PyWD2015-Qt5

User Manual

Ozan GÜZEL, Orkun ÖZDARCAN

August 16, 2022

Contents

1 Introduction

Most researchers, who are interested in modelling of observational data of eclipsing binaries, would precisely know the Wilson – Devinney (WD) eclipsing binary modelling code (Wilson & Devinney, 1971), which becomes a standard model for detailed analysis of various types of eclipsing binaries. PyWD2015 is, at its core, a "GUI wrapper" for WD code, which aims to provide a convenient interface for inputting parameters to the user and running DC and LC programs (Güzel & Özdarcan, 2020).

Along with use of PyWD2015, users should properly cite relevant Wilson – Devinney papers (Wilson, 1979, 1990; Van Hamme & Wilson, 2007; Wilson, 2008; Wilson & Van Hamme, 2014).

The first version of PyWD2015 was written in $Python2.7$ and $Qt4$ framework was used with $PyQt4$ bindings for the interface. Shortly after, serious bug fixes were made and GUI part was rewritten in $Qt5$ environment with $PyQt5$ bindings for the interface. Since support for $Python2.x$ ended, code parts were also rewritten and now PyWD2015 is $Python3.x$ compatible. PyWD2015 uses NUMPY and SCIPY libraries for numerical calculations and MATPLOTLIB to display plots and graphs and save publication quality figures.

2 Installing and running PyWD2015

After a seamless installation of PyWD2015, users are advised to take a look at Section 7 occasionally, where some hints about proper running of PyWD2015 and solutions against possible issues are provided.

2.1 Single executable file option

PyWD2015 was successfully run in Linux (64 bit) and Windows (Win 7, 8.x, 10; all 64 bit) operating systems. Users, who work on Windows systems have a very practical option for using PyWD2015. These users can download single executable file "pywd2015.exe" and run it directly by double clicking on it. Note that Windows executable file is only available for 64 bit systems. However, PyWD2015 cannot operate without LC and DC programs, which are main programs of the Wilson – Devinney code. Users must put these programs and their auxiliary files[1] to a proper working directory. This directory is used by PyWD2015 during runs of DC and LC programs (see later paragraphs for more information). Single executable binary file option is not available for Linux systems since created binary files usually complain from various reasons and may likely fail to run in different Linux distributions.

When users successfully run PyWD2015 executable file, it may take a while until the welcome screen (Figure 1) appears, because the executable file extracts all required libraries to a temporary folder and run PyWD2015 from that folder. When the welcome screen appears, it asks user to choose LC and DC programs from desired working directory. After users choose these files, main window (Figure 2) of PyWD2015 opens and users can start to work with PyWD2015.

[1] These are "atmcof.dat", "atmcofplanck.dat", "effwvl.dat" and "limcof_bp*" files.

When users close PyWD2015, all files, which are extracted to the temporary folder are removed. It means that if users re-run PyWD2015 again, welcome screen will appear again and users will have to choose LC and DC programs from desired working directory again. The same is valid when unexpected crash and/or termination of PyWD2015 occurs. For that reason, frequent save of currently studied PyWD2015 project is strongly recommended.

2.2 Running from the archive

Users, who desire to run PyWD2015 from the source code, should consider that the installation of PyWD2015 mainly consists of two steps: Downloading PyWD2015 archive file and installation of required libraries. The total size of the archive is ~ 2 MB. Main requirements of PyWD2015 are NUMPY, SCIPY, MATPLOTLIB and $PyQt5$ libraries.

Windows users must previously install PYTHON[2]. Here, it is strongly recommended to install the very latest version of PYTHON available for used operating system since older versions may lead to unexpected dll errors when importing PYTHON libraries.

NUMPY, SCIPY and MATPLOTLIB libraries can be installed relatively straightforward using Python's "pip" command line utility. Issuing:

>sudo pip install numpy scipy matplotlib

or, on Windows with an elevated command prompt:

>pip install numpy scipy matplotlib

should install these libraries.

On Windows, "pip" should be automatically installed alongside $Python2.7.9$ or greater. On Linux, one should refer to his or her distribution package manager for installing "pip".

The recent version of PyWD2015 also requires $PyQt5$ library. This library does not exist in "pip" for Linux systems, so it must be installed externally. Most Linux distributions have it on their package managers. Below are some examples:

Ubuntu/Debian:

(For Python 2.7) >sudo apt install python-pyqt5

(For Python 3.x) >sudo apt install python3-pyqt5

For Windows 7 and 8.x, use of "pip" is possible within an elevated command prompt:

>python -m pip install python-qt5

However, for the latest versions of PYTHON, users may have to modify the command as shown below:

>python -m pip install PyQT5

If Windows users encounters dll import error, it is likely that an older PYTHON version is in use. In this case, it is recommended to uninstall older version and install a newer (the latest version available is strongly recommended) version of PYTHON. After that step, please do not forget to install NUMPY, SCIPY and MATPLOTLIB libraries as described above.

Note that PyWD2015 has not been tested on Windows 11 yet.

[2] https://www.python.org/

On Debian, users may encounter a "backports.functools_lru_cache" and/or "tkinter" error on a fresh matplotlib installation under $Python2.x$. To fix this, user can issue:

>sudo apt install python-backports.functools-lru-cache python-tk

In addition, files required by WD code (atmcof.dat, atmcofplanck.dat, effwvl.dat, limb darkening coefficient files, LC and DC executable files) must be placed in a directory, which is desired to be used for LC and DC computations. There should be no limitation for path and name of this directory. However, since WD code was written in FORTRAN, users must be careful about that FORTRAN codes sometimes may not work due to long path names (e.g. /home/user/directory1/directory2/.../.../.../.../PyWD2015). Therefore, "/home/user/" directory for linux systems (e.g. /home/user/PyWD2015) and a directory directly under "C" drive (e.g. C:\PyWD2015) for Windows systems might be recommended to copy PyWD2015 directory. PyWD2015 can also be copied to different drives (e.g. "D" drives, instead of "C"). Another important point is that the user must compile LC and DC programs with an appropriate FORTRAN compiler for corresponding platform and set execution permissions correctly. However, pre-compiled LC and DC binaries are provided for users who do not want to spend time on code compilation.

After all WD files are in place and required libraries and software are installed, enter PyWD2015 directory and run the program from a terminal by typing

> python3 pywd2015.py

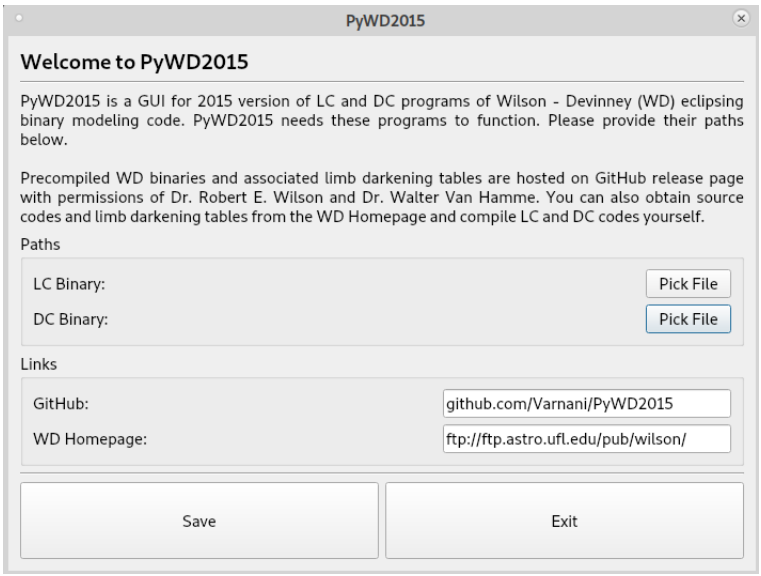

Figure 1: Welcome screen.

When everything works fine, a welcome screen appears (Figure 1). In this screen, path of LC and DC executable files must be defined by clicking "Pick File" buttons. Then, the screen is closed by pressing "save" button. After the welcome screen is closed, main window of PyWD2015 opens (Figure 2).

3 Overview of the GUI

Inspecting Figure 2, one finds five vertically arranged tabs. These tabs are "Input", "LC2015", "DC2015", "Tools" and "About". Under "Input" tab, there are four horizontally arranged tabs, "Main", "System", "Surface", "3rd body".

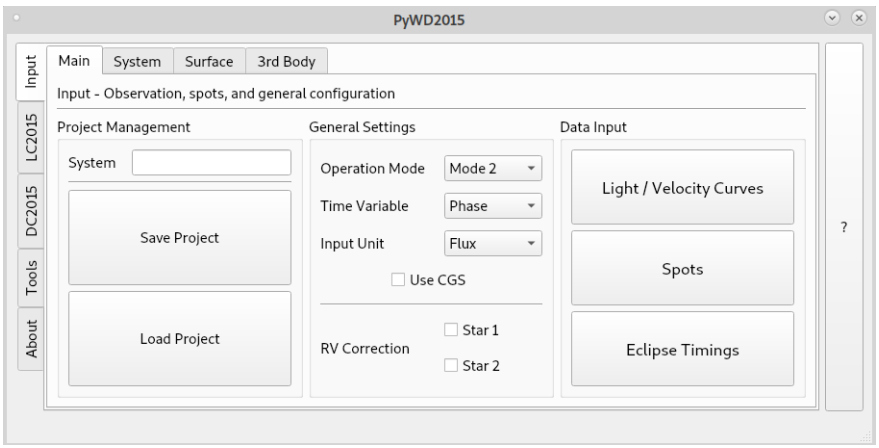

Figure 2: Main window.

