"lightcurve" Data Processing program v1.0

Build 8/22/2012

Using Lightcurve 1

Lightcurve Command Line Parameters 2

Lightcurve Outputs and Files 3

Lightcurve Examples 3

Data File Formats 4

Lightcurve Processing Sequence 6

Python Plotting Scripts 6

Structured Comments for Python Plotting Scripts 7

Warning About Uncertainties 7

Algorithmic Details 8

References 9

All the examples are subject to change, but should be close to the version of lightcurve given above. Version 8/10/2102 made a tiny definition change from earlier (experimental) versions of lightcurve.

"**lightcurve**" processes data in a flexible way to allow user-driven processing for typical lightcurve analysis. Lightcurve can subtract theoretical curves (e.g. best fit), and produces residual data files which can then be further processed by lightcurve or other programs.

Lightcurve is a command-line program, written in standard C++. It currently runs on Windows, but could be fairly easily ported to Unix and Mac platforms. The author has written other programs that run on all 3 platforms. While source-code is available, the goal is to make Lightcurve flexible enough that source changes won't often be needed. Please send me your suggestions for enhancements to lightcurve.

Lightcurve also comes with Python plotting scripts based on the standard MatPlotLib library of functions modeled after Matlab's plotting capabilities. Use of these scripts is purely optional.

On Windows, you can use the Inductive Logic utility 'execute' to process any number of files with a single command. Typically, you write a batch file for processing a single file given as a parameter, and then you 'execute' that batch file for each of a selected list of files. 'execute' allows you to specify a very flexible set of selection criteria, including: name (with wildcards), extension, date/time, size, and attributes. 'execute' can also process over directory trees, with directory wildcards. Alternatively, it also allows you to create your own list of arbitrary files to be 'execute'd.

On Unix, one can often create similar capabilities with shell scripts and 'sed'.

Lightcurve accepts data files with FORTRAN-style 'D' and 'Q' exponential notation (as well as the usual 'E'), for easy processing of other programs' output files.

Using Lightcurve

A typical lightcurve command might be

```
LightCurve \temp\gen.out
```

This reads the given file, and creates several default output files: a Lomb-Scargle (LS) periodogram, the best-fit sinusoid parameters, and the residuals after subtraction of the sinusoid. Lightcurve also prints a small amount of analysis results to standard output, which you can redirect to a file. E.g.,

```
LightCurve \temp\gen.out >>logfile.txt
```

You must have a directory named “c:\temp\” on your computer.

The order of command line parameters is irrelevant. Lightcurve examines all the parameters first, and performs the processing in its standard order, regardless of the order of parameters.

Usage: entering “lightcurve” with no parameters gives a brief usage summary, including command line parameters and their defaults:

```
>lightcurve
0 cmd line parms recognized
Usage: lightcurve infile >outfile [parm=val ...]
  Multi-faceted time-series data processor
  optionally folds, optional DFT, LS periodogram
  finds dominant frequency component in data, writes residuals
           foldp=0.               time fold interval
           foldf=0.               time fold frequency = 1/foldp
             dft=none             DFT type
             fit=ls               FIT type
          lsfmax=0.               LS max frequency
            lsnf=400              LS number of frequencies
         frange=15.:200.          fit frequency range
```

Lightcurve units: lightcurve accepts data as pairs of an independent variable and a dependent variable, which are typically time and intensity. The nature of the quantities and units are completely up to the user, and lightcurve does not need to know them. However, you must enter all values in the same units. E.g., if your time unit is “days”, then your frequency units are “cycles/day.”

Lightcurve Command Line Parameters

Defaults are given in parentheses, but subject to change. For enumerators, the first choice is the default.

dft	{none, min, ext} Specifies to perform an exact DFT, and how to choose the fundamental frequency. min ≡ fundamental is $1/(t_{max} - t_{min})$. ext ≡ fundamental is $\left(\frac{n}{n-1}\right)\frac{1}{t_{\max} - t_{\min}}$, which reproduces a standard DFT if the samples are uniformly space.
fit	{ls, dft} Defines the starting frequency for the sinusoidal fit, either LS maximum or DFT maximum frequency.
fitof	(\temp\lcfits.dat) The fit output file, to which lightcurve appends one line of fit information.
frange	(15.:lsfmax) fit frequency range: lightcurve finds the max component within this range. If not folding, lightcurve uses this as the starting frequency for the sinusoidal fit.
foldf	(none = 0) frequency at which to fold the data. See also foldp. foldf overrides foldp.
foldp	(none = 0) period at which to fold the data. foldf overrides foldp.
inconvert	{none, mag} converts input magnitudes to fluxes before calculating.
isig	(0.02) uncertainty of intensity measurements, in post-linearized units (if inconvert = mag).
lsnf	(400) number of LS frequencies in periodogram.
lsfmin	(0) The minimum LS frequency in the periodgram.
lsfmax	($n/t_{interval}$, where n ≡ the number of samples) The maximum LS frequency in periodogram.

