Implementing the FM technique

by Simon J. Murphy

1. Introduction

As you saw in the lectures, the frequency modulation (FM) technique is one way of determining the orbital parameters of a binary system. In this Exercise, you will learn to implement the FM technique using a *Kepler* binary system containing a δ Sct pulsator. The FM technique can be readily applied to binaries of a wide range of periods and with companions ranging in mass from planets to black holes.

The FM technique works by measuring the properties of sidelobes to Doppler-shifted pulsation frequencies. While it is relatively easy to measure an orbital period, obtaining other orbital parameters is quite difficult. Furthermore, there is no existing software to do it for you. In this Exercise, you will learn how to implement FM for yourself in a Jupyter Notebook. It will then be possible to use code that you write and apply it to other systems. Today, we'll work with KIC 9594022.

Warning: this Exercise is extremely challenging! You will be using `python` to analyse this binary system. If you are not very familiar with `python`, *please* team up with somebody who is. You will struggle otherwise. To make the Exercise easier, some of the harder tasks have been done for you and you will be able to copy-paste the necessary functions from this document. You only have 90 minutes, so if you are struggling, please ask for help. The instructions here are quite explicit to try to guide you through the process.

Why does this Exercise rely so heavily on `python`? `Python` is the most widely-used computer language in astronomy. Much of modern astronomy is conducted in `python`, and an understanding of `python` will open a lot of doors for you. There are excellent libraries to use, as well as a lot of open source software. As we saw in the data retrieval exercises, many official tools for obtaining and analysing lightcurves are written in `python`.

2. Learning Objectives

At the end of this tutorial you should be able to

- Identify frequency multiplets in the Fourier transform using `Period04`.
- Use a `python` function to fit pulsation frequencies to a data set (i.e. a lightcurve).
- Calculate orbital parameters for binary systems.
- Propagate uncertainty in the orbital parameters.

3. Plan

You should work in pairs or groups of three for these exercises, even if you each work on a Jupyter Notebook on your own computer. Don't worry if you don't finish the whole Exercise. The instructions are set up so that you can continue at a later time, if you wish.

You should begin by starting a new Jupyter Notebook, giving it a name, then following the instructions below. You'll also need to open `Period04`. Locate the light curve file, kic9594022_lc.txt, in the online directory.

4. Finding the orbital sidelobes

4.1 Importing and inspecting the data

Your first task is to import the light curve to `Period04`. You should check the light curve (*Display graph*) to make sure it is okay. Do any outliers need removing? You'll notice this is a full four years of *Kepler* data in long-cadence mode. That's a lot of data!

Calculate a Fourier transform of the light curve. You'll need to think carefully about the frequency range – it will need to be high enough to help you distinguish Nyquist aliases. Also think a little about the step rate of the Fourier transform. By default, 'high' oversamples by a factor of 20. This is much more than you need. Using a lower step rate will make the computation quicker.

Decide which are the strongest real peaks in the Fourier transform. Remember that real peaks will have higher amplitudes than their Nyquist counterparts. You can add them by right-clicking and choosing *Add peak to frequency list*, but be sure you get the peak and not another one nearby – use zoom carefully. You **do not** need to also select the Nyquist aliases of each real peak. If you select the real peak, and *calculate* a fit of this frequency to the light curve, its Nyquist alias will disappear. But beware: if you choose the Nyquist alias, the real peak will disappear! If you select a Nyquist alias now by mistake, then later in the Exercise you'll end up rediscovering *Kepler*'s orbit around the Sun – still a fun thing to do, but not our main goal today. Extract just a few of the strongest peaks for now.

Using the *Fit* tab, select the peaks you have extracted (using the check-boxes) and click *Calculate*. This determines a linear least-squares fit of the selected frequencies to the light curve. After this, click *Improve all*, which will run a non-linear least-squares fit of these frequencies to improve the frequencies, amplitudes and phases.

Now return to the *Fourier* tab, change the *Calculations based on:* from *Original data* to *Residuals at original*, and recalculate the Fourier transform over the same frequency range. Notice that the real peaks and their Nyquist aliases are now gone.

4.2 Extracting the orbital sidelobes

This is a known binary system, so each of the pulsation frequencies is modulated by the orbital frequency – the pulsating star is like an FM radio! In the Fourier domain, this causes each pulsation frequency to be split into a multiplet. We have already identified and fitted the pulsation frequency, which is the central component of the multiplet; now we are going to identify the sidelobes of that multiplet.

