Detecting Exoplanets and recovering Parameters

A short recap of my work in week 5

What did Tamanini and Danielski do?

1. Derive $\frac{\partial h_I}{\partial \lambda}(t)$ for the 9 par.s of interest:

   $K, P, \varphi_0\,, f_0, \ln(A)\,, \theta_S, \varphi_S, \theta_L, \varphi_L,$

2. Compute numerically the integral

$$\Gamma_{ij} = \frac{2}{S_n(f_0)} \sum_{\alpha=I,II} \int_0^{T_0} dt\, \frac{\partial h_\alpha}{\partial \lambda_i}(t) \frac{\partial h_\alpha}{\partial \lambda_j}(t) \;\; \Rightarrow \;\; \sigma_i^2 = \Sigma_{ii} = \left(\Gamma^{-1}\right)_{ii}$$

Regarding the position and angular momentum

The position and angular momentum of the binary source influences the strain non-trivially, see [3]:

$$h_I(t) = \frac{\sqrt{3}}{2} A_+ F_I^+(\theta_S, \phi_S, \psi_S) \cos(2\pi f t) + \frac{\sqrt{3}}{2} A_\times F_I^\times(\theta_S, \phi_S, \psi_S) \sin(2\pi f t) \,, \quad (3.11)$$

$$F_I^+(\theta_S, \phi_S, \psi_S) = \frac{1}{2}(1 + \cos^2\theta_S) \cos 2\phi_S \cos 2\psi_S - \cos\theta_S \sin 2\phi_S \sin 2\psi_S \,, \quad (3.12a)$$

$$F_I^\times(\theta_S, \phi_S, \psi_S) = \frac{1}{2}(1 + \cos^2\theta_S) \cos 2\phi_S \sin 2\psi_S + \cos\theta_S \sin 2\phi_S \cos 2\psi_S \,. \quad (3.12b)$$

$$\cos\theta_S(t) = \frac{1}{2}\cos\bar\theta_S - \frac{\sqrt{3}}{2}\sin\bar\theta_S \cos(\bar\phi(t) - \bar\phi_S) \quad (3.16)$$

$$\phi_S(t) = \alpha_1 + \pi/12 - \tan^{-1}\left[\frac{\sqrt{3}\cos\bar\theta_S + \sin\bar\theta_S\cos(\bar\phi(t) - \bar\phi_S)}{2\sin\bar\theta_S \cos(\bar\phi(t) - \bar\phi_S)}\right] \quad (3.17)$$

$$\tan\psi_S(t) = \left(\hat L^a z_a - \hat L^a n_a z^b n_b\right) \Big/ \left(\epsilon_{abc} n^a \hat L^b z^c\right) \quad (3.19)$$

$$\hat L^a z_a = \frac{1}{2}\cos\bar\theta_L - \frac{\sqrt{3}}{2}\sin\bar\theta_L \cos\left(\bar\phi(t) - \bar\phi_L\right) \quad (3.20)$$

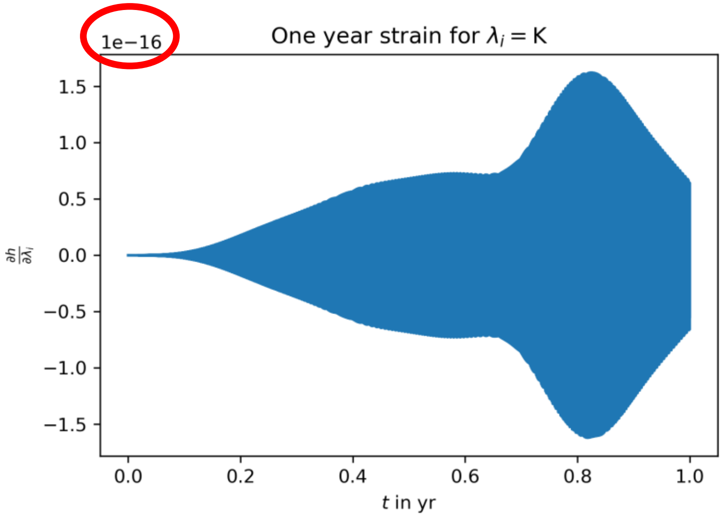
$$\hat L^a n_a = \cos\bar\theta_L \cos\bar\theta_S + \sin\bar\theta_L \sin\bar\theta_S \cos\left(\bar\phi_L - \bar\phi_S\right) \quad (3.21)$$