Lightcurve Outputs and Files

Lightcurve’s *t_max* output is always within +/- a half cycle of the initial time, so it is the *nearest* maximum to the initial sample time. Similarly, *t_min* is the nearest minimum to the initial sample time.

Lightcurve produces several output files, in the same format as input files, and you may feed some of the output files (e.g., the residual intensity file) back into lightcurve for another round of processing. Currently, all the output files have fixed names, and you must move the ones you want to keep before running lightcurve again. I’m thinking about changing this. The file format specifics are detailed later. Lightcurve produces the following files:

\temp\lcdft.out	if the user requested a DFT, the DFT components in various forms: frequency index, frequency, cosine amplitude, sine amplitude, magnitude, power, and phase.
\temp\lcls.out	the Lomb-Scargle periodogram, with frequencies and powers
\temp\lcres.out	the residual data samples, after subtracting the fitted sinusoid
\temp\lccomp.out	samples of the fitted sinusoid suitable for plotting (especially along with the original data) and the Fourier interpolation between the given data points (essentially, samples of the inverse DFT evaluated over many time points).

Lightcurve Examples

It is common to choose your own frequency limit for the LS periodogram, using the lsfmax parameter:

```
lightcurve obs.dat lsfmax=200
```

A batch file for performing multiple processing steps on a single file might be:

```
rem Testing for LightCurve, and examples of its use

rem Step 1: Make original LS to 200 cyc/day, best-fit, and residuals
lightcurve %1 lsfmax=200
rem Plot the original data, LS periodogram, and the best-fit comparison
start pgmf.py %1 "ti=Original data" 012r.
start barc.py \temp\lcls.out 12k
imove/ov \temp\lcresid.out %1.res1
pause

rem Step 2: Find the phase by fitting a sinusoid to folded data
lightcurve %1 foldf=164.38 lsfmax=200 fitof=phase164.dat
start pgmf.py \temp\lccomp.out 022m fi=\temp\lcfolded.dat 012r.
pause

rem Step 3: Make a 2nd LS, best-fit, and new residuals
lightcurve %1.res1 lsfmax=200
start barc.py \temp\lcls.out 12g
imove/ov \temp\lcresid.out %1.res2
pause

rem Step 4: Find the phase by fitting a sinusoid to folded residuals
lightcurve %1.res1 foldf=85.43 lsfmax=200 fitof=phase85.dat
start pgmf.py \temp\lccomp.out 022m fi=\temp\lcfolded.dat 012r.
pause

rem Step 5: Make a 3rd LS from 2nd residuals
lightcurve %1.res2 lsfmax=200
start barc.py \temp\lcls.out 12b "ti=2nd residual"
```

Data File Formats

Input and output files are mostly columns of data: line after line of several columns. Each line must have the same number of columns. However, files also include comments for self-documentation, and structured comments to facilitate plotting with the Python scripts. Example:

```
# from AM293-5.txt.res1: subtracted f = 85.4884, acos = -0.0019, asin = -0.0063
#y time
#y residual
293.666586210   0.0278323827
293.668086159   0.0218431755
293.669686104   0.00237456678
:                         :
```

Any line starting with a "#" is a comment, and lightcurve ignores it. However, my Python plotting scripts will respond to lines starting with "#t", "#y", etc., as documented elsewhere. This makes generating accurate, self-documenting figures much easier. Lightcurve includes documentation and plotting comments in its output files.

The number and content of the structured comments will change as LightCurve is updated, and columns will likely be added.
Do not count on an exact number of comments, or spaces, etc. The only reliable property is space-separated columns of numbers. We may add columns in the future, so any subsequent processing program should ignore any columns beyond those it understands.

