exostriker Documentation

Release latest

Trifon Trifonov

Nov 02, 2022

INSTALLATION

CHAPTER ONE

REVIEW

Transit and Radial velocity Interactive Fitting tool for Orbital analysis and N-body simulations: The Exo-Striker.

The Exo-Striker analyzes exoplanet orbitals, performs N-body simulations, and models the RV stellar reflex motion caused by dynamically interacting planets in multi-planetary systems. It offers a broad range of tools for detailed analysis of transit and Doppler data, including power spectrum analysis for Doppler and transit data, Keplerian and dynamical modeling of multi-planet systems, MCMC and nested sampling, Gaussian Processes modeling, and a long-term stability check of multi-planet systems. The Exo-Striker can also perform Mean Motion Resonance (MMR) analysis, create fast fully interactive plots, and export ready-to-use LaTeX tables with best-fit parameters, errors, and statistics. It combines Fortran efficiency and Python flexibility and is cross-platform compatible (MAC OS, Linux, Windows).

CHAPTER

TWO

FEATURES

- Keplerian and Dynamical modeling of RV & Transit photometry exoplanet data.
- RV signal and alias search: via GLS periodogram & maximum lnL periodogram (MLP).
- Activity index signal search via GLS periodogram.
- RVs vs. Activity time-series correlation analysis/plots.
- RV auto-fit (RV automated planet-finder algorithm).
- Transit signal search (via “TLS”).
- Interactive transit photometry detrending (via “wotan”), interactive outlier removal, and more.
- Linear models for detrending ground-based transit photometry.
- GP modeling (via “celerite”).
- Joint RVs + Transit + GPs best-fit optimization (internal Fortran Simplex and L-M minimizers, or many more via “SciPyOp”).
- Joint RVs + Transit + GPs MCMC/Nested Sampling (via “emcee” & “dynesty”).
- TTVs extraction.
- TTVs and/or joint TTVs + RVs analysis.
- Fit for apsidal orbital precession, or apply General Relativity (GR) precession.
- Long-term stability check of multiple planetary systems using SyMBA, MVS, and MVS with a GR precession.
- Instant AMD stability check for multiple planetary systems (including during optimization or MCMC/Nested Sampling).
- Multi-platform: It works on MAC OS (10.6+), Linux (Suse, Mint, Ubuntu, etc.) and Windows 10.
- Import/Export of work sessions and multi-sessions.
- Export plots to a matplotlib window for further customization.
- Export ready to use LaTeX tables with best-fit parameters, errors, and statistics.
- Print the GUI screen into a .jpeg/.png image (useful for sharing quick results).
- Fully interactive, super-fast, high-quality, exportable plots.
- Integrated Bash-shell (Linux only).
- Integrated Jupyter shell.
- Handy “cornerplot” GUI control.

- Instant online access to the “RVBank” database (over 212 000 RVs and activity indices of about 3000 HARPS stars & over 64 000 RVs and activity indices of about 1700 HIRES stars).
- Handy text-editor and calculator tools.
- Direct import of TESS lc.fits and CHEOPS SCI_COR.fits files.
- Importable as a standard python library (i.e., “import exostriker”).

CHAPTER

THREE

WHAT IS TO BE IMPLEMENTED

- Larger arsenal of N-body/dynamical simulation/analysis tools (+ “REBOUND” is planned to be included).
- Internal TTV and photo-dynamical modeling (i.e. the external “TTVFast” will become a secondary option).
- Swap “celerite” with “celerite2” (the dSHO kernel from “celerite2” is available).
- Combined modeling with Astrometry (work in progress).

CHAPTER

FOUR

DEVELOPERS

- **Trifon Trifonov**, MPIA Heidelberg.
- with contributions by **Mathias Zechmeister**, **Jakub Morawski**, **Man Hoi Lee**, **Stefan Dreizler**, **Grigorii Smirnov-Pinchukov**, **Stephan Stock**, **Jonas Kemmer**, **Harry Psarakis** and **Desislava Antonova**.

4.1 Linux

Prerequisites

1. To **install** the Exo-Striker, first please make sure that you have installed the following packages:

- csh
- gfortran
- pip

2. Use **Python 3** for installing the Exo-Striker! The tool works with Python 2; however as of Jan 1, 2020, Python 2 is no longer being maintained and eventual problems may not be addressed.
3. Do not uninstall the Python version which was default to your system, this may cause unexpected side effects. If you have an older version of Python, keep it and install the newer one as well.

Currently there are **three ways to install/run** the Exo-Striker:

- The **first way** is to **git clone** using the Linux command line:

```
git clone https://github.com/3fon3fonov/exostriker
```

```
cd exostriker
```

and then **load the GUI**:

```
python3 exostriker_gui.py
```

or

```
./exostriker_gui.py
```

This installation process clones the repository in a local folder and installs the tool locally.

To update the tool:

```
git pull
```

- The **second way** to install the tool from the source:

```
git clone https://github.com/3fon3fonov/exostriker
```

and then:

```
cd exostriker
```

```
python3 setup.py install
```

This command will initiate a global installation on your system.

Then,

```
python3 exostriker_gui.py
```

This should start the Exo-Striker GUI

- and **third way** is to **pip3 install**:

```
pip3 install git+https://github.com/3fon3fonov/exostriker
```

This command will initiate a global installation on your system.

Then,

```
python3 exostriker_gui.py
```

or

```
exostriker
```

This should start the Exo-Striker GUI.

In order to update the tool, when a new version is available, the syntax is:

pip3 install git+https://github.com/3fon3fonov/exostriker -U

Note: When installing and running for the first time, some additional libraries are being decompiled and it takes several minutes more to start the GUI.

Generally, **you do not need to install anything if you already have all the required dependencies** for the tool to run. The dependency list, including the version of each dependency, can also be seen at the "setup.py" file.

Important: List of required **dependencies** :

- numpy
- scipy
- matplotlib
- PyQt5
- jupyter
- pathos
- emcee
- celerite
- qtconsole

- dynesty
- wotan
- corner
- transitleastsquares
- ttvfast-python

In some occasions it will be necessary to manually install packages/modules such as tqdm, wotan, batman, etc, which usually depends on the Linux distribution and Python version.

To do so use pip3, eg.

```
pip3 install tqdm
```

For more details visit *Common issues and solutions*.

Tip:

- Install the simple and extensible rxvt terminal emulator:

See rxvt documentation here: https://wiki.archlinux.org/title/rxvt-unicode

- Install Anaconda package which manages the dependencies and may resolve installation issues.

See Anaconda documentation here: https://docs.anaconda.com/anaconda/install/.

4.2 Windows 10&11

Prerequisites:

1. **Windows subsystem for Windows - WSL & WSL2**

   Installation on Windows 10 and 11 works only troughs the "Windows Subsystem for Linux".

   Please follow this guide:

   Install WSL

   Follow all the steps carefully! This way you will be able to run all Linux native programs on your WINDOWS 10 & 11 bash shell, which is very useful in general!

   Please install the Ubuntu 20.04 LTS !!!

2. **XServer**

   To make The Exo-Striker work, however, you also will need an XServer installed.

   Follow these instructions carefully:

   Install XServer

3. **Update the Linux kernel**

   After installing Ubuntu 20.04 LTS app on your Windows OS, open the app and update the Linux kernel:

   ```
   sudo apt update
   ```

```
sudo apt dist-upgrade -y
```

When done, write the following lines in the Linux Terminal:

```
sudo apt install build-essential curl git gfortran gcc+ csh xterm
```

4. **Fix the Qt libraries**

   For some unknown reason, some qt binaries in Ubuntu 20.04+ are missing. Just in case, run all these commands:

   ```
   sudo apt install libxcb-xinerama0
   ```

   ```
   sudo apt install libxkbcommon-x11-0
   ```

   ```
   sudo apt install qt5-default
   ```

5. **Python3.8 and pip3.8**

   Python 3.8+ is strongly recommended!!! On your WSL on Windows and Ubuntu 20.04, you will likely have Python 3.8.2 installed, but to install the newest Python 3.8.13 (or later) alongside your system Python 3.8.2, you can try these instructions (for 18.04 but should be OK for 20.04):

   Install Python

   Let's assume, you now have Python3.8, then it is recommended to install pip3.8:

   ```
   curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
   ```

   ```
   python3.8 get-pip.py
   ```

6. **Python-dev**

   For yet another unknown reason, a fresh Ubuntu 20.04 does not include the Python Development files.

   ```
   sudo apt install python3-dev
   ```

7. **Install Exo-Striker**

   At this point, your system should be clean of all problems preventing the Exo-Striker being installed. Keep in mind, your python3.8 system is likely quite "empty". To install all the dependencies in one-go, the recommended way is:

   ```
   pip3.8 install git+https://github.com/3fon3fonov/exostriker --user
   ```

   /for user installation – recommended !!!!!!/

   Then just run:

   ```
   exostriker
   ```

Note: Probably the GUI screen scaling will be 150%. Go to "Change the resolution of the display" and reduce it to 100% and the GUI window will look fine.

Bonus tip

It is generally recommend to work with git clone / git pull to always get the new version - e.g.:

```
cd ~
```

```
git clone https://github.com/3fon3fonov/exostriker
```

```
cd exostriker
```

```
python3.8 exostriker_gui.py
```

The latter command will open the GUI inside the root "exostriker" directory and will be easy to find your sessions and files. One can even rename the directory:

`mv exostriker exostriker_GL486`

or

`cp -r exostriker exostriker_GL486`

`cp -r exostriker exostriker_GL876`

`cp -r exostriker exostriker_GL1148`

etc.,

Your analysis of GL486 will be stored in ~/exostriker_GL486. To work and/or update the repository:

`cd exostriker_GL486`

`git pull`

`python3.8 exostriker_gui.py`

If problems occur:

`python3.8 exostriker_gui.py -debug`

and open an issue on the GitHub repository.

