Manual for the Eclipsing Light Curve Code
Version 3

Jerome A. Orosz
Department of Astronomy
San Diego State University
5500 Campanile Drive
San Diego, CA 92182-1221

November 1, 2012

1 Citation

The proper reference for the ELC code is:

Orosz, J. A., & Hauschildt, P. H. 2000, A&A, 364, 265

2 Introduction

Yorham Avni was interested in, among other things, mass determinations of the compact objects in high mass X-ray binaries (e.g. Cyg X-1, Cen X-3, etc.). During the course of his research he wrote a FORTRAN code to compute the ellipsoidal light curve of a single (usually) Roche-lobe filling star (Avni & Bahcall 1975; Avni 1978). Avni passed on this code to Jeff McClintock and Ron Remillard shortly before his death in March of 1988 (see McClintock & Remillard 1990). The code came to me via Ron Remillard sometime in 1994 when I was a graduate student at Yale. During the spring of 1995 it became clear that the black hole binary GRO J1655-40 was an eclipsing system and that the Avni code in its original form was not adequate. I substantially modified the code in the summer of 1995 to include light from an accretion disk and to account for eclipses (Orosz & Bailyn 1997; OB97).

The code described in OB97 is somewhat unwieldy. It is very difficult to read and modify. I decided it was time to have a new code that is more general, more modular, easier to read, and most importantly, easier to modify and build upon. This present code goes a long way in meeting this goal. This present code is nearly a completely new code, written from the ground up. Although I have used much of Avni's notation, and I have followed his basic method of setting up the Roche geometry and integrating the observable flux, most of the code is my own (any errors are due to me). I have been liberal with comments in the code, so hopefully it will be much more readable to others.

The major features of this current version include:

- the ability to model any binary star system with a circular orbit, except for over-contact binaries (i.e. W UMa stars)
- the extension to eccentric orbits
- the ability to add an accretion disk around the second star
- the detailed treatment of the reflection effect (Wilson 1990)
- the use of local intensities computed from detailed model atmospheres
- more accurate integration, especially during eclipses and transits with extreme ratios of radii
- fast analytic approximations for modeling transit light curves of extra-solar planets
- more variety in the ways one can specify the geometry
- the ability to calculate light curves in time units
- better graphics (using a separate program)

3 Exhortation and Disclaimer

Apparently you have observations of some close (or not so close) binary star and you want to derive some quantities of astrophysical interest. You may use my codes to do so. However, please understand that I cannot guarantee that the codes are 100% error free or that they are appropriate for your situation. These codes seem to work well for me. I have experience in modeling binary light curves, and I can usually tell when something does not make sense. If you find a bug or find something that does not make sense, please let me know.

I don't mind if you modify the code, provided that (i) you *really* know what you are doing; and (ii) you don't give out modified copies to others. I cannot overemphasize the importance of really knowing what you are doing when tinkering with the code. Changes in the code in one place may cause errors in a different part of the code, and in some cases these errors may not be obvious.

4 Outline of the Methods

As time permits, I will include a detailed outline of the algorithms used in the code and details on their implementation. Until then, you can read any number of papers on the subject (e.g. Wilson & Devinney 1971; Wilson 1979; Wilson 1990; Avni & Bahcall 1975; Avni 1978; OB97; Orosz & Hauschildt 2000).

5 Getting and Compiling the Codes

In this current release, you should have the following:

```
-rw-r--r--  1 jorosz  staff  8930837 Oct 31 21:08 ELC.atm
-rw-r--r--  1 jorosz  staff     9965 Oct 31 21:08 ELC.for
-rw-r--r--  1 jorosz  staff     7208 Oct 31 21:09 ELC.inp
-rw-r-----  1 jorosz  staff  1034946 Oct 31 21:08 ELChubeny.atm
-rw-r-----  1 jorosz  staff   106320 Oct 31 21:08 amoebaELC.for
-rw-r-----  1 jorosz  staff     3072 Oct 31 21:08 b-w_linear.tab
-rw-r--r--  1 jorosz  staff    12910 Oct 31 21:08 checkfit.for
-rw-r-----  1 jorosz  staff   152401 Oct 31 21:09 geneticELC.for
-rw-r--r--  1 jorosz  staff    87485 Oct 31 21:09 gridELC.for
-rw-r-----  1 jorosz  staff    21204 Oct 31 21:09 intplot.for
-rw-r-----  1 jorosz  staff   989604 Oct 31 21:09 kurucz.atm
-rw-r-----  1 jorosz  staff   830572 Oct 31 21:10 lcsubs.for
-rw-r-----  1 jorosz  staff     1981 Oct 31 21:10 ldmod.for
-rw-r-----  1 jorosz  staff     9146 Oct 31 21:10 ldtab.for
-rw-r--r--  1 jorosz  staff    81904 Oct 31 21:10 limbdarksubs.for
-rw-r--r--  1 jorosz  staff    38920 Oct 31 21:10 loopELC.for
-rw-r--r--  1 jorosz  staff   337690 Oct 31 21:10 manual.ps
-rw-r-----  1 jorosz  staff    73096 Oct 31 21:10 markovELC.for
-rw-r--r--  1 jorosz  staff   220465 Oct 31 21:11 optimizesubs.for
-rw-r-----  1 jorosz  staff    46780 Oct 31 21:11 randomELC.for
-rw-r-----  1 jorosz  staff     3072 Oct 31 21:11 red_temperature.tab
-rw-r-----  1 jorosz  staff     3072 Oct 31 21:11 std_gamma-II.tab
```

The main program is `ELC.for`. It includes the routines in `lcsubs.for` using the FORTRAN `include` statement. Similarly, the file `limbdarksubs.for` is included in `lcsubs.for`. The input parameters are in the file `ELC.inp`. The model atmosphere table is in `ELC.atm` (don't mess with this file under any circumstances!). This atmosphere file has the NextGen models for the stars cooler than 10,000 K, and Kurucz models thereafter. The file `kurucz.atm` is a table made up entirely of Kurucz models. The file `ELChubeny.atm` is made up with models from Lanz & Hubeny's OSTAR2002 and BSTAR2006 grids (Lanz & Hubeny 2003, 2007).

The program `gridELC.for` is an optimizer program based on the "grid search" routine given in Bevington (1969). You specify a set of folded light curves and radial velocity curves and a set of parameters to adjust, and the code will attempt to adjust the parameters and find the minimum χ^2.

The program `loopELC.for` will use the input file for `gridELC.for` and simply loop over the indicated parameters and compute chi χ^2. No optimization is done in this case.

The program `randomELC.for` will use the input file for `gridELC.for` and simply create random parameter sets over the range indicated parameters and compute chi χ^2. No optimization is done in this case.

The program `amoebaELC.for` is an optimizer program based on the downhill simplex method (Press et al. 1992). The input file is the same format as the input to `gridELC.for`.

The program `geneticELC.for` is an optimizer program based on a genetic algorithm (Charbonneau 1995). The input file is the same format as the input to `gridELC.for`.

The program `markovELC.for` is an optimizer program based on a Monte Carlo Markov Chain algorithm. The input file is the same format as the input to `gridELC.for`.

The program `checkfit.for` is a program that will take an output model in linear units, convert it to magnitudes, and fit it to a specified folded light curve.

The programs `ldmod.for` and `ldtab.for` will plot the limb darkening behavior (i.e. the intensity vs. the angle μ) for a model, and from the table, respectively.

Finally, the program `intplot.for` is a crude plotting program. It uses the PGPLOT libraries and needs the `*.tab` files.

To compile the codes, simply issue the standard FORTRAN compiler command appropriate for your system. I suggest you use an optimize flag. On Linux systems with the Portland FORTRAN compiler:

```
pgf90 -Mextend -Mlarge_arrays -tp nehalem-64 -O2 -o ELC ELC.for
```

The `-O2` flag is for the optimization (this is the letter O). The `-Mextend` flag is for those lines which are slightly longer than standard. The `-Mlarge_arrays` flag lets the code use larger arrays. The `-tp nehalem-64` targets the processor on my MacBookPro. If you have a different type of processor, you should omit that flag.