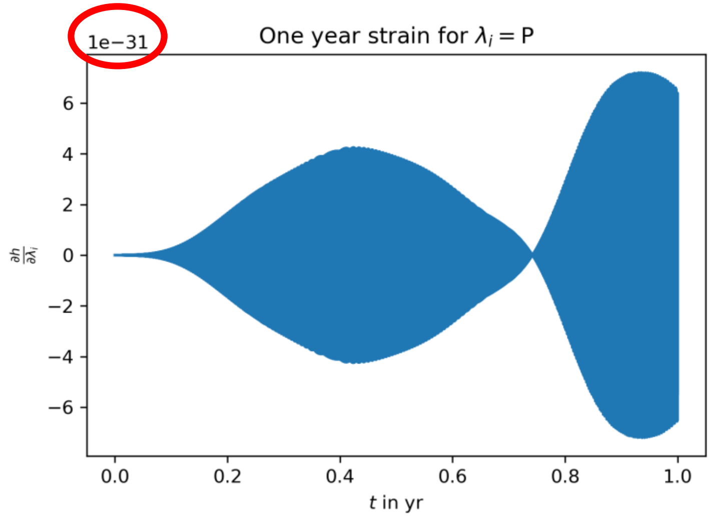
$$\epsilon_{abc} n^a \hat L^b z^c = \frac{1}{2}\sin\bar\theta_L \sin\bar\theta_S \sin\left(\bar\phi_L - \bar\phi_S\right) - \frac{\sqrt{3}}{2}\cos\bar\phi(t)\left(\cos\bar\theta_L \sin\bar\theta_S \sin\bar\phi_S - \cos\bar\theta_S \sin\bar\theta_L \sin\bar\phi_L\right) - \frac{\sqrt{3}}{2}\sin\bar\phi(t)\left(\cos\bar\theta_S \sin\bar\theta_L \cos\bar\phi_L - \cos\bar\theta_L \sin\bar\theta_S \cos\bar\phi_S\right) . \quad (3.22)$$

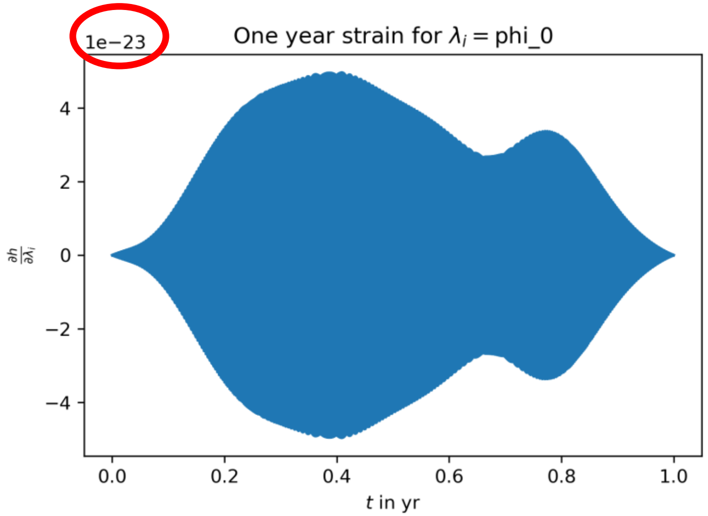
…so we do what Cutler proposes

Thus to evaluate the Fisher matrix (4.4), we need the derivatives of $A_\alpha(t)$ and $\chi_\alpha(t)$ with respect to the seven physical parameters $\ln \mathcal{A}$, φ_0, f_0, $\bar{\theta}_S$, $\bar{\phi}_S$, $\bar{\theta}_L$, $\bar{\phi}_L$. Clearly one might straightforwardly use the chain rule with Eqs. (3.15) and (3.32) to determine the partial derivatives of $A_\alpha(t)$ and $\chi_\alpha(t)$ with respect to the four angles $\bar{\theta}_S$, $\bar{\phi}_S$, $\bar{\theta}_L$,and $\bar{\phi}_L$, though the final expressions would be cumbersome. In our calculation, we preferred simply to take these derivatives numerically. The remaining partial derivatives are:

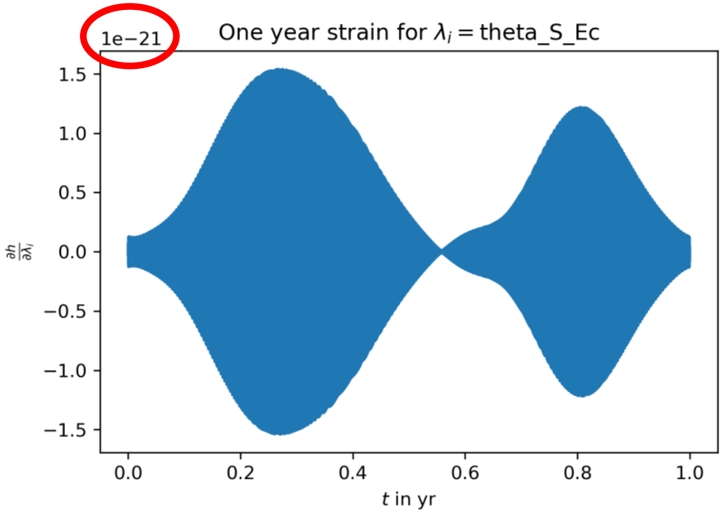
In our case we initialise a second class instance of our class binary(position + 1e-6) and compare it via (binary(position + 1e-6) – binary(position)) / 1e-6 (same as scipy.misc.derivative)

