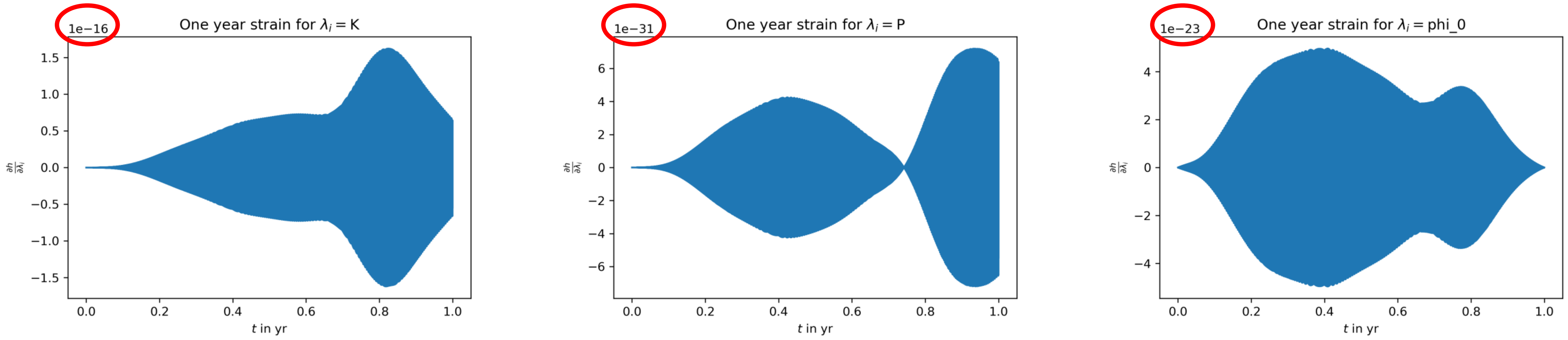
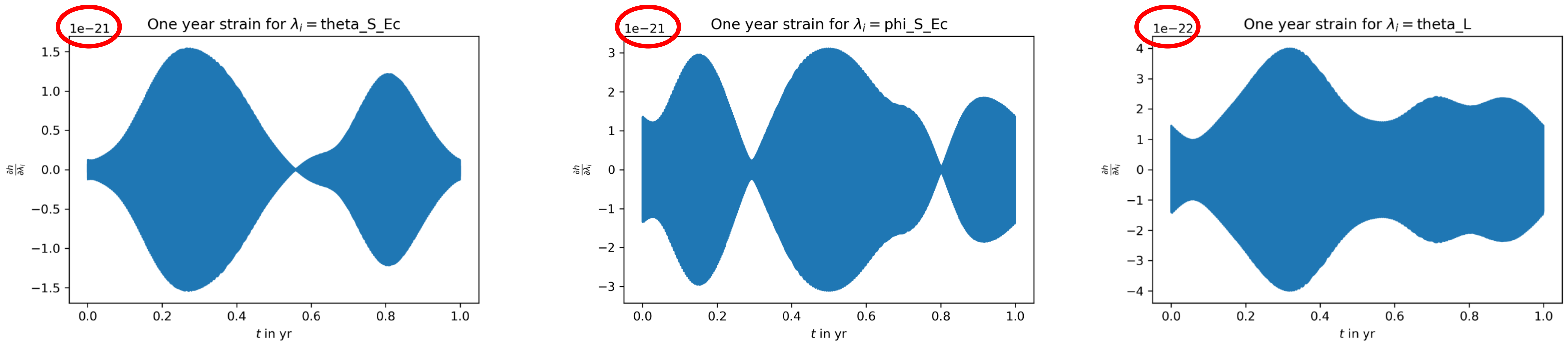
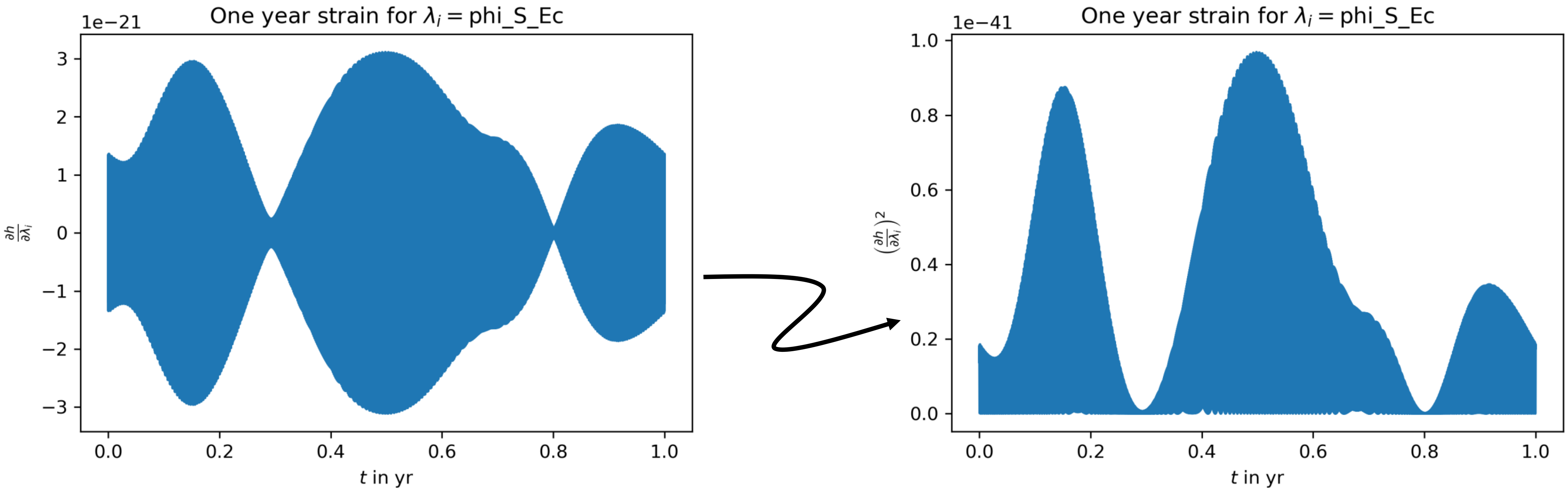
Detecting Exoplanets and recovering Parameters