I have had trouble with the g77 and gfortran compilers. Often the code does not compile, and when it does, the resulting executable can give incorrect results.

6 Running the Codes and the Input Parameters

6.1 The Main Code

All of the parameters the light curve codes need are in `ELC.inp`. If you try to run the code when the file `ELC.inp` does not exist, the program will write a sample file with default parameters. The parameters *must* be in the order given. There is no input format statement, so you can have leading spaces. The variable names printed by default are ignored by the code.

The code can compute the light curve from one star (star 1), the light curves from two stars (star 1 and star 2), or the light curve from one or two stars and a disk. If you have a single star, it is always referred to as "star 1". If you have a disk, it *must* be around star 2. Note that star 2 can be invisible, as in an X-ray binary.

The phase convention is as follows: for circular orbits, star 1 is closest to the observer at phase 0.0. Star 2 (if present) and the disk (if present) would be eclipsed (for sufficiently large inclinations) at this phase. For eccentric orbits, phase 0.0 corresponds to the time of periastron passage.

Here is a summary of the parameters:

`Nalph1, Nbet1, Nalph2, Nbet2`: These specify the number of grid elements used to integrate

the intensity. The "alpha" direction is the latitude on the star, running from the "north pole" (ialf=1) to the "south pole" of the star (ialf=Nalf) and represents equal steps in the angle. At each alpha, you loop over the "beta" direction which is the longitude on the star. The beta steps are equal in longitude, and there are a total of 4*Nbet points. Minimal values for these parameters are Nalph=10 and Nbet=6. In the subroutine lcsubs.for you will find these lines near the top

```
parameter(ialphmax1=1000,ibetmax1=1000)

parameter(ialphmax2=200,ibetmax2=800)
```

You can adjust these numbers as needed: smaller numbers will save a bit of memory, larger numbers will allow you to include more points in the integration.

fill1, fill2: These are the Roche-lobe filling factors for star 1 and star 2, respectively. fill=1.0 means the star exactly fills its Roche lobe. The filling factor must be larger than 0.0 and less than or equal to 1.0. The values of fill1 and/or fill2 may be overridden if one or more of the parameters primrad, ratrad, frac1, and frac2 are set below.

omega1, omega2: These parameters specify the ratio of the rotational frequency of the star to the orbital frequency of the binary. If the star is tidally locked, then omega=1.0.

dphase: The step size (in degrees!) used when "turning the binary in space". The flux at phase 0.0 is always computed, and fluxes will be computed for dphase, 2*dphase, ... 360-dphase. Note, however, that the phase units of the output files (see below) are normalized to 1.

Q: This is the mass ratio of the binary. It is the ratio of the mass of star 2 to that of star 1. For an X-ray binary like A0620-00, Q would be on the order of 15. The value of Q may be overridden if one or more of the parameters asini, primmass, and primK are set below.

finc: This is the inclination of the orbital plane, specified in degrees. The orbital plane is viewed edge-on for finc=90.

Teff1, Teff2: These are the *mean* temperatures of star 1 and star 2, respectively. Teff1 must be larger than 0. If you want to make star 2 invisible, set Teff2 to a negative value. Note that in the previous (OB97) code, the *polar* temperature of the star was specified. As discussed by Wilson (1979), the intensity-weighted mean temperature is a more representative of what is actually observed from the spectral type of the star. The value of Teff2 may be overridden if temprat>0 (see below).

Tgrav1, Tgrav2: These are the "gravity darkening" exponents for star 1 and star 2, respectively. The value of Tgrav should be 0.25 for stars with radiative envelopes (von Zeipel 1924) and roughly 0.08 for stars with convective envelopes (Lucy 1967). If the flag igrav is larger than 0 (see below), then these input values might be ignored.

betarim: This is the opening angle (in degrees) of the disk rim above the plane.

rinner: This is the inner radius of the disk, in the same units as fill2. Hence rinner should be equal to or larger than fill2. Currently the optimizer codes assume fill2=rinner for cases when star 2 is visible and there is a disk.

`router`: This is the outer radius of the disk, expressed as a fraction of star 2's effective Roche lobe radius. This parameter should be less than 1. If the inner disk radius is larger than the outer disk radius in the internal program units, the program will stop and complain.

`tdisk`: This is the temperature of the *inner* disk. Note that on the old code (OB97) the temperature of the outer rim was specified. If `tdisk` is negative, the disk will be dark. It will still cause eclipses for high inclinations, but it will not contribute to the total light.

`xi`: This is the power-law exponent on the temperature profile of the disk:

$$T(r) \propto T_d(r/r_{\rm inner})^{\xi}$$

For a "steady-state" disk, `xi` should be -0.75. For disks that are irradiated by the central source, `xi` might be larger, perhaps on the order of -0.425 (i.e. the temperature profile on the disk is flatter and the outer parts of the disk are hotter than they would be otherwise).

`Ntheta, Nradius`: The number of grid points on the disk in the azmithual and the radial direction, respectively. You need a relatively large number of points in the radial direction to get the flux integration to converge reasonably well.

`alb1, alb2`: These are the bolometric albedos used in the computation of the reflection effect. The albedo should be 1.0 for stars with a radiative envelope, and 0.5 for stars with a convective envelope.

`Nref`: The number of iterations for the reflection effect (see Wilson 1990). To skip the reflection effect routines entirely, set `NREF=-1`. If `NREF=0` a "simplified" computation is used. This approximation is exact for spherical stars and becomes worse as the stars deviate from spherical symmetry. If `NREF > 0`, then the "detailed" reflection is called `NREF` times. `NREF=1` is adequate for most purposes. In most situations in the black body mode, the computation of the reflection effect takes most of the time.

If you have an X-ray binary and want to compute the effects of X-ray heating, set `Nref=0` and set the parameter `lx/Lopt` accordingly. By default, the X-ray source is assumed to be a thin disk. If you want to make the X-ray source a point source, set `iXheat=1` (see below).

`Lx/Lopt`: This is the logarithm (base 10) of the X-ray luminosity of star 2 in ergs s^{-1} in the cases where it is invisible—i.e. `Teff2 < 0`. This number is used in the computation of X-ray heating.

`Period`: This is the orbital period of the binary in days.

`fm`: This is the mass function of star 2 in solar masses, measured from the radial velocity curve of star 1. The mass function is used to compute the orbital separation in cases where the orbital separation is not expressly given. This feature is mostly obsolete.

`separ`: This is the orbital separation in solar radii. If this number is negative, then the orbital separation is *initially* computed from the mass ratio, period, inclination, and the mass function `fm`. If `separ` is positive, its value may be overridden by one or more of the parameters `asini`, `primmass`, and `primK` below. Once the value of `separ` is set, then the component masses are computed from the period (using Kepler's third law) and the mass ratio. The orbital separation is used to compute the surface gravity of each element when model atmosphere intensities are used, so it is helpful if the scale of the binary is known (the light curve amplitudes in the black

body mode are independent of the orbital separation). Note: for eccentric orbits you must input a positive separation.

`gamma velocity`: This is the velocity zero point for the radial velocity curves. This number does not necessarily have to be specified correctly since the fitting programs adjust this number internally.

`t3`: The temperature of the "third light" star. If there is a third star associated with the binary (i.e. a distant star which forms a triple system), then set this number to some positive value. To set the third light to zero, enter a negative value for `t3`.

`SA3`: The surface area of the "third light" star, expressed as a fraction of the surface area of star 1. To set the third light to zero, enter a negative value for `SA3`.

`g3`: The surface gravity of the "third light" star, expressed in the usual convention: the common logarithm of the gravity in cgs units. This number is needed for completeness so the code can pick out a flux from the model atmosphere table. As before, to set the third light to zero, enter a negative value for `g3`.