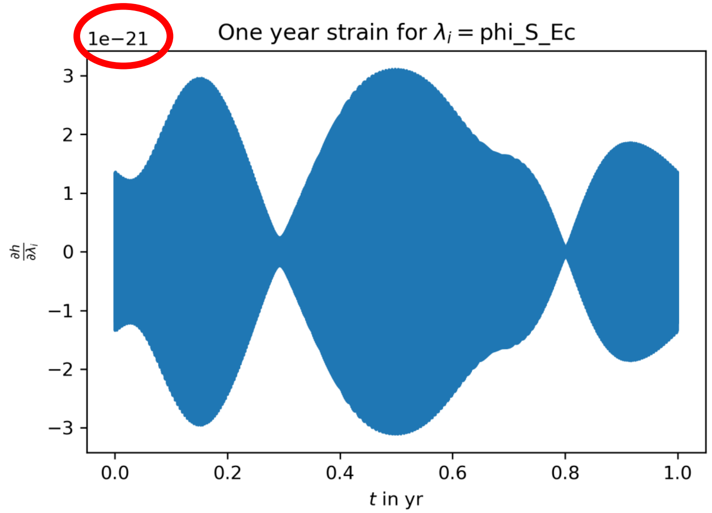


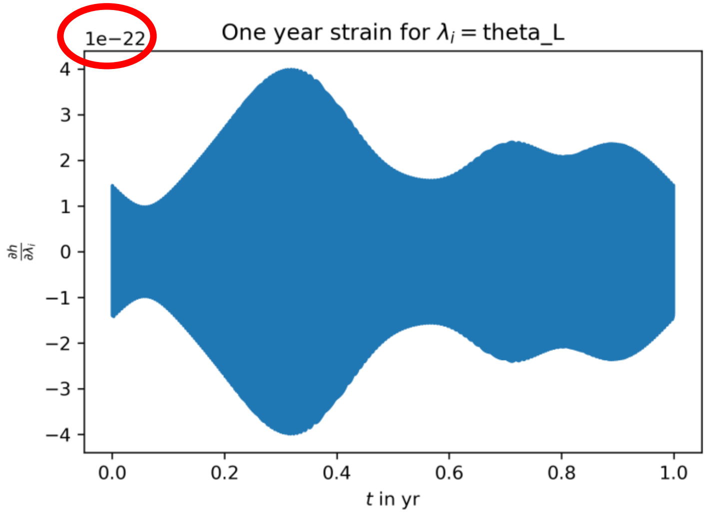


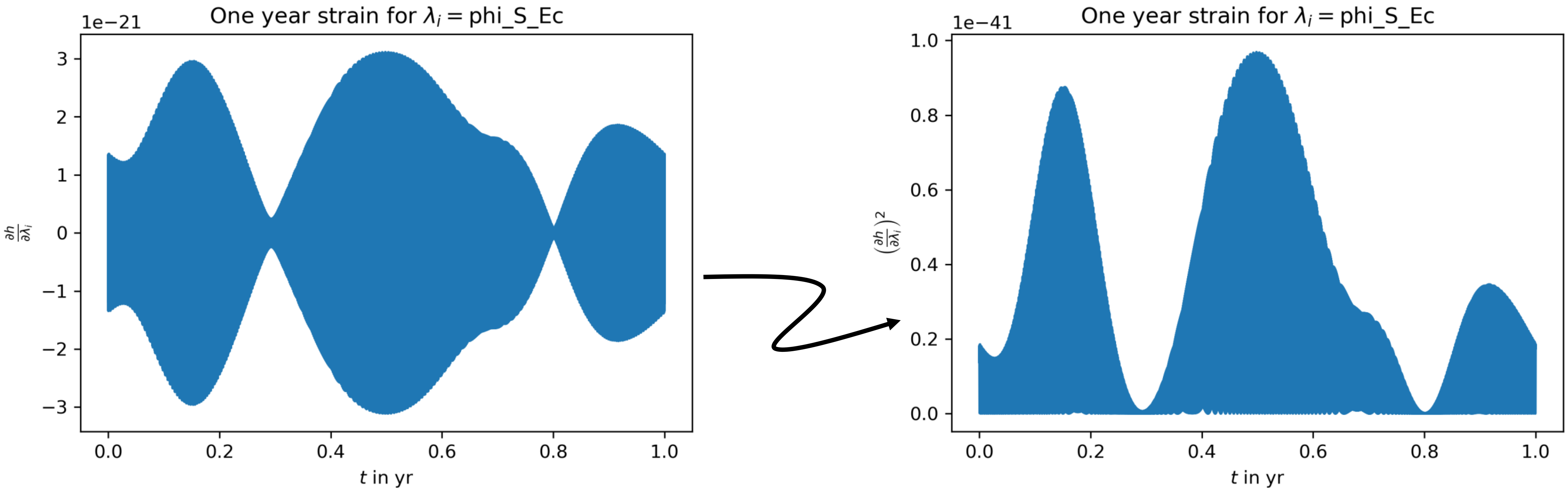


The derivatives $\frac{\partial h_\alpha}{\partial \lambda_i}$ look good, compare the analytic derivatives of the exoplanet par‘s (above) with the numerical derivatives of the position par‘s (below) – same structure: envelope * oscillating function









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 $\Gamma_{ij} \propto \left[\int_0^{T_0} dt \, \frac{\partial h_\alpha}{\partial \lambda_i}(t) \frac{\partial h_\alpha}{\partial \lambda_j}(t)\right]_{ij}$

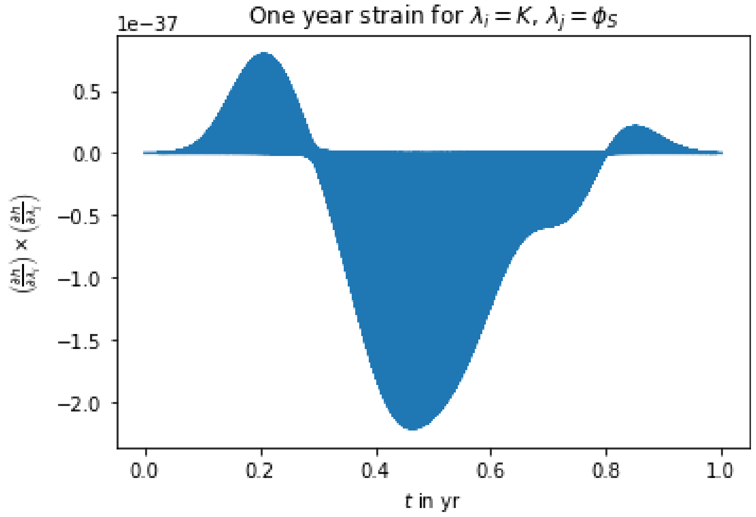
We know, diagonal elements Γ_{ii} will be in phase, so the integrand will go like $A^2 \sin^2 \omega t$ and we find for diagonal elements a result which goes roughly like $\Gamma_{ii} \propto \frac{1}{2} \int_0^{T_0} A^2(t) dt + O\left(\frac{A_0}{4\omega}\right) \approx \frac{1}{2} {A_0}^2 T_0 > 0$ with 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 $A_i A_j \sin \omega_i t \sin \omega_j t = \frac{A_i A_j}{2}(\cos(\omega_i - \omega_j)\, t \cos(\omega_i + \omega_j)\, t)$ and if they are uncorrelated, we'll find over long observations $\Gamma_{ij} \approx 0$