A short recap of my work in week 5/6



The derivatives $\frac{\partial h_\alpha}{\partial \lambda_i}$ look good, compare the analytic derivatives of the exoplanet par‘s (above) with the numerical derivatives of the position par‘s (below) – same structure: envelope * oscillating function





Numerical difficulties in the integration come from the product $\frac{\partial h_\alpha}{\partial \lambda_i}(t)\frac{\partial h_\alpha}{\partial \lambda_j}(t)$:

For amplitudes of $\frac{\partial h_\alpha}{\partial \lambda_i} \sim O(10^{-20})$ we get time dependant amplitudes $(\frac{\partial h_\alpha}{\partial \lambda_i})^2 \sim O(10^{-40})$ and this will strongly hinder the integration! (Machine epsilon for doubles: 2e-16)

Quick fix: $\int_0^{T_0} dt \frac{\partial h_\alpha}{\partial \lambda_i}(t)\frac{\partial h_\alpha}{\partial \lambda_j}(t) = 10^{-40} \times \int_0^{T_0} dt (\frac{\partial h_\alpha}{\partial \lambda_i}(t) \times 10^{20}) \times (\frac{\partial h_\alpha}{\partial \lambda_j}(t) \times 10^{20})$

A new problem has approached

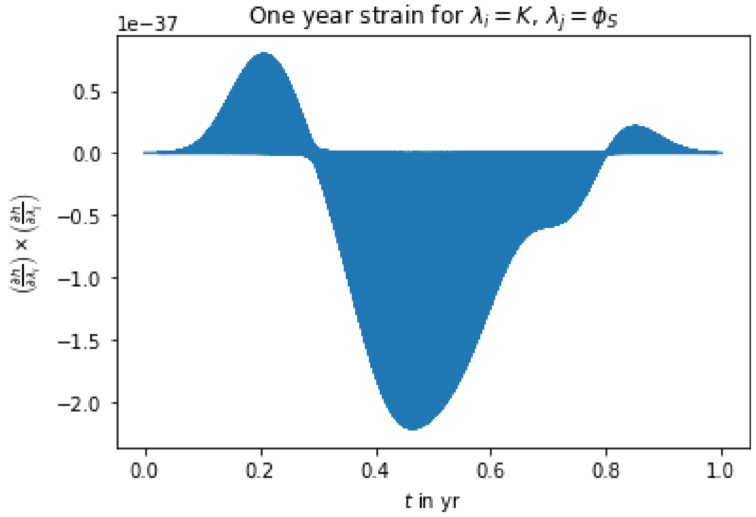
We compute a 9x9 symmetric matrix $\Gamma_{ij} \propto \left[\int_0^{T_0} dt \frac{\partial h_\alpha}{\partial \lambda_i}(t) \frac{\partial h_\alpha}{\partial \lambda_j}(t)\right]_{ij}$

We know, diagonal elements Γ_{ii} will be in phase, so the integrand will go like $A^2 \sin^2 \omega t$ and we find for diagonal elements a result which goes roughly like $\Gamma_{ii} \propto \frac{1}{2}\int_0^{T_0} A^2(t)dt + O\left(\frac{A_0}{4\omega}\right) \approx \frac{1}{2} {A_0}^2 T_0 > 0$ with A_0 some representative value of the amplitude

A new problem has approached

We compute a 9x9 symmetric matrix $\Gamma_{ij} \propto \left[\int_0^{T_0} dt\, \frac{\partial h_\alpha}{\partial \lambda_i}(t)\, \frac{\partial h_\alpha}{\partial \lambda_j}(t)\right]_{ij}$

On the other hand, off-diagonal elements will go as $A_i A_j \sin\omega_i t \sin\omega_j t = \frac{A_i A_j}{2}\left(\cos(\omega_i - \omega_j)\, t \cos(\omega_i + \omega_j)\, t\right)$ and if they are uncorrelated, we'll find over long observations $\Gamma_{ij} \approx 0$



Question: What does *close to zero* mean for $\frac{1}{2} A_0{}^2 T_0 \sim O(10^{-40+7})$? Where do we make the cut?

A short recap on numerical integration

scipy.integrate calls QUADPACK from Fortran and then using a Clenshaw-Curtis method which uses Chebyshev moments computes the integral

Two parameters we can play with: epsabs and epsrel

The numerical integral `result` is returned if for the actual integral `i`

```
abs(i-result) <= max(epsabs, epsrel*abs(i))
```

Which can be estimated analytically

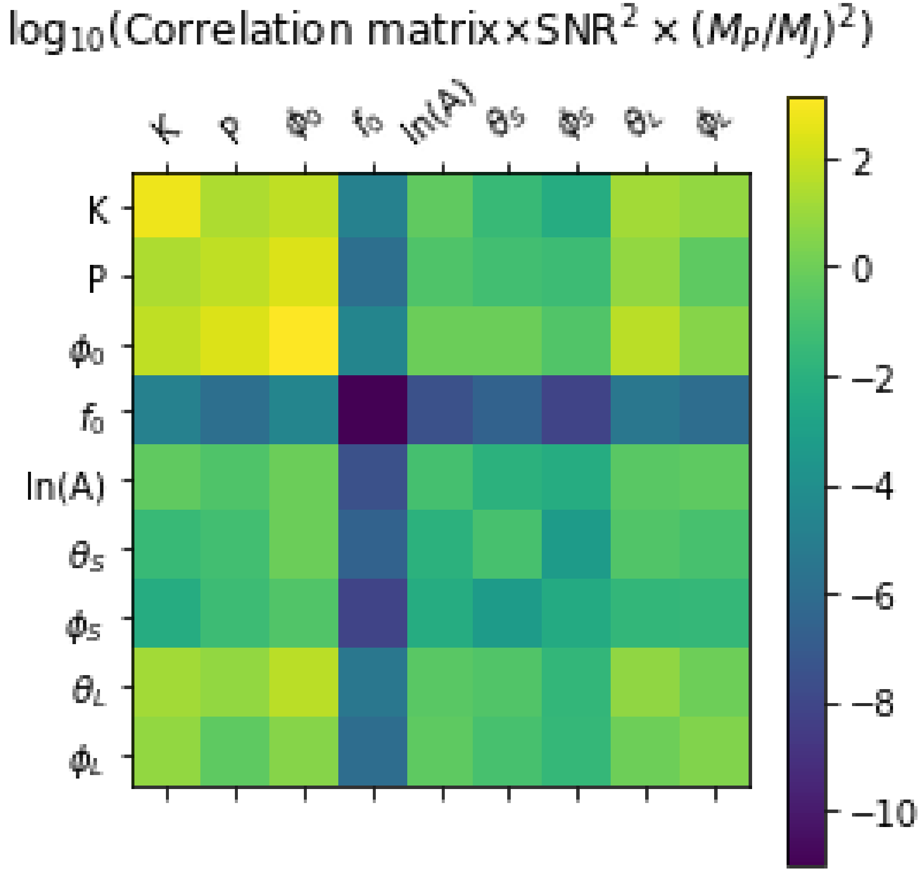
My trick for quicker computations: Set epsrel at 1.49e-2 and then set epsabs for off-diagonal integrals as multiple of geometric mean $\sqrt{\Gamma_{ii} \times \Gamma_{jj}}$ ×1.49e-3 -> don’t waste time on irrelevant integrals

