Short instructions for fitting radial velocity curves with *rvfit*

Ramón Iglesias Marzoa

Contents

1 Introduction

This document is a short guide on how to use the `rvfit` code and its associate procedures. It doesn't describe the internal algorithms, nor details about the calculations, nor the performance of the code. If you need those details, a full description can be found in Iglesias-Marzoa et al. (2015).

`rvfit` is written in IDL v7.0, so you need this or newer versions to run it. If you use this code to fit the radial velocities of -your- stellar system, please, cite Iglesias-Marzoa et al. (2015).

2 What this code does

`rvfit` fits non-precessing keplerian radial velocity (RV) curves for double-line and single-line spectroscopic binaries or exoplanets (see Hilditch, 2001, sec. 2.7). To do this, it uses an Adaptive Simulated Annealing (ASA) algorithm (Ingber, 1996) which fits the following six keplerian parameters (seven for a double-lined binary star): the orbital period P, the time of periastron passage T_P, the eccentricity e, the argument of the periastron ω, the systemic velocity γ, and the amplitude of the radial velocity K_1. In the case of a double-line binary the seventh parameter is the amplitude K_2 for the secondary star.

3 What this code doesn't do

This code doesn't fit keplerian orbits with precession due to interactions with a third body which can be modelled with derivatives in ω (Ford, 2005, eq. 15) .

This code doesn't fit radial velocity curves with Rossiter-McLaughlin effect. To do this you need a more complex physical model of the two components of the binary, including sizes, rotation, and limb-darkening effects (see Ohta et al., 2005; Giménez , 2006).

Also, it doesn't include relativistic or tidal effects (see Sybilski et al., 2013).

4 Files needed

To run `rvfit` you need the following files:

- `rvfit.pro`, contains the main program with the ASA algorithm.

- `RVlib.pro`, contains common functions needed in several tasks: Kepler equation, computation of radial velocities, computation of χ^2, cumulative distribution functions (CDF), read/write of parameter files, etc...
- `computeRVparams.pro`, contains the computations of the physical parameters and the plots. It is called by the `rvfit.pro` program.
- `pxperfect.pro`, a program from Craig B. Markwardt to make pretty Postscript plots for publication.

Also, if you want to perform a Markov Chain Monte Carlo (MCMC) analysis of the uncertainties you will need:

- `MCMCerrors.pro`, to perform a MCMC simulation centered in the parameter values computed with `rvfit`.
- `MCMCanalysis.pro`, to analize the data generated by `MCMCerrors.pro`. Of course you can use your own code to do this.

The next files are the datasets to run the example for the eclipsing binary GU Boo:

- `GUBoo_RV1.dat` and `GUBoo_RV1.dat`, contains the RV data from López-Morales & Ribas (2005).
- `GUBoo.conf`, an example configuration file which you can change for your system.

5 Description of the RV data files.

The RV data must be arranged in files with columns separated by tabulators or spaces, with this structure:

```
HJD RV s(RV)
```

In the first column you must put the Heliocentric (or Baricentric) Julian Day, and in the second and third columns the RV values, in km/s, and their uncertainties, also in km/s. It is mandatory to set the uncertainties for each RV. If, for some reason, you don't know those uncertainties you can fix them all to 1.0 in the data files. Notice that the program will not run if the s(RV) column is missing values.

If you want to fit a double-lined binary you need another file with the secondary RV and the same structure.

6 How to set a configuration file

`rvfit` uses a parameter configuration file for your object as input. Lets call this file `object.conf`. This can be a plain text file with this structure:

```
\a
rvfile1  = object_RV1.dat
rvfile2  = object_RV2.dat
fitparam = [1,          1,              1,       1,       1,      1,      1]
valparam = [0.4887280d, 2452723.9811d,  0.0d,    0.0d,   -24.57d, 142.65d,145.08d]
L        = [0.1d,       2452722.9811d,  0.,      0.,     -100.,   0.,     0.]
U        = [10.0d,      2452724.9811d,  0.999,   360.,    100,    150.,   150.]
```

Of course, you can copy and paste the previous lines to your `object.conf` file and modify it to meet your needs. Also, you can write comments in the lines following this lines but not before them, since the first six lines are reserved for those parameters.

