1. (a) The fit was done by iteratively adding an additional order, starting with a 1st order chebyshev fit. The error was always taken to be the absolute value of the final coefficient, since we were truncating the series there. Thus the n-1 first coefficients would be the returned coefficients of the fit. This produced the following output:

```
Order 0 gives error 0.4923648
Order 1 gives error 0.0421532
Order 2 gives error 0.0048173
Order 3 gives error 0.0006197
Order 4 gives error 0.0000850
Order 5 gives error 0.0000122
Order 6 gives error 0.0000018
Order 7 gives error 0.0000003
```

Indicating that a 7 terms where needed. Comparing the errors resulted in:

```
Maximum Errors:
        Cheby: 3.228338e-07
        Poly: 3.507079e-07
RMS Errors:
        Cheby: 1.932287e-07
        Poly: 1.800043e-07
```

So we see that both fits performed rather well, with the polynomial winning in RMS, but the Chebyshev fit gave a smaller maximum error. This is visible in the residuals of Fig. 1 where the Chebyshev fit results in less deviation at the ends, but the regular polynomial looking better in the middle.

(b) Any number > 0 can be expressed as $m * 2^x$ with m the mantissa $\in (0.5, 1]$ and x some integer exponent. The mantissa is always $> \frac{1}{2}$ Since otherwise we could just subtract one from x and multiply the mantissa by 2. Similarly, it won't be greater than 1 as we could then add one to x and divide the mantissa by 2.

With that in mind, we can then express the base 2 logarithm of any positive number as:

$$log_2(m * 2^x) = log_2(m) + log_2(2^x) = log_2(m) + x$$

We can thus use the previous fit for the logarithm between 0.5 and 1 and `np.frexp` to calculate $log_2(x)$ for a number in any range, allowing us to produce Fig. 2.

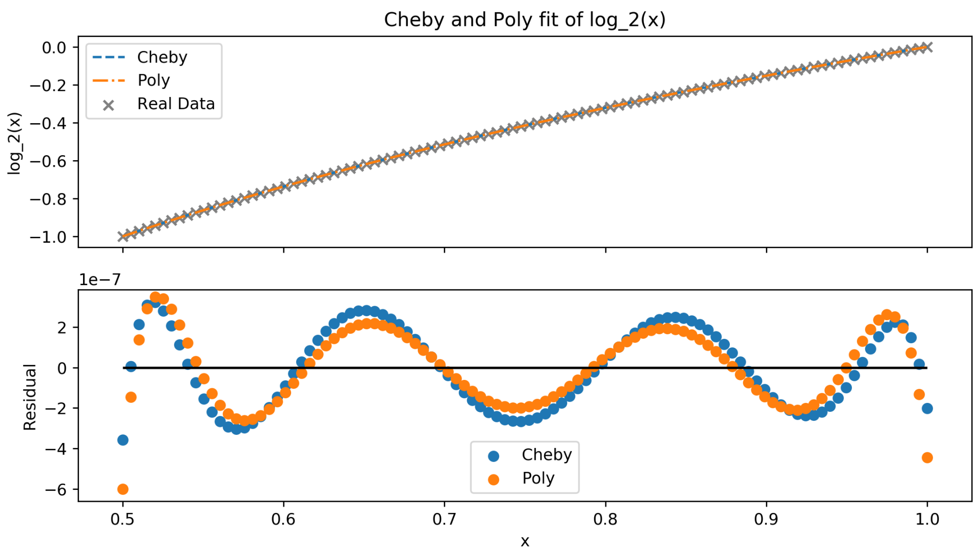

Figure 1: Chebyshev polynomial fit vs regular polynomial fit for function $\log_2(x)$.

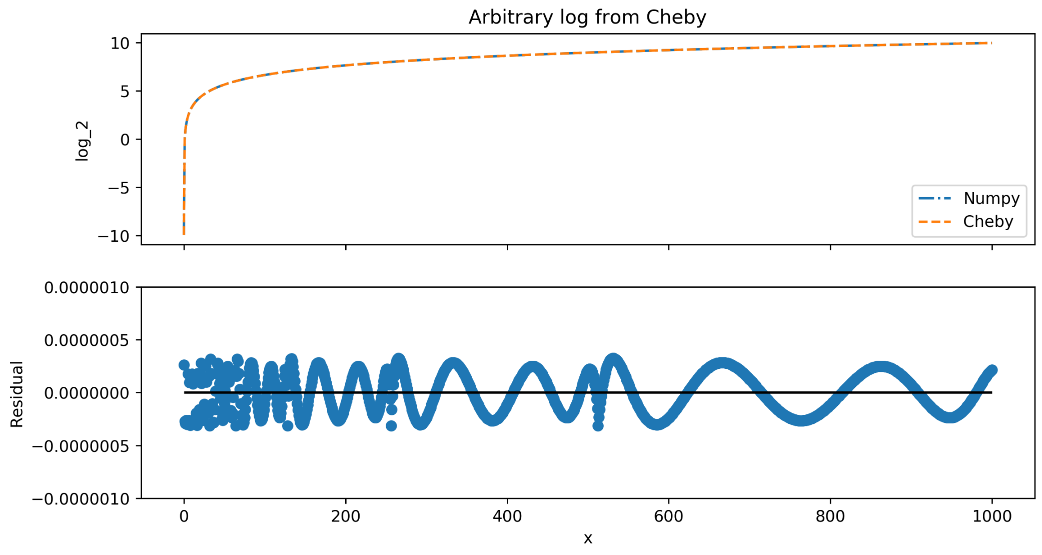

Figure 2: $log_2(x)$ calculated using a chebyshev polynomial fit from 0.5 to 1 and `np.frexp`, compared to `np.log2`.

