Computing Binary Star Observables

R. E. Wilson
Astronomy Department, University of Florida, Gainesville, FL 32611

Revision of September, 1998

1. Introduction

The model has been described and quantified in papers by Wilson and Devinney (1971) and by Wilson (1979, 1990, 1993) that include its main theory, organization, and concepts, as well as much of the mathematics. The reader is referred to those papers for background. Since it would be impractical to cover the programming ideas within a reasonable length, only the 1993 paper goes significantly into programming, and there only in abbreviated form. This booklet also is not an explanation of programming ideas - its purpose is to tell how to use the model.

The overall program consists of a main FORTRAN program (**LC**) for generating light and radial velocity curves and spectral line profiles, a differential corrections main program (**DC**) for parameter adjustment of light and velocity curves by the least squares criterion, and somewhat over two dozen subroutines used by both main programs. [In this monograph, program and subprogram names are in boldface type (*e.g.* **LC**), and FORTRAN variable names are in the sans serif style (*e.g.* XINCL)]. The present program has grown over the years from much simpler beginnings. Growth has consisted of occasional improvements in regard to generality, speed, and elimination of bugs, perhaps about every 6 months, punctuated by three revisions. The revision of 1982 introduced eccentric orbits, asynchronous rotation, capacities to do several new kinds of constrained solutions, computation of velocity curves (with proximity and eclipse effects), simultaneous light and radial velocity solutions, and a simple star spot capability. Most features of the 1982 version are described in Wilson (1979). The revision of 1992 had options for detailed reflection and non-linear (logarithmic) limb darkening, adjustment of spot parameters, an optional provision for spots to move with the rotating surface, capability for following light curve development over large numbers of orbits, and greater speed. At that time an accuracy improvement was expected for the near term, but insufficient resources have been available for completion of that project, which is still on the "back burner". Instead, this third revision (1998) includes semi-transparent circumstellar clouds, a simple spectral line profile capability for fast-rotating stars, inclusion of the Marquardt λ factor in differential corrections solutions, generation of sky coordinates for images, an option to work with either observed times or phases, additional solution parameters (T_0, P_0, dP/dt, and $d\omega/dt$), conversion of the entire program to double precision, and some other improvements that are listed near the end of the booklet. An accuracy improvement still is expected, but the effective date is hard to predict.

2. Running the Programs

To make things simple, the light and velocity curve program (**LC**) and the differential corrections program (**DC**) are supplied complete with sample input data sets, so that all one needs to do to get started is to run the programs with the input data. There should be few, if any, machine-dependent problems, as those have been eliminated via feedback from users. If such a problem is found, please communicate it to R.E.W. However there may be a trivial problem with the arc sine and arc cosine routines, which are DASIN and DACOS on some systems and DARSIN and DARCOS on others. That problem can easily be fixed. Then just change the numbers to run a particular binary star problem. Be sure to keep at least one copy of the supplied sample data in case the input formatting gets scrambled or shifted. Of course, one always can re-construct the correct format by comparing the program's READ and FORMAT statements with the input lines, but save the original data anyway in the interest of keeping things simple. It is not recommended to change the programs, but if you must, be sure to keep a copy of the original version for comparison. Tinkering can introduce a bug that may not show up in particular cases, but jumps out at you later on.

When compiling the programs, the simplest procedure is to form separate executable modules for **LC** and **DC**. The programs are supplied in one large file, with the first few hundred lines being the **LC** main program, the next roughly 1500 lines being the **DC** main program, and the rest being the two dozen or so subroutines used by both main programs. The light-velocity-line profile module should contain **LC** and all of the subroutines except **SQUAR**, which it does not need. Of course, no harm is done by including unnecessary subroutines. The differential corrections module should contain **DC** and all of the subroutines except **MLRG**, which it does not need.

The type of output produced by **LC** is determined by control integer MPAGE and can be light curves (MPAGE=1), radial velocity curves (MPAGE=2), spectral line profiles (MPAGE=3), star dimensions (MPAGE=4), or sky coordinates for producing images (MPAGE=5).

For MPAGE=1 (light curves), the first four columns contain time (col. 1), phase (col. 2), separate light for stars 1 and 2 (cols. 3, 4), and the combined system light from stars 1 and 2 plus third light (col. 5). **Star 1 is, by definition, the one at superior conjunction near phase zero when parameter PSHIFT is entered as zero**. It will be the one eclipsed near phase zero if there are eclipses. Light is in the program unit, which is explained in Wilson (1993). The program unit is, in a sense, an absolute unit of observable flux because it can be converted to absolute flux if definite star luminosities and a definite observer distance are specified. Light (observable flux) is discussed in section 4. Column 6 contains the light of column 5 re-scaled (normalized) to a specified input value (labeled FACTOR) at a specified input phase (labeled PHN). For example, one can require that the normalized light be 1.2000 at phase 0.1500. This re-scaling provision is only for convenience in working with graphs, and has no relevance to differential corrections solutions. That is, the **DC** program works only with absolute flux (column 5 of **LC** output) and does not even know about normalized light. Column 7 is the star separation, with the unit being the relative orbital semi-major axis ($a = a_1 + a_2$). Column 8 is the system brightness expressed in magnitudes, with a user-specified zero point (labeled MZERO). For example, if MZERO is +7.300, column 8 will read 7.300 at the phase of normalization (PHN), which is the same phase of normalization used for column 6.

For MPAGE=2 (radial velocities), the first two output columns are again time and phase. Columns 3 and 4 are dimensionless radial velocities for the two stars (in circular relative orbit circumference [$2\pi a$] per orbit period). Columns 5 and 6 are the eclipse-proximity corrections corresponding to columns 3 and 4, respectively, and in the same dimensionless unit. Columns 7 and 8 are velocities in kilometers per second (if VUNIT was entered as 1.00, or in general the velocities in unit VUNIT).

For MPAGE=3 (spectral line profiles). The output is by blocks according to phase, with results for stars 1 and 2 given first and second, respectively, within each phase block. Column 1 is the equivalent velocity difference between the reference wavelength and the wavelength of the profile point, in unit VUNIT. Column 2 is the corresponding wavelength difference, in microns. The reference wavelength is that entered with the main binary star parameters (labeled "wv lth" in the output). Column 3 has the actual wavelengths of the profile points, in microns. Column 4 is the profile in terms of a flat continuum at flux 1.0000000. Column 5 is the profile in terms of a continuum that can be shifted vertically and can have a slope. Guard against generating excessive output with MPAGE=3 (usually one does only one phase at a time). Input for MPAGE=3 operation is described in Section 6.

For MPAGE=4 (star figures), **LC** lists the pole, point, side, and back radii of each star vs. time and phase. This provision can be useful for eccentric orbit cases, as it shows the variation of figure with phase.

For MPAGE=5 (binary star images), **LC** produces only two output columns for each time/phase (beyond the usual header information, essentially composed of the input quantities and their labels). Those columns are y_{sky} and z_{sky} rectangular plane of sky coordinates of the projected surface elements of the two stars. A picture of the binary at a given phase can be made by sending just those two columns to a plot program such as GNUPLOT, MONGO, etc. Any spots or parts of spots that may be in view will show in the pictures, although there is no distinction in the pictures between bright and dark spots. The images show only spot location, not spot surface brightness (there is no gray scale). The origin $[0, 0]$ of the image coordinates is at the system center of mass.

Adjustment of parameters while fitting light and velocity curves normally involves both subjective (**LC**) and objective (**DC**) iteration. It is assumed here that the user has reasonable background knowledge of binary stars and can show good judgment in deciding which parameters to fix from theory, which to fix from other kinds of observations, and which to adjust. Solution constraints, if there are to be any, must also be decided upon. In most situations, some of the decisions are obvious while others are debatable, and seldom will two persons make the same set of choices, even if given exactly the same circumstances. Therefore this section goes only into the "mechanics" of solutions, which means how to use the **LC** and **DC** programs.

Most persons will want to begin with a quick graphical fit, so as to be somewhere near to a proper solution. It is a good idea to interface **LC** to a graphics package to facilitate eye inspections. The following discussion specifically concerns light curves (MPAGE=1 in **LC**), although radial velocities (MPAGE=2) would be fitted in much the same way. It is best to be ready to feed either column 5 (light in program units) or column 6 (normalized light) to the graph. Remember that the normalized light column has no counterpart in **DC**, so after a few first cuts with normalized light (to reach rough agreement with the observations) switch from use of column 6 to use of column 5. At this point one should effect a correct transfer such that the column 5 numbers will be approximately the same as the column 6 (normalized) numbers that have been made to fit the light curve(s). This little problem involves the estimation of a scaling constant, or sometimes more than one. Please consult section "**Luminosity vs. Light (Flux)**" at this point if you are not conversant with the distinction between "light" or "flux" on the one hand (an output quantity), and "luminosity" on the other (an input quantity). This will sound trickier than it is, but do not be concerned with making a perfect transfer from column 6 to column 5 output, because you are iterating anyway and only need to get close. Remember that the light of each star scales with its luminosity, while third light (l_3) is just a direct add-on. Also remember that in all modes of program operation except modes 0 and -1, the luminosity of star 2 (L_2) is computed by the program and scales with that of star 1 (L_1), assuming fixed values for all other parameters. Therefore the main, or perhaps only, scaling parameter in most situations is L_1. The simplest case is in mode 1, 2, 3, 4, 5, or 6 with no third light. Then, for example, doubling L_1 will double column 5 output light. If there is third light, that also should be doubled if column 5 light ($l_1 + l_2 + l_3$) is supposed to double. In mode 0, L_2 does not scale with L_1, so L_2 also would then need to be doubled. In mode -1, L_1 has only minor influence (through reflection heating), so L_2 and l_3 mostly control the scaling, although L_1 also needs to be scaled in the general case. So to make column 5 agree with column 6, bump L_1 and l_3 (most modes) or L_1, L_2, and l_3 (modes 0 and -1) up or down by a fixed factor.

When you use normalized output the idea is to select some phase where you make the synthesized curve match the observed curve essentially exactly, so that you can concentrate on form rather than the absolute scale. Usually one chooses some innocuous phase outside eclipse where the brightness is not changing very fast. Vertical shifts of the whole normalized curve are made by changing input quantity FACTOR, if working in direct light, or MZERO if working in magnitudes. After you switch to column 5 output (getting ready for **DC**), control of vertical scaling passes to L_1, L_2, and l_3, as explained above. After gaining some experience with **LC**, you will be using the normalized output only for a few preliminary runs and staying with program-unit light thereafter.

To start running **DC**, begin with the sample data, which contains control integers, initial parameter estimates, radial velocity curves for a primary and a secondary star, and light curves for the sample binary at three effective wavelengths. The sample data file is set up to do a simultaneous solution of the two velocity curves and the three light curves. Run **DC** and inspect the output to be sure you understand everything. You may want to change a few things in the input to verify that the output responds as expected.

To begin a solution with your own data, change the sample input numbers to those appropriate to the observed binary, being careful to keep the original format. By this time you should have made a reasonable number of fitting experiments with **LC**, decided on the mode of program operation, decided on the parameter set and subsets to be adjusted, and estimated the SIGMA's (standard deviations) of the observed curves (see below). If you will be fitting normal points, they should have been formed. You should also give a reasonable amount of thought to the sizes of the increments (DEL's) used in forming the numerical derivatives. It is easy to fall into the error of trusting the DEL's in the sample data to be appropriate for your binary, and sometimes they will be, but you cannot rely on that assumption. For example, suppose the mass ratio DEL is 0.01 and you just leave that value in for two binaries with mass ratios of 5.00 and 0.10. The increment for the first binary is one part in 500 (probably too small), while that for the second is one part in 10 (grossly too large, of course). Also remember not to try star spots that are unduly small in angular radius, because there are no fractional areas to the spots - they have staircase-like boundaries where their idealized circular outlines encounter the finite grid elements. Of course, the roughness of this simple treatment becomes particularly troublesome for small spots, and very small spots might not show up at all. Differential correction of spot parameters requires fine grids and considerable patience.

