Eclipsing Binaries
– Modeling and Analysis –

WD2006 User Guide

Josef Kallrath & Eugene F. Milone

Report : 2.0
Date : September 25, 2006
State : in progress

WD2006 User Guide

Josef Kallrath & Eugene F. Milone

September 25, 2006

Contents

Preface

This documentation describes the mathematical foundation and the capabilities of the program `WD2006` developed by Josef Kallrath, and serves as a User Guide for this program. The program runs under the operating system *LINUX* as well as in a DOS Window/DOS box under Win95, Win98, WinNT, Win2000 and WinXP. It should also run under other UNIX operating systems and *CygWin*; however, this has not been thoroughly tested. `WD2006` enables the user make use of the Wilson-Devinney[1] (`WD`) program to compute eclipsing binary light curves[2] among other observables, and to analyze data, *i.e.*, to fit light curves or merely to compute a synthetic light curve. `WD2006` is more than a wrapper around the `WD` program. It has tight link with the original `WD` code. The package itself is still maintained and further developed solely by Josef Kallrath. The documentation has been improved significantly and, from 2005 on, is maintained by J. Kallrath and E. F. Milone.

It should be understood that modeling eclipsing binaries and solving inverse problems in such a context is a major research effort and requires some expertise to use the software effectively. Wherever possible, great effort has been invested to make the software as stable as possible.

`WD2006` has been developed within the framework of a several preceding software packages named `LCCTRL`, `WD93`, `WD95`, `WD98`, and `WD2002`. `WD2002` was the first release that combined all previous developments and included all stellar atmospheres improvements added by E. F. Milone and C. R. Stagg, and was also the first release which ran under *LINUX* and other *UNIX* operation systems.

`WD2006` adds simulated annealing as a minimization algorithm, and produces tables of absolute dimensions, and, in particular, LaTeX output tables. It also supports and embeds a Fortran subroutine for the computation of limb-darkening coefficients as provided by Walter Van Hamme.

The procedure for creating the flux files is described in the appendix along with changes to the atmospheres option in `WD` programs. The use of the flux ratio files that make use of Kurucz atmospheres models are coded in *Subroutine ATM2000* (which replaces subroutine ATM) by C. R. Stagg.

We provide the LaTeX source file, *docu.tex*, in the `WD2006/DOCUMENT` directory to encourage the new user to add appropriate comments, corrections or extension directly into this file. That way, we can eliminate problems in a short time. We also appreciate feedback in case you experienced difficulties.

September 2006 Josef Kallrath[3] and Eugene F. Milone[4]

[1]Release 1998 distributed by Robert E. Wilson (University of Florida).
[2]The term *light curves* is here used in the general sense of eclipsing binary observables [see, Kallrath & Milone (1999)].
[3]Weisenheim am Berg (Germany), josef.kallrath@web.de
[4]Calgary (Canada), milone@ucalgary.ca

Chapter 1

Basics of `WD2006`

In short, `WD2006` is an eclipsing binary simulation and optimization tool which enables the user to

- carry out simulations using the orginal Wilson-Devinney programs `LC` and `DC`,
- to use the Kurucz atmospheres and to model better the wavelength-dependent stellar flux,
- fit eclipsing binary observables using the simplex method, differential corrections, a Levenberg-Marquardt scheme, or simulated annealing,
- to do automatic iterations,
- produce `gnuPlot` graphics files, and
- to obtain best-fit solutions for grids or tables over some fixed parameters which otherwise may be poorly determined.

The program `WD2006` is a stand alone Fortran77 program for solving direct and inverse problems in the context of eclipsing binaries. It is called from a UNIX script file or a DOS batch file, and connects to the main subroutines of our version of the Wilson-Devinney program, namely, `LC` and `DC`. These are fed the input data through the `WD2006` following input files:

1. a general control file keeping all information of *what* and *how* `WD2006` should work, **.inf*;
2. a file containing upper and lower bounds on the adjustable parameter to be fitted. **.con*;
3. a `DC` input file controlling the Wilson-Devinney subroutine `DC` to evaluate light and radial velocity curves for given time or phase values or to fit the model parameters to those data, **.dci*;
4. an `LC` input file controlling the Wilson-Devinney subroutine `LC` to evaluate light and radial velocity curves and other observables for a grid of time or phase values, **.lci*;
5. a set of flux files providing the ratio between the Kurucz atmosheres and a black body radiator; *Rflux*, *Bflux*, *Vflux*, *Vm05*, *Im10*, etc. [The latter two files involve atmospehere models which have non-solar metallicities]; and

6. an ASCII file containing the gnuPlot commands to produce graphics; *fit.gnu*.

The main parts of the program `WD2006` are:

1. subroutines to read the **.inf* and **.con* input files;
2. a subroutine to interface with and call `LC` and `DC`;
3. subroutines that control the solution to the inverse problem through the simplex method, the Levenberg-Marquardt scheme, or Simulated Annealing;
4. a subroutine to implement a variant of the simplex algorithm ([12], [7]) also described in Kallrath & Milone (1999);
5. a subroutine to implement a variant of the Levenberg-Marquardt scheme described in Kallrath *et al.* (1998) and Kallrath & Milone (1999);
6. a subroutine to implement a variant of Simulated Annealing ([1], [4], [2], [11], [10]), and
7. several subroutines to handle character strings in Fortran.

The main output is the solution to the inverse problem, *i.e.*, light curve fittings. Structurally, the general structure of `WD2006` is summarized as follows:

$$\begin{tikzcd}
1 & 2 & 3 & \text{return to} \\
\textit{r case} \arrow[r] & \textit{wd2006.for} \arrow[r] & \texttt{LC,DC} \arrow[r] & \textit{2} \\
\text{csh script} & \text{Fortran} & \text{Fortran} &
\end{tikzcd}$$

Chapter 2

Installation of `WD2006`

2.1 The Language

The PC-versions of `WD2006` can be compiled with the LAHEY Fortran compiler 5.0 in a DOS box, or with the *gnuFortran compiler* which is part of the SUSE 7.0 Linux distribution. The program language for all *wd2006.f* is Fortran 77. Additional source files are *parsinf.f* and *lcdc98b.for*; the batch command `MAKE` (DOS) or the script file *c* (*LINUX*) control the compilation process.