Findings

Looking at the correlation matrix for the 9 parameters of interest:

I again used the same position as Cutler, f_0 = 10 mHz and P = 2 yr

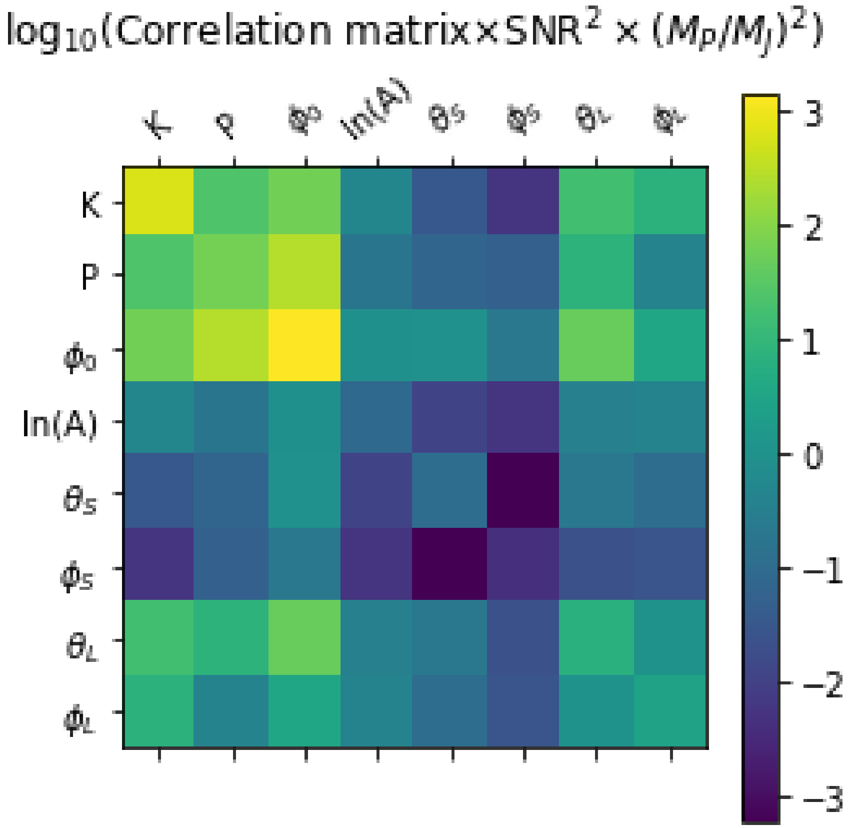
We can easily discard the f_0 fit, as the determination of the GW frequency isn‘t very problematic



Findings

Looking at the correlation matrix for the 8 parameters of interest:

I again used the same position as Cutler, f_0 = 10 mHz and P = 2 yr

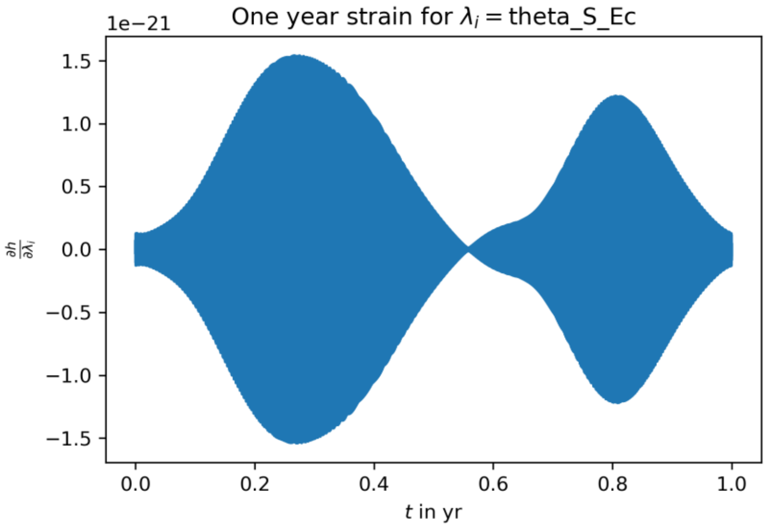
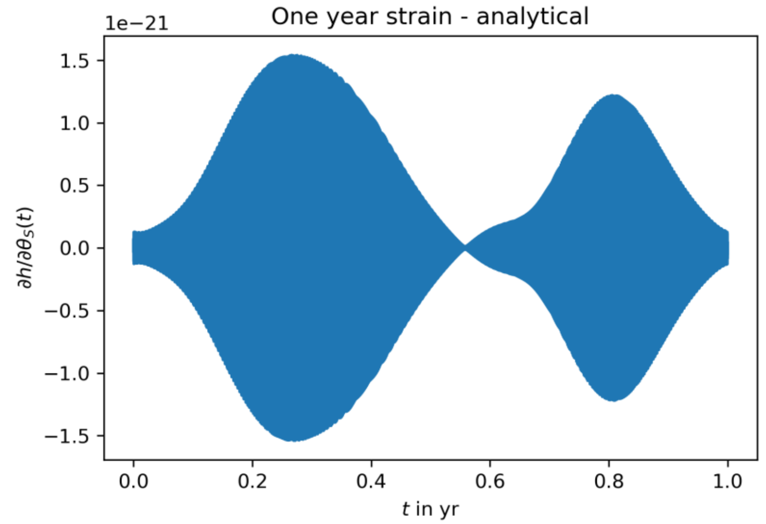
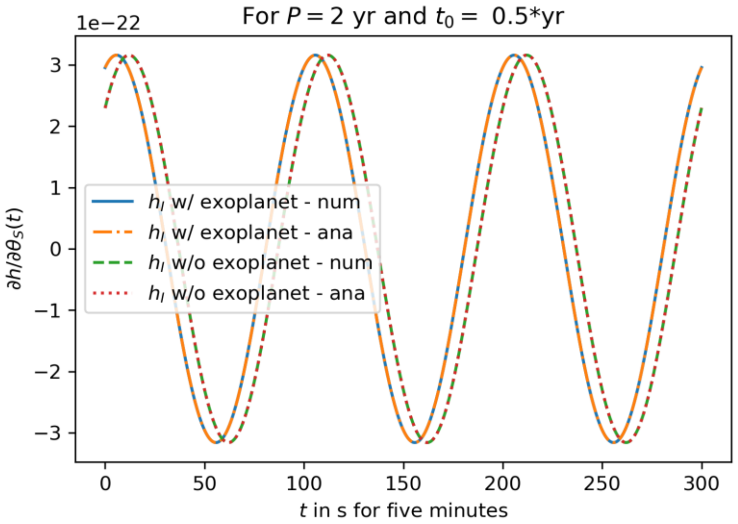
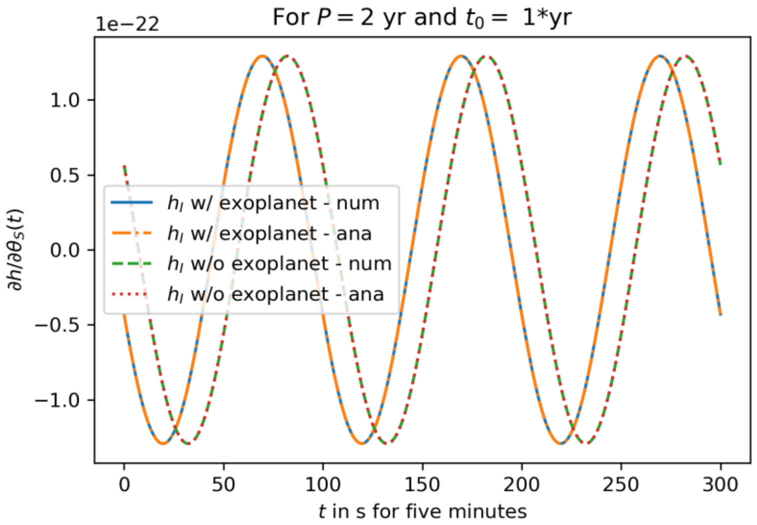