The parameters `rvfile1` and `rvfile2` are the names of the data files with the RV measurements (see Section 5). In this case we want to fit a double-line binary with the data in these two files: `object_RV1.dat` and `object_RV2.dat`.

If you want to fit a single-line binary (or exoplanet) you must set `rvfile2 = ''`. In this case the code understands that you have only one radial velocity curve, and the system is handled as as single-line object.

The vectors called `fitparam`, `valparam`, `L`, and `U` all have the same positional structure for the keplerian elements of the system:

`[P,Tp,e,omega,gamma,K1,K2]`

where the elements and their units are:

- `P`: the orbital period P in days,
- `Tp`: the time of periastron passage T_P, in Heliocentric Julian Days (HJD),
- `e`: the orbital eccentricity e, in the range [0,1),
- `omega`: the argument of the periastron ω in decimal degrees,
- `gamma`: the systemic radial velocity γ of the system measured in km/s,
- `K1`: the amplitude of the radial velocity K_1 for the primary (most massive) star in km/s,
- `K2`: the same for the secondary (less massive) star K_2 in km/s.

Since the primary star is the most massive of the system then $K_1 \leq K2$, but this condition is not checked in the code.

`fitparam` is a seven element vector which sets the parameters to be fitted. If you set a parameter position to 1, that parameter will be fitted. If you set that position to 0 the parameter will not be fitted, instead the code will use the value of that parameter set in `valparam`. You must fit at least two parameters.

`valparam` is a seven element vector with the values used for the fixed parameters, i.e., the parameters set to 0 in `fitparam`. In the position of the parameters to be fitted, i.e. those set to 1 in `fitparam`, you can put any value. However, the values must be within a reasonable range to not obtain weird results. As an example, lets think of an eclipsing binary for which the exact period orbital from the ligh curve is known. Then you can fit the RV curve by fixing to 0 the first element of `fitparam` and inserting the known period in the first element of `valparam`. The code will not fit the period and will use that value.

`L` is a seven element vector with the lowest value limit for the fitted parameters. It is only used for the parameters set to 1 in `fitparam`. The value of all the parameters in `L` must be less than those in `U`.

`U` is a seven element vector with the highest value limit for the fitted parameters. It is only used for the parameters set to 1 in `fitparam`. The value of all the parameters in `U` must be greater than those in `L`.

In `valparam`, `L` and `U` you must use double precission floating point numbers for the parameters where the number of significative digits is important, i.e., in the period P and in the time of passage for the periastron T_P. Look at the 'd' at the end of these numbers in the example provided.

If you are fitting a single-line object (remember, with `rvfile2 = ''`), the parameter K_2 is never fitted, and any value set in the last position of `valparam`, `L` and `U` is discarded. **But you must set it to a value!**. You can set it to 0. or 1. or whatever number you want, otherwise an error is issued and the program stops.

7 Options

`rvfit` can be run with a number of optimal input parameters and *keyword parameters*. As optional parameters we have:

- `configfile='object.conf'` If set, the parameter file `object.conf` is used to tell the code the execution parameter values and filenames (see Section 6). If not set, then `rvfit` expects to use six mandatory parameters (see Section 9).
- `outfile='object_param.out'` If set, the name of the output file will be the pointed out instead the default `rvfit.out`. Notice that the results from different fits will get overwritten in the default file `rvfit.out` unless you specify an outfile name.

- `evolfile='object.evol'` If set, the code saves in the file `object.evol` the values of the parameters after each acceptance. This checks the evolution of the parameter values as the algorithm approaches the minimum. This will slow down the execution time a bit.