2.2 Installation

`WD2006` comes with an installation disk for running in a *DOS* box. Follow the instructions in the *readme.txt* file on the installation disk.

LINUX user should just take the files in WD2006/Linux. We plan to provide the tar file *wd2006.tar.gz* as an alternative. To install the software, the following operations are applied to this file: It produces the following directory tree:

WD2006	all source files,[1] script files, documentation
WD2006/DOCUMENT	this documentation file *wd2006.pdf* and *docu.tex*
WD2006/Linux	all files relevant to LINUX users
WD2006/TestWD	a set of input and output files to test the installation
WD2006/Utility	a set of utility files (for DOS users)

In addition, the *star.tar.gz* file can be installed at any location using the commands

gzip -d *star.tar.gz*	This unzips the file and produces *star.tar*
tar xvf *star.tar*	This un-tars the file

The unzipping may be accomplished also by the command: gunzip *star.tar.gz*. These procedures will produce the following directory tree:

STARS	the stars directory
STARS/SVC06	a starname directory containing files to analyze the eclipsing binary star SV Cam

[1] The download distribution does not contain the source files. Send a request to josef.kallrath@web.de to obtain them if necessary.

The current version of the software is started in a DOS box or in a *csh* or *tcsh* shell using the DOS batch file *r.bat* or the UNIX script file *r*, *i.e.*, by the command

r case

where *case* corresponds to a set of task for `WD2006` to perform described in the *case.inf* input file. After this command has been used once in an directory, it is sufficient just to type `r` because the `name` is memorized. The TEST2006 scenario can be used to test the proper installation of `WD2006` using the command *r v570p*. The script file can also be called as

$$r\ /h \tag{2.1}$$

to produce a help screen showing all options available. The user has to define the environment variables $WDNAME$, $WDPATH$, *GnuPLOT*, and $STAR$ in the *.cshrc* script file and to set it to the name of the directory in which `WD2006` resides. In Win2000 and WinXP the environment variables can be set in the Windows setup or in a DOS box using the *wdini.bat* file provided in the `WD2006` directory. The *wdini.bat* might look like

```
REM   set the name for %WDNAME%
       SET WDNAME=WD2006

REM   set the path where %WDPROG% is located
       SET WDPATH=J:\%WDNAME%

REM   set the path for GnuPLOT
       SET GNUPATH=e:\math\gnuplot
```

Under *LINUX* or UNIX systems, the command to do so might, for instance, look like this:

```
   setenv WD2006 /home/yourusername/ASTRO/WD2006
   setenv STAR /home/yourusername/ASTRO/STAR
```

See the comment lines in the *r.bat* and *test.inf* files indicating where the user should make changes)
Another script file, *c*, can be used to compile the software and provides the following commands:

c	to compile and link *wd2006*	(2.2)
c /c	to remove all temporary files	
c /h	to show a help file on how to use the script *c*	
c /pf	to plot the observed & computed light curves	

It is recommended that each scenario or case has its own subdirectory in order to support several jobs running simultaneously. For further details, the user is referred to Section 3.5.

2.3 Producing Plots

Plots can be produced using `gnuPlot`. Currently, it is possible to plot the *light curve fit* [*r /pf*] and the residuals.. The files *fit.gnu* describe how the plot is produced. This file needs to be adjusted for each case depending on the number of radial velocity curves and photometric light curves. When these commands are used, `gnuPlot` produces automatically the associated PostScript files. The observables to be plotted, and the plot labels and the axes labels may be altered in a file named *fit.gnu*, which should be placed in the particular stars working directory. The TestWD/Tst-01 directory contains an example of a *fit.gnu* file.

Chapter 3

Description of Input Files

In `WD2006` the hierarchy of input files is as follows:

$$\begin{tikzcd}
& \textit{*.inf} \arrow[d] \\
\textit{*.dci} \arrow[d] & \textit{, *.lci ,*.con , *.pot} \ \{\textit{*.spt}\} \\
\textit{new.dci, new.lci} &
\end{tikzcd}$$

The user should inspect the files that are located in, for instance, the TestWD\Tst-01 directory. Great care is needed to transfer ASCII files from *LINUX* to *DOS* or from *DOS* to *LINUX*. The proper interpretation of strings can be disturbed by additional or missing *carriage returns* and *line feeds*. If such a problem occurs, the following procedure may help to cure the problem: create a new file, say *newfile.dat*; edit this file with the *vi* editor and include the problematic file, say *problem.dat*; save the new file *newfile.dat*; delete problem.dat file and rename the *newfile.dat* to *problem.dat*. Alternatively, when using the vi-editor, one may use the commands

$$\text{set fileformat=unix} \tag{3.1}$$

or

$$\text{set fileformat=dos} \tag{3.2}$$

to adjust the file format.

3.1 Description of the *wd2006.inf* file

To understand this control file, the reader might inspect the file *v570p.inf* in the *TestWD\Tst-1* directory. `WD2006` always asks for the control file under the name *wd2006.inf*. In order to run different jobs started with the csh script *r* one can produce different **.inf* files. The very *r* files copy **.inf* in *wd2006.inf*. To avoid confusion, each sceanario this should be done in different subdirectories.

A particular *wd2006.inf* contains information on how to use the **.inf* file itself. For convenience, the user may divide *wd2006.inf* into different sections according to the tasks to be performed. It is best not to restructure this file; probably such an action would produce chaos!) *wd2006.inf* has been set up in a way to make it easy to enter keywords [see Appendix A for a complete list] and values to be assigned to parameters. Most keywords require one parameter, *e.g.*,

MAXIT 5

This keyword and the value 5 assigns the value 5 to the internal integer Fortran-Variable `MAXIT`. The input in *wd2006.inf* is free-format, *i.e.*, the keyword may appear anywhere in the file and the value is allowed to follow the keyword with some space allowed in between. However, nothing should follow upon the value. Regarding the value the implicit Fortran declaration rules apply, *i.e.*,

$$\begin{array}{l} \text{IMPLICIT} \quad \text{INTEGER} \quad (\text{I} - \text{N}) \\ \text{IMPLICIT} \quad \text{DOUBLE PRECISION} \quad (\text{A} - \text{H}, \text{O} - \text{Z}) \end{array}$$

