User Guide For Peakbagging Made Easy

Martin Bo Nielsen & Emanuele Papini

September 20, 2018

Contents

1 Introduction to PME

This is a guide for using Peakbagging Made Easy (PME). PME is meant to be an easy to use, modular peakbagging tool box, for fitting the oscillation power spectra of solar-like stars. It may be possible to fit more evolved or hotter stars, but there is no guarantee that PME won't be confused by this choice and explode.

PME is written in Python and with a bit of FORTRAN for some of the more basic repetitive calculations. The fitting is done using maximum likelihood estimation (MLE), where the parameterspace spanned by the fit variables is sampled using an afine-invariant Markov Chain Monte Carlo sampler for Python called EMCEE [Foreman-Mackey et al., 2013].

1.1 Setup

1.1.1 Dependencies

1. PME requires NumPy, Scipy and EMCEE.
2. A fortran compiler for the system you are using (Intel/Gfortran/AMD/others?)

1.1.2 Getting ready for a peakbagging run

1. Download PME from here
2. Place the contents of PME (*.py/*.f90 files) into a directory of your choice.
3. Place the power spectrum or time series of your star into a directory of your choice.
4. Make sure the power spectrum file has 2 columns corresponding to frequency and power and that it has the *.pow extension, and only 1 file with this extension is in the directory. Header lines preceded by # will be ignored. Additional columns will be ignored.
5. Use f2py to compile flops.f90 with your favorite Fortran compiler (f2py is included in a typical installation of Numpy). Example using the Intel compiler for Fortran:

```
f2py --f90exec=ifort --f90flags=-O3 -c flops.f90 -m flops
```

2 Peakbagging with PME

PME is broken down into three main functions, which are chosen as options when calling PME in the command line.

- **-autoguess** provides automatic initial guesses for peakbagging
- **-peakbagging** performs a maximum likelihood fit to the power spectrum using MCMC, **-autoguess** must be run prior to using this option
- **-plot** plots the result of the fit.
- **-output** produces an csv file with the best-fit mode parameters.

The number of options available in PME have over time become quite substantial, and as such it is beyond the scope of this document to go through every one of them. For a complete list of options for PME do

```
python PME.py --help
```

The main functionality of PME, lies in the parameterization of the variables, and the user should see the individual **parameterization.py* files for information about each variable set (see 3.2).

2.1 Autoguess

Autoguess attempts to provide initial guesses for the background noise in the spectrum, as well as any oscillation modes that may be present.

To use **-autoguess**, do the following.

1. Run PME with the **-autoguess** option:

```
python PME.py /folder/with/spectrum starID -autoguess
```

   -autoguess creates *starID_output.npz* which will contain all the information relevant to the whole peakbagging run, and *starID_initial_guesses.txt* which contains the initial guesses for the modes in human readable form. This should be used to manually add modes that are missed by **-autoguess**.

2. Check the contents of *starID_initial_guesses.txt*. This can be done graphically by:

```
python PME.py /folder/with/spectrum starID -autoguess -check
```

   Simply add the missing modes on separate lines in the **initial_guesses.txt* file. Order in this list is irrelevant, and only the manually added frequencies need to be more or less precise (heights and widths not so much). Provided that the frequencies are reasonable (solar-like) an attempt at assigning mode IDs will also be done automatically in the next step. Note that adding $l = 3$ modes will cause the mode ID routine to fail, add these at the end.

3. Now run PME with the **-autoguess -refit** options. This will go through all the modes in **initial_guesses.txt* and perform the whole mode identification rigmarole again, but with your inputs as override. Provided at least one $l = 2, 0$ pair has been added to the **initial_guesses.txt* file, **-refit** will try to assign l values to the rest of the modes.

4. Iterate 2 and 3 once or twice to make sure you're happy with the modes

The background fit is done using harvey laws and a white noise term, and this is usually very robust, so manual interaction is rarely needed. If it fails it is usually because ν_{max} is not estimated correctly. Use the **-nu_max** option to manually set ν_{max} when calling PME with **-autoguess**.

The mode identification routine is less robust. It works in two steps: peak detection and mode ID based on frequency. **-autoguess** only tends to pick up the highest amplitude modes. This can be partially fixed if you include the **-min_sign X** option, where X is a small value (5, say).

The mode identification works by finding the smallest frequency separation and assuming that this is equal to the small separation, and that this pair is then $l = 2, 0$. **-autoguess** then counts outward in frequency, sequentially identifying modes. If no frequency separation is found that resembles a small separation, this step fails and modes are labeled $l = -1$. Note that $l = 3$ modes also cause this part of **-autoguess** to fail.

2.2 Peakbagging

Once you have obtained a satisfactory list of modes to fit, the next step is running PME with the **-peakbagging** option. **-peakbagging** uses the list of initial guesses from *starID_initial_guesses.txt* for the modes and *starID_output.npz* for the background terms.

