HW 8

Derek Walker

1: Equispaced nodes

(a): Lagrange interpolation

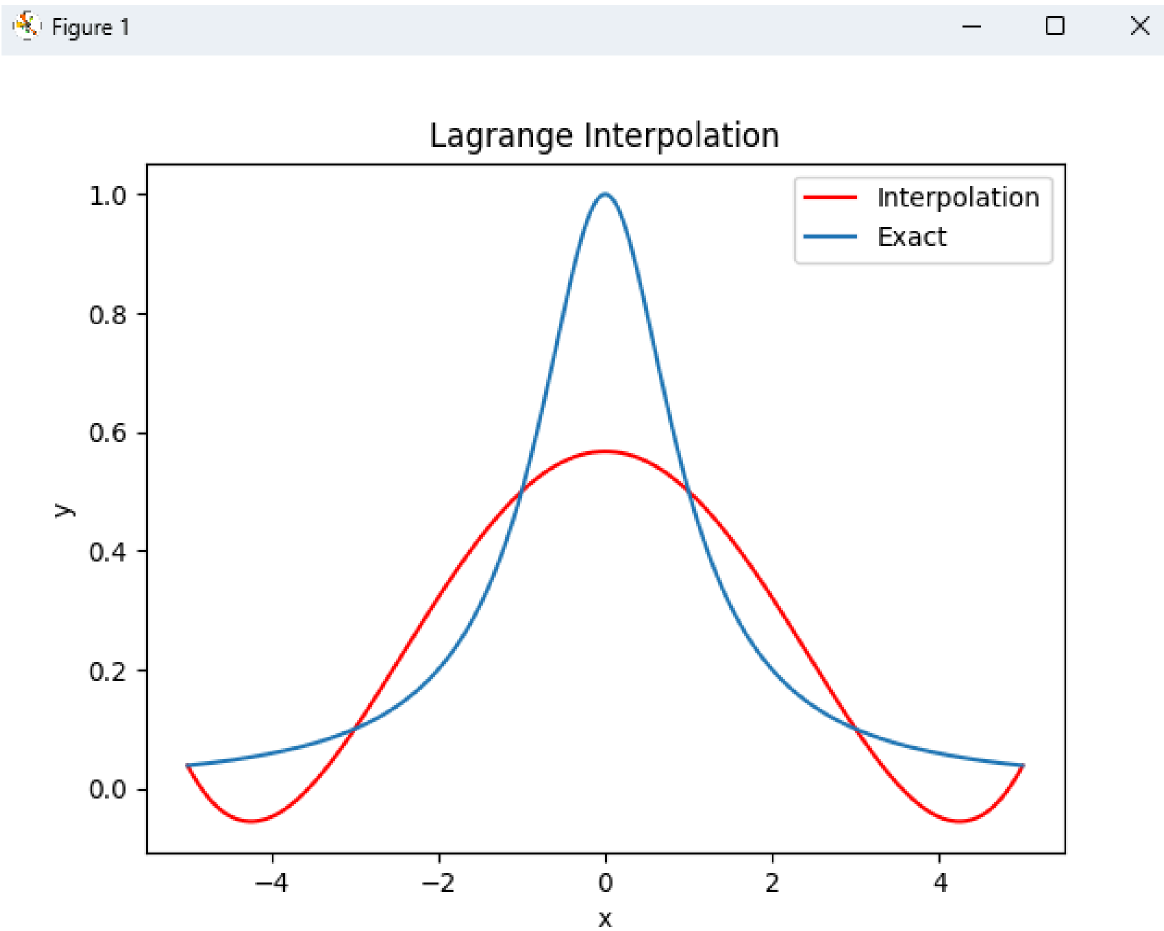


Figure 1: N = 5

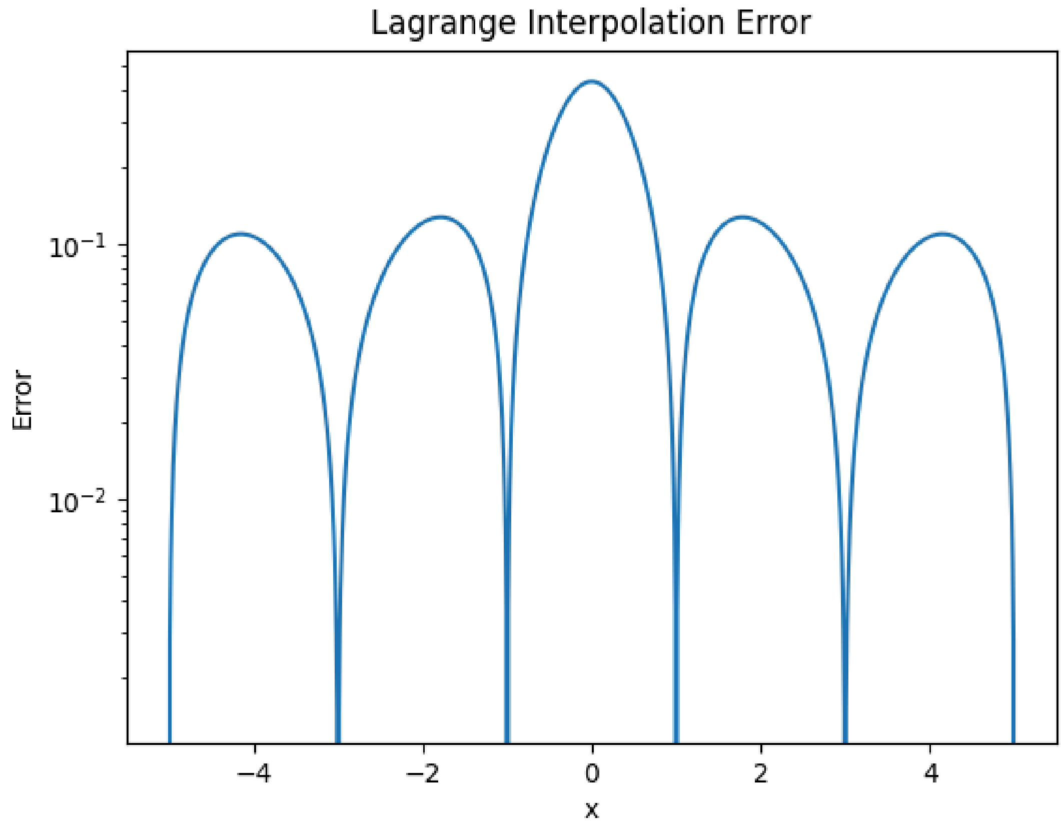


Figure 2: N = 5

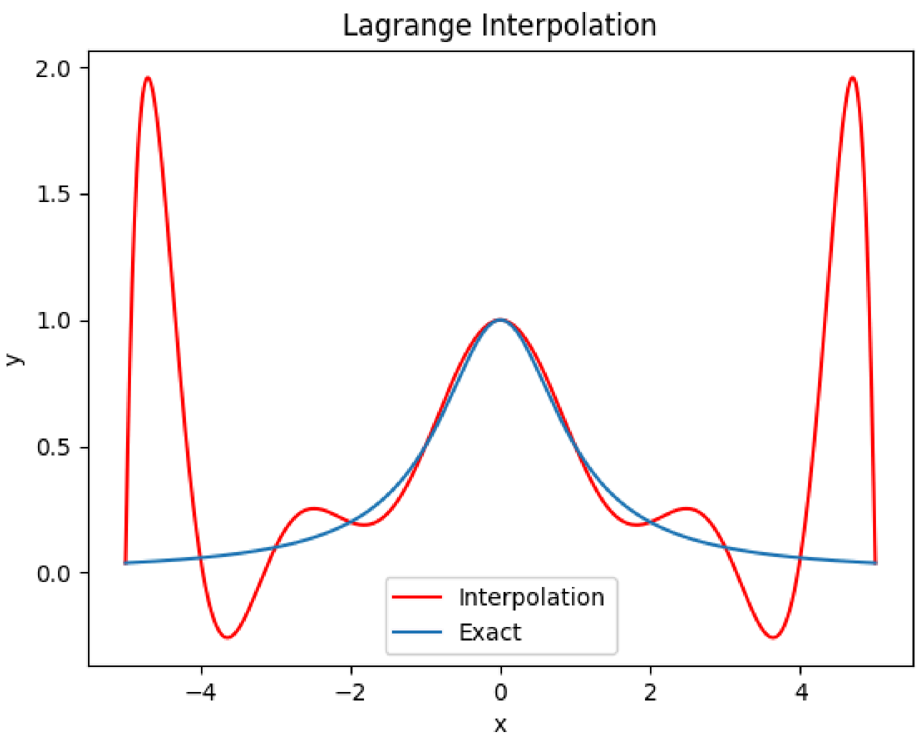


Figure 3: N = 10

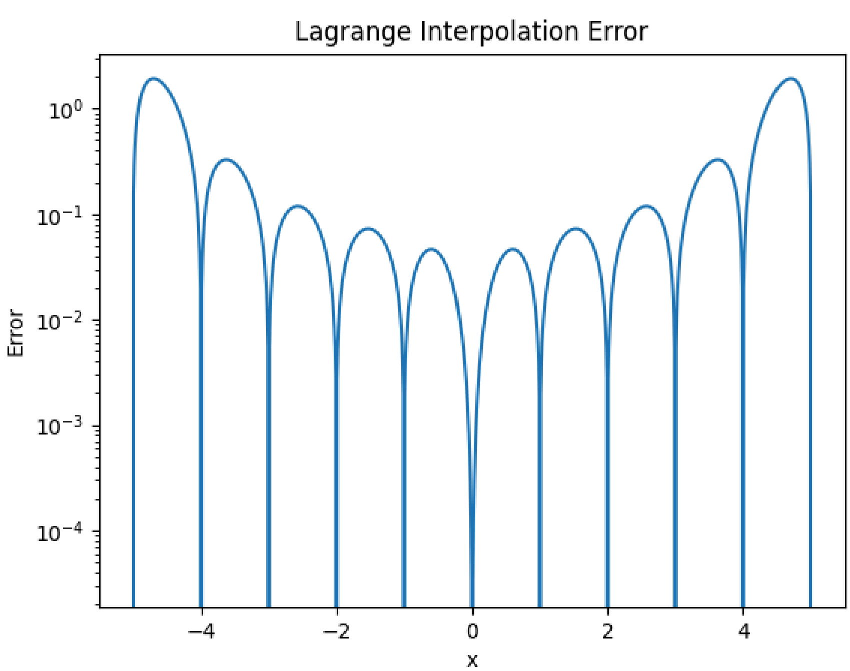


Figure 4: N = 10

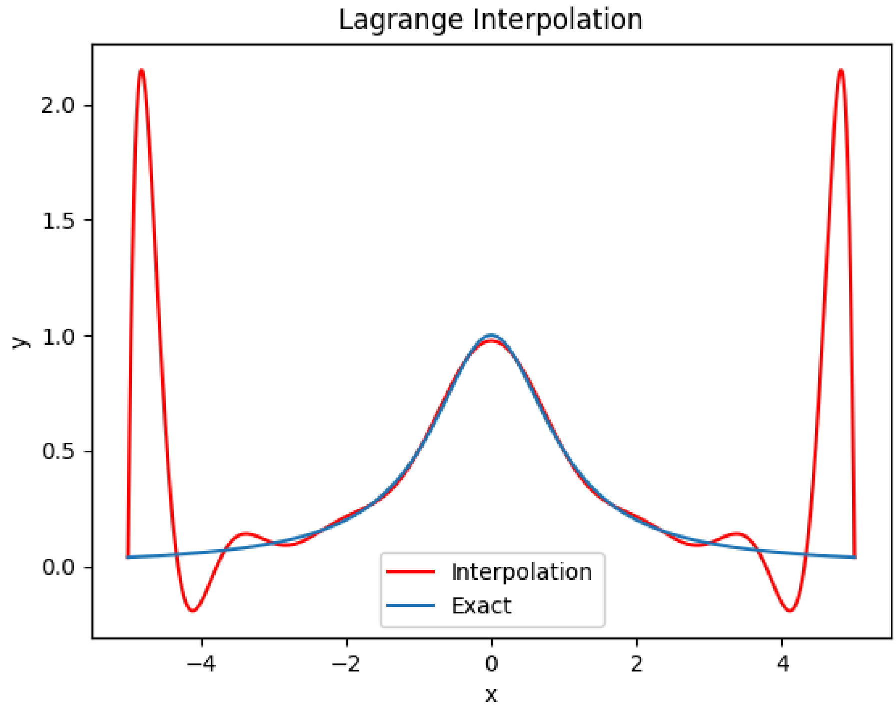


Figure 5: N = 15

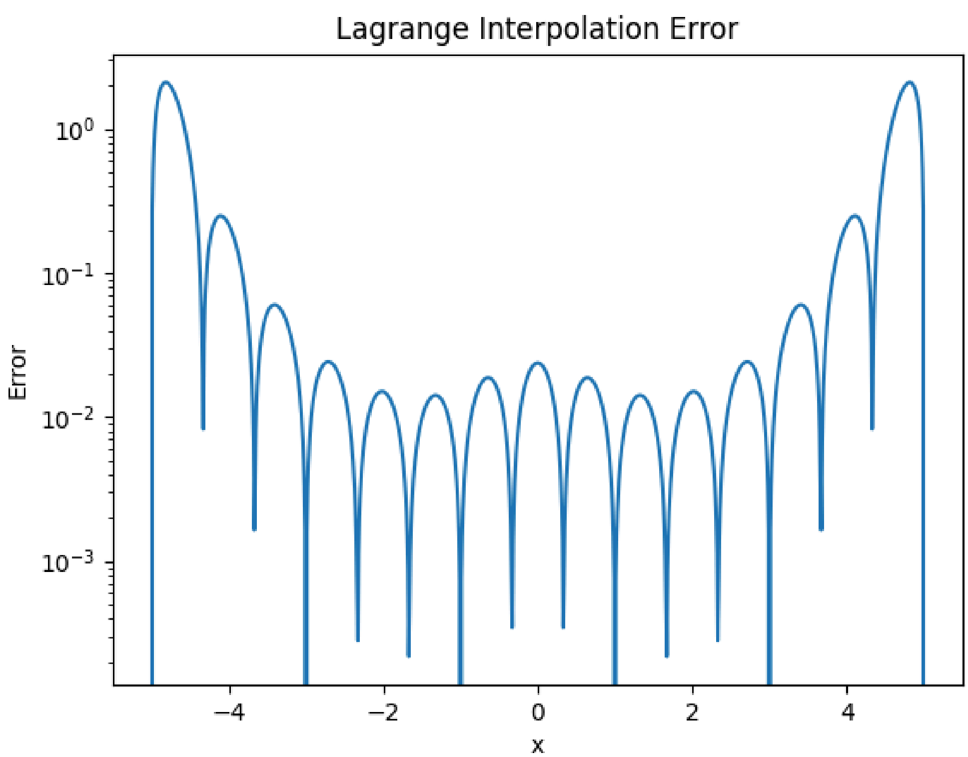


Figure 6: N = 15

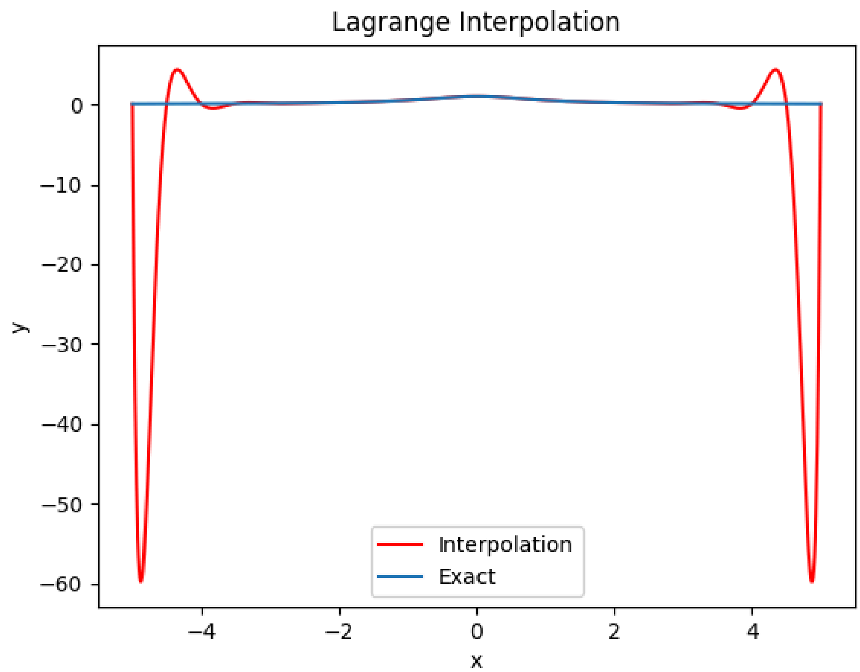


Figure 7: N = 20

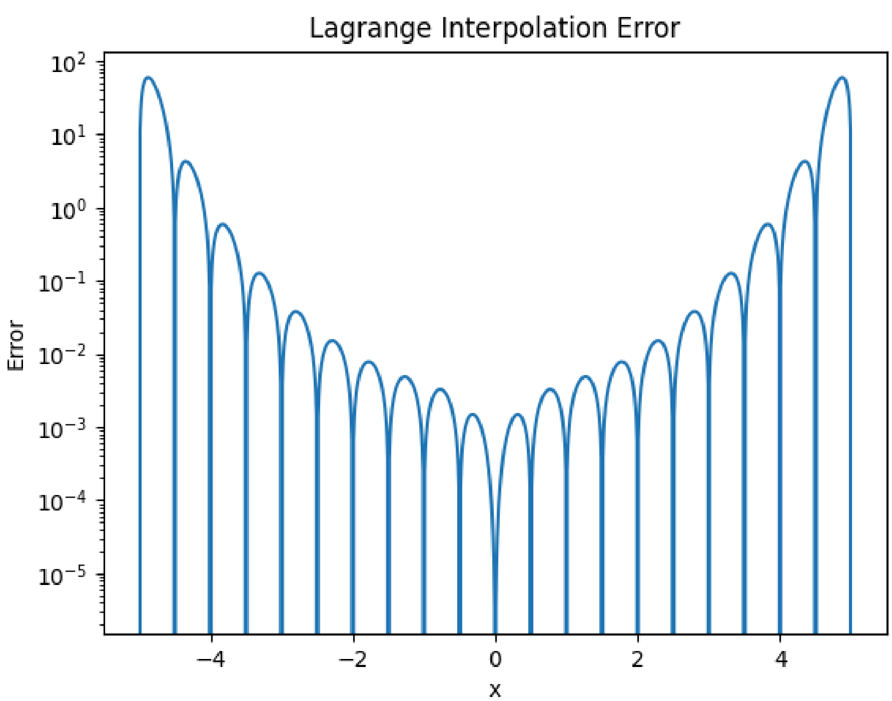


Figure 8: N = 20

Lagrange interpolation performs terribly once N gets larger due to the error at the endpoints which is caused by the choice of interpolation nodes (Runge's phenomenon). However, it performs quite well for relatively small x.