4.3 Common issues and solutions

If the GUI does not start after a successful installation, please try:

`python3 exostriker_gui.py -debug`

This command starts the GUI and at the same time outputs an error message with few guidelines, which are otherwise displayed in the 'stdout/stderr' tab of the GUI.

1. Broken appearance of GUI under Windows

In case there is a problem of the appearance of the GUI the problem could be your DPI setup. On high DPI displays with Windows display scaling activated (>100%), the X-server DPI has to be set, otherwise, the Exo-Striker will not display correctly (text clipping). Edit the file *.Xresources* in the home directory of the WSL installation, for example with *nano ~/.Xresources*. Add the line *Xft.dpi: 100* and save & close with Ctrl+O Ctrl+X, then reload the X configuration with *xrdb ~/.Xresources*. On Ubuntu WSL you might need to install *x11-xserver-utils* with *sudo apt install x11-xserver-utils* for xrdb to be available.

Launch the Exo-Striker and check the scaling. If text is clipping, the DPI needs to be set lower, if everything is too small, the DPI needs to be higher. Remember always to reload the configuration with *xrdb ~/.Xresources*. For the configuration to automatically load at startup, add the xrdb command to your ~/.bashrc, after the *export DISPLAY=:0.0*.

2. Broken appearance of GUI under Linux

A computer display with low resolution may not properly display the interface of Exo-Striker (text clipping, missing plot parts, missing buttons). There is a workaround using Xrandr.

`xrandr -q`

The command will output current monitor and available resolutions.

Set the necessary parameters, for example:

`xrandr --output eDP-1 --mode 1366x768 --scale 1.20x1.20 --panning 1920x1080`

Here eDP-1 is the current monitor, mode is the current resolution, scale is the scale factor for desired resolution, panning is the whole area available with scrolling after the new resolution is set. If panning is not used, the whole screen fits in the display window.

More issues and more thorough explanations and solutions are available in the *Issues section* of the GitHub repository.

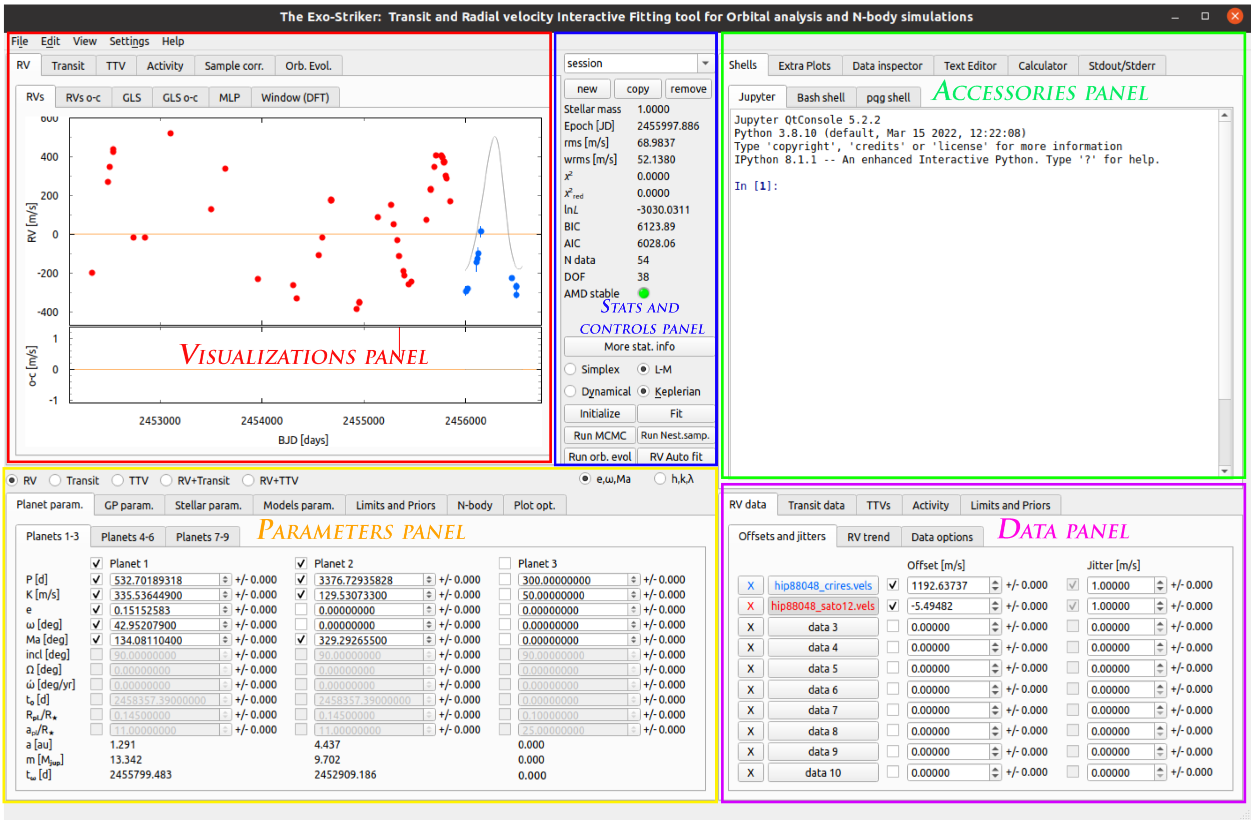
4.4 Report an issue

If you run into issues or bugs, do not hesitate to report a **New issue** on the GitHub repository or send a PM to trifonov@mpia.de.

Feedback and **help** in further development will be highly appreciated! A wish-list with your favorite tools and methods to be implemented is also welcome!

4.5 GUI Layout

This documentation provides information regarding the basic parameters that Exo-Striker uses to determine the goodness of a fit but also the ones that describe the planets orbitals. All of these parameters can be seen on the frontend of the GUI.



Stellar mass	1.0000
Epoch [JD]	2455997.886
rms [m/s]	68.9837
wrms [m/s]	52.1380
χ^2	0.0000
χ^2_{red}	0.0000
ln$L$	-3030.0311
BIC	6123.89
AIC	6028.06
N data	54
DOF	38
AMD stable	🟢

```
Jupyter QtConsole 5.2.2
Python 3.8.10 (default, Mar 15 2022, 12:22:08)
Type 'copyright', 'credits' or 'license' for more information
IPython 8.1.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]:
```

	✓	Planet 1		✓	Planet 2		☐	Planet 3	
P [d]	✓	532.70189318	+/- 0.000	✓	3376.72935828	+/- 0.000	☐	300.00000000	+/- 0.000
K [m/s]	✓	335.53644900	+/- 0.000	✓	129.53073300	+/- 0.000	☐	50.00000000	+/- 0.000
e	✓	0.15152583	+/- 0.000	☐	0.00000000	+/- 0.000	☐	0.00000000	+/- 0.000
ω [deg]	✓	42.95207900	+/- 0.000	☐	0.00000000	+/- 0.000	☐	0.00000000	+/- 0.000
Ma [deg]	✓	134.08110400	+/- 0.000	✓	329.29265500	+/- 0.000	☐	0.00000000	+/- 0.000
incl [deg]	☐	90.00000000	+/- 0.000	☐	90.00000000	+/- 0.000	☐	90.00000000	+/- 0.000
Ω [deg]	☐	0.00000000	+/- 0.000	☐	0.00000000	+/- 0.000	☐	0.00000000	+/- 0.000
$\dot{\omega}$ [deg/yr]	☐	0.00000000	+/- 0.000	☐	0.00000000	+/- 0.000	☐	0.00000000	+/- 0.000
t_0 [d]	☐	2458357.39000000	+/- 0.000	☐	2458357.39000000	+/- 0.000	☐	0.00000000	+/- 0.000
$R_{pl}/R_\star$	☐	0.14500000	+/- 0.000	☐	0.14500000	+/- 0.000	☐	0.10000000	+/- 0.000
$a_{pl}/R_\star$	☐	11.00000000	+/- 0.000	☐	11.00000000	+/- 0.000	☐	25.00000000	+/- 0.000
a [au]		1.291			4.437			0.000	
m [M_{jup}]		13.342			9.702			0.000	
t_ω [d]		2455799.483			2452909.186			0.000	

			Offset [m/s]			Jitter [m/s]	
X	hip88048_crires.vels	✓	1192.63737	+/- 0.000	✓	1.00000	+/- 0.000
X	hip88048_sato12.vels	✓	-5.49482	+/- 0.000	✓	1.00000	+/- 0.000
X	data 3	☐	0.00000	+/- 0.000	☐	0.00000	+/- 0.000
X	data 4	☐	0.00000	+/- 0.000	☐	0.00000	+/- 0.000
X	data 5	☐	0.00000	+/- 0.000	☐	0.00000	+/- 0.000
X	data 6	☐	0.00000	+/- 0.000	☐	0.00000	+/- 0.000
X	data 7	☐	0.00000	+/- 0.000	☐	0.00000	+/- 0.000
X	data 8	☐	0.00000	+/- 0.000	☐	0.00000	+/- 0.000
X	data 9	☐	0.00000	+/- 0.000	☐	0.00000	+/- 0.000
X	data 10	☐	0.00000	+/- 0.000	☐	0.00000	+/- 0.000

Fig. 1: *Frontend of the GUI.*

Action menu bar

- *File* :

  New session: New session can be opened from the **new** option in the **Stats and controls panel**.

  Open session: Open an already created session from a .ses file.

  Open multi session: Open an already created session from a .mses file.

  Save session: Save the current session in a .ses file.