After killing f_0:

An argument can also be made to discard the φ_S fit

```
(np.sqrt(3)*(-(np.cos(self.theta_L)*np.cos(self.phi_S_Ec)*np.sin(self.theta_S_Ec)) + np.cos(self.theta_S_Ec)*np.cos(self.phi_L)*np.sin(self.theta_L))*np.sin(t*omega_E))/2.,
np.cos(self.theta_L)/2. - (np.sqrt(3)*np.cos(self.phi_L - t*omega_E)*np.sin(self.theta_L))/2. -
(np.cos(self.theta_S_Ec)/2. - (np.sqrt(3)*np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))/2.)*(np.cos(self.theta_S_Ec)*np.cos(self.theta_L) + np.cos(self.phi_S_Ec - self.phi_L)*np.sin(self.theta_S_E
np.sin(2*(t*omega_E + ArcTan(-2*np.sin(self.theta_S_Ec)*np.sin(self.phi_S_Ec - t*omega_E),np.sqrt(3)*np.cos(self.theta_S_Ec) + np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec)))) -
(1 + (np.cos(self.theta_S_Ec)/2. - (np.sqrt(3)*np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))/2.)**2)*
((-2*np.sin(self.theta_S_Ec)*(np.cos(self.theta_S_Ec)*np.cos(self.phi_S_Ec - t*omega_E) - np.sqrt(3)*np.sin(self.theta_S_Ec))*np.sin(self.phi_S_Ec - t*omega_E))/
((np.sqrt(3)*np.cos(self.theta_S_Ec) + np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))**2 + 4*np.sin(self.theta_S_Ec)**2*np.sin(self.phi_S_Ec - t*omega_E)**2) -
(2*np.cos(self.theta_S_Ec)*(-(np.sqrt(3)*np.cos(self.theta_S_Ec)) - np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))*np.sin(self.phi_S_Ec - t*omega_E))/
((np.sqrt(3)*np.cos(self.theta_S_Ec) + np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))**2 + 4*np.sin(self.theta_S_Ec)**2*np.sin(self.phi_S_Ec - t*omega_E)**2))*
np.sin(2*ArcTan(-0.5*(np.sin(self.theta_S_Ec)*np.sin(self.theta_L)*np.sin(self.phi_S_Ec - self.phi_L)) -
(np.sqrt(3)*np.cos(t*omega_E)*(np.cos(self.theta_L)*np.sin(self.theta_S_Ec)*np.sin(self.phi_S_Ec) - np.cos(self.theta_S_Ec)*np.sin(self.theta_L)*np.sin(self.phi_L)))/2. -
(np.sqrt(3)*(-(np.cos(self.theta_L)*np.cos(self.phi_S_Ec)*np.sin(self.theta_S_Ec)) + np.cos(self.theta_S_Ec)*np.cos(self.phi_L)*np.sin(self.theta_L))*np.sin(t*omega_E))/2.,
np.cos(self.theta_L)/2. - (np.sqrt(3)*np.cos(self.phi_L - t*omega_E)*np.sin(self.theta_L))/2. -
(np.cos(self.theta_S_Ec)/2. - (np.sqrt(3)*np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))/2.)*(np.cos(self.theta_S_Ec)*np.cos(self.theta_L) + np.cos(self.phi_S_Ec - self.phi_L)*np.sin(self.theta_S_E
np.sin(2*(t*omega_E + ArcTan(-2*np.sin(self.theta_S_Ec)*np.sin(self.phi_S_Ec - t*omega_E),np.sqrt(3)*np.cos(self.theta_S_Ec) + np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec)))))))/
(4.*np.sqrt(self.a0**2*(np.cos(self.theta_S_Ec)*np.cos(self.theta_L) + np.cos(self.phi_S_Ec - self.phi_L)*np.sin(self.theta_S_Ec)*np.sin(self.theta_L))**2*
((np.cos(2*(t*omega_E + ArcTan(-2*np.sin(self.theta_S_Ec)*np.sin(self.phi_S_Ec - t*omega_E),np.sqrt(3)*np.cos(self.theta_S_Ec) + np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))))*
(1 + (np.cos(self.theta_S_Ec)/2. - (np.sqrt(3)*np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))/2.)**2)*
np.sin(2*ArcTan(-0.5*(np.sin(self.theta_S_Ec)*np.sin(self.theta_L)*np.sin(self.phi_S_Ec - self.phi_L)) -
(np.sqrt(3)*np.cos(t*omega_E)*(np.cos(self.theta_L)*np.sin(self.theta_S_Ec)*np.sin(self.phi_S_Ec) - np.cos(self.theta_S_Ec)*np.sin(self.theta_L)*np.sin(self.phi_L)))/2. -
(np.sqrt(3)*(-(np.cos(self.theta_L)*np.cos(self.phi_S_Ec)*np.sin(self.theta_S_Ec)) + np.cos(self.theta_S_Ec)*np.cos(self.phi_L)*np.sin(self.theta_L))*np.sin(t*omega_E))/2.,
np.cos(self.theta_L)/2. - (np.sqrt(3)*np.cos(self.phi_L - t*omega_E)*np.sin(self.theta_L))/2. -
(np.cos(self.theta_S_Ec)/2. - (np.sqrt(3)*np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))/2.)*(np.cos(self.theta_S_Ec)*np.cos(self.theta_L) + np.cos(self.phi_S_Ec - self.phi_L)*np.sin(self.theta_S_E
np.cos(2*ArcTan(-0.5*(np.sin(self.theta_S_Ec)*np.sin(self.theta_L)*np.sin(self.phi_S_Ec - self.phi_L)) -
(np.sqrt(3)*np.cos(t*omega_E)*(np.cos(self.theta_L)*np.sin(self.theta_S_Ec)*np.sin(self.phi_S_Ec) - np.cos(self.theta_S_Ec)*np.sin(self.theta_L)*np.sin(self.phi_L)))/2. -
(np.sqrt(3)*(-(np.cos(self.theta_L)*np.cos(self.phi_S_Ec)*np.sin(self.theta_S_Ec)) + np.cos(self.theta_S_Ec)*np.cos(self.phi_L)*np.sin(self.theta_L))*np.sin(t*omega_E))/2.,
np.cos(self.theta_L)/2. - (np.sqrt(3)*np.cos(self.phi_L - t*omega_E)*np.sin(self.theta_L))/2. -
