Tags: #logbook - Denison
Links:

Logbook_08_220207

A Numeric Project

> **Aims**
>
> - [ ] Repeat analysis except first letting the soliton stabilise

A.1 Notes

If we are applying the frequency shift before giving the soliton a chance to stabilise, we may be unfairly forcing it into a disfavourable configuration. Instead, a more realistic simulation would be to apply the shift to a stable soliton. We can do this by setting the initial E_0 to be the stable electric field.

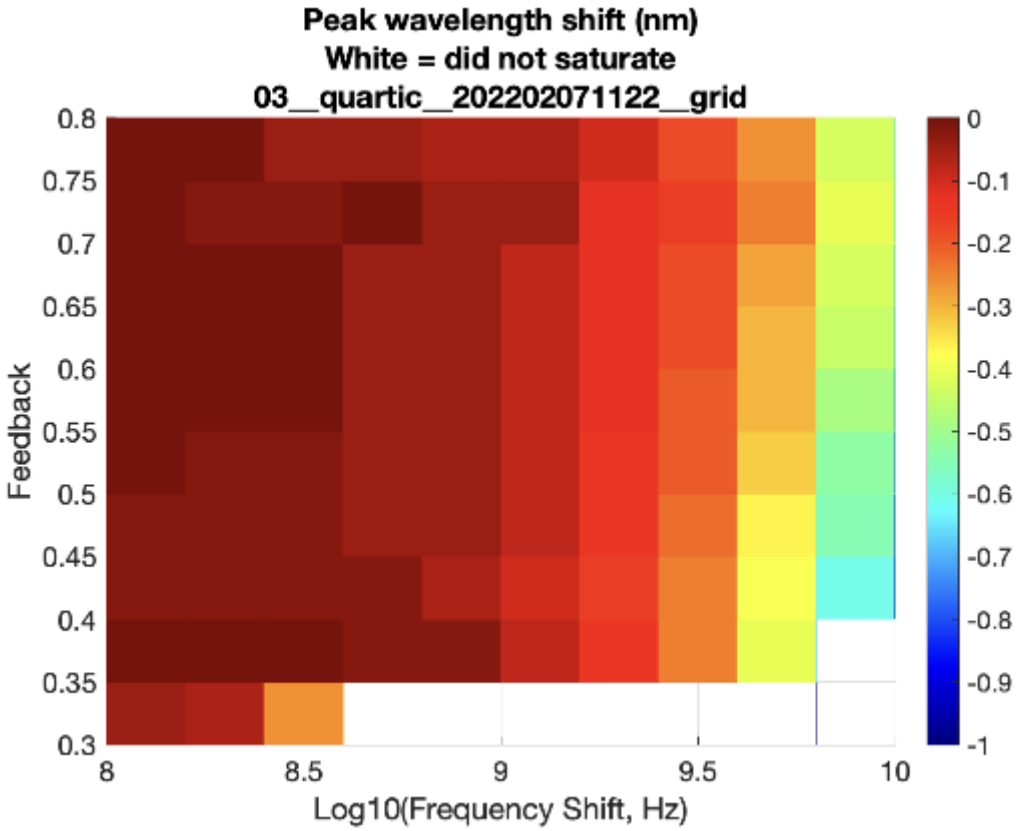
A.2 Results

A.2.1 Quartic Dispersion

Using `quartic__220206__zero.m` to calculate and save the E_0 values for each `feedback` level, without any frequency shift. Then use those as an initial profile when running `quartic__220206__grid.m` to repeat the analysis in Logbook 07.

Data: `03__quartic__202202062259__zero.mat` and `03__quartic__202202062351__grid.mat`

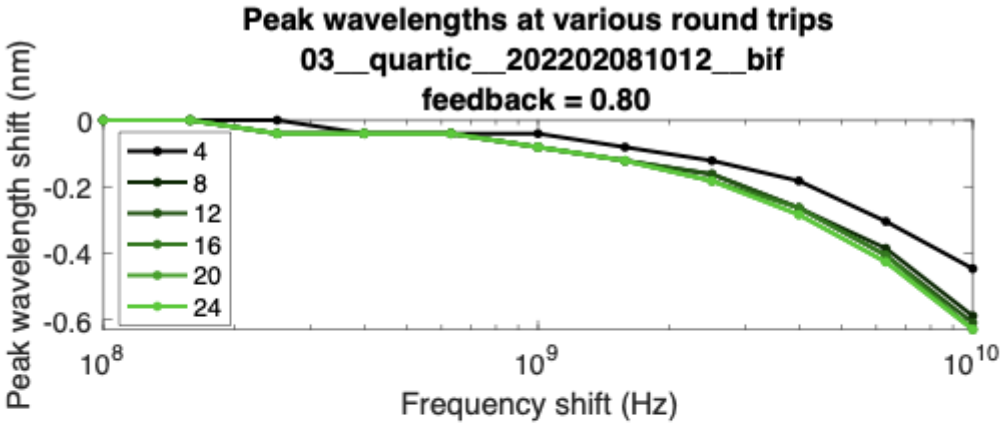
> `03__quartic__20220207122__grid.fig`

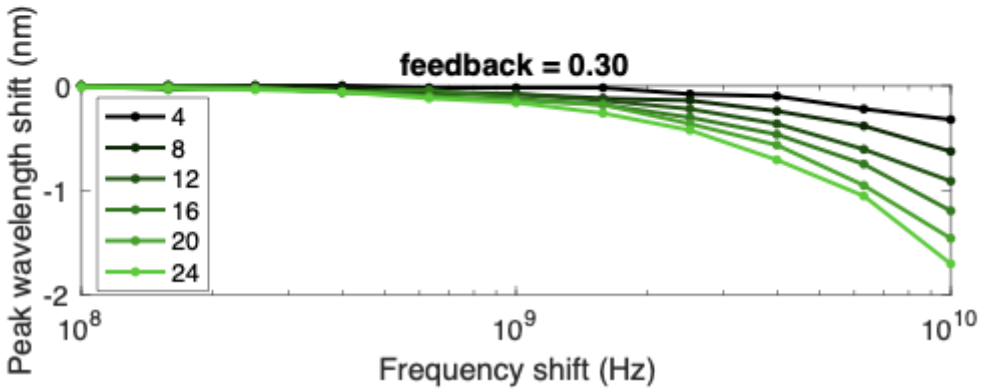


A.2.1.1 (Incorrect) Bifurcation Plots

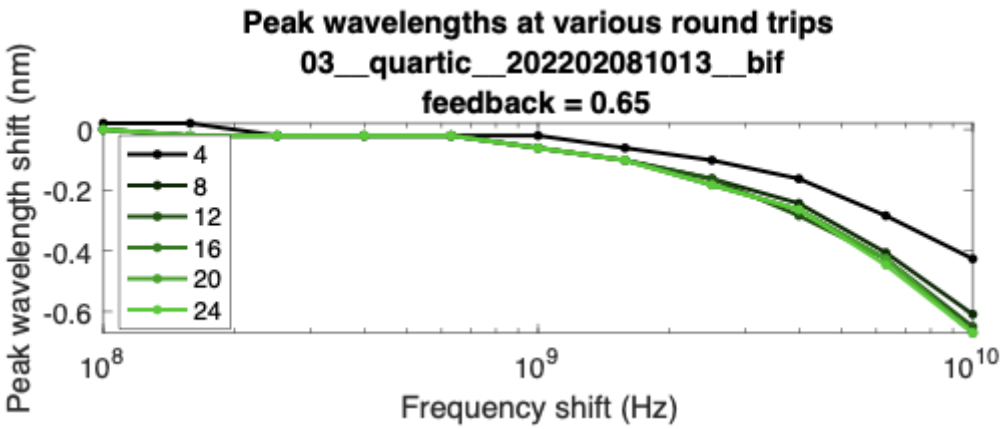
These have been superseded by the plots below.
Take a line from the grid above and plot that as a bifurcation plot:

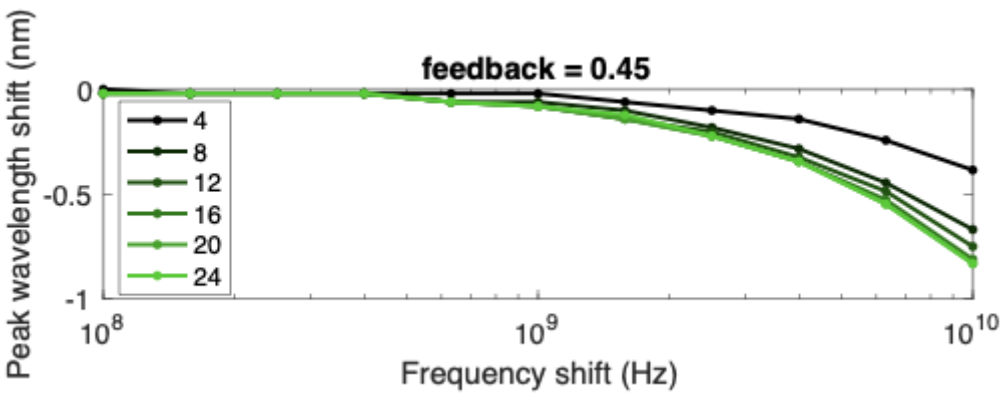
```
03__quartic__202202081012__bif.fig
```






```
03__quartic__202202081013__bif.fig
```





Each line represents a different round trip, and so we can see in the feedback = 0.65 and 0.8 that they group together after ~ 8 trips, indicating saturation.