`density`: The surface gravity of star 1 in cgs units. Owing to the magic of Roche geometry, the surface gravity of a star in a close binary is almost uniquely set by the scale of the binary (e.g. the period and separation). Conversely, if the density of star 1 is known independently, then the range of the parameters `fill1` and `Q` are restricted. Use the `density` parameter to force the density of star 1 to be the specified value. For this to work, the parameters `frac1`, `primmass`, and `primrad` below all must be set to 0. I am not sure how well fitting for this parameter works in practice.

`onephase`: If the value of the integer flag `ionephase` is 1, then the program will compute the flux for only one phase whose value is set by this parameter. This feature is useful if you want to make a picture of the binary at a specific phase.

`usepot1, usepot2`: These are the "Ω-potentials" which are the Wilson Devinney equivalent of the `fill1, fill2` variables above. If `iusepot=1` (see below), then the filling factors of the two stars are computed using these potentials. If the entered value of a potential is smaller than the critical potential, then the filling factor is set to 1.

`T0`: The epoch of the phase zeropoint when fitting light curves with time units. The value of `T0` must be in the same units as the input data, and the flag `itime` (see below) must be set to 1. `T0` is the time of inferior conjunction of star 1 for circular orbits, or `T0` is the time of periastron passage for eccentric orbits. The time of primary eclipse can be set with the Tconj parameter below.

`idraw`: The control integer for writing output files. Use `idraw=1` to write output files that the separate graphics programs use. You will have files of the following form:

`star1coo.000.00000`—sky coordinates for star 1's grid points
`star1horiz.000.00000`—sky coordinates for star 1's horizon
`star1inty.000.00000`—contains instructions for making the intensity map for star 1
`star1temp.000.00000`—contains instructions for making the temperature map for star 1

`star1rotkern.000.00000`—The rotational broadening kernel for star 1
`star1tempgrav.dat`—contains the temperatures and gravities of all of the surface elements for star 1
`star2tempgrav.dat`—same as above, except for star 2
`star2coo.000.00000`—same as above, except for star 2
`star2horiz.000.00000`—same as above, except for star 2
`star2inty.000.00000`—same as above, except for star 2
`star2temp.000.00000`—same as above, except for star 2
`star2rotkern.000.00000`—The rotational broadening kernel for star 2
`star2tophor.000.00000`—The horizon of star 2 above the disk (when present)
`diskcoo.000.00000`—the sky coordinates of the disk face
`diskedge.000.00000`—the sky coordinates of the disk edge
`diskinty.000.00000`—contains instructions for making the intensity map of the disk
`diskhoriz.000.00000`—the sky coordinates of the disk horizon
`disktophor.000.00000`—the sky coordinates of the upper rim of the disk

The last 8 numbers specify the orbital phase in degrees. Hence the file `star1inty.010.00000` contains the intensity map of star 1 at the phase of 10 degrees. Note that all of the files may not be written. For example, if one of the stars or the disk is eclipsed at a particular phase, the "coo" files or the "tophor" files may be missing.

You can use any plotting package to plot the sky coordinates. The only the visible points are output. Drawing "grids" or making grayscale or color intensity maps is a bit more involved, and I do this in the code `intplot.for`. However, note that it is necessary to have a complex "clipping" routine to account for partially eclipsed grid elements. This clipping work for the case when there are two stars, but still needs work for the case of two stars and a disk. I will try to improve this routine as time permits. Note that `intplot` looks at the `ELC.inp` file for the value of `separ`. This is used to set the scale of the plot, and can be adjusted to taste. *This number should be set to a positive number when* `intplot` *is run.*

Use the draw option if you want to compute the broadening kernels, or if you want to look at the limb darkening of a particular phase.

`iecheck`: If the disk is axisymmetric (i.e. there are no spots), the integrated flux does not depend on the phase, except for possible eclipses. Thus you only need to compute the disk flux during the eclipse phases, and only once thereafter. Set `iecheck=0` to compute the disk flux at each phase, or `iecheck=1` to skip the computation at the phases outside of eclipse. If you set `iecheck=-1` the code will not check for eclipses. Thus you can see how much difference the eclipses make. The increase in the execution speed is quite small when the eclipse checking is turned off. Note: if you have `iecheck=1`, then the intensity maps for the disk (see the `idraw` option above) will be *incorrect* for most phases (the light curve will of course be correct).

When you have a normal binary star with no disk, the flag `iecheck` can be used to cut down the execution time. Set `iecheck=5` and the code will compute the light curve at 2 degree intervals outside of eclipse. Set `iecheck=9` and the code will compute the light curves only during eclipses. A constant "reference" flux will be used for the out-of-eclipse light curves. This is useful for the computation of transits of extrasolar planets where the ratio of radii is often extreme.

`idint`: This integer is used to switch on the disk. Set `idint=0` to leave out the disk, and `idint=1` to include the disk in the light curve computation.

`iatm`: Set `iatm=0` to compute the local intensities using the Planck blackbody formula and the limb darkening law, and `iatm=1` to compute the local intensities using the model atmosphere table. If you set `iatm=1`, you must specify the scale of the system correctly since the model atmosphere flux depends on the local gravity in physical units. Thus you should give the code the correct orbital period and mass function (or separation). Caution: The code presently does very little checking to see if the temperatures required by the input values of `Teff1, Teff2, tdisk` are within the range of the table. Ditto for the gravities. The atmosphere table will be expanded in the near future, and as time permits I will add more safeguards.

You can read the first few "header" lines of the table to see which models are there and for which filters.

If you set `iatm=2`, then the code will use the ELC.atm file to set the intensity at the surface normals. For other angles, the limb darkening law is used. Thus, this is like the blackbody mode, except that the model atmospheres are used to set the luminosity scaling of the two stars.

`ism1`: Set `ism1=0` to force the code to compute the entire light curve and `ism1=1` to only compute "half" of the light curve. If the orbit is circular, only the phases 0 to 180 degrees are computed if `ism1=1`, and the rest of the phases are filled in by symmetry. Note that the time savings is far less than a factor of two on average since the "setup" time dominates (the initial geometry and reflection effect computations). If the orbit is eccentric, the program makes use of a symmetry of the *geometry* around periastron if `ism1=1`. For example, the geometry (instantaneous orbital separation, filling factors, etc.) of the binary 10 degrees before periastron is the same as the geometry 10 degrees after periastron. Since the time to compute a light curve is dominated by the "setup", setting `ism1=1` for eccentric cases almost always saves a factor of two in computing. Note if you have spots, then you must set `ism1=0` since you no longer have symmetry.

If you get a strange looking radial velocity curve for an eccentric orbit, set the `ism1` flag back to zero.

`icnU, icnB, icnV, ...`: Switches for the "filter wheel". To save overhead computing time, the code computes the light curves for 8 different wavelengths (or filters) at each phase. In the blackbody mode the wavelengths are given at the end of the input file with the limb darkening parameters (see below). In the model atmosphere mode, the intensities are filter-integrated (i.e. not monochromatic). The current table contains intensities for the Johnson $UBVRIJHK$ filters. Other filters can used without much effort—contact me in case you need a different filter. In the model atmosphere mode, set `icnU` to 1 to compute the light curve for the U band, or 0 to skip; set `icnB` to 1 to compute the B-band light curve, 0 to skip, etc.

`iRVfilt`: This is the index of the wavelength (or filter) used to compute the radial velocity curves. Note that the form of the radial velocity curves is practically independent of the input wavelength. This switch also controls which light curve is output in the `lcstar?.linear` and `lcdisk.linear` files. Thus set `iRVfilt` to 3 to compute radial velocity curves for the V band and to output individual V-band light curves, etc.