(np.cos(self.theta_S_Ec)/2. - (np.sqrt(3)*np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))/2.)*(np.cos(self.theta_S_Ec)*np.cos(self.theta_L) + np.cos(self.phi_S_Ec - self.phi_L)*np.sin(self.theta_S_E
(np.cos(self.theta_S_Ec)/2. - (np.sqrt(3)*np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))/2.)*
np.sin(2*(t*omega_E + ArcTan(-2*np.sin(self.theta_S_Ec)*np.sin(self.phi_S_Ec - t*omega_E),np.sqrt(3)*np.cos(self.theta_S_Ec) + np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec)))))**2 +
(self.a0**2*(1 + (np.cos(self.theta_S_Ec)*np.cos(self.theta_L) + np.cos(self.phi_S_Ec - self.phi_L)*np.sin(self.theta_S_Ec)*np.sin(self.theta_L))**2)**2*
((np.cos(2*ArcTan(-0.5*(np.sin(self.theta_S_Ec)*np.sin(self.theta_L)*np.sin(self.phi_S_Ec - self.phi_L)) -
(np.sqrt(3)*np.cos(t*omega_E)*(np.cos(self.theta_L)*np.sin(self.theta_S_Ec)*np.sin(self.phi_S_Ec) - np.cos(self.theta_S_Ec)*np.sin(self.theta_L)*np.sin(self.phi_L)))/2. -
(np.sqrt(3)*(-(np.cos(self.theta_L)*np.cos(self.phi_S_Ec)*np.sin(self.theta_S_Ec)) + np.cos(self.theta_S_Ec)*np.cos(self.phi_L)*np.sin(self.theta_L))*np.sin(t*omega_E))/2.,
np.cos(self.theta_L)/2. - (np.sqrt(3)*np.cos(self.phi_L - t*omega_E)*np.sin(self.theta_L))/2. -
(np.cos(self.theta_S_Ec)/2. - (np.sqrt(3)*np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))/2.)*(np.cos(self.theta_S_Ec)*np.cos(self.theta_L) + np.cos(self.phi_S_Ec - self.phi_L)*np.sin(self.theta_S_E
np.cos(2*(t*omega_E + ArcTan(-2*np.sin(self.theta_S_Ec)*np.sin(self.phi_S_Ec - t*omega_E),np.sqrt(3)*np.cos(self.theta_S_Ec) + np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))))*
(1 + (np.cos(self.theta_S_Ec)/2. - (np.sqrt(3)*np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))/2.)**2))/2. -
(np.cos(self.theta_S_Ec)/2. - (np.sqrt(3)*np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))/2.)*
np.sin(2*ArcTan(-0.5*(np.sin(self.theta_S_Ec)*np.sin(self.theta_L)*np.sin(self.phi_S_Ec - self.phi_L)) -
(np.sqrt(3)*np.cos(t*omega_E)*(np.cos(self.theta_L)*np.sin(self.theta_S_Ec)*np.sin(self.phi_S_Ec) - np.cos(self.theta_S_Ec)*np.sin(self.theta_L)*np.sin(self.phi_L)))/2. -
(np.sqrt(3)*(-(np.cos(self.theta_L)*np.cos(self.phi_S_Ec)*np.sin(self.theta_S_Ec)) + np.cos(self.theta_S_Ec)*np.cos(self.phi_L)*np.sin(self.theta_L))*np.sin(t*omega_E))/2.,
np.cos(self.theta_L)/2. - (np.sqrt(3)*np.cos(self.phi_L - t*omega_E)*np.sin(self.theta_L))/2. -
(np.cos(self.theta_S_Ec)/2. - (np.sqrt(3)*np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))/2.)*(np.cos(self.theta_S_Ec)*np.cos(self.theta_L) + np.cos(self.phi_S_Ec - self.phi_L)*np.sin(self.theta_S_E
np.sin(2*(t*omega_E + ArcTan(-2*np.sin(self.theta_S_Ec)*np.sin(self.phi_S_Ec - t*omega_E),np.sqrt(3)*np.cos(self.theta_S_Ec) + np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec)))))**2)/4.)) -
(np.sqrt(3)*np.sqrt(self.a0**2*(np.cos(self.theta_S_Ec)*np.cos(self.theta_L) + np.cos(self.phi_S_Ec - self.phi_L)*np.sin(self.theta_S_Ec)*np.sin(self.theta_L))**2*
((np.cos(2*(t*omega_E + ArcTan(-2*np.sin(self.theta_S_Ec)*np.sin(self.phi_S_Ec - t*omega_E),np.sqrt(3)*np.cos(self.theta_S_Ec) + np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))))*
(1 + (np.cos(self.theta_S_Ec)/2. - (np.sqrt(3)*np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))/2.)**2)*
np.sin(2*ArcTan(-0.5*(np.sin(self.theta_S_Ec)*np.sin(self.theta_L)*np.sin(self.phi_S_Ec - self.phi_L)) -
(np.sqrt(3)*np.cos(t*omega_E)*(np.cos(self.theta_L)*np.sin(self.theta_S_Ec)*np.sin(self.phi_S_Ec) - np.cos(self.theta_S_Ec)*np.sin(self.theta_L)*np.sin(self.phi_L)))/2. -
(np.sqrt(3)*(-(np.cos(self.theta_L)*np.cos(self.phi_S_Ec)*np.sin(self.theta_S_Ec)) + np.cos(self.theta_S_Ec)*np.cos(self.phi_L)*np.sin(self.theta_L))*np.sin(t*omega_E))/2.,
np.cos(self.theta_L)/2. - (np.sqrt(3)*np.cos(self.phi_L - t*omega_E)*np.sin(self.theta_L))/2. -
(np.cos(self.theta_S_Ec)/2. - (np.sqrt(3)*np.cos(self.phi_S_Ec - t*omega_E)*np.sin(self.theta_S_Ec))/2.)*(np.cos(self.theta_S_Ec)*np.cos(self.theta_L) + np.cos(self.phi_S_Ec - self.phi_L)*np.sin(self.theta_S_E
np.cos(2*ArcTan(-0.5*(np.sin(self.theta_S_Ec)*np.sin(self.theta_L)*np.sin(self.phi_S_Ec - self.phi_L)) -
(np.sqrt(3)*np.cos(t*omega_E)*(np.cos(self.theta_L)*np.sin(self.theta_S_Ec)*np.sin(self.phi_S_Ec) - np.cos(self.theta_S_Ec)*np.sin(self.theta_L)*np.sin(self.phi_L)))/2. -
```