As keyword parameters, we have:

- `/physics` This option calls the `computeRVparams.pro` procedure once `rvfit.pro` is finished to make the computation of the physical parameters and to plot the fitted RV curve with the data and the residuals. Sure, you may want to use always this option.
- `/autodom` If you don't know a suitable search range for the parameters, i.e, a search domain, you may try to use this option, which computes one range for you. Be careful if you use this option in the case of a system with short orbital period observed over a large interval. In this case, the code may not find the right fit.

8 Compilation

To compile `rvfit` in IDL type:

```
.comp rvfit computeRVparams
```

This command compiles `rvfit` and `computeRVparams` simultaneously. All should compile smoothly. Only if a file or procedure is missing an error will occur.

9 Running the program.

The code is designed to work in two ways:

- as a program used in an object-by-object basis. This option is useful if you have only a few objects to analyze. In this case, the best way to run the code is to generate a configuration file as described in Section 6, one for each object. Then you can run the program using:

  ```
  rvfit,configfile='object.conf',outfile='object_param.out',/physics
  ```

  In the configuration file you can set the suitable values for each object. In this case we also run the program setting the `outfile` parameter to `object_param.out`, which changes the name of the output file with the fit results. If this parameter is not set, then `rvfit` saves the solution in a file called `rvfit.out`.

- as a routine called from another program, like a batch file to process a list of several objects from a spectroscopic survey or observational program. In this case you can run `rvfit` as a procedure fed with variables containing the names of the RV data files or the data files themselves:

```
rvfit,'object_RV1.dat','object_RV2.dat', $
 [1,1,1,1,1,1,1], $
 [0.,0.,0.,0.,0.,0.,0.], $
 [0.,0.,0.,0.,0.,0.,0.], $
 [1.,1.,1.,1.,1.,1.,1.],/physics,/autodom
```

  Note that the six parameters passed to `rvfit` are the same which were saved in the configuration file. Also, note that the `fitparam` vector have all its elements set to 1 to fit all the parameters. As a consequence, the elements of `valparam` can hold any values. As an advice, you can set all of them to meaningless values to clearly show that they will not be fitted, e.g., 0.0.

  Note also the use of the option `/autodom` to set automatically the range of the parameter space (the domain). This is because we don't know beforehand their values. Thus we also set the elements of `L` and `U` to meaningless values, but you must remember to keep all the elements of `L` lower than of `U`.

After finishing running, `rvfit` gives the name of the output file where the resulting parameter values with uncertainties are saved and, if the `/physics` keyword is set, the outputs also include the values of those parameters, the derived physical quantities of the system, and a plot of the fitted RV curve with its residuals (see Section 12).

10 Computing the MCMC uncertainties

`rvfit` uses the Fisher matrix to compute an initial estimate of the uncertainties in the fitted parameters. We provide another program to compute Markov Chain Monte Carlo (MCMC) uncertainties if desired. Also, you can use your own MCMC to do this. The code we provide, called `MCMCerrors.pro` uses the output file saved to disk by `rvfit`. To run `MCMCerrors` you must type:

```
MCMCerrors,'object.out','object_RV1.dat','object_RV2.dat', $
  outfile='object_MCMC.out'
```

The first parameter called is the output file from `rvfit`, in this case `object.out`. The next two parameters are the same RV data files used in `rvfit`, called `object_RV1.dat` and `object_RV2.dat`, and the last parameter called is the name of the output file to save the MCMC chain, in this case `object_MCMC.out`. Remember that for a single-line object you must set the name of the secondary RV data file to `''`.