2. (a) i. Since this model requires multiple parameters and a nonlinear function ($\exp(x)$) is involved, it is not a linear model.

ii. Since we are fixing the starting time t_0, the model requires three parameters, an offset f_0, an amplitude A and a time constant τ. The model function willl then be a decaying exponential given by

$$f(x) = f_0 + A \exp\left\{-\frac{(t - t0)}{\tau}\right\},$$

or in python code:

```
y = p[0] + p[1] * np.exp(-(t-t0)/p[2])
```

I took a starting offset of 1.0 an amplitude of 0.26 since the peak value looks a bit above 1.25 and a time constant of 0.05, since 0.25 seconds after the peak the exponent is already basically at 0 which should correspond to 5 time constants. See part (b) for plot with both initial guess and optimized fit.

(b) After optimizing the optimial parameters are found to be

```
[1.00194174 0.26085036 0.01540617]
```

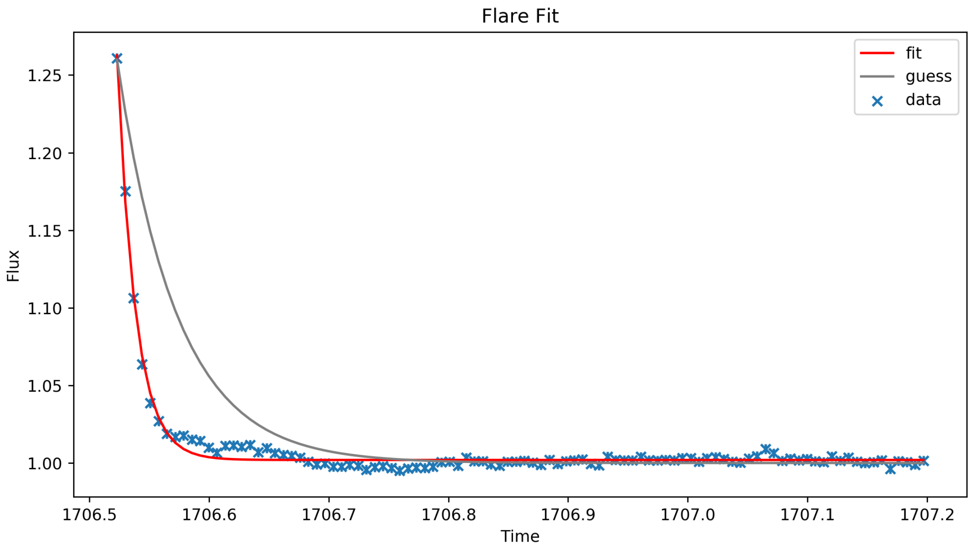

Figure 3: Initial guess and fit of decaying exponential to flare data.

(c) In class we saw that

$$\sigma_m^2 = diag\left(\langle (m-m_t)(m-m_t)^\top\rangle\right) = diag\left(\left(\frac{1}{2}\frac{\partial^2\chi^2}{\partial m^2}\right)^{-1}\right).$$

Since in Newton's method we estimate that

$$\frac{\partial^2\chi^2}{\partial m^2} \approx 2A'(m)^\top N^{-1}A'(m),$$

we can take the errors on the parameters to be

$$\sigma_m = \sqrt{diag\left((A'(m)^\top N^{-1}A'(m))^{-1}\right)}.$$

To get an idea of what N should be, I took the standard deviation of all the data before the flare time σ_d. N would then simply be $\sigma_d^2 I$, plugging this in results in

$$\sigma_m = \sigma_d \cdot \sqrt{diag\left((A'(m)^\top A'(m))^{-1}\right)}.$$

The script produces the following results:

```
With parameters: [1.      0.26  0.05],
Iteration 0 gives chi2 of 8245.470399
With parameters: [1.00367122 0.20254194 0.02291222],
Iteration 1 gives chi2 of 702.552041
With parameters: [1.00204386 0.25281272 0.01399754],
Iteration 2 gives chi2 of 210.257034
With parameters: [1.00197512 0.26099529 0.01535158],
Iteration 3 gives chi2 of 148.926113
With parameters: [1.00194174 0.26085036 0.01540617],
Iteration 4 gives chi2 of 148.885625
Converged!
With parameters: [1.00194089 0.26083417 0.01540966],
Iteration 4 gives chi2 of 148.885625
Errors on fit: [0.00032999 0.00285143 0.00030042].
Estimated data error: 0.003113.
```

These errors are relatively sensible compared to the parameters, they are a small but not insignificant fraction.

(d) Since I already considered the error by looking at the data before the flare, these parameter errors seem reasonable since, with a few hundred points, the error on each fit parameter is roughly $\sigma_d/\sqrt{N}$.