple problem like this (where we only need to query a few specific points), both methods provide sufficiently accurate results.

However, were we given even more data (hundreds or even thousands of data points), and tasked with providing an interpolation function for finding sufficiently accurate data at **all** inputs, quadratic splining becomes significantly more appealing than Newton polynomial.