`ionephase`: Set this flag to 1 if you want to compute the flux for only a single phase. The specific

phase you want is set by the variable `onephase` above. Note that if the orbit is eccentric, then the output phase given in the files listed above may not be the same as `onephase`.

`isquare`: This flag is used to control the number of longitude points per latitude row. `isquare=0` gives you the "elliptical" arrays that are used in the Wilson & Devinney code (fewer longitude points near the poles), and `isquare=1` gives you "square" arrays (the same number of longitude points at all latitude rows). The code might choke if `isquare=0` and if the number of latitude and longitude points (`Nalph, Nbet`) points is small. In this case, set `isquare=1`. If you want to draw the binary, then set `isquare=1`.

`iusepot`: Controls the input variables for the filling factors of the two stars. Set `iusepot=0` to specify the filling factors using `fill1, fill2` (the usual mode). Set `iusepot=1` to use the Ω potentials `usepot1,usepot2`. The use of the Ω potentials makes it much easier to compare the ELC light curves with Wilson Devinney light curves.

`ifixgamma`: Set to zero to have the `gridELC` code adjust the `gamma` velocities when fitting the radial velocity curves. Set to `ifixgamma=1` to have the input value of `gamma` used in the fitting one or both velocity curves. Set `ifixgamma=2` to have a common value of `gamma` fit to *both* velocity curves.

`ilaw`: This controls the form of the limb darkening law. Set `ilaw=1` for the linear law:

$$I(\mu) = I_0(1 - x + x\mu)$$

(where μ is the cosine of the foreshortening angle of the grid element). Set `ilaw=2` for the logarithmic law:

$$I(\mu) = I_0(1 - x + x\mu - y\mu \ln \mu).$$

Set `ilaw=3` for the square root law:

$$I(\mu) = I_0(1 - x + x\mu - y(1 - \sqrt{\mu})).$$

Finally, set `ilaw=4` for the quadratic law:

$$I(\mu) = I_0(1 - x + x\mu - y(1 - \mu)^2).$$

You can find tables of the coefficients in Van Hamme (1993). Note that the same law is used in the computation of the reflection effect and in the wavelength dependent intensity computations.

If you have a binary with nearly identical stars, you can set the `ilaw` flag to the negative of one of the above values to force the limb darkening laws for both stars to be the same. Thus, if `ilaw=-2`, then both stars will have the same logarithmic law with the same coefficients.

The next 8 lines of the input file contain the wavelengths and the limb darkening coefficients for the 8 filters. (The code has a "filter wheel" with 8 positions. To save some computing time, light curves for different wavelengths are computed at the same time. To compute a light curve, you have to define the geometry. This needs to be done only once at phase 0. Once this is done, turning the binary in space takes very little time. This method of computing 8 light curves at once thus takes less time than running the code 8 different times.) Each line should contain the following type of numbers:

```
waveU xbol1(U) ybol1(U) xbol2(U) ybol2(U) x1(U) y1(U) x2(U) y2(U)
```

The coefficients `xbol1, ybol1, xbol2, ybol2` are the coefficients for the limb darkening law used in the reflection effect (Wilson 1990). Van Hamme (1993) has tabulated these coefficients for most temperatures and gravities. These bolometric limb darkening coefficients do not depend on the wavelength, and only the ones for the first line are used by the program. Note that you still have to specify coefficients for the other filters as well—you just don't have to specify them *correctly*! The coefficients `x1, y2, x2, y2` are wavelength dependent. These you should try to specify correctly.

Here is something to note: Some of the `x` coefficients can be larger than 1 for certain situations (for example cool stars in the bluer filters combined with the square root law). If you change the limb darkening law to linear and use the same coefficients, you might get numerical errors.

For the quadratic law, the sum of the coefficients is not allowed to exceed 1 when using the optimizer codes.

The default wavelengths are for the Johnson $UBVRIJHK$ system.

`ecc`: The eccentricity of the orbit. The code has problems with lobe-filling stars and eccentricities greater than about 0.75. The Wilson-Devinney code also has problems with these binaries, and apparently the difficulties are related to the physical model rather than to program coding. These binaries probably do not exist in Nature anyway, so this problem presumably causes few troubles in practice. The value of `eccen` will be overwritten if you are fitting for $e\cos\omega$ and $e\sin\omega$ (see below).

`argper`: The argument of periastron, in degrees! This parameter will be overwritten if you are fitting for $e\cos\omega$ and $e\sin\omega$ (see below).

`pshift`: The phase shift to be applied to the output light and velocity curves. The shift must be between -1.0 and 1.0. This parameter can be used to effectively switch stars 1 and 2 (e.g. put `pshift=0.5` to have the same phase convention as the Wilson-Devinney code).

`asini`: For use with pulsar binaries. This parameter is the projected semi-major axis of the pulsar's orbit *in light seconds*. If `Teff2 < 0` and `asini > 0`, then the orbital separation `separ` is computed from the values of the mass ratio `Q`, the inclination `finc`, and `asini`:

$$a = \frac{\texttt{asini}(1+Q)c}{\sin i}.$$

`rmed`: If `rmed > 1`, then the genetic fitting code will use the absolute deviation as the measure of fitness, rather than the normal χ^2.

`sw7, sw8`: These two parameters can be used to specify a restricted range of phases to compute. Set `sw7 > 0`, `sw8 > 0`, and `sw7 < sw8`. The units are degrees.

`sw9`: This is the time step (in days) used if one is computing the light curves in time units (`itime=2`, see below).

`ikeep`: Set to 1 to place the eclipse of star 2 at phase 0.0, and `ikeep=2` to place the eclipse of star 1 at phase 180.0. If `ikeep=0`, then the phase of the eclipse depends on the eccentricity and argument of periastron, as in the Wilson & Devinney code.

isynch: Controls whether the stars rotate synchronously at periastron for eccentric orbits. Set isynch=0 to have the code use the input values of omega1, omega2 and isynch=1 to have the code compute the values of omega1, omega2 for you:

$$\Omega = \sqrt{\frac{1+e}{(1-e)^3}}.$$

ispotprof: Controls the type of star spot used. ispotprof=0 is the default (constant temperature scaling). ispotprof=1 uses a linear change in the spot profile, so that the temperature scaling is greatest near the center of the spot and falls off near the spot edges. ispotprof=2 uses a Gaussian profile for the temperature scaling.

igrav: Set igrav=0 to use the input values of Tgrav1, Tgrav2, set igrav=1 to have the code choose Tgrav1 for you based on the input value of Teff1, set igrav=2 to have the code choose Tgrav2 for you based on the input value of Teff2, and set igrav=3 to have the code choose *both* Tgrav1 and Tgrav2 for you based on the input values of Teff1 and Teff2 (Claret 2000, see Figure 1).

itime: If itime=0, the light curves are computed in phase units, and the input data being fitted with the optimizer codes are given in phase units. If itime=1, then the fitting routines will assume the input light and velocity curves are in time units rather than phase units. The light curves are still computed internally in phase units. If itime=1, then the values of period and T0 are used to fold the light curves in phase. You may fit for period and/or T0. If itime=2, the light curves are fitted in time units. The value of sw9 is the time step (in days), and the values of t_start and t_end give the starting and ending times. **When using the optimizers with itime=2, make sure the time range of the model is larger than the time range of the data.** That is, the first time of the model should be smaller than the smallest time in the input data, and the last time of the model should be larger than the largest time in the input data.

MonteCarlo: This flag is mainly for the computation of accurate transit light curves for extrasolar planets. If MonteCarlo=0, then a simple interpolation scheme is used to correct for partially eclipsed pixels. When MonteCarlo>10, a Monte Carlo integration scheme is used to correct for fractionally eclipsed pixels. In practice, MonteCarlo should be set to something between 200 and 1000.

ielete: Used with the genetic and Markov fitting codes. If ielete=1, a light curve with the parameters given in ELC.inp is inserted into the initial population (genetic), or is used as the initial starting point (Markov). If ielete=0, the initial population is random, or the initial starting point is random.

