Version history (PyOTE description is at end)

version 3.3.4:
- To shorten the time to find 'solutions', I used the Numba JIT (just-in-time) platform independent compiler that produces machine code from Python byte-code. You may notice a very slight increase in the time to start-up PyOTE because I do those compile operations while PyOTE is being loaded.

version 3.3.2:
- Adds a false-positive probability calculation and printout in the final report. This number is the fraction of 'false drops' found during the 50,000 tests that are greater than or equal to the drop value extracted from the actual observation.

version 3.3.1:
- Adds a 'false positive' detection to the final report. A new plot has been added to the error bar plot. It shows the distribution of drop sizes (B-A) for an event of the size (duration) extracted from the actual observation, but with only correlated noise in the sample (the number of points in this sample is equal to the number of points used in the lightcurve extraction). 50,000 attempts are made to find the deepest event that appears (falsely) when there is only noise being analyzed. If the drop from the actual observation is greater than the maximum size of a 'false drop', we have some assurance that the event extracted from the actual observation did not happen 'by chance'.

version 3.2.9
- Changed main plot so that the scroll wheel only zooms the x axis.
- Changed lightcurve plot so that it conforms properly to 'start-of-exposure'.

version 3.2.8
- Changed font size in help files --- it was fine for Mac but too big for Win10

version 3.2.7
- Removed the 'hover-for-help' and replaced it with a 'right-click-on-item' to get help. This scheme was introduced in PyMovie and I found it easier to use than the 'hover' scheme. In practice, the 'hover' popped up when it was not needed, so most users eventually disabled it. As a result, it became so tedious to look at help --- enable hover; hover; read; disable hover --- that the help system was used less and less. The right-click-for-help is always available and easily invoked --- hopefully this will encourage more frequent reference to it.

version 3.2.6
- This is a 'cosmetic' release --- there should be NO detectable differences from version 3.2.5 in terms of functionality.
- All python files were brought into compliance with PEP 8 coding standards. Only I care about that.
- More significantly, I removed the dependency on C code by using Numba as a code accelerator instead of Cython. As a result, I no longer need to compile

separate code versions for Mac, Windows, and Linux. That makes my life easier, but you should experience no operational changes.
- All this 'cosmetic' work is in preparation for working on PyOTE issues again.

version 3.2.5
- Added special test for Tangra files to detect the empty fields (which MUST be fixed) that Tangra outputs whenever it has trouble extracted a value from an aperture. It prints a message and stops all further processing, forcing the user to attend to and deal with the missing values.

version 3.2.4
- Modified the test for newer version to accommodate the different strings returned by pip 18.1 and pip 19.0+
- Added ability to invoke PyOTE from PyMovie with an externally supplied csv file that is automatically opened.

version 3.2.3
- fixed a long overlooked bug in the loading of the data table (at lower left corner of GUI): when there are four lightcurves, LC4 was set in the table from LC3 (i.e., LC3 == LC4 whenever there was an actual LC4. It was correct in the lightcurves themselves, so no observation analyses have been affected by this bug. It was cosmetic only.

- Added support for the PyMovie csv format

version 3.2.1
- this version makes PyOTE more robust to a common 'cockpit error' that users have been making with Tangra files. Specifically, if a Tangra csv file is opened in a spreadsheet program, then saved from that spreadsheet program, the original csv file gets modified and overwritten by the addition of empty fields at every row sufficient to match the number of columns in the longest header/comment row --- the spreadsheet program did this to satisfy its internal requirement that every row have an equal number of columns. The result is superfluous commas at the end of data lines (that Tangra did NOT put there). Until this version, that 'butchered' file could not be read. This version adds code to parse data lines only up to the first non-empty column. Hopefully this will not have ramifications in the future (like a format change that has empty fields followed by non-empty fields --- not a likely expectation).

version 3.2.0
- Changed GUI to better align text on min max edit boxes to avoid confusion.

version 3.1.9
- Fixed a bug in the test for a min/max solution search being constrained by a too large min value.

version 3.1.8
- version 3.1.7 was released without an updated version history. Here is what

was changed in 3.1.7:

- Added the ability to write the data table that is displayed in the lower left corner of the GUI out as a csv file. Now, if timestamps and block integration operations are performed on the input file, those results can be preserved in a csv file.

  A 'file save' dialog is provided should you wish to change the default name and location of the resulting file. The default name is that of the input file with the text .PYOTE inserted to the left of the .csv extension. The default location is the directory of the input file. It is recommended that you accept these defaults unless you have compelling reasons to do otherwise.

version 3.1.6:
- Values entered in the Manual Timestamp Entry dialog box are now 'sticky', thus making corrections easy to do without requiring re-entry of all data.

  Also trapped is the case where a user has entered a custom frame time but failed to click the radio button indicating that it is to be used.

version 3.1.5:
- Added additional tests of candidate solutions against a straight line so that there should always be agreement between a solution found by a min/max event size search and a marked D and R region search of the same area.