In case of doubt, the user should consult the table in Appendix A. Note that "MAXIT 5.0" would lead to an error: only integer values are allowed. Some keywords require a character string, *e.g.*, keywords passing file names. In that case, the string to be passed must not contain spaces. If the user does not specifically invoke a keyword, a default value is assigned to the associated parameter. The following table shows some of the available keywords , their default values and a brief description:

ALPHA	1.00	simplex reflection parameter
BETA	0.50	simplex contraction parameter
EPSSTP	0.003000	define when to stop simplex iterations
FNDCI	init.dci	name of the DC input file
FNDCO	init.dco	name of the DC output file
FNSMPE	init.spe	name of the simplex output file (errors)
FNSMPO	init.spo	name of the simplex output file (parameters)
FNSMPT	init.spt	name of the simplex restart file
FORMO1	F10.5	specify format for output of the parameters
FORMO2	F9.5	specify format for output of the errors
GAMMA	2.00	simplex extension parameter
IFSPOT	1	use spot data from spot file *.pot
IFOUTL	0	control how to apply the outlier criterion
IGRAPH	2	control graphics output
INITD	1	divide all simplex sizes by INITD
IOSTYN	1	store simplex tableau
IRSTRT	0	restart from a previous simplex run
IWRITE	1	produce a restart file *.spt
JOB01	-1	specify the task
KSIGMA	3	quantify the outlier criterion
LPR	0	specify print level
MAXILM	15	specify the number of Levenberg-Marquardt iterations
MAXISX	15	specify the number of Simplex iterations
MINIT	1	control the setup of the initial simplex

wd2006.inf also contains some parameters which are transferred to subroutine *simplex.for*. These are, in particular, the number of iterations and the geometrical parameters for the operations *reflection*, *contraction*, and *expansion* described in Kallrath & Milone (1999,[8]). The parameter `MAXIT` for defining the maximum number of iterations can also be used in a somewhat different meaning. `MAXIT`$= -1$ causes the program to construct only the initial simplex without evaluating the objective function, *i.e.*, the standard deviation. This feature is useful to see how the initial simplex looks. `MAXIT`$= 0$ constructs the initial simplex and evaluates the standard deviation of the fit.

The user must specify the parameters' increments which define the size of the initial simplex. The parameter `MINIT` defines which mode should be used to construct the initial simplex. See Kallrath&Linnell (1987,[7]) for the mathematical differences of these modes.

`MINIT`$= 1$ will calculate an initial simplex according to Yarbro&Deming (1974).

`MINIT`$= 2$ produces an initial simplex with its center near the initial solution.

`MINIT`$= 3$ embeds the initial simplex in a hyper cube.

The parameter `IOSTYN` allows to specify the output format. With the value 1 this parameters causes all simplex tableaus with their mean value in each column to be printed out.

Some commands do not have only one argument they pick-up but rather have several. Such commands are called *multi-commands* and are supported by `WD2006`. As an example we give the command

syntax	:	`RMLOOP` *strt stop incr dir*
example	:	`RMLOOP` 0.80 0.901 0.02 UP

which forces `WD2006` to compute the best fits for fixed mass ratios, $q = 0.8, 0.82, \ldots, 0.90$. Instead of `UP`, one can also use the keyword `DN`. The multi-commands `PELOOP` and `XILOOP` work similarly and construct loops for the orbital period and the inclination, respectively.

Another multi-command is

syntax	:	`PHSRNG` *phsstrt phstop phincr*
example	:	`PHSRNG` 0.0 1.0 0.01

which sets the phase range in subroutine LC, *e.g.*, light curve points separated by 0.01 in the phase range between 0 and 1.

3.2 The DCI Input Files *.*dci*

Although the dci file in our single-iteration unix-box Wilson-Devinney program, *wd98k93h* (*h* is the latest version at present writing), has a number of lines at the top of the file to indicate DC ('2'), log g for stars 1 and 2, flux file names (one for each curve), and the `END` statement, the `WD2006` **.dci* file begins with the increments as per the Wilson manuals (ref: WILSON 1993). Now, however, there are 34 adjustable parameters (a 35th is available for future use). These take 3 lines. The order of the delta values for the parameters is the same as for the 'keeps', which we describe next.

Line 4 contains the 'keeps', binary switches to indicate which parameters are to be adjusted ('0' if yes; '1' if no). The order of these switches is as follows: 1111 1111 1111111 11111 11111 11111 11111 1 1 1 The first two sets, eight keeps in total, specify spot parameters to be adjusted, as in previous versions of the Wilson-Devinney program. The first quartet specifies parameters for spot 'A", the second quartet, those of spot 'B'. Each quartet adjusts the latitude, longitude, star spot radius, and temperature factor of the spot to the local temperature expected in the absence of the spot. the first three parameters of each quartet are to be given in radians. The third set specifies : a, e, ω, F_1, F_2, pshift, V_γ. The fourth set specifies: ι, g_1, g_2, T_1, T_2. The fifth set specifies: A_1, A_2, Ω_1, Ω_2, q. The sixth set specifies: t_0, P_0, $\mathrm{d}P/\mathrm{d}t$, $\mathrm{d}\omega/\mathrm{d}t$. The seventh set specifies: L_1, L_2, x_1, x_2, and l_3. The three separated switches at the end of the line control, respectively: the printing of the derivatives (`ifder`); the matrix of the normal equations (*ifm*); and the radii and the derivatives (`ifr`), respectively.

The fifth line contains four control integers, which determine, respectively: which star has spot A (`KSPA` = 1 or 2); which of the many possible spots of that star the 'A' spot is (`NSPA` = 1,2,3,4, ...); which star has spot B (`KSPB` = 1 or 2); and which of the many possible spots of that star the 'B' spot is (`NSPB` = 1,2,3,4, ...).