<u>**Question:**</u> 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

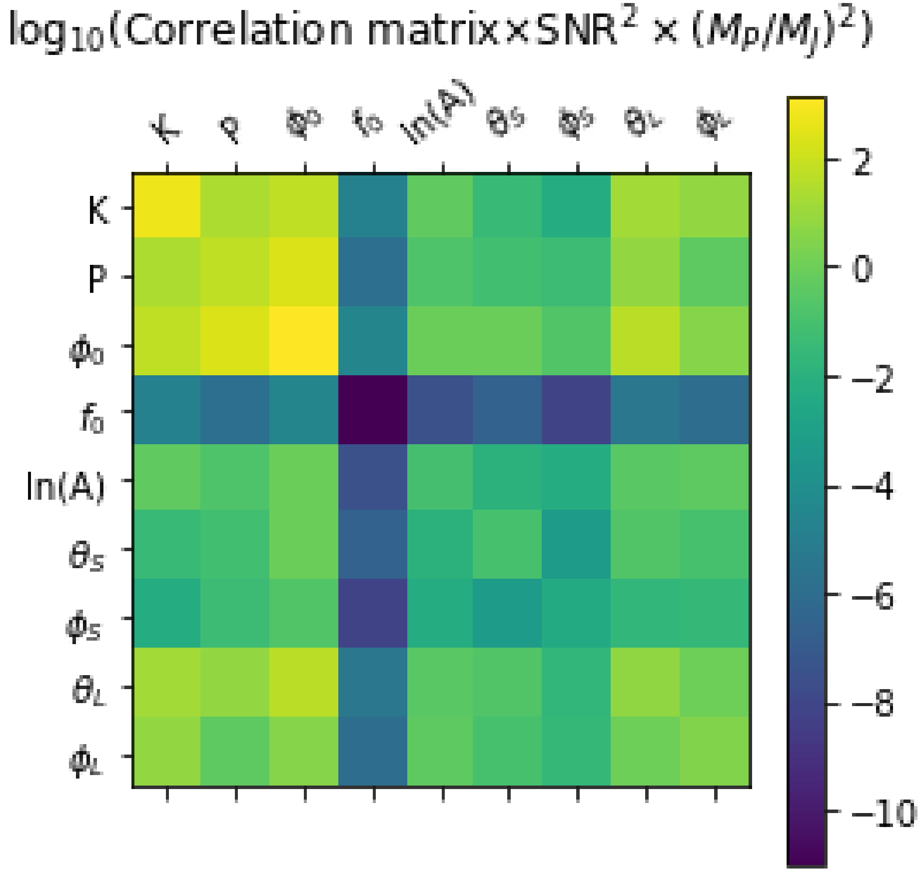
My trick for quicker computations: Set epsrel at 1.49e-2 and then set epsabs for off-diagonal integrals as multiple of geometric mean $\sqrt{\Gamma_{ii} \times \Gamma_{jj}}$ ×1.49e-2 -> don’t waste time on irrelevant integrals