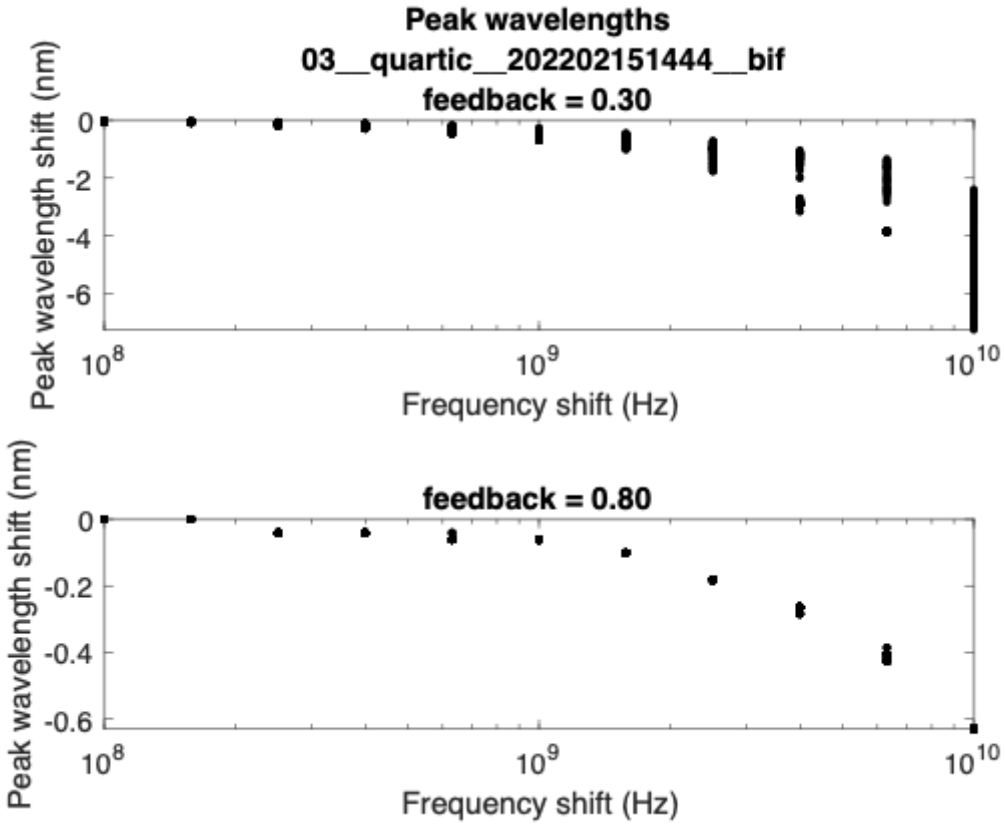
On the other hand, the feedback = 0.3 and 0.45 do not, indicating that it is not saturated yet (or rather that the soliton did not remain stable).

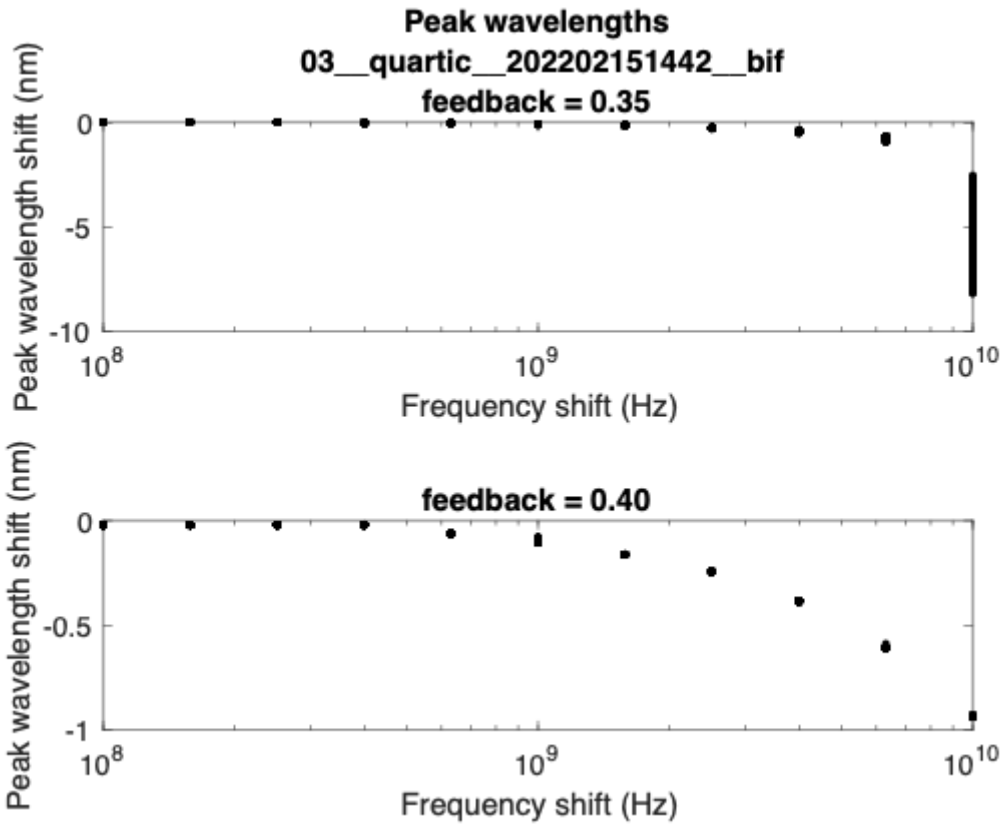
- However, I'm not sure how bifurcation would emerge in such a plot.

A.2.1.2 (Fixed) Bifurcation Plots

Using `quartic__220210__bif.m`, created bifurcation plots by:

1. For a particular frequency shift and feedback, start the simulation from the stable soliton (generated by running with no frequency shift)
2. Simulate until it stabilises (or maximum round trips reached)
3. Plot the final 100 peak wavelength points for each simulation
4. Repeat for the next frequency shift
5. Repeat for different feedbacks





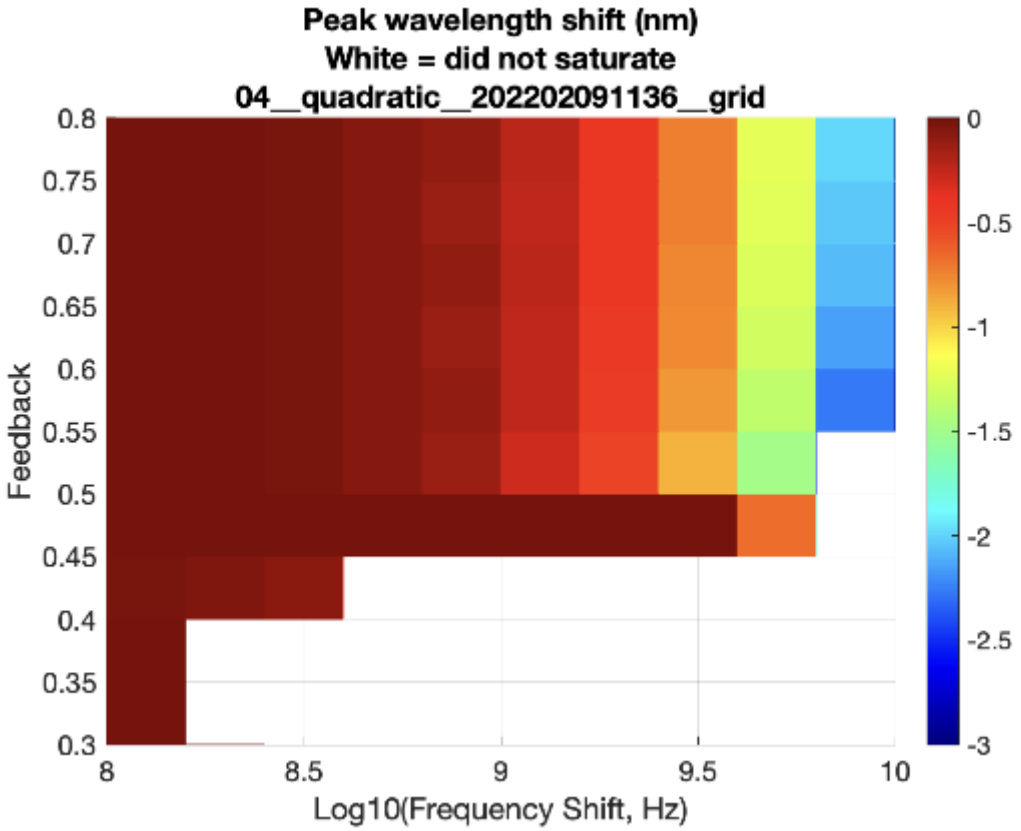
- We see that there is a a range of data points occurring at large frequencies for feedbacks smaller than 0.35
- This indicates instability / not reaching saturation

A.2.2 Quadratic Dispersion

Repeating the stabilisation procedure for quadratic dispersion, we get:

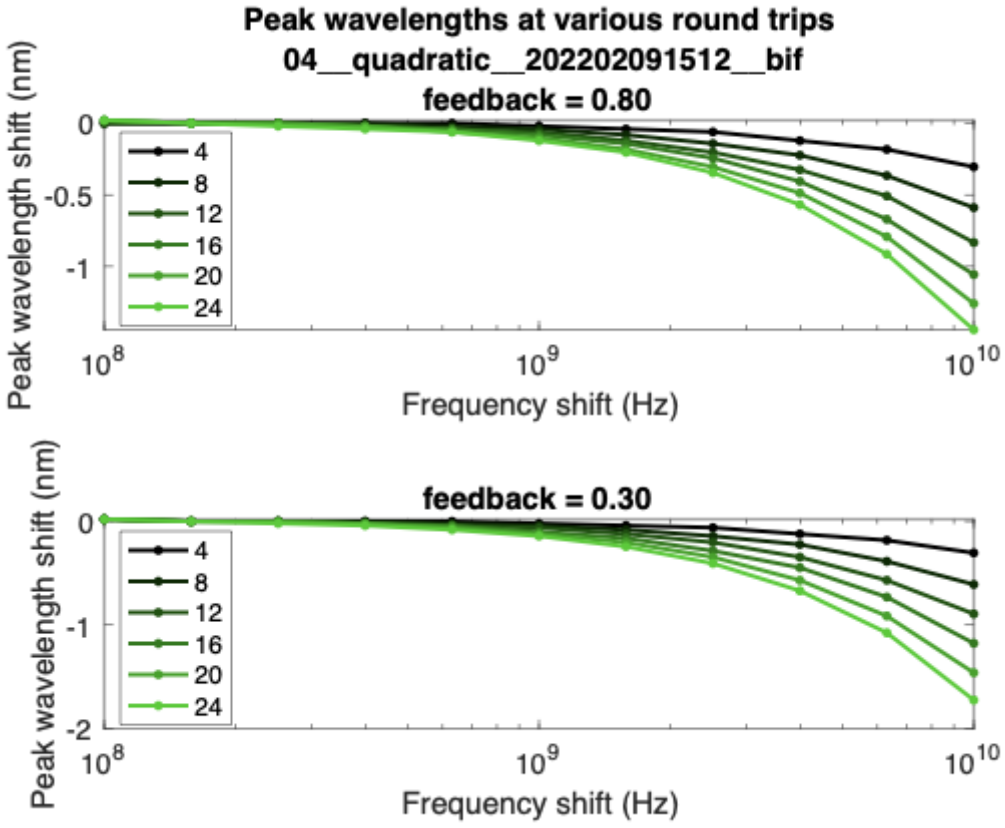
Data: `04__quadratic__202202082154__zero.mat` and `04__quadratic__202202082343__grid.mat`.

> `04__quadratic__202202091136__grid.fig`

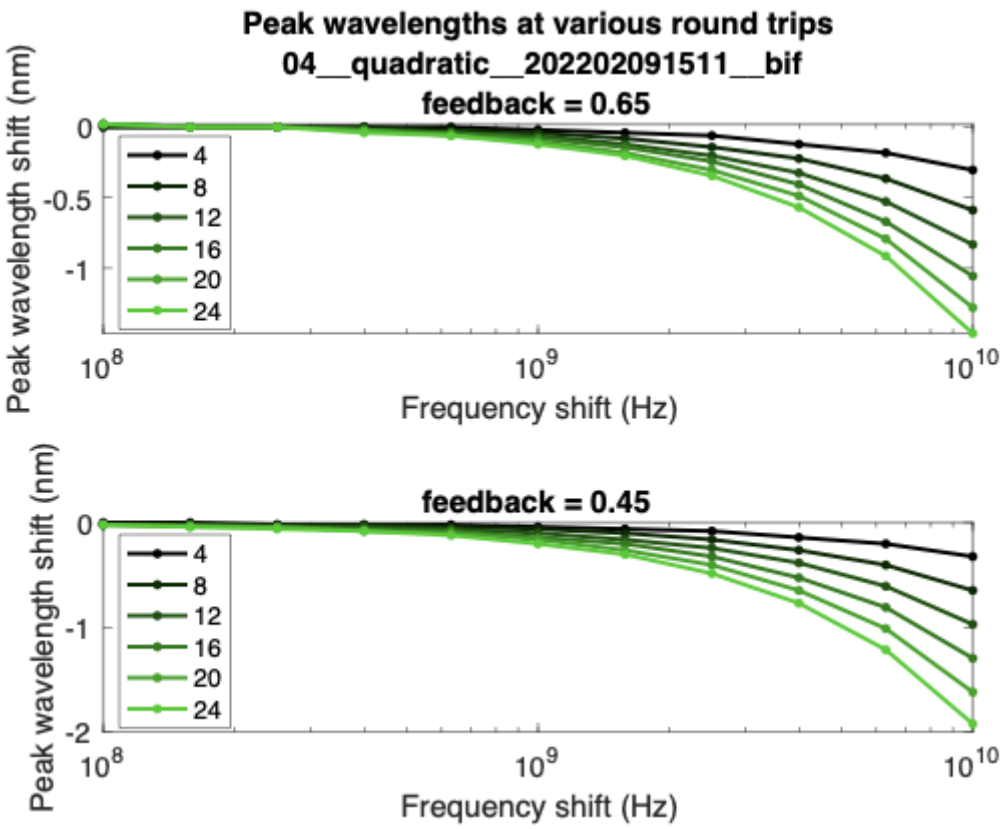


A.2.2.1 (Incorrect) Bifurcation

```
04__quadratic__202202091512__bif.fig
```



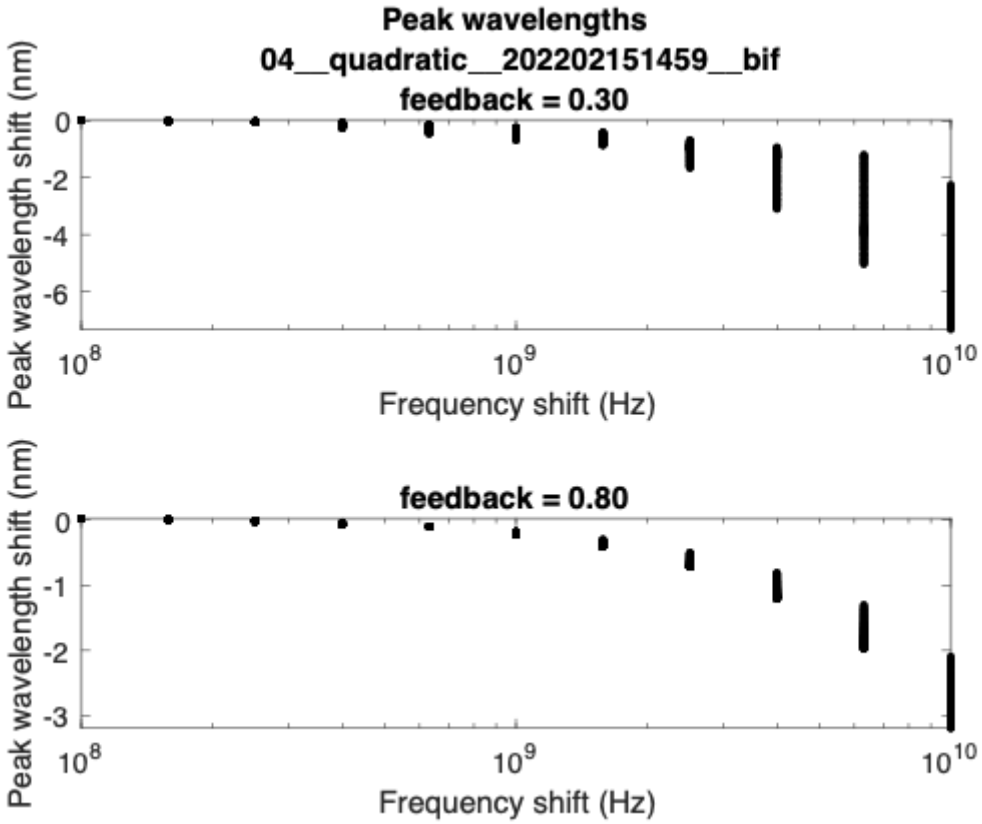
```
04__quadratic__2022020901511__bif.fig
```