The first horizontally arranged tab is "Main", and is seen in Figure 2. In this tab, system name is entered to the corresponding box (next to "System" label in Figure 2) and operation mode, independent variable (time or phase) and dependent variable (flux or magnitude) can be chosen. If flux data is in cgs unit system, "Use CGS" box must be checked. Checking "RV Correction" boxes enables proximity and eclipse corrections for radial velocity of corresponding component. This tab also includes three buttons under "Data Input" label (Light/Velocity Curves, Spots, Eclipse Timings). These buttons enable user to define observational input data and cool/hot spots on component stars. Details on these buttons are given in next sections.

The whole window has a small but useful tips for labels. If one holds mouse pointer on, say "Use CGS" label, for a second, a small text box appears, which includes explanation for that label. At the same time, abbreviation of that label ("IFCGS" in this case) in WD convention appears at the lower left bottom of the whole window. This property is also found in other tabs.

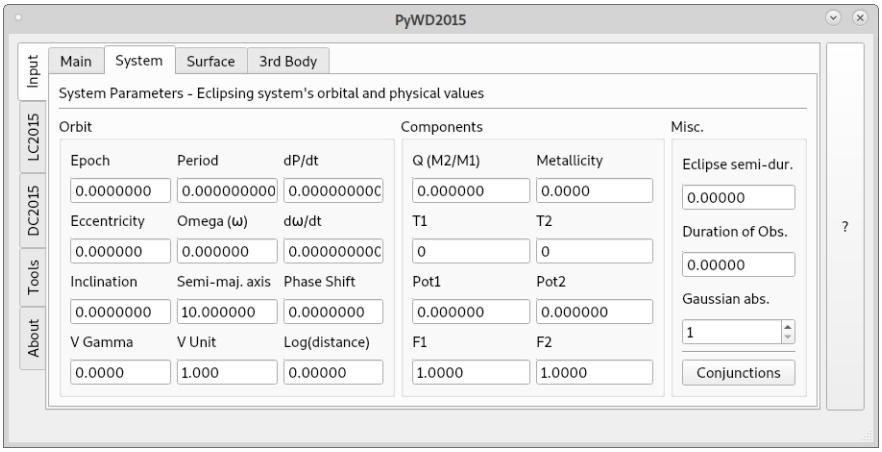

Figure 3: System tab.

The next tab is "System", where orbital and physical parameters of the eclipsing binary can be entered (Figure 3). PyWD2015 follows WD convention in terms of parameter units, except temperatures (T1 and T2), where user must supply temperatures in absolute values[3]. Angular units must be entered in radian, except inclination angle, which must

[3]If temperature of the primary component is 5 000 K, the T1 must be entered as 5 000. PyWD2015 divides this number to 10 000 and converts it to 0.5 when writing input file.

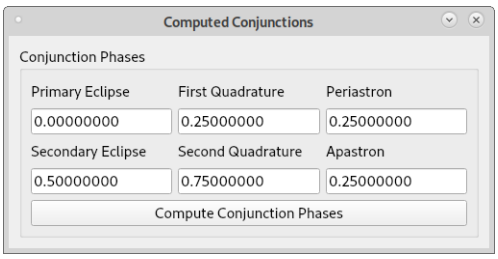

Figure 4: Conjunction information window.

be entered in degree. Pressing "Conjunctions" button located lower right corner in Figure 3 opens a small window (Figure 4), which gives information on the phase positions of the eclipses, apastron and periastron. The information provided by this window is particularly useful for eclipsing binaries with eccentric orbits.

The third horizontally arranged tab is "Surface", where radiative properties of the components are defined (Figure 5). This tab is quite self explanatory. Under "Limb Darkening" header, one must check boxes under "Set Fixed" label in order to use fixed (user defined) limb darkening coefficients. If these boxes are unchecked, PyWD2015 sets numbers, which represent limb darkening laws in LC/DC input files, to negative values in LC/DC input files[4]. Hence, LC/DC programs ignore user-defined limb darkening coefficients (if any) and compute them internally during DC or LC run. In this case, PyWD2015 removes limb darkening coefficients (x_1, x_2) from adjustable parameter list and prevents user to choose them as adjustable parameter. Please note that, internal computation of limb darkening coefficients is not valid for bolometric limb darkening coefficients. If bolometric limb darkening coefficients are strictly required (e.g. for computations with detailed reflection assumption), user must enter or update these coefficients by hand for each LC and DC run.

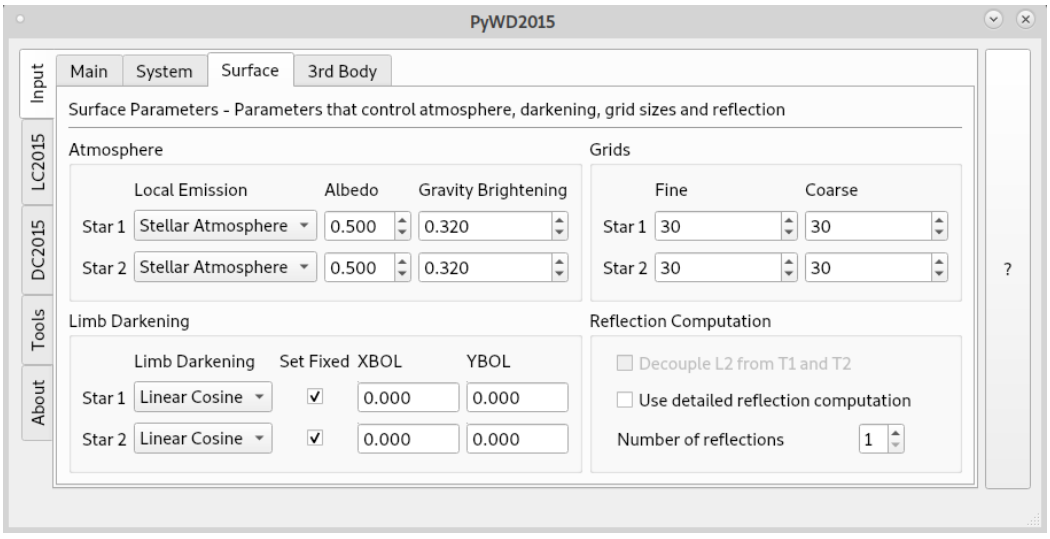

Figure 5: Surface tab.

The last horizontally tab is "3rd Body" and includes orbital parameters of the third body (Figure 6). Before entering third body parameters, "Enable third body parameters" box must be CHECKED! Otherwise, PyWD2015 locks all these boxes and does not allow users to edit. In this case, LC or DC will run by adopting third body parameters as zero.

[4] -1, -2, -3 in WD convention; linear, logarithmic and square-root laws, respectively

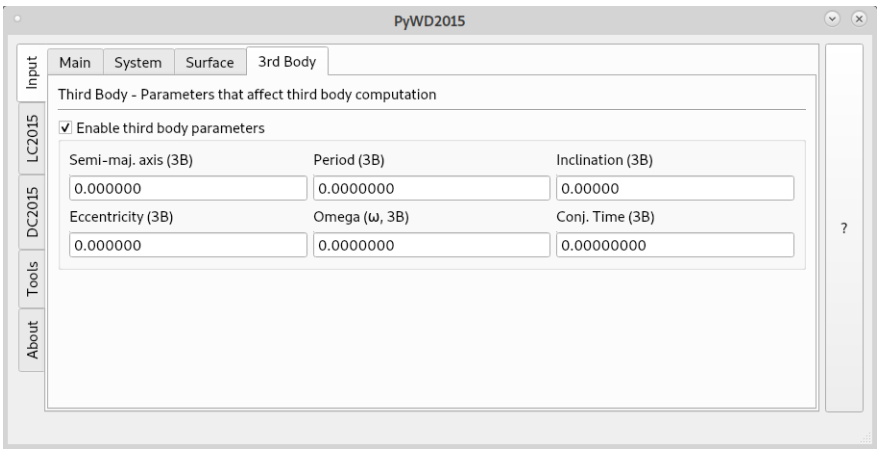

Figure 6: Third body tab.

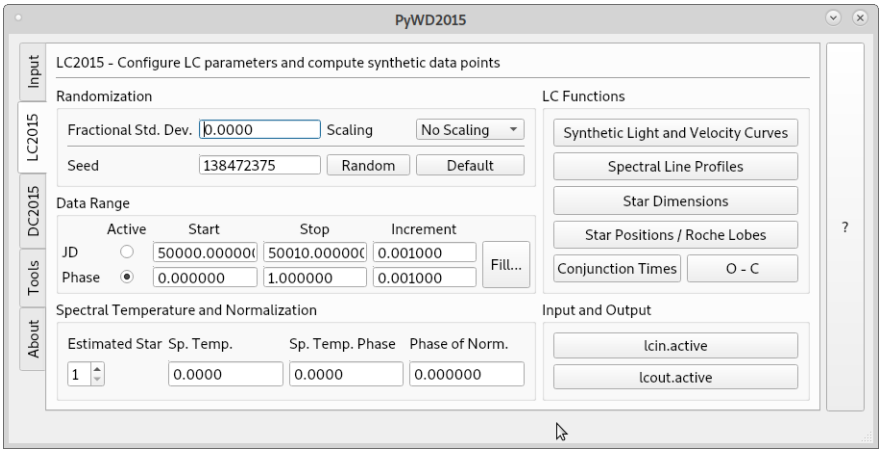

Figure 7: LC2015 tab.