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



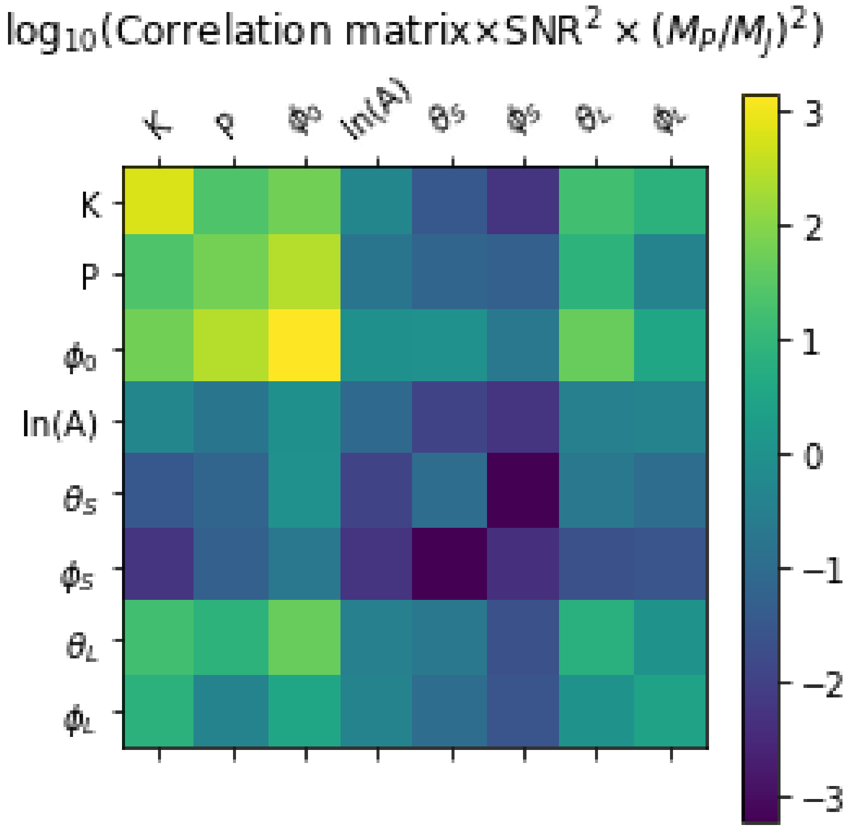
Findings

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