DC writes a table of numbers that are useful for estimating the SIGMA's (standard deviations=σ's) of the 1 or 2 velocity curves and NLC light curves. The table is labeled "*Sums of squares of residuals for separate curves, including only individual weights*", which is exactly what is listed (neither curve-dependent nor level-dependent weights are included). The sums of squares are just listed in the order their curves occur in the input. To form SIGMA's, calculate

$$\sigma = \sqrt{\frac{\Sigma W r^2}{n - m}}$$

where n is the number of data points in a curve and m is the number of adjusted parameters in the subset of primary interest. Usually n is much larger than m, so that the actual value of m makes little difference. The SIGMA's are used by **DC** to calculate curve-dependent weights - that is, to control the relative influences of the various curves. One can enter rough guesses for the SIGMA's in the first few iterations, then calculate them properly for the middle runs and fine tuning runs. The rough guesses can be made by putting a straight line through a representative section of the data.

3. Modes of Program Operation (Solution Constraints)

Imagine that you know or are convinced of something about possible parameter values, and your "fact" is not in the form of a definite parameter value, but rather is a functional relationship among parameters. You do not claim to know the value of parameter c, but if someone tells you the values of parameters a and b, you then can compute c. The most common example is connected with one of the stars accurately filling a limiting lobe, as in a semi-detached case. It is impossible (or anyway, inadvisable) to say in less than a small monograph why we believe in the phenomenon of lobe filling and our ability to recognize semi-detached binaries as such. Of course, there are binaries for which we cannot be sure - the lobe may or may not be filled - but there are many for which we would "bet the ranch" that they are. In fact, for more than a few of the classical Algol type binaries, conflicts between mass ratios determined from radial velocities and mass ratios found from a lobe filling condition (through light curves) were eventually resolved in favor of lobe filling and light curves. S Cancri is a good example. For the simple case of a circular orbit and synchronous rotation, the mass ratio, q, of a semi-detached binary determines the relative size of the lobe filling star, R_lobe/a. Since the model expresses star size via the Ω potential energy function, only q or Ω (but not both) should freely be adjusted. If we have lobe filling with non-synchronous rotation there is a functional relation among three parameters (now also F to express the rotation rate), and we add parameters e and ω if the orbit is eccentric. In general, application of the lobe filling condition, or another condition, reduces the number of free parameters by one. Now because every parameter enlarges and complicates the parameter correlation matrix and thus weakens the solution, we should take advantage of every opportunity to eliminate free parameters. Essentially we thereby introduce information from sources external to the light and velocity curves, and we tell the least squares program "here is something about the binary that you cannot figure out from the data, but we are letting you know so that you can eliminate a whole dimension of incorrect solutions". Each such condition can be represented as a constraint among possible parameter values, and a constraint or set of constraints is identified with a particular **operation mode** of **LC** and **DC**. Parameters that are set by constraints are no longer free parameters. The program computes them from the constraint relations and ignores their input values. Of course, they cannot be adjusted. Unfortunately, a few authors have failed to understand the idea of constrained solutions and have made comments something like "our program allows each star to have the radius that gives the best fit to the light curve, so that application of a lobe filling constraint is unnecessary". This is akin to saying "we amputate the legs so as to make walking unnecessary".

Mode -1: This mode is for X-ray binaries for which the eclipse duration of a compact object is known from X-ray observations. The compact star must be star 1. The applied constraint is that the surface potential of the ordinary star (star 2) must be such as to produce the observed X-ray eclipse duration. Therefore the program computes the Ω_2 potential from the input q, F_2, e, ω, i, and X-ray eclipse semi-duration, ϕ_e (FORTRAN name is THE). Note that solutions are not limited to circular orbits or synchronous rotation - any e between 0 and 1 is allowed, as is any F_2. Note also that lobe filling is not implied.

Mode 0: No constraints are applied in mode 0, and the component luminosity ratio is not even required to be consistent with the surface temperatures. That is, the program uses L_1 and L_2 as supplied, and does not re-compute L_2, based on the input temperatures. A star may even be larger than its Roche lobe in mode 0 (and will then have a hole in one end). Among the various modes, mode 0 is the closest analog to the Russell model.

Mode 1: This is a mode for overcontact binaries, such as W UMa stars. Seven constraints are applied. The first is that the surface potential of star 2 is equal to that of star 1 ($\Omega_2 = \Omega_1$). Another is that the polar temperature of star 2 is set by the gravity brightening law of the entire common envelope. This is done via equation 8 of Wilson (1979). Other constraints are that the gravity brightening, bolometric albedo, and limb darkening parameters of star 2 are the same as those of star 1. Finally, the luminosity of star 2 is computed from the other parameters via black body or stellar atmosphere radiation formulas. Therefore the seven parameters Ω_2, T_2, g_2, A_2, L_2, x_2, and y_2 are not free in mode 1. A consequence of mode 1 operation is that there is a smooth variation of surface brightness over the entire common envelope with no discontinuity where the stars join together and thus one continuous gravity brightening law. In mode 1 light curve fitting experiments, there will be essentially no freedom to change the relative primary and secondary eclipse depths, which will be set almost entirely by geometry (although a little by gravity brightening, limb darkening, etc.).

Mode 2: This is for detached binaries with no constraints on the potentials. The only constraint is that the secondary luminosity, L_2, is computed from the other parameters via the specified radiation prescription (black body or approximate stellar atmosphere). That is, L_2 is coupled to the temperatures. The temperature-luminosity coupling can be severed by setting control integer IPB to 1 (normal value = 0). Mode 2 is the same as mode 0 except for the one constraint, and is exactly like mode 0 if IPB is set to 1.

Mode 3: This is for overcontact binaries and is the same as mode 1, except that the constraint on T_2 is not applied.

The other six constraints of mode 1 are applied in mode 3, but T_2 is a free parameter and can be adjusted. The stars can be overcontact in mode 3, yet have much different surface brightnesses. In structural language, they can be in geometrical contact without being in thermal contact.

Mode 4: This is for semi-detached binaries with star 1 accurately filling its limiting lobe, which is the classical Roche lobe for synchronous rotation and a circular orbit, but is different from the Roche lobe for non-synchronous and eccentric cases. The applied constraints are that Ω_1 has the lobe filling value and that L_2 is coupled to the temperatures (unless IPB=1).

Mode 5: The same as mode 4, except that it is star 2 that fills its limiting lobe. This is the usual mode for Algol type binaries.

Mode 6: This is for double contact binaries, in which both stars accurately fill their limiting lobes (Wilson, 1979). Astrophysically this makes sense only if at least one of the stars rotates non-synchronously. Mode 6 applies the surface potential constraints of modes 4 and 5 together.

Note on lobe filling: The program operates under a general definition of a limiting lobe that applies for non-synchronous as well as synchronous rotation and for eccentric as well as circular orbits. The classical Roche lobe is a special case that obtains for the synchronous, circular orbit case. In the general definition, a limiting lobe is an equipotential for which the effective gravity is zero on the line of star centers at periastron. In a strict sense, equipotentials do not exist in the eccentric orbit case, but in essence the concept of equipotentials still is useful in most realistic situations (Avni, 1976; Wilson, 1979).

4. Luminosity vs. Light (Flux)

It is essential to emphasize the very important, yet commonly ignored, distinction between "light" or "flux" on the one hand, and luminosity on the other. As simple as this is, it has led to remarkable confusion in the literature. The problem has its roots in the non-spherical models of past decades that could have made the distinction but did not, and it has carried over into the general binary star literature of today. A luminosity as discussed here is characterized by an effective wavelength and other bandpass properties, and usually is described "monochromatic" in common usage, although "polychromatic" would be a more logical term. Where it is necessary to refer to a bolometric luminosity, the qualifier "bolometric" can specifically be attached. In some papers, unqualified luminosity means bolometric luminosity, but in this document unqualified luminosity means monochromatic luminosity.

A discussion of this point is in the description of subroutine **LUM** in Wilson (1993). The main thing to remember is that luminosity is a global quantity, represented by a single number for each star and bandpass. It does not depend on aspect (neither on inclination nor on phase). Some authors write of "normalizing luminosities by the sum of the luminosities outside eclipse". However, luminosity has nothing to do with eclipses or their absence, nor with any other aspect-dependent variation. Flux (or, in more relaxed terminology, "light") is a directly observable (aspect-dependent) quantity, while luminosity is a model parameter that cannot be observed directly but must be inferred as part of a photometric solution. The star luminosities, L_1 and L_2, still can be normalized by their sum (a common and reasonable practice) but two points need to be kept in mind. The first is that the normalization must be made only after completion of a solution, not during its progress. The reason is that the luminosities are the main handles for scaling of (output) flux. As such they need to float freely so that the model is able to match the observed fluxes, which almost always are in an arbitrary unit, since the measurements are made with respect to the flux from a comparison star. The second point is that L_1 and L_2 are not the same kind of quantity as third light (a flux), and are not to be compared directly with third light. This is important because many papers have listed values of third light with no indication of what the number means. Not only is the printed l_3 thereby rendered useless, but strictly speaking the entire solution is made meaningless because l_3 usually is significantly correlated with other parameters. An estimate of third luminosity, based on third light, involves an assumption about the direction dependence of the radiation of the third source. If that radiation is emitted isotropically, then third luminosity, L_3, will be 4π times l_3, which is per $1/4\pi$ of the area of a sphere centered on the binary ("a steradian's worth of area"). That L_3 value is to be compared with the un-normalized L_1 and L_2. A meaningful way to specify the unit of a published l_3 is to express it in the light (relative flux) of the multiple star system at a definite phase. For example, suppose the solution output light is 1.0500 at reference phase 0.2500 for a particular light curve. Then divide l_3 and its error estimate by 1.0500, list those numbers, not the direct l_3 program output, and tell the reference phase in a footnote to the table of results.