This code can be run with this optional parameters:

- `C=0.6827` This sets the percentage of samples in each confidence interval of the marginalized histogram to compute the uncertainties. If not set, the default is `C=0.6827` but you can use a `0.9545` confidence interval or a `0.9973` confidence interval.
- `nbins=50` This sets the number of histogram bins to the desired number. If not set, the default is `nbins=50`. The number of bins is used for discretizing the marginalized distribution for each parameter and for plotting the histograms.
- `nsample=500000L` This sets the length of the Markov Chain. If not set, the default is `nsample=100000L`. Note the `L` at the end of the number, since this is a 'long' integer. You also can use the format `nsample=ulong(1e6)` to set this parameter. This code doesn't discard initial chais since it begins their computation in the value found by `rvfit`.

While running, you will see the evolution of the histogram for each fitted parameter in a window for inspecting purposes (see Figure 2).

The output file of this tool is a table with this appearance:

```
0.4888800 2452724.01697 0.01036500 127.321619   -24.576906  142.295240  144.793683
0.4888800 2452724.01697 0.01036500 127.321619   -24.576906  142.295240  144.793683
0.4888800 2452724.01697 0.01036500 127.321619   -24.576906  142.295240  144.793683
0.4888687 2452724.01900 0.00971103 128.202881   -24.605307  142.555258  144.764000
0.4888687 2452724.01900 0.00971103 128.202881   -24.605307  142.555258  144.764000
0.4888687 2452724.01900 0.00971103 128.202881   -24.605307  142.555258  144.764000
```

In this table, the columns are sorted in the same order than in `rvfit`, i.e.:

```
[P,Tp,e,omega,gamma,K1,K2]
```

Note that if a parameter is not fitted, it will have a constant value in their column along the table.

11 Computing the physical parameters and plotting the RV curves

Usually, you will run `rvfit` setting the `/physics` option, which provides the physical parameters of the binary or exoplanet system. But if you need to recompute those parameters or plot the RV data with the fit and the residuals you can call the `computeRVparams` directly:

```
computeRVparams,readparfile='object_param.out', $
  rvdatafile1='object_RV1.dat', $
  rvdatafile2='object_RV2.dat',/fase,/ps,/latextable
```

This procedure has the following options:

- `/fase` Plots the RV curve in phase using the orbital period value fitted or provided.
- `/latextable` This saves to disk a file ended in `.tex` with the same information displayed in the screen but in a LaTeXtable for publication. Take a look.
- `/ps` This saves a Postscript file with the RV curve and the fit with residuals for publication. This is why `rvfit` uses the `pxperfect.pro` procedure.

12 Working example

This is a working example of a fit to the RV data of the double-line eclipsing binary GU Boo (López-Morales & Ribas, 2005). The configuration file for this example is:

```
rvfile1  = GUBoo_RV1.dat
rvfile2  = GUBoo_RV2.dat
fitparam = [1,          1,              1,       1,       1,      1,      1]
valparam = [0.4887280d, 2452723.9811d,  0.0d,    0.0d,   -24.57d, 142.65d,145.08d]
L        = [0.1,        2452723.9811d,  0.,      0.,     -100.,   0.,     0.]
U        = [10.,        2452724.9811d,  0.999,   360.,    100,    150.,   150.]
```

The RV data files are `GUBoo_RV1.dat` and `GUBoo_RV2.dat`. Notice the values of the `fitparam` set to 1 which states that all the parameters will be fitted. The values of `valparam`, `L` and `U` are set to those of the fit in (López-Morales & Ribas, 2005, Table 3) but in this case they are not being used, since all the parameters are being fitted.

Now we launch `rvfit` with the previous configuration file. The command issued is:

```
rvfit,configfile='GUBoo.conf',outfile='GUBoo_param.out',/physics
```