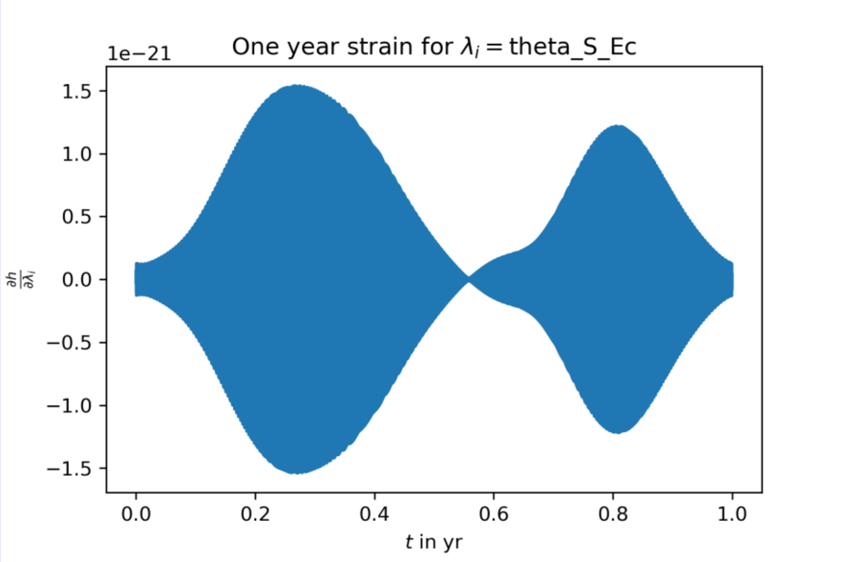
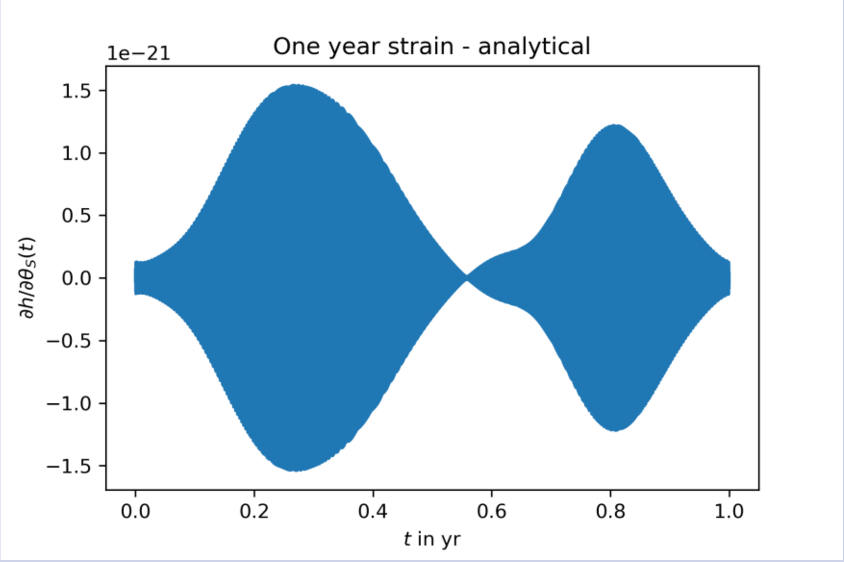
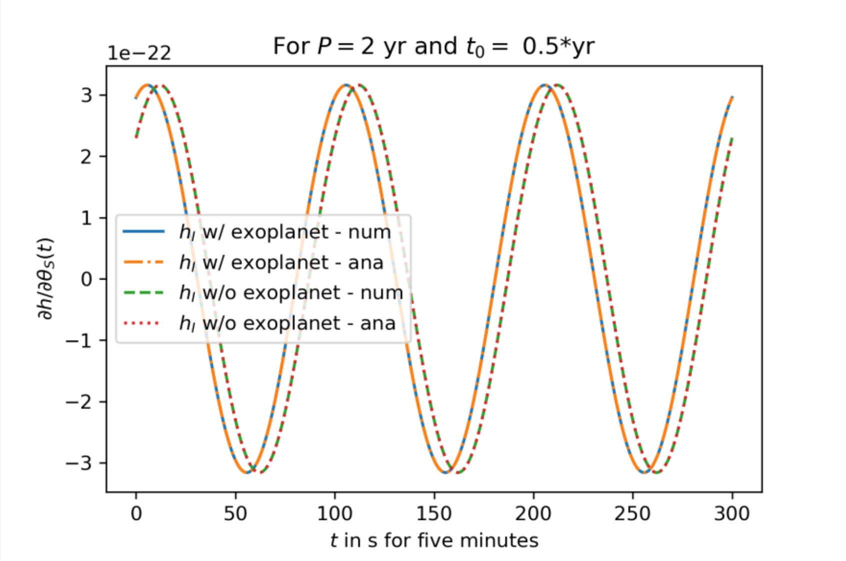
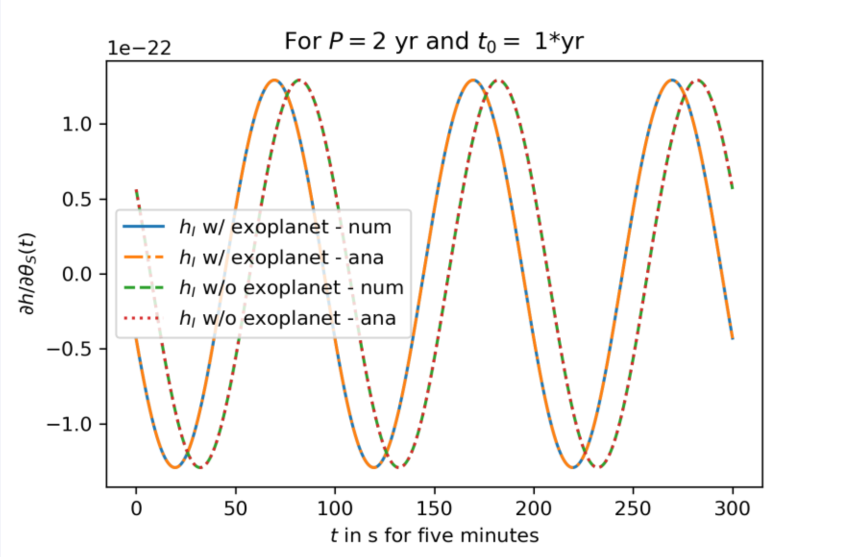
After killing f_0:

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…

<table>
<tr><th></th><th>Numerical derivative</th><th>Analytic derivative</th></tr>
<tr><td>Shape of the Amplitude</td><td>
</td><td>
</td></tr>
<tr><td>Shape of the frequency (1 hour after 0.3*yr)</td><td>
</td><td>
</td></tr>
</table>

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]
```

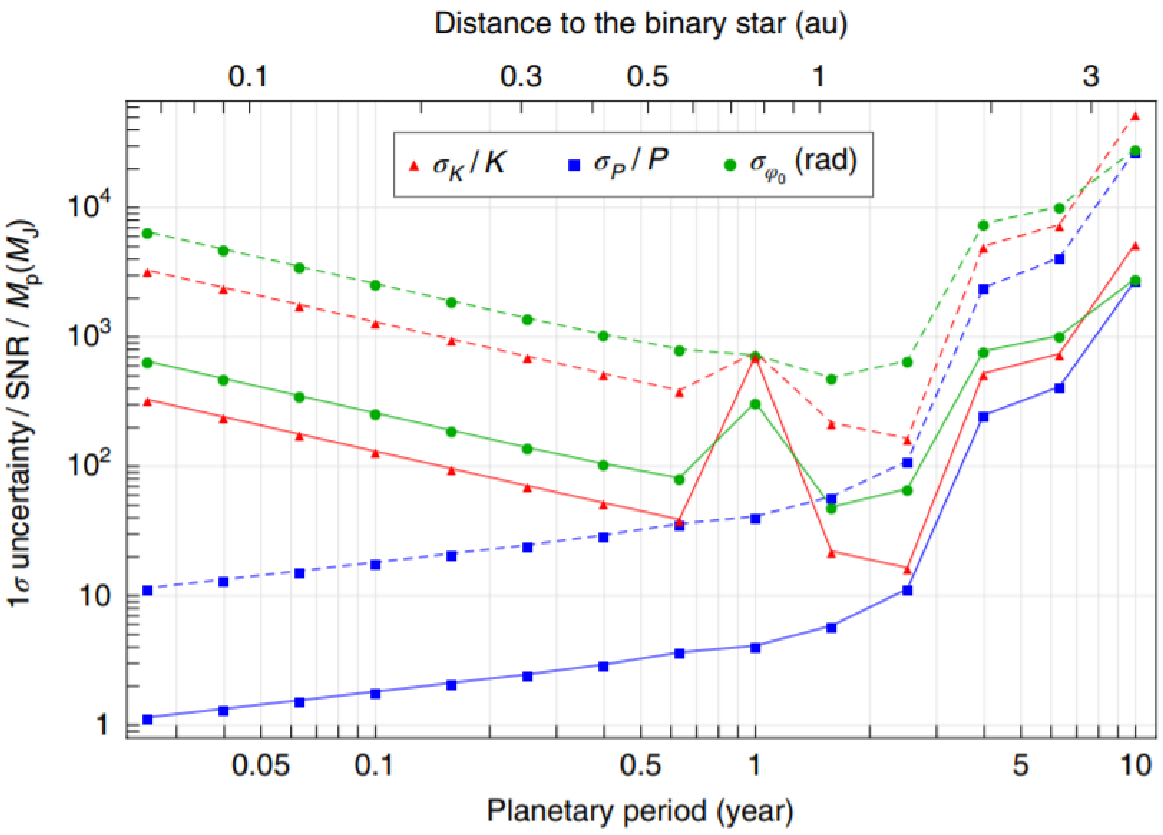
leite ab

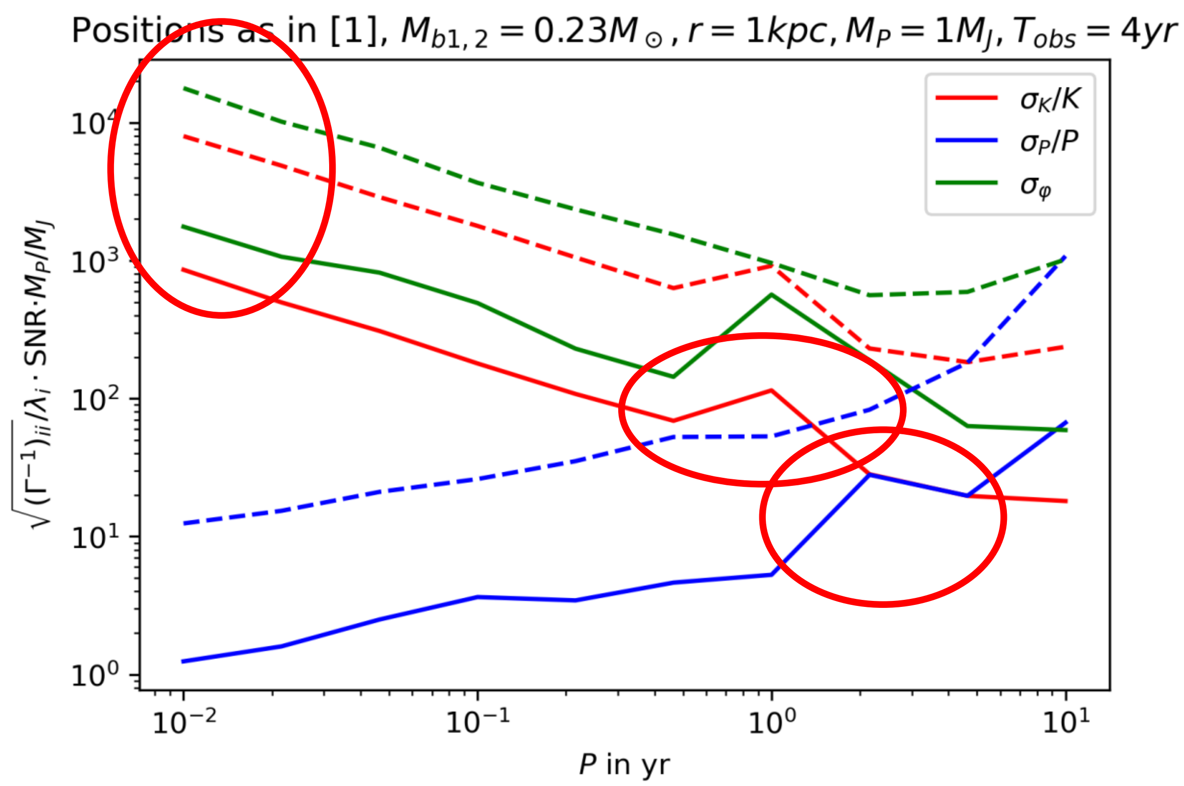
```
(√3 Cos[ψobs + ArcTan[1/2 A0 (1 + … 1 …^2) × (1/2 Cos[2 ArcTan[… 1 …, … 1 …]] Cos[2 (t ω + ArcTan[… 1 …])]
    (1 + (Cos[θEc]/2 - 1/2 √3 Cos[… 1 …] Sin[θEc])^2) - … 1 …), … 1 …] + (2 AU fGW π Cos[ϕEc-t ω] Sin[θEc])/c] (… 1 …)) /
  (4 √((A0^2 (Cos[θEc] Cos[θL] + Cos[ϕEc - ϕL] Sin[θEc] Sin[θL])^2 (1/2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





This plot took ~11 h of computation time: 10 P's * 2 frequencies * 28 integrals * 500'000 oscillations

My guess for the differences: This calculation was without fit for $\ln(A)$, f_1

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

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

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

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

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

  to see if it could see most promising exoplanet candidates/Jupiter-like planets
- Try to combine measurements of IceGiant and LISA

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