Calculate the Fourier residuals in a narrow range, say $\pm 0.03\,\mathrm{d}^{-1}$, around the strongest peak. You should now see the frequency multiplet, minus the central component which you've already fitted. Right click on the first sidelobes of this peak. They should be the stronger and more central peaks that remain. Go to the *Fit* tab, select these new peaks, and *Calculate* and *Improve all* on **all** peaks that you've extracted so far.

By this stage, you should have extracted the strongest peak (we'll call it f_1) and its first sidelobes (let's call them f_{1p1} and f_{1m1}, where subscripts p and m denote 'plus' and 'minus', representing $f_1 + 1 * f_{\mathrm{orb}}$ and $f_1 - 1 * f_{\mathrm{orb}}$, respectively). **If you are not sure that you have now reached this stage, ask for help!** You may also have extracted a few additional pulsation frequencies, but don't worry about the sidelobes of those other frequencies – we are going to focus on f_1 and its sidelobes for now.

5. Analysing the orbit with `python`

While it is important to learn to use python to analyse astronomical data sets, your ability to use python should not be a barrier to completing this Exercise. Every task here will have hints and/or solutions available that you can refer to if you are stuck. `Google` is your friend! E.g. if you are not sure what the syntax is for `np.loadtxt` (the program for reading a .txt file using the `numpy` library), you can quickly find the answer on `Google`.

5.1 Getting started

Begin by importing `numpy` and importing `optimize` from the `scipy` package. The light curve file has only two columns: time (d) and brightness (mag). Read this in so that you have a time array and a flux array [hint-5,1]. You will also need the frequencies f_1, f_{1p1} and f_{1m1}. Copy those from `Period04` and assign them to variables. It would be wise to collect them in an array, or list. [solution-5,1]

5.2 Preparing to optimize

Soon we are going to optimize a fit of f_1, f_{1p1} and f_{1m1} to the *Kepler* light curve, because we need to know the best-fitting frequencies, amplitudes and phases of those, *and* we need the uncertainties on those quantities. The uncertainties are a necessary step in determining the uncertainties on the binary orbit, later.

Our time array has a large, non-zero mean. It's bad practice to optimize on an array of times with such a large mean, because any uncertainties in frequency will accumulate. (You might like to think about why this is the case.) Subtract the mean time, so that the times array is centered on zero. [hint-5,2] [solution-5,2]

5.3 A function that optimizes a multi-sine-wave fit

The pulsational variability in the light curve can be expressed as a sum of N sinusoids, i.e.

$$\Delta L = \sum_{i=1}^{N} A_i \sin(2\pi f_i t + \phi_i), \tag{1}$$

where L is the stellar luminosity, A_i and ϕ_i are the pulsation amplitude and phase of the frequency f_i, and t is the independent variable – the times of observation.

Writing your own code to use the `scipy` optimizer is quite difficult. Unless you really want a challenge, I suggest that you copy the two functions-5,3 into your notebooks and run them both. Do try to understand what these functions are doing. Now, all you have to do to get the values of your parameters and the uncertainties is to run this line:

```
a_out, freq_out, phi_out, a_err, freq_err, phi_err = optimizer(time,mag,freqs)
```

where `time` is your array of times, `mag` is the equally-sized array of magnitudes, and `freqs` is your frequency array. This line collects the optimised amplitudes, frequencies, phases, amplitude errors, frequency errors and phase errors for your input frequency array, and puts them in equally sized arrays. If your input array was

```
freqs = [f1, f1p1, f1m1]
```

then your a_{out} array will be

```
a_out = [a1, a1p1, a1m1]
```

and so on. The order you used in your `freqs` array that you then sent to the `optimizer` function will be the same order of frequencies and corresponding amplitudes and phases that you get in the outputs. It is not important which order you choose, but you will need to be consistent. You can see the values by typing

```
a_out
```

on its own in a cell, and you can access an individual element by its index, e.g. the amplitude of the pulsation frequency is accessed by typing

```
a_out[0]
```

For ease of access of these variables later, you may like to now reassign your variables so that you don't get confused with array indexing. E.g.

```
f1, f1p1, f1m1 = freq_out

⋮

a1_err, a1p1_err, a1m1_err = a_err
```

But make sure that you refer to *your* ordering of the `freqs` array. You can ignore the phases and phase errors for now – we won't need them in this Exercise. [solution-5,3]