(b): Hermite interpolation

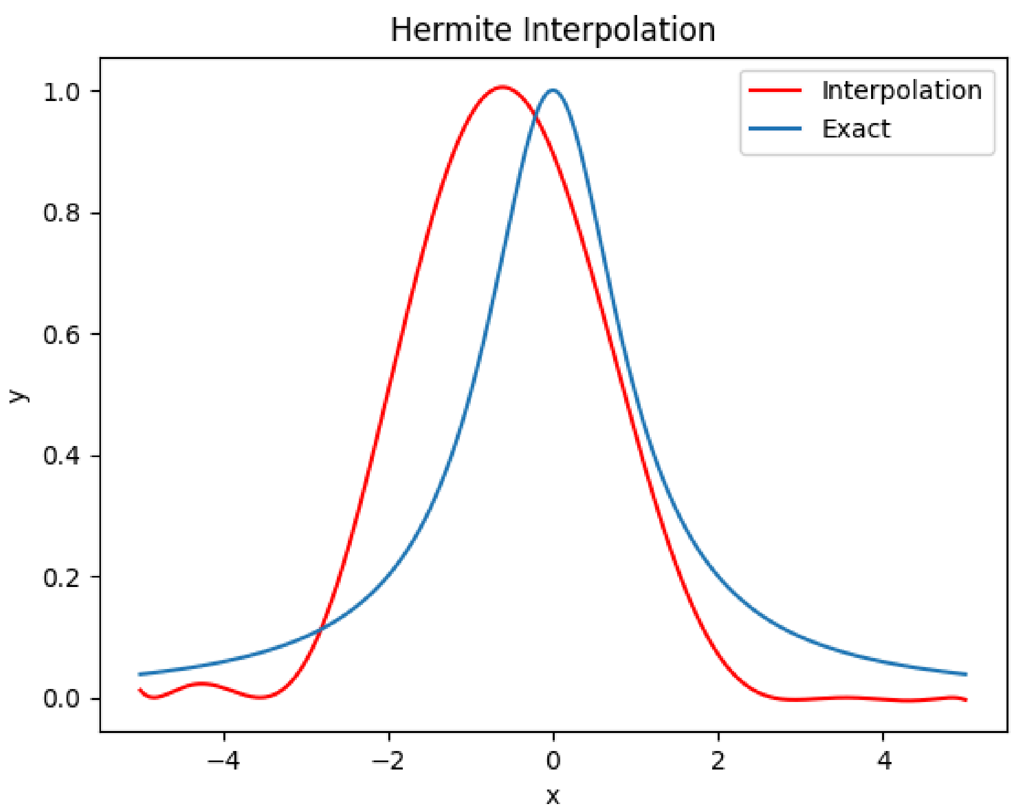


Figure 9: N = 5

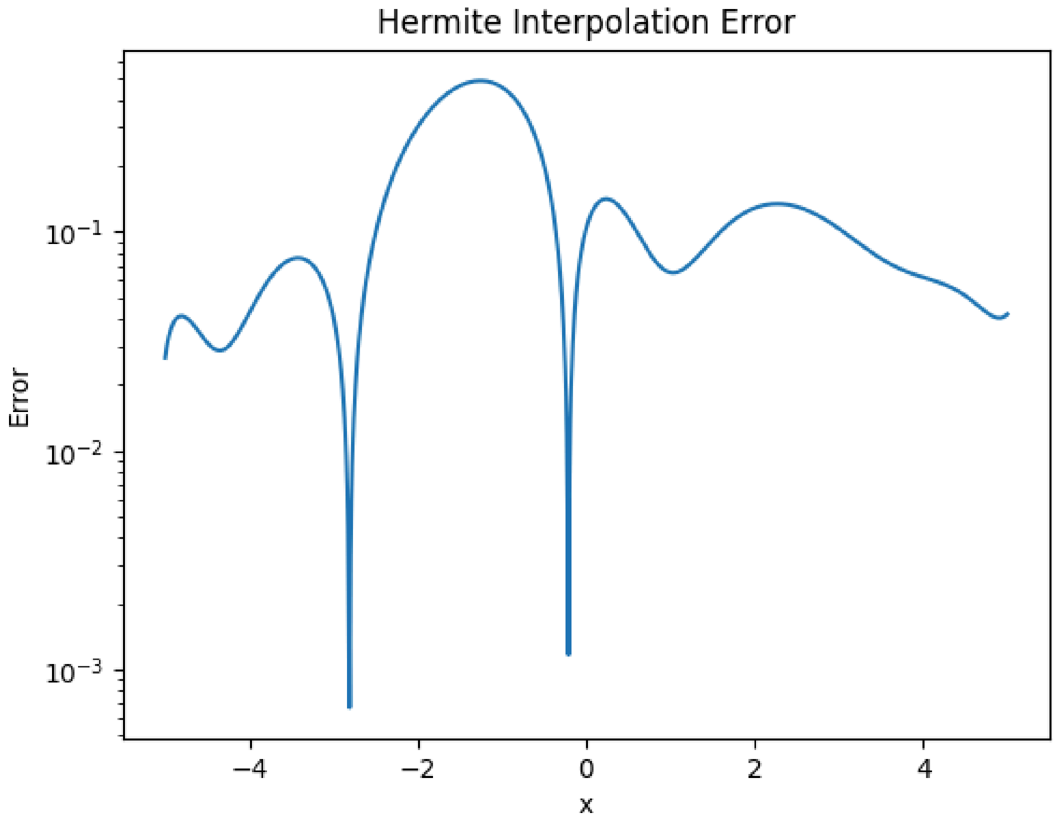


Figure 10: N = 5

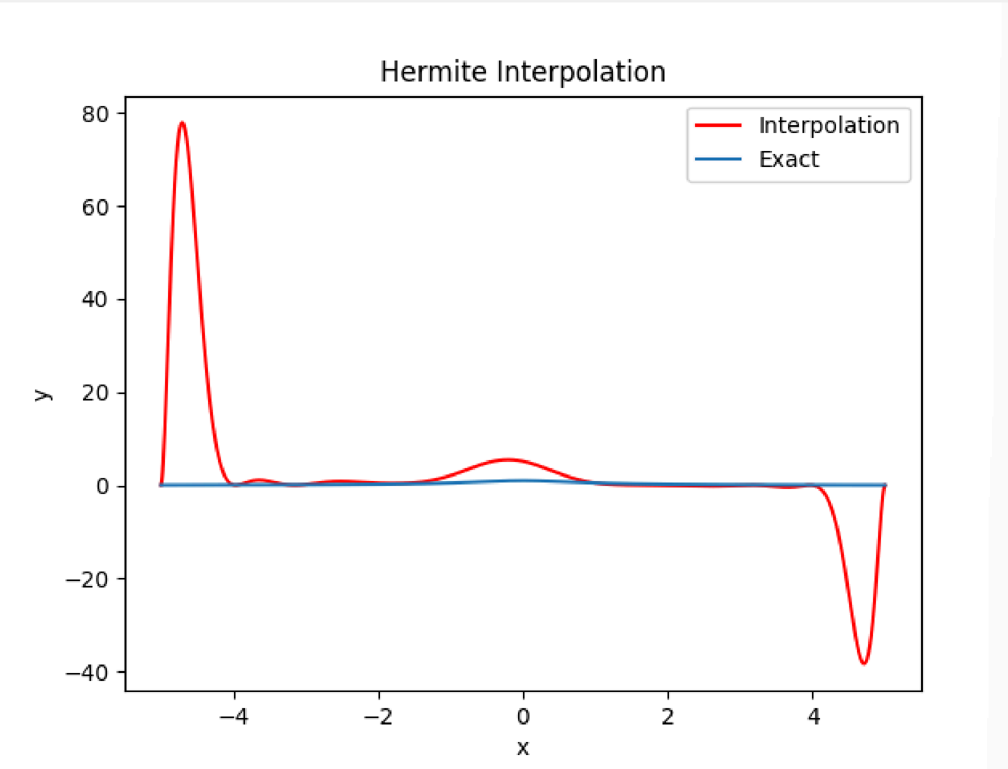


Figure 11: N = 10

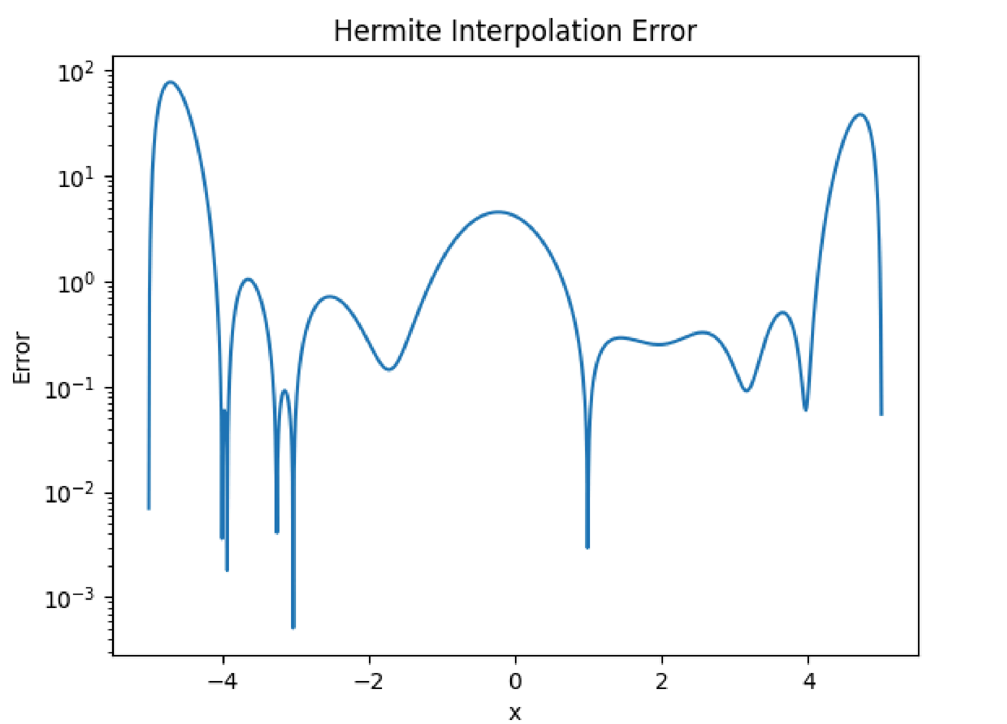


Figure 12: N = 10

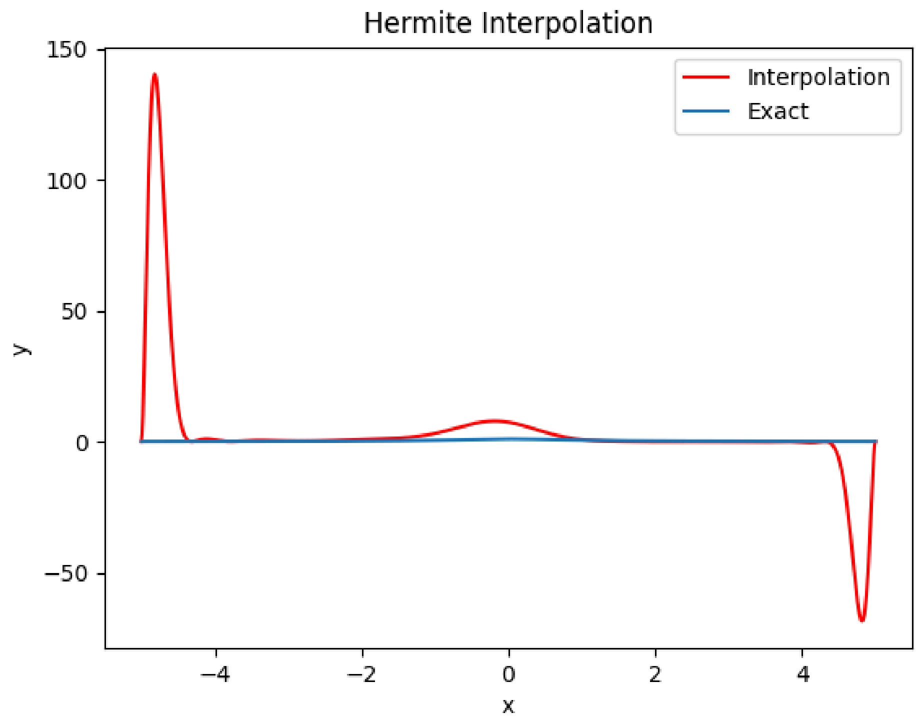


Figure 13: N = 15

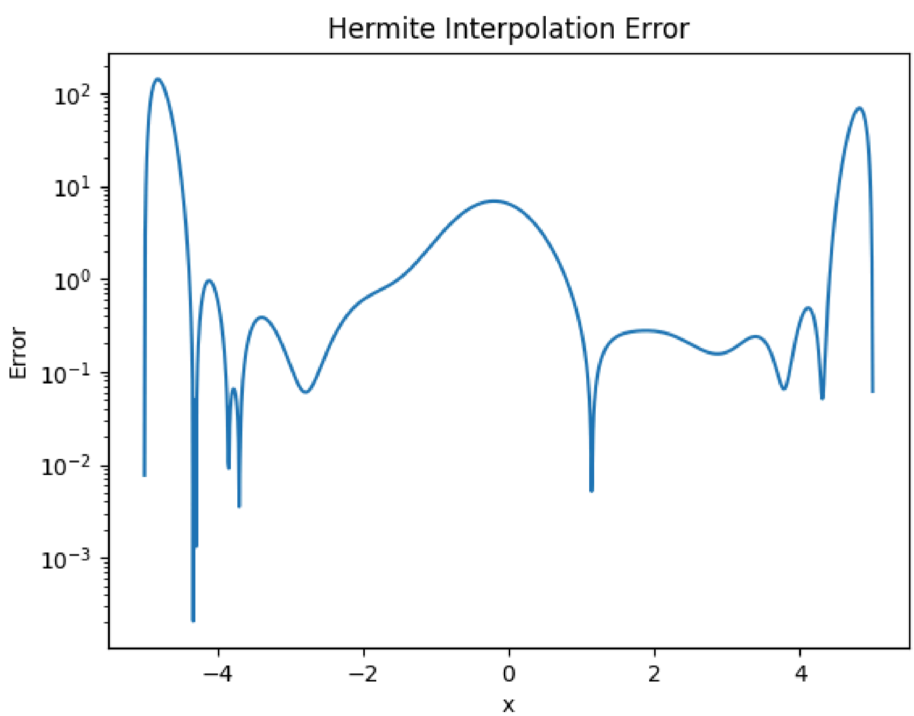


Figure 14: N = 15

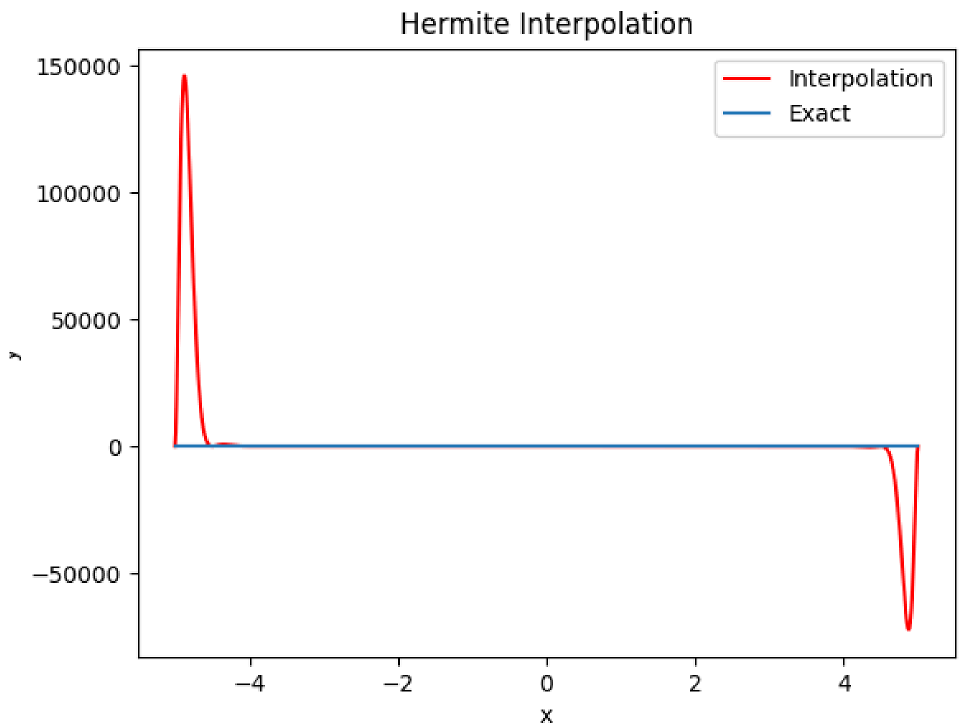


Figure 15: N = 20

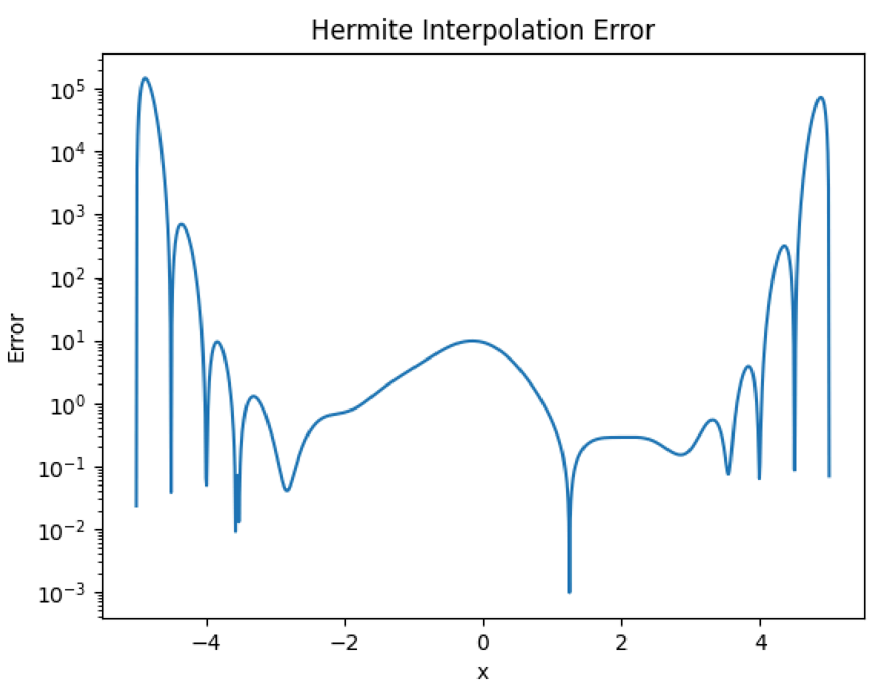


Figure 16: N = 20

Like Lagrange interpolation, Hermite interpolation suffers from Runge's phenomenon, so the endpoints get less accurate for larger N. I would say it's worse than Lagrange in this scenario looking just at the graphs of the interpolant because even at N = 10 Runge's phenomenon kicks in pretty good.