5. Circumstellar Light-Attenuating Regions

The light curves of some interacting binaries show effects of attenuation of star light by circumstellar matter. Most such matter is gaseous, although dust attenuation may be significant in a few cases. The attenuation may be due to Thomson scattering, Rayleigh scattering, another scattering process, or true absorption. Since circumstellar matter follows dynamical trajectories, one might expect there to be little interest in attenuating regions that are fixed in a coordinate frame that rotates with the binary, but such is not entirely the case. A few binaries (e.g. RZ Sct, AX Mon) have approximately stationary loci of circumstellar gas that distort their light curves. The loci may be stream-stream, stream-disk, or stream-wind interaction regions. Efforts to represent such light curve distortions via bright or dark star spots sometimes can rule out a spot explanation and point to essentially fixed attenuation regions (hereafter "clouds", for brevity). The model includes n spherical semi-transparent clouds specified by their locations (x, y, z) [in a rectangular frame that co-rotates with the stars], cloud radius (r), density (ρ), electron density (n_e), and mean molecular weight per free electron (μ_e). The x coordinate is zero at the center of star 1 and increases toward star 2, reaching $+D$ at the center of star 2. The x, y, z system is right handed and serves for the entire binary system. The part of the line of sight that passes through the various clouds is computed individually for the lines of sight to all surface points and individually for all clouds. Regions of variable density can be made by nesting individual clouds. Regions of non-spherical shape can be approximated by overlapping spherical clouds. Each cloud is allowed its own attenuation law, whose general form is

$$\frac{d\tau}{ds} = \sigma_e n_e + (\kappa_\lambda + \kappa_{sb})\,\rho \tag{1}$$

where τ is optical thickness, σ_e is the Thomson scattering cross section per electron, s is distance along the line of sight (in cm), κ_λ is a wavelength-dependent opacity, and κ_{sb} is an additional opacity for a specific bandpass (κ in cm^2/g). The κ_{sb} term might represent, for example, opacity due to absorption lines averaged over a particular bandpass. The κ_λ term is

$$\kappa_\lambda = \kappa_0 \lambda^\alpha \tag{2}$$

where κ_0 and α are input quantities. Each cloud has its individual κ_0, α, and κ_{sb}. However to make it easy to change the κ_{sb}'s of all clouds together, the κ_{sb}'s are not entered directly as individual numbers. Instead one enters an overall κ_{sb} and the fraction of it that applies for each cloud. Thus

$$\kappa_{sb} = f_i \kappa_{\rm sb0} \tag{3}$$

where all the f_i can be unity if κ_{sb} is to be the same for all clouds, or non-unity if κ_{sb} is to differ from cloud to cloud. The program figures absolute lengths from the system geometry, including the orbital semi-major axis.

At present the clouds only attenuate starlight that passes through them, but they may be made to scatter starlight toward the observer in a future program version.

6. Spectral line profiles

Profiles of absorption and emission lines are generated when input integer MPAGE is set to 3. The profiles are for rotation only, although other broadening mechanisms will be added later. Blending is incorporated, including blending of mixed absorption and emission lines. Lines can originate either from an entire star or from designated sub-areas of the surface, as explained below. Only **LC** (not **DC**) computes line profiles at present. Spectra are formed by binning, with the spectra of the two stars formed separately. The user can add them (weighted by observable flux) if spectra of the binary are needed. Computations for each star are characterized by four quantities related to spectral and accuracy characteristics (BINWM1, SC1, SL1, NF1 for star 1 and BINWM2, SC2, SL2, NF2 for star 2), that have the following meanings:

BINWM1(2): The bin width in microns. Too small a bin width gives noisy profiles. Too large a bin width gives insufficient spectral resolution.

SC1(2): The continuum scale (continuum flux at the reference wavelength). The unit is decided by the user.

SL1(2): The continuum slope in flux units per micron.

NF1(2): Grid fineness for micro-integration on each surface element. NF1(2)=1 means that there is no micro-integration. NF1(2)=n breaks each surface element into n^2 pieces, thus improving integration accuracy.

With MPAGE = 3, an **LC** input line with the above quantities is required for each star, even if one of the stars is assigned no spectral lines (see sample input file LCIN.DAT3). **DC** program input does not have the analogous data lines at present.

Following the star-dependent input are data for individual spectral lines in two sets (for star 1, then star 2). The quantities are (FORTRAN names):

WLL1(2): The line wavelength in microns for a line of star 1 or (2).

EWID1(2): The line equivalent width, in microns - the traditional measure of line strength, for a line of star 1 or (2).

DEPTH1(2): Rectangular line depth for a line of star 1 or (2). Line profiles are formed by binning of Doppler shifted elements that have rectangular profiles, each with a depth and a width. The user supplies the depth and the program then calculates the width needed to reproduce the specified equivalent width. The depth is relative to a unit continuum, so 0.80000 means that 80 percent of the continuum flux is missing within the rectangular profile element, or that the residual flux is 20 percent of the continuum. Negative depths correspond to emission lines, so -0.50000 means 50 percent above the continuum.

KKS: This integer specifies a surface region associated with a given spectral line. If KKS=0, the line is not specific to a location but applies to the entire star. If KKS=1, then the line applies only to the first spot on that star; if KKS=2 it applies only to the second spot, and so on. Naturally the star must have spots for this scheme to work, but the spots need not be hot or cool spots - they can have temperature factors of unity. Negative KKS specifies avoidance of regions. Thus KKS=-4 means that the spectral line applies everywhere on the star except within spot 4. If you find this confusing, just set KKS=0 and the line applies in the old simple way - everywhere on the star.

7. Model parameters

These are the same for the **LC** and **DC** programs, although some of the parameters are not among the 35 that can be adjusted by the present version of **DC**. The actual number of parameters subject to adjustment is greater than 35 for two reasons. First, although **DC** can adjust the parameters of at most two star spots in any one run (iteration), successive iterations can adjust the parameters of different spots (viz. definitions of KSPA, KSPB, NSPA, and NSPB in Section 9). Second, curve dependent parameters (limb darkening coefficients, relative monochromatic luminosities, third light) have n values for n simultaneously fitted light curves. For example, specifying the adjustment of l_3 in a simultaneous solution of four light curves will produce four l_3 corrections, one for each curve. The word "curve" is used here in the sense of "light curve", "radial velocity curve", etc. Parameters are said to be "curve dependent" or "curve independent" rather than "wavelength dependent" or "wavelength independent" because **DC** may sometimes deal with more than one curve at a given wavelength (for example, wide and narrow band light curves at the same effective wavelength). Parameters are identified below according to their FORTRAN names.

7.1. Curve-independent parameters that cannot be adjusted by DC are:

THE (ϕ_e): The semi-duration of primary eclipse in phase units (*i.e.* range 0 to 1 for a whole cycle). This parameter is used only in mode -1 and relates only to binaries in which star 1 is a compact object (*i.e.* has negligible size compared to the other star). The idea is to fix ϕ_e according to X-ray eclipses of a neutron star, black hole, or white dwarf, and require the overall solution to be consistent with that value. The orbit can be eccentric and rotation can be non-synchronous. Parameter ϕ_e is ignored in all other operation modes. See the explanation of mode -1 in Section 3.

XBOL1, XBOL2 (x_{bol1}, x_{bol2}): The coefficients of $\cos\gamma$ in the bolometric limb darkening law. Used only in computation of "detailed" reflection (Wilson, 1990), with MREF=2. See y_{bol1}, y_{bol2} below for the complete law and explanation.

YBOL1, YBOL2 (y_{bol1},y_{bol2}): If control integer LD=2, these are the coefficients of the $\cos\gamma\,(\ln\cos\gamma)$ term in the bolometric logarithmic limb darkening law. The complete logarithmic law is:

$$\frac{I}{I_0} = 1 - x + x\cos\gamma - y\cos\gamma\,(\ln\cos\gamma)\,,$$

which was advocated by Klinglesmith and Sobieski (1970). If LD=3, they are the coefficients of the bolometric square root law. The complete square root law (Diaz-Cordoves and Gimenez, 1992) is:

$$\frac{I}{I_0} = 1 - x + x\cos\gamma - y\,(1 - \sqrt{\cos\gamma})$$

Coefficients for all these darkening laws have been tabulated by Van Hamme (1993).

XCL, YCL, ZCL: Rectangular coordinates of the centers of spherical circumstellar clouds. See above section *Circumstellar Light Attenuating Regions.*

RCL: Radii of individual circumstellar spherical attenuating regions in unit of a.

EDENS: Electron densities, n_e, in cm^{-3} for individual attenuating clouds. For a given cloud, n_e is constant.

XMUE: Mean molecular weight in atomic mass units per free electron, μ_e, for individual attenuating clouds, and constant throughout a given cloud.

ENCL: Exponent α in the wavelength-dependent term of the attenuation law for individual attenuating clouds (see *Circumstellar Light Attenuating Regions).* The program internally computes densities, ρ, from n_e and μ_e.

7.2. Curve-dependent parameters that cannot be adjusted by DC are:

WLA: The observational wavelengths in microns of each light or velocity curve. Wavelengths need to be entered for velocity curves, although they have little effect on the output, and any wavelength somewhere near the spectral region of interest should be adequate.

Y1A, Y2A (y_1, y_2): These are the wavelength-specific limb darkening coefficients *in the non-linear terms.* The laws have the same forms as for bolometric limb darkening (above). There is one value for each of these coefficients for each light or velocity curve or set of line profiles. Enter integer parameter LD=1 for the linear cosine law (in which case y_1 and y_2 are assumed to be zero), LD=2 for the logarithmic law, or LD=3 for the square root law. Although y_1 and y_2 cannot be adjusted by the present version of **DC**, they may be adjustable in a future version.

7.3. Curve-independent parameters that can be adjusted by DC are:

HJD0 (t_0): This is the zero point of the orbital ephemeris. Usually one uses Heliocentric Julian Dates, although that is only a convention and any consistent system of time can be used.

PERIOD (P): The binary orbit period at the reference time, t_0, ordinarily in mean solar days. P affects radial velocity amplitudes and is used to compute phase from time (if JDPHS=1) and time from phase (if JDPHS=2). P can be adjusted only if JDPHS=1 and observation times (rather than phases) are entered.

DPDT ($\frac{dP}{dt}$): The first time derivative of the orbital period. Second and higher derivatives are not used in the present program. DPDT can be adjusted only if JDPHS=1 and observation times (rather than phases) are entered. This quantity is dimensionless.

E (e): Binary orbital eccentricity.

PERR0 (ω_0): Initial argument of periastron **for star 1**. The argument of periastron for star 2 differs by π radians. PERR0 is ω at the reference time of the ephemeris, t_0. The program is written so that, as the argument of periastron changes, both eclipses have excursions in phase in the same way as a real binary. In former versions the input unit was degrees but the corrections produced by **DC** were in radians, so one had to convert units to apply the corrections. In the new version, both the input and the corrections are in radians, which should reduce confusion. The DEL for ω also is in radians. For circular orbits, the program ignores the input value and sets ω to $\pi/2$ radians.

DPERDT ($\frac{d\omega}{dt}$): The first time derivative of ω. DPERDT can be adjusted only if JDPHS=1 and observation times (rather than phases) are entered. The instantaneous $\omega = \omega_0 + \frac{d\omega}{dt}(t - t_0)$. The present program does not consider any more complicated variations of ω. The unit is radians per adopted time unit (usually a mean solar day).

A (a): The length of the semi-major axis of the relative orbital ellipse, in solar radii ($6.960\,10^5$ km). It is the sum of the two absolute semi-major axes, so $a = a_1 + a_2$.

F1, F2 (F_1, F_2): The ratio of the (constant) axial rotation rate to the mean orbital rate for stars 1 and 2, respectively. The angular rotation is assumed to be uniform (not latitude dependent). Value unity represents synchronous rotation in a circular orbit. In eccentric cases it is expected that rotation will tend to synchronize to the periastron angular rate because of the strong dependence of the tide raising force on distance. Periastron-synchronized F is given by:

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

The F's affect star figures, surface brightnesses (gravity effect), limiting lobe sizes, and thus both light curves and radial velocities.

VGA (V_γ): The radial velocity of the binary system center of mass (assumed constant) in the unit (so many km/sec) specified by the input quantity VUNIT.

PSHIFT (ϕ_0): A constant shift applied to computed phases. Usually one enters +0.0000 for ϕ_0, but it may be convenient to shift the phases. For example, a phase shift can effectively interchange the star labels (1 vs. 2) without altering the observational data. The main purpose of ϕ_0 is to allow the **DC** program to adjust for a zero point error in the ephemeris used to compute the phases. The unit is an orbital cycle. One should not adjust both PSHIFT and HJD0 because they will be perfectly correlated.