5.4 Calculating the orbital period

We saw in the lecture that the sidelobes should be equally split from the central frequency, and this splitting is equal to the orbital frequency. That is,

$$\nu_{\rm orb} = f_1 - f_{1m1} = f_{1p1} - f_1 \tag{2}$$

Check that your frequency splittings are equal (at least to the first two significant figures), and take the average splitting as $\nu_{\rm orb}$. The orbital period, $P_{\rm orb}$, is the reciprocal of $\nu_{\rm orb}$. [solution-5,4a]

You can calculate the uncertainty on the orbital period. Since we have used the average splitting as our orbital frequency, the uncertainties on the individual frequencies will need to be combined in quadrature. The orbital period has the same *fractional* uncertainty as the orbital frequency. That is now calculable, using eq 3. [solution-5,4b]

$$\frac{\sigma(P_{\rm orb})}{P_{\rm orb}} = \frac{\sigma(\nu_{\rm orb})}{\nu_{\rm orb}}. \tag{3}$$

If we wanted to be sure that this is really a binary system, we could check several pulsation modes and make sure that they all give the same orbital frequency, since every pulsation mode should respond to the orbit in the same way. For now, let us look at extracting some of the other orbital parameters using only the f_1 multiplet.

5.5 Estimate the eccentricity

Take a break from `python` and go back to `Period04`. We need to measure the amplitudes of the second sidelobes to calculate the eccentricity. Don't worry, we aren't going to import them back into `python` and run our optimizer again, we're going to cheat and use the values measured in `Period04`.[1]

Using the same zoomed-in frequency interval around f_1 that you used before, locate the second sidelobes. You have an expectation of the frequency of these already, so you can use that to confirm that you get the right peaks. Add them to the frequency list, click *Calculate* and then *Improve all.* Assign these values as variables in your Jupyter Notebook.

The eccentricity is calculated from the amplitudes of the sidelobes as follows:

$$e \simeq 2\frac{A_{1p2} + A_{1m2}}{A_{1p1} + A_{1m1}} \tag{4}$$

This approximation to the eccentricity is very good, unless the $e \sim 1.0$. Calculate the eccentricity using eq 4 now. In this case, we see the eccentricity is not too large, and the approximation is valid.[2] [solution-5,5]

---

[1] In any case, the optimizer is uses non-linear least-squares to optimize the frequencies, which is not the best choice in this case. We'd rather use linear least-squares because we know the second sidelobes should be separated by exactly $2\nu_{\rm orb}$, i.e. we know the frequencies. But we don't want to set up a new function right now, and we don't need to.

[2] If you are using this tutorial to analyse a more eccentric binary, see Shibahashi et al. (2015) MNRAS 450, 3999 for a more complete formalism.

5.6 Calculate alpha

The binary mass function, $f(M)$, and the (projected) semi-major axis of the pulsating star's orbit, $a_1 \sin i$, require the calculation of the parameter α, which relates the amplitude of the first sidelobes to the central peak. To first order, α can be approximated as

$$\alpha \simeq \frac{A_{1p1} + A_{1m1}}{A_1}, \tag{5}$$

but this is only valid for $\alpha \ll 1$. A more accurate formula is

$$\alpha + \frac{\alpha^3}{8} + \mathcal{O}(\alpha^5) = \frac{A_{1p1} + A_{1m1}}{A_1}, \tag{6}$$

where we shall ignore terms of $\mathcal{O}(\alpha^5)$.

Depending on your `python` ability, you can either write your own code to solve this cubic equation for α, perhaps implementing the `np.roots` function and using [hints-5,6], or you can copy function-5,6 which will do it for you.

The uncertainty on α is calculable by combining the uncertainties on A_1, A_{1p1} and A_{1m1} in quadrature. [solution-5,6]

5.7 Calculate the binary mass function

The mass function relates the component masses, m_1 and m_2 of a binary system,

$$f(M) = f(m_1, m_2, \sin i) = \frac{m_2^3 \sin^3 i}{(m_1 + m_2)^2}, \tag{7}$$

where i is the orbital inclination. Two equal-mass components seen at an inclination of 90° (i.e. edge-on, eclipsing) will have a mass function of 0.5. The same system seen at $i = 0°$ will have a mass function of 0. Generally, a lower mass function suggests a lower companion mass, but we rarely know the inclination of a system. This method, like the RV and PM methods, is unable to determine the inclination separately.