  Save multi session: Save all opened sessions in an .mses file.

  Open RVmod init file:

  Open RVBank file: *In developer mode!*.

  Quit: You will be prompted to save your session.

- *Edit*:

  Push Jupyter var. to GUI:

  RV Auto fit: Automatically fit the radial velocity data, an option for auto-fit is also available in the Stats and controls panel.

  Reset mid-panel: *In developer mode!*.

- *View*:

  Print window GUI to file.: An option to save the current GUI window in a .png file. Does not output the file.

  Print f-test FAP: Prints hypothesis testing statistical information in the **Stdout/Stderr** window of the Accessories panel.

  Get LaTeX table with parameters: Outputs a command line to the **Jupyter shell** where table parameters can be modified. When executed, the LaTeX table with best fit parameters is saved to **path** as a .tex file.

  Get LaTex table with priors: Outputs a command line to the **Jupyter shell** where table parameters can be set. When executed, the LaTeX table with priors is saved to **path** as a .tex file.

  Get RV model: Outputs a command line that saves a .txt file with modeled radial velocities in m/s per epoch in days.

  Get RV data: Outputs a command line that saves a .txt file with data points information as follows:

    - *BJD [days]*
    - *RVs [m/s]* **or** *RVs o-c [m/s]* with or without jitter and offset added to it, depending on the command line setting.
    - *RVs errors [m/s]*
    - *Rank* - The index number of dataset starting from 0, that can be modified in the command line.

  Get Orb evol.: Outputs a command line that saves a .txt file with data as follows:

    - *Rank*
    - *Time[years]*
    - *Semi-major axis[au]*
    - *Eccentricity*
    - *Argument of periastron[deg]*

    - *Mean anomaly[deg]*
    - *Inclination[deg]*
    - *Longitude of the ascending node[deg]*

  Get All plots: All plots from the analysis are exported to "exported_plots" folder on your local exostriker directory. Note that in order for the plots to be exported properly, each plot should be opened in advance.

  Confidence interval table.: A new window pops-up with confidence intervals and a Chi-square distribution table below with Chi-values for each P-value/degree-of-freedom pair.

- *Settings*:

  Change widget style: Change style depending on your OS.

  Set GUI font: Change font style and size of the GUI.

  Set plots font: Change font style and size of the plots.

- *Help*:

  Exostriker page on Github: Opens the Git hub page of the project in a browser.

  Credits: Opens a pop-up window with credits and utilized packages information.

Note: If the GUI freezes or crashes, the most recent session is saved in the **autosave** folder in the **Exostriker** directory after each click on **fit** or **initialize** options, and can be initialized with the following command:

```
python3 exostriker_gui.py -last
```

4.5.1 Plotting widgets panel

This panel provides a large number of visualizations of transit and radial velocities data and residuals in the frequency domain, stellar activity, sample correlations, evolution of orbital parameters over time. The plots in this panel are interactive and can be expanded, narrowed, zoomed in and out, and exported. If the plot's dimensions are changed, a small button with the letter **A** appears on the left of the plot which restores the default view when clicked.

- ***Options menu*: Available with right click on the plot.**
    - **View all**: Click to zoom the plot to default view (same as the **A** button).
    - **X axis**: Options for modification of the 'x' axis.
    - **Y axis**: Options for modification of the 'y' axis.
    - **Mouse mode**: Chose between one or three options to control the plot with the mouse.
    - **Plot options**: Modify the entire plot.
    - **Export…**: Each plot can be exported as a separate image file, vector graphics, matplotlib file, csv, etc. There is an option to save one of the plots in a window or both (entire scene option), which is still not available for export in matplotlib.
- RV

Parameter name	Description
RVs	Radial velocities time series.
RVs o-c	Radial velocities residuals as a function of time (BJD)
GLS	Generalized Lomb-Scargle periodogram of the initial signal
GLS o-c	Generalized Lomb-Scargle periodogram of the residual signal
MLP	Maximum Likelihood periodogram
Window (DFT)	Discrete Fourier Transform periodogram

For more detailed information check the *Radial Velocity data* section.

- Transit

Parameter name	Description
Tran.	Transits time series - relative flux as a function of BJD
Tran. o-c	Transits residuals
TLS	Transit Least Squares of the initial signal, (Signal Detection Efficiency (SDE)/period[d])
TLS o-c	Transit Least Squares of the residual signal, (SDE/period[d])

TLS is also periodogram but for transit data.

For more information check the *Transit data* section.

- TTV

Parameter name	Description
TTVs	Transit-timing variations
TTVs o-c	TTVs residuals

- Activity

Parameter name	Description
Time series	Activity time series
GLS	Generalized Lomb-Scargle periodogram of the activity indicator
Correlations	Check the correlation between the RV data and the RV indicators

- **Sample correlation**

  This sections is for visualization of correlations between chosen parameter samples that are generated through the MCMC or Nested Sampling algorithms. These samples have to be generated in advance by running MCMC or Nested sampling.

  For more information check *Parameter Posterior Distributions* section.

- Orbital Evolution

Parameter name	Description
Orb.elem	Tabs with orbital elements: semi-major axes, eccentricities, argument of periastron, inclination/omega, energy
Orb. view	Graphical representation of the planets' orbits
Res. angles	Tabs: Period evolution, Delta omega, Res. angle (theta)

Orbital parameters evolution time series.

For more information check *Stability analysis* section.

Additional controls:

- A radio button provides a choice to display the signal power as a function of period [d] or frequency [1/d]. Usually the period is plotted on the X axis instead of frequency for better visualization of the peaks.
- A button **Print info** is also available and it outputs statistical information, FAP (False alarm probability) levels and the first 10 strongest peaks.
- A **Cross Hair** check box when checked helps to find the power and period corresponding to a given peak and also to inspect alias peaks if *show aliases in cross hair* is enabled in **Plot options** > **GLS/MLP/TLS** tab.

4.5.2 Stats and controls panel

- **Control Sessions**: Navigate through all active sessions with the drop-down menu, or create new session, copy/remove the current session. Navigating through several sessions is convenient for comparative analysis.
- **Statistical parameters**

Parameter name	Description
rms [m/s]	Root-mean-square
wrms [m/s]	Weighted root-mean-square
^{2}	Chi-squared
$^{2}_{\text{red}}$	Chi-squared reduced
lnL	Log-likelihood function
BIC	Bayesian information criterion
AIC	Akaike information criterion
N data	Number of data/observations
DOF	Degrees of freedom
AMD stable	Checking the stability of a system (Green/Red)

- **More stat.info**: Provides information about the fit quality & RV data rms/wrms.
- **Control parameters**

Parameter name	Description
Simplex	Fitting curves using the Simplex algorithm.
L-M	Fitting curves using the Levenberg-Marquardt algorithm.
Keplerian	Perform a Keplerian analysis.
Dynamical	Perform a Dynamical analysis.
Initialize	Fitting any change without optimizing (pressing Enter).
Fit	Optimization parameter
Run MCMC	Triggers samples using the Markov chain Monte Carlo algorithm.
Run Nest.samp	Triggers samples using the Nested sampling algorithm.
Run orb. evol.	Perform orbital parameter evolution.
RV auto fit	RV automated planet-finder algorithm.

4.5.3 Help widgets panel

- **Shells**

  Exostriker offers 3 command-line interpreters.

  Jupyter: An Qt-based interactive Python interpreter for working with Jupyter kernels. It provides a number of enhancements only possible in a GUI, such as inline figures, proper multi-line editing with syntax highlighting, graphical call-tips, and much more. For more information visit qtconsole documentation.

  Bash shell. Integrated bash shell which can be used from inside the GUI for convenience when for example working with directories and files.

  pqg shell: Shell based on the PyQtGraph graphics and user interface library for Python. It's advantage is that it offers one-click command history review and exceptions handling. For more information visit pyqtgraph documentation.

- **Extra plots**

  In this section plots of the most prominent peaks of the RV data are displayed phase folded (phase diagrams). A slider button is available to look through phases for each of the planets found. Additionally, periodograms of the RV data are included.

  Fig. 2: *Extra plots.*

- **Data inspector**

  Inspect the data on your local machine through the options **This computer** or **RVBank** and load them to exostriker. The plots are interactive and the information can only be informatively displayed without being loaded to Exo-Striker. The pop-up data window can be closed after the data is selected.

  Fig. 3: *Data inspector.*

  – *This computer*: Offers a fast visualization of graphical data available in a local directory, as well as review of statistical data related to it with the **Print info** button.

  – *RVBank*: Contains the whole RV spectrographic data available from *HARPS RVBank* and *HIRES NZP* up to 2018. Different types of **RV data** sets (RVs SERVAL + NZP correction etc.) and stellar **Activity indicators** such as CRX, dLW, Halpha, FWHM_DRS, bisector, etc can be selected for analysis. Certain stellar activity patterns can resemble a planetary signal and lead to false positives.

  Fig. 4: *RVBank.*

  Activity indicators can also be modified.

  – *Cross hair*: Show intersection of X and Y value.

  – *Export*: GUI crashes if there is no data in the plot area.

  – *Load to ES*: Load RV data from your computer or from RVBank to the **Data panel** for analysis. After loading, the data is displayed also in the **Visualizations panel**. in the If you try to load Transit data, the GUI crashes.

  – *Print info*: Outputs data log with information such as file folder path, data count, statistical data, etc.

Fig. 5: *Activity indicators.*

- **Text editor**

  Through the *text editor* you can inspect and perform a quick edit of data files, scripts, Latex files, etc.

Fig. 6: *Text editor.*

- Calculator
- Stdout/Stderr

  Version of GUI, progress of all processes, calculated parameters and error messages are displayed in this window.