The sixth line specifies the number of curves and other control integers: a binary switch to indicate of a radial velocity curve of star 1 is included below in this dci file (`IFVC1`; a similar switch for star 2 (`IFVC2`; the number of light curves (*i.e.*, not including the radial velocity curves) included in this file `NLC`; the use of a 'scratch pad' to store results for future use (`KO` = 0, 1, or 2),[1] a control integer to use the scratch pad or not (`KDISK` = 0 or 1) if used;[2] a control integer to establish whether asymmetric or symmetric derivatives are to be used, (`ISYM` = 0 for the asymmetric case, or 1 for the symmetric case. We recommend 1 for all work where the highest precision and accuracy are desired, at the cost of a factor of two in computing time); and the number of triads per line of input velocity or light curve data in the dci file (`NPPL` = 1,2,3, ... We generally prefer 3 as a balance between efficiency and readability).

The seventh line also contains control integers: the number of reflections in a multiple reflection scenario (`NREF` = 1, 2, ...); a control integer to determine the type of reflection effect to include in the model (`MREF` = 1 simple, single-reflection reflection effect or 2 multiple reflections); control integers to determine whether any spots on the stars are permitted to have fixed longitudes or are allowed to move with a (nonsynchronous) rotating surface (`IFSMV1`, `IFSMV2` = 0 for fixed case, or 1 for varied longitudes); control integers to determine if proximity effects are to be applied to radial velocities (`IFCOR1`, `IFCOR2` = 0 to not apply them, or 1 to apply them); and the limb-darkening mode (`LD` = 1 (for linear), 2(for logarithmic) and 3 (for square root) for the various limb-darkening laws.

The eighth line has: a control integer specifying if the data are given in heliocentric Julian day numbers or in phase (decimals of a cycle) (`JDPHS` = 1 for time or 2 for phase); the epoch in JDNs; the period in days; the rate of change of period; the rate of change of ω; the phase shift to be added to a synthetic or computed light curve phases.

The ninth line contains: a control integer which establishes the `Mode` (the basic model that the light curve must fit: -1 for x-ray binary models in which x-ray eclipses are observed; 0 simulating the Russell-Merrill decoupling of the stars' luminosities with each other and with their temperatures; 1 for overcontact binaries, where internal constraints on the Ω, T, g, A, L, x and y are not adjustable for the second star; 2 for detached binaries; 3 for overcontact binaries but no constraints are placed on the temperature or the limb-darkening in `WD2006` of the second star; 4 for semidetached binaries in which star 1 fills its lobe exactly; 5 for semidetached binaries in which star 2 fills its lobe exactly; and 6 for double contact binaries in which both stars exactly fill their Roche lobes , *i.e.*, their inner Lagrangian surfaces); a control integer that allows or severs the coupling between temperature and luminosity (`IPB` = 0 for normal coupling and 1 for decoupling. We recommend 0 for nearly all cases); control integers to permit stellar atmospheres fluxes to replace blackbody fluxes for each star (`IFATM1`, `IFATM2` = 0 if not or 1 if so. We recommend using stellar atmospheres fluxes over blackbody fluxes whenever possiblea. In `WD2006` we use Kurucz atmosphere models of both solar and non-solar compositions. There is a lower temperature limit for these models); numbers of grid elements per star (`N1`, `N2` $\geq$30, typically); numbers of course grid integers on each star (`N1L` and `N2L`, normally set to half the values of `N1` and `N2`); perr0, the argument of periastron, in radians; the rate of change of perr0, dperdt, also in radians, `THE` ; and the units of radial velocity (`Vunit`), which, when multiplied with the data specified in the **.dci* file, gives the radial velocity

[1] 0 means do not use; 1 causes the program to read in the derivatives and residuals written to the scratch pad in a previous run; 2 causes the program to write the residuals and derivatives to the scratch pad for future use or to use with subsets in the current set. As `WD2006` does not make use of scratch pads or of subsets the parameter `KO` should be set to 2);

[2] `KDISK` = 0 if not used or 1 if used. In `WD2006`, set `KDISK` to 0.

in km/s.

The tenth line specifies: eccentricity, e; semimajor axis, a; the ratio for each star of the rotation to the revolution angular speed (`F1`, `F2` = 1 for the synchronous case); the radial velocity of the system's center of mass (V_γ); the inclination of the orbit to the plane of the sky (ι); gravity brightening coefficients, sometimes called gravity darkening coefficients for each star (g_1, g_2).

The eleventh line contains: the temperatures of each star, T_1, T_2; the bolometric albedos of each star, A_1, A_2; the potentials of each star, Ω_1, Ω_2; the mass ratio of star 2 to star 1, q; the limb darkening coefficients for each star, $x(bol)_1$, $x(bol)_2$; the second order limb darkening coefficients, for non-linear limb darkening, for each star ($y(bol)_1$, $y(bol)_2$;

The lines that follow immediately are curve dependent; i.e., they are determined by what curves are included in the data. For each radial velocity curve, there must be a line that includes: the effective wavelength; the passband 'luminosity' for each star (L_1, L_2); the passband limb darkening coefficients for each star ($x(WL)_1$, $x(WL)_2$) a measure of the standard error of the RV curve (`sigma` in units of `Vunit`). For each light curve, there must be a line that provides: the specifications of effective wavelength of the passband and the passband 'luminosities' of each star, the x and y limb darkening coefficients as in the RV curves. These are followed by: the third light (l_3, a quantity to be added at every phase to the computed light); a control integer to apply level-dependent weights (the `noise` parameter = 0 for no level dependent weight, 1 for noise which scales like shot noise, 2 for noise that is proportional to the light level, such as scintillation); the standard error in the light curve (`sigma`, as in the RV lines).

These lines are followed by spot specifications, one line for each spot: latitude; longitude; spot angular radius; and spot temperature factor.

These are followed by the data: triads of: time or phase, velocity (in `Vunit` values) or normalized light (*i.e.*, normalized to maximum light) and weight.

Fianlly, there are some end indicators in the form of -10001, -10002, or -10003 depending how many entries of a row are really fulfilled.

wd2006.inf passes the indicated **.dci* files to `WD2006`. See the *Wilson-Devinney User Guide*[3] for instructions on how to set up a **.dci* file.

3.3 The LCI Input Files **.lci*

wd2006.inf passes the indicated **.lci* files to `WD2006`. See the Wilson-Devinney User Guide for instructions on how to set up a **.lci* file. The **.lci* files can be generated by `WD2006`.