You will see an output similar to this in your screen:

```
Setting the initial Ta...
Initial Ta = 9030.38
Re-annealing...
Re-annealing...
Re-annealing...
Ta=7.1082E+03 param=[1.2151619,2452724.5754,0.055657,327.174,-31.213,142.294,130.666] chi^2=7108.2495
Re-annealing...
Re-annealing...
```

```
Re-annealing...
Re-annealing...
Ta=6.3988E+03 param=[1.6945986,2452724.0747,0.672956, 29.964,-14.294, 93.940, 71.048] chi^2=6398.7804
Re-annealing...
Re-annealing...
Re-annealing...
Ta=6.2033E+02 param=[0.51698117,2452723.9941,0.002304,146.432,-26.583,131.068,114.962] chi^2=620.32932
Ta=1.8209E-14 param=[0.51658774,2452724.0859,0.000001,183.908,-25.243,141.422,143.724] chi^2=133.14697
Ta=7.3218E-20 param=[0.51663777,2452724.0834,0.000000,185.475,-24.534,141.129,143.658] chi^2=131.81993
Re-annealing...
```

Each time a re-annealing is done a message is printed in the screen. When the temperature is decreased the acceptance temperature, the parameter values and the best χ^2 value found are printed. As the algorithm evolves toward the minimum, you will see how this χ^2 value gets smaller and the parameters values stabilize.

After a few seconds of computation the code ends, and produces a plot like the one shown in Fig. 1. The following information is also printed on the computer screen:

```
Result:
-------------------------------------------------
              Fitted parameters
-------------------------------------------------
P (d)               = 0.48885000 +/- 6.1000000e-05
Tp (HJD/BJD)        = 2452724.08595 +/- 0.0046060000
e                   = 0.0090660000 +/- 0.0074930000
omega (deg)         = 175.79346 +/- 5.2621380
gamma (km/s)        = -24.548910 +/- 0.71723100
K1 (km/s)           = 142.47152 +/- 1.3960170
K2 (km/s)           = 144.99616 +/- 1.4016130
-------------------------------------------------
              Derived quantities
-------------------------------------------------
M1sin(i)^3 (Msun)   = 0.60681779 +/- 0.013175756
M2sin(i)^3 (Msun)   = 0.59625202 +/- 0.013006363
q = M2/M1           = 0.98258822 +/- 0.013524576
a1sin(i) (10^6 km) = 0.95767933 +/- 0.0093848732
         (Rsun)     = 1.3760328 +/- 0.013484570
a2sin(i) (10^6 km) = 0.97464971 +/- 0.0094224931
         (Rsun)     = 1.4004166 +/- 0.013538623
asin(i) (10^6 km)   = 1.9323290 +/- 0.013298843
        (Rsun)      = 2.7764494 +/- 0.019108321
-------------------------------------------------
              Other quantities
-------------------------------------------------
chi^2               = 46.773599
Nobs (primary)      = 103
Nobs (secondary)    = 103
Time span (days)    = 9.3005000
rms1 (km/s)         = 4.4254676
```

```
rms2 (km/s)        = 5.0803908
Processed in 22.597659 seconds.
FIN.
```

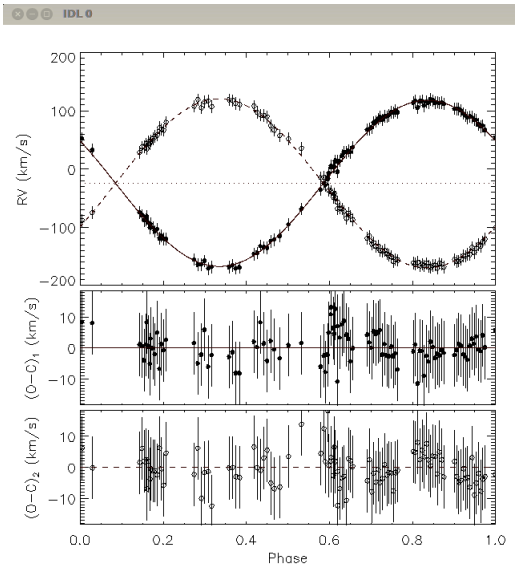

Figure 1: Result of the fit to GU Boo data.