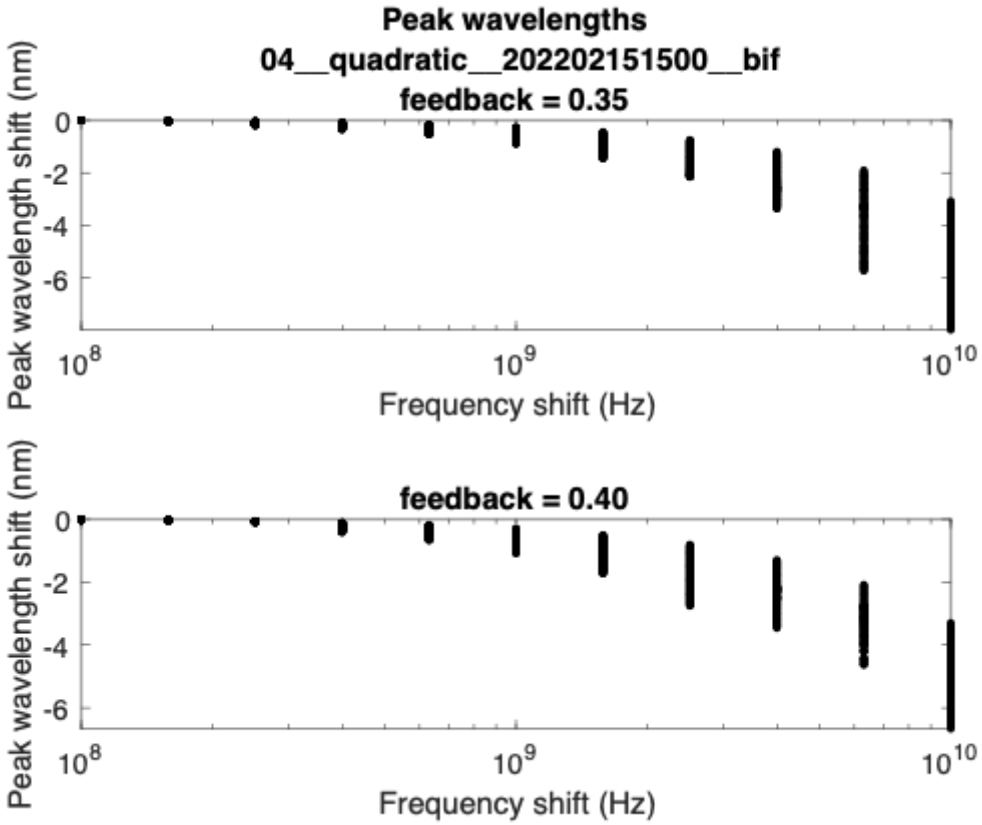


A.2.3 (Fixed) Bifurcation Plots

These have been superseded by the plots below.
Using `quadratic__220215__bif.m`:

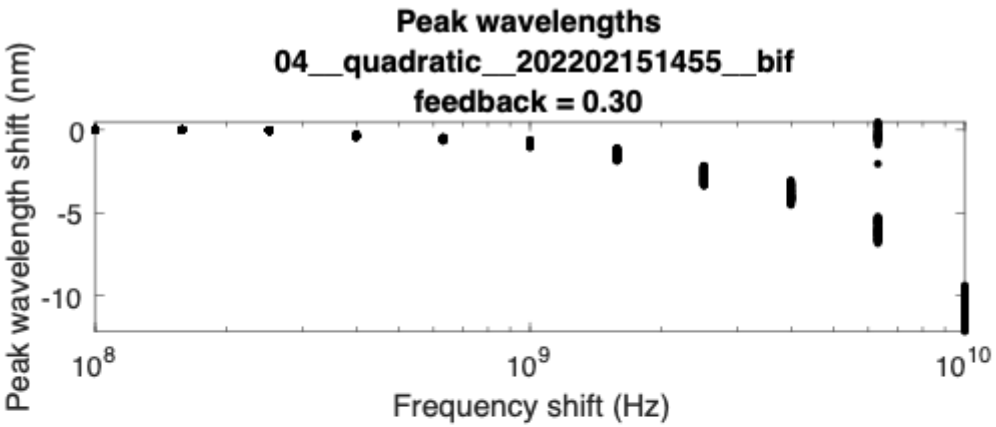
- Measuring from round trip $42 - 142$, to be consistent with the quartic plots above

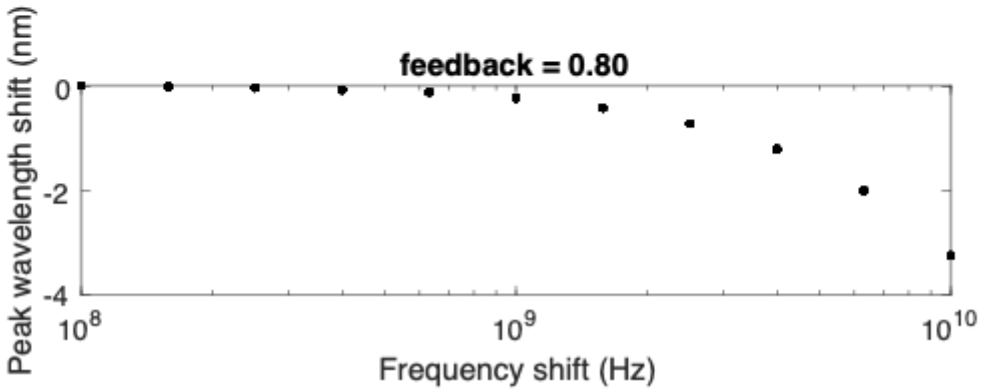


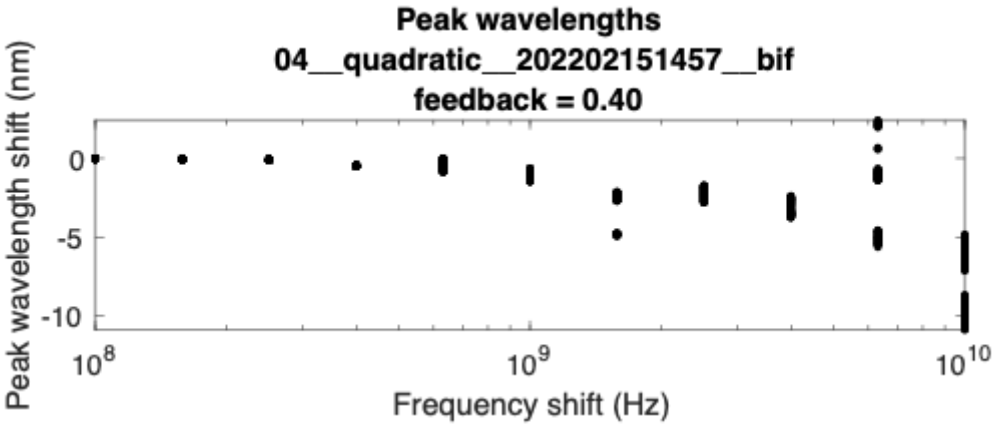


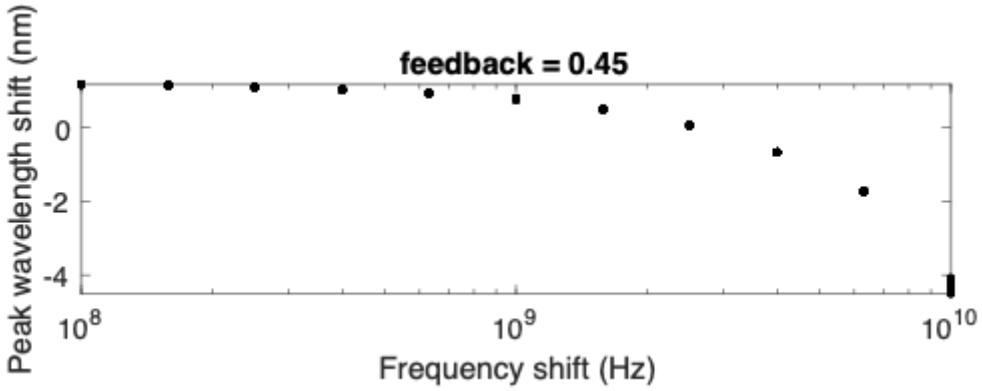
- So quadratic dispersion takes longer to stabilise at all feedback levels

If we measure from $175 - 250$ round trips, then we get:









- So the higher feedbacks still do stabilise (as expected), but requires longer time and is a higher threshold in general

A.3 Outcomes

- We are able to achieve wavelength shifts of ~ 2 nm for quadratic dispersion, and ~ 0.5 nm for quartic
- Although it takes longer for the quadratic dispersion to saturate, the difference between 50 and 250 round trips is negligible in practice, where the laser would be doing thousands per second.
- In both cases, smaller feedbacks cause larger wavelength shifts, but too low and the soliton will not saturate (or will not form correctly)
    - On one hand this makes sense, since if we have no feedback, then we are essentially starting from scratch each round trip
    - However, why does less feedback mean larger shifts?