The second uppermost vertically arranged tab is “LC2015” (Figure 7). In this tab, common LC parameters for constructing theoretical light and velocity curves, projected surface element of the components, spectral line profiles and conjunction times are found. Boxes and labels are self explanatory. “Random” and “Default” buttons are related to “Seed” parameter, which determines noise excursion in theoretically constructed model. Pressing “Default” button will adopt default seed number coming in original LC input files, while pressing “Random” button will change this number randomly. Pressing “Fill...” button enables user to adopt start and end point of observational data in time/phase axis automatically. However, user needs to define an input observational data previously in order to use this feature. “lcin.active” and “lcout.active” buttons open lcin and lcout files for the very last LC run. Files are opened by default text editor defined in operating system. These buttons are practical for quick view of lcin and lcout files. For each LC function, separate buttons were designed. Depending on the purpose, clicking on a proper button will open a new window.

The next vertically arranged tab is “DC2015” (Figure 8). Type of derivatives, designated extinction and its band are defined in this tab. There are two buttons on this tab. Pressing “Differential Correction” button opens a new window, which enables user to run iterations and trace outputs as lists and plots. “Solution History” button opens another plot window. In this window, user can trace the previous values of any adjusted parameter during iterations. This is particularly useful in terms of tracing the model convergence during successive DC iterations.

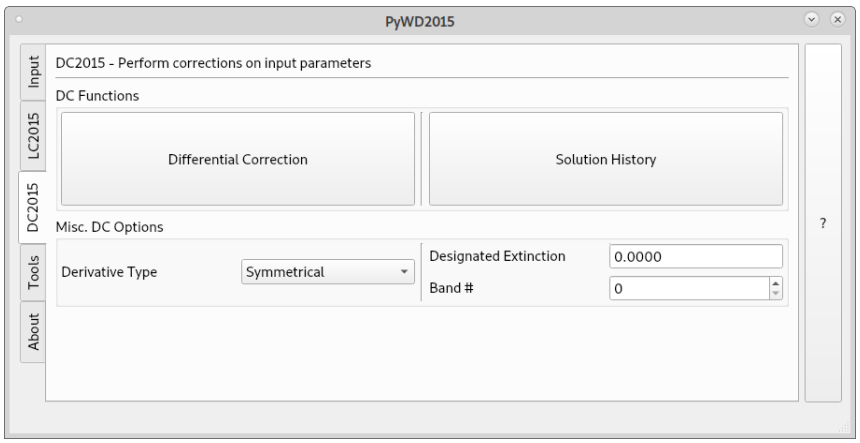

Figure 8: DC2015 tab.

The fourth vertically arranged tab is "Tools" and includes three horizontally arranged tabs (Figure 9).

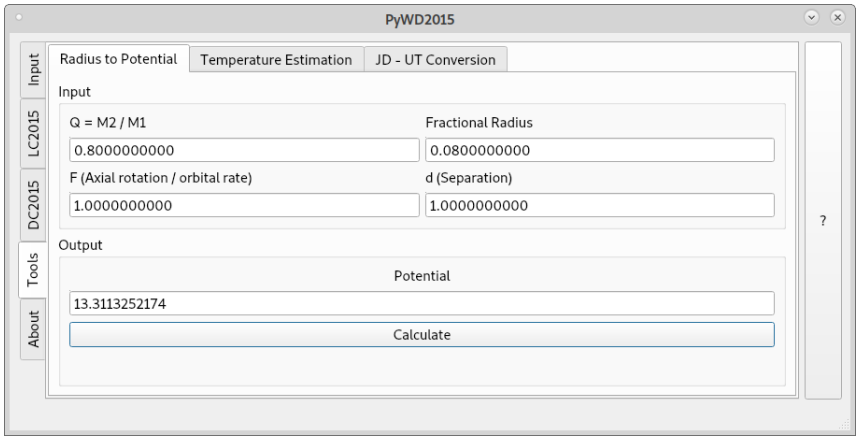

Figure 9: Dimensionless Ω potential calculation tool under "Tools" tab.

Among these tabs, the first one, "Radius to Potential", is used for calculation of dimensionless Ω potential based on mass ratio of the system, instantaneous distance between components, fractional radius and rotation parameter of the component (Figure 9). This tool is useful when estimating a reasonable Ω potential value at the beginning of a solution.

The second horizontally arranged tab is for estimating effective temperatures from color index (from $B-V$ and JHK colors; Figure 10). Numerous color – temperature calibrations were adopted from various references (Gray, 2005; Drilling & Landolt, 2000; Flower, 1996; Popper, 1980; Tokunaga, 2000). These are implemented by using 7th order polynomials (except for Gray, 2005), whom adopts 4th order polynomial for cool stars and 5th order polynomial for hot stars). User only needs to enter color index and its observational uncertainty in unit of magnitude. Then, pressing "calculate" button will give estimated effective temperature and its uncertainty.

The last horizontally arranged tab under "Tools" tab is for time conversion between UT and Julian Date (Figure 11). Both JD to UT and UT to JD conversions are possible.

The lowermost vertically arranged tab is "About", which includes some information on PyWD2015, and some theme options depending on the used operating system and desktop settings.

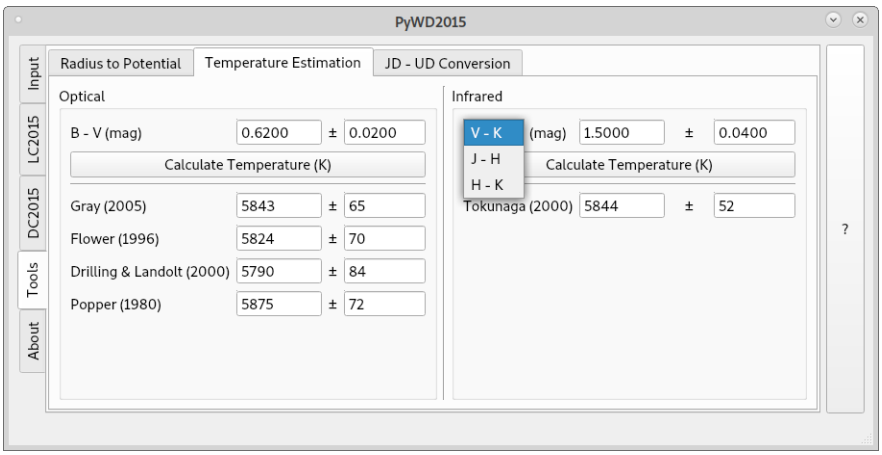

Figure 10: Effective temperature estimation tool under “Tools” tab.

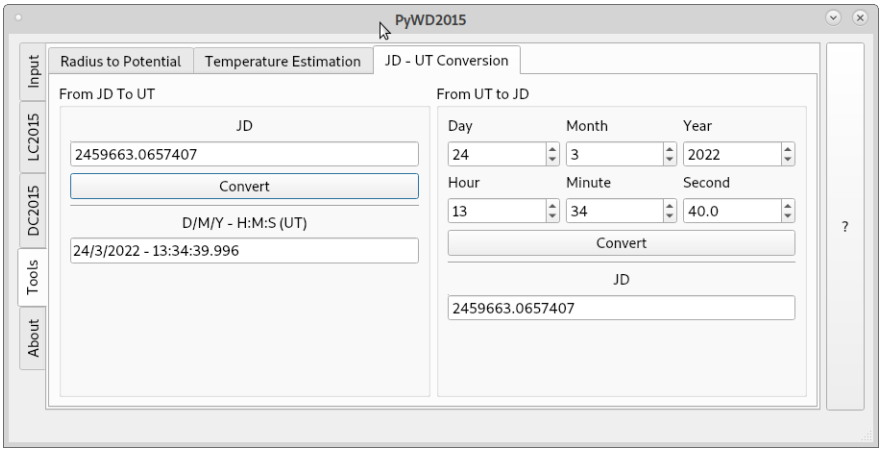

Figure 11: Time conversion tool under “Tools” tab.

4 Additional features

4.1 Eclipse times

If eclipse times are available for a system, "O–C" button under LC2015 tab can be clicked to see an $O-C$ diagram[5]. However, eclipse time data must be loaded to PyWD2015 previously in order to construct an $O-C$ diagram. For data loading, one may press "Eclipse Timings" button under "Main" tab and a new window opens (Figure 12).