  If GUI is initialized with:

  ```
  python3 exostriker_gui.py -debug
  ```

  all error messages appear in the terminal instead of in the Stdout/Stderr. This can be useful to analyze reasons for the program to crash in some occasions.

  Warning: Before starting any project make sure that you run the latest version of *exostriker*. You can be updated about the latest version/updates of *exostriker* on Exo-Striker's github page.

4.5.4 Input/Output parameters panel

- Planet parameters

  Include the parameters of up to 9 planets in a planetary system. These are the parameters that are updated and optimized when fitting the model. The values on the right of the parameter boxes are the model uncertainties which are optimized only with the Levenberg-Marquardt algorithm, but not with the Simplex algorithm. The parameters can also be manually changed.

  Note: It is possible for the model to optimize only those parameters that have the tick-box on the left **checked**!

  Depending on which radio-button on the left is selected (RV,Transit, TTV, RV+Transit, RV+TTV), different parameters are active.

Input/Output parameters

Parameter name	Description
P [d]	The planet's period
K [m/s]	RV amplitude
e	Orbital eccentricity
[deg]	Argument of periastron
Ma [deg]	Mean anomaly at the first observational epoch
inc [deg]	Inclination
[deg]	Longitude of the ascending node
[deg/yr]	Rate of argument of periastron
t_0[d]	Time of the first transit
R_{pl}/R*	Planet's radius in units of stellar radius
a_{pl}/R*	Planet's semi-major axis in units of stellar radius
a [au]	Semi-major axis of the planet's orbit in astronomical units
m [**M_{jup}**]	Mass of the planet in units of one Jupiter mass
t[**d**]	Time of periastron passage

Depending on the radio-buttons on the upper-right side of the panel, the planet parameters change as follows:

- **e** changes to **h=esin()**
- changes to **k=ecos()**
- **Ma** changes to **[deg]**, the mean longitude
- **GP parameters.**

  Gaussian processes parameters. Gaussian processes include statistical instruments for modeling the stochastic variations of signals with a wide range of sources. GP based algorithms can be very useful for modeling stellar activity, because it searches for correlated noise and includes it in the model.

  - RV GP
  - Transit GP

Both RV and Transit Gaussian processes include the same kernel options:

Tab	Description
SHO Kernel	Simple harmonic oscillator kernel
dSHO Kernel	damped Simple harmonic oscillator kernel
Rot. Kernel	Kernel of rotation matrix **?**
Matérn=3/2	Matérn covariance function
DRW kernel	Damped Random Walk kernel

More information regarding the kernel and the Gaussian processes as a whole is available with the **READ ME** option which redirects to scientific articles.

- Stellar parameters

  In case the host star characteristics, including their standard deviations are known, this tab allows for manually editing them and enhances model accuracy. * Stellar mass in units of Solar mass * Stellar radius in units of Solar radius. * Stellar luminosity in units of Solar luminosity * Effective temperature of the star * v sin i in km/sec - projected trajectory velocity. Knowing this parameter we can estimate the star rotation velocity.

- Models parameters

  - **Models**

  RV Model

You can choose between Fortran77 and SciPy only for radial velocity data. In case of transit data the minimizer is always SciPy!

RV Fort. param.: still in development.

- **SciPy parameters**

Set minimizer parameters when fitting transit data. Usually the Truncated Newton (TNC) and Nelder-Mead methods are used.

- **GLS/MLP/TLS parameters**

For most options use the default values.

In **GLS**:

Min. period [d]: option to remove aliases from daily measurement patterns if setting the minimal orbital period to more than 1 day.

Max. period [d]: option to put constrains on the maximum orbital period.

GLS oversampling factor: set higher for smooth periodogram.

- **MCMC parameters**

Burning phase samples: This is the number of the samples for the initial "warm-up" phase of the algorithm execution. Usually 100.

MCMC phase samples: A greater number of samples means higher accuracy of the model results; however the algorithm has a high computational cost and requires high performance machines. A reasonable number is 5000, is executed on a personal computer.

N threads/CPUs: Number of processors used, depending on your system. Leave as is.

Init. Gaussian Ball:

N walkers factor [* N dim.]: Determines the model dimensions. The number of walkers is multiplied by the number of parameters that the model is optimizing, including planet parameters for all planets, and offset and jitter for all data sets.

There are several other options for visualization of the model's progress in the Stdout/Stderr window, as well as choice between several statistical parameters to represent the MCMC results. If **Use the start param** is selected, the initial planetary orbit parameters do not change.

Go to "Make cornerplot": redirects you to the **Plot opt** tab for setup of the MCMC plot options.

READ ME option redirects to a help window with a link to the user guide of the **emcee** package used for implementation of the MCMC algorithm.

- **Nested Sampling parameters**

Static/Dynamic: choose Dynamic modeling to include orbital evolution.

Dynesty samp. out.: choose dynesty algorithm, usually the default rwalk is used.

N threads/CPUs: Depending on your system. Leave as is.

dlogz stop: if the change of logz is less than this number, the algorithm will stop. Depending on the other parameters this can take fairly long computational time.

Live points factor [* N dim.]: Determines the model dimensions. The number of walkers is multiplied by the number of parameters that the model is optimizing, including planet parameters for all planets, and offset and jitter for all data sets.

There are several other options for visualization of the model's progress, as well as choice between several statistical parameters to represent the NS results. If **Use the start param** is selected, the initial planetary orbit parameters do not change.

Go to "Make cornerplot": redirects you to the **Plot opt** tab for setup of the NS plot options.

The check boxes on the lower right can also be used to constrain the modeling time no matter if the algorithm has converged, by setting the maximum number of its iterations or calls to it.

READ ME option redirects you to a help window with a link to the user guide of the **dynesty** package used for implementation of the NS algorithm.

- **Auto fit parameters**

Used to set the parameters for the algorithm that runs when clicking on the RV Auto fit option on the **stats and controls panel**.

- Set maximum number of planets that the automatic RV fit algorithm will search for.
- Set the **false alarm positives** thresholds above which a peak is considered significant.

For now utilizing GLS for the the RV auto fit is the only option.

- Limits and Priors

  Set bounds to planetary parameters for each planet before the simulations and thus enhance model accuracy.

  There are three options to set constrains on the planet parameters:

  1. Set minimum and maximum value for each parameter.
  2. Set mean and standard deviation for each parameter (Gaussian distribution).
  3. Set mean and standard deviation for each parameter (Jeffreys prior distribution).

  There is also an option to set priors if Gaussian processes are used for RV or transit analysis.

- N-body

This tab is used to set parameters for long-term stability check of multi-planet systems when running orbital evolution.

- **General**

  Maxi time of evolution [yr]." Usually set 10 or 100 thousand years to check how the orbital parameters change over this period of time and weather planet orbits become too eccentric or unstable.

  Time step [d]: It is important to set the number of days less than period time of the inner planet.

There is also an option to set SyMBA/MVS/MVS_GR

- **Test arbitrary configuration**

This tab gives the option to test the stability of randomly chosen parameters over time. Set K,P or m,a and click **Run orb.evol**.

- **Integrate MCMC/Nested sampling**

For more information check the *Stability analysis* section.

- Plot options

Fig. 7: *Plot options*

Plot options provides the opportunity to customize the appearance of the GUI, including plots, data points, text for the different types of data.

The following options apply to RV, transit and partially to TTVs:

Size: Change the size of the data points on the corresponding plot.

Alpha: Change the opacity of the data points on the plot.

Symbol: Choose from several symbols to represent the data points.

Color: Change the color of the data points and of the text representing the data in the **Data panel**.

There is also an option to customize the appearance (color, width) of the model itself.

GLS/MLP/TLS

This tab offers several options including:

- Select the number of (highest) peaks that will be marked with an arrow.
- Check the cross hair check-boxes and show aliases with choosen color.
- Set alias periodicity in days.

Fig. 8: *Cross hair enable*

N-body

This tab holds the controls for visualization of the orbital evolution parameters on the Visualizations panel. In *Incl/Omega* tab choose to show either the evolution of inclination or that of the longitude of the ascending node. In *Energy* select to visualize either energy or momentum. In *Delta omega* show the argument of periapsis change and choose plot overview. In *Res. angles* inspect the evolution of the resonance angles. There is an option to visualize any of the resonance angles that depend on the mean motion resonance order, to select inner or outer planet's eccentricity, and to show the trajectories of the librating angles.

Cornerplot

Fig. 9: *Make Cornerplot*

After running MCMC or NS the MCMC and NS samples are saved in the tool's directory and the option to create a cornerplot is available. The cornerplot is a neat way to plot model errors and demonstrate any correlations between the orbital parameters which have been modeled, including jitter and offset.

In the **Customize cornerplot** window the parameter fields as well as the plot appearance can be modified:

1. Rename the parameters for example to include metric units or planet name.
2. Include planetary masses if all necessary parameters are available and choose units (in Earth, Solar or Jupiter mass).
3. Choose which statistic to represent with cross hair.
4. Choose number of bins for the histogram.
5. Choose error representation including fill and color palette.

The button below **Make cornerplot** allows browsing a folder in which to save the pdf file with the cornerplot.

4.5.5 Data panel

This is the GUI area where the data is imported. Depending on the type of data that you are trying to fit, you can choose between **Radial Velocities** (RV data), **Transits** (Transit data) and **TTVs** (Transit-timing variations).

- RV data

  Load RVs: Load the RV data from a local folder. Checkbox should be checked to allow for offset and jitter modeling. You can manually enter approximate values for both help the model.

  Choose an RV trend

  Data options: modulate standard deviation and determine outliers. This option offers a method to remove outliers by selecting a smaller value in **Outlier clip []**. Also if too many observations are made per night you can set **Bin Data** to a number smaller than 1 to limit those.