DFT output file:

```
#t \temp\gen.out Spectrum, N = 40, fold = 0
#y frequency index, k
#y frequency
#y magnitude
#y ~power
#y phase, deg
#y cosine amplitude
#y sine amplitude
# k   freq      mag     ~power    phase    cos    sin
  0              0    3.90595479e-17  1.50043769e-4932   0.00    0.00   39.00
  1          0.025       0.022751704    0.000517640033   -16.10       0.0218592975    -0.00630960741
  2           0.05       0.024753898    0.000612755466   -31.45       0.0211182193     -0.0129141891
  :              :            :                 :           :              :                :
```

LS periodogram output file:

```
#t \temp\gen.out, N = 40, fold=0
#y freq index
#y frequency
#y Lomb-Scargle power
# intensity statistics: n=40, avg dev [min, max]: 0.01105  0.703 [-0.9995, 0.9995]
    1   0.012820513   0.00534704743
    2   0.025641026   0.00664661005
    3   0.038461538   0.00605589098
    :       :               :
```

Residual output file:

```
# from \temp\gen.out: subtracted f = 15.0142, acos = 0.8750, asin = 0.8750
#y time
#y residual
 0.000000000   0
 1.000000000   0
 2.000000000   0
 3.000000000   0
      :        :
```

Comparison output file:

```
#t from \temp\gen.out, N = 40, foldp = 0
#y time
#y Fourier interpolation
#y Best-fit component
0.000000000   0  0.875
0.097500000   0  -0.65568869
0.195000000   0  0.402751978
      :       :       :
```

Fit output file:

```
# Best fit summary file
#y Best frequency
#y Best magnitude
#y Best phase, hemicycles
#y t_max
#y Input energy (after avg subtraction)
#y Energy reduced
#y Fraction of energy reduced
#y frequency uncertainty, 1-sigma
#y t_max uncertainty, 1-sigma
#y t_min
#y reduced chi^2
 160.0073     0.01178  -0.1100    5.000117     0.06707      0.0287      0.4279     0.009874  0.0001428
4.996992    0.966  # gen0100.out
 159.9859     0.01315   0.1010    5.000124     0.07133      0.0365      0.5116     0.007542  0.0001346
4.996999    0.878  # gen0101.out
 159.9997     0.01157  -0.0780    5.000253     0.06503     0.02889      0.4443      0.01017  0.0001489
4.997128    0.910  # gen0102.out
     :         :        :       :         :           :           :           :          :
```

The fit parameters are appended to the fit output file, so that many observation files can be processed in sequence, and produce a single, summary file for the set of observations. If the file does not exist, lightcurve creates it and adds the header lines. Any columns of the resulting file can be plotting directly with the Python plotting scripts that are available with lightcurve.

The definitions of times, parameters, and basis functions are shown below. Note that the last sample time generally occurs inside a fold period, as shown.

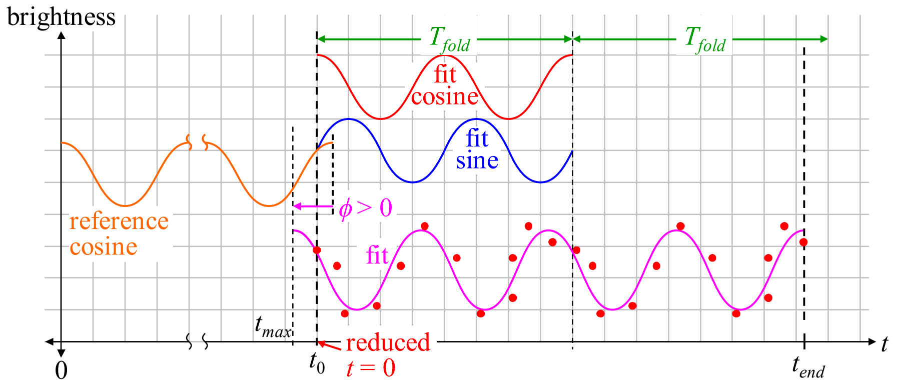

Figure 1: Definition of times, basis functions, and fit-phase for ‘lightcurve’.