Note that the analytic derivative does exist, but it is a) too long for pretty code – you see a snippet of the 657 lines of terms for $\frac{\partial h_\alpha}{\partial \theta_S}(t)$ in python code, see delh_delthetaS.txt on Git and…

	Numerical derivative	Analytic derivative
Shape of the Amplitude		
Shape of the frequency (1 hour after 0.3*yr)		

b) results in the same function anyways, plus…

And sadly life would be too easy if we could integrate it analytically :c

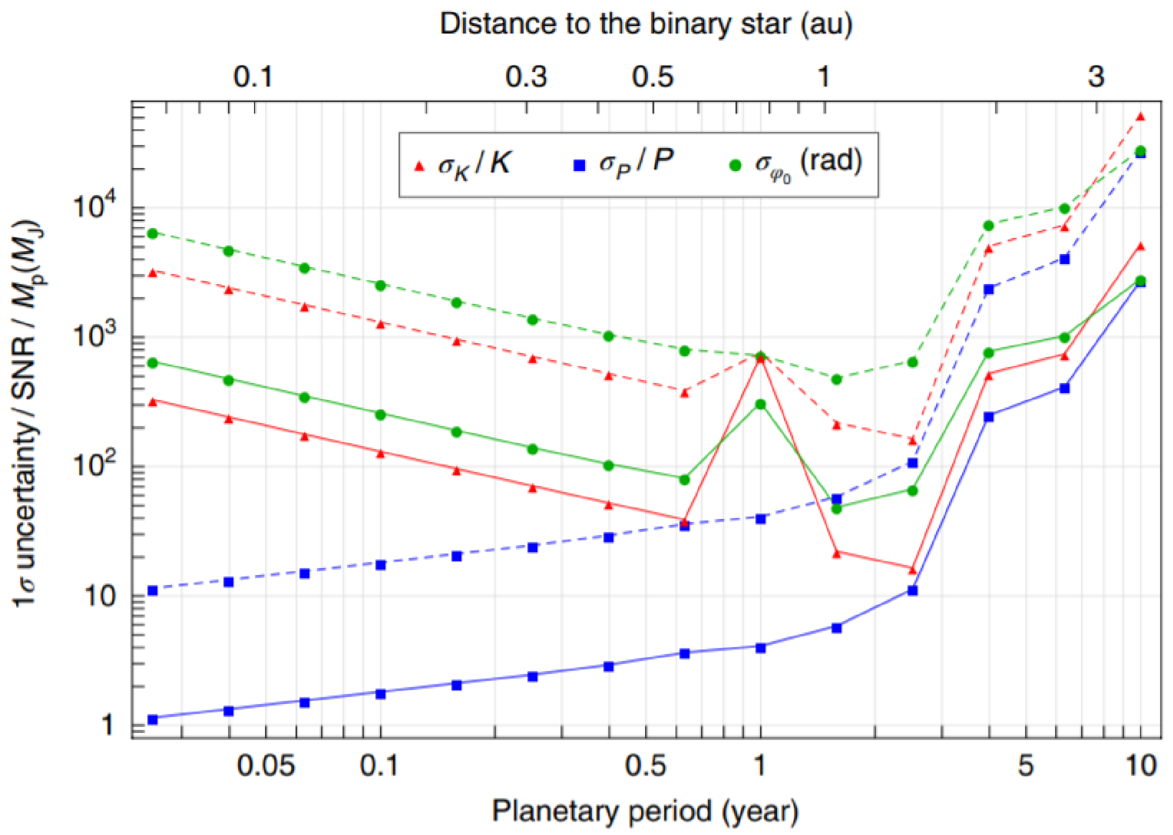
```
dhdθS = D[h[θL, θEc, ϕL, ϕEc, t], θEc]
        leite ab
```