The mass function is given by

$$f(M) = \alpha^3 \frac{P_{\rm osc}^3}{P_{\rm orb}^2} \frac{c^3}{2\pi G}, \tag{8}$$

where $P_{\rm osc}$ is the period of the oscillation mode (i.e. the reciprocal of its frequency), c is the speed of light and G is the gravitational constant. For a mass function in S.I. units, i.e. in kg, you'll need to convert $P_{\rm osc}$ and $P_{\rm orb}$ from units of days to units of seconds. You can then convert the answer to solar masses.

Once more, the uncertainty is calculable by combining other uncertainties in quadrature. Be careful of the exponents! [hint-5,7] [solution-5,7]

5.8 Calculate the semi-major axis

Since we cannot determine the inclination separately, we can only compute the *projected* semi-major axis, meaning $a_1 \sin i$, convolved with inclination. For this, we use α again

$$a_1 \sin i = \frac{\alpha c}{2\pi \nu_{\rm osc}}. \tag{9}$$

Make sure you think about units carefully! You can convert an S.I. quantity to more useful units; au and $R_\odot$ are often used. My own preference is units of light-seconds, i.e. to quote $a_1 \sin i/c$ in units of seconds, because this corresponds neatly to light arrival-time delays measured in seconds. [solution-5,8]

5.9 Calculate the (minimum) companion mass

The lowest companion mass implied by the mass function occurs for $i = 90°$. We can compute this minimum mass by making an assumption on the primary mass. Since we believe the primary star is a typical δ Sct star, it has a mass somewhere around $m_1 = 1.7\,\mathrm{M}_\odot$. Copy the two functions-5,9 for computing the companion mass to estimate the mass of the companion in solar masses. [solution-5,9]

The uncertainty on the minimum companion mass is difficult to specify analytically, though easy to determine numerically. You can simply send `fm_msun - fm_msun_err` to the function `compute_m2`. However, since we do not know i, and we simply guessed that $m_1 = 1.7\,\mathrm{M}_\odot$, we cannot consider this to be an accurate uncertainty.

The End

That's the end! As stated in the introduction, this is a challenging Exercise. It is okay if you didn't finish all of the steps. You have the data and these instructions and the hints/solutions available, so you can continue in your own time if you wish. If you finished early, or if you wish to pursue the method further, continue reading the next section.

Following this Exercise, you should have got parameters similar to those in Table 1.

6. Challenges / future work

Every pulsation frequency of the star should be similarly affected by the binary orbit. Move on to the next few pulsation frequencies of this star. Obtain your f_2, a_2, and ϕ_2 (etc.) results, for both pulsation frequencies and their sidelobes, simultaneously with the fit for f1. You should get similar answers from the other pulsation frequencies. You can combine the results from the different pulsation frequencies into a single orbital solution.

Finally – and this something you really won't have time to do in class – you can look at other binaries. KIC 11754974 is a nice example of a circular binary, with the added challenge of gaps in *Kepler* data coverage. There's a whole catalogue of *Kepler* binaries you can play with, with solved orbits so that you can check your solutions (Murphy et al. (2018) MNRAS 474, 4322).

7. Things to note if you use this for research

Please cite the two FM papers. It would be wise to read and understand these papers. There is some additional complexity that was not accounted for in this Exercise.

Parameter	units	value	uncertainty
P_{orb}	d	412.1	0.7
$a_1 \sin i/\mathrm{c}$	s	170.3	3.5
$f(M)$	$\mathrm{M}_\odot$	0.0312	0.0011
e		0.44	
$m_{2,\mathrm{min}}$	$\mathrm{M}_\odot$	0.54	[0.01]

Table 1: Table of orbital parameters for KIC 9594022, as determined via this Exercise.

One should check that the frequency modulation originates from binary motion. There is a mathematical test for this. One must force the frequency splittings of the first sidelobes to be equal (to their mean), and then compute their phases with respect to a time of inferior or superior conjunction. If the origin is binarity, the sidelobe phases should be different by $\pi/2$ rad. See the FM1 paper for details.

Note that the orientation of the orbital ellipse (specifically the longitude of periastron), ϖ, influences the derived value of α. This example was chosen for ϖ to be close to $3\pi/2$ so that $\cos^2\varpi \sim 0$, for simplicity. For any other system, it will be necessary to use the phases of the second sidelobes to determine ϖ, in accordance with eq.(53) of the FM2 paper.

It is necessary to implement a least-squares fitting procedure, rather than the current non-linear least-squares procedure, in order to analyse the second sidelobes and to check the aforementioned $\pi/2$ phase relationship.