XINCL (i): The binary orbital inclination to the plane of the sky, in degrees. If the inclination is in the range 0 to 90°, the binary orbits counter-clockwise as projected onto the plane of the sky, while above 90° it orbits clockwise, according to the coordinate conventions adopted for the program. Those conventions were different prior to the revision of 1992.

GR1, GR2 (g_1, g_2): The exponents in the bolometric gravity brightening (*a.k.a.* darkening) law for stars 1 and 2 respectively. Value 1.000 means that bolometric flux is proportional to local effective gravity, while value 0.000 means that it is constant over the surface (ignoring other effects such as spots, reflection heating, etc.). The g's are expected to be unity for radiative envelopes, while they should be smaller for convective envelopes, perhaps about 0.3. Some other programs use a gravity brightening exponent as expressed in terms of effective temperature, for which the usual symbol is β. The quantities are related by $g = 4\beta$.

TAVH, TAVC (T_1, T_2): The mean surface effective temperatures of stars 1 and 2, respectively. The mean is weighted by the local bolometric flux. The program accepts T_1 and T_2 as model parameters and converts to local surface temperatures for internal computations. The conversion between mean and polar temperatures is made via eqn. 8 of Wilson (1979), and the local surface temperatures are then computed from the polar temperatures and the gravity brightening law. The unit is 10000 K.

ALB1, ALB2 (A_1, A_2): The bolometric albedos for reflection heating and re-radiation on stars 1 and 2, respectively. The bolometric albedo is the local ratio of re-radiated bolometric energy to received bolometric energy. It is assumed to be constant for each star. The expected value for radiative envelopes is 1.00, while for convective envelopes it should be perhaps 0.5, although observations sometimes indicate values between 0.5 and 1.0.

PHSV, PCSV (Ω_1, Ω_2): These are the "potentials" for stars 1 and 2, respectively, that originally were defined by Kopal for the synchronous, circular orbit case. They would be actual potentials, except that a term was deleted in Kopal's convention. The deleted term depends on mass ratio but not on position, so Ω gradients are equivalent to potential gradients. Fixed Ω specifies a constant potential energy (gravitational plus centrifugal) over the surface of each star. A generalized defining equation (eqn. 1 of Wilson, 1979), based on contributions by Plavec (1958), Limber (1963), and Avni (1976), allows the Ω's to serve also for non-synchronous rotation and eccentric orbits. Together with the mass ratio, rotation rate, orbital eccentricity, argument of periastron, and phase, the Ω's specify the size, figure, surface gravity, and certain other geometric properties of the stars. Special values of the Ω's correspond to exact filling of limiting lobes.

RM (q): The mass ratio, m_2/m_1, of stars 1 and 2.

XLAT: The "latitude" of a star spot center, measured from 0 radians at the "north" ($+z$) pole to π radians at the "south" pole. XLAT's are subscripted by star (1 or 2) and by spot number on a star (viz. explanation of KSPA, NSPA, KSPB, and NSPB in Section 9.

XLONG: The longitude of a star spot center, measured counter-clockwise (as viewed from above the $+z$ axis) from the line of star centers from 0 to 2π radians. XLONG is subscripted in the same way as XLAT.

RADSP: The angular radius of a star spot, in radians. The angle is subtended by the spot radius at the center of the star. RADSP is subscripted in the same way as XLAT.

TEMSP: The "temperature factor" of a spot, that specifies the local spot temperature compared to the local temperature that would obtain without the spot. A TEMSP larger or smaller than unity corresponds to the spot being hotter or cooler than the un-spotted surface, respectively. TEMSP is constant for a given spot, but local temperature will vary over a spot if the underlying un-spotted temperature varies. TEMSP is subscripted in the same way as XLAT.

7.4. Curve-dependent parameters that can be adjusted by DC are:

HLA, CLA (L_1, L_2): Monochromatic luminosities for stars 1 and 2, respectively. There has been some confusion about the units of luminosity and of light, which is partly due to a tradition in the binary star field of normalizing luminosity and light separately, and failing to recognize that they are fundamentally distinct (although connected) quantities (*viz.* Section 4). The program luminosity unit is effectively user-supplied, and determines the unit of output light (i.e. ℓ_1 or ℓ_2 monochromatic flux). The output flux will integrate to the luminosity over a sphere at any large distance, centered on the binary system. The computed fluxes will be in the unit $1/4\pi$ luminosity units/(steradian's worth of area). Now this may not sound like correct flux units - with "steradian" in there it sounds like intensity. However that last item, "steradian's worth of area" is indeed an area, not a solid angle. So to be formally correct, replace "steradian's worth of area" with "d^2 cm^2", where d is the assumed binary - observer distance in cm. Pictorially, imagine the observer's detecting instrument covering 1 steradian (a little on the big side, but one can always re-scale to more practical units). To keep things simple, imagine that the star radiates isotropically and that we enter a luminosity of 4π, in our chosen unit, for one of the stars. The program will produce an output flux of 1.000 for that star for all phases and inclinations (outside eclipse), and since that flux refers to 1 steradian, the 4π steradians surrounding the star would have 4π units, which is just what we entered as the luminosity. If we specified the luminosity unit to be, say, $1\,10^{33}$ erg/sec/micron, then the luminosity is $4\pi\,10^{33}$ erg/sec/micron and the flux (light) is $1\,10^{33}$ erg/sec/micron/$d^2 cm^2$ (constant in this special case).

X1A, X2A (x_1, x_2): These are the wavelength-specific limb darkening coefficients in the linear terms. The laws have the same forms as for bolometric limb darkening (above). There is one value for each of these coefficients for each light or velocity curve or set of line profiles.

EL3A (l_3): Third light. There is one value for each light curve, but of course no value for a radial velocity curve. The unit should be the total system light at a specified phase. For example, suppose l_3 (program input-output value) for some particular light curve is 0.0500, the specified phase is chosen to be 0.2500, and the total system light produced by **LC** at phase 0.2500 is 1.0400. Then the number to be published for the l_3 of that curve would be 0.0500 divided by 1.0400, or 0.0481. The standard deviation printed by **DC** for l_3 also should be divided by 1.0400 for publication. A footnote can be placed in the publication to tell the reference phase.

8. Parameter Order

The 35 **DC** parameter channels are assigned as follows:
(1) - Spot 1 latitude
(2) - Spot 1 longitude
(3) - Spot 1 angular radius
(4) - Spot 1 temperature factor
(5) - Spot 2 latitude
(6) - Spot 2 longitude
(7) - Spot 2 angular radius
(8) - Spot 2 temperature factor
(9) - Orbital semi-major axis, $a = a_1 + a_2$
(10) - Orbital eccentricity, e
(11) - Initial argument of periastron, ω_0
(12) - Rotation parameter for star 1, F_1
(13) - Rotation parameter for star 2, F_2
(14) - Phase shift = phase of primary conjunction (for $\omega = \pi/2$), ϕ_0
(15) - System center of mass radial velocity (systemic velocity), V_γ
(16) - Orbital inclination, i
(17) - Gravity law exponent for star 1, g_1
(18) - Gravity law exponent for star 2, g_2
(19) - Mean surface temperature of star 1, T_1
(20) - Mean surface temperature of star 2, T_2
(21) - Bolometric albedo of star 1, A_1
(22) - Bolometric albedo of star 2, A_2
(23) - Surface "potential" of star 1, Ω_1
(24) - Surface "potential" of star 2, Ω_2
(25) - Mass ratio, $q = m_2/m_1$
(26) - Reference time in ephemeris (initial epoch, usually Heliocentric JD), t_0
(27) - Orbital period at initial epoch, P_0
(28) - First time derivative of the orbital period, dP/dt
(29) - First time derivative of the argument of periastron, $d\omega/dt$
(30) - Unused channel reserved for future expansion
(31) - Monochromatic luminosity of star 1, L_1
(32) - Monochromatic luminosity of star 2, L_2
(33) - Monochromatic linear limb darkening coefficient for star 1, x_1
(34) - Monochromatic linear limb darkening coefficient for star 2, x_2

(35) - Monochromatic third light, l_3

9. Control Integers, Units, Scaling Factors, Special Quantities

These numbers control program operation and are assigned according to the aims of the user. They are given here by their FORTRAN names.

9.1. Those common to LC and DC

JDPHS: This is 1 if the independent variable is time and 2 if it is phase. Setting JDPHS=1 in **LC** causes time (ordinarily Heliocentric Julian Date) to be stepped from HJDST to HJDSP in uniform intervals of length HJDIN (see below). Phases for those times are computed according to the supplied ephemeris. Setting JDPHS=2 in **LC** causes phase to be stepped from PHSTRT to PHSTOP in uniform intervals of length PHIN. Times for those phases are computed according to the supplied ephemeris. Regardless of whether JDPHS is 1 or 2, both time and phase are listed in the output (columns 1 and 2, respectively).

Setting JDPHS=1 causes **DC** to treat the entered independent variable as time and therefore compute phases from the input times according to the supplied ephemeris. Setting JDPHS=2 in **DC** causes it to treat the independent variable as phase. At present, the ephemeris includes only the initial epoch (t_0), the period at that epoch (P_0), and the first time derivative of the period, dP/dt.

MODE: This integer can be -1, 0, 1, 2, 3, 4, 5, or 6 according to the constraints or lack of constraints to be applied. The operation modes are described in Wilson (1993) and in Section 3.

IPB: Assign IPB = 0 for normal MODE 1, 2, 3, 4, 5, or 6 operation in which star 2's luminosity (L_2) is to be computed from temperatures T_1 and T_2, the luminosity of star 1, and the radiation laws (as well as other information known by the program about system geometry, etc.). If you want to set L_2 independently (perhaps because you have no trust in the radiation laws in a particular situation), set IPB = 1 and the program will use the input L_2 value. Modes 0 and -1 always accept the input L_2, so they operate as if IPB = 1. See Van Hamme and Wilson (1986) for ideas on the use of IPB in practical situations.

IFAT1, IFAT2: These control whether a black body or an approximate stellar atmosphere formulation is used for local emission on stars 1 and 2, respectively. Set IFAT1(2) = 0 or 1 for black body or atmosphere, respectively. The atmosphere corrections are calculated from approximation functions fitted to the Carbon and Gingerich (1969) model stellar atmospheres. A newer atmosphere routine is available from Milone, Stagg, and Kurucz (1992).

N1, N2: These are the grid size integers for stars 1 and 2, respectively. Each is the number of latitude rows per hemisphere. The number of surface elements in longitude scales with N1(2) and scales approximately with the sine of the "latitude" coordinate, which runs from 0 at the "North" ($+z$) pole to π radians at the "South" ($-z$) pole.

VUNIT: This is the unit for radial velocity input and output, in km/sec. Usually it is a round number, such as 100 km/sec, of the order of the input velocities for **DC**.

MREF: The reflection effect can be handled either in detail (viz. Wilson, 1990) or by the inverse square law, with corrections for penumbral and ellipsoidal effects. The latter method is much faster and is adequate for many realistic situations. Set MREF = 1 for the simple treatment and MREF = 2 for the detailed treatment. It is not advisable to use the detailed treatment for eccentric orbit cases because the required computing time will be almost prohibitively long. Cases for which the detailed treatment is especially recommended include super-synchronous rotators and overcontact binaries.

NREF: If detailed reflection is selected (MREF = 2), then NREF specifies the number of reflections in a multiple reflection effect. Set NREF = 1 for 1 reflection from each star, NREF = 2 for 2 reflections, etc. More reflections use more computing time. If MREF = 1, the value of NREF is irrelevant.