The basic run of **-peakbagging** would be:

1. Check that the walkers initialize properly by doing a short **-peakbagging** run of just a few steps

```
python PME.py /folder/with/spectrum starID -peakbagging
                                    -mcmc_walkers XXX
                                    -some_additional_parameterizations
                                    -prior parameter.prior
```

   -mcmc_walkers sets the number of MCMC walkers in the fit. This should be an even number, and at least twice the number of fit parameters (after the parameterization has

been chosen), and preferably more than this (300+ usually). PME will warn you if this needs to be increased.

The parameterization refers to the group of parameters, e.g., **-freqs**, **-widths** etc. (see Section 3.2) Peakbagging in PME can be done with a number of different parameterizations for the different groups of parameters. Adding new parameterizations is possible by adding them to respective *parametrization.py file.

Priors may be used with the **-prior** options. Priors must be provided as a two column ascii file with probability density of the relevant parameter. Use **-peakbagging -show_prior_list** for a list of valid file names.

2. Start a full length peakbagging run

```
python PME.py /folder/with/spectrum starID -peakbagging
                                    -mcmc_walkers XXX
                                    -some_additional_parameterizations
                                    -prior parameter.prior
                                    -mcmc_steps XXXXX
                                    -mcmc_substeps XXX
```

Use the **-mcmc_steps** option to set the number of steps the walkers should take through the parameterspace. The choice of this number should be large (10^4 to 10^5).

For long runs use the **-mcmc_substeps** option to save progress every XXX steps. Useful if the runs may be interrupt for some reason.

3. To resume a previous run, use the **-mcmc_burnt** flag. Note this uses most of the previous settings, so only **-mcmc_steps** and **-mcmc_substeps** are required. In principle this means that by using **-mcmc_burnt** you can skip point 2.

```
python PME.py /folder/with/spectrum starID -peakbagging
                                   -mcmc_burnt
                                    -mcmc_steps XXXXX
                                    -mcmc_substeps XXX
```

2.3 Plot

After **-peakbagging** has run, or the first subsection of the run has been saved, you can use the **-plot** option to check the state of the fit:

```
python PME.py /folder/with/spectrum starID -plot
```

Be aware that **-plot** will display a large number of plots, so machine memory may be an issue. (an option to limit plot output will be implemented)

The point of this option is to provide diagnostic plots that will show if something is wrong in the fit, and potentially when the fit has converged. The majority of the plots are of the MCMC chains projected onto each axis of the parameter space. These are shown in terms of percentiles of the distribution of the walkers (This will be replaced by KDE to pick up bimodalities).

The negative logarithm of the log-likelihood is also shown (yes, log of log). Walkers will likely have converged once this flattens out and stays flat for a long time.

In addition to the above, the best-fit model is also plotted, as well as an eschelle diagram, which will helps identify problems with the mode ID.

2.4 Output

Using PME with the **-output** option takes the percentiles of the last few steps of the PME run, and outputs them to an csv file. Ensure that the MCMC chains have burnt in before using this. Use **-plot** for this. Usually the likelihood flattening out is a good indicator, and that there is no significant gradient over the last few thousand steps in any of the parameter plots.

Unless the number of walkers is large the default settings for **-output** do not produce a reliable representation of the posterior (small numbers). You can use **-output -sample_posterior XX** to increase the number of samples drawn from the posterior so that the statistics bare more robust.

2.5 Example run

1. Get initial guesses with manual ν_{max}

```
python PME.py /peakbagging/Data/6679371 kic006679371 -autoguess -nu_max 940
```

2. Manually check for missing modes

```
python PME.py /peakbagging/Data/6679371 kic006679371 -autoguess -check
```

3. Add missing modes to *kic006679371_initial_guesses.txt*
4. Update initial guess fits for amplitude and widths

```
python PME.py /peakbagging/Data/6679371 kic006679371 -autoguess -refit
```

5. Check everything is in order

```
python PME.py /peakbagging/Data/6679371 kic006679371 -autoguess -check
```

6. Peakbagging using Apporchaux 2014 width parameterization (short test run)

```
python PME.py /peakbagging/Data/6679371 kic006679371 -peakbagging -mcmc_walkers
    300 -mcmc_threads 4 -widths aporchucks
```

7. Start a long run, saving every 100 steps

```
python PME.py /peakbagging/Data/6679371 kic006679371 -peakbagging -mcmc_burnt
    -mcmc_steps 100000 -mcmc_substeps 100
```

8. Repeat previous step as necessary until burn-in is completed.
9. Output best-fit mode parameters drawn from posterior (1000*300 samples)

```
python PME.py /peakbagging/Data/6679371 kic006679371 -output -sample_posterior
    1000
```

3 Background

3.1 Likelihood function

MLE seeks to find the set of parameters $\mathbf{\Theta}$ of a model $M(\mathbf{\Theta}, \nu_j)$ that has the highest probability of explaining a set of observations. In this case the observations consist of the power $P(\nu_j)$ of a time series, at frequency ν_j (PME assumes ν is in units of μHz). Here, $\{\nu_j\}$ is a set frequencies that span the range of interest, i.e., the p-mode envelope.