Clicking "Load" button opens a file picking dialog. From this dialog, user can choose data file and load it to PyWD2015. Then all data columns are shown in "Eclipse Timings" window. If desired, "IFTIME" box can be checked. Note that, if "IFTIME" box is not checked, DC output file will not contain statistical curve information and user will not be able to see this information in DC widget (see later section). Therefore, it is strongly recommended to check "IFTIME" box in the existence of eclipse time data. "KSD" and "SIGMA" values are set to proper numbers depending on desired standard deviation apply method and data quality[6]. Particularly, user must be careful about not to leave "SIGMA" value as zero because DC program may return weird values (e.g. "NaN") or crash. When data loading finishes, user can close the window.

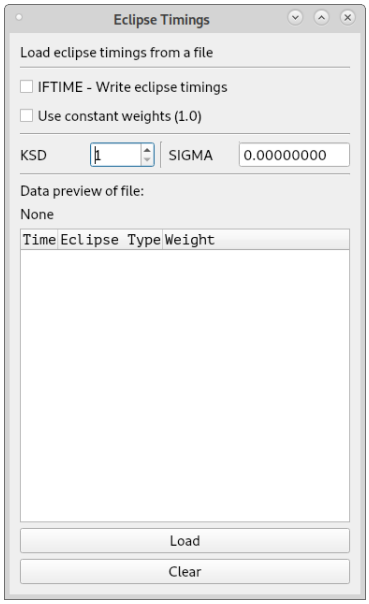

Figure 12: Eclipse timings data loading window.

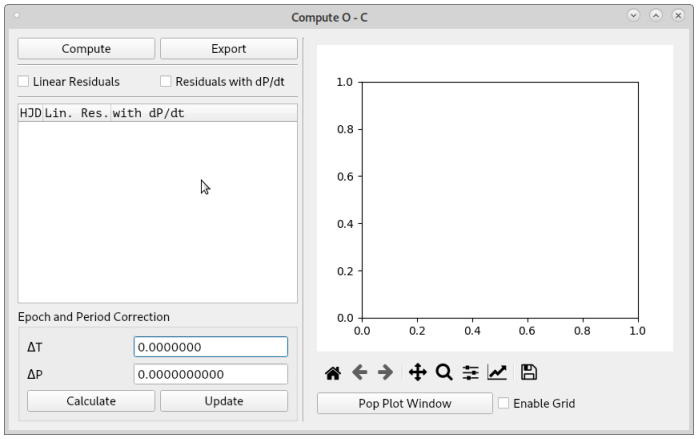

Figure 13: $O-C$ window.

The next step is to go LC2015 tab and press "O–C" button. This opens the window shown in Figure 13.

[5]Since $O-C$ concept refers to residuals from a best fitting model, it is also used in other research areas outside of astronomy. More proper concept is suggested as eclipse time variation. However, traditional $O-C$ concept is followed in this manual.

[6]Please refer to Section 7 for caveat on input KSD and SIGMA values

Pressing “Compute” button in Figure 13 lists input eclipse times, linear residuals from these times and residuals with dP/dt considered. If desired, listed data can be saved as plain text file by pressing “Export” button.

After that, user may check “Linear Residuals” box and see plotted $O - C$ data as filled circles in blue color (Figure 14, left panel). In case of dP/dt is different from zero, “Residuals with dP/dt” box can be checked. This time, residuals with dP/dt are plotted as filled circles but in red color (Figure 14, right panel). Both boxes can be checked and both residuals can be plotted at the same time, however, if dP/dt is zero, then both data sets will be identical and appear as overlapped. Please note that, after plotting residuals, if dP/dt value is updated from main window (system tab), user must press “Compute” button again in order to see actual (correct) residuals in the plot window.

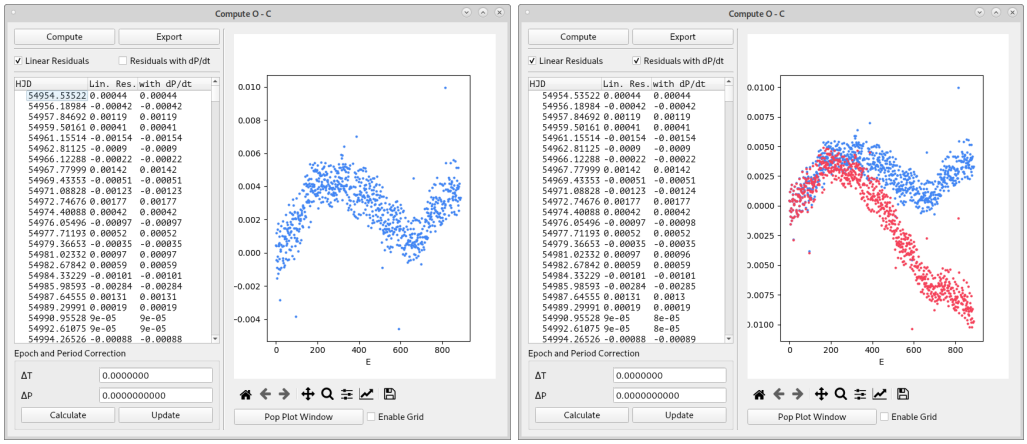

Figure 14: Linear residuals are shown in the left panel. In the right panel, residuals with dP/dt are plotted, with an assumed value of $dP/dt = 2 \times 10^{-8}$.

It is possible to compute linear corrections to the epoch and the period by fitting a line to the $O - C$ diagram shown in the left panel of Figure 14. In order to compute corrections, user must check only “Linear Residuals” box and “Residuals with dP/dt” box must be unchecked. Then, pressing “Calculate” button in lower left corner of the window, user can see linear fit as a red continuous line as overplotted in the plot window, and corrections in the epoch (ΔT) and the period (ΔP) in the boxes located lower left of the window (Figure 15).

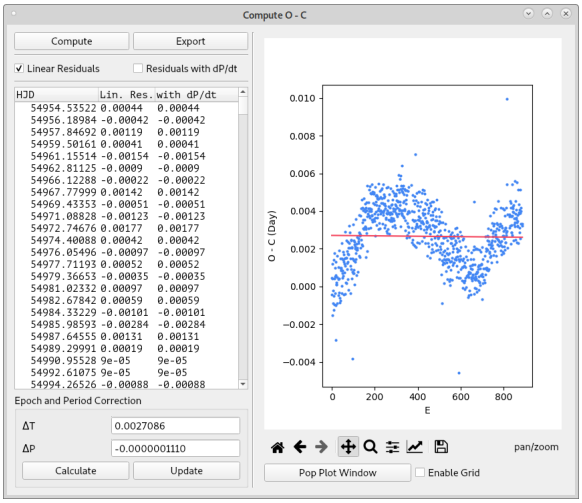

Figure 15: Computed corrections to the epoch (ΔT) and the period (ΔP) and overplotted linear fit to the “Linear Residuals”, with $dP/dt = 0$.

It is also possible to apply ΔT and ΔP corrections to the epoch and the period by pressing “Update” button. In this case, the epoch and the period values entered to boxes under “System” tab will be updated. After updating values, there is no “undo” action for this, thus user must proceed carefully.

4.2 Spectral line profiles

PyWD2015 can also be used to visualize spectral line profiles computed by LC code. Pressing "Spectral Line Profiles" button on the "LC2015" tab brings up the line profile widget. This window can be seen on Figure 16. Here, one can add spectral lines to each of the components to generate. Each list label contains tooltips and these can be viewed by simply hovering the mouse over the labels and waiting for a second. Pressing "Add" button adds a spectral line to the corresponding component's list and each parameter can be modified by single clicking. Any amount of lines can be added to either component. Specifying the phase and hitting the "Plot" button runs the LC and visualizes the results. Blue lines are used for the first component, while red is used for the second.

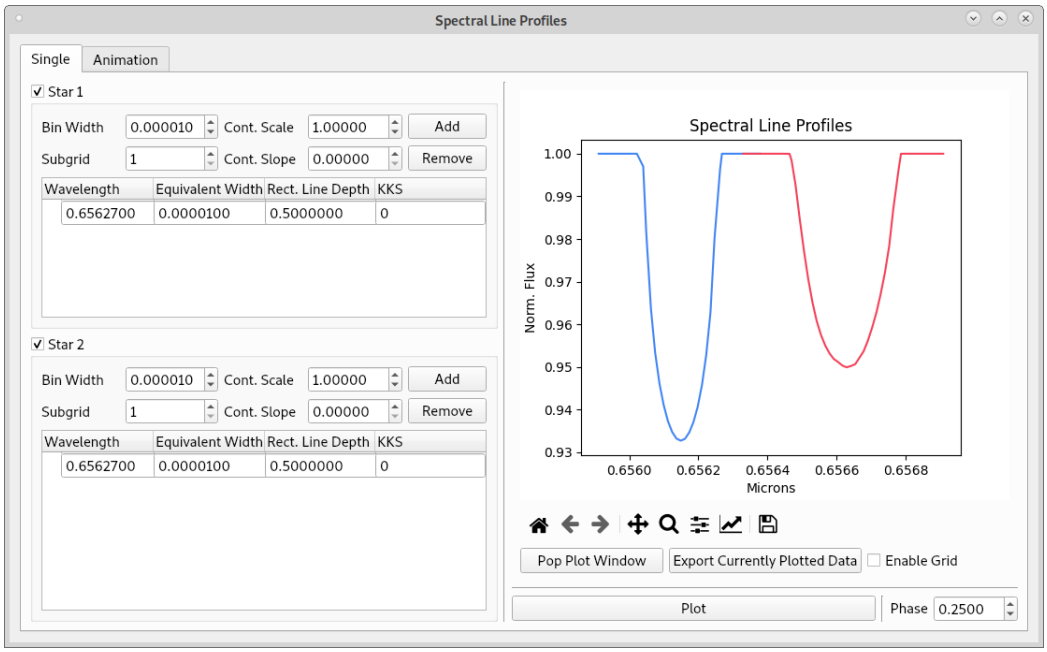

Figure 16: Spectral Line Profile widget.