The next 24 lines are parameters for spots. Each spot has 4 parameters, and each body (star 1, star2, and the disk) can have two spots each. For each spot you need to specify:

Tfac: The "temperature" factor of the spot. Make this number larger than 1 for a hot spot and a positive number less than 1 for a cool (or dark) spot. The temperature of a surface element within a spot is the underlying temperature times the temperature factor of the spot, subject to the underlying scaling function determined by the ispotprof flag above.

Lat: The latitude of the spot for the stars *or* the azimuth of the spot if the body in question is

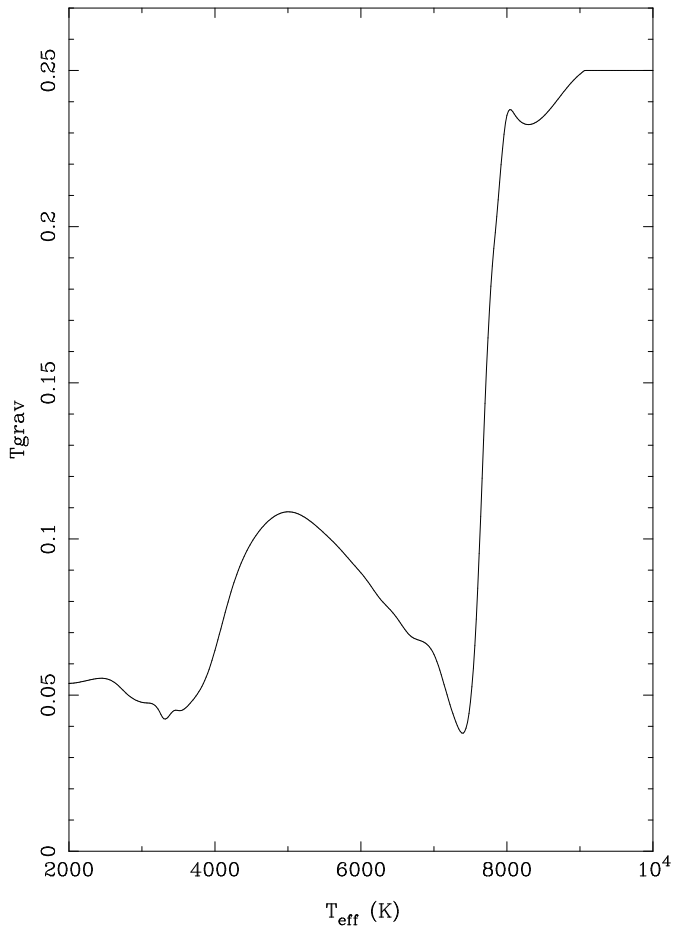

Figure 1: The values of `Tgrav1, Tgrav2` set by the code when `igrav>0`. From Claret (2000).

the disk. The units are *degrees*. For stars, the latitude is 0 degrees at the "north pole" (positive z-value), 90 degrees at the equator (orbital plane), and 180 degrees at the "south pole." For the disk, a spot at azimuth 90 degrees is seen directly ($\mu = 1$) at orbital phase 90 degrees (circular orbit), a spot at azimuth 200 degrees is seen directly at orbital phase 200 degrees, etc.

`long`: The longitude of the spot for the stars *or* the radial cutoff of the spot if the body in question is the disk. The units are *degrees*. For stars, the longitude is zero at the inner Lagrangian point and 180 degrees at the back end. A latitude of 90 degrees corresponds to the side of star 1 that is seen directly at orbital phase 90 degrees (circular orbit), or the side of star 2 that is seen directly at orbital phase 270 degrees (circular orbit). For the disk, the spot is always on the rim. How far down the disk is controlled by the cutoff radius. Set the cutoff radius to 0.9 to have the spot fill the outer 10% of the disk sector, etc.

`rad`: The radius of the spot (in degrees) for the star spots *or* the angular size of the disk spot.

If you want to turn on a spot, set the temperature factor to a positive number. If you don't want any spots at all, set all of the temperature factors to negative numbers.

The next 6 parameters let you specify the input geometry in various ways. How these work will

be described below.

primmass: The mass of star 1 in $M_\odot$.

primK: The K-velocity of star 1 in km sec^{-1}.

primrad: The radius of star 1 in $R_\odot$.

ratrad: The ratio of star 1 radius and star 2 radius.

frac1: The fractional radius of star 1: R_1/a, where a is the orbital separation.

frac2: The fractional radius of star 2: R_2/a, where a is the orbital separation.

Depending on which of the above 6 parameters are set, the values of Q, separ, fill1, and fill2 may be changed:

- if primrad=0 and frac1>0 then fill1 is computed and set. If the fractional radius is too large, then fill1 is set to 1.
- if frac2>0, fill2 is computed and set. If the fractional radius is too large, then fill2 is set to 1.
- if primrad>0 and frac1=0, the value of fill1 is computed and set, if possible. If the specified radius turns out to be large than the Roche lobe radius, then fill1 is set to 1.
- if primrad=0, frac1=0, and ratrad>0, the value of fill1 is adjusted to get the radius of star 1, based on the value of the radius of star 2 (which either comes from the value of fill2 directly, or from the value of fill2 computed from frac2). If the computed radius of star 1 turns out to be large than the Roche lobe radius, then fill1 is set to 1.
- if frac1=0, frac2 =0, and ratrad>0, the value of fill2 is set to get the radius of star 2, if possible. If the radius of star 2 turns out to be larger than the Roche lobe radius, then fill2 is set to 1.
- if primmass>0, and primK=0, the value of separ is set.
- if primK>0 and primmass=0, then the value of separ is set.
- if asini>0, Teff2 < 0, and primK>0, the values of Q and separ are set.
- if primmass>0 and primK>0, the values of Q and separ are set.

The values of fill1, fill2, Q, and separ will be reported in the file ELC.out (see below).

ecosw: The phase difference between secondary and primary eclipses for an eccentric orbit. The units are normalized phase units such that 2πecosw is the phase difference in radians. If ecosw>0 and ecc>0, the value of argper is computed and set. If you have been fitting for this parameter to get initial guesses for ecc and argper, don't forget to set ecosw=0 if you wish to fit for ecc and argper.

`temprat`: The ratios of the effective temperatures of star 2 and star 1: `temprat`=`Teff2`/`Teff1`. If this parameter is greater than zero, then the value of `teff2` is overwritten.

`idark1`: If `idark1=1`, star 1 will not contribute to the light.

`idark2`: If `idark2=1`, star 2 will not contribute to the light. This might be the case for a transiting extrasolar planet, where star 2 is usually the planet.

`Npoly`: For cases when both stars are nearly perfect spheres (e.g. transiting extrasolar planets), set `Npoly=0` for the normal numerical integration (which may be relatively slow), and `Npoly > 0` for to use the analytic expressions given in Gimenez (2006a, 2006b). When the analytic expressions are used, `Npoly` represents the number of terms in the series expansions. Values of `Npoly` between 10 and 200 are recommended. **Note that the limb darkening law must be linear (`ilaw=1`) or quadratic (`ilaw=4`) when the analytic expressions are used.** The square root law (`ilaw=3`) will be added soon.

This analytic mode is mostly obsolete. Use `Nterms` as discussed below.

`ifasttrans`: This flag was part of an effort to speed up the computation of transit light curves of extrasolar planets. If the star has symmetry (e.g. no spots or the like), the flux from the parts of the star that are not eclipsed will be the same always. Setting `ifasttrans=1` will have the code use fewer pixels in the parts of the star that are not eclipsed. However, it turns out that this flag does not really help, so using `ifasttrans=0` is recommended.