- Transit data

  Load Transits: load data from a local folder. Checkbox should be checked to allow for offset and noise modeling.

  Add trends:

  Limb-darkening parameters:

  Data options: Detrend options lets the algorithm remove trends and make the model flat.

- TTVs (Transit-Timing Variations)

  Use the following TTV file format:

N Transit	t_0[BJD]	sigma t_0[d]
1	2458000.5	0.022
2	2458020.5	0.023
4	2458060.5	0.021
…	…	…
8	2458140.5	0.026

  The selected epoch MUST be always slightly before the time of the first observed transit, Using the example above, the epoch should be earlier than 2458000.5, e.g., 2458000.0. Otherwise, the TTV model is likely to skip the first transit and start from the next!

  When RV+TTVs are modeled the epoch is ALWAYS chosen to be the epoch of the RV model.

  To change the RV epoch go to:

  Models param. –> Models –> RV Model

  Then, uncheck "first RV" and add whatever epoch you like, as long as it is slightly before the time of the first transit in your TTV input file. Make sure that the time baseline of "End of Model" - "Epoch" > last t0 - first t0 in your TTV input file.

- **Activity**

  Load Activity indicators from a local folder for analysis.

  Modify data: A separate window opens with several parameters to modify the stellar activity data and investigate how it contributes to the planetary model.

- **Limits and Priors**

  Set limits to the data. This imposes restrictions on the model in order to achieve more precise results. It is very important to apply constrains on the priors before running MCMC and NS because they vary in wide ranges and these algorithms require a great deal of computational resources and may take unreasonably long time to converge. The same applies to setting limits to the orbital parameter priors.

4.6 Radial Velocity data

4.6.1 Loading RV data

There are 3 ways to load radial velocity (RV) data on exostriker.

- the **first** is directly from the selection buttons in the **Data panel**.
- the **second** is using the **This computer** selection button in the **Data inspector**.
- the **third** is also through the *Data inspector* but from the **RVBank** option.

Fig. 10: *Load RV data*

4.6.2 Fitting RV data

The radial velocity data can be analyzed utilizing either Levenberg-Marquardt (L-M) or Simplex algorithm, and either Keplerian or Dynamic model. The periodicities in the RV signal are found using the Generalized Lomb-Scargle (GLS) periodogram which scans the data and fits a sinusoidal curve on it. GLS o-c provides GLS periodogram of the residuals. Fitting with LM algorithm provides information regarding where the data offset is, i.e. it finds the data mean velocity in [m/s].

There are several things to do before initializing the data modeling.

1. Refining the RV data.
   - Enter the available stellar data in the **Stellar parameters** tab including the standard deviations.
   - Make sure that the check-boxes next to the data are selected which will allow the algorithm to model the data offset. If the signal from any of the data sets is known to have a large offset, this can be entered manually to guide the algorithm.
   - Remove aliases by selecting a minimal period larger than 1 day in the **Models param** > **GLS/MLP/TLS** tab.
   - Select options for automatic fit from the **Models parameters** > **Auto fit param** tab.

Fit using the **Fit** or **RV auto fit** option. If auto fit is chosen, the algorithm will perform the next step automatically and find the orbital parameters for the selected number of planets with the certainty depending on the Signal significance threshold.

Fig. 11: *Set parameters for RV data automatic fit*

2. Fit manually.
   - Click the **Fit** button to calculate offsets.
   - If manual fit is selected than the individual set of parameters for each planet should be modeled consecutively. This is done by adopting the best period (highest peak) from the residuals *GLS o-c tab* in the **Visualizations panel** which adds the planet's orbital parameters to the analysis.
   - After adopting the first strongest peak, again click fit and the orbital parameters for the first planet will be refined.

- The sinusoidal curve implies circular orbit; however planetary orbits are hardly ever circular. Check the *orbital eccentricity (e)* and the *argument of periastron ()* parameters to allow the model to refine these parameters for ellipsoid orbits (is not defined for circular orbits), and fit again to find the **best fit**. Note that in **Extra Plots** a phase diagram of each significant peak, i.e. for each planet found, is displayed and the model will almost match the observational data if a best fit is found.
- If there is another planet in the system, its peak may already be present or becomes evident in the residuals plots after the first peak adopt. Repeat the process until there are no more peaks higher than the false alarm positives (FAP) threshold. Note that a peak is considered significant if it is above the highest of the three FAP thresholds.
- The calculated by the model orbits can be seen in the **Orb. evol.** > **Orb. view** tab. The arguments of periastron is shown with a line.
- Each of the generated plots can be exported with a right click on it that opens a dialog window with export settings.

Fig. 12: *Fit data and adopt best period*

4.6.3 L-M or Simplex algorithm

- The **L-M** method uses Chi-squared minimization technique to find the best fit parameters including their errors; however it is not suitable for anomaly detection **jitter** cannot be added to the data uncertainty and thus the overall error of the model is underestimated, because it includes observational biases but not the stochastic ones that related to the star itself.
- The **Simplex** algorithm utilizes maximization of the log-likelihood value. On the other hand, the **Simplex** method allows adding the jitter squared as a part of a penalty term to the model error. When jitter is optimized the Chi-squared reduced ($^2{}_{\text{red}}$) becomes close to 1. However the Simplex method clears the previously calculated parameter uncertainties, so fit again with L-M to evaluate them.

Neither L-M, nor Simplex methods can optimize both parameter errors and jitter/jitter error. The approach here is to utilize algorithms for calculating parameter posterior distributions, such as *Markov chain Monte Carlo* and *Nested Sampling*. For more information see *Parameter Posterior Distributions* section.

- To see how much the jitter adds to the overall uncertainty of the model, check the *Split jitter* check-box in the **Plot opt** > **RV** tab and choose color. It is now visible in all RV plots.

Fig. 13: *Fit with Simplex algorithm*

- Furthermore, calculate *MLP (Maximum Likelihood Periodogram)* on **MLP** section, to investigate which periods have significant likelihoods. Detailed statistical information is available with the **Print info** option.

Fig. 14: *Maximum Likelihood Periodogram.*

Note: Every significant peak needs investigation on whether it is a planet or stellar activity (Check **Activity indicators** on help widgets area, GUI Layout section).

4.6.4 Multiplanetary systems

In case of **multiplanetary systems** it is kind to consider the planets masses and distances from each other (close orbits). Massive planets with close distances from the host star, will surely interact with each other due to gravity. Then a further investigation using the **Dynamical model** is necessary. That will take into account the gravitational interactions between the massive bodies by integrating the equations of motion using the *Gragg-Bulirsch-Stoer* method.

Fig. 15: *Dynamical fit.*

Before enabling the **Dynamical** option make sure that the orbital parameters that are acquired so far correspond to the **best Keplerian fit**, because they will be used as an initial guess for this *Dynamical fit*. The next thing to be noticed is that the orbital parameters inclination (i) and the longitude of the ascending node () become available. The dynamical model has the advantage of being able to fit for mutually inclined orbits. For the purposes of this tutorial we assume edge-on coplanar orbits (i=90, =0) for consistency with the unperturbed Keplerian frame and in order to work with minimum dynamical masses.

4.7 Transit data

There are 3 ways to load Transit data on exostriker.

1. Through the *Data panel*.
2. Through the option *This computer* in **Data inspector** on the *Accessories panel*. If the file has .tran extension, Exo-Striker populates the data in the Transit data tab, while .vels files are automatically entered in the RV data tab.

Fig. 16: *Load transit data*

Note: Usually gaps in transit data from the Transiting Exoplanet Survey Satellite (TESS) signify the so called downlink when the telescope is sending data to Earth and cannot perform observations.

3. There is an option to download transit data in .fits (Flexible Image Transport System) format. In this case a pop up appears with two options:

   SAP FLUX (Simple Aperture Photometry) : raw data. PDCSAP FLUX : cleaner data than the SAP flux with fewer systematic trends (recommended).

Fig. 17: *Load fits data*

Fits transit data from several missions including TESS is available in the data archive web page "The Mikulski Archive for Space Telescopes". From here data for a given planetary system can be downloaded in a .zip file. After extracting, choose the file with "lc" in the end of the file name.

4.7.1 Detrend the Light curve

After the transit data is inserted, it still needs to be processed, because it has systematic errors, outliers and other artifacts which can mask the real transits. Physical processes behind these errors include but are not limited to star activity, starspots that can migrate and also vary in number, systematic errors from the observational instruments, light contaminators in the field of view.

For that process *Exo-Striker* has **detrend** option in the *Data options* tab. When clicked, the detrend button opens a new window with 2 interactive plots. The upper shows the model applied on the transit data.The lower one displays the residuals by default but with the radio buttons below the **Try!** option, it can be modified to show a periodogram of the light curve or the model. The wotan package is implemented in *Exo-Striker* and it automatically filters trends from time-series data by interpolating a model, which can be polynomial, splines, moving average, etc. on the raw transit data. The normalization algorithms include different kernels to choose from, and in addition the Gaussian Processes model has an option robust, which, when checked, will filter the transits and apply only to non-transit deviations in data.

There is an option to bin the data per a chosen time period, and also to save the error free detrended data as a .tran file. More information regarding the package is available in the **READ ME** option in the **Detrend** window as well as in wotan and its documentation links.

Inside the **Detrend** window, there is also an option to remove outliers, one by one by clicking on each, or using a scalable region of interest (ROI) box for bulk removal. The coordinates of the removed points appear in the **Stdout/Stderr** window.

Warning: In order for the Gaussian processes to be an available option it is necessary to additionally install the **sklearn** Python library!