4.3 Star dimensions

If interested, one can also use the "Star Dimensions" widget found on the "LC" tab to generate the fractional radius changes of the system for a whole orbital cycle. This is highly useful for eccentric binaries that are also passing closer to each other during periastron passage. Simply pressing the "Calculate Component Radii" button runs LC and the radii changes are computed. Then, one can use the tickers found on the lower part of the widget as shown in Figure 17 to show or hide each fractional radius. Each line is color coded with its own ticker.

Looking at Figure 17 upper panel, one can easily notice that for a binary on a circular orbit, no radius variation happens during an orbital cycle and pole, point, side and back radii differ only due to slightly distorted shapes of the components. Those differences arise from proximity effect. However, for binaries on eccentric orbits, radius variation over an orbital cycle might be noticeable, especially in the case of close eclipsing binaries with massive and large components. In the lower panel of Figure 17, radii difference of the components of a hypothetical eclipsing binary are shown. This hypothetical binary is composed of components with 2.3 $M_\odot$ and 1.7 $M_\odot$ masses, 1.9 $R_\odot$ and 1.6 $R_\odot$ mean

Figure 17: Star Dimensions widget with all radii shown. In the upper panel, radii of the components of a binary on a circular orbit are shown. In the lower panel, stellar radii are shown for a binary on an eccentric orbit.

radii on an eccentric orbit ($e = 0.2$) with 1.9 day period. It must be stressed that it is difficult to obtain similar graph in the case of a low mass eclipsing binary, even with the same eccentricity and orbital period. For such a low mass binary, since components would have considerably small sizes, radius variation over an orbital cycle due to the eccentricity becomes extremely small. In case of such low mass systems, user may get plots either similar to upper panel of Figure 17, or weird plots where abrupt changes in radii occurred in random orbital phases. Such weird plots possibly arise from low grid values (see grid values shown in Figure 5). Low grid values makes numerical noise dominant over extremely small radius variation.

4.4 Conjunctions

"Conjunction" widget can be used to generate conjunction times. These times then can be used as theoretical mid-eclipse times, if desired. Widget also has the ability to calculate UTC from HJD and convert HJD to JD. For that, widget requires the system's equatorial coordinates. This can be useful for observation planning. Cycle step parameter is used to skip over fixed amount of conjunctions. If it is 2, every second conjunction is computed. If it is 3, every third conjunction is computed and so on. Before computing conjunction times, user must enter desired start and stop values of in JD unit in LC2015 tab (Figure 7). The widget will compute conjunction times between start and stop JD values entered in LC2015 tab. After inputting relevant parameters, "Compute" button can be used to calculate the conjunction times. After that, computed conjunction times will be listed in the widget as shown in Figure 18. User may notice that computed conjunction times in Figure 18 are between start and stop JD value seen in Figure 7. Changing these start and stop values, user can compute expected times of conjunctions for a wider or narrower time range.

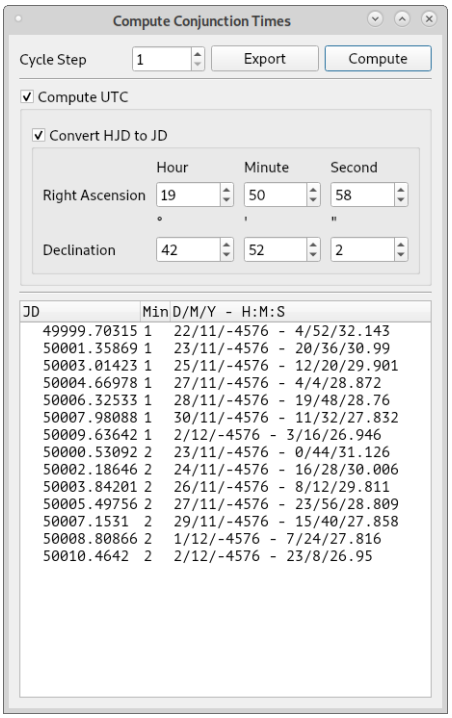

Figure 18: Conjunction widget.

5 Observational data structure

Users only need to prepare simple three column data files without any formatting restriction. PyWD2015 automatically reads these files, arranges them with respect to the format required by DC program. In the case of a light curve data, the columns are time (or phase), flux (or magnitude) and weights. If radial velocity data is considered then the columns are time (or phase), radial velocity (in km s^{-1}) and weights. Please note that, if user provides radial velocities in absolute values (in km s^{-1}), then "V Unit" box under "System" tab (see Figure 3) must be set to unity (1.0). If provided velocities are divided by 100, then "V Unit" value must be set to 100. Although this is WD related property, it would be appropriate to stress this tiny but important detail here in order to avoid weird results in simultaneous light curve and radial velocity curve solutions. Please also note that, even if input radial velocities are scaled by dividing, say 100 ("V Unit" must be set to 100 in this case!), defined plot functions for DC and LC widgets always multiply input velocities by "V Unit" and radial velocities are plotted in absolute km s^{-1}unit in plot windows[7]. If eclipse times are available, then the columns are eclipse time (JD/HJD/BJD),

[7]In this case, scaled radial velocities are still used as scaled (divided by 100 in this example) for LC and DC computations, but they are only converted to absolute km s^{-1}unit for plotting purpose, which is done by PyWD2015.

type of eclipse (1 for primary minimum, 2 for secondary minimum) and weight. After data files are prepared, they can be loaded into PyWD2015.

Although there is no strict formatting restrictions for data files, there are limitations for number of decimal digits for each defined quantity (time/phase, flux/magnitude and weight) in data files. These have already been defined in the source code of DC program. According to these limitations, DC program only considers 5 (five) digits after decimal point for time/phase quantity. If loaded data file includes time/phase quantity with, say 8 decimal digits, DC program only considers first 5 decimal digits and ignores remaining decimals. Similar limitation is valid for flux/magnitude and weights. In the case of flux/magnitude quantity, DC accepts 6 decimal digits, while for weights it only accepts 3 decimal digits.

Particularly, modelling attempts with absolute fluxes in CGS unit, (where very small numbers are involved, like 10^{-8} or 10^{-4}), loss of numerical precision is a potential issue. However, this issue can be overcome by using "XUNIT" multiplier (see Figure 21 in the next section). "XUNIT" multiplier[8] allows user to load "scaled" input data to DC program. For instance, let CGS flux of a component in an arbitrary bandpass is measured as 0.000326287. If user loads this number to DC program "as it is", DC reads this number as "0.000326" (remember limitation; 6 digits decimal for fluxes/magnitudes) thus ignores last three decimals, which causes loss of precision. If user multiplies this number by, say 1000, and load this number to DC program, DC reads this number as 0.326287 so there is no loss of precision. However, user must set "XUNIT" to 0.001 in order to tell DC that input fluxes are multiplied by 1000. After defining proper "XUNIT" value, DC program multiplies input fluxes by the value of "XUNIT", puts scaled input fluxes back to their original scale and carries out computations with true fluxes.

Please also note that, absolute flux solution can be done with light curve data in standardized magnitude unit provided that "Use CGS" box must be checked in the main window (see Section 7 for further notes).

[8]Note that "XUNIT" is not a PyWD2015 feature but it is already defined in DC code.

6 Example usage

In this section, an example on basic usage of PyWD2015 is provided. The example is based on modelling light curve and radial velocity data of a semi–detached eclipsing binary system possessing a hot spot on the surface of the primary component. In this system, secondary component filled its Roche lobe. The light curve data are phase–folded normalized fluxes, while radial velocity data are phase–folded absolute radial velocities. Assuming all observational data are ready, one can start with main window by filling "System Name" and choosing proper options (Figure 19).

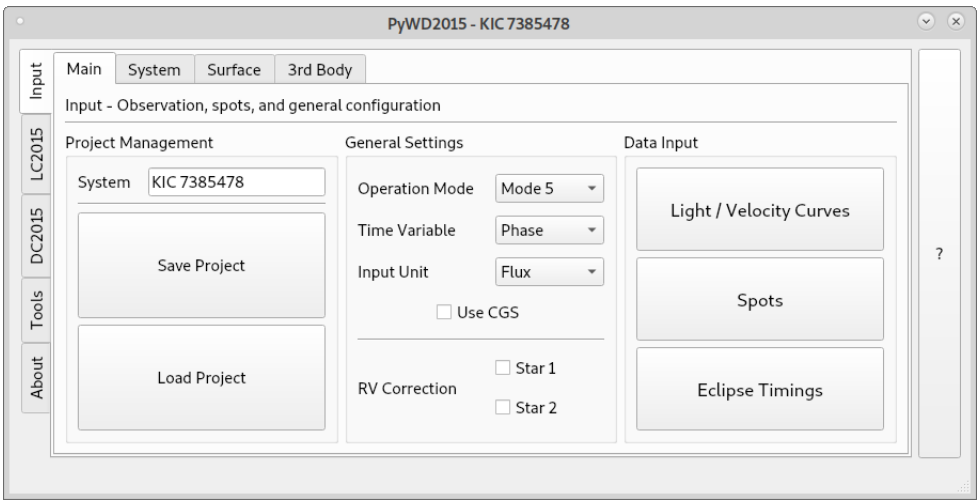

Figure 19: The filled "Main" tab.