IFSMV1, IFSMV2: These integers tell whether spots on stars 1 and 2, respectively, are to move in longitude due to asynchronous rotation and orbital eccentricity. For example, if IFSMV1 is set to 0, the spots on star 1 remain at fixed longitudes, referenced to the line of centers of the two stars. This behavior is expected for hot spots due to an accretion stream. If IFSMV1 = 1 and the star rotates asynchronously, then the spots on star 1 follow the physical surface as time progresses. This behavior is expected for magnetic spots. In both cases, there is no motion in latitude.

ICOR1, ICOR2: These integers refer to the proximity and eclipse effects on radial velocities for stars 1 and 2, respectively. Value 0 turns the effects off, 1 turns them on.

LD: This integer sets the limb darkening law. LD = 1 for the linear cosine law, LD = 2 for a logarithmic law, and LD

= 3 for a square root law.

9.2. Those for LC only

HJDST: The time at which **LC** is to start computing output points. HJDST and the next two quantities, HJDSP and HJDIN, are utilized only if JDPHS=1. They are ignored if JDPHS=2.

HJDSP: The time at which LC is to stop computing output points.

HJDIN: The time increment for output points. HJDIN=0.001 will produce output points spaced by 0.001 day.

PHN: The phase of normalization, which is the phase at which the column of normalized light is normalized to the input value FACTOR and the magnitude column is caused to equal the magnitude zero point, whose name is ZERO.

PHSTRT, PHSTOP: The first and last phases at which output points are to be produced. PHSTOP should be larger than PHSTRT, but neither has to be in the range 0 to 1. For example, PHSTRT = -3.2000, PHSTOP = 27.4422 is a valid phase range.

PHIN: The phase increment for output points. PHIN = 0.020 will produce output points every 0.020 in phase, within the range PHSTRT to PHSTOP.

MPAGE: This integer is 1, 2, 3, 4 or 5 according to whether the output is a light curve, radial velocity curves, spectral line profile(s), star radii, or sky image coordinates, respectively.

ZERO: This is the reference point for output magnitudes (the magnitude at phase PHN).

FACTOR: This is the scaling factor for the normalized light column. The number in that column will equal FACTOR at phase PHN.

9.3. Those for DC only

DEL's: Most of the partial derivatives needed by **DC** must be computed numerically. The DEL's are the parameter increments applied when approximating those derivatives by finite differences. A few derivatives can be computed by scaling relations and require no DEL's. That is why the number of DEL's is smaller than the number of parameters. However the numerical labels of the DEL's match those of the parameters. The order in which the DEL's appear in the input lines can be read from the output of **DC**, where they are labeled by parameter name, or from the **DC** output in Appendix Ib, where they are labeled by parameter number. Give some thought to reasonable values for the DEL's. DEL's that are too large will cause systematic errors in the derivatives, and DEL's that are too small will cause excessive numerical noise.

KEP or KEEP: These mean the same thing generically. The difference in meaning in a programming sense is not something the user needs to be concerned with. The KEEP's determine which parameters are to be adjusted, and have only two possible values, 0 to allow adjustment and 1 to keep a fixed value. There are 35 adjustable parameters and therefore 35 KEEP's in the present program (actually 34 adjustable parameters plus 1 parameter channel reserved for future expansion, so that 35 KEEP's are entered). They are all entered together on an input line, separated into blocks so as to be easy to count off. There is one such input line for the base set, early in the data stream, and a set of n such input lines at the end of the data stream (for the n subset solutions). The order of appearance within the string of 1's and 0's is the same as that of the 35 parameters, which is listed in Section 8 and also printed with every run of **DC**.

IFDER, IFM, IFR: These are three print control integers that follow the KEEP integers on the same input lines. Each is set to 1 for "yes, print" or to 0 for "do not print". IFDER decides on the printing of the matrix of derivatives and (O-C) residuals, or observational equations (both the unweighted and the weighted matrices). Usually one would print the derivatives and residuals only for the base set solution, since the numbers are unchanged in the subset solutions (except for missing columns). IFM decides on the printing of the matrix of normal equations, the product matrix of the normal equations times their inverse (which should be an identity matrix), the matrix of correlation coefficients, and the sum of absolute values of residuals (a single number) obtained by back-substitution of the answers into the normal equations. The user may want to see these numbers for all subset solutions or perhaps only for certain solutions. IFR decides on the printing (after each subset solution) of a block of information about radii, the derivatives of radii with respect to surface potential and mass ratio, and the standard errors of the radii.

XLAMDA: The Marquardt λ multiplier (see next section, subsection "Marquardt λ"). XLAMDA is the final entry on the "KEEP" lines and is the only floating point number on those lines.

KSPA, NSPA, KSPB, NSPB: Each of the two stars may have many spots, but **DC** can adjust the parameters of at most two spots in any one run. This does not imply a limit to the number of spots adjusted overall, since other spots can be adjusted in other runs. Think in terms of spot A and spot B, which may be on the same or different stars. KSPA, NSPA, KSPB, and NSPB are best understood as a set. Given that some spot parameters are to be adjusted (which is determined by KEEP's 1 to 4 for spot A and KEEP's 5 to 8 for spot B), the new integers (KSPA, etc.) determine for which of the (perhaps) many spots on the two stars those parameters will be adjusted. KSPA=1 means that spot A is on star 1, and KSPA=2 means that it is on star 2. NSPA determines which spot on the thus determined star is spot A. If KSPA=2 and NSPA=3, then star 2's third spot is spot A. Star 2 might have some large number of of spots, say 10, and spot A will be the third one. Of course, the program will do something crazy if you ask it to adjust the third spot on a star that has only two spots - I have not bothered to find out what. The order assigned to the spots is just the order in which their parameters are read in the input lines. Naturally, KSPB and NSPB apply the same way in identifying spot B. If you want to adjust, say four spots, you can use the MMS (see next section) and adjust two of them in one **DC** iteration, the other two in the next iteration, and so on.

IFVC1, IFVC2: These tell **DC** whether or not to expect a radial velocity curve in the input stream for star 1 and star 2 respectively (0 for no, 1 for yes). Velocity curves precede light curves in the input. So with IFVC1=0 and IFVC2=1, **DC** expects the first curve it sees to be a velocity curve of star 2.

NLC: This integer is the number of light curves of the binary in the input stream. With IFVC1=1, IFVC2=1, and NLC=3, **DC** expects to encounter two velocity curves (the first for star 1, the second for star 2) and then 3 light curves. It will do a simultaneous adjustment (one iteration) of these five curves together, producing one correction for each parameter that is the same for all curves (*e.g.* mass ratio) and 3 corrections for each parameter that is different for each of the 3 light curves (*e.g.* limb darkening). The 3 curve-dependent parameter corrections for a given stellar component are printed in direct succession. Input and output parameters, their differences (corrections from input values) and their estimated standard deviations are listed by parameter number (see list in Section 8). Curve-dependent parameters are listed also by curve number, according to the input order of the curves. "Curve 0" means "curve-independent".

KO: **DC** can process multiple parameter subsets in a given run, as explained under the method of multiple subsets (MMS). Control integer KO provides three options to the user who has established a "scratch pad" data file on the local computer. Such a scratch file should be designated as input-output unit 9. The purpose of this provision is to save computing time by writing all of the numerical derivatives and residuals on the scratch file so that they can be read back for further subset solutions in a later submission of the program (in case you think of some subsets that you first thought were not needed). Different weighting also can be used in the re-submission. The later run(s) then will take extremely little computing time, since all the hard computation has been done. This feature is much less important than it once was because of improvements in machine speed and intrinsic program speed, but it is still available. Multiple subsets can be processed within a given run without using the scratch pad (set KO=2, KDISK=0). KO can be 0, 1, or 2, as follows:

KO=0. Do the base set and stop. Actually KO=0 serves no significant purpose, as simple omission of subsets has the same result.

KO=1: Read in the derivatives and residuals that were written to the scratch pad in a previous run. Process these according to the weighting schemes and subsets requested in this run. Do not re-compute any derivatives. A run with this option should take only a few seconds or less of computing time.

KO=2: Write the derivatives and residuals generated in this run on the scratch pad for future use. KO=2 can be used with KDISK=0 if the user does not wish to change the present contents of the scratch pad (or if there is no scratch pad). In this case the program still can process multiple subsets of the parameter list but, of course, cannot come back later to process the data further.

KDISK: Set KDISK=1 to use the scratch pad, or KDISK=0 not to use it. Caution: the program will crash if no scratch pad has been set up and you set KDISK=1.

ISYM: **DC** can carry out its solutions with either asymmetrical (ISYM=0) or symmetrical (ISYM=1) derivatives. Solution convergence is somewhat better with symmetrical derivatives, but they take almost twice as long to compute as asymmetrical derivatives. Because of the improved convergence, fewer iterations may be needed with ISYM=1, but the individual iterations will take more machine time. Recommendations are to use ISYM=0 for strong (well determined) solution situations, and ISYM=1 when convergence is a problem or you are in the final fine-tuning stages. Setting ISYM=1 is particularly advisable when large increments (DEL's) are used to compute the numerical derivatives. See Wilson and Biermann (1976) or Wilson (1979, eqns. 20, 21).

NPPL: (number of points per line) The number of data triplets (time or phase; velocity or light; weight) on each line of the input data stream. The number of points per output line is the same. NPPL can be as small as 1 or as large as 5.

N1L, N2L: These are the coarse (low) grid integers. As explained under "Special Features", they apply to certain derivatives for which computing time can be saved by using coarse grids. Otherwise they are just like N1 and N2.

NOISE: This integer is on the same input lines as the curve-dependent parameters for light curves, so there is one value for each light curve (no values for velocity curves). Noise specifies how level-dependent weights are to be applied (see discussion of weighting in Section "Input Lines, Including Observations and Weights")

SIGMA: The estimated standard deviations of the various observed light and velocity curves. SIGMA specifies the relative weights of the curves in the solution (weight is inversely proportional to SIGMA squared). See discussion of weighting in Section "Input Lines, Including Observations and Weights").

10. Problems with Solution Convergence

There are three main reasons for poor convergence or convergence failure in differential corrections solutions, and it is important to understand all three. The problems can occur in combination, but let us try to understand them separately.

One problem is a broad, shallow minimum in parameter space. This problem can result in slow convergence even when only a few parameters are being adjusted, if the minimum is very shallow indeed. It is a problem not only for **DC**, but for any minimization scheme, since it is intrinsic to parameter space itself. Nothing can be done except to compute with precision (use a fine grid) and do many iterations.

A second problem, also easy to understand, is lack of precision in computing residuals and derivatives. Since most derivatives are computed numerically in **DC**, we have two obvious sources of error connected with finite differencing, which approximates the true derivative of a synthesized observable, f, by the slope near to a given operating point, either by

$$\frac{\partial f}{\partial p_n} = \frac{f\left(p_n + \delta p_n\right) - f\left(p_n\right)}{\delta p_n}$$

or by

$$\frac{\partial f}{\partial p_n} = \frac{f\left(p_n + \frac{1}{2}\delta p_n\right) - f\left(p_n - \frac{1}{2}\delta p_n\right)}{\delta p_n}.$$

Here, p_n is a parameter and δp_n is its increment. Pseudo-random errors arise from lack of accuracy in the computed f while systematic errors come from approximation of $\partial f/\partial p_n$ by $\delta f/\delta p_n$. A symmetrical derivative is a better representative of the true derivative, but takes longer to compute. Of course, the systematic error can be reduced by taking smaller δp_n, but the increment should not be too small or errors in computation of f will become important. Inaccuracy in computing f can be reduced by use of a finer numerical grid, but at the cost of increased machine time.