This time series can in principle be anything that reflects the oscillations of a star, i.e., photometric intensity measurements or radial velocity. However, PME assumes that the measurements in the time series are independent (no smoothing), Gaussian random variables and at least somewhat regularly spaced[1].

[1] Power spectra of highly irregularly sampled time series are more complicated to fit, and at the moment PME is not setup to do so.

The above implies that the observed power is distributed according to a χ^2 distribution with 2 degrees of freedom [e.g, Woodard, 1984, Duvall and Harvey, 1986, Gizon and Solanki, 2003]. The probability of observing a given value P_j in the power spectrum is thus an exponential distribution:

$$f_j\left(P_j|\mathbf{\Theta}\right) = \frac{1}{M(\mathbf{\Theta},\nu_j)}\exp\left(-\frac{P_j}{M(\mathbf{\Theta},\nu_j)}\right) \tag{1}$$

The best-fit parameters are the parameters that maximize the log-likelihood, which is the logarithm of the joint probability function:

$$L\left(P_j|\mathbf{\Theta}\right) = \ln\prod_{j=1}^{J} f_j\left(P_j|\mathbf{\Theta}\right) = -\sum_{j=1}^{J}\left(\ln M\left(\mathbf{\Theta},\nu_j\right) + \frac{P_j}{M\left(\mathbf{\Theta},\nu_j\right)}\right). \tag{2}$$

Note that the logarithm of the joint probability is used for better numerical stability in the optimization (again, explosions may otherwise occur).

3.2 The model

The model, or expectation value of the spectrum is given by [Harvey et al., 1988, Anderson et al., 1990]:

$$M\left(\mathbf{\Theta},\nu\right) = \sum_{n}\sum_{l=0}\sum_{m=-l}^{l}\frac{S_{nlm}/\Gamma_{nlm}\,\mathscr{E}_{lm}\left(i_{nlm}\right)}{1+\left(2/\Gamma_{nlm}\right)^2\left(\nu-\nu_{nlm}\right)^2} + \sum_{k=1}^{K}\frac{A_k/\nu_k}{1+\left(2\pi\nu/\nu_k\right)^{\alpha_k}} + W. \tag{3}$$

In the sums over n,l, and m, S_{nlm} is the mode height, Γ_{nlm} is the full-width at half maximum of the Lorentzian profile, and ν_{nlm} is the mode frequency. $\mathscr{E}_{lm}\left(i\right)$ is the modulation of the azimuthal components of a given multiplet, due to the inclination of the rotation axis of the star. PME can in principle handle as high l as you want, but stick to $l \leq 3$.

The sum over k contains the frequency dependent background terms, with amplitude A_k, characteristic frequency ν_k and slope α_k. W is the frequency independent white noise level. PME can handle up to $K = 3$, but usually $K = 2$ is sufficient.

In its most basic form, PME would use each of the independent variables in Eq. 3 as fit parameters. However, this is not recommended (explosions), and in fact it doesn't always make sense to do this (i_{nlm} for example). This setup was chosen to make the code as flexible as possible, but at the cost of efficiency and computational speed.

It is often far more beneficial to parameterize the fit parameters as a function of, e.g., frequency. PME treats each parameter in terms of sets, and the user can select a parametrization for each set. These sets are:

1. Mode frequencies (ν_{nlm}; this includes splittings)
2. Mode heights (S_{nlm})
3. Mode widths (Γ_{nlm})
4. Mode inclination (i_{nlm})
5. Background power-law exponent (α_k)
6. Background characteristic frequency (F_k)
7. Background amplitude (A_k)
8. Background white noise (W)

Bibliography

E. R. Anderson, T. L. Duvall, Jr., and S. M. Jefferies. Modeling of solar oscillation power spectra. *ApJ*, 364:699–705, December 1990. doi: 10.1086/169452.

T. L. Duvall, Jr. and J. W. Harvey. Solar Doppler shifts: Sources of continuous spectra. In D. O. Gough, editor, *NATO Advanced Science Institutes (ASI) Series C*, volume 169 of *NATO Advanced Science Institutes (ASI) Series C*, pages 105–116, 1986.

D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman. emcee: The MCMC Hammer. *PASP*, 125:306–312, March 2013. doi: 10.1086/670067.

L. Gizon and S. K. Solanki. Determining the Inclination of the Rotation Axis of a Sun-like Star. *ApJ*, 589:1009–1019, June 2003. doi: 10.1086/374715.

J. W. Harvey, F. Hill, J. R. Kennedy, J. W. Leibacher, and W. C. Livingston. The Global Oscillation Network Group (GONG). *Advances in Space Research*, 8:117–120, 1988. doi: 10.1016/0273-1177(88)90304-3.

M. F. Woodard. *Short-Period Oscillations in the Total Solar Irradiance.* PhD thesis, University of California, San Diego., 1984.