(c): Natural cubic spline interpolation

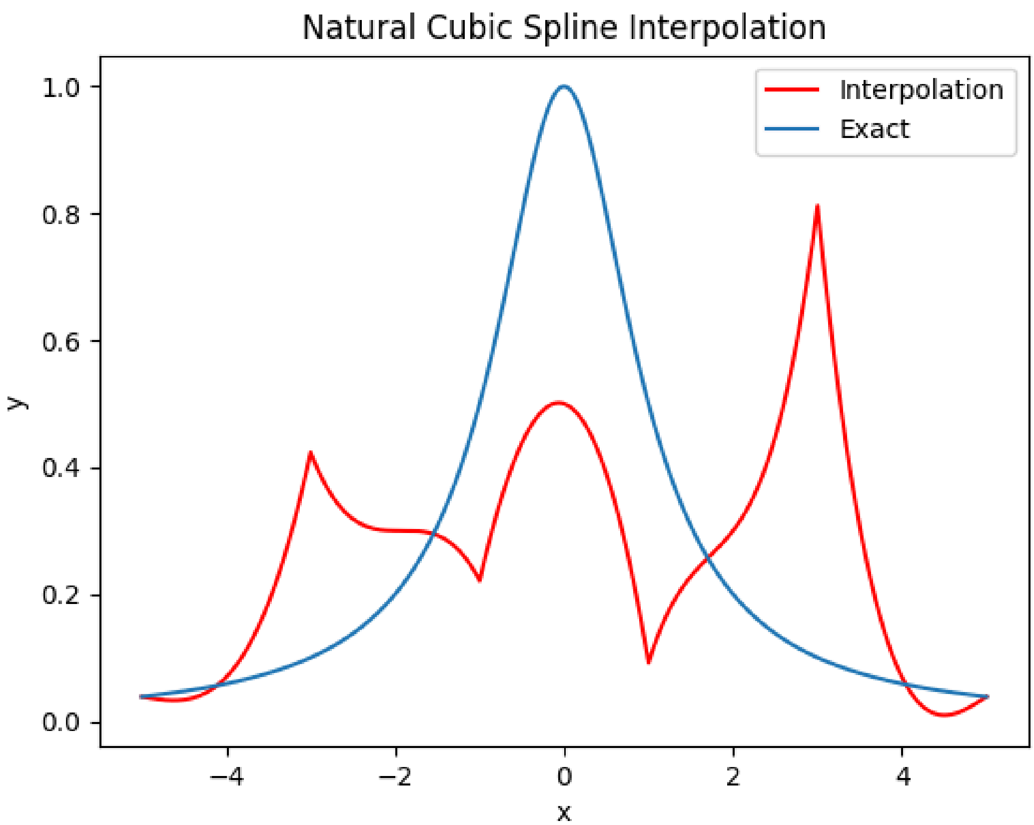


Figure 17: N = 5

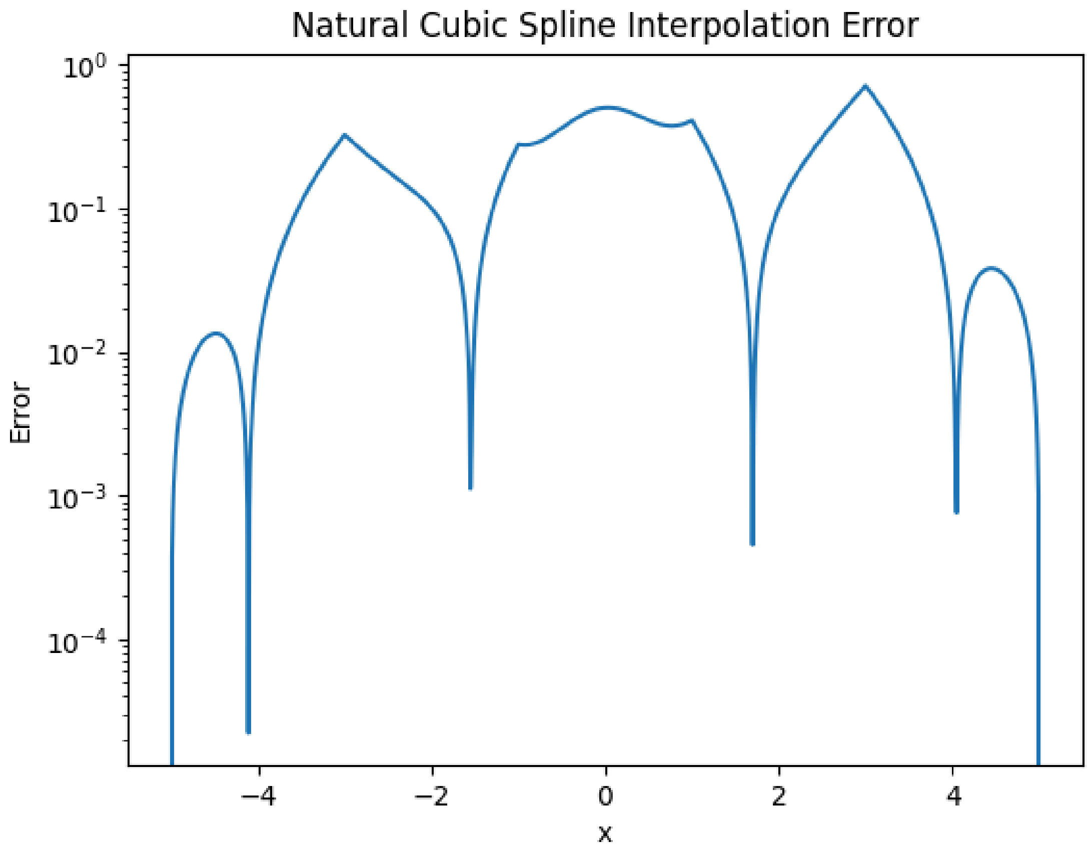

Natural Cubic Spline Interpolation Error
Error
x
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
−4
−2
0
2
4

Figure 18: N = 5

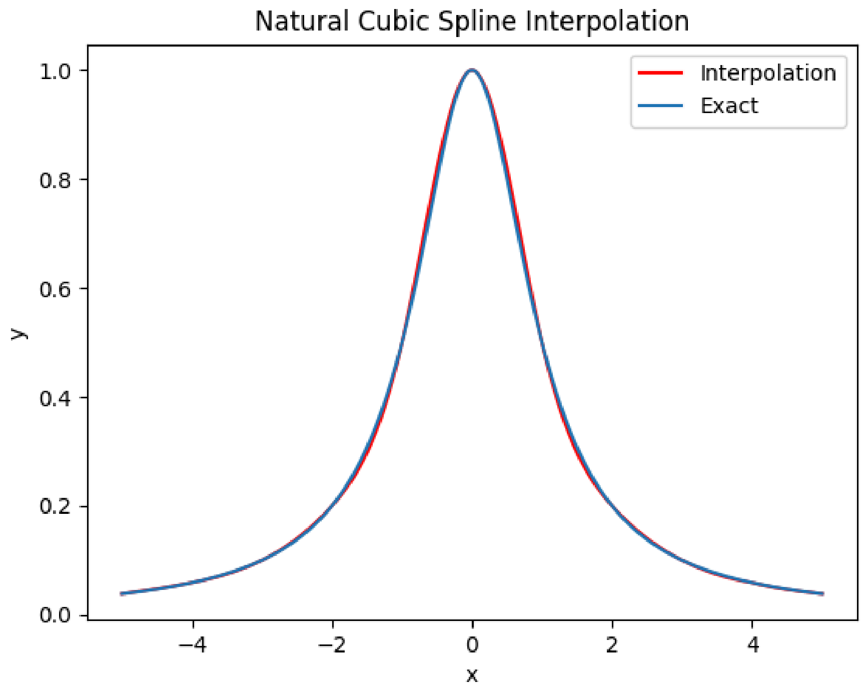


Figure 19: N = 10

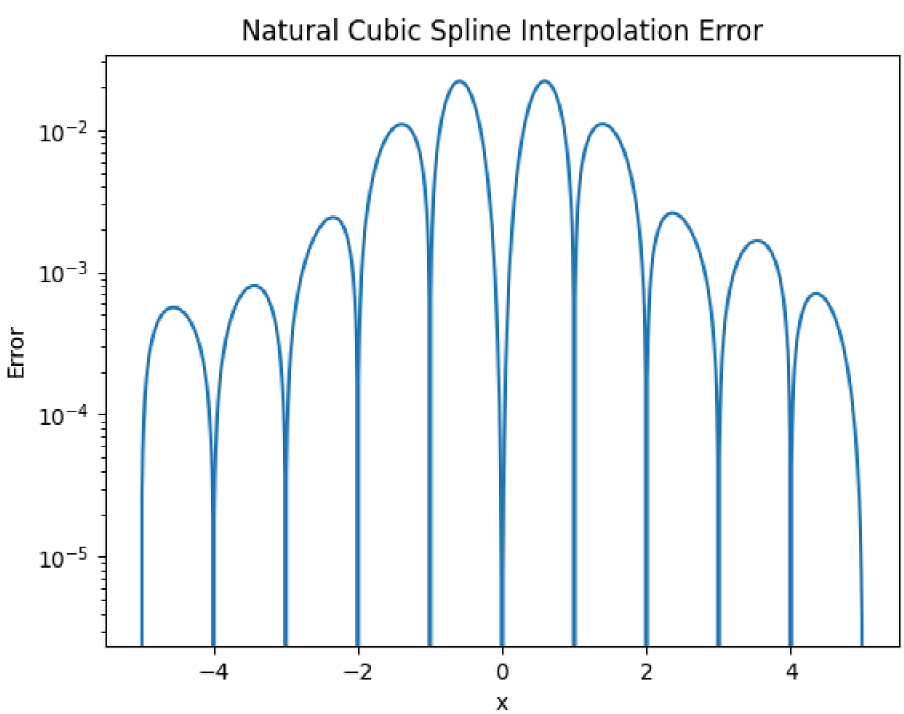


Figure 20: N = 10

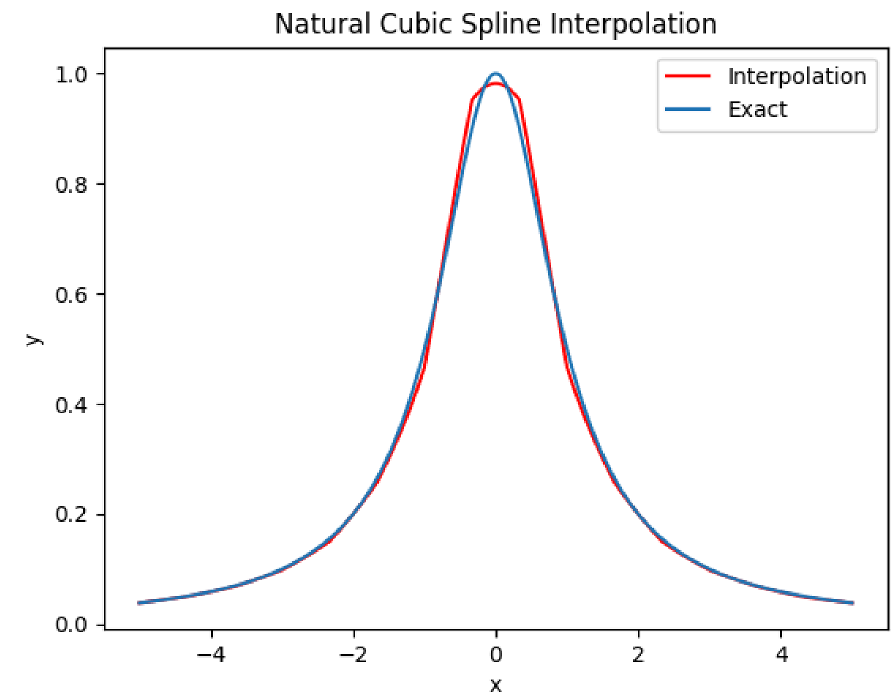


Figure 21: N = 15

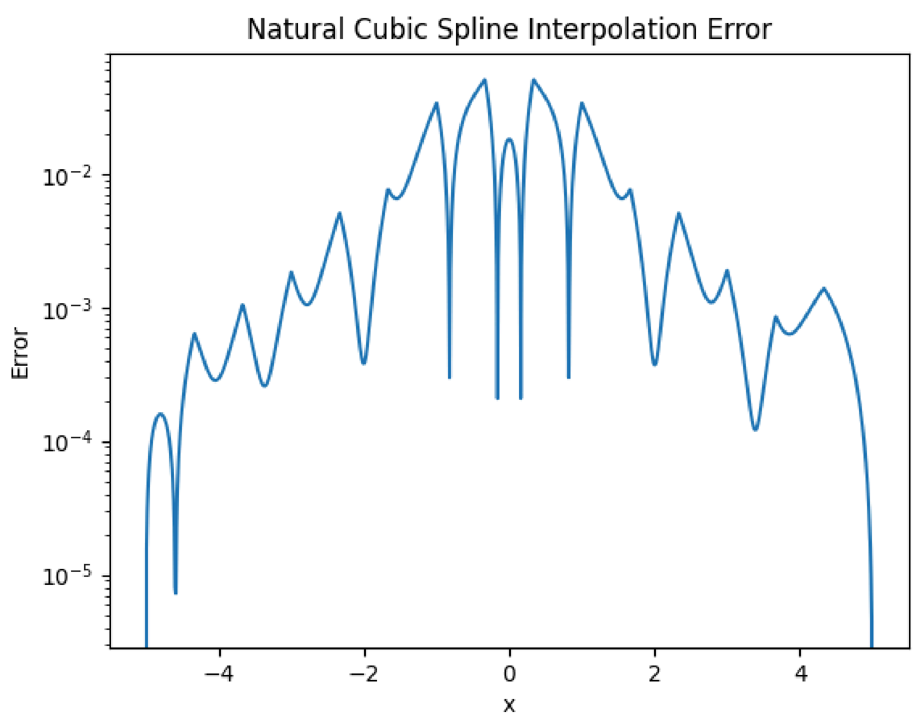


Figure 22: N = 15

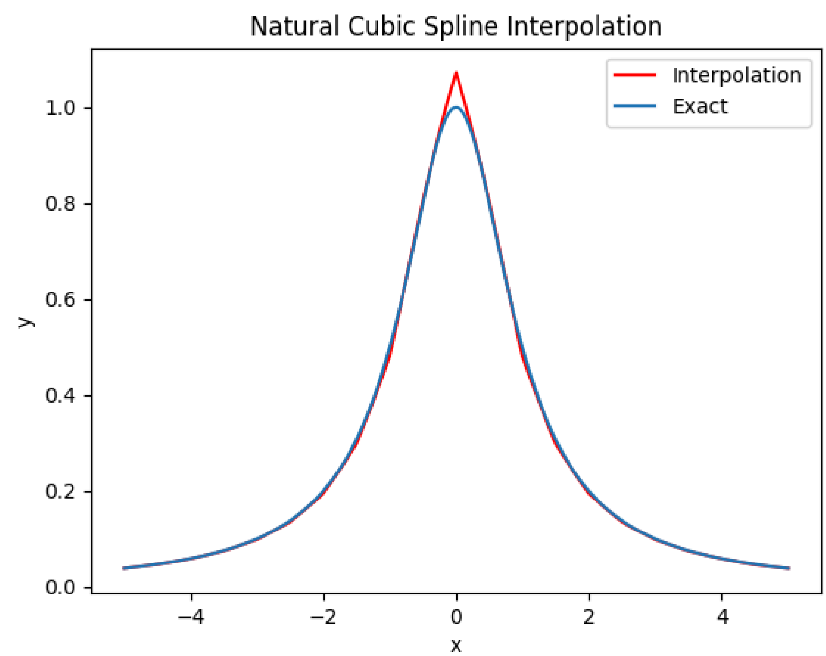


Figure 23: N = 20

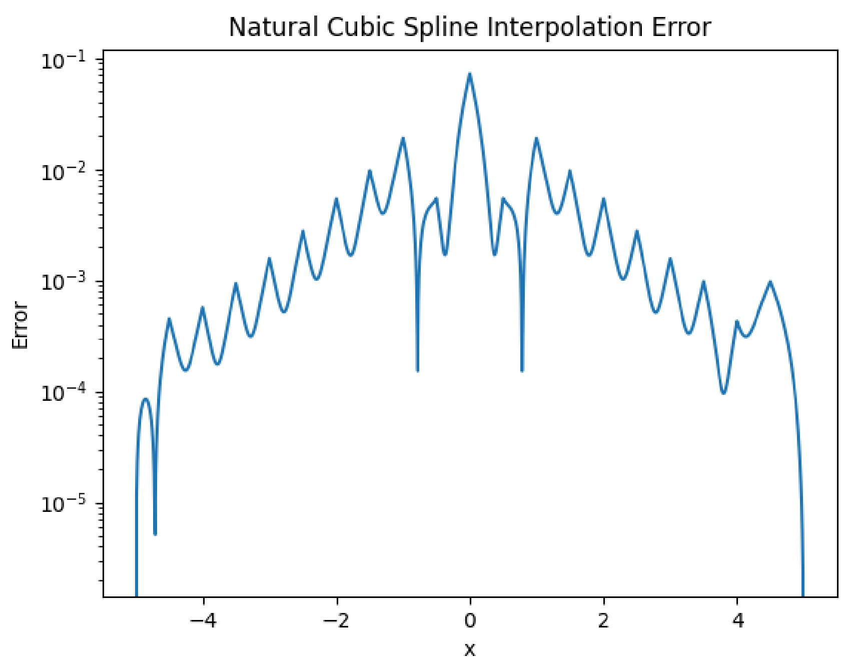


Figure 24: N = 20

Cubic spline performs not so great for small N, but even at N = 10 it starts to perform really really well. The error never reaches 1. Interestingly, if N is too large the error starts to increase.