`ialign`: Set `ialign=0` to have the rotational axis of star 1 aligned with the angular momentum vector of the orbit. Set `ialign=1` to have the rotational axis misaligned. Adjust the parameters `axis_I` and `axis_beta` below accordingly. This parameter will mainly make subtle changes to the radial velocity curve of star 1 during transit (e.g. the Rossiter-McLaughlin effect).

`ifastgen`: Set `ifastgen=1` to enter the "fast genetic mode." This can be useful if it takes a fair amount of time to compute a single model. In the early phases of the genetic code, most models do not come close to fitting the data. When `ifastgen=1`, a model with a coarse grid in `Nalph` and `Nbet` is computed, and the χ^2 is computed. If the χ^2 value is less than $3\chi^2_{\rm min}(N_{\rm data} - N_{\rm terms})$ then the curve is recomputed using the full grid. This flag helps in the beginning, but becomes less and less useful as the genetic code converges.

`isw23`: Currently inactive.

`ifracswitch`: Set `ifracswitch=1` to enable geneticELC and markovELC to write the "luminosity ratios" and "disk fractions" to the ELCratio.??????? files. See below for more information on the optimizers.

Next in the input file comes the "power law coefficients", which are the limb darkening coefficients for the analytic mode. **These are not yet fully implemented, so put 8 lines of 9 columns of zeros.**

`axis_I`: This is the inclination of the rotational axis of star 1 if `ialign=1`. See Hosokawa (1953) for helpful diagrams and equations.

`axis_beta`: This is the angle of the rotational axis of star 1 with respect to the orbit if `ialign=1`.

See Hosokawa (1953) for helpful diagrams and equations.

`t_start`: When `itime=2`, the light curves and velocity curves are internally computed in time units. `sw23` is the starting time.

`t_end`: When `itime=2`, the light curves and velocity curves are internally computed in time units. `sw24` is the ending time.

`asini_error`: In the "millisecond pulsar mode", the parameter `asini` is the projected semimajor axis of the pulsar in light seconds. `asini_error` is the uncertainty in `asini`, and if nonzero will be used in the genetic and Markov codes. Values of `asini` will be drawn from a Gaussian distribution with a standard deviation of `asini_error`.

`disk_frac_ref`: If this parameter is positive, it gives the reference phase for the disk fraction when using the optimizer codes (see below). Otherwise, the median of the disk fraction over the whole orbit is used.

`radfill1`: If this parameter is greater than zero, it sets the Roche lobe filling factor of star 1 in terms of $R_{\rm eff}$, rather than in terms of the distance to the L_1 point.

`radfill2`: If this parameter is greater than zero, it sets the Roche lobe filling factor of star 2 in terms of $R_{\rm eff}$, rather than in terms of the distance to the L_1 point.

`lcbin`: This parameter is the exposure time of the typical photometric observation you have, in minutes. For example, if you are working with *Kepler* data, use `lcbin=29.4744`. For this flag to work properly, you need to set the `dphase` parameter such that there are 20 to 30 steps per time bin (e.g. one to three minutes for Kepler long cadence data). For example, if you have a binary with a period of 30 days, a step size of one minute in time is a step size in terms of the normalized orbital phase of 0.00002315, which works out to be $360 \times 0.00002315 = 0.008333$ degrees. In this case, you can round up a bit, and set `dphase=0.01`.

`rvbin`: Similar to above, but for the typical exposure time (in minutes) of the spectroscopic observations that were used to find the radial velocities. In practice, setting this parameter hardly makes any difference in the cases I have tried so far.

`contam`: This is the contamination to be applied to models of *Kepler* data. In many of the *Kepler* eclipsing binaries and transiting planets, there is light from other stars in the photometric aperture. The extra light causes the relative depths of the eclipses (or the transits if you have a planet) to go down. If this parameter is greater than zero, an offset is applied to the model *Kepler* curve (in the U filter position if you have the special ELC.atm file):

$$y_{\rm off} = \frac{k y_{\rm med}}{1-k}$$

where k is the value of *contam* parameter and $y_{\rm med}$ is median value of the model light curve. The value of `contam` should not exceed 1.

`Tconj`: The time of a primary eclipse. To use this parameter, set the `Tconj` flag below to 1.

`beam1`: The Doppler boosting factor for star 1 (see van Kerkwijk et al. 2010). If this parameter is zero, no correction to the light curve is applied. A typical value is 2.5 for an A-star.

beam2: The Doppler boosting factor for star 2 (see van Kerkwijk et al. 2010). If this parameter is zero, no correction to the light curve is applied. A typical value is 2.5 for an A-star.

isw25: If isw25=1, the X-ray source is assumed to be a point source when computing X-ray heating. If isw25=0, then a foreshortening correction for the X-ray heating is applied, assuming the X-ray emitting part of the accretion is a thin disk in the orbital plane.

isw26: Currently inactive.

Nterms: This flag activates the fast analytic mode. See the description of Npoly above. Useful values are between 100 and 500. For various technical reasons, light curves are computed much faster when Nterms=300 compared to when Npoly=300.

iuseTconj: Set this flag to 1 to use a time of primary eclipse as the reference time rather than a time of periastron passage.

iecosflag: Set this flag to 1 to specify the eccentricity and argument of periastron by $e\cos\omega$ and $e\sin\omega$.

isw30, isw31, isw32, isw33, isw34: These are currently inactive integer input flags. Use 0, one per line.

e*cos(omega): If iecosflag=1, compute the eccentricity using $e\cos\omega$ and $e\sin\omega$. This flag gives $e\cos\omega$.

e*sin(omega): If iecosflag=1, compute the eccentricity using $e\cos\omega$ and $e\sin\omega$. This flag gives $e\sin\omega$.

sw42, sw43, sw44, sw45, sw46, sw47, sw48, sw49: These are currently inactive real*8 input parameters. Use 0.0, one per line.

That is it for the input parameters. You should have a total of 155 lines.

After you run the code, you will have the following output files:

ELC.out—An output file that lists the input parameters, prints some interesting system parameters, and shows and other program messages
ELC.phases—The phases used for the light curves
ELC.parm—An output file that lists computed binary parameters
modelU.linear—Light curves in linear units (cgs soon)
modelB.linear
modelV.linear
modelR.linear
modelI.linear
modelJ.linear
modelH.linear
modelK.linear
modelU.mag—Light curves in magnitude units
modelB.mag
modelV.mag

`modelR.mag`
`modelI.mag`
`modelJ.mag`
`modelH.mag`
`modelK.mag`
`lcstar1.linear`—The light curve of star 1, linear units. The wavelength or filter is set by the `iRVfilt` switch.
`lcstar2.linear`—The light curve of star 2 (if present), linear units. The wavelength or filter is set by the `iRVfilt` switch.
`lcstar3.linear`—The "third light" light curve (if present), linear units. The wavelength or filter is set by the `iRVfilt` switch.
`lcdisk.linear`—The light curve of the disk (if present), linear units. The wavelength or filter is set by the `iRVfilt` switch.
`star1.RV`—The radial velocity curve of star 1, in km/sec
`star2.RV`—The radial velocity curve of star 2, in km/sec
`star1.delRV`—The corrections to the radial velocity curve of star 1, in km/sec
`star2.delRV`—The radial velocity curve of star 2, in km/sec

6.2 Notes

The fast analytic mode is for use when the two stars (or star and planet) are nearly perfect spheres. The reflection effect won't work, and ellipsoidal modulations cannot be mimiced. The computed rotational velocities will not be correct. The Doppler beaming corrections will be applied if `beam1 > 0` and/or `beam2>0`. The spheres can be limb darkened, and you must use either the linear law `ilaw=1` or the quadratic law `ilaw=4`. I usually set `iatm=2` and `ilaw=4`. If you compute a light curve in this mode, and want to switch over to the blackbody mode, set Nterms=0. The resulting light curve should be very similar to the light curve found when `Nterms > 0`, to the extent that the stars are actually spherical and where the reflection effect is unimportant.