The third problem is more subtle - particularly because it is due to a combination of *two* conditions, neither of which causes convergence failure when acting alone. It is important to realize that this third problem is quite distinct from the other two, and can destroy convergence even if the other problems are completely absent. The two interacting conditions that cause the problem are *non-linearity in the appropriate equation of condition* and *parameter correlation.* The first of these conditions can be expressed equivalently by saying that second or even higher order derivatives are required in the differential corrections equation of condition (which normally is written with only first derivatives). One way of seeing into the nature of this problem is to notice that a correct solution algorithm needs to know about parameter correlations, but a linear least squares algorithm cannot compute them correctly because it has no knowledge of the higher order derivatives. As a result there will be virtual tradeoffs among the invisible higher order terms that result in incorrect predictions of parameter corrections. It usually is not practical to add second order terms to the equation of condition for several reasons, so two procedures that improve convergence greatly while keeping only linear terms will now be discussed.

10.1. Method of Multiple Subsets (MMS)

One realistic remedy for the problem is to break the full parameter set into subsets and thus reduce the complexity of the correlations in a given iteration (Wilson and Biermann, 1976). This procedure, in which subsets A and B, or A, B, and C, etc. are solved iteratively in a closed loop has come to be known as the Method of Multiple Subsets (MMS), and it works very well in practice. It seems that a **DC** solution has only minor difficulty in dealing with one or two fairly strong correlations among parameters. The real difficulty comes when there are many large or even moderately large correlations, and this circumstance accounts for the success of the MMS. Two important points need to be emphasized. First, the published standard errors should come from a final run that includes all adjusted parameters together - not from the the subset solutions (which will give unrealistically low error estimates because they see only part of the correlation matrix). It must be said that even the standard errors for the full set will be of questionable accuracy when there is a complicated correlation matrix, but they will have to do in the absence of more rigorously computed standard errors. The second point is that one must resist the temptation to apply the corrections printed in this final run (which is for error estimates only). It is the nature of the problem that, when the MMS is needed, it is needed all the way to the end. Even if you start from the exactly correct least squares minimum, the correlation problem discussed here will produce relatively large, erroneous corrections for the full parameter set. This has nothing to do with the accuracy of the derivatives computed by **DC**, but only with the form of parameter space. Some persons have misunderstood this point and assumed that correlation problems occur only away from the correct solution and that they will disappear at the correct solution. Not only do the problems not then disappear, but they do not even lessen in severity.

10.2. Marquardt λ

A very good trick for effecting convergence is the Marquardt (1963; see also Levenberg, 1944) procedure, which has been utilized with excellent success in several binary star solution programs such as those by Hill and Rucinski; by Kallrath, Milone, and Terrell; and by Djurasevic. The Marquardt algorithm operates on the matrix of normal equations so as to effect a compromise between the ordinary **DC** corrections (which usually give fast convergence but sometimes lead to convergence failure) and those of the method of steepest descent (convergent but sometimes slow). The normal equations are re-scaled (so as to have diagonal elements of unity) in the **DC** program supplied here, as prescribed by Marquardt, although it would be possible to apply the procedure without re-scaling. At the heart of the scheme is a quantity λ that is added to each diagonal element of the re-scaled normal equations. A decision faced by all those who adopt the scheme is that of how to fix the value of λ, and various kinds of iterations have been used in the literature to optimize λ, including one in the original Marquardt (1963) reference. The supplied **DC** program operates on a somewhat different idea, following experience that - at least in binary star problems - λ values over a broad range typically give nearly the same corrections. Usually such λ values are very small, and one finds results for say $\lambda = 10^{-4}, 10^{-5}$,or 10^{-6} that differ by so little that the choice is inconsequential. Because the **DC** program already has been set up to make solutions for many subsets of the main parameter set, it is easy to do solutions for many λ values, so that no iteration on λ is needed. One simply includes a line of data for each such solution. For example, if not sure of the λ value to use, just do solutions (same **DC** submission) for as many values as deemed potentially interesting, all for the same parameter set or subset. The lines will look like this:

```
1111 1111 0111100 01110 11010 11111 01110 0 1 0 1.000d-6
1111 1111 0111100 01110 11010 11111 01110 0 1 0 1.000d-5
1111 1111 0111100 01110 11010 11111 01110 0 1 0 1.000d-4
```

where the integer 1's and 0's are the KEEP's and print control integers for a subset solution, while the final 1.000d-n is λ for that solution. The execution time to do the several solutions (even very many) is negligible. Usually it makes little difference which λ value is adopted within such a range, although 10^{-11}or 10^{0} may give much different answers.

10.3. MMS or Marquardt?

Either the MMS or Marquardt algorithm will nicely solve most convergence problems of the correlation - non-linearity type. The Marquardt algorithm is the more convenient because it does not extend the level of iteration. However Marquardt alone cannot handle the most severe convergence problems, where a combination of Marquardt and MMS may be needed.

10.4. Error Distributions and Standard Error Estimates

Estimated standard deviations of parameters are printed along with the parameter corrections for the base set and subset solutions. Published comments occasionally appear to the effect that the **DC** program produces unrealistic error estimates, although they are computed from the covariance matrix by the standard method. For example, a 1997 paper comments about **DC**: " . . . *the estimates of errors of the adjustable parameters are unrealistically small. The reason is partly the strong correlation between the relatively many parameters, and partly the non-normal distribution of measurement errors*". Such remarks have some formal validity, but are misleading for the following reasons:

"*Strong correlation between the relatively many parameters*". In significantly non-linear situations the standard deviations will be only approximately right, but that is also the case with all other binary star fitting programs that now exist. The shortcoming is not peculiar to **DC**. However an impression that the problem *is* peculiar to **DC** obviously has arisen, and one can speculate as to why that is so. An obvious point is that some solution programs avoid such complaints by not computing error estimates at all, or by doing so with invented rules that produce comfortably large estimates. However some other solution programs do compute standard error estimates, *but usually not for adjusted mass ratios.* Experience shows that mass ratios cause most of the problems with fitting and with error computation. This is so because mass ratio often has major correlations with other parameters, and it also shows particularly non-linear behavior. Programs that cannot adjust the mass ratio parameter (most programs) will not encounter the problem.

"*The non-normal distribution of measurement errors*": The cited phrase probably does not literally refer to *measurement errors*, whose distributions for light curves are as close to Gaussian as one is likely to find for any astrophysical measurements. (Observers measure light with respect to nearby comparison stars whose brightness and color are close to that of the variable star. They typically expend considerable thought and effort to eradicate possible systematic errors). The phrase more likely refers to *non-normal distribution of residuals*, which one finds when a real binary does not match the model very well (perhaps it has un-recognized spots or circumstellar gas, etc.) or if the solution is at an incorrect local minimum. If interpreted in this way, the statement is quite correct, although trivially obvious. Under any interpretation, the comment has nothing to say about **DC**.

D. Terrell has carried out experiments by solving synthetic light curves by means of **DC**. The light curves contained simulated Gaussian errors and, of course, their parameters were known. He found that the parameter errors [solution - known] agreed with those expected from the standard errors in output from **DC**.

11. Interactive Branching and the Lack of Automatic Iteration in DC

Many persons have been surprised at having to re-submit the program at each DC iteration. The lack of automatic iteration is not an oversight, but a way to force users to look at the progress of a solution. Even without automatic iteration, it is not unusual for users to believe whatever comes out of the machine, even when the results violate common sense. In this highly non-linear problem, the solution process can indeed get into trouble and produce inappropriate corrections. To some extent it is possible to have a program detect inconsistent and strange results. There are a few such kinds of detection and message generation, and further messages about such problems may be worked into future versions. However until such screening has been made very extensive, the user is encouraged to think about every iteration.

There is another important reason to avoid automatic iteration. In real situations, an experienced user will decide which parameters to adjust and which to hold fixed as part of a "dynamical" process of interaction with the iterative solution. Occasionally the set of adjusted parameters will remain unchanged from start to finish, but usually it will not. Because of this reality, **DC** is set up for interactive branching, by which each iteration produces solutions of as many subsets of the main (base) set as are requested (control integer KO must be set to 2). Notice that the sample input data supplied with the program requests several subset solutions via extra lines of KEEP's at the end of the data block. There is no limit to the number of such subset solutions - you can ask for a hundred or more if you like - and they require almost no machine time (since solving the observational equations takes almost no time compared to generating them). The only penalty for requesting a large number of subset solutions is the proliferation of output, with possible attendant confusion. So the idea is to decide, after each iteration, which set of adjustable parameters to follow from that point on (to the next such change). Interactive branching is discussed in Wilson (1988).

However the most cogent reason for personal monitoring of **DC** iterations is simply this: *There is more to astrophysics than parameter estimation.* Recognition of strange and unusual behavior points the way to the really major insights and discoveries. Many close binaries are not well represented by even our best generic models, and in some cases the problem is not due just to approximate computation but to quite unsuspected features. To drive this point home, imagine applying one's favorite "standard" light curve model automatically to one of the classic strange objects - perhaps one with an unusual kind of circumstellar disk. Persons familiar with β Lyrae might amuse themselves by imagining the outcome for that binary. One will get answers (meaningless ones, naturally), but will have at best only a much reduced chance for insights into the real problems. Often one really needs to get in there and watch things develop.

12. Special Features

Coarse and fine grids: In order to save computing time, **DC** uses both coarse and fine computing grids, and applies the fine grids only where accuracy requires them. The fine grids are for the residuals ([O-C]'s) and for the derivatives with respect to parameters e, ω, F_1, F_2, ϕ_0, i, Ω_1, Ω_2, q, spot latitude, spot longitude, and spot radius. Fine grids are required for parameters that are in some way geometrical. The derivatives of the non-geometrical parameters g_1, g_2, T_1,

T_2, A_1, A_2, L_1, L_2, x_1, x_2, and spot temperature are computed with the coarse grids. The fine grids are specified by [N_1, N_2] and the coarse grids by [N_{1L}, N_{2L}] for stars 1 and 2 respectively.

Synthetic Noise: Light curves with synthetic Gaussian noise can be produced by entering a non-zero value for the input quantity STDEV [FORTRAN name], which is labeled "fract. sd." in the output. As the label suggests, it is in the unit of the light at the reference phase (PHN). The way that the noise (scatter) scales with light level is controlled by input integer NOISE. Scatter is proportional to light level for NOISE = 2, proportional to the square root of the light level for NOISE = 1, and independent of light level for NOISE = 0. The random number generator needs a seed [FORTRAN name SEED], labeled "seed" in the output. SEED should be larger than 100000001. and smaller than twice that value. At present there is no provision for synthetic noise in radial velocity curves or line profiles. Users who have a favorite random number generator should find it easy to replace the program generator with their own.

13. Simultaneous Solutions

A capability of **DC** that should be used in most circumstances is that of making simultaneous multi-bandpass light curve solutions as well as simultaneous (one or two curve) radial velocity and multi-bandpass light curve solutions (Wilson, 1979). The advantages are to avoid inconsistencies among solutions of the separate curves, to reduce the number of free parameters (no need to have both photometric and spectroscopic mass ratios, nor to have a separate inclination for each light curve), and to utilize information that is discarded in separate solutions (the knowledge that there is only one true mass ratio, one true eccentricity, etc.). Weighting is very important in simultaneous solutions and is discussed in Wilson (1979) and in the next section. The sample data supplied with the program are for a simultaneous solution of two radial velocity and three light curves. There can be 0, 1, or 2 velocity curves and any number of light curves (the program is dimensioned for up to 25 light curves, which seems ample for realistic situations). Input integers IFVC1 and IFVC2 tell the program how many velocity curves to expect, and NLC tells it how many light curves to expect. The **DC** solution will produce one correction for each adjusted curve-independent parameter (most of them) and n corrections for each adjusted curve-dependent parameter (L_1, L_2, x_1, x_2, l_3). It takes no more (actually somewhat less) machine time for a simultaneous solution of n curves than for n individual curve solutions, so there really is no reason not to take advantage of this feature. If one has misgivings about the simultaneous solution, separate solutions always can be carried out in addition. One particularly illogical practice in a few papers has been to publish averaged parameters from separate solutions and to offer the averages as a substitute for a simultaneous solution. However, the average of separate solutions will not be a correct solution of *any* of the separate curves. Ask "what is the proper way to take such an average?" The answer is that there is no self-consistent way - the only way to have the results of a simultaneous solution is to *do* a simultaneous solution in the first place.