3.4 The Spot Input Files **.pot*

If the parameter `IFSPOT` is set to `IFSPOT`= 1, the spot data are taken from the file **.pot*. The structure of this file is demonstrated by the following example:

```
! Name of this file: rtlacme3e.pot

! This example contains two spots on star number 1

1 1 90.000 270.000 45.000 0.970 1 1 1 0 32.120 10.200 2.300 -0.324
```

[3]The original LC and DC programs can be downloaded from Robert E. Wilson's homepage at the university of Florida. A documentation is part of this.

```
1 2 90.000 000.000 45.000 0.950 1 1 1 0 32.120 10.200 2.300 -0.324
```

Note that the symbol '!' as well as the '*' sign can be used to comment out a line. Empty lines are ignored. The meaning of the entries is as follows:

1	number identifying which star the spot is on
2	number identifying which spot on that star is being described
3	spot "latitude" (actually its co-latitude) in degrees
4	spot longitude in degrees
5	spot radius in degrees
6	a dimensionless temperature factor (ratio of spot to photosphere temp.)
7-10	retain (1) or adjust (0) the parameters 3 to 6
11-14	size parameters to construct the initial simplex

From this data, the number of adjustable spots is detected automatically.

3.5 The UNIX Script Procedure *r* [csh/tcsh]

It is recommended that all jobs be started by a script procedure. The *r* files copy the local **.inf* file into the *wd2006.inf* file which is a dummy file as seen from `WD2006` and used to start `WD2006`. In addition, several other commands and options are supported. The command *r /h* prints the help screen shown below.

```
 The UNIX script file 'r' can be used in 'csh' or 'tcsh' shell environments.
 It follows the following syntax:

   r          : run 'wd2006' with the current wd2006.inf file
   r /c       : clean the directory and remove all temporary files
   r /g       : copy script files from WD2006 directory to current directory
   r /h       : print this help screen
   r /pf      : plot the light curve fit
   r /s  name : copy all relevant files to a directory named NAME
   r /u  name : copy r to all scenario directories

 Usually, a modeling run starts with "r" followed by the name of the star
identified in the .dci, .inf, and .con files. For example:
   r  rtlacme3e

 In addition, the UNIX script file 'c' can be used in 'csh' or 'tcsh' shells.
 In the WD2006 directory the following commands are available:

   c     : compile and link 'wd2006' with GNU's Fortran Compiler
   c /b  : copy all files in the WD2006 directory to a backup directory
   c /c  : clean the directory and remove all temporary files
   c /d  : compile/debug & link 'wd2006' with GNU's Fortran Compiler
   c /h  : print this help screen
   c /s  : save all relevant files in the wd2006 directory to WINDOWS
```

The user should carefully maintain this file. If changes are made, the *r* script file needs to be distributed to all scenario directories. In order to do so, the user has to modify the UPDATE section and to include new scenarios, and to excute the command *r /u*.

Chapter 4

Description of Output Files

In `WD2006`, depending on the tasks described in Chap. 5 the following output files are generated:

tasks	list of output files
1	*.log, *.lco
2	*.log, *.dco
3	*.log, *.spo, *.spe, *.spt
4	*.log, *.dco
5	*.log, *.dcc, *.dd
6	*.log, *.dco
7	*.log, *.sa

Chapter 5

Overview on the TASKS

`WD2006` provides several algorithms called by individual tasks to fit given data to the Wilson-Devinney light curve model. Each task corresponds to one job to be executed. It is possible to do up to 9 successive jobs in one run of `WD2006` by assigning the appropriate task number to the keywords `JOB01` to `JOB09`. These assignments are made within the *.inf* file. The following tasks are available:

1. call `LC` for one simulation run;
2. call `DC` for one differential correction step;
3. improve the shape using Nelder & Mead's simplex algorithm (Kallrath&Linnell, 1987);
4. call `DC` for one simulation run;
5. –not implemented in `WD2006`–;
6. Levenberg-Marquardt scheme;
7. Simulated Annealing.

Some tasks can follow each other, *e.g.*, one first could use the simplex algorithm to produce an initial solution and then switch to a derivative based method. Information can be exchanged between tasks.

5.1 Description of Task 1

Task 1 calls `LC` for one simulation run and produces a **.lco* file. This provides computed or synthetic light andor radial velocity curves.

5.2 Description of Task 2

In this task, subroutine `DC` is called for one differential correction step.

5.3 Description of Task 3

This mode initializes light curve fitting with the simplex algorithm. It will produce **.spo* and **.spe* output files containing the information about the iteration and also about the mean errors. The current implementation observes bounds on the parameters.

Furthermore a **.spt* file is produced if `IWRITE`= 1. There are some good reasons to make use of this feature. Unlike the **.spo* which is 'locked' during the whole job, the **.spt* file is open to the user immediately after it has been written. Therefore this provides a tool for the user to control the job [*** needs to be implemented ***]. The user can interact with a running job using the control file *info.lct* which contains only the value 0 or 1. After each iteration the simplex routine checks this value. If it is 0 than it will continue with the next iteration. Otherwise it will read the next *wd2006.inf* file. If there is no other *.inf* file then it will stop the batch job. In all cases it is a controlled stop and all data are saved for a later restart.

As described in Section 9.3 it is possible to continue the optimization from an intermediate solution by exploiting the restart parameter `IRSTRT`.

5.4 Description of Task 4

In this task subroutine `DC` is called to evaluate the light curve for the given data in time or phase.

5.5 Description of Task 5

This task is not implemented in `WD2006`.

5.6 Description of Task 6

In this task a variant of the Levenberg-Marquardt algorithm described in Kallrath *et al.* (1998) and Kallrath & Milone (199) is used to fit the light curve parameters. This damped least squares option obviates the need for subsets, and these are not permitted in any of the WD2006 programs. (However the subset option is available in the stand-alone versions of WD98k93, which also make use of the damped least squares option.)

5.7 Description of Task 7

In this task simulated annealing as described in Kallrath (2004) is used to fit the light curve parameters. Simulated annealing may require a lot of computer time and involve some 10^4 to 10^5 light curve evaluations. Therefore, it is best to adjust only a small number of parameters at a time. The computer time required to run the algorithm scales linearly with the number of data points.

Chapter 6

Detailed Description of Multi-Commands

This chapter describes all multi-commands in more details and in alphabetic order. Multi-commands are used in the **.inf* files.