A.4 To Do

- [x] ~~Repeat for quadratic dispersion~~
- [x] ~~Only show the points not the connecting lines for the bifurcation plots~~

B Analytic Project

Aims

- [ ] Compute the remainder for the Long data

B.1 Notes

B.1.1 Derivative of Long Data

We want to be able to take the fourth derivative of the Long data to compare the fit to our ansatz functions. However, numerically taking the fourth derivative in Mathematica via

```
fdata = Interpolation[data];
fdata''''[x]
```

does not produce a result due to noise that is amplified at each successive derivative. Instead, we can use a property of the Fourier transform:

The nth derivative of the function f is given by

$$\mathcal{F}\left\{\frac{\mathrm{d}^n}{\mathrm{d}x^n}f(x)\right\} = (2\pi i\xi)^n\mathcal{F}\left\{f(x(\xi))\right\}$$

Therefore we can take a nth derivative by

1. Compute the Fourier transform
2. Multiply by ω^n
3. Compute the inverse Fourier transform

https://www.tutorialspoint.com/time-differentiation-property-of-fourier-transform

```
(* Analytic derivative using Fourier *)
FD[f_, n_] :=
  InverseFourierTransform[FourierTransform[f[x], x, w] (I w)^n, w,
```

However, when computing the numerical FFT, we need to be careful with ensuring that the frequencies are shifted appropriately. To implement this in Mathematica, I followed https://www.phys.ksu.edu/personal/washburn/Teaching/Class%20Files/NQO/Tutorials/Tutorial8_FFT.pdf:

```
(* Numerical derivative using Fourier *)
NFD[tdat_, ydat_, n_?IntegerQ] :=
  Module[{num, range, \[Delta]t, \[Delta]\[Omega], nydata},
   num = Length[tdat];
   range = Max[tdat] - Min[tdat];
   \[Delta]t = range / num // N;
   \[Delta]\[Omega] = (2 Pi)/(num \[Delta]t) // N;
   time = Table[(j - (num/2))*\[Delta]t, {j, 1, num}];
   freq = Table[(j - (num/2))*\[Delta]\[Omega], {j, 1, num}];
   nydata = RotateLeft[data[[;; , 2]], num/2 - 1];
   f\[Omega]data = Chop[RotateRight[Fourier[nydata], num/2 - 1]];
   (* Multiply by omega^n *)
   nydata =
```

```
 RotateLeft[Table[f\[Omega]data[[j]] (I freq[[j]])^n, {j, 1, nur
  num/2 - 1];
ft2data = Chop[RotateRight[InverseFourier[nydata], num/2 - 1]];
 Abs[ft2data]
];
```

Info

We amplify noise because multiplying by ω puts more emphasis on the higher frequencies

B.1.2 Finite Difference

Alternatively, we can use the finite difference method: https://en.wikipedia.org/wiki/Finite_difference_coefficient to approximate derivatives to a desired accuracy. We can also improve the noise levels by taking every xth data point, instead of consecutive ones.

B.1.3 Combined Ansatz

Our current ansatz works well in the tails, but does not accurately fit our solution near the origin (it does converge to the zeroth order, but oscillations emerge in the fourth derivative).

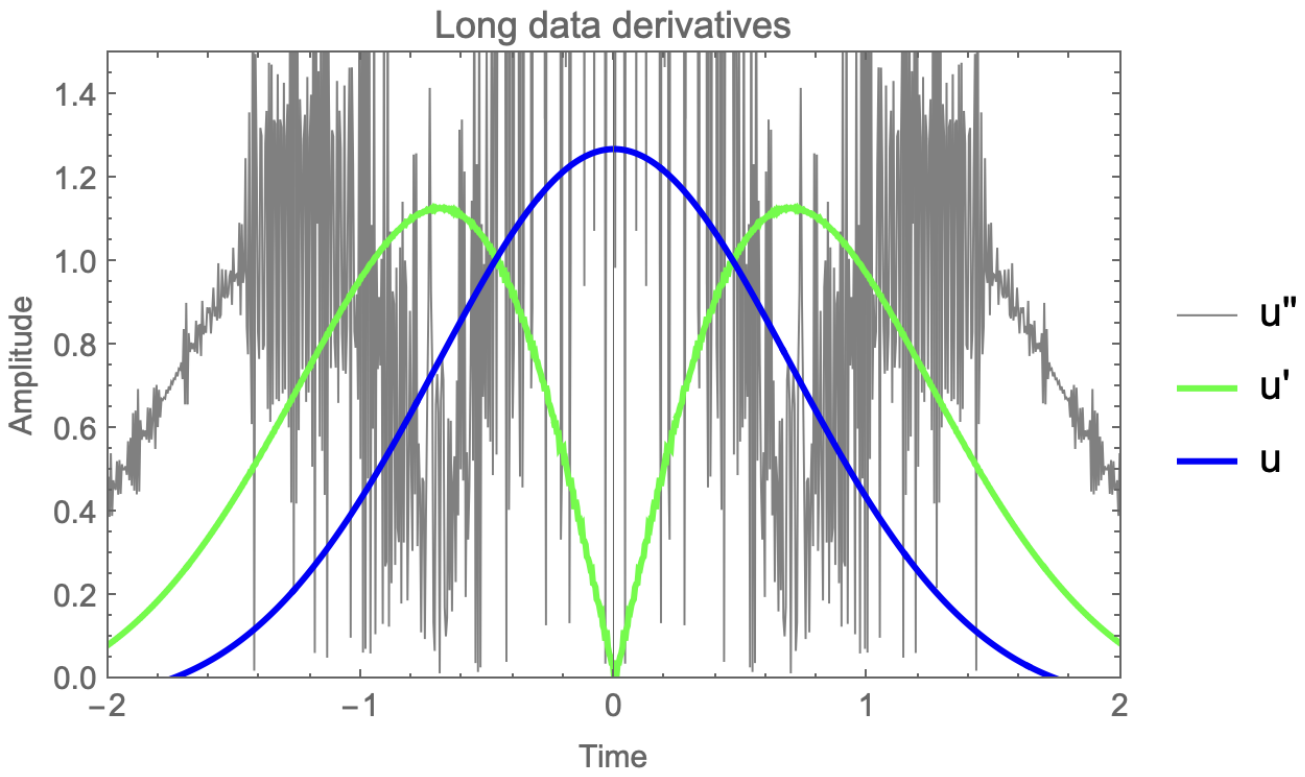
Perhaps we can use a weighted average to fit a better function near $x = 0$, and then use the previous ansatz in the tails.

B.2 Results

B.2.1 Derivative of Long Data

Using `06_remainder_220207.nb`:

B.2.1.1 Fourier



This is still too noisy to be able to use.

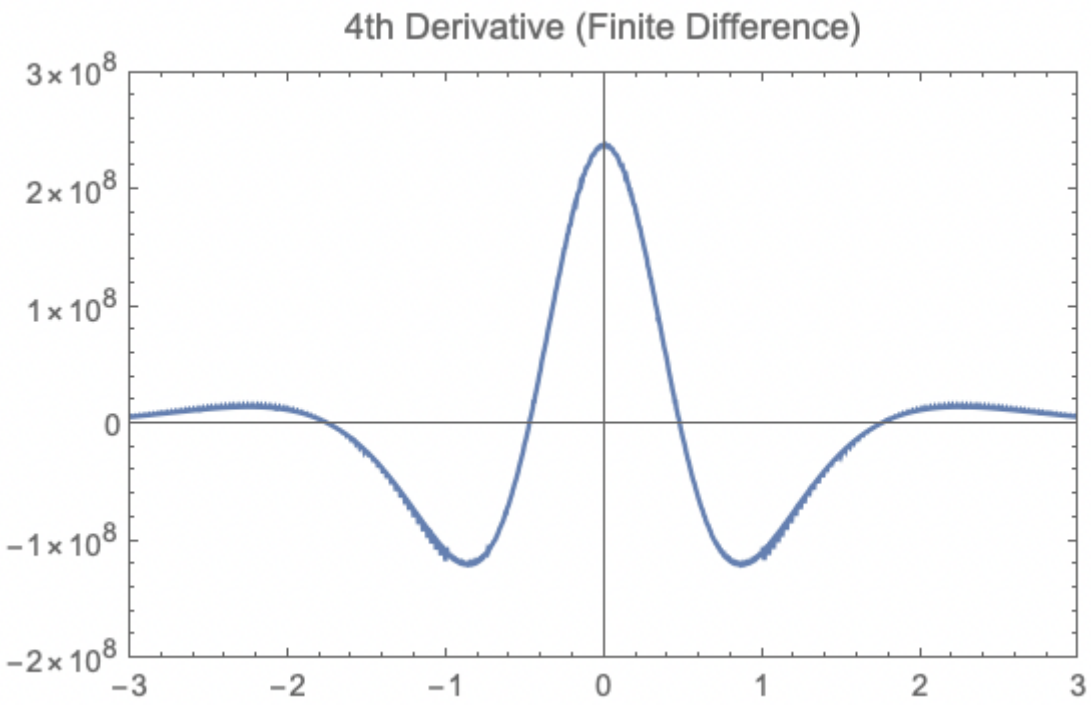
B.2.1.2 Finite Difference

We can instead use a finite difference approach, where we can take larger steps to help reduce noise:
https://en.wikipedia.org/wiki/Finite_difference_coefficient

Using a moving filter of span 480:

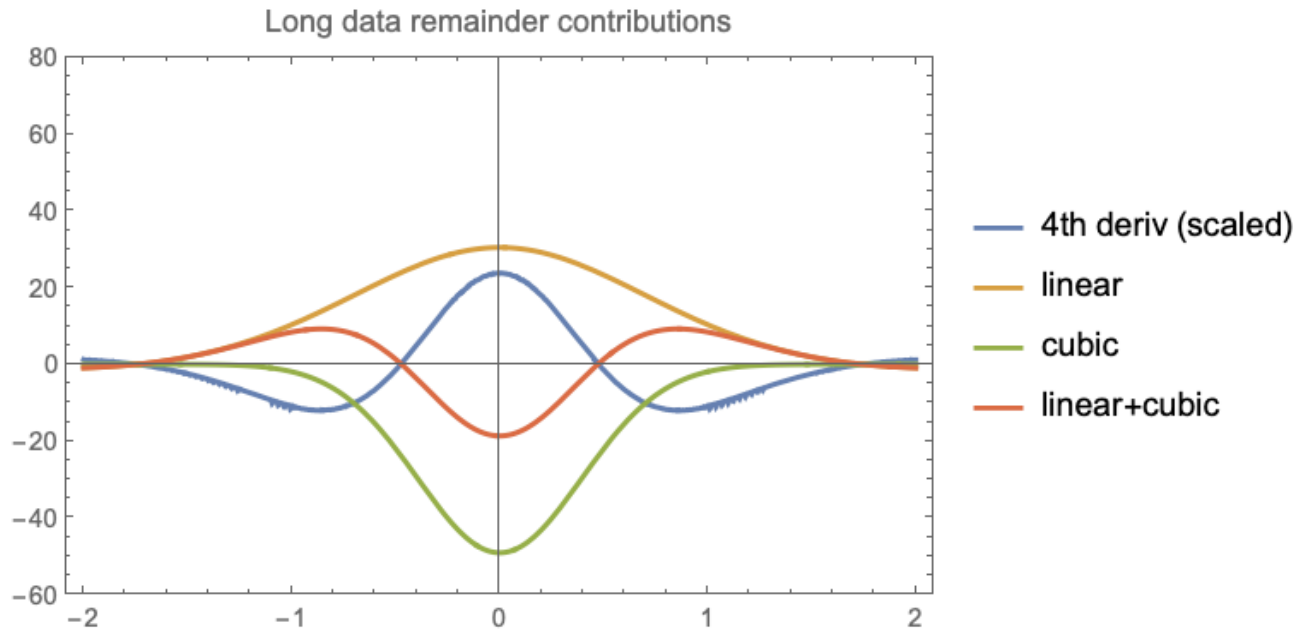
```
FiniteDifference[dat_] :=
  Transpose[{MovingMap[#[[241]] &, dat[[;; , 1]], 480],
    MovingMap[(7 (#[[1]] + #[[481]]) -  96 (#[[61]] + #[[421]]) +
          676 (#[[121]] + #[[361]]) - 1952 (#[[181]] + #[[301]]) +
          2730 #[[241]])/240 &, dat[[;; , 2]], 480] /
     MovingMap[(#[[241]] - #[[240]])^4 &, dat[[;; , 1]], 480]}];
```

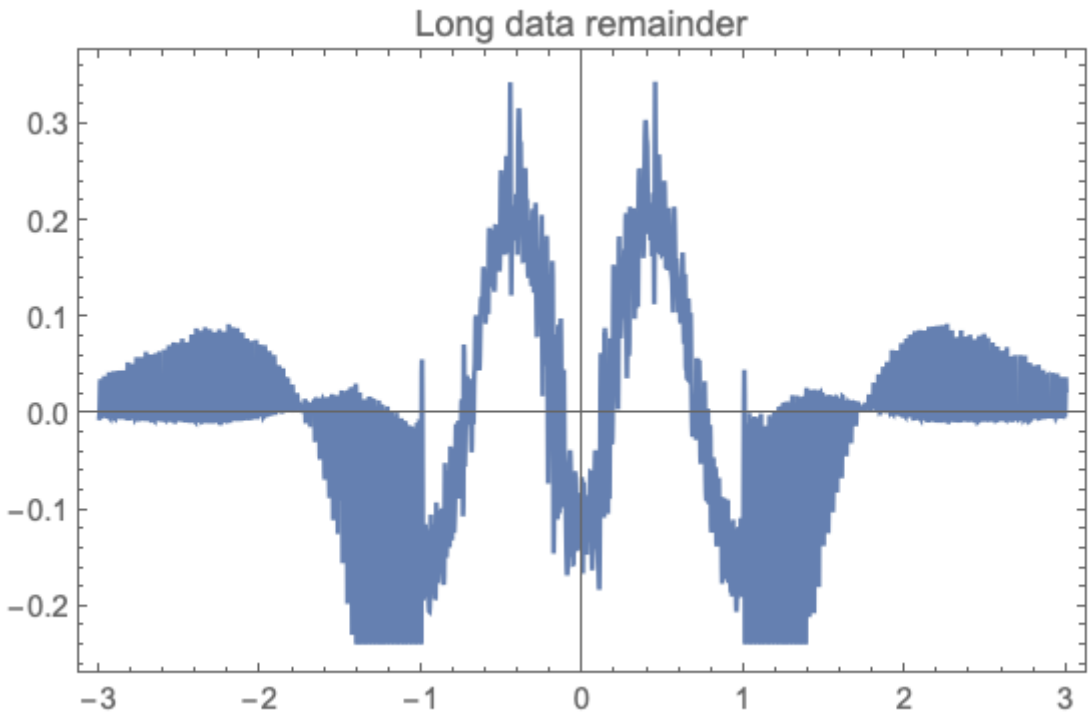
We get a much cleaner result



- But the scaling is off by a factor of $\sim 10^7$
- Perhaps I am taking the denominator incorrectly

When we substitute it into the DE (and scale appropriately by eye):





The contributions sum to near 0, as expected.

B.2.2 Find A, B using higher u

I also investigated briefly if our estimates of A, B change when using $u^{(1)}$ etc. instead of just $u^{(0)}$

- They theoretically shouldn't, because we chose A, B so that the fit in the tails is good
- And higher u shouldn't affect the fit at large x

Revisited the process to find estimates of A, B except now considered $\ldots + u^{(3)}$. Wrote the script in `f01_220208_compare.m`.

- Also redefined θ so that the amplitude doesn't change:

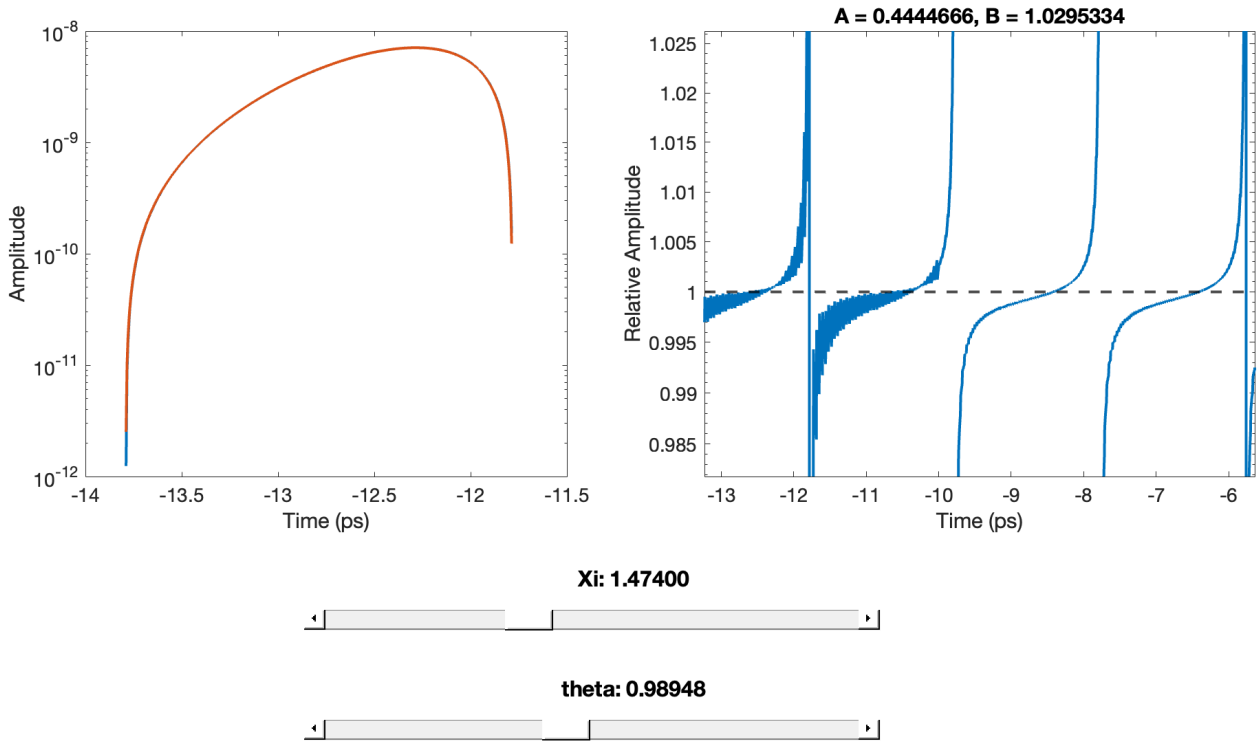
$$A = \Xi \cos^2\theta;\ \ B = \Xi \sin^2\theta$$