Just below the computed parameters you will see the derived physical quantities for the system, in this case for a double-line binary. In the case of a single-line binary you will see the mass function and the $a_1 \sin i$ quantity (see Hilditch, 2001, p. 45)

Below the title 'Other quantities' you will see the best χ^2 value found, the number of observations for each RV curve (primary and secondary), the time span of the observations, the rms values for the residuals for each RV curve, and the total processing time.

Those fitted parameters are also saved in a file called `GUBoo_param.out`. If you take a look at this file you will see:

```
0.488850        0.000061        0.000000
2452724.085953  0.004606        0.000000
0.009066        0.007493        0.000000
175.793458      5.262138        0.000000
-24.548910      0.717231        0.000000
142.471521      1.396017        0.000000
144.996162      1.401613        0.000000
46.773599
103
103
9.3005000
```

The first seven lines are the parameter values and their uncertainties. The third

column is used in the case of asymmetric uncertainties, but, as `rvfit` only computes the Fisher matrix they are always symmetric. So the third column is 0.0 at this stage.

If you look at the uncertainty in eccentricity you will see that it is very similar to the e value. So we may presume that e can be zero and this orbit can be circular.

GU Boo is an eclipsing binary star, so we can fix the orbital period to the value from the photometry, as it has more precise time marks (the eclipses), and see what happens. The configuration file is now:

```
rvfile1  = GUBoo_RV1.dat
rvfile2  = GUBoo_RV2.dat
fitparam = [0,          1,              1,       1,       1,      1,      1]
valparam = [0.4887280d, 2452723.9811d,  0.0d,    0.0d,   -24.57d, 142.65d,145.08d]
L        = [0.1,        2452723.9811d,  0.,      0.,     -100.,   0.,     0.]
U        = [10.,        2452724.9811d,  0.999,   360.,    100,    150.,   150.]
```

Note the 0 value in the first element of `fitparam`. This states that the code must use the first element of `valparam` for the period.

If you repeat the `rvfit` command with this configuration file, you can get a result like this:

```
Result:
-------------------------------------------------
               Fitted parameters
-------------------------------------------------
P (d)               = 0.48872800 +/- 0.0000000
Tp (HJD/BJD)        = 2452724.42859 +/- 0.0077100000
e                   = 0.0000000 +/- 0.0064460000
omega (deg)         = 59.418623 +/- 5.5411690
gamma (km/s)        = -24.571545 +/- 0.71726600
K1 (km/s)           = 142.55902 +/- 1.3790820
K2 (km/s)           = 145.06977 +/- 1.3898500
-------------------------------------------------
               Derived quantities
-------------------------------------------------
M1sin(i)^3 (Msun)   = 0.60772980 +/- 0.013067118
M2sin(i)^3 (Msun)   = 0.59721172 +/- 0.012870749
q = M2/M1           = 0.98269282 +/- 0.013379385
a1sin(i) (10^6 km)  = 0.95806773 +/- 0.0092681188
         (Rsun)     = 1.3765909 +/- 0.013316812
a2sin(i) (10^6 km)  = 0.97494121 +/- 0.0093404852
         (Rsun)     = 1.4008354 +/- 0.013420791
asin(i) (10^6 km)   = 1.9330089 +/- 0.013158370
        (Rsun)      = 2.7774263 +/- 0.018906483
-------------------------------------------------
```

```
              Other quantities
-----------------------------------------------
chi^2              = 50.994517
Nobs (primary)     = 103
Nobs (secondary)   = 103
Time span (days)   = 9.3005000
rms1 (km/s)        = 4.7573830
rms2 (km/s)        = 5.1839000
Processed in 9.6678770 seconds.
FIN.
```

Note two things here: first, the e value is now exactly 0 with an uncertainty of 0.006; second, the ω value is 59 degrees but in the previous run was 175. These are hints of a circular orbit in this system, since for a circular orbit ω can adopt any value between 0 and 360 deg.