6.1 IFREDUC

Here we use a binning algorithm to reduce the number of data points so the computing is done faster with any of the minimization routines.

The multi-command `IFREDUC` is used to work with reduce data sets. This saves a significant amount of computing time in the initial phase of working with new data sets for a star, and it is also very helpful when using simulated annealing.

syntax	:	`IFREDUC`	$ifreduc$	N_p	N_i	Δ
example	:	`IFREDUC`	0	2	20	0.025

where the input parameter have the following meaning:

- 1. The integer flag *ifreduc* controls whether this feature is activated or not; [0].
  2. The integer parameter, N_p, specifies how many data points should be used in each bin; [2].
  3. The integer parameter, N_i, specifies how many equally sized intervals or bins should be used; [20].
  4. The real parameter, Δ, specifies the width of the minima. The number of data points is not reduced in the minima; [0.025]. If they should not be reduced, set Δ to a very small value, say $\Delta = 0.001$.

Chapter 7

Basic Assumptions, Physical Foundations and Limitations of `WD2006` and `LC/DC`

The mathematical and physical foundations of `WD2006` and `LC/DC` are described in the Wilson-Devinney User Guide. For the mathematical foundations we refer to Kallrath and Milone (1999). Here we make only a few, hopefully useful, comments.

7.0.1 Period Changes

The current (1998) version of the Wilson-Devinney program supports only linear period changes described by one constant parameter, $\dot{P} = \mathrm{d}p/\mathrm{d}t$. The manual accompanying the Wilson-Devinney program expresses, that this quantity is dimensionless. Actually, more precisely, the dimension is

$$\left[\dot{P}\right] = \frac{[\mathrm{d}p]}{[\mathrm{d}t]} = \frac{\mathrm{days}}{\mathrm{days}} \tag{7.1}$$

It is useful, to relate the Wilson-Devinney parameter, which is used in the *.*dci* and *.*lci* input files to the quantity $\mathrm{d}P/P$ or the coefficient, C, used in the ephemeris specified as

$$JD_{\min} := E_0 + PE + CE^2 \quad , \tag{7.2}$$

where $JD_{\min}$ gives the time of the minimum associated with epoch, E, E_0 is some reference epoch, P is the period and C the coefficient of the quadratic term. Then the following relationships hold among $\dot{P}$, $\mathrm{d}P/P$, and C:

$$C = \frac{1}{2}\frac{\mathrm{d}P}{P} = \frac{P}{2}\dot{P} \quad , \tag{7.3}$$

or, equivalently,

$$\dot{P} = \frac{2C}{P} = \frac{1}{P}\frac{\mathrm{d}P}{P} \quad , \tag{7.4}$$

and

$$\frac{\mathrm{d}P}{P} = 2C = P\dot{P} \quad . \tag{7.5}$$

Furthermore, we note that

$$365.25 \times 86400\dot{P} \tag{7.6}$$

gives the period change in seconds/year.

Chapter 8

Comments on the Minimization Algorithms

The package WD2006 includes the simplex method, differential correction, a Levenberg-Marquardt scheme, and simulated annealing as algorithms to solve the nonlinear least squares problem. The first three algorithms are well covered in Kallrath & Milone (1999). Therefore, here we treat only those of the simulated annealing method that is embedded in the package.

8.1 Simulated Annealing

Simulated annealing is a global optimization method that distinguishes between different local extrema. Starting from an initial point, the algorithm takes a step and the function is evaluated. When minimizing a function, any downhill step is accepted and the process repeats from this new point. An uphill step may be accepted. Thus, it can escape from local minima. This uphill decision is made by the Metropolis criteria. As the optimization process proceeds, the step length decreases and the algorithm closes in on the global optimum. Since the algorithm makes very few assumptions regarding the function to be optimized, it is quite robust with respect to non-quadratic surfaces. The degree of robustness can be adjusted by the user. In fact, simulated annealing can be used as a local optimizer for difficult functions.

This WD2006 implementation of simulated annealing is based on the Corona *et al.* (1987,[3]) simulated annealing algorithm for multimodal and robust optimization as implemented and modified by Goffe *et al.* (1994,[6]). The implementation also contains a multimodal example from Judge *et al.* (1985,[5]).

8.1.1 Learning to use SA: step by step instructions

Use the sample function from Judge with the following suggestions to get a feel for how SA works. When you have done this, you should be ready to use it on almost any function with a fair amount of expertise.

1. Run the program as is to make sure it runs okay. Take a look at the intermediate output and see how it optimizes as the temperature, T, falls. Notice how the optimal point is reached and how falling T reduces the step size, VM.

2. Look through the documentation to SA so the following makes sense to you. The core of the algorithm is described in Judge on pp. 68-70 and on pp. 94-95. Also see Corona *et al.* (1987,[3]).

3. To see how it selects points and makes decisions about uphill and downhill moves, set `IPRINT` $= 3$ (very detailed intermediate output) and `MAXEVL` $= 100$ (only 100 function evaluations to limit output).

4. To see the importance of different temperatures, try starting with a very low one (say $T = 10^{-5}$). You will see that (i) it never escapes from the local extremum (in annealing terminology, it "quenches") & (ii) the step length, **s**, (VM) will be quite small. This is a key part of the algorithm: as temperature, T, falls, step length does too. As a minor point here, note how **s** is quickly reset from its initial value. Thus, the input **s** is not so critically important. This is all the more reason to examine **s** once the algorithm is underway.

5. To see the effect of different parameters and their effect on the speed of the algorithm, try $R^{\mathrm{T}} = 0.95$ and $R^{\mathrm{T}} = 0.1$. Notice the vastly different speed for optimization. Also try $N^{\mathrm{T}} = 20$. Note that this sample function is quite easy to optimize, so it will tolerate big changes in these parameters. R^{T} and N^{T} are the parameters one should adjust to modify the runtime of the algorithm and its robustness.

6. For light curve analysis, try constraining the algorithm with the lower and upper bounds specified in the **.con* file.

Chapter 9

How to Use `WD2006`

After setting up a scenario in a directory, `WD2006` could be started using the command *r name*. Depending on the task, `WD2006` is called with

9.1 Setting Up a Consistent Set of Input Files

When `WD2006` is started, a validation routine checks whether or not various input data are provided consistently. The validation routine tries to detect problems early in the running of `WD2006`. Nevertheless, it is a good idea to check the input files carefully. If a job stops almost immediately, the log file *wd2006.log* may contain information on why the program stopped so soon.