  Previously it was possible for the min/max search, which searches the entire light curve, to be tripped up by what we call a 'competitor'. A 'competitor' is an 'event' with good statistics. However, that 'competitor' may have a small magDrop and so later be rejected when we compare with a straight line solution. That 'competitor' would thus mask an event with slightly worse statistics but a larger magDrop. The change was to test every candidate against a straight line during the search. This does make the search time longer, but not too much longer.

version 3.1.4:
- Fixed error in new dropped reading detection logic when light curve was processed in field mode.

- Cleaned up some language in tooltips.

version 3.1.3:
- Expanded manual timestamp preset time deltas to include NTSC and PAL field times. Also added ability to evaluate numeric expressions entered in the 'Custom time' box: now you can type 1.001/60.0 in that box if you wish.

- Eliminated the 'entry num' column in the data matrix at the lower left of the GUI. The 'entry num' is unused and a possible source of confusion with the frame or field number for the unwary.

- Added **all** the light curves read from the input file to the data matrix display. Previously, only the first light curve values were displayed. This is done in anticipation of adding a 'write csv' button to memorialize the result of a manual timestamp entry.

version 3.1.2:
- Added a test for possible dropped frames identical to that done in R-OTE when manual timestamp is utilized. The test is to calculate the expected number of frames based on standard NTSC/PAL frame times and compare that number with the count of frames enclosed by the early and late timestamps. If there is a mismatch of more than 0.12 frames, a warning is popped up and a log entry made. It is possible to use a 'custom' frame time if your camera differs from either of those standards.

version 3.1.1:
- A convenient way to search for a 'solution' is to set a min and max event size rather than mark D and R regions. This is particularly useful in low snr situations where the D and R edges may be quite diffuse. However, if one sets the min event too large or the max event too small, the resulting 'solution' will be artificially constrained and thus be wrong. This situation is now detected and a log entry as well as a pop-up alert will tell the user to change the limits and try again.

- Three magDrop values are now calculated for each confidence level: the largest magDrop calculated using B + err(B) along with A – err(A); the nominal magDrop calculated using B and A; the minimum magDrop calculated using B – err(b) along with A + err(A)

- The labels on the Find Event button and the Calc Err Bar button were changed to more clearly suggest that after finding an 'event', one should then press the 'report' button to the right in order to complete the process.

version 3.1.0:
- Added a Mac version of a pyote startup file. It is automatically placed on the Desktop the first time pyote is run. Double-clicking on that Desktop file icon will start pyote thereafter.

version 3.0.8:
- Added a Windows batch file to the distribution that, when executed, will startup pyote. The file is called PYOTE.bat and is automatically copied to C:\Anaconda3 (if it is not already there) when pyote is first run. Now, to create a clickable desktop icon for starting up pyote, a user need only go to the C:\Anaconda3 directory, locate the PYOTE.bat file, create a shortcut to it, and drag the shortcut to the desktop. Remember, that file does not appear until the first run of pyote.

    The 'skipped' version numbers were caused by the need for repeated testing of this new feature, each test requiring a new version, even though functionality did

not change,

version 3.0.1:
- Restored the vertical splitter between the command/plot area and the table/report area. Somewhere along the line this capability was accidentally removed, and the lack of the splitter was not noticed. Now it's back.

version 3.0.0:
- No code changes. This version is the same as 2.1.6 except that it is built on python 3.7. The previous versions used python 3.6. This allows new users to install the latest Anaconda3 version (which installs python 3.7) without fiddling with archived Anaconda3 versions.

version 2.1.6:
- We now disable the Accept integration button on the first left click in the light curve. As such a click removes the color bars that result from the automatic block integration analysis, it seems intuitive to disable the Accept integration button at that time as well.

version 2.1.5:
- Disable the Accept integration button when user overrides an automatic block analysis with a manual block selection followed by a click on the Block integrate button.

version 2.1.4:
- Corrected a bug that kept manual selection of block integration from being performed after a refusal to accept the automatic block analysis results.

version 2.1.3:
- A minor change to how color bars are plotted when the automatic block integration feature is employed. The edges now appear between data points so the bands are easier to see, particularly for 2 point block sizes.

version 2.1.2:
- To ease usage of the automatic block integration feature, accepting the automatically determined block integration parameters no longer uses a modal query box, which interfered with the ability to explore/expand the light curve plot. Now there is separate button which gets enabled after an automatic block integration completes.

version 2.1.1
- Added progress bar tracking of block integration analysis because it can take an extended amount of time to complete the analysis when the light curve has many points.

version 2.1.0
- Added automatic determination of 'correct' block size and offset for block integration when user clicks *Block integrate* button without selecting the two

points normally required to specify integration block beginning and end. The user can choose to accept or reject **pyote**'s opinion of the correct parameters to use when the automation determination is invoked.