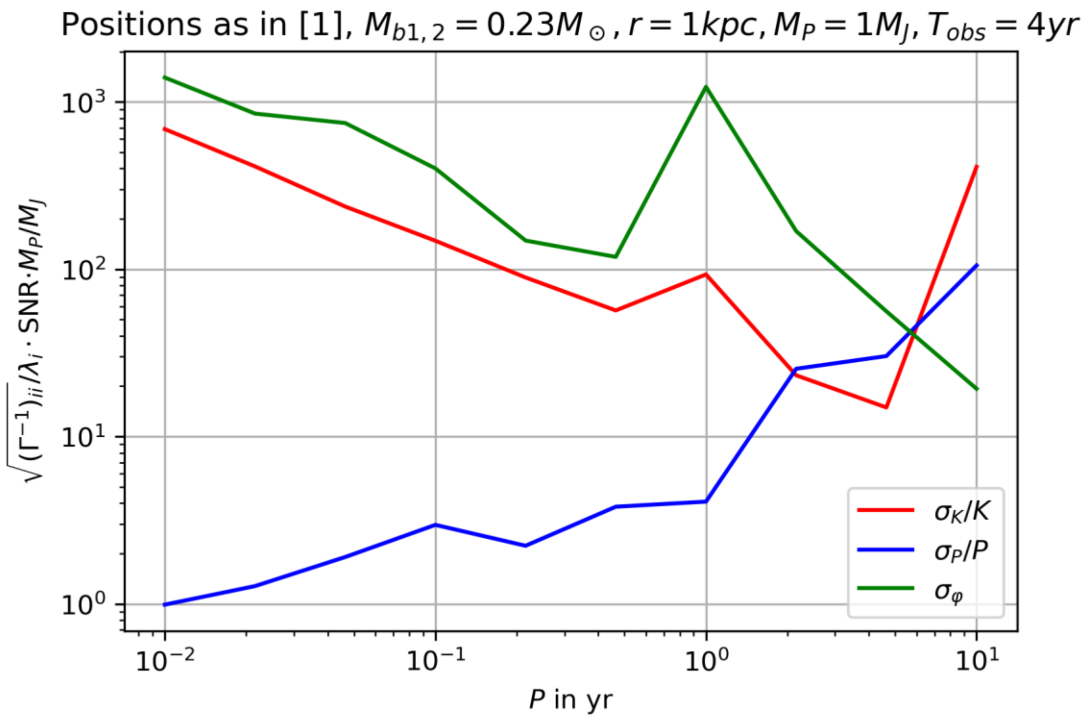
```
(√3 Cos[ψobs + ArcTan[1/2 A0 (1 + «1»^2) × (1/2 Cos[2 ArcTan[«1», «1»]] Cos[2 (t ω + ArcTan[«1»])])
        (1 + (Cos[θEc]/2 − 1/2 √3 Cos[«1»] Sin[θEc])^2) − «1»), «1»] + (2 AU fGW π Cos[ϕEc−t ω] Sin[θEc])/c] («1»)) /
  (4 √(A0^2 (Cos[θEc] Cos[θL] + Cos[ϕEc − ϕL] Sin[θEc] Sin[θL])^2 (1/2 Cos[2 (t ω + ArcTan[«1»])] (1 + «1»^2) Sin[2 «1»] + «1»)^2 +
      1/4 A0^2 (1 + «1»)^2 («1» − «1»)^2)) −
1/2 √3 √«1» («1») Sin[ψobs + ArcTan[1/2 «3», «1»] + «1»/c]

large output | show less | show more | show all | set size limit...
```