The parameters are defined as follows:

best frequency if not folded: the best-fit frequency; if folded: the given fold frequency (not fit).

best magnitude best-fit magnitude of the cosine wave

best phase, hemicycles the phase ϕ in the best fit $A\cos(\omega t + \phi)$, divided by π: hemicycles = ϕ / π. ϕ is in $(-\pi, \pi]$, so best-phase is in $(-1, +1]$.

t_max the time of cosine maximum nearest to t_0 (the first sample time).

input “energy” after removing any average, the sum of squares of the input samples

“energy” reduced the reduction in the sum of squares of the samples after subtracting the best-fit sinusoid

fraction of energy reduced (energy reduced) / (input energy)

frequency uncertainty uncertainty in “best frequency”, in cycles/[time]

t_max uncertainty uncertainty in “t_max”, in [time]

Lightcurve Processing Sequence

For more extensive processing, you must invoke lightcurve several times, choosing the options you want at each step, and keeping the outputs you want. In particular, you can feed residuals from one invocation as input to another invocation.

For each invocation, lightcurve performs the following sequence of steps:

- Reads the given input file as (time, value) pairs, and then (a) if requested, converts from magnitude to flux; (b) sorts the data by time; (c) removes the DC offset.
- Creates a copy of the processed data, and then (a) reduces the time to offsets from the initial time; (b) if requested, folds it on the given period, re-sorts it by time, and writes the folded, sorted data to a file (useful for plotting or further processing).
- If requested, creates a full, exact DFT of the unfolded data. For 2,000 data points, this may take 5 minutes or so. Writes the DFT components to a file. (NB: contrary to common belief, a DFT is possible with irregular time sampling; it's just slow.)
- Creates a standard, normalized Lomb-Scargle periodogram of the unfolded data. Writes the periodogram components to a file.
- Creates a sinusoidal fit, depending on whether user specified folding:
  - If user chose folding: fits the cosine and sine amplitudes to the folded data. The frequency is fixed at the folding frequency.
  - If user did not choose folding: fits the frequency, cosine, and sine amplitudes to the unfolded data, starting with the maximum frequency from the spectrum chosen by the 'fit' command line parameter.
  - From the fitted sinusoid, estimates the time of maximum light nearest to the first data point time (t_0), and its uncertainty.
  - Appends the best-fit results to the 'fitof' file. If the file does not exist, lightcurve first creates it, and writes appropriate headers.
- Subtracts the best-fit from the *unfolded* data, and writes the residuals to a file.

Python Plotting Scripts

The plotting scripts are pgmf.py (Plot General Multiple Files), and barc.py (BAR-chart Columns). They have almost identical command line arguments. For example,

```
pgmf data.out 032b
```

plots the file 'data.out'. Use column 0 (the first column) for the horizontal, and column 3 for the vertical. Make the line 2 points wide, and color it blue. The colors are the standard MatPlotLib colors, along with any plot-symbols. For example,

```
pgmf data.out 032r.
```

means make the plot with dots as the data symbols (and not connecting them with lines). pgmf passes the color/symbol specification directly to MatPlotLib. See the MatPlotLib documentation for a full description of all the color and symbol plotting options (http://matplotlib.sourceforge.net/).

The 'barc' command line has a similar structure, but there is no "width" specifier. E.g.,

```
barc \temp\lcls.out 12k
```

creates a bar chart, with column 1 on the horizontal, and bars given by column 2, colored black.

Multiple-file plotting: TBS

Structured Comments for Python Plotting Scripts

Lightcurve includes documentation and plotting comments in its output files.

#y	labels the columns. Include one #y comment for each column of data
#t	title text for a figure
#s	filename to save the figure as a PNG
others	TBS

Example: The DFT output file contains 7 columns of data, all self-documented. In addition, the title text includes the number of input data points, and any folding period:

```
#t \temp\gen.out Spectrum, N = 40, foldp = 0
#y frequency index, k
#y frequency
#y magnitude
#y ~power
#y phase, deg
#y cosine amplitude
#y sine amplitude
# k   freq      mag     ~power     phase    cos    sin
  0             0    3.90595479e-17  1.50043769e-4932   0.00    0.00   39.00
  1         0.025       0.022751704    0.000517640033   -16.10      0.0218592975    -0.00630960741
  2          0.05       0.024753898    0.000612755466   -31.45      0.0211182193     -0.0129141891
```

Warning About Uncertainties

When lightcurve detects an uncertainty that is numerically unreasonable, it prints a diagnostic, and returns 0 for the uncertainty. *However, only the user can determine if the numerical result from lightcurve is truly reasonable*, as described below.