Lets run now `MCMCerrors` to see how the marginalized histograms behave near the solution found. To do this, first we need to compile the program:

```
.comp MCMCerrors
```

and then issue the following command:

```
MCMCerrors,'GUBoo_param.out','GUBoo_RV1.dat','GUBoo_RV2.dat', $
  outfile='GUBoo_MCMC.out'
```

In this case, you will see a multiple plot with two rows and seven plots in each row (see Figure 2). The upper row shows the marginalized histogram of each parameter, with the value found by `rvfit` marked as a solid vertical line, and the computed confidence interval marked by within the dashed vertical lines. The bottom row shows mixing plots for each parameter. The plots for the parameters not fitted are void. Please note that this plot is not intended for publication, just for analysis.

Now, look at the plots of T_P and ω: they have the same irregular and ill-behaved shape, spreaded over a large range in values (110 degrees in ω and half a period in T_P), and lack of a unique and clearly defined maximum. And the eccentricity plot piles up around zero. Clearly, this binary has zero eccentricity. The next step would be to repeat the analysis changing the `rvfit` configuration file and set `fitparam` to deal with a circular orbit:

```
rvfile1  = GUBoo_RV1.dat
rvfile2  = GUBoo_RV2.dat
fitparam = [0,          1,              0,       1,       1,      1,      1]
valparam = [0.4887280d, 2452723.9811d,  0.0d,    0.0d,   -24.57d, 142.65d,145.08d]
L        = [0.1,        2452723.9811d,  0.,      0.,     -100.,   0.,     0.]
U        = [10.,        2452724.9811d,  0.999,   360.,    100,    150.,   150.]
```

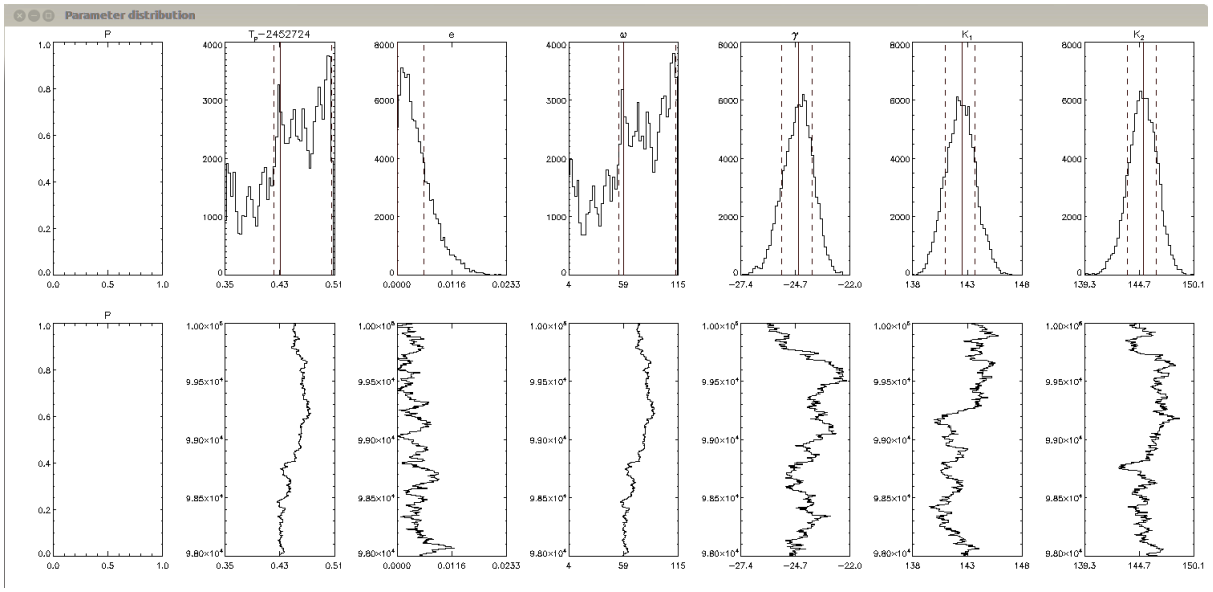

Figure 2: Marginalized histograms shown during the MCMC run.

