Version history

version 1.46

- adds automatic recalculation of baseline and event noise parameters utilizing all available data points during a second solution pass; this removes the variability in calculated error bars due to user selection of a necessarily less complete set of data points for noise analysis during the first solution pass.

- adds bold blue text in the 'Excel' portion of the final report to indicate whether or not the light curve was block integrated, trimmed, or normalized. Failing to block integrate a light curve that needed it is a common error. Highlighting the presence or absence of block integration in the most looked at portion of the final report will hopefully help reduce the number of such errors.

Version 1.45

- the initial fully functional release of pyote.

Introduction to *pyote*

Bob Anderson (bob.anderson.ok@gmail.com)

pyote is an occultation timing extraction utility program written primarily in python and distributed through PyPI (the python package repository).

This program is specifically designed for those who will use such a program infrequently; it has been designed to the best of my ability to produce consistent results in the hands of both infrequent and frequent users --- the same results should be obtained no matter who processed the data.

One important feature of the program intended to give confidence to the occasional user is the production of a log file that documents all processing steps/decisions made in sufficient detail that anyones result can be reviewed by more experienced users easily --- it is sufficient to simply send such a reviewer just two things: the light curve and the log file.

1. *pyote* is designed for ease-of-use in the analysis of so-called square wave occultation light curves (defined as occultation recordings that exhibit no detectable diffraction effects). Such light curves are common with star/asteroid occultations when the star is effectively a point source and the asteroid transit speed is such that the disappearance/reappearance events occur much faster than the frame rate of the video recorder.

2. Correlated noise caused by atmospheric scintillation is frequently present in occultation observations recorded at normal video rates of 25 or 30 frames per second. *pyote* utilizes statistically rigorous calculations to properly characterize the increased uncertainty in D/R time estimates due to such correlated noise.

3. Physically realistic models are fit to the light curves with all decisions about details (complexity) of the model used made using the Akaike Information Criterion (AIC). In particular, an AIC calculation is always used to justify or reject sub-frame timing.

4. Maximum Likelihood Estimation is used throughout to determine 'best fit' of model light curves to the actual data.

The Maximum Likelihood Estimation technique requires a good estimate of the noise in an observation. For this reason, before thesearch for D and/or R values can begin, *pyote* needs the user to select one or more groups of data points that are clearly in the baseline of the light curve and preferably a group from the event region as well. From this selection of points, pyote estimates the needed noise parameters and measures the degree of correlation in that noise. I point this out because it is somewhat unusual and peculiar to MLE methodology.

While the initial selection of points for noise parameter estimation should be done as thoroughly as possible, it is not critical; pyote automatically performs a second solution pass with recalculated noise parameters based on all available data points (since the D and R values from the first pass are now available for use in identifying data points which are baseline points, which are event points, and which are transition points that should be excluded from noise calculations).

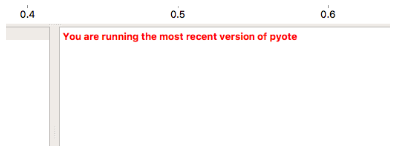
The gui for *pyote* is designed to lead the user through the necessary steps by enabling the buttons in sequence as each task is performed. So, initially, only two principal buttons are enabled: the 'info' button that brought up this document and the 'Read light curve' button. After reading this document, open a light curve, and follow the enabled buttons.

All of the major buttons have hover text associated. To learn (or refresh) how to use the program to analyze a light curve, spending a little time 'hovering' on the buttons will pay dividends.

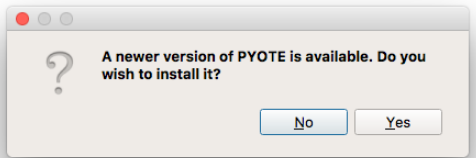
pyote will never change the input light curve, so experimentation is encouraged. There is a 'Start Over' button at the bottom that I encourage you to use freely.

Every step you make in the analysis is recorded in a log file. This is done because experience has shown that some light curves are touchy to analyze and it is useful to ask someone more experienced in running the program to look over your work. With the original light curve and a copy of the log file, your work can be exactly duplicated by someone else. And that log file is never deleted once it is opened for a particular light curve; it is simply appended to, so a record of each 'experiment' is thus always available.

Every time *pyote* is started, it connects to PyPI (assuming you have an internet connection) and checks to see if a more recent version of *pyote* has been added to the repository. If your version is completely up-to-date, you will see this



in the log file panel in the lower right-hand corner of the gui. Otherwise, this will appear:



Normally, you will want to click 'yes'. That will cause your current version of *pyote* to install (but not run) the newest version. Of course, to execute that new version, you will need to do a close and reopen.

As convenient as this is, there is always a small risk that a new version will actually 'break' something and that the 'cure' may take some time to be posted. But it is always possible to return to a specific previous version of *pyote.* The procedure to do this is explained below.

Open an Anaconda Prompt window if you are running Windows.

For a Mac installation, open a command window and type ***source activate***.

Then, type the following line in that command window:

1. pip install pyote==1.42

This command will uninstall the current (flawed) version of *pyote* and installs a specific version, in this case, version 1.42. Note the double == followed by the specific version number to be installed. (You can always determine a version of *pyote* that was working for you by opening a recent log file --- the *pyote* version that produced that log file is recorded there.)