Misfits: One must interpret lightcurve's uncertainties with care. Misfits may be far off the user's intended value, but lightcurve cannot know that.

Lightcurve gives the uncertainty of the peak it found,
which may be different than the one you "wanted."
Thus, the actual "error" may be much bigger than the reported uncertainty.

You can reduce misfits by narrowing the "frange" parameter, but that requires tighter prior knowledge of the result you are interested in.

Also, you can use the "fraction of energy reduced" by the fitted sinusoid to give you information about how good the fit is. If the fraction reduced is small, you may have a misfit. For automated processing, you may want to discard any runs with a small fraction reduced. You must decide based on your own experience with *your* data how small is "small."

Frequency: Lightcurve's frequency uncertainty is verified by simulation: 100 runs of 400 points each, with $f = 160$ cycle/[time], $\phi = 0$, $A = 0.01$, and gaussian noise at $\sigma = .01$ magnitudes. The time samples are random, uniformly distributed over a 3-day interval. (Similar simulations with the realistic noise $\sigma = 0.02$ magnitudes further verified lightcurve's uncertainty calculation.)

Each simulated data file (gen01??.out) was processed with this command (where ?? = 00 to 99):

```
lightcurve gen01??.out lsfmax=200 frange=159.5:160.5 isig=.01
```

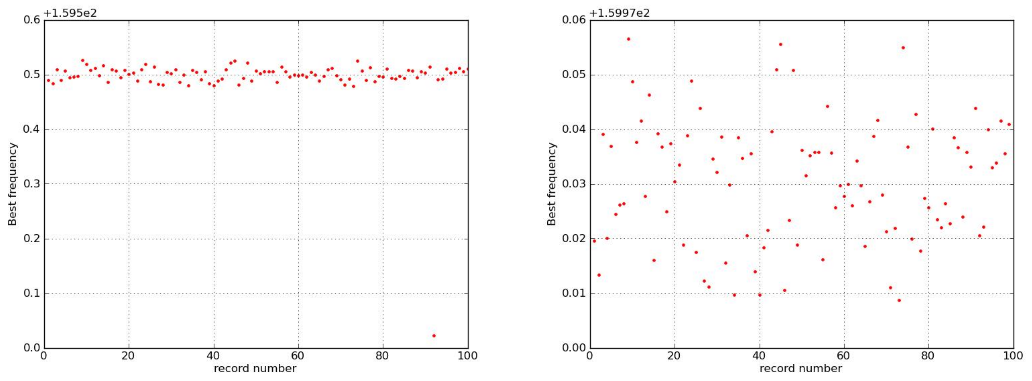

Figure 2: (Left) Raw simulation, with misfit at record 91. (Right) Misfit removed.

The raw simulation has a single misfit at record 91 (of 0 to 99). There is a spurious correlation at f = 159.52, and lightcurve's uncertainty estimate is ~0.01, even though f is almost 0.5 off of the desired value. Lightcurve cannot know that this is a misfit, and that the peak near 160 cycle/[time] was the intended peak. If your data has a misfit, your actual error may be much larger than lightcurve's estimated uncertainty.

Note that over the 100 simulated runs, including this misfit, the standard deviation of the "best frequency" is grossly inflated to 0.049, but lightcurve gives an uncertainty of only 0.011. Removing the misfit by hand gives perfect agreement between the simulated standard deviation of the frequency estimate (0.011), and lightcurve's estimated uncertainty.

The statistics of lightcurve's results over the 99 simulations (with run 91 hand-deleted) are:

```
      nrow= 99 ncol= 9
 0   160.000367676768    0.011241757328 Best frequency
 :      :
 7    0.011458555556     0.001219364842 frequency uncertainty, 1-sigma
```

We see that the actual σ of the frequency estimate (parameter 0) is 0.11, and the average uncertainty given by lightcurve (parameter 7) is also 0.11.