Ended up with

$$\Xi = 1.474;\ \ \theta = 0.98948$$

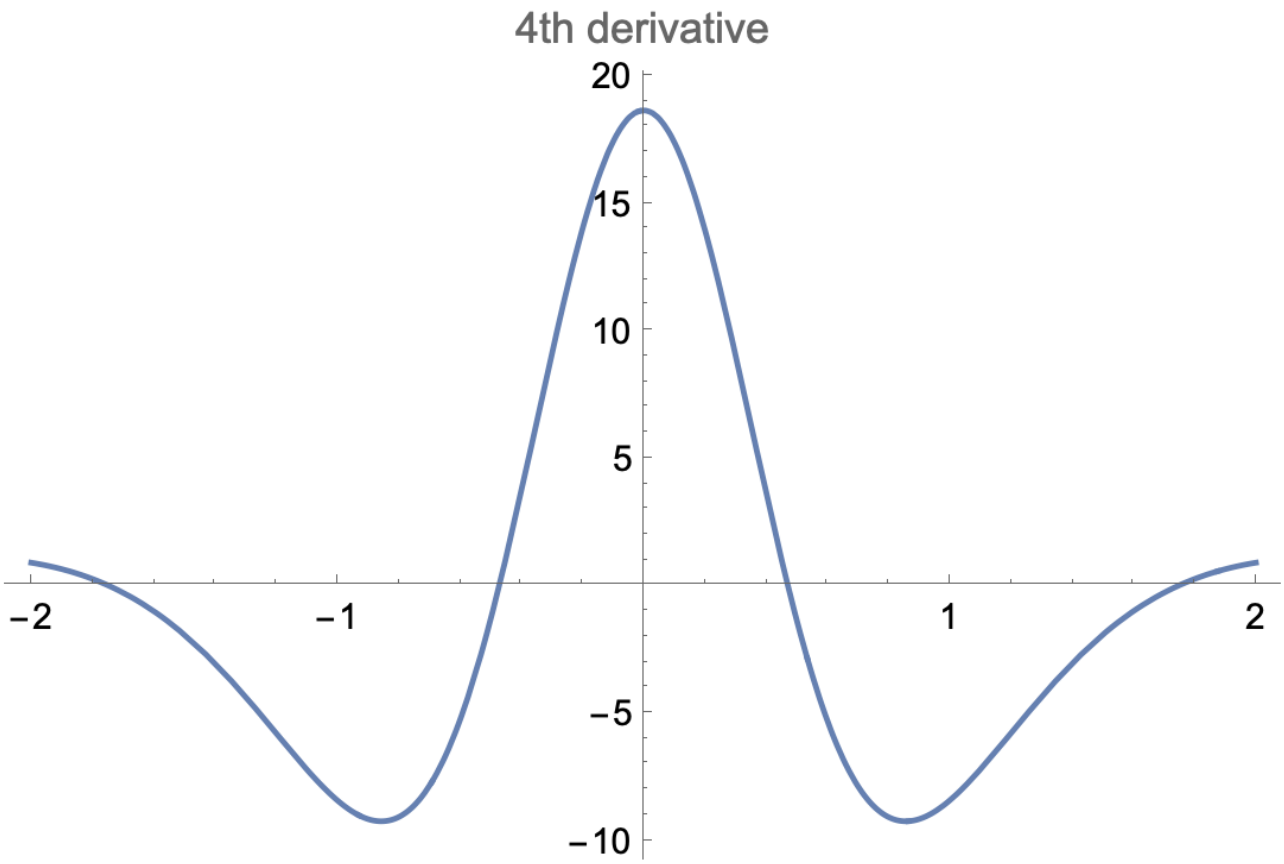
$$A = 0.4445, B = 1.0295$$

which is very close to the previous estimates.



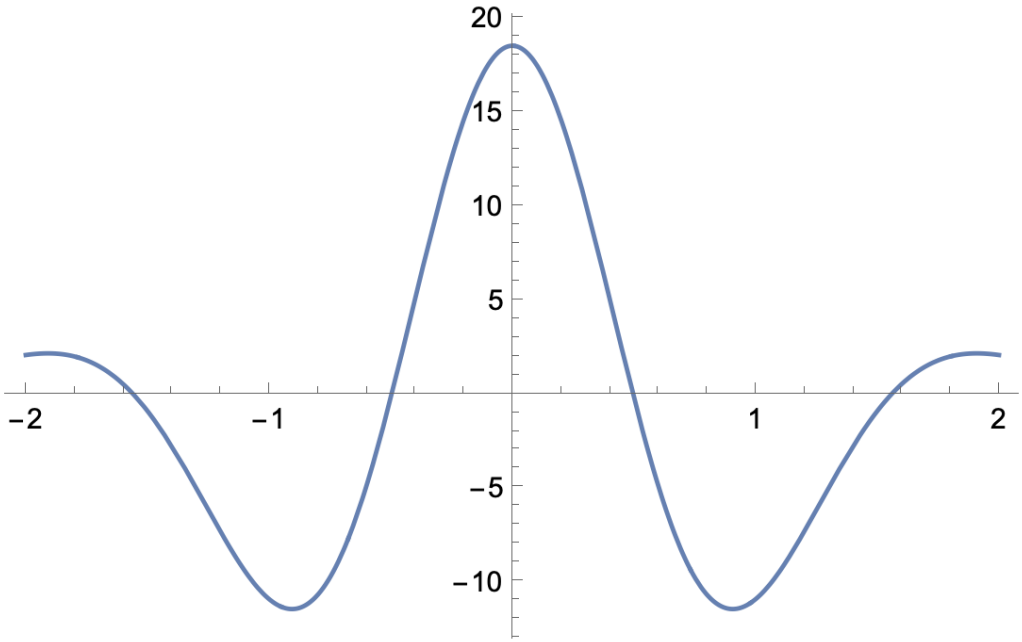
B.2.3 Combined Ansatz

We know that the function looks qualitatively like a gaussian near the origin, and we can find the form of the 4th derivative using Long's data (either by taking a numerical fourth derivative or by just considering the negative of the linear + cubic terms):



This looks similar to the 4th derivative of a Gaussian:

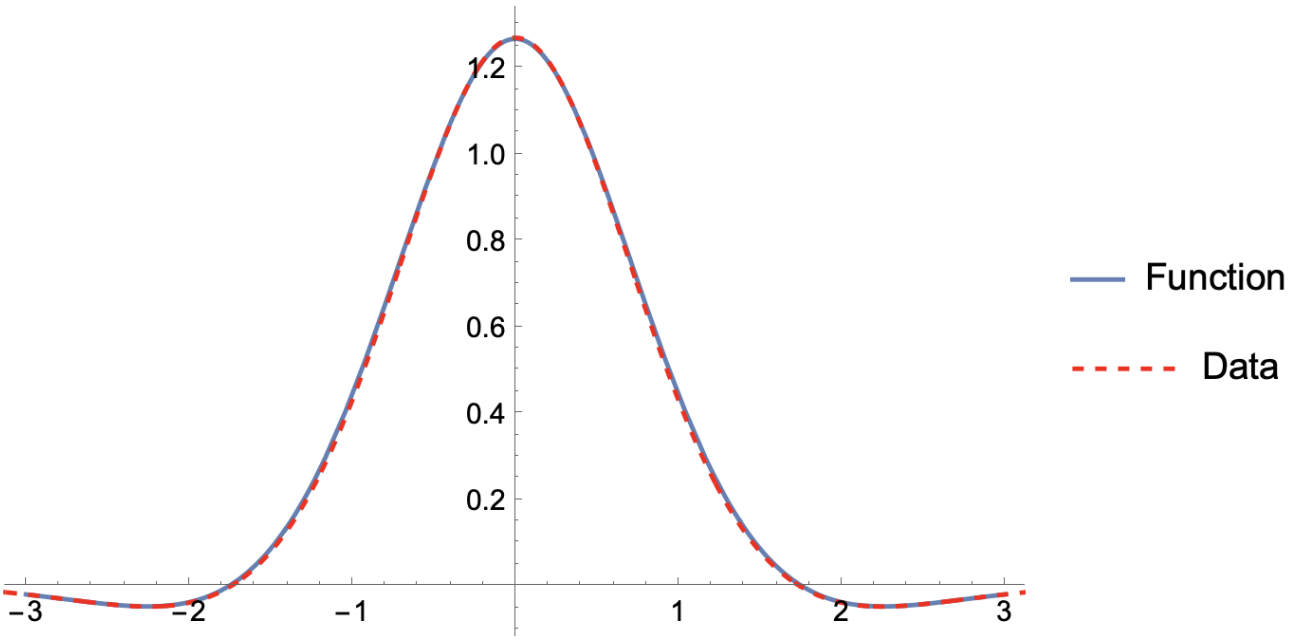
```
Plot[de1[x ↦ 1.23 Exp[- 1.12 x^2]][x] /. param, {x, -2, 2}]
```