c) neither `Simplify` nor `Integrate` will return

Computing the uncertainties

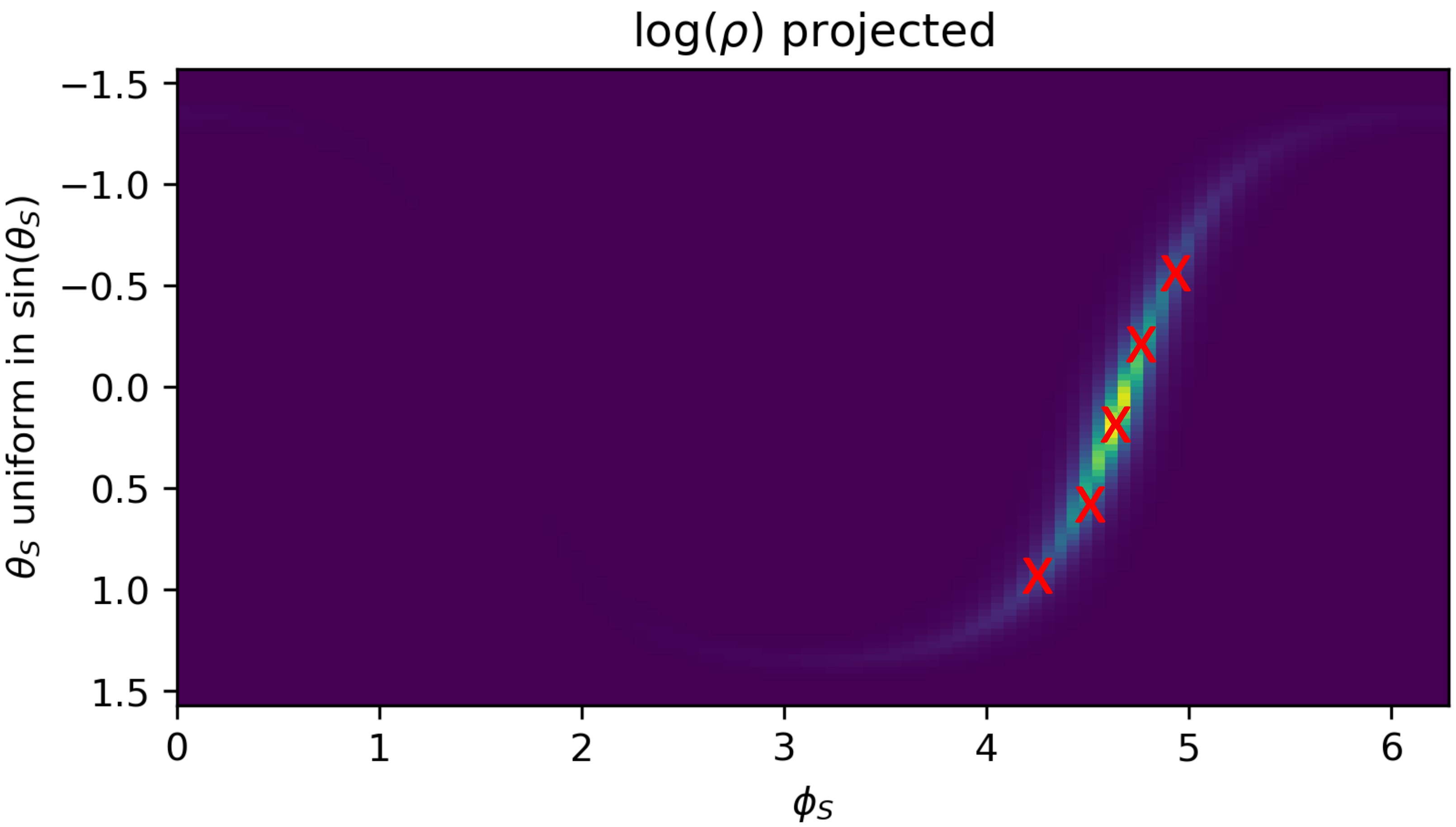




Sadly it still doesn‘t work, even with higher precision in integration…
At this point I‘m finally going to stop though

How will the position be a factor for the uncertainty?

Look for a change in uncertainties over relevant region in milky way



Constraining the number of exoplanets

c.f. [2]:

DETECTIONS (4 years)

	A) $\mathcal{U}_a$ (0.1-200 au)		B) log $\mathcal{U}_a$ (0.1-200 au)		C) logNormal$_a$ (0.1-200 au)		D) $a^{-0.61}$ (0.1-200 au)	
	CBPs	BDs	CBPs	BDs	CBPs	BDs	CBPs	BDs
1) $\mathcal{U}_M$ (1M$_\oplus$ - 0.08 M$_\odot$)	**3 (0.011%)**	79 (0.302%)	**83 (0.317%)**	**2218 (8.482%)**	18 (0.069%)	503 (1.924%)	28 (0.107%)	820 (3.136%)
2) $M^{-1.31}$	6 (0.023%)	**14 (0.054%)**	30 (0.115%)	316 (1.209%)	5 (0.019%)	85 (0.325%)	13 (0.050%)	131 (0.501%)

How did they do it? Simulation:

1. Generate primary from Kroupa imf, range [.95, 10]$\times M_\odot$
2. Generate partner from uniform mass ratio [0,1] and log-flat seperation, thermal eccentricity distribution
3. Place binaries in MW according to star formation rate times density
4. Add planet with $f_{\mathrm{CBP}} = 0.5$ and mass/separation distribution as in table

How well will icegiant constrain exoplanets?

40 day measurement over ten time slots:
Question: What are the distances T_2?

I‘m going to take the direction towards Neptune via JPL HORIZONS for simplicity and then multiply my results by $\sqrt{2}$

Then I can repeat the whole exercise with

$$h(t) = \frac{\mu - 1}{2}\Psi(t) - \mu\Psi\left(t - \frac{\mu + 1}{2}T_2\right) + \frac{\mu + 1}{2}\Psi(t - T_2)$$

Next milestones

- ✓Understand the fluctuations a bit better and potentially fix them
- ✓Do the calculations in Mathematica
- ✓Add $\ln(A)$, f_1 fit to see actually the same plot as Tamanini
- ✓Look at the parameter space for positions/angular momentum for the 25‘000 potential DWDs with SNR > 7 and calculate positional dependance of the planetary parameters -> we want a function:

```
rel_uncertainty(pos, M, sep)
```

-> Unrealistic to compute a good grid in limited time

- Assuming a prior on the DWD parameters and planetary parameters (mass, inclination, separation), we can then take the integral as in [4], constraining f_{CBP} the fraction of circumbinary partners given N_{bin} detections via Bayesian inference

Next milestones

$$N_{\mathrm{bin}} = \int_0^z \int_0^\infty \int_{\nu_{\mathrm{ISCO}}}^{\nu} \frac{\mathrm{d}V}{\mathrm{d}z} \frac{\mathrm{d}\mathcal{R}}{\mathrm{d}M} \mathcal{D}(M, q, \nu, z) \frac{\mathrm{d}\nu}{\dot{\nu}_{\mathrm{GW}}} \mathrm{d}M \mathrm{d}z, \qquad (11)$$

$$\text{with} \quad \mathcal{D}(M, q, \nu, z) \equiv \mathcal{H}\left[h_c(M, q, \nu, z) - \rho_c h_n(\nu)\right], \qquad (12)$$

-> already done by Danielsky et al. [2]

- Then repeat the exercise with the strain and signal to noise of the IceGiant mission:

$$y_2^{\mathrm{GW}}(t) = \frac{\mu - 1}{2} \bar{\Psi}(t) - \mu \bar{\Psi}\left(t - \frac{\mu + 1}{2} T_2\right) + \frac{\mu + 1}{2} \bar{\Psi}(t - T_2), \qquad (1)$$

  to see if it could see most promising exoplanet candidates/Jupiter-like planets

- Combine measurements of IceGiant and LISA -> just add Fisher information prior to inversion ☺

References

[1] Tamanini, N., Danielski, C. (2019). The gravitational-wave detection of exoplanets orbiting white dwarf binaries using LISA. *Nat Astron 3*, 858–866. https://doi.org/10.1038/s41550-019-0807-y

[2] Danielski, C., Korol, V., Tamanini, N., & Rossi, E.M. (2019). Circumbinary exoplanets and brown dwarfs with the Laser Interferometer Space Antenna. *Astronomy and Astrophysics, 632.*

[3] Cutler, C. (1998). Angular resolution of the LISA gravitational wave detector. *Physical Review D, 57*, 7089-7102.

[4] Soyuer, D., Zwick, L., D'Orazio, D., Saha, P. (2021). Searching for gravitational waves via Doppler tracking by future missions to Uranus and Neptune. *MNRAS: Letters*, 503, 1, L73-79. https://doi.org/10.1093/mnrasl/slab025

[5] Maggiore, M. (2008). *Gravitational Waves Volume 1: Theory and Experiments*. Oxford University Press