14. Input Lines, Including Observations and Weights

Getting started should only be a matter of running the programs with the supplied sample input data and changing the numbers to those of particular binaries. Identifying the various input quantities can be done in several ways, for example via Appendices *Ia* or *Ib*, by examining the output where the numbers are printed with labels, or by comparing the input data lines with the FORTRAN "READ" statements.

For **LC**, there is a "more or less standard" set of input lines with control integers and parameters for each curve that is to be computed, and n such sets of lines can be concatenated to produce n output curves. An output curve can be a light curve, a radial velocity curve, or a spectral region containing mixed absorption and emission spectral lines (perhaps blended). The "standard set" of input data lines is only "more or less standard" because there can be extra lines to tell about circumstellar attenuating clouds and/or about bright or dark star spots and/or about spectral lines. For light or velocity curves, the number of input lines for a given curve is $(10 + c + s)$, where c is the number of circumstellar clouds and s is the number of star spots for the two stars combined. One of the 10 mandatory lines is the *stop* line for clouds and two others are stop lines for star spots, one to follow the lines of spot parameters for each star. The cloud stop line should contain a number greater than 100. but less than 200. in the first field, which corresponds to a cloud "x" coordinate. Each of the two stop lines for spots should contain a number greater than 200. in the first field, which corresponds to a spot latitude on the normal spot parameter lines. According to the value of MPAGE, **LC** computes a light curve (MPAGE=1), a radial velocity curve (MPAGE=2), a set of spectral regions (MPAGE=3), star dimensions (MPAGE=4), or plane of sky coordinates for images (MPAGE=5). Spectral line profile computation requires extra input lines to enter the line characteristics. Therefore the sample data includes an input file for line profiles (called *lcin.dat3*).

The sample files for light curves and velocity curves are called *lcin.dat1* and *lcin.dat2*, respectively. For dimensions and images they are *lcin.dat4* and *dat5* respectively. The several sets of input lines need not be of the same binary, since each curve computation is an independent operation. A final line stops execution and should contain integer 9 in the field normally occupied by MPAGE (that final line is not counted in the 10+c+s lines of the individual curves).

For **DC**, three lines of DEL's enter first. The next line pertains to the base parameter set and contains 35 KEEP's for the 34 adjustable parameters and one blank channel, 3 print control integers (IFDER, IFM, and IFR), and the Marquardt λ. Next comes a line of spot identification integers (KSPA, NSPA, KSPB, NSPB), then five lines of control integers and curve-independent parameters, then as many lines of curve-dependent parameters as there are observed curves, then as many lines of spot parameters as there are spots, then as many lines of circumstellar cloud parameters as there are clouds, then the observed radial velocities, then the observed light curves, and finally as many lines of KEEP's for subset solutions as the user wants. These final KEEP lines are of the same format as the KEEP line for the base set and also contain the 3 print control integers and the Marquardt λ. Note that the solutions that are triggered by these KEEP, *etc.* lines can be for varied λ as well as for selected parameter subsets. For example, one can run two solutions for exactly the same parameters but with different λ.

The observed radial velocity curves and light curves of the **DC** input data stream are entered as triplets (phase; velocity or light; weight), with up to 5 triplets (*i.e.* data points) per line. The number of data points per line is set by input integer NPPL (*number of points per line*). A possible formatting problem, caused by velocities not being of the same order of magnitude as light measures, is solved by providing for the separate entry of a convenient sized velocity unit. The (0, 1, or 2) velocity curves and NLC light curves are separated by "data stop lines" that serve two purposes. They identify the last data line for each velocity or light curve and also tell how many data points are on that last line (1, 2, 3, 4, or 5). The only number on a data stop line should be in the first field and should be -(10000+k), where k is the number of data points on the preceding line. Example: if there are 2 data points on the preceding line, the "stop" number should be -10002. Some computing systems require "something" in the remaining fields of the line and blanks are suitable, but there must be at least blanks (not an absence of all characters). No special stop information is to follow the final curve because the program already knows how many curves to expect, and it knows which is a velocity curve and which is a light curve because it has already read IFVC1, IFVC2, and NLC. However a line with integer 2 in column 2 should follow the final line of subset KEEP's to signal the end of subset processing and of the entire job.

Weighting of observations is discussed in Wilson (1979, pp. 1064, 1065), Wilson (1988) and Kallrath and Milone (1999). Briefly, the program applies three kinds of weights, which are "intrinsic" weights (assigned by the user to the individual observations), "curve-dependent" weights (based on estimated standard deviations [SIGMA's] at a reference phase), and "level-dependent" weights (computed by **DC** according to the input parameter NOISE, which tells how observational scatter scales with light level. NOISE should be set to 1 for scatter that scales with the square root of the light level, such as pulse counting statistics, and to 2 for scatter that scales with the light level, such as scintillation noise or fluctuations in sky transparency. If NOISE is set to 0, no level-dependent weighting is applied. Level-dependent weighting does not apply to velocity curves. SIGMA's pertain to the directly entered velocity and light values. An exception is where the SIGMA's were actually measured from individual observations, but normal points (i.e. averages) are entered. In that case, the number of individual points in each normal point should be incorporated in the "intrinsic" weight of each point. Some persons have contrived their curve-dependent weighting (through the SIGMA's) so as to increase the influence of one or a subset of the curves according to pre-conceptions - for example, of the relative importance or radial velocity and light curves. However the SIGMA's should properly be based on measurements, not prejudice.

15. The Scaling of Run Time

The main computational activity of **LC** consists of summations over surface grid points in forming the observable fluxes and velocities of the two stars. The number of grid points on each star is essentially proportional to the square of the grid fineness integer for that star (number of latitude rows per hemisphere), so run time very nearly scales with $N_1^2 + N_2^2$. Computations of all other kinds take negligible time. Of course, run time in **LC** also scales with the number of phase points, or more precisely with that number plus 1 (because one extra point is done at the phase of normalization). Eccentric orbits take longer than circular orbits because the local physical computations must be done anew at each phase. Note that a very small eccentricity (say 0.000001) differs in this regard from one of exactly zero, because the program skips the extra computations only if $e = 0$. The actual time factor depends on whether the old approximate (MREF=1) or new detailed (MREF=2) reflection model is specified. The MREF=2 case takes longer for both circular and eccentric orbits, but it takes *much* longer (usually impractically longer) for eccentric orbits. If MREF=2, then run time also depends on

NREF, the number of multiple reflections. There will also be some dependence on the particular binary star configuration. For example, runs for stars with small $r = R/a$ go faster than those with large r. The situation is sufficiently complicated so that actual speed factors are best estimated via a few experiments at low grid fineness.

For **DC**, many of the same considerations apply as for **LC**. However there are four grid fineness integers because **DC** uses both a high and a low grid for each star (viz. Section 12), so run time scales with $(P+1)\left(N_1^2+N_2^2\right)+P_L\left(N_1^2+N_2^2\right)_L$, where P is the number of fine grid parameters and P_L is the number of low (coarse) grid parameters under adjustment in the main (base) set of parameters. The first term involves $P+1$ rather than just P because not only derivatives but also residuals ($[O-C]'s$) must be computed, and the residuals are done with the fine grid. Run time in **DC** also scales with the number of observations that are processed. Do not add 1 because there is no phase of normalization. The numbers of parameters in subset solutions are not relevant to run time because the run time for the subsets is negligible. **DC** iterations with ISYM=1 (i.e. symmetrical derivatives) will take nearly twice as long as those with ISYM=0 (i.e. asymmetrical derivatives).

16. Common Difficulties

1. **Minimum dimensioning**: One of the most frequent reasons for failure in program execution is under-dimensioning of arrays. These failures are disconcerting because very strange things can happen and usually no logical interpretation of the machine error messages is apparent. The dimensioning of **LC** and **DC**, as supplied, may not be sufficient for all cases. Array dimensions need be changed only in the main programs (**LC** and **DC**), not in the subroutines. See the "Dimensioning vs. Grid Fineness Table" at the end of the booklet for minimum dimensions of the arrays RV, GRX, GRY, GRZ, RVQ, GRXQ, GRYQ, GRZQ, SLUMP1, SLUMP2, SRV, SGRX, SGRY, SGRZ, SRVQ, SGRXQ, SGRYQ, SGRZQ, SRVL, SGRXL, SGRYL, SGRZL, SRVQL, SGRXQL, SGRYQL, SGRZQL, SLMP1, SLMP2, SLMP1L, SLMP2L, FR1, FR2, GLUMP1, GLUMP2, GRV1, GRV2, XX1, XX2, YY1, YY2, ZZ1, ZZ2, GMAG1, GMAG2, CSBT1, CSBT2, RF1, RF2, RFTEMP, SXX1, SXX2, SYY1, SYY2, SZZ1, SZZ2, SGMG1, SGMG2, SGRV1, SGRV2, SGLM1, SGLM2, SCSB1, SCSB2, SRF1, SRF2, SGLM1L, SGLM2L, SGRV1L, SGRV2L, SXX1L, SXX2L, SYY1L, SYY2L, SZZ1L, SZZ2L, SGMG1L, SGMG2L, SCSB1L, SCSB2L, SRF1L, SRF2L, ERV, EGRX, EGRY, EGRZ, ELMP1, EGLM1, EGRV1, EXX1, EYY1, EZZ1, EGMG1, ECSB1, ERF1, ERVQ, EGRXQ, EGRYQ, EGRZQ, ELMP2, EGLM2, EGRV2, EXX2, EYY2, EZZ2, EGMG2, ECSB2, ERF2, ERVL, EGRXL, EGRYL, EGRZL, ELMP1L, EGLM1L, EGRV1L, EXX1L, EYY1L, EZZ1L, EGMG1L, ECSB1L, ERF1L, ERVQL, EGRXQL, EGRYQL, EGRZQL, ELMP2L, EGLM2L, EGRV2L, EXX2L, EYY2L, EZZ2L, EGMG2L, ECSB2L, ERF2L, SFR1, SFR!L, ERF1, ERF1L, SFR2, SFR2L, EFR2, and EFR2L. The minimum dimension of each of these arrays depends on the grid fineness. For example, if the grid fineness integer (N1, N2, N1L, or N2L) is 30, the minimum dimension is 762. The arrays STLDH, STLDL, ETLDH, and ETLDL are dimensioned to the sum of the minimum dimensions for both stars (in above long list of arrays). For N's of 30, this would then be 762+762=1524. The arrays PHAS, FLUX, and WT are dimensioned to include all the observations in all bandpasses *plus* the blanks on the last data lines *plus* the blank observations on the data stop lines (see **DC** sample input file). Arrays OBS and HOLD are dimensioned to [(number of observations) X (number of parameters in least squares solution +1)]. A curve-dependent parameter counts n times for n light curves. Thus if you had 98 observations in star 1's velocity curve, 102 in star 2's velocity curve, 470 in one light curve, and 530 in another, the first factor would be 1200 = 98 + 102 + 470 + 530. Then if you adjust i, g_2, L_2, and x_1, the second factor is 7 = 2 + 2 X 2 +1 (remember, L_2 and x_1 are curve dependent and count twice each, since there are two light curves. The last +1 is for the (O-C) residuals. OBS and HOLD need then be dimensioned to a minimum of 1200 X 7 = 8400.