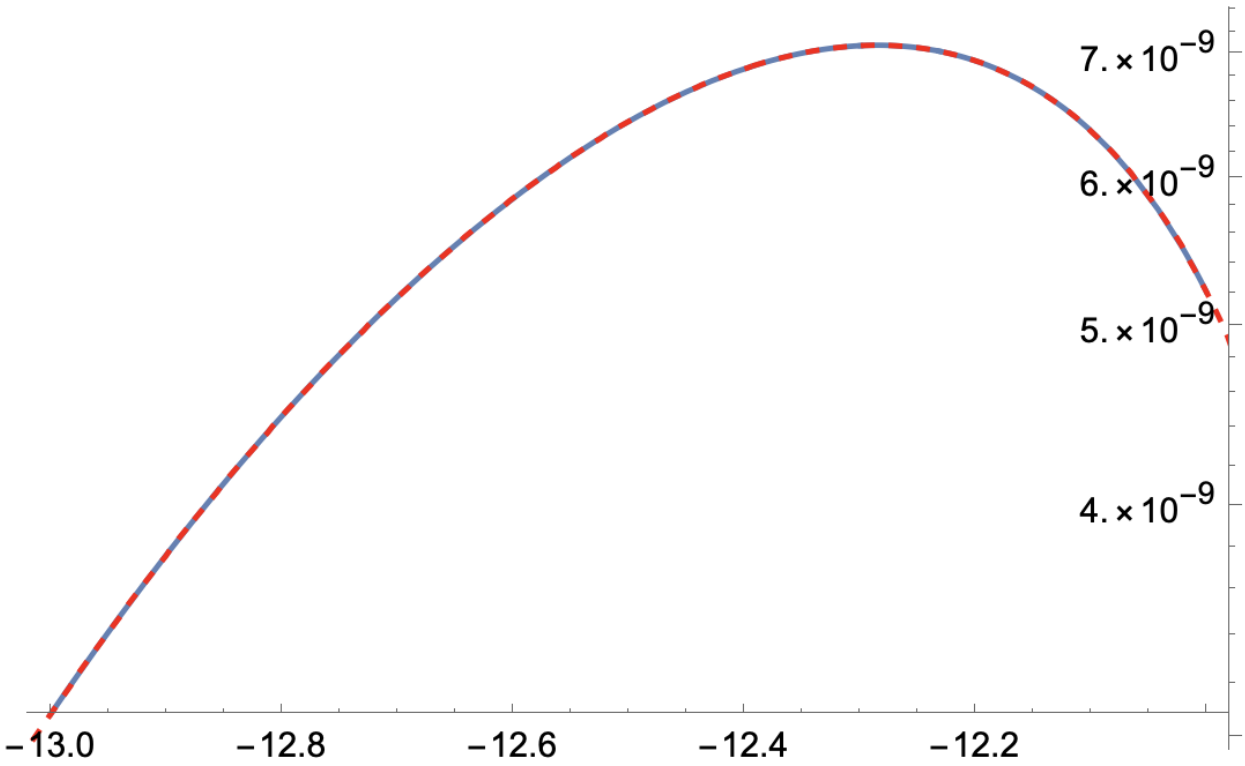


So what if we used a weighted sum to combine a Gaussian with $u^{(0)}$? For example:

$$v(x) = \frac{u^{(0)}(x) + 0.81e^{-1.874x^2}}{1 + 0.81e^{-x^2}}$$

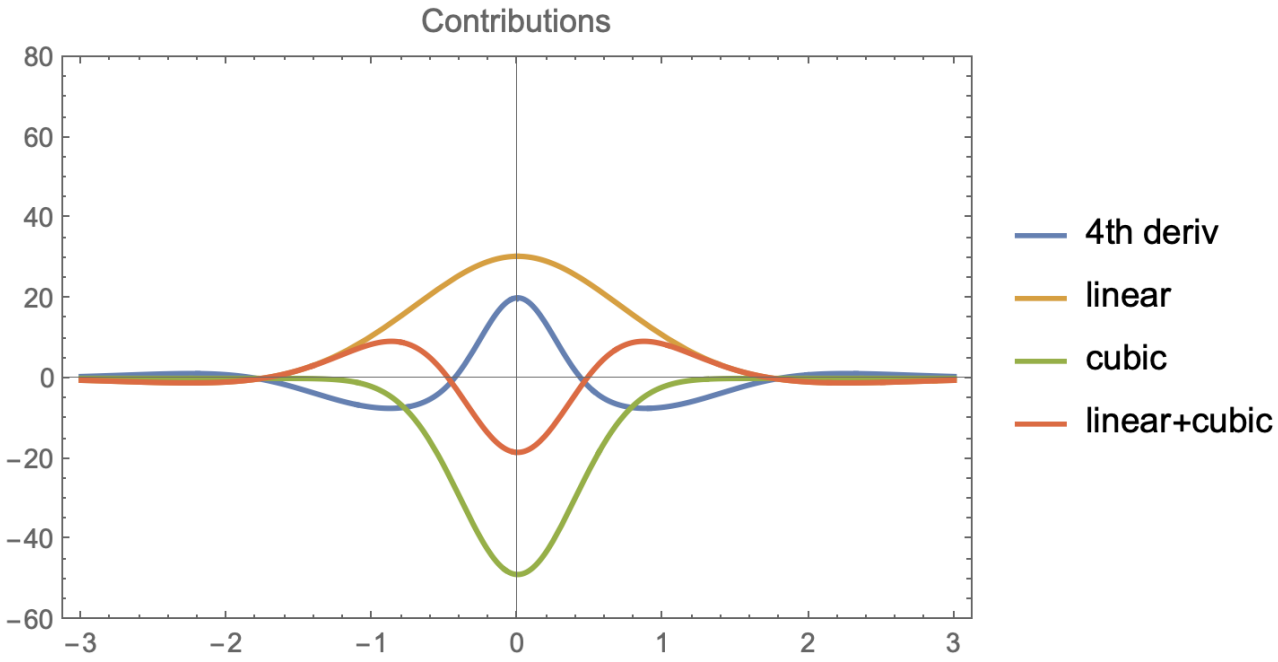
(with numbers found by visually matching the function to the data)

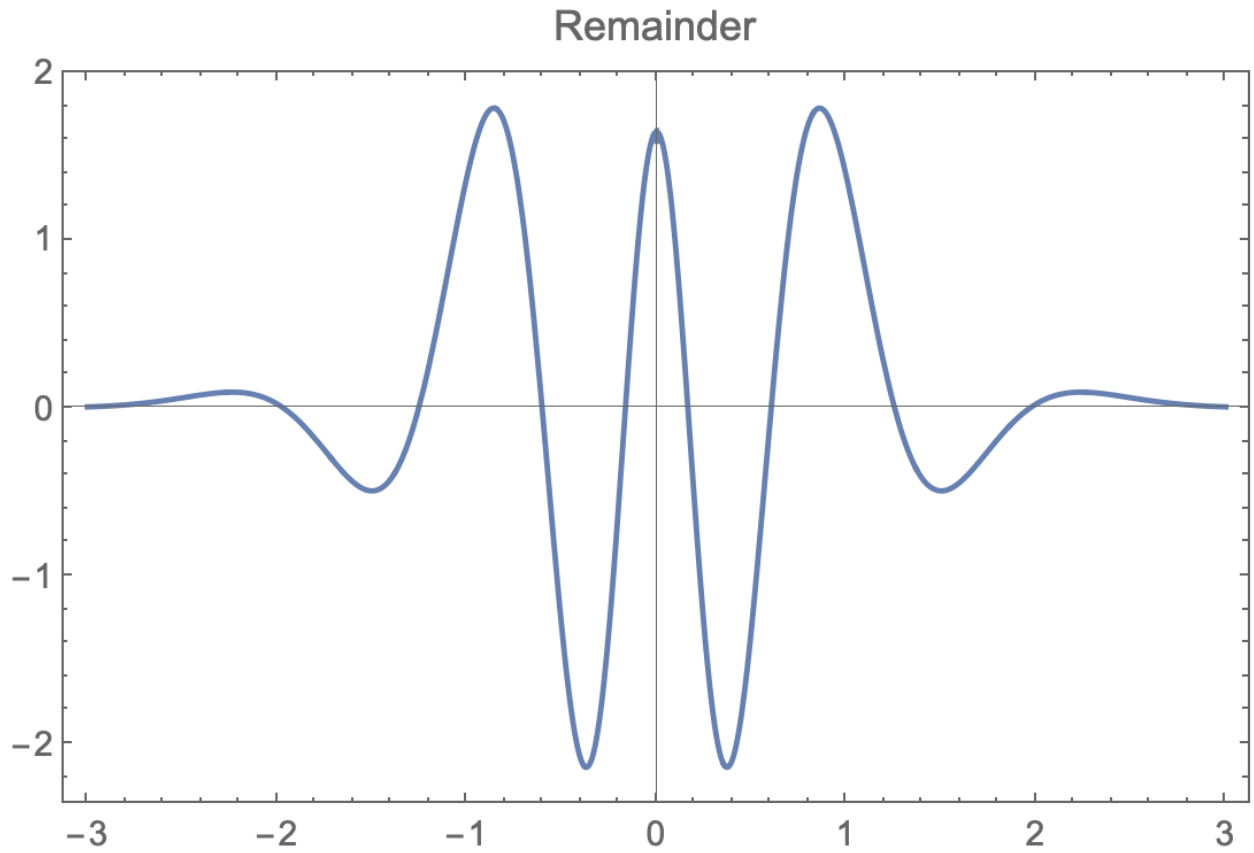




It fits well in both the peak and the tails.

The contributions to the DE also look much better:





with the remainder only being ~ 2 in magnitude, compared to ~ 20 for just the $u^{(0)}$. (See Logbook_06_220201)

B.3 Outcomes

- We have to consider how there could be a better approach for this, more systematic than just plugging in weights and functions (ideally not dependent on the Long data)

B.4 To Do

- [x] ~~Substitute Long solution into the DE and look at components~~
- [x] ~~Find values of A, B for ... + u3 etc., not just u1 and see if it changes anything~~
- [ ] Why does this functional form work? Look at the numerical expressions etc.
- [x] ~~Send the slides to Martijn~~