(d): Clamped cubic spline interpolation

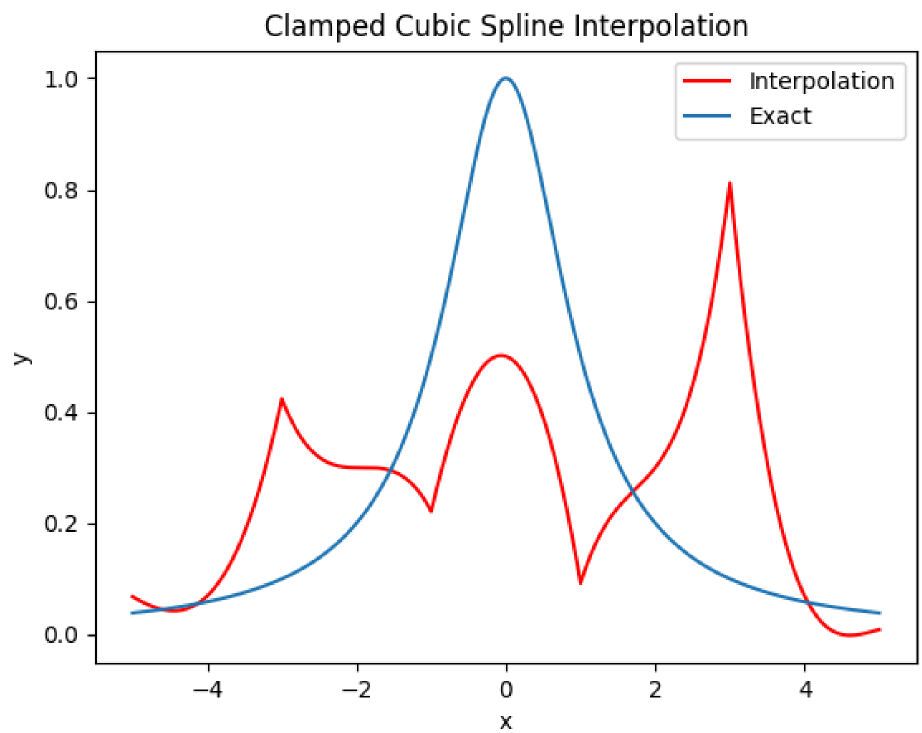


Figure 25: N = 5

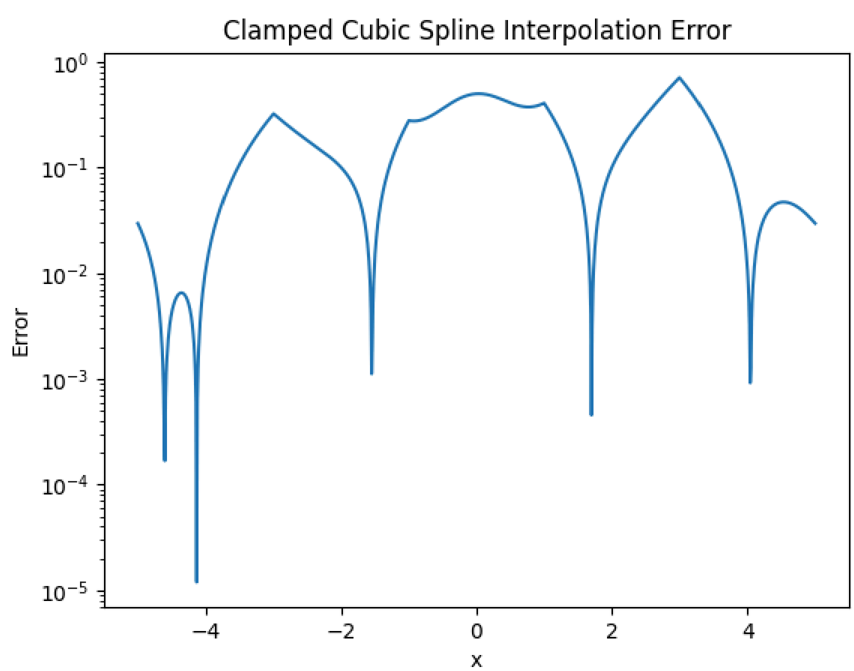


Figure 26: N = 5

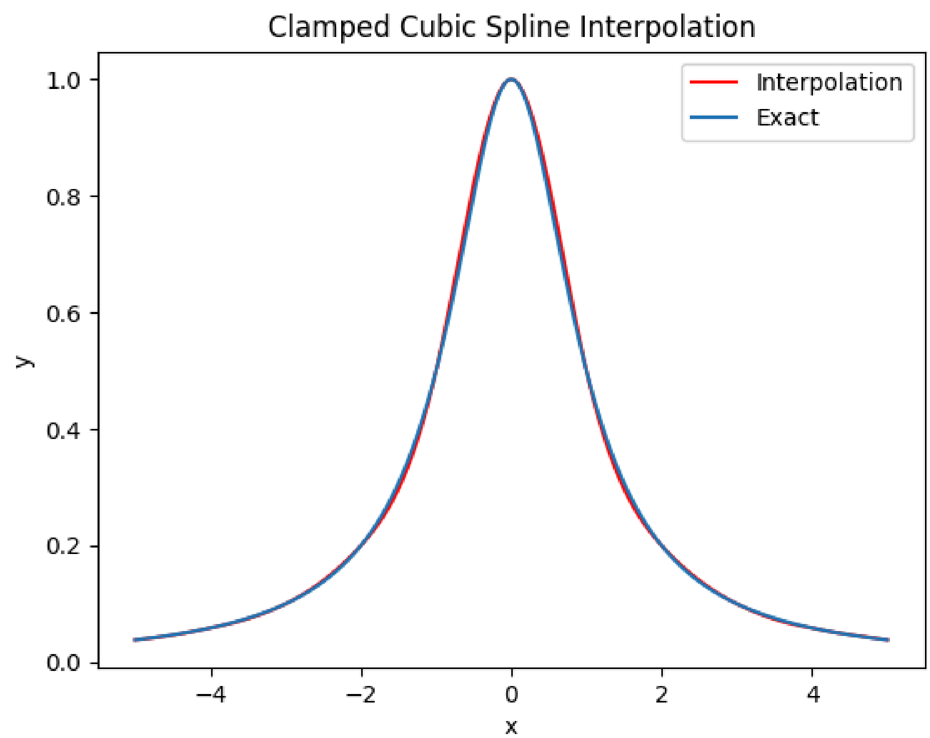


Figure 27: N = 10

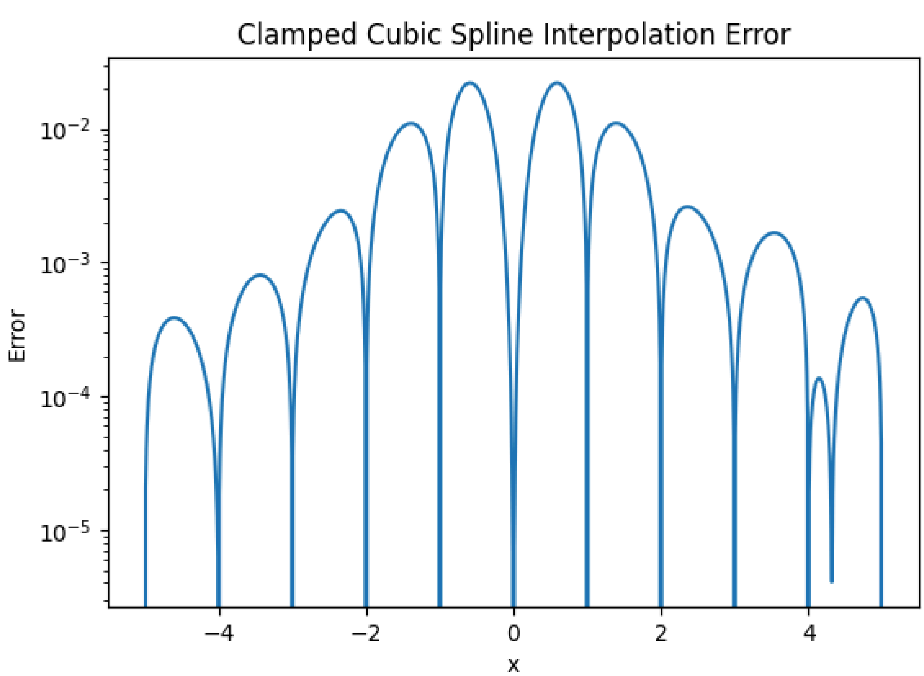


Figure 28: N = 10

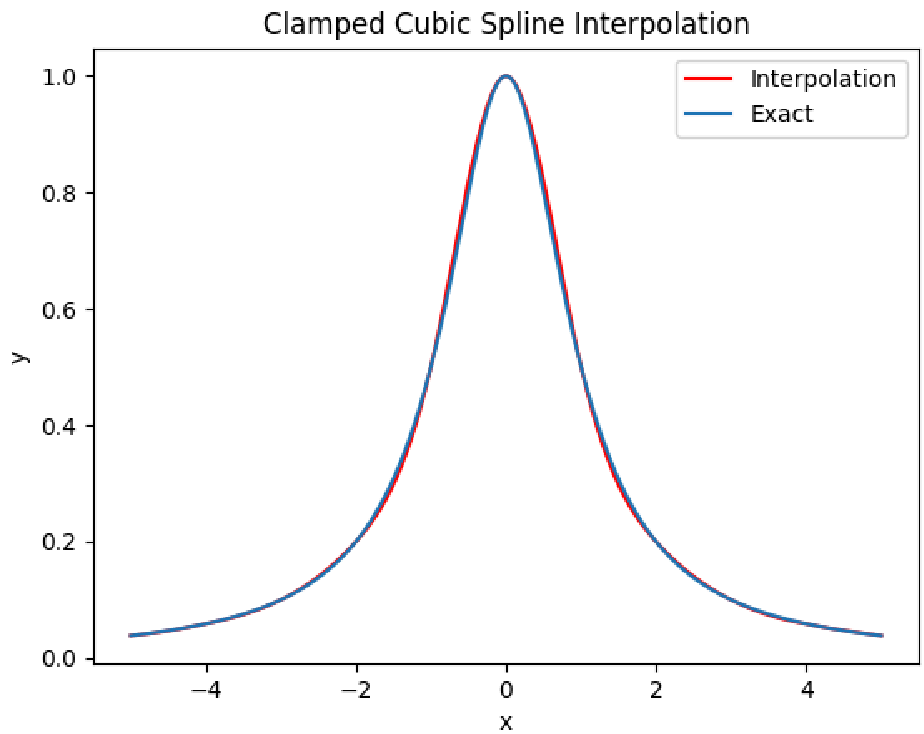


Figure 29: N = 15

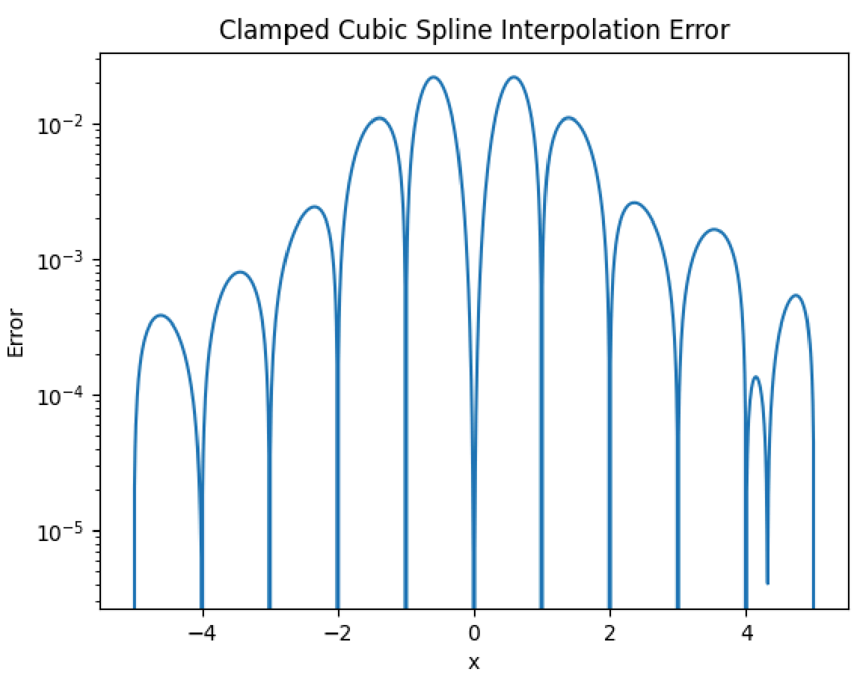


Figure 30: N = 15

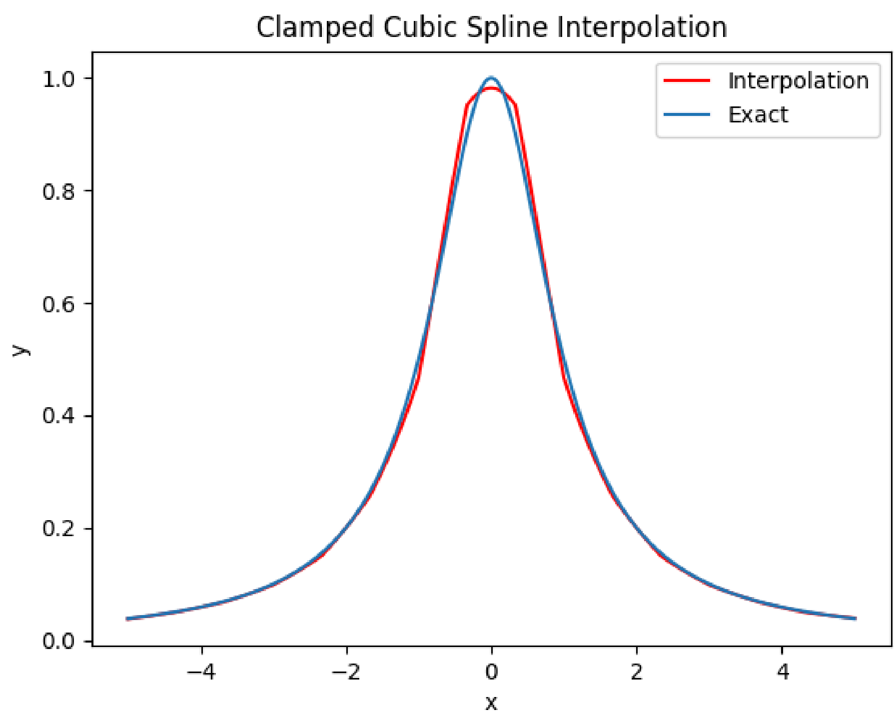


Figure 31: N = 20

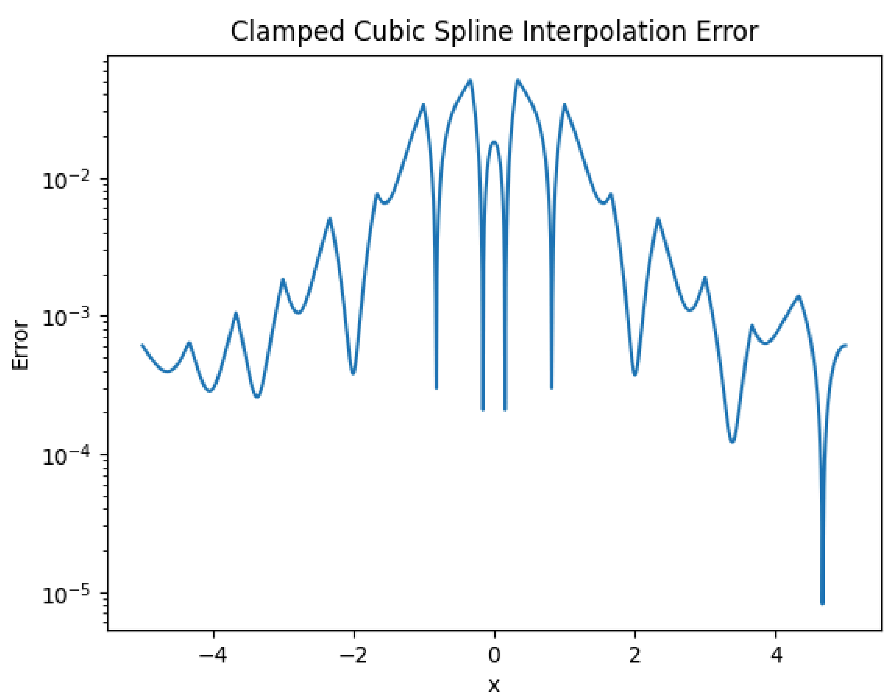


Figure 32: N = 20

Clamped cubic spline seems to perform even better than natural cubic spline. The error isn't quite as large at N = 20 as with natural cubic spline, and the graph looks qualitatively similar to the original function.