2. The parameter increments must be neither too large (gives systematic errors) nor too small (gives numerical noise). These increments are called the DEL's (FORTRAN name). For most parameters, DEL's of about 1 % of the parameter value are appropriate. However particular circumstances sometimes affect that guideline, so common sense and experience are the best guides. Finer grids allow smaller DEL's. Use of the ISYM = 1 option allows larger DEL's before curvature effects become important.

3. It is best to have the initial parameter guesses for differential corrections based on experiments with the light - velocity program. Column 5 in the main block of output from **LC**, which is $l_1 + l_2 + l_3$, should approximately match the observed light values.

4. The distinction between direct light (column 5 of **LC** output) and normalized light (column 6) can be a source of confusion. Remember that normalized light is only intended for convenience in initial graphical trials and that it has no counterpart in **DC**, which deals with direct light only. If this continues to confuse you, forget that normalized light exists and work always with direct light.

5. Always check Ω_1 and Ω_2 to be sure they are in the permitted range for given q, F, and e. In differential corrections,

be sure that the incremented values of Ω_1 and Ω_2 are within allowed ranges. For overcontact systems, Ω should be between the critical values for inner and outer contact (see table of critical Ω's). For detached stars, Ω is greater than the critical Ω for inner contact.

6. Certain parameters cannot be adjusted in certain program modes. The reason is either that the parameters are not free, but functionally determined from the mode logic, or that they have no effect on the computed light or velocity. Attempts to adjust those parameters are the most common cause of blowups in subroutines **SQUAR** and **DMINV**. L_2 cannot be adjusted in any mode greater than 0 unless IPB has been set to 1. In mode -1, parameters g_1, T_1, T_2, A_1, Ω_1, Ω_2, and x_1 cannot be adjusted. In mode 3, parameters g_2, A_2, Ω_2, L_2, and x_2 cannot be adjusted. In mode 4, parameters Ω_1 and L_2 cannot be adjusted. In mode 5, parameters Ω_2 and L_2 cannot be adjusted. In mode 6, parameters Ω_1, Ω_2, and L_2 cannot be adjusted. As a practical matter, one should try to adjust both temperatures only under unusual circumstances.

7. Luminosities (input L_1 and L_2) are approximately 4π times larger than computed light values (output l's). Therefore to obtain $l_1 + l_2$ of about unity outside eclipse, enter input luminosities that add to about 4π. Note that the traditional treatment of third light as if it were third luminosity is incorrect (see section with discussion of luminosity vs. light).

8. Critical Ω's depend on e. Although a table of lobe filling potentials for synchronous and non-synchronous rotation is provided with this document (on diskette), such tables for eccentricity and rotation combined would be too extensive for practicality. Therefore to find critical eccentric Ω_1 and Ω_2, run **LC** in mode 6. The program will replace your input Ω's for both stars with the critical Ω's in the output listing.

9. Overcontact binaries, such as W UMa stars, require finer grids than do detached and semi-detached binaries. This is because the numerics of the neck region are particularly difficult to treat accurately. N's about 50 % larger than normal are recommended for overcontact systems.

10. The programs can apply the detailed reflection model of Wilson (1990) for eccentric as well as circular orbits, but eccentric cases use an enormous amount of machine time. If you are doing an eccentric binary and the programs run almost forever, check to see if you set MREF = 2. In most realistic situations, the old approximate reflection (MREF = 1) should be entirely adequate for eccentric binaries.

11. A very common cause of failed runs is simply inadvertant shifting of numbers on the input lines out of their proper fields, so that leading digits or signs are clipped away, or the numbers are not read at all, or they are read under the wrong name. Be sure to keep a copy of the sample data in exactly its original form, for later comparison.

12. Sometimes users make changes that cause the programs not to work. Perhaps the program will work for the immediate application, but will fail in another situation. Be sure to keep a copy of the entire program in exactly the form supplied, so that you can see whether it runs correctly in the circumstances under question.

13. Be sure not to mix subroutines from different versions of the program.

17. Differences from Pre-1992 Versions

1. Star spot parameters can now be adjusted.

2. Optional non-linear limb darkening laws (pre-1992 versions had only the linear cosine law).

3. The reflection effect now can be computed either with the detailed model of Wilson (1990) or with the approximate reflection model of the old program (which is faster). Multiple reflection is included in the detailed model.

4. The present **LC** and **DC** programs are much faster than pre-1992 versions.

5. The specification of phase range in **LC** was made more convenient in 1992, with allowance for phases outside the range 0 to 1.

8. The stars now orbit counter-clockwise (for $i < 90°$), rather than clockwise as in the old program. This make a difference only for pictures of the binary (MPAGE=5).

9. Messages about exceeding limiting lobes are now generated if the stars exceed the lobes at all, instead of only when at least one grid point falls in the hole near the inner Lagrangian point, as in the old program.

10. Star spots now can be made to move in longitude, keeping pace with the physical surface of an asynchronously rotating star, rather than being tied to the coordinate grid. This provision is optional.

11. For both stars, **LC** now provides absolute mass, bolometric luminosity, equivalent sphere radius, and approximate absolute mean surface gravity in the output.

18. Summary: Differences between 1998 & 1992 Versions

1. The output from **LC** now is determined by setting integer MPAGE to 0, 1, 2, 3, 4, or 5. Older versions had light and velocity curve output on the same pages, which made the page format inconveniently wide. The output should now be easier to read. The input file format now is different for the various values of MPAGE (samples are provided with the distributed program).
2. **DC** solutions can include the Marquardt λ factor.
3. The least squares normal equations are in re-scaled form (diagonal terms are unity before application of λ).
4. Semi-transparent circumstellar clouds (at fixed locations in rotating frame) can be included.
5. Rotational spectral line profiles can be computed (other broadening mechanisms no; blending yes).
6. The program now is entirely in double precision.
7. Non-linear limb darkening via a square root law is an added option (1992 version had logarithmic law and linear law).
8. One can now use either time or phase as the independent variable in **DC**. Previously only phase could be used. Use of time as the independent variable allows solutions for the ephemeris parameters $t_{0,}P$, and dP/dt, as well as the apsidal motion parameter $d\omega/dt$.
9. The stepped independent variable in **LC** now can be either time or phase. Previously only phase could be stepped. If time is the stepped variable, phase is computed and also listed. If phase is the stepped variable, time is computed and also listed.
10. **DC** now can read the observational input in 1, 2, 3, 4, or 5 data triplets per line, according the the value of NPPL. The old version read only 5 triplets per line.
11. Input and output formats for the parameters of both **LC** and **DC** have been expanded to more digits. Some quantities that are likely to range over many orders of magnitude are now entered and printed in *D* format. See the sample input data sets and Appendices *Ia* and *Ib* for examples.
12. The argument of periastron, ω, now is in radians rather than degrees. The new parameter $d\omega/dt$ is in radians per day (since t is Julian Date in days).
13. Spot longitudes, latitudes, and angular radii now are in radians, as are their DELS and corrections. Previously they were in degrees.
14. The output format for solution results has been changed and should now be more convenient.
15. **LC** now generates coordinates of plane of sky projected images for use with an external plot program (MPAGE=5).
16. Error estimates printed by **DC** are now standard deviations. Previously they were probable errors. Standard deviations seem to be more commonly used in the literature than probable errors.
17. Simulated observational scatter can be applied to the light curves computed by **LC**. Similar provisions for radial velocities and line profiles are not yet included.
18. The **DC** output now lists the corrected parameter values in addition to the corrections. Previously one had to apply the corrections by hand - the idea being to make certain that users compare new and old values at every iteration. In deference to the significant amount of grumbling generated by that practice, the machine now does the addition. However **DC** still does only one iteration in a given submission.

19. Summary of [0,1] Control Integers

Integer	0	1
IPB	normal value	decouple L from T
IFAT1, IFAT2	black body	atmospheres
IFSMV1, IFSMV2	spots fixed in longitude	spots move in longitude
IFDER	not print derivative matrix	print derivative matrix
IFM	not print normal equations	print normal equations
IFR	not print radii & their derivatives	print radii and their derivatives
ISYM	asymmetrical partial derivatives	symmetrical partial derivatrives
ICOR1, ICOR2	not apply RV proximity corrections	apply RV proximity corrections
KDISK	not use disk scratch pad	use disk scratch pad

20. Summary of [0,1,2] & [1,2] Control Integers

Integer	0	1	2
JDPHS	n.a.	independent variable is time	independent variable is phase
NOISE	no level-dependent weights	scatter scales with $\sqrt{\text{level}}$	scatter scales with level
KO	**DC** does base set only	read from scratch pad	write on scratch pad
MREF	n.a.	approximate reflection	detailed reflection

21. Summary of [1,2,3] & [1,2,3,4,5] Control Integers

Integer	1	2	3	4	5
LD	linear cosine law	log law	square root law	n.a.	n.a.
MPAGE	light curves	velocity curves	line profiles	radii vs. phase	image data

22. Acknowledgments

Thanks continue to the many persons who helped with suggestions, identification of bugs, and direct testing of the 1992 and earlier versions, as cited in the 1992 Documentation booklet. Much testing and problem identification was carried out for the new version by W. Van Hamme. Several bugs in the 1992 version were found by M.A. Cerruti and W. Tobin. Suggestions for improvements in this booklet were received from W. Van Hamme and J. Kallrath.

REFERENCES

Avni, Y. 1976, ApJ, 209, 574

Carbon, D.F. and Gingerich, O. 1969, in *Theory and Observation of Normal Stellar Atmospheres*, ed. O. Gingerich, Cambridge, Mass, MIT Press, p.377

Diaz-Cordoves, J. and Gimenez, A. 1992, A&A, 259, 227

Kallrath, J. and Milone, E.F. 1999, *Modeling and Analysis of Eclipsing Binary Observations*, Springer Publ.

Klinglesmith, D.A. and Sobieski, S. 1970, AJ, 75, 175

Levenberg, K. 1944, Quarterly of Applied Mathematics, 2, 164

Limber, D.N. 1963, ApJ, 138, 1112

Marquardt, D.W., J. Soc. Indust. Appl. Math, 11, 431

Milone, E.F., Stagg, C.R., and Kurucz, R.L. 1992, ApJS, 79, 123

Plavec, M.J. 1958, Mem. Soc. R. Liege, 20, 411

Van Hamme, W. 1993, AJ, 106, 2096

Van Hamme, W. and Wilson, R.E. 1986, AJ, 92, 1168

Wilson, R.E. 1979, ApJ, 234, 1054

Wilson, R.E. 1988, in "Critical Observations vs. Physical Models for Close Binary Systems", ed. K.C. Leung, Gordon and Breach Publ., p. 193

Wilson, R.E. 1990, ApJ, 356, 613

Wilson, R.E. 1993, in *New Frontiers in Binary Star Research*, ed. K.C. Leung and I.S. Nha, A.S.P. Conf. Ser., 38, 91

Wilson, R.E. and Biermann, P. 1976, A&A, 48, 349

Wilson, R.E. and Devinney, E.J. 1971, ApJ, 166, 605

This plano tables was prepared with the AAS LaTeX macros v4.0.