Note: Where shortcuts existed, I opted to use them in my function. However, here are some snapshots of the by-hand process. I had my function output the vector of polynomial coefficients, p, so that I could try these out this way.

Linear calc by eyeballing:

(4021.24, 1.21003) to (32132.2, -143.895)

-(-143.895 - 1.21003) / (32132.2 - 4021.24) = 0.0052

5.2ms



A screenshot of a computer

Description automatically generated etc.



A screenshot of a table

Description automatically generated etc.

A graph of a graph

Description automatically generated with medium confidenceA graph of a function

Description automatically generatedA graph of a group delay

Description automatically generated

4. To start, all three methods gave answers that we can largely consider to be “reasonable”. Linear estimation gave of an estimate of 5.2ms, while our polynomial fit ranged from 12.6ms to just barely below 0, and while first order difference had a lot of variance, most of the group delays it yielded fell into that 2-15ms range. In terms of closeness to one another, the linear estimate fell roughly in the middle of the other 2 methods’ ranges. First order difference appeared to approximately vary about the estimates obtained from the polynomial fit, i.e., they captured the same overall trend in changing group delay across frequency. The estimates obtained from each method could never be mistaken for one another, but they do tell related stories. In terms of accuracy, the polynomial appeared to provide the best estimates. It only started to provide nonsensical group delay right at the edge of the high frequencies, whereas first order difference repeatedly entered into negative values. Unlike linear modeling, our cubic model also captured (somewhat accurately) the change in group delay across frequencies.

The main issue with the first-order difference as we used it here is that it has a very high variance. This means that even if it captures the overall shape of the group delay, we 1. can’t trust the estimate that it gives at any individual point and 2. have to handle a large number of nonsensical random values. With that in mind, we want to apply some type of method to smooth out the estimate it provides. One thought I have is to collapse it into a series of averages. If we were to take a window of, as an example, the first 5 group delay values, average those together, then slide the window forward so that we then average across values 2-6, and so on, we would end up with a much smoother estimate that should still capture the change across frequencies. A sliding window does seem like it might create convolutions further down the line, so we could also consider just windowing it in chunks, and having each of the frequencies within that chunk have the “same” group delay.

Another method that may work better would be to try fitting something other than a polynomial. My gut instinct is that the unwrapped phase resembles a -log function curve, both in terms of appearance and general behavior. Logarithmic scales are very commonly appropriate when we deal with the cochlea, so I suspect that that may be the case here as well.