After the transit data is cleaned, the **Detrend** window can be closed as the detrended curve is in the tool's memory. Then check the box on the left of the detrend button and click **apply**. The detrended data can also be saved from the window in a .dat file which contains information for the original data as well.

Fig. 18: *Detrending and remove outliers.*

4.7.2 Finding the transit period

When the light curve is detrended, a *period search* using the *Transit Least Squares (TLS)* algorithm can be performed. This is done from the window **TLS o-c** with the residuals from the normalized light curve, with the button **(Calcute TLS)**. The result is a periodogram with a prominent peak and its harmonics (1/2x, 2x, 4x,.. of the main signal). The empirical transit detection threshold is SDE > 8, Signal Detection Efficiency, (Aigrain et al. 2016), but the SDE levels can be set from the tab **Model param** > **GLS/MLP/TLS param**. There is also an option to set minimum and maximum period of transit after obtaining it visually from the curve (you can use cross hair check boxes in **Plot opt.** > **Transit**, to check where the transits are if they are distinct enough).

When TLS o-c is being calculated, the number of tested periods is displayed in a progress bar in **Stdout/Stderr**.

Fig. 19: *Transit period search.*

After the transit period is calculated, there is additional statistical and transit parameters information in the tool's memory which can be obtained by calling the fit object in the **Jupyter shell** window:

```
fit.tlc_o_c
```

4.7.3 Fitting planetary parameters.

A crude light curve that fits the data is built by clicking **Adopt best period**. It can be further refined by optimizing the planetary parameters in the *Parameters panel*. Those included in the analysis are the transit period (**P [d]**), and phase (**t_0[d]**), the inclination, **R_{pl}/R***, which gives the transit depth, **a_{pl}/R***, that gives the length of the transit, as well as the *Offsets and jitters* denoted as offset rel. flux and Rel. flux noise in the Transit data. Additional parameters to be included are the **Limb-darkening parameters**, which follow from the gradual decrease in brightness of the disk observed from the center of a star to its edge (limb). The exact transit signal shape can be described by setting one of four types of LD coefficients: uniform, linear, quadratic and nonlinear. Uniform and linear are seldom used as they are too simple and set too rough edges of the transit signal. Quadratic coefficients are used the most. The **u** coefficients are set per each data set, because they differ for different telescopes. Do not use **Limits and priors** > **Limb-dark params** to set LD priors as they are not well defined.

Because the Limb-darkening coefficients are the same for a given telescope, the last option **L-D grouping** allows for grouping data by telescope which it was collected with, by assigning a number to it. For example, if the first 3 data sets in the Transit data are from the one telescope, and the following 2 from another telescope, set the number 1 to data1, data2, and data3, and the number 2 to data4 and data5.

After setting the parameters to be included in the model, there is an option to set minimizing parameters in the **Models param** > **SciPy param. tab**. Note that the only available minimizing algorithm for transit data is based on SciPy, while for RV data also Fortran based algorithm can be used. Usually the default Truncated Newton (TNC) and Nelder-Mead methods are applied, and there is choice to run each of them more than once. The TNC is more or less similar to the LM method and the Nelder-Mead method works more or less like the Simplex method for RV data. MCMC can also be used as minimizer.

Fig. 20: *Fitting a transit.*

The **phase folded planetary signal** can be investigated on *Plot opt.*, when the option *plot phase-folded* is enabled.

Fig. 21: *Phase folded transit.*

- Checking the **residual signal**.

If there are any other periodicities left on the residual signal they can be calculated on the *TLS o-c section*. If there aren't any peaks left on the *TLS o-c graph*, a message that *You have reached the maximum number of TLS peaks* will be shown on *Stdout/Stderr* panel.

Fig. 22: *Residual signal.*

4.8 RVs & Transits combined

4.8.1 Obtaining the Mass of a planet

It is difficult to obtain the planetary mass directly from transit signal. This implies that it will be also difficult to distinguish for example a hot Jupiter planet from a brown dwarf. For this reason often transit data is combined with radial velocity data if it is available. After loading Transit and RV data for a planetary system, choose the *RV+Transit* option in **Visualizations panel**. Then click initialize. Thus the log likelihood of both transit and RV data will be calculated and a combined model will be constructed.

In order to optimize the planetary parameters, it is a good idea to start by first optimizing only the period (P [d]), semi-amplitude (K [m/s]), and the offset (with L-M) and jitter (with Simplex) using the **RV** option in order to help the model. After that select **RV + Transit**, check the other parameters and optimize again for both types of data to construct a joint model. This trick works because RV data optimizing uses Fortran which is faster than SciPy (Python).

Fig. 23: *Combining RV and Transit data.*

In the *Models param.* > *Models* tab check **first RB [BJD]** for epoch which will default the epoch to the first observation. This is especially useful when the order of data loaded to Exo-Striker is not chronological.

Attention: Keep in mind that when optimizing RV + transit data with Keplerian model, the parameter **t_0[d]** in the Parameters panel is the main parameter which defines the planet's phase (the time of the *first* transit) and the mean anomaly (Ma, which defines where on it orbit is the planet in a given epoch) is automatically calculated. On the other hand, if we choose dynamical model, the main parameter is Ma. Also the parameter **t[d]** (Epoch) represents the time of the *first* RV measurement.

After combining both data sets, the first estimation of the **planets mass** is obtained. In order to have a better estimation acquired, it is necessary to include the stellar parameters such as stellar mass and radius to the model. As soon as the statistical parameters are optimized, the the model is ready to go through MCMC or Nested sampling for the obtaining posterior errors. MCMC and NS can also be used as optimizers for the joint model by checking the best log likelihood option.

(For more information check *Obtaining the best fit parameters uncertainties & sections*).

4.8.2 Binning data points

Sometimes it will be necessary to **bin RV data points** when the data set has many observations during a very short period of time. This procedure can lead to a better model estimation.

Fig. 24: *Binning data.*

In the example above the data set has ~10 RV observations in a time span of 1 hour. By binning the data through eg. 0.02 days, you only accept RV measurements with minimum period of ~30 minutes between 2 observations.

4.9 Parameter Posterior Distributions

4.9.1 Markov Chain Monte Carlo method

Exo-Striker uses two approaches for approximating parameter posterior distribution and estimating uncertainties of the best fit parameters: the *Markov chain Monte Carlo* and the *Nested Sampling* methods.

The general idea behind the Monte Carlo algorithms is to approximate the posterior distribution of parameters of interest that cannot be calculated exactly, by generating random samples from this distribution. The Markov chain part implies that each sample drawn is probabilistically depending on the current state and not the past. Each sample is added to the chain within the limits and priors that have already been set, and with probability determined by how well it describes the observational data. With time the samples accumulate and start to converge around the desired quantity.

In Exo-Striker there is a choise for which statistic the samples to converge to: the posterior mean, median or mode, the best log-likelihood, or just use the best fit parameters already obtained. When the process has finished the orbital parameters are adopted as this statistic or remain as they were before that (when **use the start param.** is chosen). The last option is useful when the best fit parameters are calculated using the Simplex method but we need the errors. With the **Best lnL** option, the MCMC works as an optimizer as the orbital set of parameters is updated with the one leading to maximum likelihood.

The MCMC algorithm is dependent on the initial state and as such it is important to have a warm-up step with burning phase samples that are discarded. The real phase of the MCMC starts with the best parameters estimated during the burning phase. Each of the walkers explore the parameters space trying to converge reaching maximum log-likelihood.

Fig. 25: *Markov Chain Monte Carlo method.*

Before initializing the algorithm there are several model parameters to be set in *MCMC param.* (Models param.) on Parameters panel.

- **Burning phase samp.** : First steps in the MCMC chain to let the walkers map the parameter space.
- **MCMC phase samp.** : Represents the amount of samples of the real MCMC phase (At least 5000).
- **N threads/CPUs** : Number of CPUs from your local machine that will be used for this process.
- **Init. Gaussian Ball** : How far from the starting point the sampler will start producing samples.
- **N walkers factor** : Each walker will produce a different chain that will explore the parameter space (N walkers factor * DOF).

Usually it is a good idea to save the parameters in the tool's memory so they can be reused when the same session is opened next time.

Now, everything is ready for the MCMC process to start **(Run MCMC)**.

Sometimes the MCMC process can reach several hours, depending on the amount of samples and the dimensions of the system. You can always check the MCMC progress at the **Stdout/Stderr** on the Accessories panel.

Note: Keep in mind that setting bounds on the parameters space, on the *Limits and Priors*, in the **Parameters panel** and also limiting the data in the **Data panel** is necessary to save computational time because NS maps the whole parameter space.

Fig. 26: *Customize cornerplots.*

When the process is over, the results are printed in the **Stdout/Stderr** window and include the planetary and data parameters and their errors. These values are also populated in the parameter panel where the bigger of the asymmetric errors is pointed out.

The *Go to "Make Cornerplot"* option in the **Models param.** > **MCMC param.** tab redirects to the *Plot options*. Green indicator means that there are samples in the memory that are required to generate the cornerplot. It is now also possible to print a LaTex table with the parameters and the corresponding uncertainties. If LM method is chosen no need to set asymmetric errors when printing the Latex table as they will be the same as the errors come from the covariance matrix. Otherwise, if MCMC and NS methods have been implemented, choose asymmetric (errors come form the posterior distribution).

By default the cornerplot is saved in the root directory, but this can be modified.

The **Customize cornerplot** dynamic window offers modification of the parameter labels and choise whether to include all or some of them in the final plot, to include additional parameters such as planetary mass, and orbit's inclination, to select statistic. In case of transit data, the radius relation is available, so the planet radius can also be included.

There is also an option to change the number of bins making the histogram on the top of each column. If the **show title** option is selected, the parameter and errors will appear above the histogram.