9.2 Description of the Output

`WD2006` produces the following output data and files:

1. *new0*.dci*: iterated input files for subroutine DC
2. *new0*.lci*: iterated input files for subroutine LC
3. *resid.dat*: residuals for each curve and point in curve
4. *stdev.dat*: standard deviation for each curve. This file is very useful if a reasonable looking fit has been obtained. The standard deviation then gives the internal error in (= noise of) the curve. These values can be used to set the values of SIGMA in the Wilson-Devinney input file, **.dci.* Although they are formal uncertainties, traditionally these quantities are called “curve weights”.
5. *vgam.dat:* the systemic velocity for radial velocity curve plots, specified in “vunits.” For example, if *vunit* is 100, then this output datum is in units of hundreds of $kmsec^{-1}$.
6. **.dco*: output of subroutine `DC`, containing the resultant optimized parameters, the corrections to the final parameters and the uncertainties.
7. **.ddc*: output of the Levenberg-Marquardt scheme, the optimized parameters.

8. **.lco*: output of subroutine `LC`, the synthetic curves, along with a list of the parameters used to construct the computed data.
9. **.log*: a log file containing various information about the current run
10. **.spo*: the vertices of the Simplex algorithm
11. **.spe*: the errors of the Simplex algorithm
12. **.spt*: a Simplex restart file
13. *C.0**: the calculated light curve
14. *O.0**: the observed light curve
15. *R.0**: the residuals of the light curve

9.3 How to Continue from a Previous Run

The use of the simplex algorithm in `WD2006`, an **.spt* file is produced if `IWRITE`= 1. Some thought needs to be be given to the restart parameter `IRSTRT`, the values of which may be:

- `IRSTRT` = 0 — simplex algorithm starts with an initial simplex
- `IRSTRT` = 1 — simplex starts from an **.spt* file
- `IRSTRT` = 2 — simplex reads vertices from an **.spt* file but recalculates the standard deviation for each vertex
- `IRSTRT` = 3 — simplex reads **.spt* file and constructs a **.spo* file
- `IRSTRT` = 4 — simplex reads **.spt* file and constructs the 'Hessian matrix'

Option 3 can be useful when the job was not executed for some hardware reasons *and/or* the **.spo* was lost. Option 4 is not yet really realized in version 1.0. To continue iterations with the simplex algorithm the user only has to take the present **.inf* file and to change `IRSTRT` to 1.

9.4 Using Flags to Include More Complex Features

Certain flags are used to incorporate some of the more sophisticated features of the software. These global features are shown in the table below:

flag	*meaning*	*comment*
`MAXINC`	the number of time is allowed to increase in the Levenberg-Marquardt scheme	see file **.inf*
`MAXIT`	the number of iterations	frequently used

9.5 Error Messages

This section gives a list of error messages and actions to take.

Error file *lf90.eer* missing - copy *lf90.eer* from the Utility directory to your current working directory, or into a directory which is listed in your PATH variable.

Chapter 10

Setting Up a Typical User Run in a Scenario Directory

To illustrate how `WD2006` works, the main steps describing the set up for a `WD2006` run will be demonstrated. To run `WD2006`, the following steps have to be done:

- A new scenario directory has to be created. It is recommended to do this in a directory named STARS. The subdirectory in which the data for a particular star is kept might be named NEWSTARNAME, for example. This subdirectory has to contain several critical files. Of course, once created, they may be copied from an existing star subdirectory, or from, say, \STARS\NEWSTARNAME; under LINUX this might be /STARS/NEWSTARNAME.
- First, the eclipsing binary scenario needs to be described via input parameters in the **.dci* file. Based on such file, in the first run `WD2006` can generate a corresponding **.lci* template which the user might want to modify slightly.
- The input file `newstarname.inf` tells `WD2006` what it has to do. In particular, the *inf* file contains the names of all input and output file names. The user also defines in this file the tasks that `WD2006` has to carry out. A detailed explanation of the tasks can be found in Section 4. Further discussion of `WD2006` input files can be found in Section 3.
- Having set up the optimization and defined the desired tasks that `WD2006` has to do, the optimization run can be started. This is carried out by typing the command `r name`. After this command has been used once within a particular directory, it is sufficient just to type `r` because the `name` is memorized. The run terminates if: a simulation is carried out; an optimal set of parameters is reached; the maximum number of iterations is exceeded; or if the run is stopped by the user. Each run which was stopped can be restarted. This allows a user to interrupt long optimization runs without losing previously computed results.
- During and after an optimization run it is possible to plot the light curve fit. After each plot, several PostScript files are generated displying the various observed and computed light curves and the residuals plotted against phase.

Chapter 11

Sample Cases and Tests

WD2006 comes with a set of ready-to-run sample files. Currently, the following example scenarios are provided:

name	description
tst-01	simple test routine to check the optimization algorithms
tst-02	simple test routine to check the optimization algorithms
tst-03	simple test routine to check the optimization algorithms
tst-sa	testing simulated annealing

This scenario is described in detail in the following sections.

11.1 Example 1: *tst-01*

This test set checks the proper installation of WD2006 the checks the simplex method, Levenberg-Marquardt algorithm and graphics. Just type *r*, and when the job is finished type *compare* and inspect the *diff.dat* file.

11.2 Example 4: tst-sa

This test set checks the proper installation of WD2006 and simulated annealing in particular. This test needs further description ...

Just type *r*, and when the job is finished type *compare* and inspect the *diff.dat* file.

Appendix A

Keywords

WD2006 uses a substantial number of keywords which are used to control parameters to the program. The are listed in the table below. All those keywords and there associated values have default values indicated in brackets []; if the control parameter, *e.g.*, IASELECT is set to IASELECT= 1 then those default values are overwritten by the value specified in the *.*inf* inputfile. Note that there is no need to have the keywords in the inputfile. Some of the parameters are indication parameters, *i.e.*, they force something to happen if they are set to 1; nothing will happen if they are set to 0. Others might be restricted to a certain range. Below, the keywords follow in alphabetic order.