Now, by pressing "Light/Velocity Curves" button, data files can be loaded. Pressing that button opens a new window (Figure 20). Pressing "Add" button in this window shows two options. One option is for loading radial velocity data and the other option is for loading light curve data. Choosing "Light Curve" option opens a file chooser dialog. From this dialog, one can choose light curve data file. Same process is repeated for radial velocity data but this time "Velocity Curve" option must be chosen. When mouse pointer is kept on "Velocity Curve" option, two more options appear. These options are for loading radial velocity data for the primary or the secondary component.

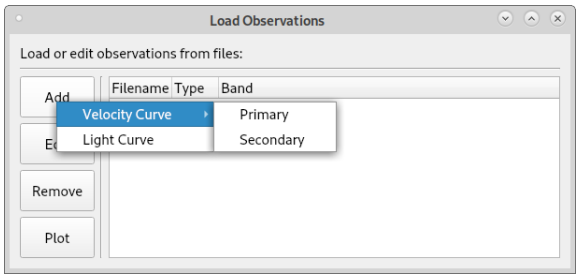

Figure 20: "Load Observation" window.

Choosing light curve option in Figure 20, a new dialog appears (Figure 21). In this dialog, each field is filled with proper numbers depending on bandpass properties of the components. If bandpass number of a filter can not be remembered, one can press "List" button and choose proper band for the corresponding filter of observed data. Then the number in the text box will be updated automatically. In case of loading a wrong file, or if user wants to load another data file instead of the current one, user only needs to press

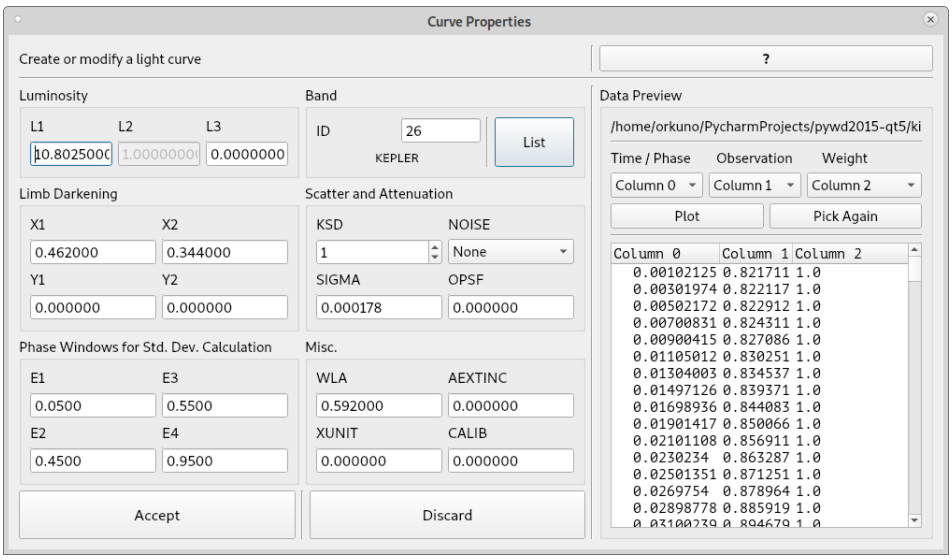

Figure 21: Light curve data loading window.

"Pick Again" button to open a file chooser dialog. In some cases, user may forget to add third column (weight column) to the data file and may load it to PyWD2015. In this kind of cases, if all data points have the same weight, then user can set third column to 1.0 automatically by pressing button under "Weight" label and choosing "Constant (1.0)" option. Actually, the buttons in this part allow user to choose different columns from a multi-column data file, which provides flexibility during input data definition step. The main aim of this feature is to choose proper columns from a multi-column file, which includes observational data only for a given bandpass.

In principle, this feature allows user to save multiple light curve (say Johnson-Bessell UBV) into a single (multi-column) file, and use that file to load observational data for each bandpass. However, this case can easily cause confusion during printing of DC outputs in DC widget (see Figure 28 below) because PyWD2015 prints bandpass dependent DC outputs (e.g. L1, $\ell 3$) under separate labels. PyWD2015 determines these labels from the name of loaded data file for corresponding bandpass. Using single file for loading data of multiple light curves in different bandpasses can easily cause confusion during evaluation of results printed by DC widget. Therefore, it is strongly recommended to use separate file for each data in separate bandpass and load these files separately to PyWD2015.

User can also plot defined data immediately by pressing "Plot" button.

If L2 is not decoupled from T2 (IPB=0 case; note that, this can be set from "Surface" tab, see Figure 5), the text box under "L2" label becomes inactive, its value appears as 1.0 and can not be edited. This is not a problem since L2 is a non-adjustable parameter in this (IPB=0) case and PyWD2015 automatically sets it to unity. If L2 is decoupled from T2, then L2 can be edited in this window. After filling all fields properly, the window can be closed by pressing "Accept" button. Same process is applied for loading light curve data in various bandpasses or radial velocity data.

After all observational data is loaded, it is possible to select any data file with left mouse click on it and plot it by pressing "Plot" button in "Load Observation" window (Figure 20). When data loading is finished, "Load Observation" window can be closed from upper right corner.

Then, the “System” tab is opened and each box is filled with proper numbers (Figure 22). In this tab, one can notice that the dimensionless (Ω) potential of the secondary component (POT2) can not be edited. This is because the operation mode is set to 5 (see Figure 19). The number seen in POT2 box is directly calculated inside PyWD2015 and corresponds to the inner critical potential value of the system. It also means that one can not adjust POT2 during DC iterations. Value of POT2 is kept fixed by PyWD2015, as long as the mass ratio is kept fixed.

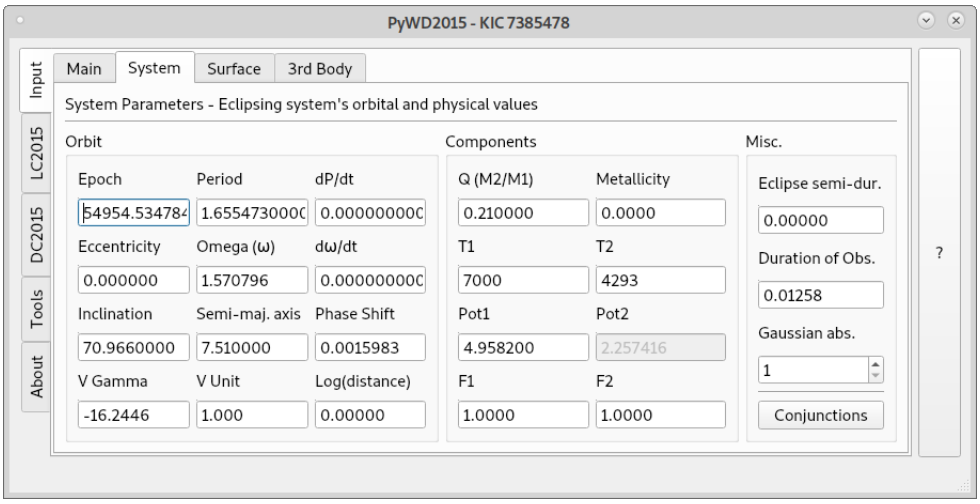

Figure 22: The filled “System” tab.

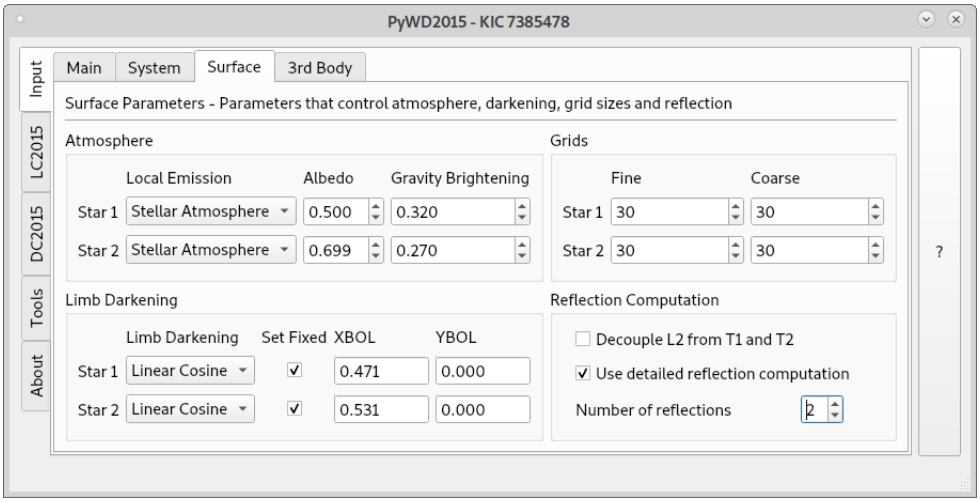

Figure 23: The filled “Surface” tab.