The cornerplot can be also be reversed in order to occupy the top right part of the plot instead of the bottom left. Usually plot data points remains unchecked because the plot becomes too large. By pressing *Make cornerplot* the final results are extracted on your local exostriker folder as a pdf file.

The cornerplot represents the posterior MCMC distribution of the fitted parameters and provide a comparison between the probability density distribution of the overall MCMC samples for each fitted parameter. The two-dimensional parameter distribution panels represent all possible parameter correlations with respect to the chosen statistic (mean, median, best fit, etc.), whose position is marked with a line. The contours around the samples denote whether they lie within 1, 2 or 3 standard deviations (). Samples outside of the contours have negligible probability while those inside the inner contour (1 credible interval), are equally likely. The best fit usually is inside 1 region. The appearance of the statistic and contours can also be modified through the **Customize cornerplot** option.

Further information about emcee sampler and its modes can be found on emcee documentation.

4.9.2 Nested Sampling method

Nested sampling is another approach for parameter distribution analysis by generating samples from the posterior distribution. NS offers different sampling options that also include breaking the parameter space in different regions, generating samples from each of them and reconstructs the original distribution by applying different weights. NS uses the *dynesty* package.

Before initializing the algorithm there are several model parameters to be set in *Models param* > *Nest. Samp. param.* tab in the the Parameters panel.

- **Static** : Static sampling, the number of samples remain constant during runtime.
- **Dynamic** : Dynamic sampling, the number of samples vary during runtime in order to maximize calculation accuracy.
- **Dynamic samp. opt.** : Sampling options. Usually random walk is used.
- **N threads/CPUs** : Number of CPUs from your local machine that will be used for the process.
- **dlogz stop** : The minimum change in logz to be reached before suspending the algorithm. In multimodal distributions it is difficult to reach dlogz of 0.01 so there is another option to limit the algorithm runtime: if *use stop* is checked, maximum number of iterations and functions calls can be set.
- **Live points factor** : Number of live points used. It is necessary to set at least 1000 life points (N walkers factor * DOF).

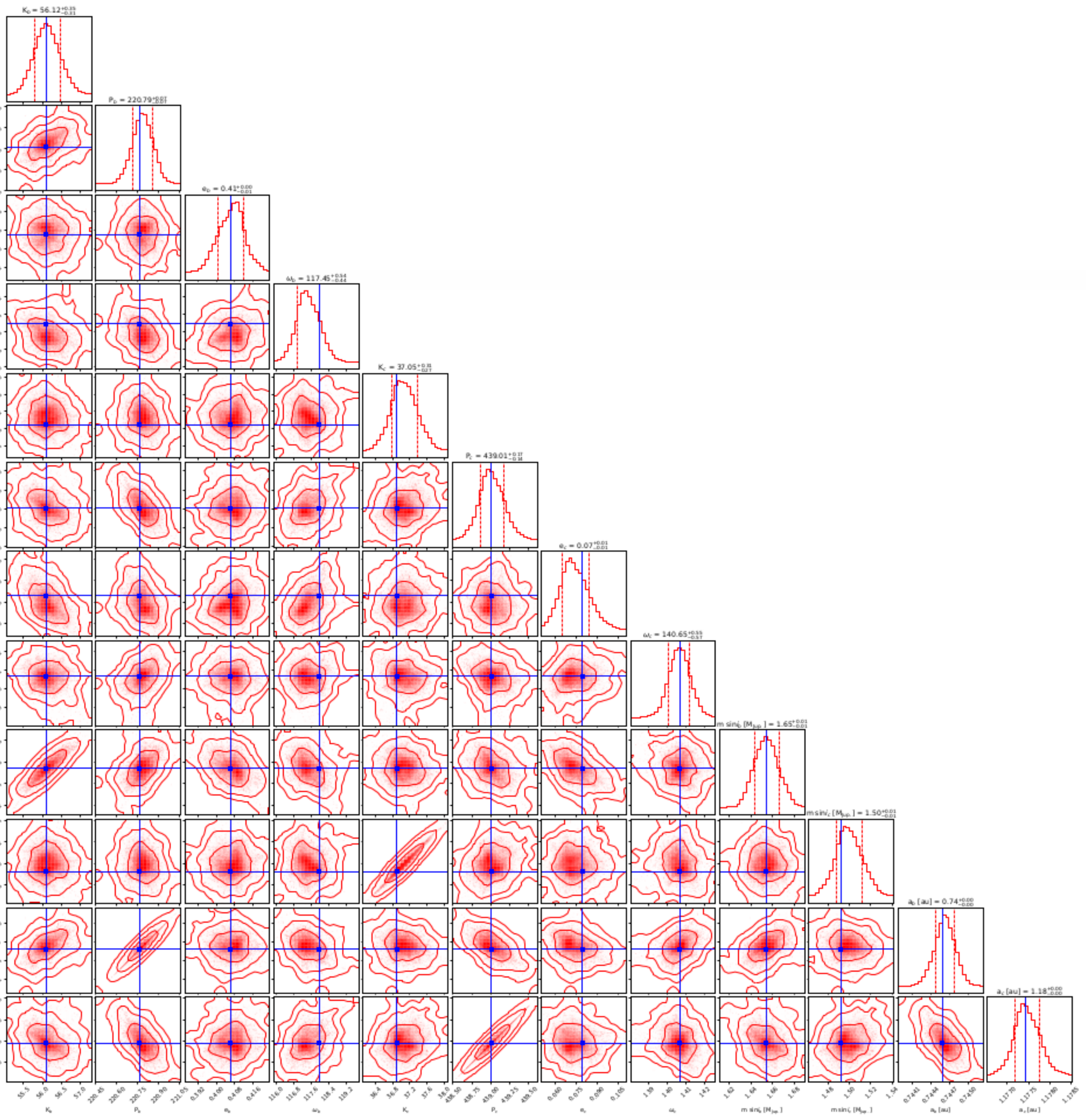


Fig. 27: *Final histograms.*

Fig. 28: *Nested Sampling method.*

- **Posterior frac.**: this is a percentage that sets weight of static vs. dynamic NS. If the fraction is set to 1 (100%) this is fully dynamic NS which converges to the posterior similarly to MCMC. Otherwise if lower percentage is chosen this means that the algorithm converges more and more to the evidence (Bayesian statistics).

Then, a selection between the options in *Adopt MCMC param. as* is required. To use NS as a minimizer and adopt NS parameters as planetary parameters, check the **best lnL option**. *Dynesty* also supports a number of options for bounding the target distribution (*Dynesty bound opt.*). Unlike MCMC, Nested Sampling starts by randomly sampling from the entire parameter space specified by the prior.

Note: Keep in mind that setting bounds on the parameters space, on the *Limits and Priors*, in the **Parameters panel** and also limiting the data in the **Data panel** is necessary to save computational time.

You can always check the progress of the sampling at *Stdout/Stderr* on the Accessories panel.

Fig. 29: *Customize cornerplots.*

When the process is over, the Cornerplot can be generated the same way as in MCMC. *Go to "Make Cornerplot"* option redirects to the *Plot options*. There a customization of the cornerplot is offered but also an option to include/exclude parameters from being printed. By pressing *Make cornerplot* the final results are extracted on your local exostriker folder as a .pdf file.

As in MCMC represents the posterior distribution of the fitted parameters with correlations marked with cross lines and the contours are constructed from the overall NS samples and indicate the confidence levels (i.e., 1 sigma, etc.).

Further information about DYNESTY and its modes can be found on dynestys documentation.

Note: The same procedure is applicable when the RVs are combined with Transits.

4.10 Stability analysis

4.10.1 Performing an orbital evolution

For the planetary parameters estimation to be complete, a stability analysis should also be performed. Not only the best fit parameters have to be those with highest probability from the posterior analysis, but they must be stable for large periods of time.

If there is only one planet in a system, the eccentricity would be a straight line; however multiple planets in a planetary system interact with each other during large periods of time, exchanging energy and momentum each time they are in their closest approach. During this interaction, the eccentricity of one of the planets increases and the other decreases, and when one is at its maximum, the other eccentricity reaches its minimum. Meanwhile the planetary axes change to a much lesser extent. What is the most important for the evaluating planetary evolution are the mean eccentricities for this large timescale. For planetary systems with more than 2 planets, other (secular) effects take place and the eccentricities are no longer cyclic functions. Orbital evolution can be analyzed using the *SyMBA N-body symplectic integrator*, that calculates how the orbital parameters develop with time.