- `IASELECT` is used to control which function should be minimized when using simulated annealing (task 7). [2];

  1. `IASELECT`= 0 using Judge's test function specified in Goffe *et al.* (1994) to check simulated annealing with the pre-set parameters for that example;
  2. `IASELECT`= 1 using Judge's test function to check simulated annealing with the parameters set in the *.*inf* file;
  3. `IASELECT`= 2 invokes the minimization of the standard deviation of the light curve fit the parameters set in the *.*inf* file;

- `IAMAXEVL` - The maximum number of function evaluations. [10000];

- `IANS` - the number of cycles, N^{C}. After $N^{\mathrm{C}}n$ function evaluations (n is the number of parameters to be adjusted), each element of the step-size is adjusted so that approximately half of all function evaluations are accepted. [20];

- `IANT` - the number, N^{T}, of iterations before temperature reduction. After $N^{\mathrm{T}}N^{\mathrm{C}}n$ function evaluations, the temperature, T, is changed by the factor, R^{T}. The value suggested by Corona *et al.* (1987,[3]) is $\max(100, 5n)$. See Goffe *et al.* (1994) for further advice.

- `SAEPS` - the error tolerance, ε, for termination. If the final function values from the last neps temperatures differ from the corresponding value at the current temperature by less than ε and the final function value at the current temperature differs from the current optimal function value by less than ε, execution terminates. [0.005];

- `SART` - the temperature reduction factor, R^{T}. The value suggested by Corana *et al.* (1987,[3]) is 0.85. See Goffe *et al.* (1994) for more advice. [0.85];

Appendix B

Creating Flux Ratio Files for a Filter

B.1 Tabulate Filter Transmission Function

Create a file of wavelength versus transmission for the filter. (All the lines are in open format):

```
line 1         FACTOR, 1, WLBEGIN (nm), WLEND (nm), NW
line 2         WLBEGIN, TRANSMISSION (WLBEGIN)
...
last line      WLEND, TRANSMISSION (WLEND)
```

FACTOR is the factor to multiply the wavelengths by to convert them to nm, *i.e.*, WLBEGIN*FACTOR = WLBEGIN (nm). The wavelengths (in nm) must correspond to those in the file FPHEAD. NW is the number of wavelengths listed.

```
1.0 1 485.0 743.0 130
 485.00  0.0000
 487.00  0.0032
...
 743.00  0.0000
```

B.2 Run Program FLUX2000

This program prompts the user for the name of the Kurucz flux file (e.g., e:\cdrom13
fluxes\fm05k2.pck), the name of the filter transmission file, (*e.g.*, Rtrans), and the name of the output file (*e.g.*, Rm05).?

Here are the extra lines the header that must be added to the top of the LC or DC input to run it with the revised Kurucz atmospheres program:

```
line 1: 1 for LC, 2 for DC
line 2: g(star1), g(star2)
line 3: flux file for first filter listed
line 4: flux file for second filter listed
...
last line: END
```

Example of lines to be added to DC input:

```
2
5.0 5.0
Uflux
Bflux
Vflux
END
```

Below you will find the program listing.

```
      program read2000
      character*2 a2
      character*18 a18
      character*22 a22
      character*65 a65
      character*2 char(35)
      DATA CHAR/
     &8*'  ',' A',' E',' W','F1','F2','PH','VG',' I',
     &'G1','G2','T1','T2','A1','A2','P1','P2',' Q',
     &'T0',' P','DP','DW','  ','L1',
     &'L2','X1','X2','L3'/
      imax=20
      open(4,file='parm')
      open(7,file='pout')
    9 write(7,35)
      do 6 n=1,2
    1 read(4,2,end=10) a2,a18,a22
    2 format(a2,25x,a18,18x,a22)
      if(a2.ne.'No') go to 1
      if(n.eq.1) write(7,25) a2,a18,a22
   25 format(a2,1x,a18,1x,a22)
    5 read(4,3) i,a18,a22
    3 format(i2,25x,a18,18x,a22)
      if(i.eq.0) go to 6
      if(n.eq.1) write(7,25) char(i),a18,a22
      go to 5
    6 write(7,35)
```

```
 35 format(1x)
    do 8 n=1,imax
    read(4,7) a65
  7 format(a65)
  8 write(7,7) a65
    imax=6
    go to 9
 10 close(4)
    close(7)
    stop
    end
```

Bibliography

[1] E. H. L. Aarts and J. H. M. Korst. *Simulated Annealing and Boltzmann Machines.* Wiley, Chichester, UK, 1989.

[2] E. H. L. Aarts, J. H. M. Korst, and P. J. M. van Laarhoven. Simulated Annealing. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 91–120. Wiley, Chichester, UK, 1997.

[3] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing Multimodel Functions of Continuous Variables with the 'Simulated Annealing Algorithm'. *ACM Transactions on Mathematical Software*, 13:262–280, 1987.

[4] R. W. Eglese. Simulated Annealing: A Tool for Operational Research. *European Journal of Operational Research*, 46:271–281, 1990.

[5] G. George, W. E. Griffiths, R. Carter Hill, Helmut L¨tkepohl, and Tsoung-Chao Lee. *The Theory and Practice of Econometrics.* Wiley, New York, UK, 2nd edition, 1985.

[6] William L. Goffe, Gary D. Ferrier, and John Rogers. Global Optimization of Statistical Functions with Simulated Annealing. *Journal of Econometrics*, 60:65–99, 1994.

[7] J. Kallrath and A. P. Linnell. A New Method to Optimize Parameters in Solutions of Eclipsing Binary Light Curves. *Astrophys. J.*, 313:346–357, 1987.

[8] J. Kallrath and Eugene F. Milone. *Eclipsing Binary Stars: Modeling and Analysis.* Springer, New York, 384 pages, 1999.

[9] J. Kallrath, Eugene F. Milone, Dirk Terrell, and Andrew T. Young. Recent Improvements to a Version of the Wilson-Devinney Program. *Astrophys. J.*, 508:308–313, 1998.

[10] Julia Kallrath. *Online Storage Systems and Transportation Problems with Applications: Optimization Models and Mathematical Solutions.* Springer, New York, 2005.

[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983.

[12] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *Comp. J.*, 7:308–313, 1965.

Index