Overall, I would say the clamped cubic spline interpolation method performs the best with these chosen nodes. My intuition is that spline interpolation doesn't seem to have Runge's phenomenon with evenly spaced nodes. Also, splines are piecewise continuous; it isn't one polynomial like in Lagrange or Hermite.

2: Chebychev nodes

(a):

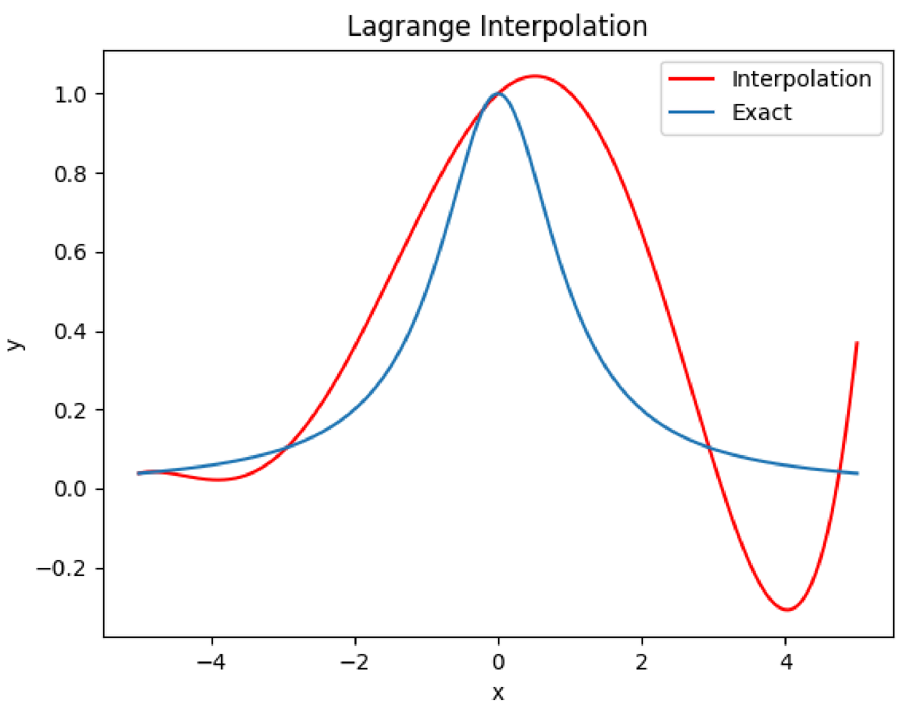


Figure 33: N = 5

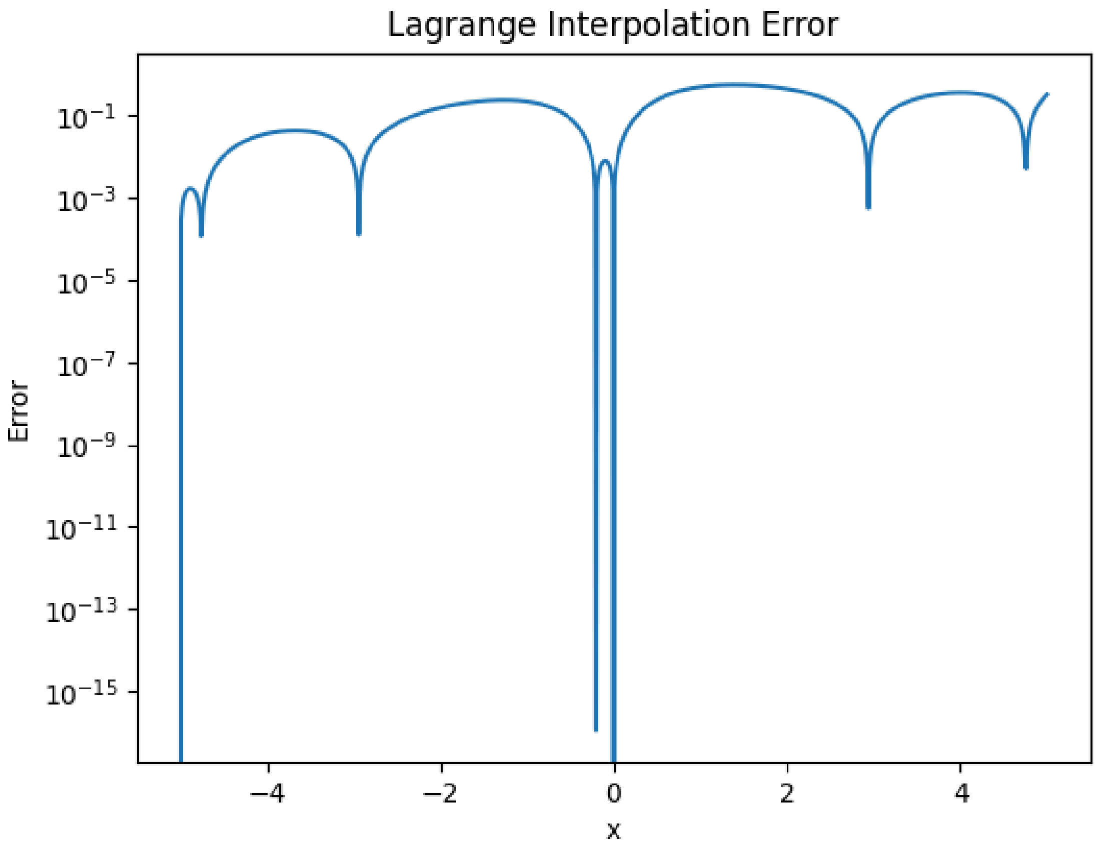


Figure 34: N = 5

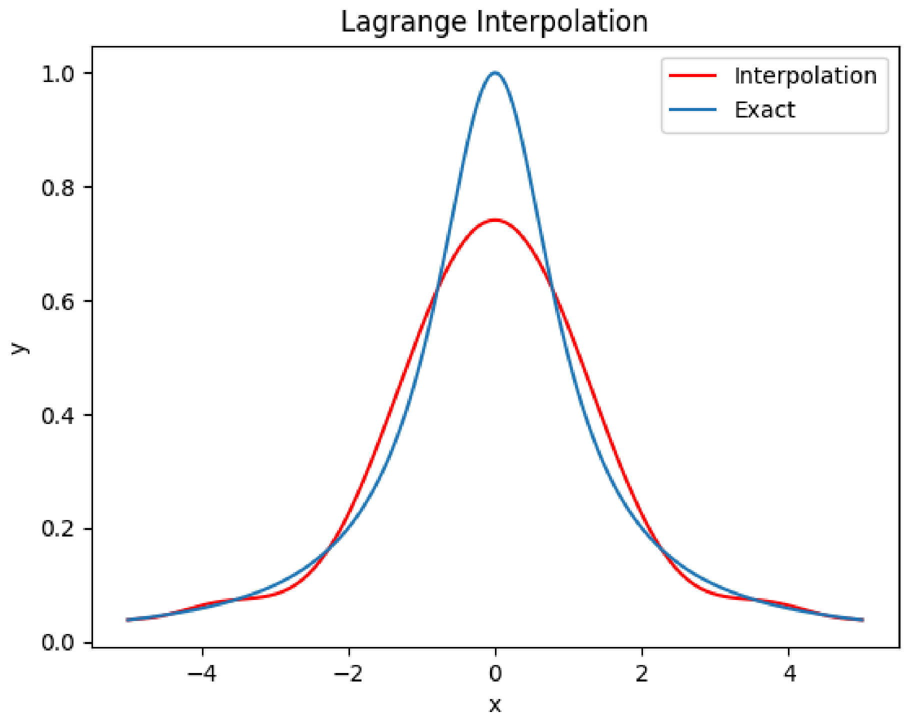


Figure 35: N = 10

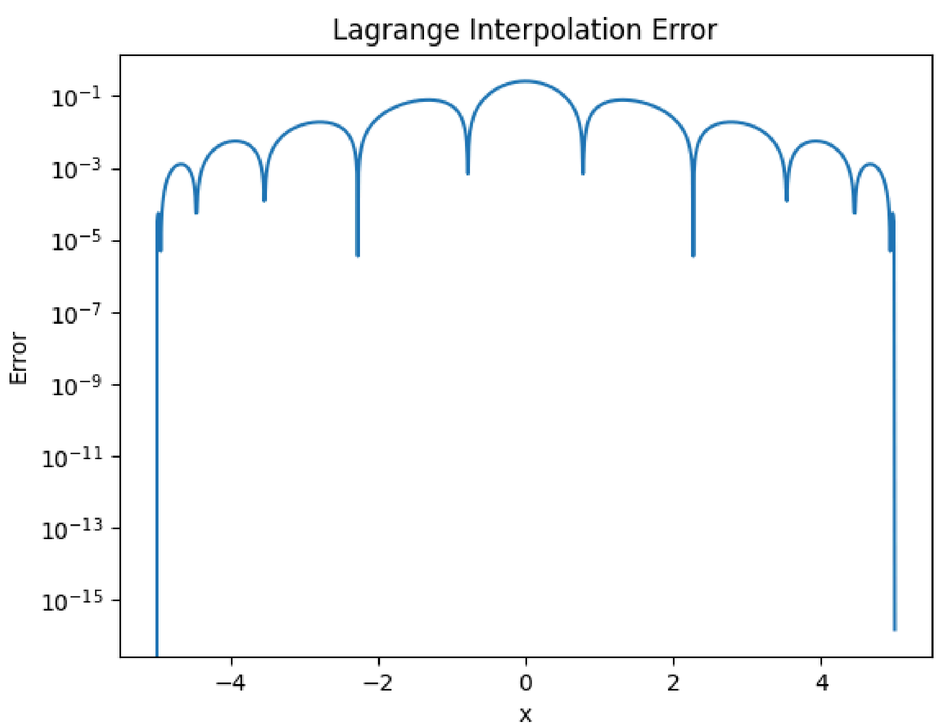


Figure 36: N = 10

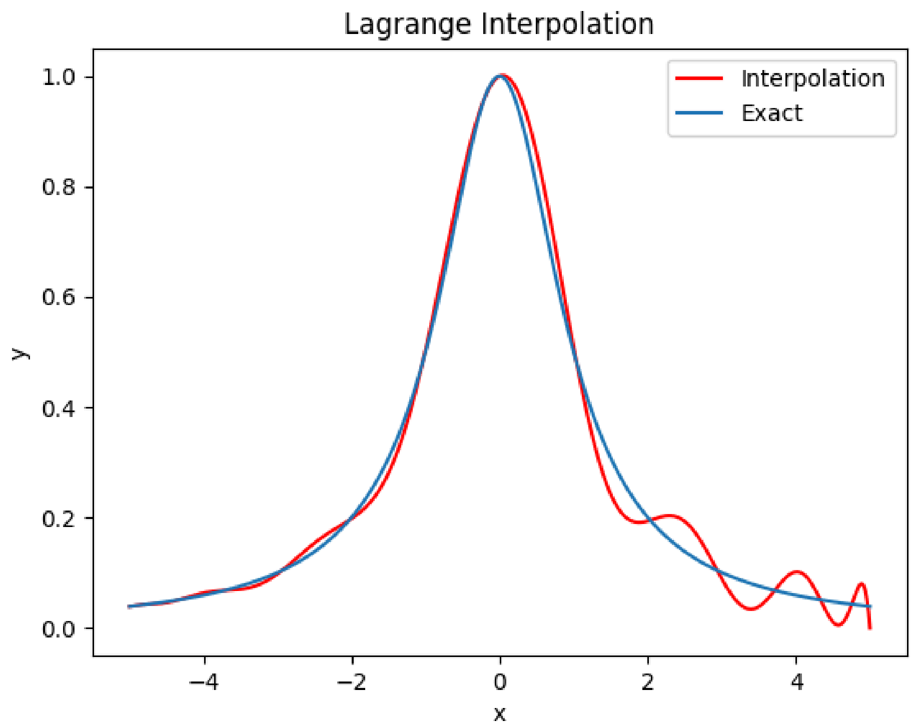


Figure 37: N = 15

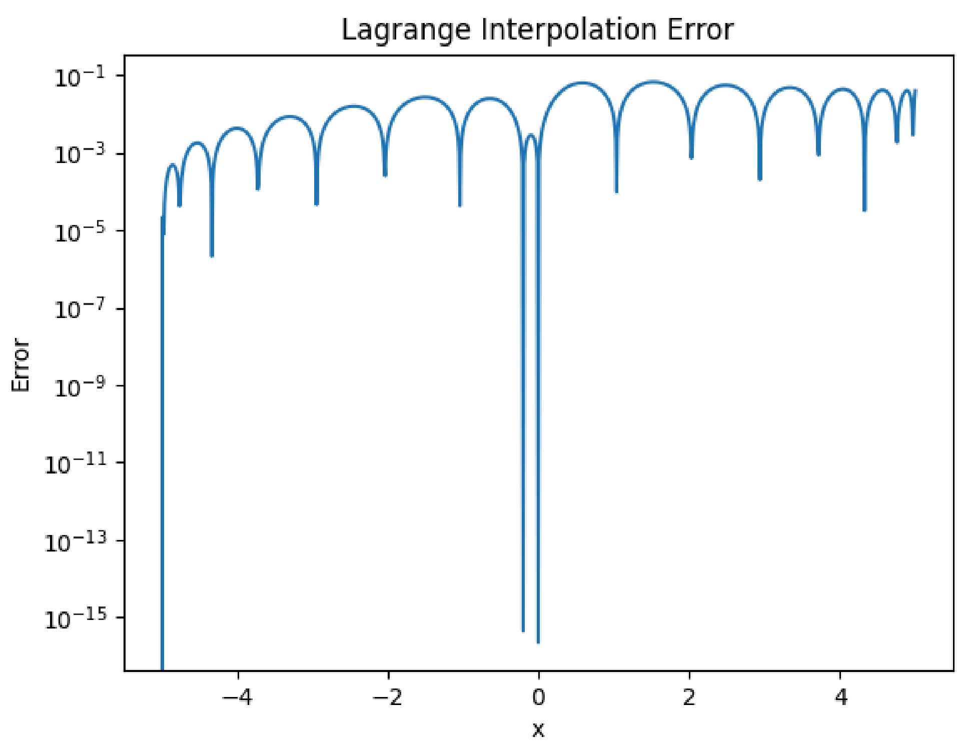


Figure 38: N = 15

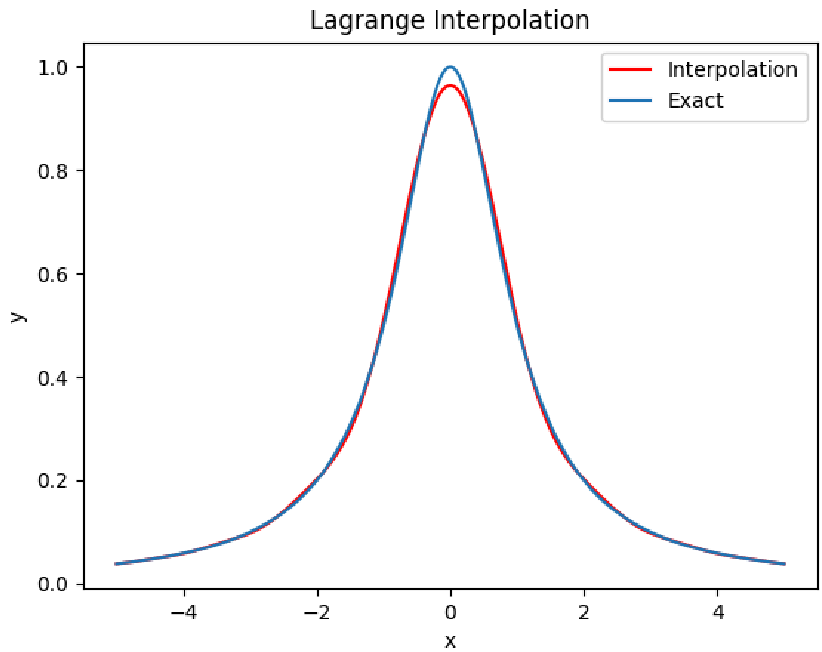


Figure 39: N = 20

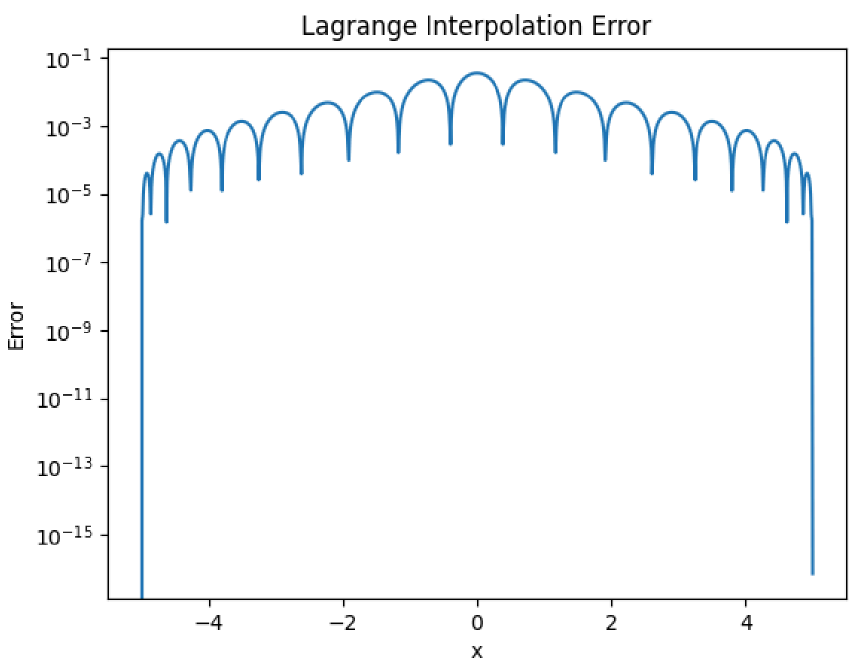


Figure 40: N = 20

Lagrange performs quite well starting at N = 10. It performs very well at N = 20. There is no Runge phenomenon with Chebychev nodes here. However, there is some error due to unwanted periodicity near the right endpoint at N = 15; this resolves at N = 20.