The orbital elements that evolve with time include:

- Semi-major axes of the planets
- Orbital eccentricities
- The arguments of periastron

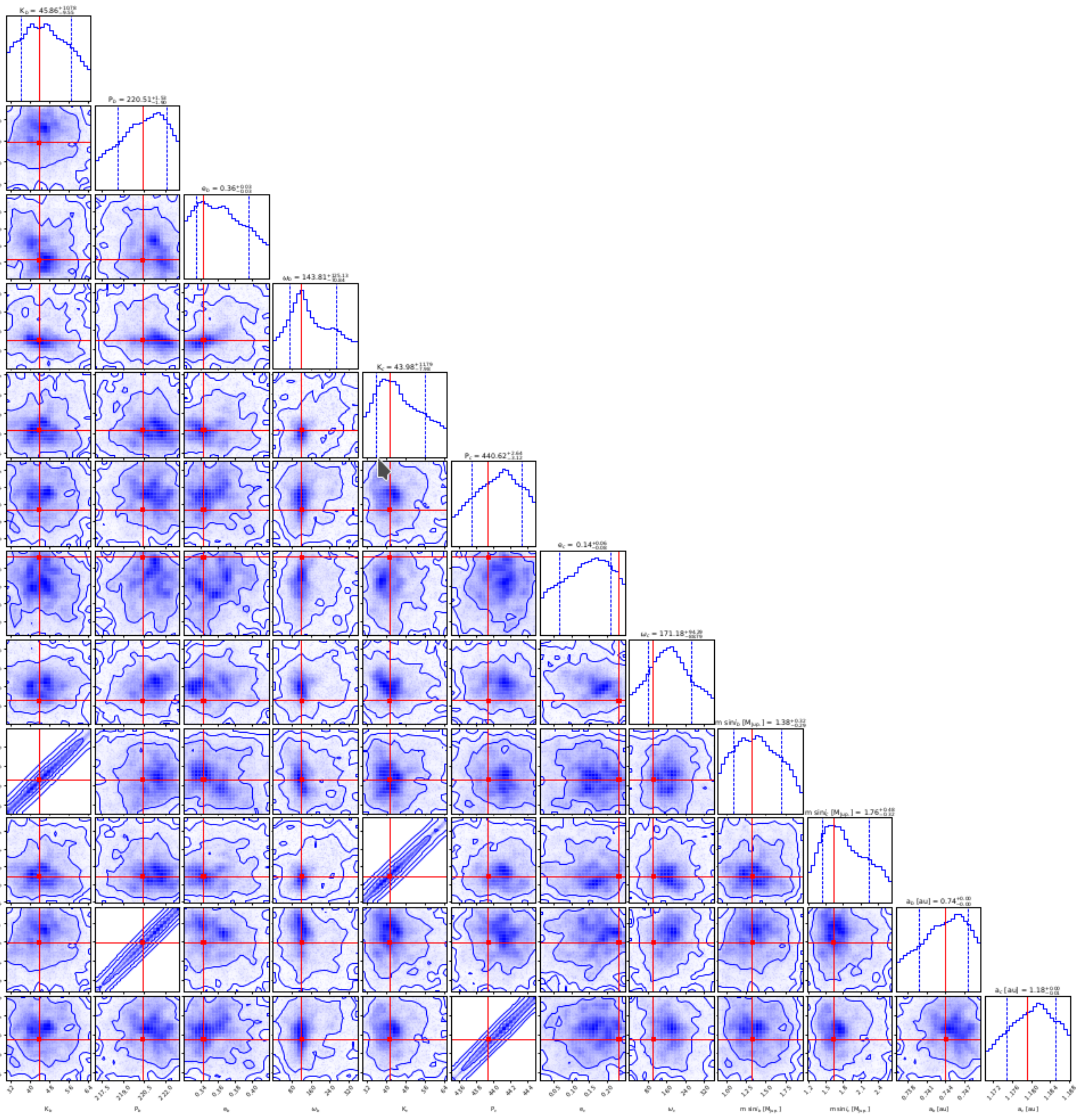


Fig. 30: *Final histograms.*

- Inclination/longitude of the ascending node
- Energy

Fig. 31: Running an *orbital evolution* with Keplerian model.

If the planetary system is stable we can investigate its dynamical properties, such as the resonant angles. To investigate the resonant angles, use the *Plot opt.* > *N-body* > *Res. Angles* tab in the Paramaters panel. On the left side from the drop-down menu choose planets in resonance and the Mean motion resonance with the alleged angles. Note that the number of angles is the resonance order plus 1. One of these is enough to librate, as well as the argument of periastron, in order the system to be in resonance. For example if the planets are in 2:1 resonance, this is a first order resonance and there are two resonant angles. Also note that for the planets to be in a mean motion resonance the ratio of their orbitals must be a little larger (about a percent) than the whole number. For example for two planets to be in 2:1 resonance the period of the outer planet must be a little more than 2 times longer than that of the inner planet.

First the stellar parameters need to be entered within the *Stellar param.* tab on the Parameters panel. Then in the *N-body* tab add the maximum time of evolution and the appropriate time step. The time step is the period at which the planetary interactions are re-calculated while the inner planet drifts on the Keplerin orbit. At least 100 points per orbit should be set so that the calculations are smooth enough. For example, if the inner planet has a period of 200 days, then a time step of 2 days is required. Running orbital evolution **(Run orb. evol)** automatically redirects to the *Orb. Evol* section, where the orbital parameters evolution is revealed.

Fig. 32: Running an *orbital evolution* with Dynamical model.

Generally, the **AMD stable inidicator**, at the Statistics and control section, automatically indicates if a system is stable, by turning into green. Besides that, evaluating the stability of a system means that the orbital parameters have to be examined long-term (e.g 1Myr). In case of planet-planet close encounters *SyMBA* automatically reduces the time step to ensure an accurate simulation with high orbital resolution. *SyMBA* also checks for planet-planet or planet-star collisions or planetary ejections and interrupts the integration if they occur.

4.10.2 Test arbitrary configuration

In the *test arbitrary configuration* section orbital evolution tests with fixed values of planetary parameters can be set to examine the resulting evolutionary parameters.

Fig. 33: *Test arbitrary configuration.*

Options between the planets RV amplitude & period **or** planets mass & semimajor axis are provided.

Integrate mcmc/Nest. Samp. (Work in progress)

This option would offer integration of many samples derived from MCMC or NS models on one plot when implemented.

4.11 Stellar activity

It is possible that radial velocity surveys for discovering exoplanets are hampered by stellar factors such as stellar magnetic activity, activity in the stellar chromosphere and photosphere that along with the star rotation can produce a signal which mimics the RV signals expected from an exoplanet around the observed star. Thus it is important to investigate and include activity data while modeling the RV signal.

In RV bank choose between the activity indices listed in the bottom window for the planetary system under analysis. When loaded to Exo-Striker, the data is populated in the **Activity** tab in the *Data panel*.

Go to the *Activity*>*Correlations* tab in the *Visualizations* panel. There are drop-down menus to choose from RV and activity data. Check plot error and plot correlations and choose **Print info** to show if there is any correlation and how strong is it.

Attention: Note that if you choose to check correlations between activity data and the residual signal, you should uncheck the box next to the corresponding planet in the **Parameters panel**, because the activity data should be juxtaposed to a model of a planetary system with one planet less.

If there is a correlation between the activity index data and the RV data from the corresponding spectrograph this means that the suspected planet signal is actually periodic stellar activity.

Fig. 34: *Finding correlations between activity and RV data.*

4.12 Fiting the RV data

If you want to use the library on the Python shell/script

In [1]: import exostriker

or e.g. to load the RV routines:

In [1]: import exostriker.lib.RV_mod as rv

In [2]: fit = rv.signal_fit(name=”hip5364”) #creates the “fit” object that contains everything.

In [3]: fit.add_dataset(“hip5364-Lick”,”./datafiles/hip5364.vels”,0.0.10.0) # add the data file, initial offset and jitter

In [4]: fit.add_planet(K=50,P=400,e=0,w=0,M0=0,i=90,cap=0) # planet 1

In [5]: fit.add_planet(K=50,P=700,e=0,w=0,M0=180,i=90,cap=0) # planet 2

In [6]: fit.fitting() #optimize the parameters

In [7]: fit.run_mcmc() # run MCMC, etc…

(However, one must be familiar with the functions… A manual on RVmod is planned, but not available at the moment.)

- All Fortran and python codes in this version need serious clean-up from junk.
- All Fortran codes are planned to be translated with f2py into python-importable libraries.
- Don’t ever run MCMC runs (or Nest. Samp.) directly in the embedded Jupyter shell! This will freeze the GUI until is the MCMC done! This is not a bug, simply the jupyter shell needs its thread and this is not done, yet. Hopefully, this will be fixed soon. Please use the GUI navigation.

4.13 Credits

If you made the use of The Exo-Striker for your paper, it would appreciated if you give credit to it. As it is unlikely that I (Trifon) will find time to write a refereed paper on the Exo-Striker soon, please cite the tool with its **ASCL ID ascl:1906.004** (see https://ascl.net/1906.004).

- Some of the dynamical RV fitting routines are done by **Xianyu Tan** (the University of Hong Kong, now in Oxford) during his Master studies with **Man Hoi Lee** (HKU).
- Some of the Keplerian RV fitting routines and other N-body codes are initially written by **Man Hoi Lee**.
- **Jakub Morawski** (Warsaw University) worked on the "RVmod" library as my intern student in the summer of 2018. His numerous contributions during the early stages of this project are highly appreciated.
- **Grigorii Smirnov-Pinchukov** helped to make the setup.py installer.
- The AMD stability check function was donated by **Stefan Dreizler** (IAG, Germany).

The Exo-Striker relies on many open-source packages, which if you had made the use of (some of) them while working with the tool, you should acknowledge too. (It is your responsibility to find the correct references in the literature):

- The interactive plotting is done with a custom version of the "pyqtgraph".
- "GLS" and "MLP" periodograms are taken from Mathias Zechmeister's repo.
- Transit least squares is taken from "TLS".
- The transit modeling is done with "batman".
- MCMC sampling is done with "emcee".
- Nested Sampling is done with "dynesty".
- TTV models are adopted from "TTVfast-python".
- The "Text editor" used in the tool is a hack between "Megasolid Idiom" and "PyEdit2".
- N-body tests are performed using a custom version of the "Swift" N-body library, modified by **Man Hoi Lee** (HKU) and **Trifon Trifonov** (MPIA).
- Additionally, the Exo-Striker uses many "standard" Python libraries like "PyQt5", "matplotlib", "numpy", "scipy", "dill", "Jupyter", "qtconsole", and more.
- The Exo-Striker project was inspired by the "Systemic project".

Scientific papers which one way or another made the use of the Exo-Striker (to our knowledge): Check in ADS.