The statistics of lightcurve's results over the 100 simulations (including the misfit) are:

```
nrow= 100 ncol= 9
 0   159.995584000000    0.049126946914 Best frequency
 :      :
 7    0.011639070000     0.002174943307 frequency uncertainty, 1-sigma
```

We see the 4X inflation of σ in frequency estimate (parameter 0) from the *single* misfit run.

t_max: The misfit problem also corrupts the t_max uncertainties. Clearly, when the fit frequency is bad, the resulting t_max is meaningless. Hand-removing the misfit gives perfect agreement between lightcurve's t_max uncertainty estimate (0.0012 [time]), and the σ of the estimates from the 99 simulations (0.0013 [time], which is within the uncertainty of the uncertainties of $\sqrt{2/99} \approx 14\%$):

```
   nrow= 99 ncol= 9
 :      :
 3    0.000013131313     0.000125881759 t_max
 :      :
 8    0.000119037879     0.000011463943 t_max uncertainty, 1-sigma
```

Algorithmic Details

The sinusoidal fit uses full double-precision resolution for all fit parameters, but iterates until the residual sum-of-squares is stable to a fraction of 10^{-5}.

The reference time for the fit phase is $t = 0$. The fit-phase is defined as an *added* angle:

$$fit(t) = A\cos(\omega t) + B\sin(\omega t) = M\cos(\omega t + \phi)$$

$$where \quad M \equiv \sqrt{A^2 + B^2}, \quad \phi \equiv -\text{atan2}(B, A) \equiv \text{full 4-quadrant arctangent.}$$

Before lightcurve computes the LS periodogram and DFTs, it creates **reduced sample times**:

$$t_j \equiv t_{given,j} - t_0 \qquad (\text{reduced sample times}),$$

to minimize the round-off errors that would occur when large times are multiplied by large frequencies. The time origin, t_0, and the associated phase-shifts, are then added back in after the algorithm, to return results in the given time system.

In addition, if folding is specified, lightcurve then folds the reduced times:

$$t_j = t_j \bmod foldp \qquad where \quad foldp \equiv \text{folding period}.$$

Since folding reorders the samples, lightcurve then sorts the samples by reduced time.

The LS periodogram has “standard” normalization (though other normalizations exist in the literature [need reference]). Before making the periodogram, lightcurve subtracts the average value of the samples, so the resulting samples, h_j, are zero-mean. The LS equation is standard:

$$h_j \equiv s_j - \langle s \rangle \qquad where \quad s_j \equiv \text{the given samples}; \qquad t_j \equiv \textit{reduced} \text{ sample times}$$

$$Sig(\omega) = \frac{1}{2\sigma^2}\left\{ \frac{\left(\sum_{j=0}^{n-1} h_j \cos\omega(t_j - \tau)\right)^2}{\sum_{j=0}^{n-1} \cos^2 \omega(t_j - \tau)} + \frac{\left(\sum_{j=0}^{n-1} h_j \sin\omega(t_j - \tau)\right)^2}{\sum_{j=0}^{n-1} \sin^2 \omega(t_j - \tau)} \right\} \qquad [[1] \text{ eq. 3 p277}]$$

The DFT uses normalization such that the sum of the components equals the given function. Thus:

$$y_i = \sum_{k=0}^{n_f} \left\{ A_k \cos\left(\frac{2\pi k}{T} t_i\right) + B_k \sin\left(\frac{2\pi k}{T} t_i\right) \right\} \qquad where \quad y_i \equiv \text{measured values},$$

$$n_f \equiv \text{max Fourier frequency index}, \quad A_k, B_k \equiv \text{Fourier coefficients}, \qquad T \equiv \text{sampled interval},$$

$$t_i \equiv \text{sample times}, \qquad i = 0, 1, \dots n, \qquad n \equiv \#\text{ samples (measurements)}$$

$B_0 = 0$, always.

For n even: $n_f = n/2, \qquad and \qquad B_{n_f} = 0$

For n odd: $n_f = (n+1)/2$

Uncertainties: The *frequency* and *t_max* uncertainties are computed from the standard method of χ^2 increase-by-1: lightcurves shifts the parameter in question, and refits the other parameters each time, until the χ^2 of the fit increases by 1 over that of the best fit.

References

[1] Press, William H. and George B. Rybicki, *Fast Algorithm for Spectral Analysis of Unevenly Sampled Data*, Astrophysics Journal, 338:277-280, 1989 March 1.

[2] Scargle, Jeffry, *Studies in Astronomical Time Series Analysis. II. Statistical Aspects of Spectral Analysis of Unevenly Spaced Data*, Astrophysical Journal, 263:835-853, 12/15/1982.

[3] Lomb, N. R., *Least Squares Frequency Analysis of Unequally Spaced Data*, Astrophysics and Space Science 39 (1976) 447-462.