(b): Hermite interpolation

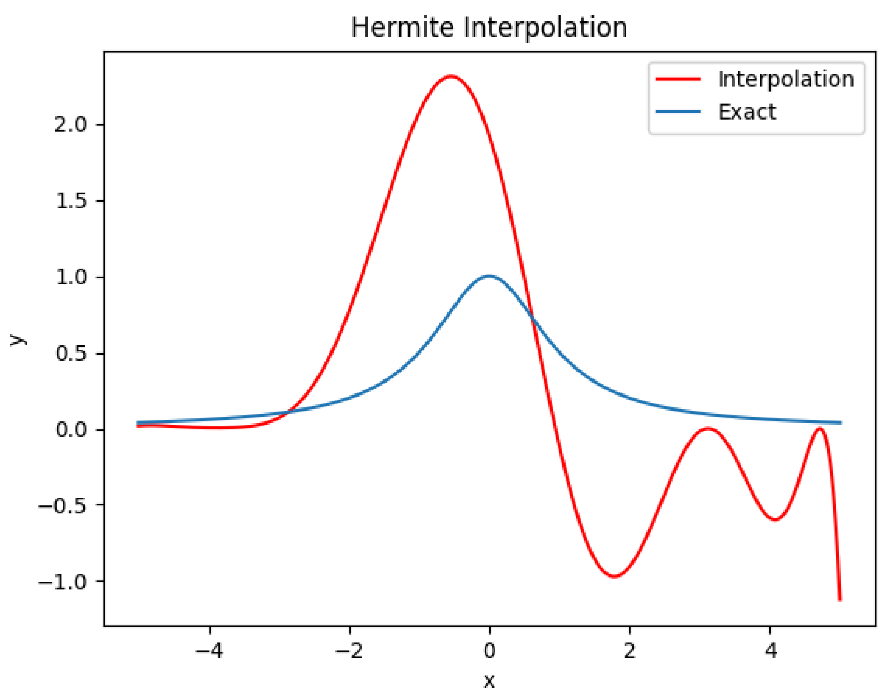
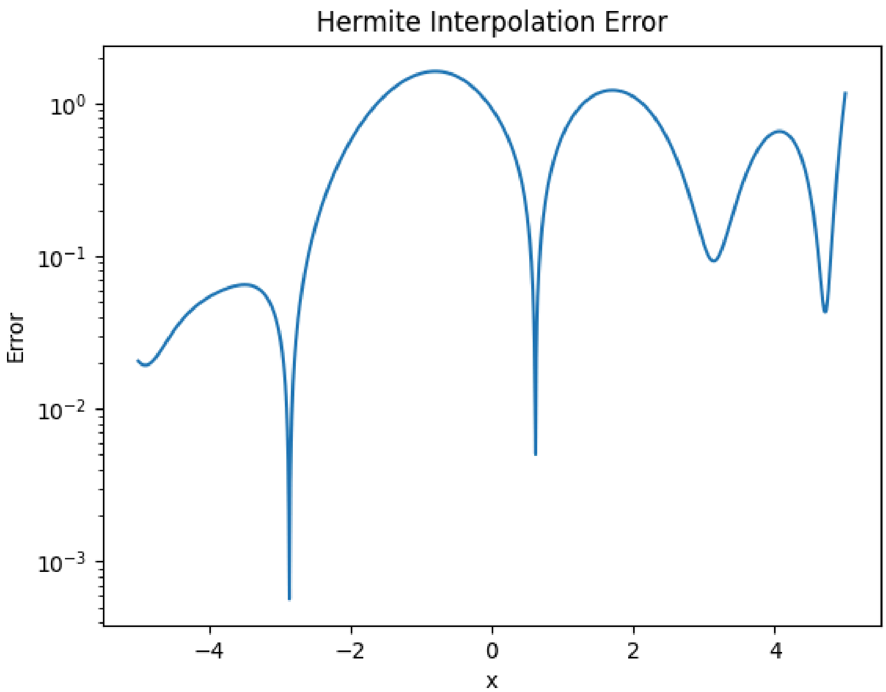
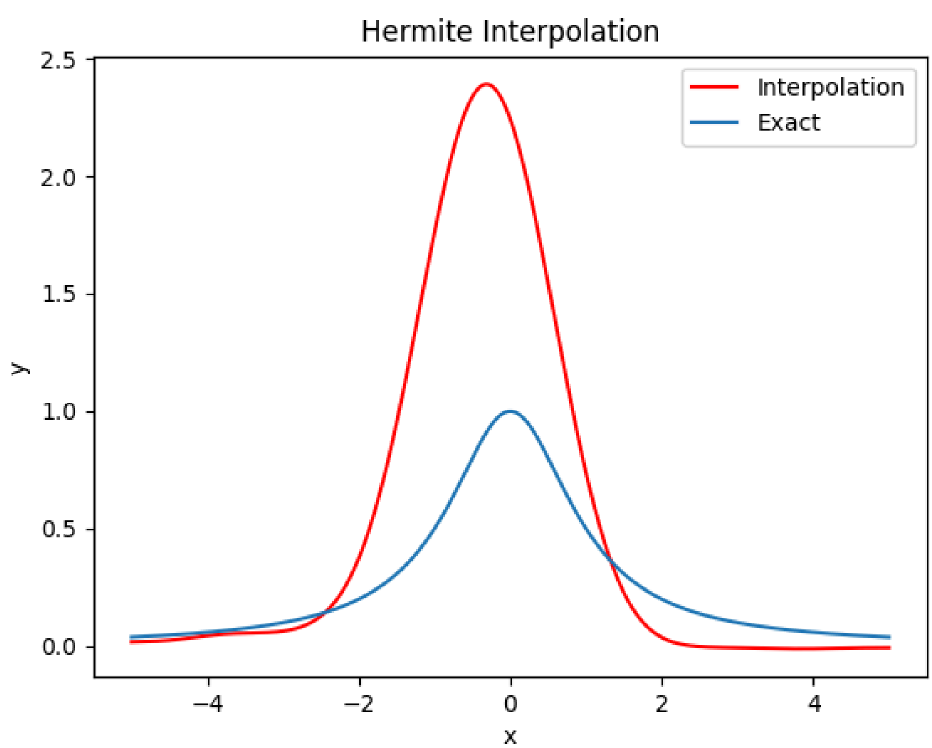


Figure 41: N = 5



Hermite Interpolation Error
Error
x

Figure 42: N = 5



Hermite Interpolation
Interpolation
Exact
y
x

Figure 43: N = 10

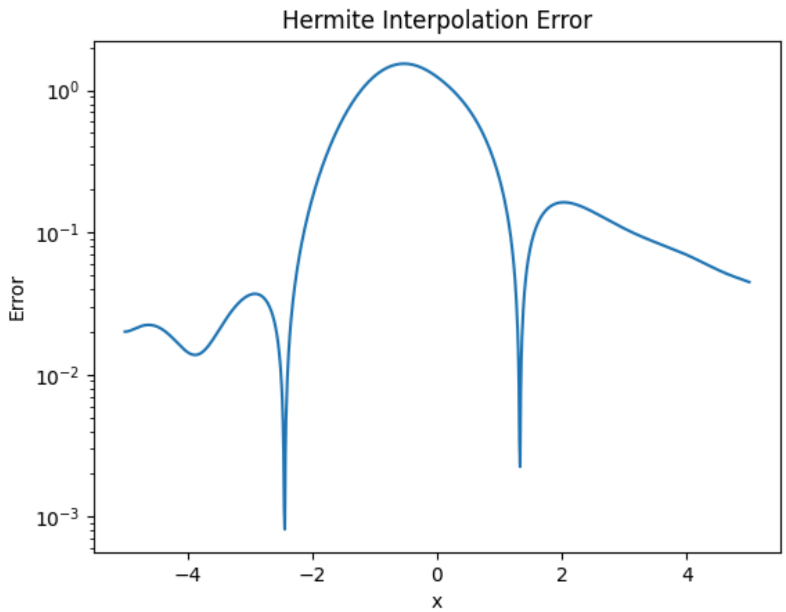


Figure 44: N = 10

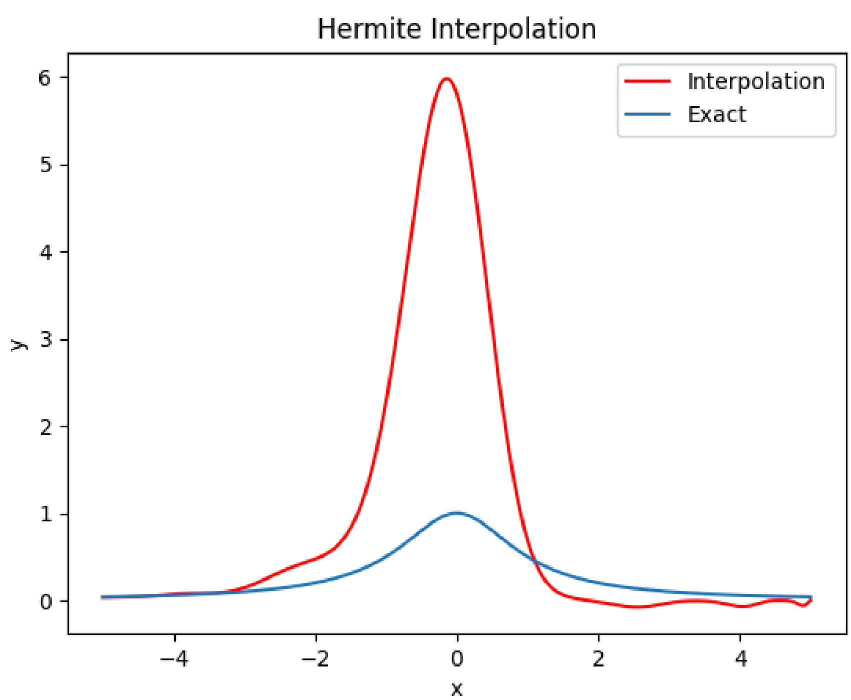


Figure 45: N = 15

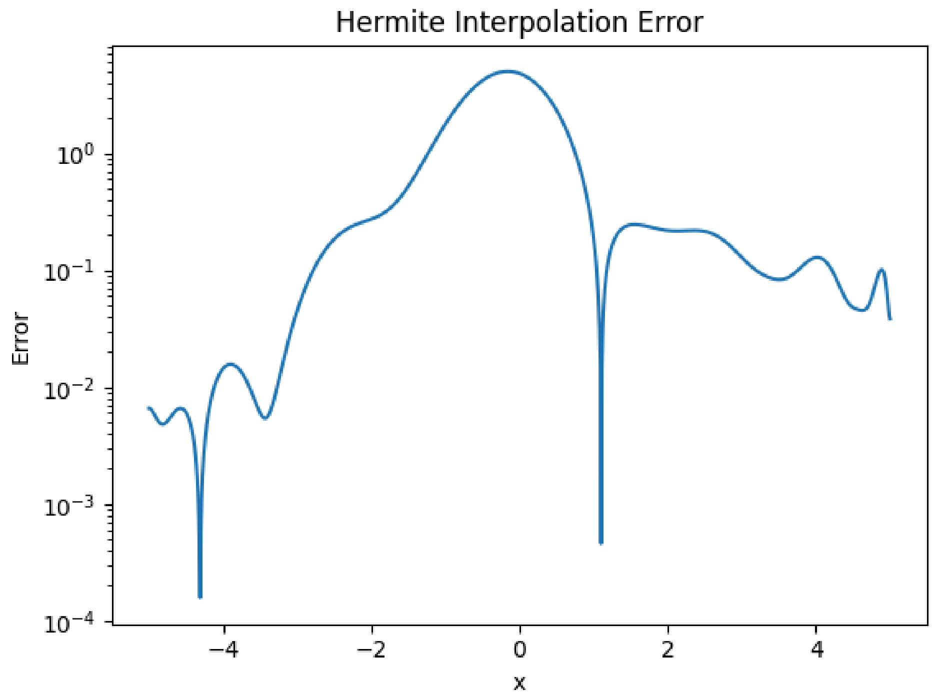


Figure 46: N = 15

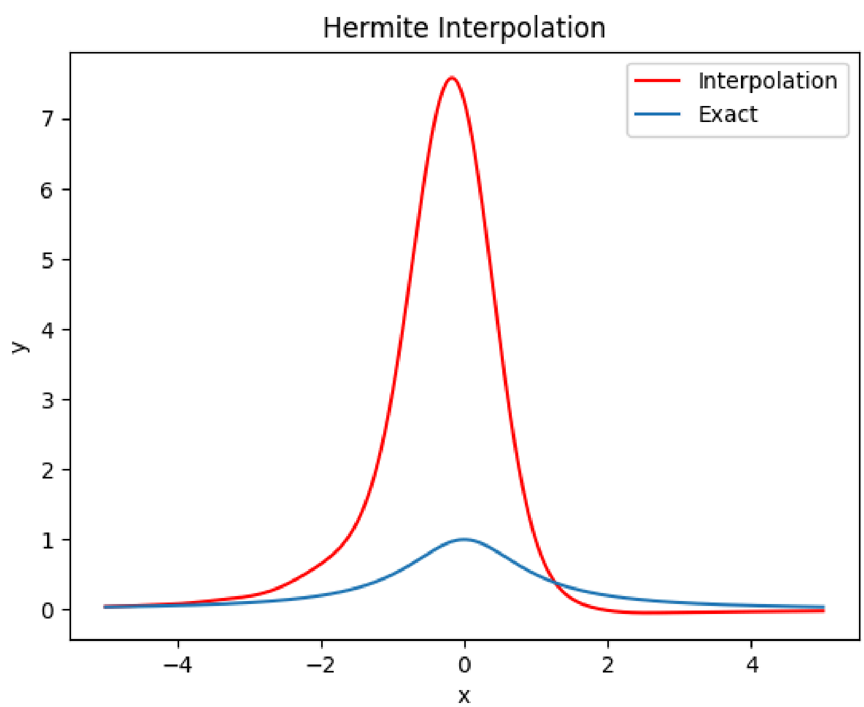


Figure 47: N = 20

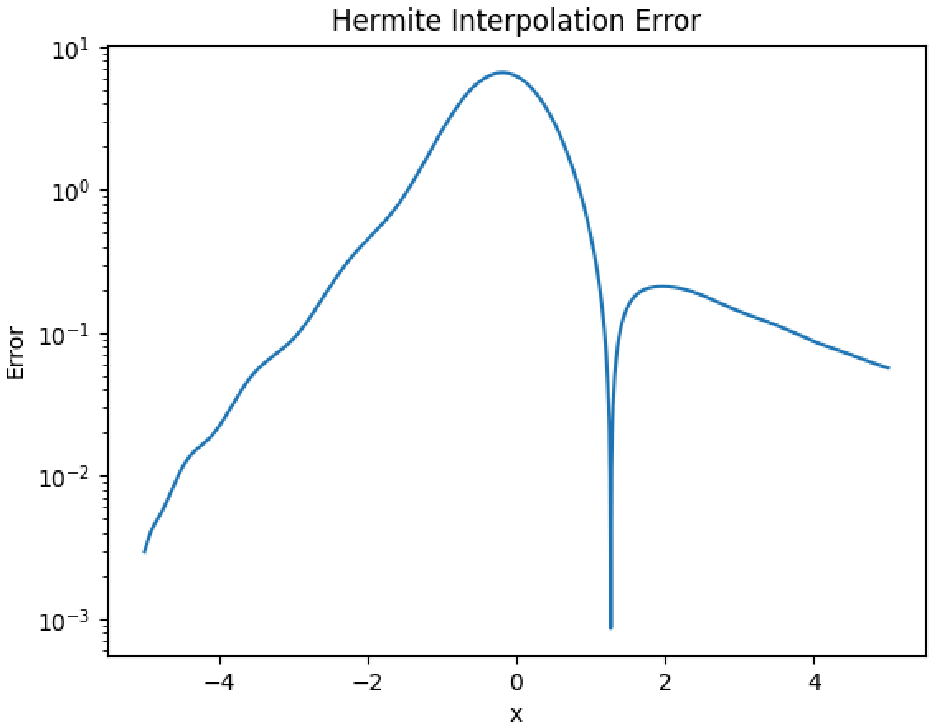


Figure 48: N = 20

Hermite performed much better with Chebychev nodes. However, it still underperforms compared to Lagrange. My guess is that the peak is much larger than the original function due to the extra derivative data held within the polynomial.