Next, the “Surface” tab (Figure 23) is opened and filled with proper numbers and selections, depending on estimated radiative properties of components. In this example, boxes under “Set Fixed” label are checked, which means limb darkening coefficients are not allowed to be computed internally by DC or LC. It also means that limb darkening coefficients can be adopted as adjustable parameters. If these boxes were not checked, then DC or LC would compute limb darkening coefficients internally in each iteration and it would not be possible to adjust these coefficients during DC iterations since PyWD2015 would remove them from adjustable parameter list.

In this example, there is no 3rd body parameters for the system, hence the last tab is skipped.

Since example system has a hot spot on the primary component, a hot spot should be defined on this component. For this purpose, one must press “Spots” button in the “Main” tab and open “Configure Spot” window (Figure 24).

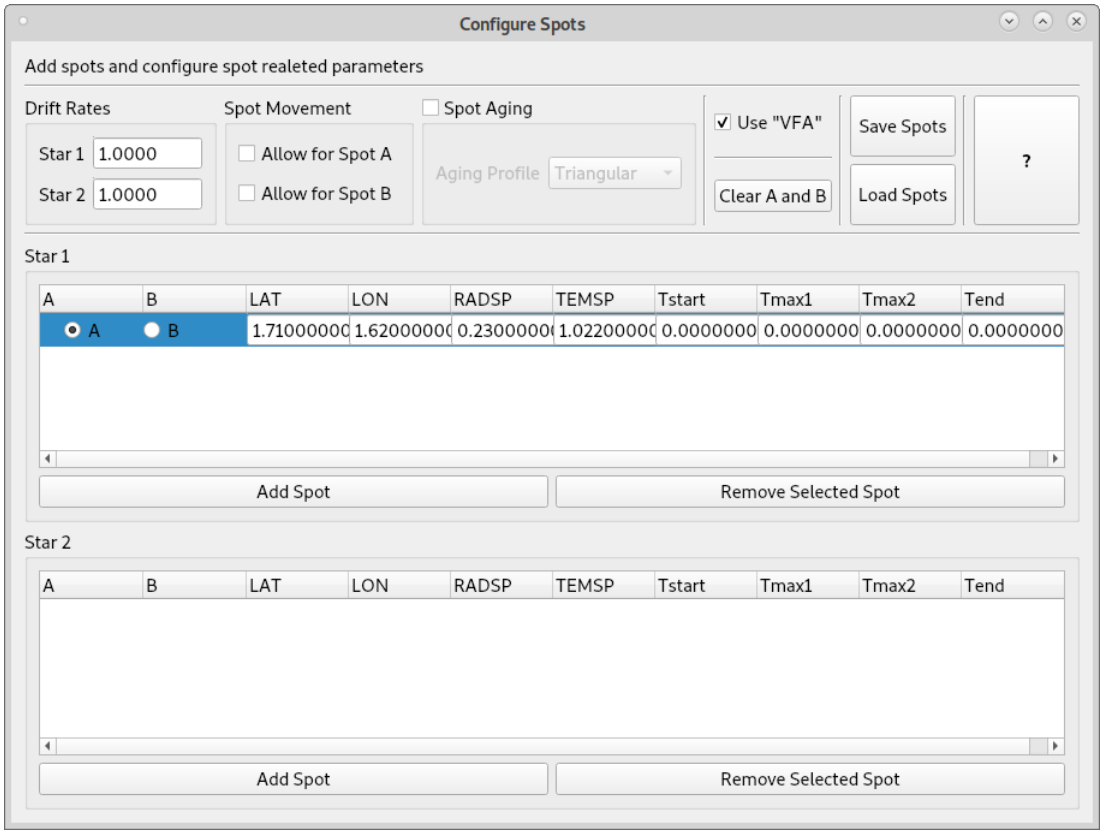

Figure 24: “Configure Spot” window.

One needs to press “Add Spot” button in order to add a spot to the surface of the desired component. In this window, a hot spot is defined on the primary component (Star 1). Since there is only one spot, “A” label was chosen for the defined spot. If there were two spots, then the second spot would have label “B”. DC code can adjust the parameters of at most two star spots in a single iteration. The labels “A” and “B” represent these two adjustable spots in this single iteration. In case of more than two spots, user can use “A” and “B” labels only for two spots at a time. Spots “A” and “B” can be on the same component or spot “A” and spot “B” can be placed on the primary and the secondary component, respectively. In case of two spots on each component (four spots in total), again, labels “A” and “B” can be used for two spots. Remaining unlabelled spots still appear in input file (dcin.active) as a separate row, contributes to synthetic model computation, but can not be adjusted in DC iterations. PyWD2015 prevents user to label more than two spots. One can notice that “Use VFA” box is checked, which means more recent and precise “Vector Fractional Area (VFA) algorithm” of DC code will be used.

The next step is to plot observational data and model curve together in order to evaluate the agreement between them. For this purpose, LC2015 tab is used. In this tab, pressing “Synthetic Light/Velocity Curves” button opens a new window (titled “Plot Synthetic Curves”, Figure 25). This window is useful when producing trial models by LC program and evaluate the agreement to the observational data. This window has “Light Curve” and “Velocity Curve” tabs.

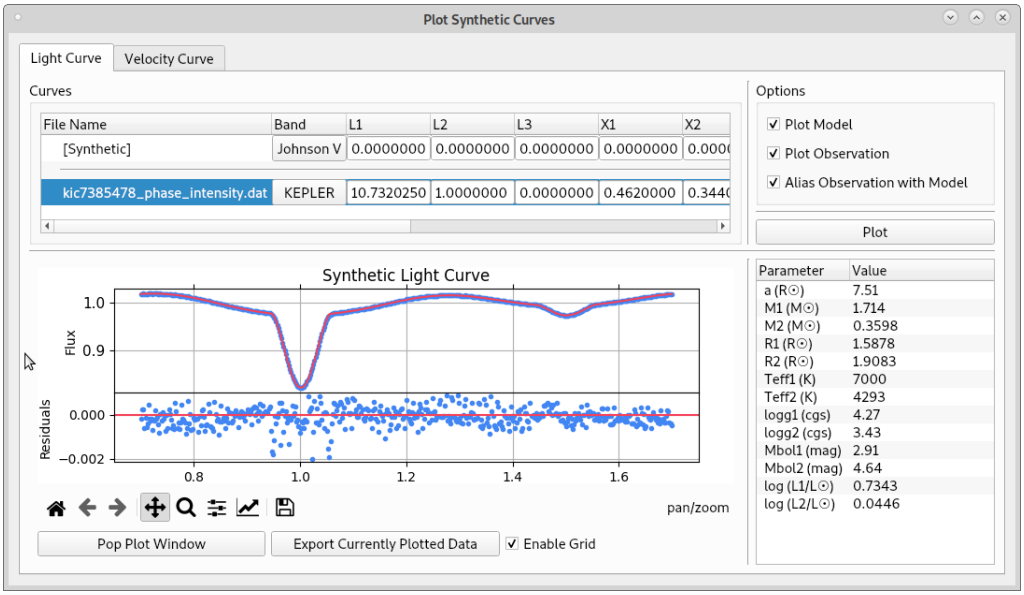

Figure 25: Light curve plotting window.

In order to plot light curve, the first step is to choose "Light Curve" tab and click the file name of desired data (under "File Name" column, see upper panel in the window). Then, desired boxes, which are located at upper right, should be checked. Here, if "Alias Observation with Model" box is checked, then PyWD2015 reads start, stop and increment values from LC2015 tab (Figure 7) and plots both observed and computed data between start and stop ranges. User must be careful about which independent variable is chosen (JD or Phase) in LC2015 tab. Choosing "JD" for phase folded input data would give weird results in plots. Depending on user needs, only observed data or only synthetic data can be plotted, too. It is also possible to put grids in plot window by checking "Enable Grid" box under the plot window. Finally, pressing "Plot" button plots desired data in this window.

"Light Curve" tab in "Plot Synthetic Curves" window includes a list, which shows absolute parameters of the components computed by the LC code, except $log(L/L_{\odot})$ values of the components. $log(L/L_{\odot})$ values are internally computed by PyWD2015 via Stefan-Boltzmann law, by adopting effective temperature of the Sun as 5780 K. These values are for giving an idea to users about the components. Furthermore, listed numbers will not be reliable in non-simultaneous solutions because semi-major axis value can be determined precisely only with simultaneous light and radial velocity solutions. It is recommended to evaluate these values separately for more precise study.

In Figure 26, radial velocity data and computed synthetic curve for each component are shown together.

One can notice that, for both light curve and radial velocity curve plots, residuals from the model are also plotted in the bottom panels. These plotted residuals in plot window are calculated by interpolating the model linearly in time (or phase) axis. These residuals are only for eye inspection purposes. For correctly calculated residuals, one must run DC (see later steps), or alternatively, time (or phase) increment value in LC2015 tab must be set to a very small value (say 0.0001) for more precise residual computation via LC code output[9].