If you are fitting an eccentric binary, try the following combination of parameters:

`ism1=0`, `pshift=0`, `ikeep=1`, `itime=1`, `primrad=0`, `ratrad=0`, `frac1>0`, `frac2>0`, iuseTconj=1, `Tconj` set to a time of primary eclipse, `iecosflag=1`, and `e*cos(omega)` and `e*sin(omega)` set accordingly. Also, I use `temprat>0`, and parameters to set the inclination and scale.

6.3 The Optimizers

I assume you have at least one light curve to fit. There are two basic options: (i) Fit input data in phase units. Set the parameter `itime=0`. You must first "fold" the light curve on the correct phase: if it is an X-ray binary fold the light curve so that the visible star (star 1) is in front at phase 0.0 (obviously you must have the velocity curve or a published value of T_0 and the orbital period so that this can be done). (ii) Fit for the period and/or the phase zeropoint. Set the parameter `itime=1`. This assumes each orbital cycle of the observed light curve is the same so that the data can be folded on the ephemeris. (iii) Fit in time units where something is changing

from orbit to orbit. Set the parameter `itime=2`. You can mimic moving spots by `omega1` and/or `omega2` to values other than 1. The locations of the spots stay fixed, but the positions of the spots with respect to the observers line-of-sight moves in orbital phase. Currently, there is no capability to have other parameters change with time.

The file containing the light curve should have *three* columns: phase (or time), magnitude, and magnitude errors. If you don't know the errors on the photometry, invent some!

All of the optimizer codes use the same basic input file (in addition to the `ELC.inp` file). This file is always called `gridloop.opt`. Run the code `gridELC.for` and it will make you a file with the correct form. The programs `amoebaELC.for`, `geneticELC.for`, `markovELC.for` and `gridELC.for` will actually optimize a fit, whereas the programs `loopELC.for` and `randomELC.for` will just print out χ^2 values.

The first 8 lines of this file contain the names of the files with the $UBVRIHJK$ light curves. If you don't have the light curves for certain filters, list the name as "none". The next two names are the file names with the folded radial velocity curves of star 1 and star 2, respectively. I encourage you to use the radial velocity curves (if available), especially when using the model atmospheres. If there are no velocity curves, put "none" as the file names. For example:

```
none
Bfold.bin
Vfold.bin
none
Ifold.bin
none
none
Kfold.bin
RV1.fold
none
```

Note that you cannot name any of your light curve or velocity curve files "none"!

The next line is an integer called `Nvar`. This is the number of parameters (1 to 40) that you can adjust. The next `Nvar` lines should contain strings. These strings give the name of the variables you want to adjust. I list below the strings you should use in `typewriter` font:

`inclination`
`mass ratio`
`f1` (this is fill1)
`f2` (this is fill2)
`o1` (this is omega1)
`o2` (this is omega2)
`rinner` [this is the inner disk radius (same units as fill2)]
`router` [this is the outer disk radius (in units star 2's Roche lobe radius)]
`Tdisk` (this is the inner disk temperature)
`betarim`
`T1` (this is Teff1)

T2 (this is Teff2)
T3
SA3
g3
xi
Lx (this is the log (base 10) of the X-ray luminosity)
separation
eccentricity
argper (this is the argument of periastron)
pshift (this is the phase shift)
period
T0
l1 (albedo of star 1)
l2 (albedo of star 2)
gamma (this is adjusted automatically, but it is listed for completeness)
b1 (temperature factor spot 1 on star 1)
b2 (longitude of spot 1 on star 1)
b3 (latitude of spot 1 on star 1)
b4 (radius of spot 1 on star 1)
b5 (temperature factor spot 2 on star 1)
b6 (longitude of spot 2 on star 1)
b7 (latitude of spot 2 on star 1)
b8 (radius of spot 2 on star 1)
c1 (temperature factor spot 1 on star 2)
c2 (longitude of spot 1 on star 2)
c3 (latitude of spot 1 on star 2)
c4 (radius of spot 1 on star 2)
c5 (temperature factor spot 2 on star 2)
c6 (longitude of spot 2 on star 2)
c7 (latitude of spot 2 on star 2)
c8 (radius of spot 2 on star 2)
d1 (temperature factor spot 1 on disk)
d2 (azimuth of spot 1 on disk)
d3 (radial cutoff of spot 1 on disk)
d4 (angular radius of spot 1 on disk)
d5 (temperature factor spot 2 on disk)
d6 (azimuth of spot 2 on disk)
d7 (radial cutoff of spot 2 on disk)
d8 (angular radius of spot 2 on disk)
oc (e*cos(omega))
os (e*sin(omega))
contam (contam)
e1 (beam1)
e2 (beam2)
tconj (Tconj)
density (density)
ai (axis_I)
ab (axis_beta)

`dpphi` (`ecosw`)
`temprat` (`temprat`)
`pm` (`primmass`)
`pr` (`primrad`)
`pk` (`primK`)
`ratrad` (`ratrad`)
`q1` (`frac1`)
`q2` (`frac2`)

The values of any parameter not specified in the `gridloop.opt` file are taken from the `ELC.inp` file. If you want to hold a parameter fixed, remove it from the list and change `Nvar`, *or* you can put the string "none" in front of its name. The value of this parameter is then read from the `ELC.inp` file.

After you list the `Nvar` strings, the last `Nvar` lines should contain three entries each. These are the starting value of the parameter, the step size, and the number of iteration loops. The first line of three entries is used for the first string, the second line for the second string, etc. For example:

```
4
mass ratio
inclination
t1
none separ
2.0 0.5 4
55.0 0.25 1
6500.  200.  1
128.  10.  1
```

In this example, I want to adjust the mass ratio, the inclination, the mean temperature of star 1, and the separation. I have temporarily held the separation fixed. The initial parameter values will be `Q=2.0`, `finc=55`, and `teff1=6500`, and the step sizes will be 0.50, 0.25, and 200.0, respectively. There will be 4 iterations.

Finally, there is an *optional* additional block where you can specify "observed" parameters. First give the value `Nobsvar`, which should be between 0 and 9, or not there at all. The next `Nobsvar` lines should be of the names of the observed variables, which can be from the following:

`m1` (this is the mass of star 1 in solar masses)
`r1` (this is the radius of star 1 in solar radii)
`g1` (this is $\log g$ of star 1 in c.g.s. units)
`v1` (this is the measured rotational velocity $V_{\rm rot} \sin i$ of star 1 in km s^{-1})
`k1` (this is the K-velocity of star 1 in km s^{-1})
`m2` (this is the mass of star 2 in solar masses)
`r2` (this is the radius of star 2 in solar radii)
`g2` (this is $\log g$ of star 2 in c.g.s. units)
`v2` (this is the measured rotational velocity $V_{\rm rot} \sin i$ of star 2 in km s^{-1})
`k2` (this is the K-velocity of star 2 in km s^{-1})

`xe` (this is the duration of the X-ray eclipse *in degrees*)
`in` (this is the inclination in degrees)
`ma` (this is the mass ratio)
`ec` (this is the eccentricity)
`ar` (this is the argument of periastron)
`t1` (this is the effective temperature of star 1 in K)
`t2` (this is the effective temperature of star 2 in K)
`su` (this is the sum of the fractional radii)
`l2` (this is the luminosity ratio of star 2 to star 1)
`d1` (this is the disk fraction in band U)
`d2` (this is the disk fraction in band B)
`d3` (this is the disk fraction in band V)
`d4` (this is the disk fraction in band R)
`d5` (this is the disk fraction in band I)
`d6` (this is the disk fraction in band J)
`d7` (this is the disk fraction in band H)
`d8` (this is the disk fraction in band K)

After you list the names of the observed variables, the next `Nobsvar` lines should have two entries each, namely the observed value and its error. In the case of the X-ray eclipse duration, you have the option of specifying an upper limit. If you have only an upper limit, set the value of `xe` to a negative value. For example if `xe = -10`, this means the eclipse duration is less than 10 degrees. If the model value of the eclipse duration is less than the upper limit, the χ^2 value is zero.