Note that if you set the `fitparam` value for the eccentricity to 0 but fit ω, the value of ω will be fixed to 90 degrees internally.

The last step will be to analyze the MCMC chain again to measure the uncertainties in the fitted parameters. Usually, the marginalized histograms will be nearly gaussian, so you can fit a gaussian to them and take the width σ as a estimate of the uncertainties. If that is not the case, then you can compute the confidence interval to some X level.

If you are happy with your fit and you want to write a paper for Nature and win a Nobel Prize, you can obtain the LaTeXtable with your solution and the Postscript plot of the fitted RV curve running again `computeRVparams` as follows:

```
computeRVparams,readparfile='GUBoo_param.out', $
  rvdatafile1='GUBoo_RV1.dat', $
  rvdatafile2='GUBoo_RV2.dat',/fase,/ps,/latextable
```

13 Important tips

- `rvfit` works best when the solution is properly constrained. The period is the most important parameter here. The better the period is constrained, the faster the convergence will be. This is because the period is the parameter that has the smallest 'value'-to-'search range' ratio, so it can be ill conditioned: some observing runs can last months or years while the period would be only of a few days or hours. Thus, that ratio will be $\sim 10^{-3}$ or less. This makes the parameter space hard to search for the algorithm. Lets think that a small change in the period can affect strongly the χ^2 value. So, usually, the solution period would be hidden at the bottom of a very narrow valley or hole in the parameter space. If the

parameter space is well constrained it is easy to find the hole because the search region is smaller and better sampled. You can increase the internal parameter `Ngen` to account for such a situation but this slows down the algorithm. If you are looking for a period of days and your observing runs spans over years, don't set the `/autodom` keyword. Instead, constrain the period in the `L` and `U` vectors to, for example, between 1 and 10 days. As a rule of thumb, if the time span of your observations is about 10 times greater than the period you are looking for, then better try not setting `/autodom`.

- In a system with circular orbit (zero eccentricity) the parameter ω is meaningless. If such a system is fitted, and you leave e and ω free, the value of ω will be in accordance with the small eccentricity you will get. But the uncertainties in ω and T_P will presumably be large. If you fix the eccentricity to zero, ω will be internally fixed to 90 degrees. This only happens if you fix the eccentricity to zero, not to any other value.

- This code implements an heuristic method, not a deterministic method. There isn't any guarantee that the obtained solution is the best one. Even so, some results show that when the number of iterations tends to infinity and the temperature (the acceptance temperature and the generating temperature) tends to zero the global optimization target is always found.

- "garbage in, garbage out", i.e., if you entered weird parameter values you will obtain weird results. Not only with this code but with any code. For example, if by a mistake you entered an eccentricity greater or equal than 1 you will have troubles.

- I built this code for my thesis with (eclipsing) binary stars in mind. Be careful if you use it with exoplanets. I checked some published single exoplanetary systems and the results are in agreement within uncertainties with the published ones. But exoplanet systems are not binaries. For more information see Iglesias-Marzoa et al. (2015).

Bibliography

Ford, E. B., 2005, AJ, 129, 1706

Giménez, A., 2006, ApJ, 650, 408

Hilditch, R. W., 2001, *An Introduction to Close Binary Stars*, Cambridge University Press

Iglesias-Marzoa, R., López-Morales, M., Arévalo, M.J., 2015, PASP, accepted, arXiv:1505.04767v1.

Ingber, L., *Adaptive simulated annealing (ASA): Lessons learned*, 1996, Control and Cybernetics, 25, N1, 33

López-Morales, M., Ribas, I., 2005, ApJ, 631, 1120

Ohta, Y., Taruya, A., Suto, Y., 2005, ApJ, 622, 1118

Sybilski, P., Konacki, M., Kozłowski, S. K., Helminiak, K. G., 2013, MNRAS, 431, 2024