(c): Natural cubic spline interpolation

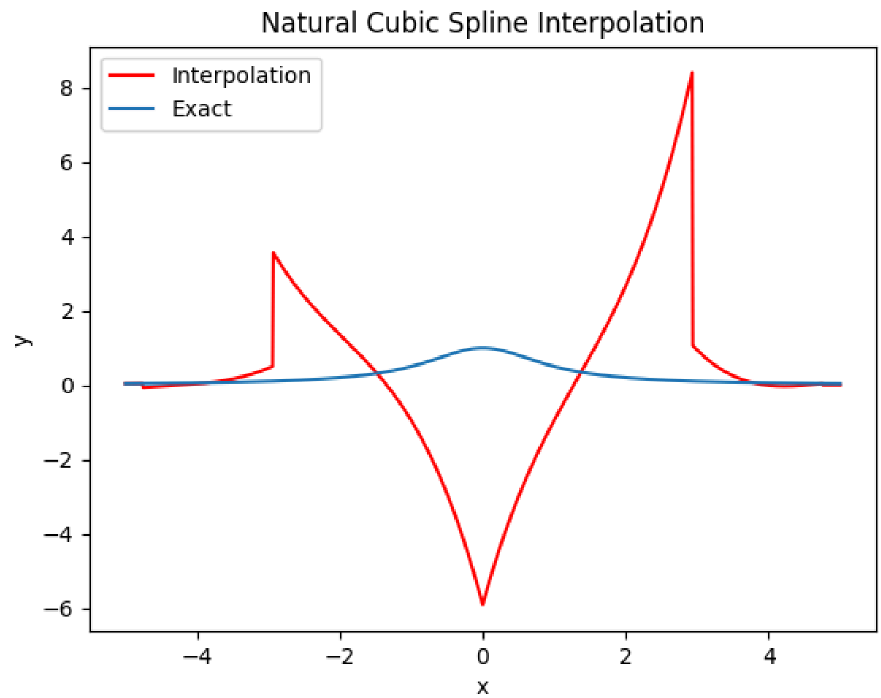


Figure 49: N = 5

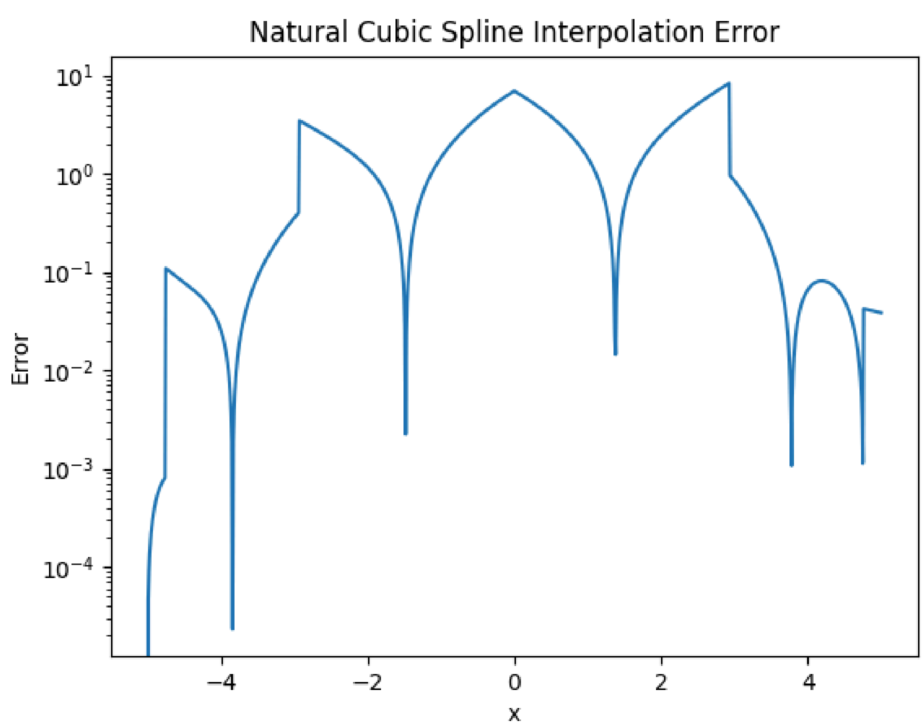


Figure 50: N = 5

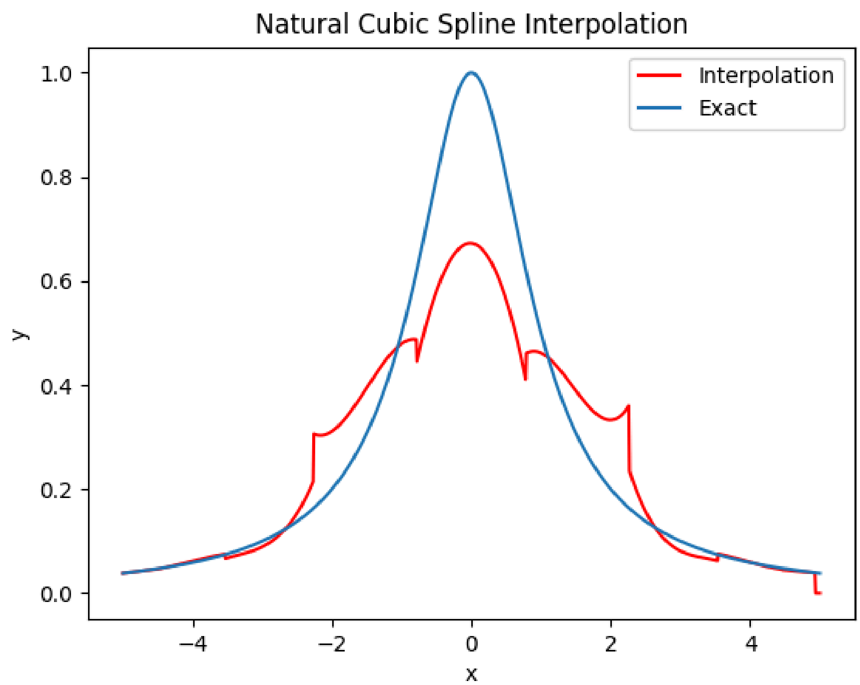


Figure 51: N = 10

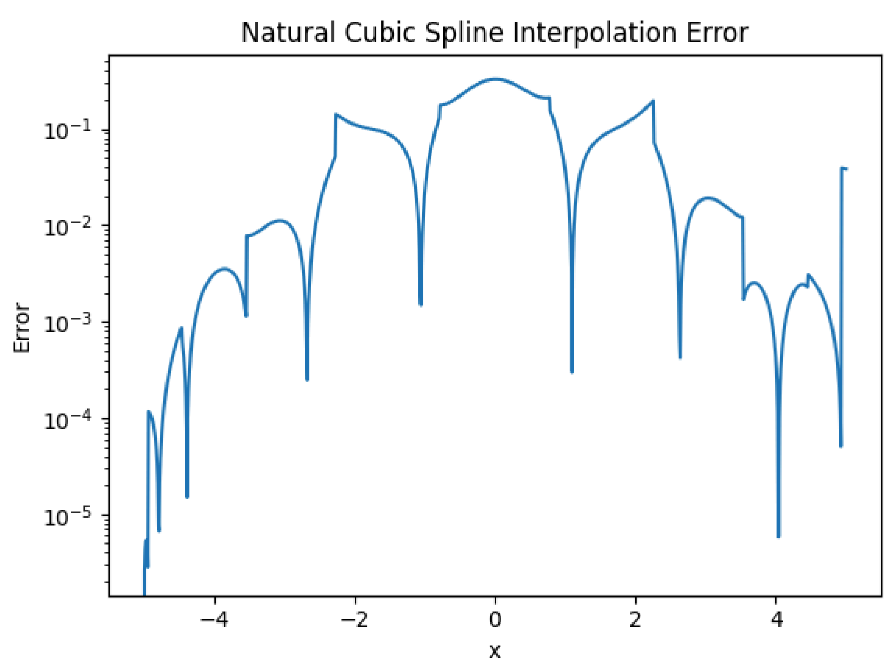


Figure 52: N = 10

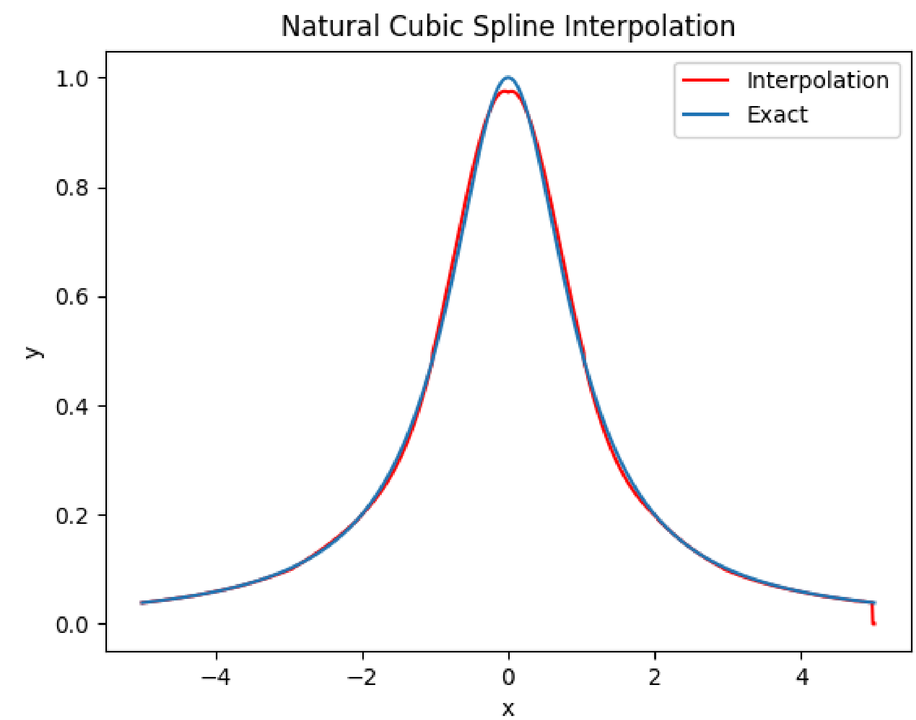


Figure 53: N = 15

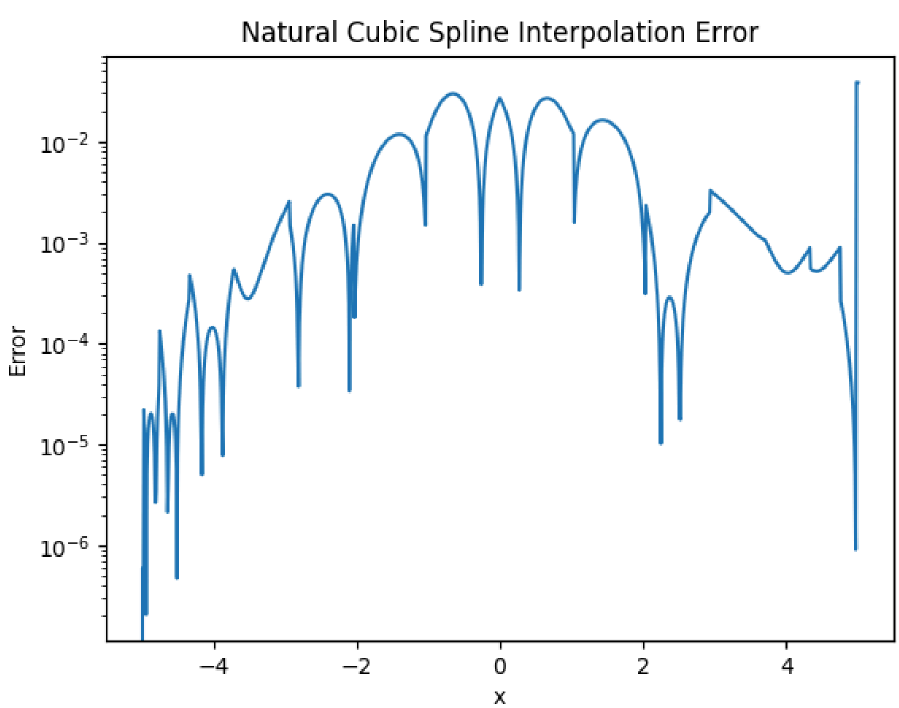


Figure 54: N = 15

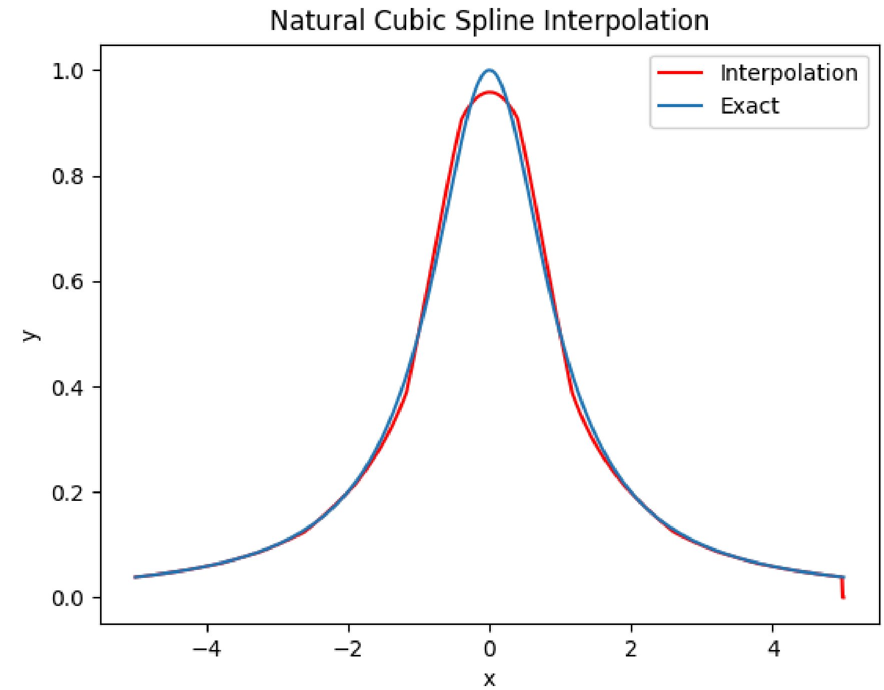


Figure 55: N = 20

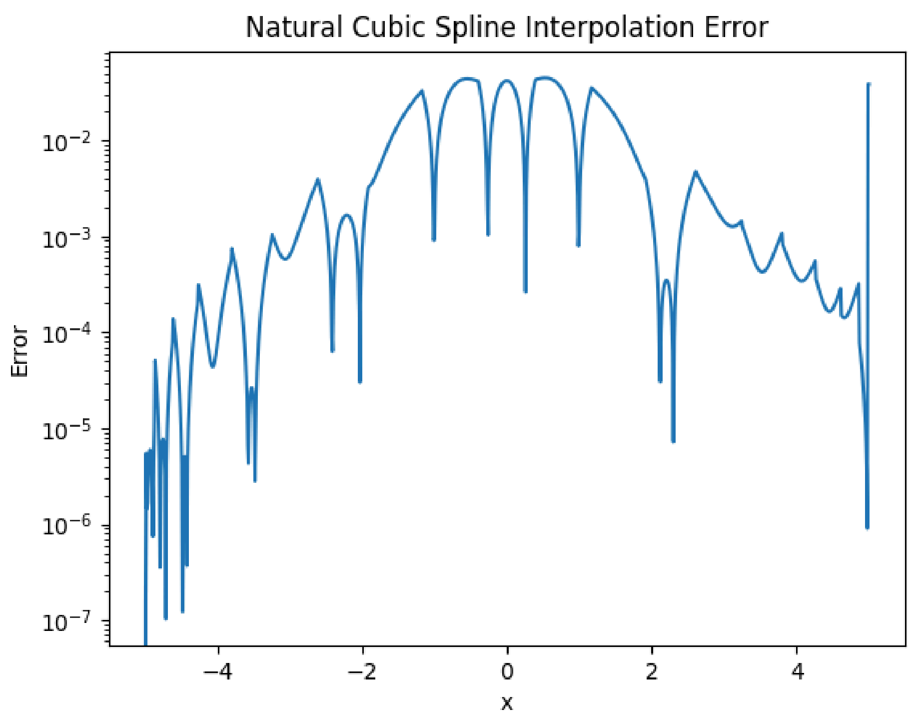


Figure 56: N = 20

For N >= 15, cubic spline seems to work better than before. It performs about as well as Lagrange at N = 15,20 regarding the error graphs. The spike at N = 20 at the apex of the original function is now gone. For N < 15, it tends to work less well than equispaced nodes.