If you run `gridELC.for`, the "grid search" will proceed in the order given. In this example, the mass ratio will be adjusted first until a minimum in χ^2 is found. Next the inclination is adjusted until a minimum in χ^2 is found, etc. If you give step sizes that are too small, the code could run a long time. If the sizes are too big, you will have trouble interpolating to the correct value at the minimum.

If you are running `amoebaELC`, you should make the step sizes in the variables larger than for `gridELC`, usually a factor of 10 or so. The number of *restarts* is specified by the value of the first step size. Usually two restarts is enough.

All three codes will write information to the screen. The information printed include the parameters being adjusted, the χ^2 values for the input light curves, and a summary of the other parameters of interest. Once they have converged to a final solution, output light curves will be written (`model?.linear`). Two useful additional files will be created: A file called `gridELC.opt` is of the same format at `gridloop.opt`, except that it contains the values of the optimized parameters in place of your initial guesses, and a file called `gridELC.inp` which is the same format as `ELC.inp` except that it contains the optimized parameters in place of the ones specified at the start of the run.

If you are running `loopELC.for`, then the input file might look like this:

```
4
mass ratio
inclination
```

```
t1
none separ
2.0 0.5 4
55.0 0.25 5
6500.  200.  5
128.  10.  2
```

You will compute $4 \times 5 \times 5 = 100$ light curves in three nested loops (the `separ` variable is not adjusted) of 4, 5, and 5 steps. The mass ratio will start at 2.0 and take on the values 2.0, 2.5, 3.0, and 3.5. The code will print the χ^2 value for each parameter set.

If your are running `geneticELC.for`, the `Nvar` lines with three columns have a slightly different meaning. The first column is the *lowest* value of a parameter and the second column is the *highest* value of a parameter. The last column is ignored, *except* for the first two values: the first one is the number of population members (usually around 100) and the second one is the number of generations (usually around 400). In the example above, the part of the input file might look like this:

```
4
mass ratio
inclination
t1
none separ
0.5 10.0 100
30.0 90.0 400
4000.  7000.  1
100.  200.  1
```

In this case, we are fitting for mass ratios in the range of 0.5 to 10, inclinations in the range of 30 to 90 degrees, and effective temperatures (star 1) in the range of 4000 to 7000 K. The `separ` parameter is not adjusted in this case. There are 100 members in the population, and they will evolve over 400 generations.

In the genetic code, if the parameter `rmed >1`, the absolute deviation is used as the measure of fitness (rather than χ^2). This might be useful if there are several outlying points in the light and/or velocity curves.

After each generation, the code will write files called `ELC.???????`, `gridELC.???????`, and `generation.???????`, where `???????`=1000000 after the initial generation, 1000001 after the first generation, etc. The `generation.???????` file contains one line for each population member, and several columns. The first column is the number of the population member (1 to N), the second column is the χ^2 value of that parameter set, and the third to `Nvar+2` columns are the values of the parameters in the order given in `gridloop.opt`. The last one or two columns are the fitted `gamma` value(s).

If your are running `markovELC.for`, the `Nvar` lines with three columns have a slightly different meaning. The first column is the *lowest* value of a parameter and the second column is the *highest*

value of a parameter. The last column is the initial number of steps for the jump:

$$s = (max - min)/N_{\rm step}.$$

In the example above, the part of the input file might look like this:

```
4
mass ratio
inclination
t1
none separ
0.5 10.0 10
30.0 90.0 10
4000.  7000.  10
100.  200.  10
```

There is an optional file called `markovELC.inp` which has three lines, each with a single integer:

`lengthchain` (length of the chains)
`Nchain` (the number of chains to be computed)
`ireport` (the number of iterations to skip before writing out model files).

See Tegmark (2004) for a discussion of the algorithm used.

The program `randomELC.for` uses the `gridloop.opt` file in a very similar manner as `geneticELC.for`. That is, one gives the *ranges* of the parameters. The first step size integer is the number or random parameter sets you want, and the second integer step size number is the number of "best" parameter sets you want to save. In the above example,

```
4
mass ratio
inclination
t1
none separ
0.5 10.0 100
30.0 90.0 4
4000.  7000.  1
100.  200.  1
```

you would generate 100 random parameter sets, and write `ELC.inp` and `gridloop.opt` type files for the 4 sets with the best χ^2 values.

If you are using `itime=1`, the input data need to be three columns, with the first column is the time in days, the second column is the observed magnitude, and the third column is the uncertainty. After an iteration of the optimizer is done, the data folded on the current best-fitting ephemeris will be written into files called:
`ELCdataU.fold`
`ELCdataB.fold`

`ELCdataV.fold`
`ELCdataR.fold`
`ELCdataI.fold`
`ELCdataJ.fold`
`ELCdataH.fold`
`ELCdataK.fold`
`ELCdataRV1.fold`
`ELCdataRV2.fold`

Once you have a solution, you can plot the solution by using any plotting program. Simply plot one of the model?.mag files over your input data, or over the `ELCdata?.fold` files. You can plot the RV curves in the same way, as the gamma velocity is added to the model curve.

The residuals of the fits will be written into files like:
`ELCresidualsU.fold`
`ELCresidualsB.fold`
`ELCresidualsV.fold`
`ELCresidualsR.fold`
`ELCresidualsI.fold`
`ELCresidualsJ.fold`
`ELCresidualsH.fold`
`ELCresidualsK.fold`
`ELCresidualsRV1.fold`
`ELCresidualsRV2.fold`.

6.4 Rotational Broadening Kernels

To compute a rotational broadening kernel, follow these simple steps. First, adjust `ELC.inp` as needed. For the purposes of computing broadening kernels, I suggest you use large numbers for `Nalph` and `Nbet`. After `ELC.inp` is ready, set `idraw=1` and run `ELC`. The rotational broadening kernel(s) will be in the files `star?rotkern.???.??`.

REFERENCES

Avni, Y. 1978, in Physics and Astrophysics of Neutron Stars and Black Holes, ed. R. Giacconi & R. Ruffini (Amsterdam: North Holland), p. 42
Avni, Y., & Bahcall, J. N. 1975, ApJ, 197, 675
Bevington, P. R. 1969, Data Reduction and Error Analysis for the Physical Sciences (New York: McGraw-Hill)
Charbonneau, P. 1995, ApJS, 101, 309
Claret, A. 2001, A&A, 359, 289
Giménez, A. 2006a, A&A, 450, 1231.
Giménez, A. 2006b, ApJ, 650, 408.
Hosokawa, Y. 1953, PASJ, 5, 88.
Lanz, T, & Hubeny, I. 2003, ApJS, 146, 417
Lanz, T, & Hubeny, I. 2007, ApJS, 169, 83
Lucy, L. B. 1967, Zs. Ap., 65, 89
McClintock, J. E., & Remillard, R. A. 1990, ApJ, 350, 386
Press, W. H., et al. 1992, "Numerical Recipes, Second Edition", Cambridge University Press
Orosz, J. A., & Bailyn, C. D. 1997, ApJ, 477, 876 (Erratum: ApJ, 482, 1086)
Orosz, J. A., & Hauschildt, P. H. 2000, A&A, 364, 265
Tegmark, M., Strauss, M. A., Blanton, M. R., et al. 2004, PhRvD, 69, 103501
Van Hamme, W. 1993, AJ, 106, 2096
van Kerkwijk, M. H., Rappaport, S. A., Breton, R. P., et al. 2010, ApJ, 715
von Zeipel, H. 1924, MNRAS, 84, 665
Wilson, R. E., & Devinney, E. J. 1971, ApJ, 166, 605

Wilson, R. E. 1979, ApJ, 234, 1054
Wilson, R. E. 1990, ApJ, 356, 613