version 2.0.9
- Made the selection of Tooltip display 'sticky'
- Duration calculation when D and R span midnight now handled correctly

version 2.0.8
- toolTips changed to invoke and display in a custom dialog box that can be moved and resized to better accommodate legacy displays
- Calc flash timing calculation fixed to properly deal with the non-integer frame numbers that can result from field processed csv files
- Flash timing has been verified to work with integrated light curves
- Made block integration 'sticky' in that a 'Start over' no longer undoes a previous block integration. As a result, once block integration has been performed after a file read, it cannot be done again; a reread of the original file is now required.

version 2.0.7
This version provides several features to ease the processing of light curves that are timed with LED flashes from iPhones (John Grismore's AstroFlashTimer) or Android phones (Eric Couto's Occult Flash) rather than VTI timestamped files

- Adds a button to calculate the edge position of an LED timing flash.
- Adds a checkbox to enable/disable the tooltip messages that appear when a control is hovered over. Tooltip display defaults to 'enabled' because tooltips are an important aid for guiding users initially. Later, when such help is no longer needed, the user can turn them off (they are annoying when you don't need them).
- Adds the ability to select which light curve is to be analyzed. Previous versions would only analyze the first light curve for D and R events. This flexibility is useful in general, but was particularly needed to support LED flash timing.
- Adds a checkbox to force manual entry of timestamp info. This is useful when OCR on a VTI timed light curve has catastrophic errors. It is always employed when using LED flash timing.
- During the error bar calculation, it is possible for the Cholesky decomposition needed for treating correlated noise to fail. Previous versions treated this as a fatal error and would not produce a final report. This version instead treats the noise as uncorrelated and continues processing to produce a final report.

version 2.0.6
- Added additional instruction in the popup that appears when no timestamps are found in the csv file. This will give casual users additional guidance and clarification for the manual timestamp entry process.

version 2.0.5
- files generated by pyote now contain PYOTE in the filename.

- Timestamps can be corrupted to the point that a timeDelta of 0.0 can result. This version traps that event and reports it clearly --- 2.0.4 failed silently with a divide by zero exception

version 2.0.4
- improves the handling of errors during the reading of Tangra files by showing the offending line in the report panel. Tangra, if it has a tracking problem (i.e., loses it) will emit an empty field for that measurement, leaving it up to the user to decide how to fill in the missing value. Prior pyote versions simply reported 'format error' without providing a printout of the offending line. This version fixes that.

version 2.0.3
- detects and handles situations in which fewer than 14 baseline points are available for calculation of correlated noise coefficients. When fewer than 14 points are available, the correlation coefficients are set to: [1, 0, …] (i.e., coefficients are set to 'no correlated noise')

version 2.0.2
- Note: this version has many significant changes. If you lose confidence in this version, remember that you can always go back to version 1.47 by typing ---

    pip install pyote==1.47

in an Anaconda console. (Be sure to use double == signs in the command.)

- improved handling of D and R region selection so that one cannot enter an invalid configuration --- automatic corrections/changes are applied.

- incorporates a new 'solver' that no longer requires an initial estimation of baseline noise. This 'solver' is also much faster. With this 'solver', the two-pass modification added in version 1.46 is no longer needed.

- removes unneeded 'analyze noise' buttons and rearranged other buttons to be in-line rather than one above the other to allow the vertical splitter between the plot area and report area more room to change (a help to those using screens with relatively low pixel densities).

version 1.47
- adds bold red highlighting to message:

! There is something wrong with timestamps at D and/or R or frames have been dropped !
so that it is harder to miss.

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 occultation light curves that can be modeled reasonably well with a model based on geometrical optics. 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 diffraction effects are masked by the natural integration effect of the camera operation coupled with 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. Additional noise correlation is often present due to the relatively slow electronics present in low-cost frame grabbers and in the camera electronics responsible for generating the composite video output.

3. Physically realistic models (but based on geometrical optics) 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.
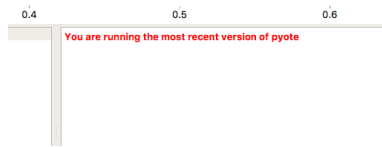
The gui for *pyote* is designed to lead the user through the necessary steps by enabling the buttons in sequence as each task is performed. So, initially, only two principal buttons are enabled: the 'info' button that brought up this document and the 'Read light curve' button. After reading this document, 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.
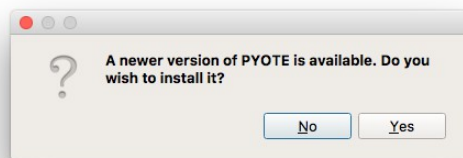
pyote will never change the input light curve, so experimentation is encouraged. There is a 'Start Over' button at the bottom that I encourage you to use freely.

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.)