(d): Clamped cubic spline interpolation

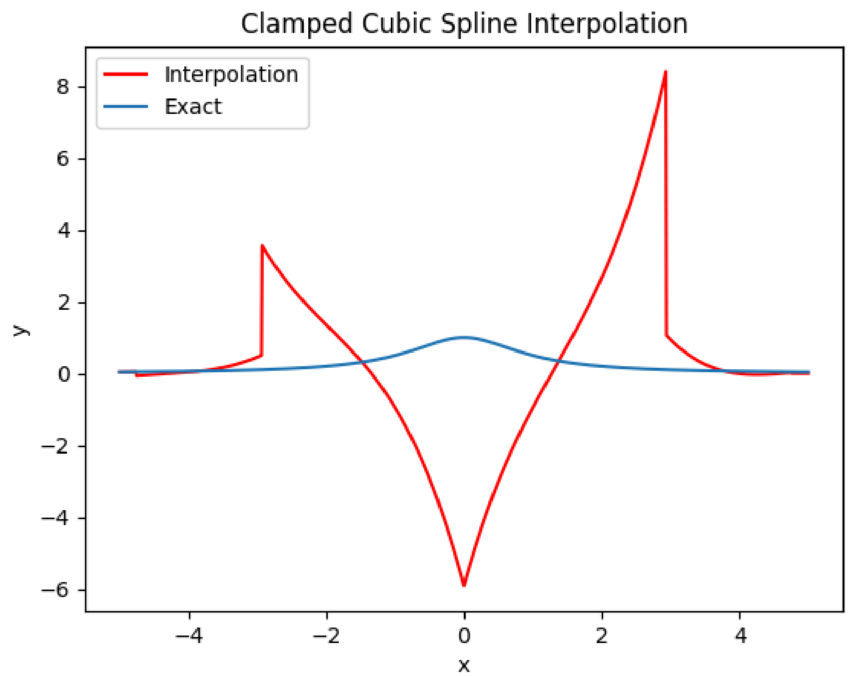


Figure 57: N = 5

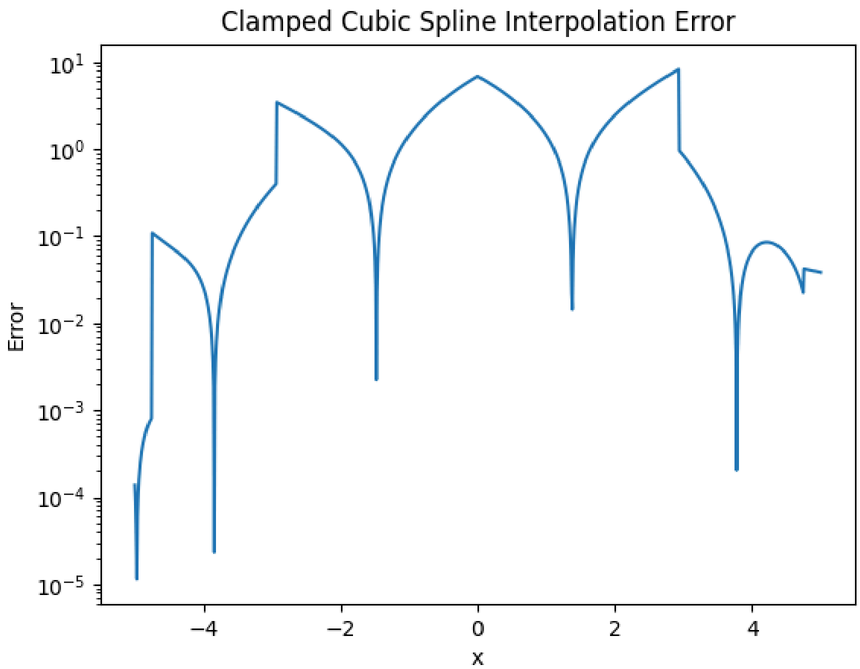


Figure 58: N = 5

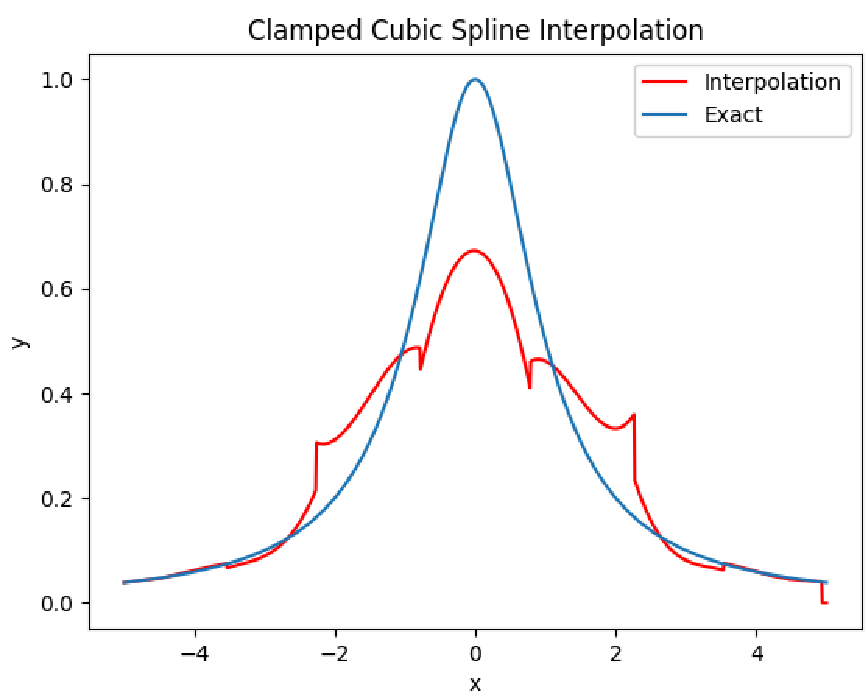


Figure 59: N = 10

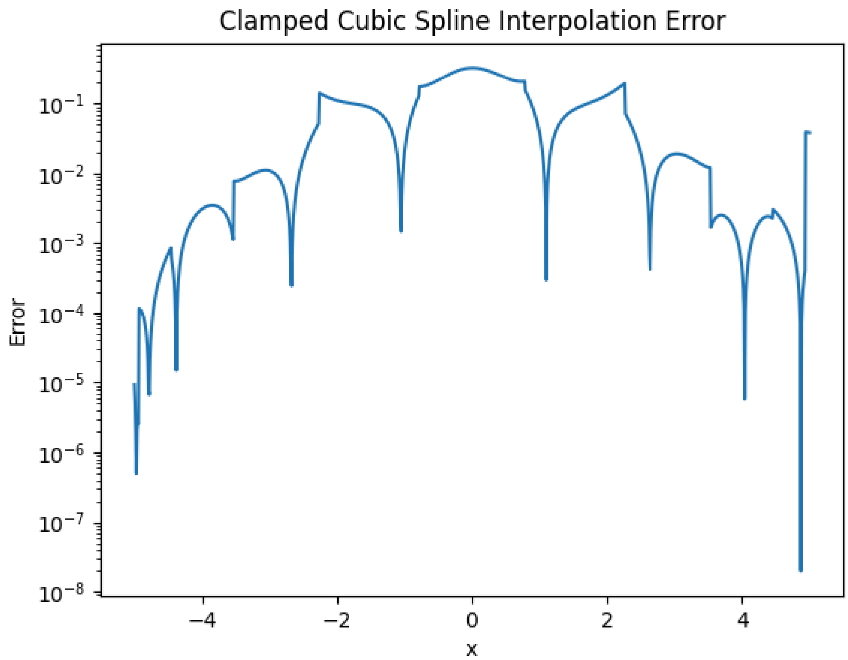


Figure 60: N = 10

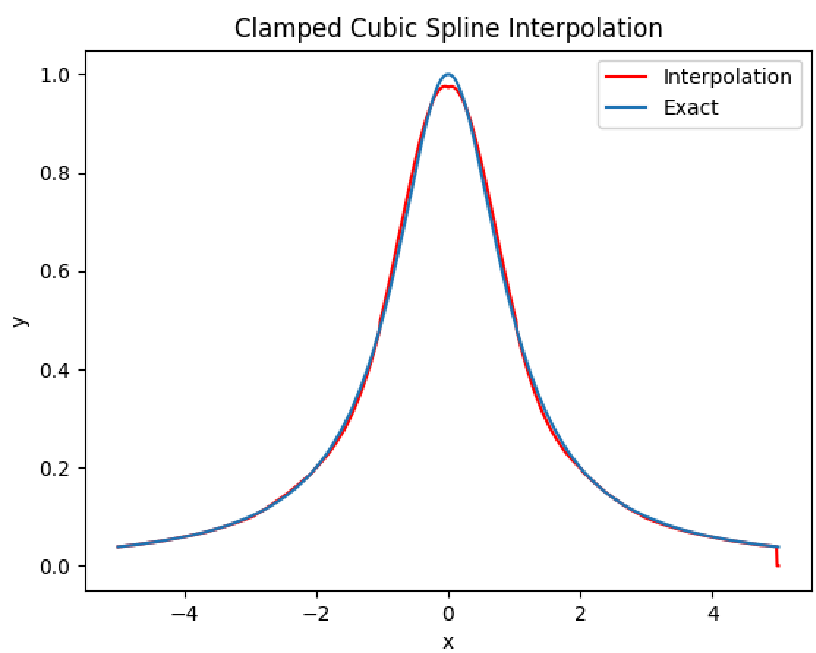


Figure 61: N = 15

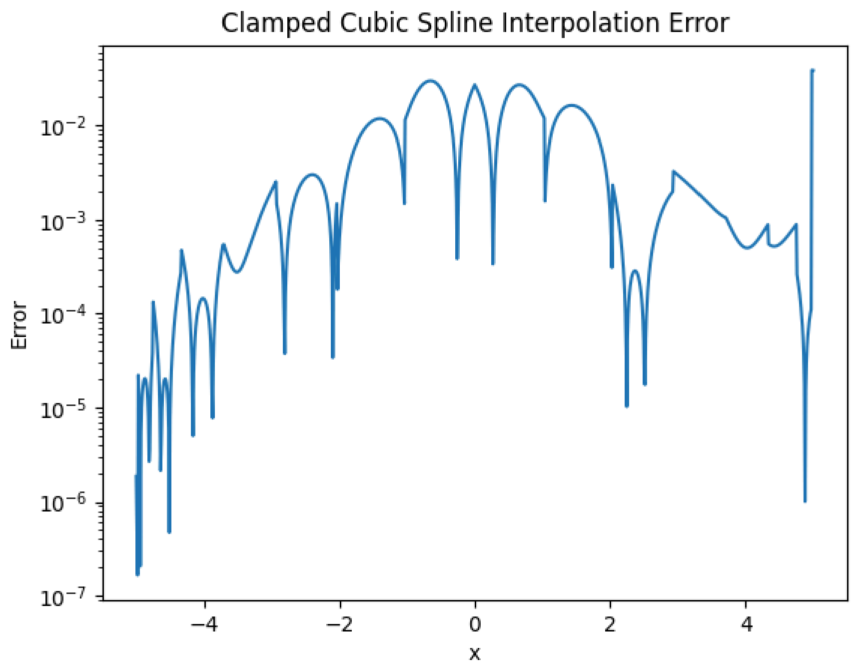


Figure 62: N = 15

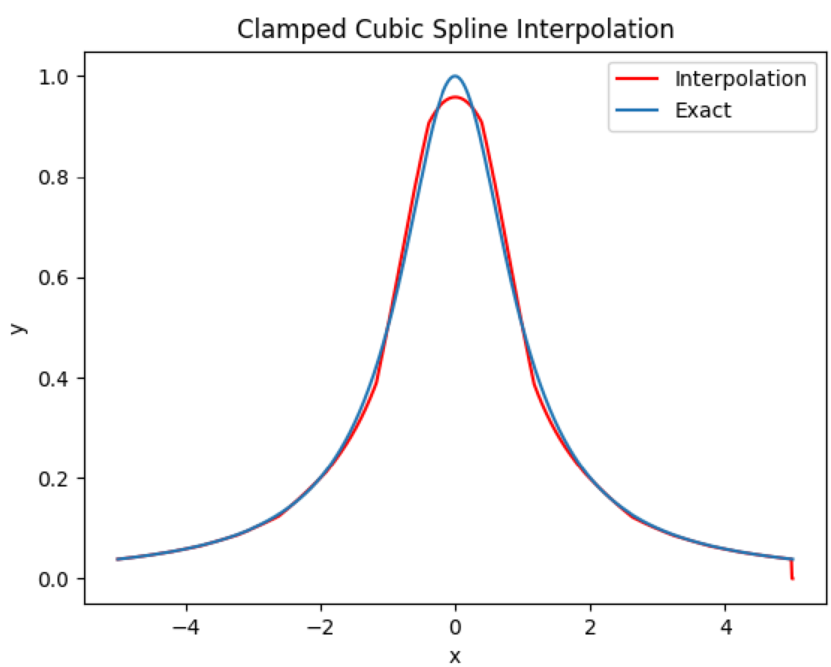


Figure 63: N = 20

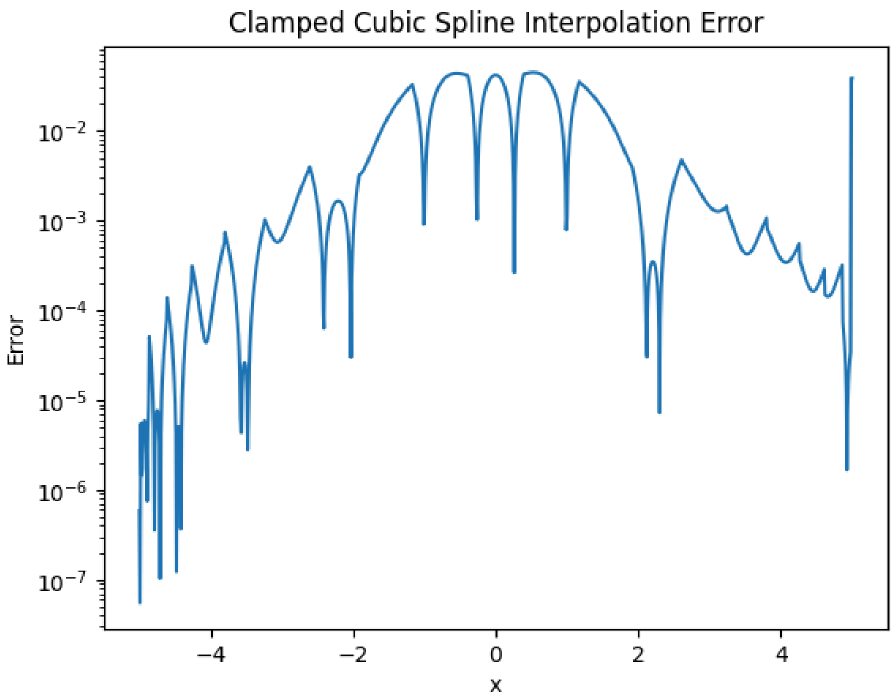


Figure 64: N = 20

Clamped cubic spline performs similarly to natural cubic spline with Chebychev nodes. It seems that any N value under 15 does not work well whereas with equispaced nodes it did just fine; it's the same with natural cubic spline.

Overall, I would say that Lagrange performed the best but one could argue that clamped cubic splines win as well – just judging from the error graphs qualitatively. Hermite performed the worst.

3:

For the interpolant to be periodic, the following conditions must hold:

1. y(0) = y(2pi)
2. y'(0) = y'(2pi)
3. y''(0) = y''(2pi)

This would mean that M[0] = M[N]. But for the first derivatives, you need to set the endpoint derivatives equal to each other to get a new equation. It turns out that all you need to do is modify A slightly by setting A[0,N] = h[N] and A[N,0] = h[N]. This links the first and last rows so that the first derivatives are equal at the endpoints which creates a cyclic nature to the splines.

I wasn't able to get an implementation of periodic splines working using this theory, but I'm confident that I understand how the changes to A come about. I really couldn't figure out the implementation though. My attempt is below:

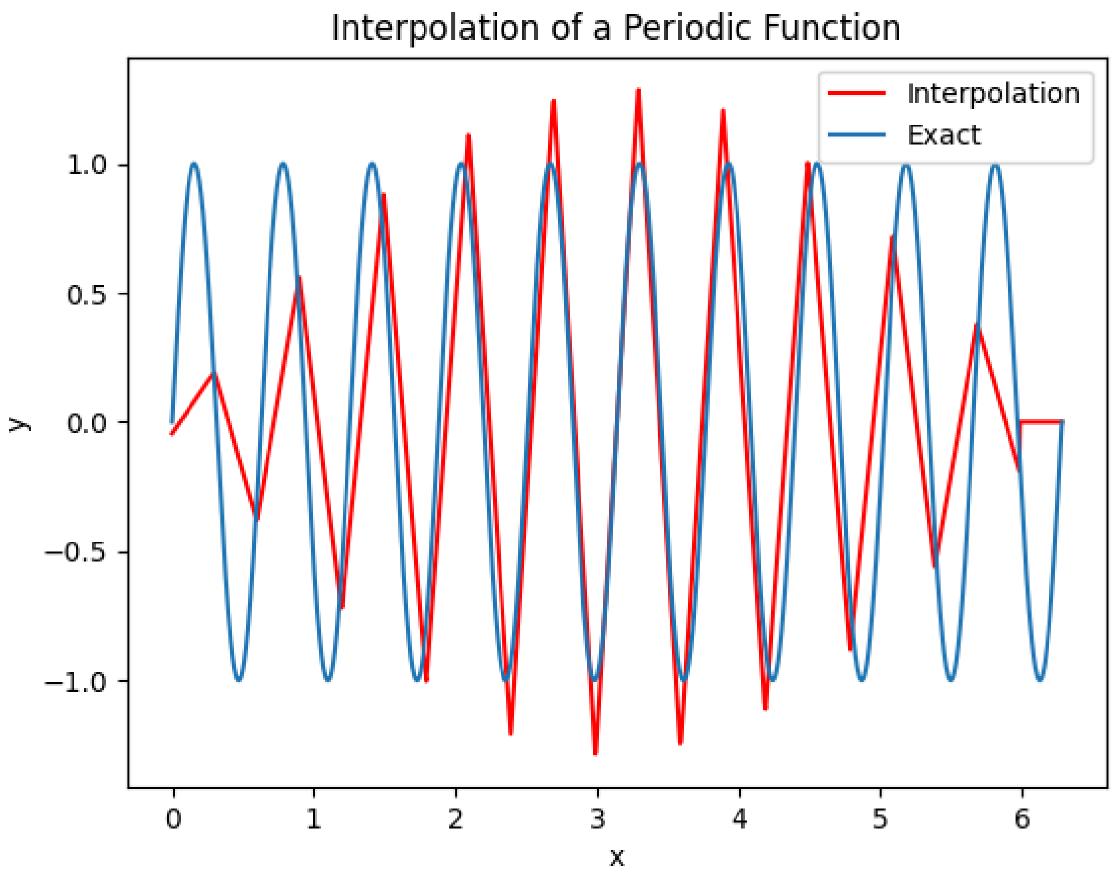


Figure 65: Attempt at periodic cubic splines. It didn't work.