[9]It is also recommended to set coarse/fine grid values (see Figure 5) to higher numbers in order to suppress numerical noise.

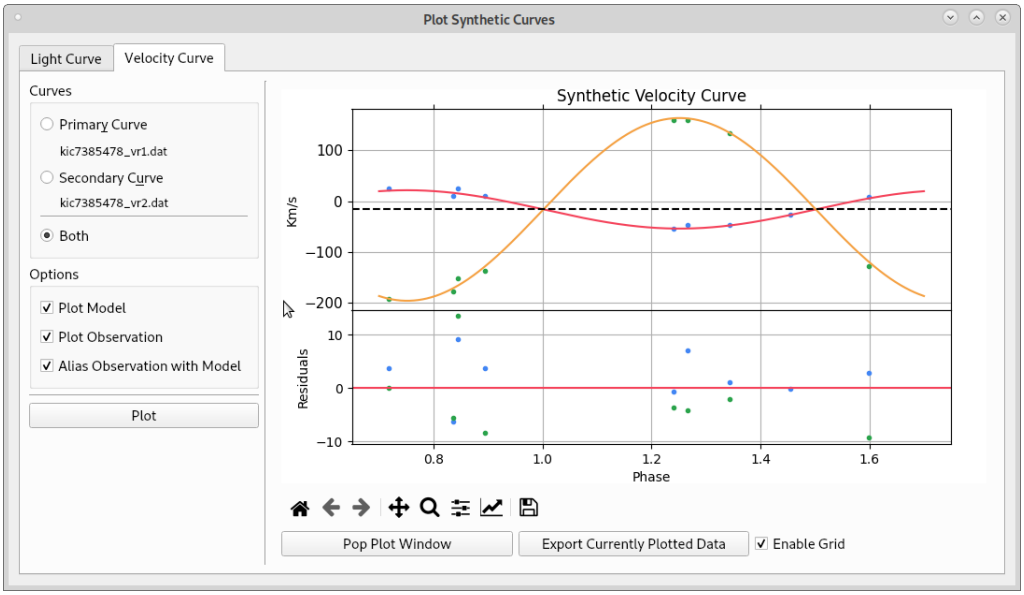

Figure 26: Radial velocity curve plotting window.

Users can easily change bandpass parameters from the upper panel. Single clicking on L1 or L2 or other values enables user to edit these numbers and change them to another value. Changing these numbers also changes the values entered in the light curve data loading window (Figure 21). Similarly, other parameters of the binary can be changed from tabs under "Input" tab. After changing values, pressing "Plot" button will update the plot. Same steps can be followed for plotting radial velocity data and model.

"Pop" button opens a separate plot window, which only shows currently plotted data and residuals. Thanks to the Matplotlib library, this window enables user to save publication quality figures in desired size and various formats. "Export" button can be used to save plotted data and residuals to a text file.

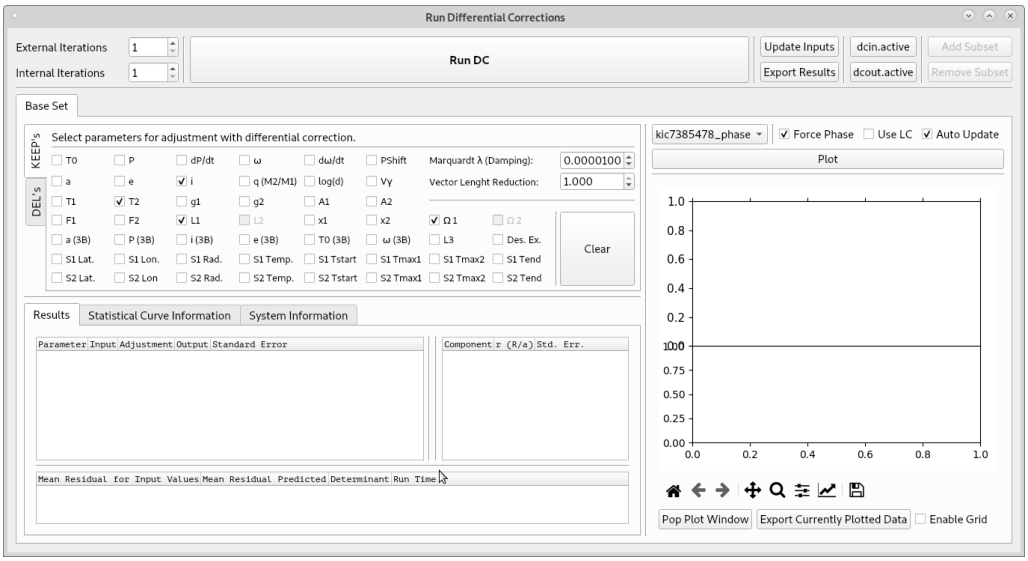

Figure 27: Differential corrections window.

Trial and error method can be applied in light/velocity curve plotting window in order to find a reasonable initial parameter set for the eclipsing binary. Then DC iterations can be started. To open differential corrections window, one must go to the DC2015

tab and press “Differential Corrections” button. This opens a relatively larger window (Figure 27).

In this window, adjustable parameters can be chosen by checking the related boxes in “KEEP’s” tab. The Marquardt multiplier (λ) and vector reduction parameter values are also defined in this tab. Under “DEL’s” tab, user can find parameter increment values, DEL’s, and change them to more proper values[10]. From the upper left part of the window, external and internal iteration numbers can be set. Here, internal iteration number is the same as $NITER$ parameter in DC program. On the other hand, external iteration number tells the number of times the GUI needs to iterate the DC solution externally. By setting external iteration to a number larger than 1, it is possible to track solution progress visually from graphs plotted in “Solution Explorer” window.

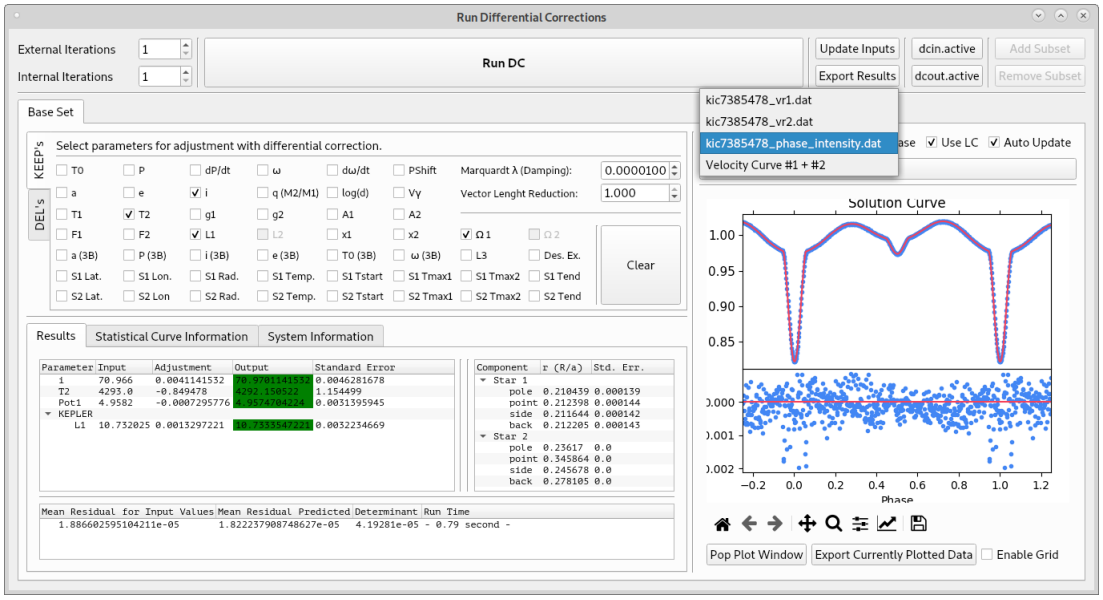

Figure 28: “Results” tab in differential corrections window.

If user sets both internal and external iterations numbers to 1 and press “Run DC” button, PyWD2015 saves all input parameters and observational data into a “dcin.active” file, runs DC with this file, gets iteration results back from resulting “dcout.active” file and shows them in the panel below KEEP’s/DEL’s tabs. Note that, during the iteration, “Run DC” label changes to “Abort (Iteration # of #)”. If this is the first DC run after opening PyWD2015, the plot window will show nothing. Here, user can choose a data file name from the combo box (just above the plot window). Then, DC widget will plot observational data from the selected file, together with the computed theoretical model (Figure 28). One can notice green colors under “output” label in “Results” tab. The green color means that the correction is smaller than the standard deviation. If the correction is larger than the standard deviation, numbers are shown in black/white. It is also possible to see fractional radii and their standard errors in this panel. The lowermost part of the panel shows mean residual for input values and predicted mean residual, together with run time of the last iteration in “second” if the iteration takes less than two minutes. If the last iteration took more than two minutes, run time value would be shown in “minute”.

[10]Common practice is to set DEL incements to 1% of the parameter value. This approach could be adopted in PyWD2015 automatically, however, automatically determined DEL increments prevents user from conscious control over adjustable parameters. Therefore, PyWD2015 does not set DEL increments automatically. As mentioned by Dr. Robert E. Wilson, users should think on each number (including DEL increments!) and set them to proper values.

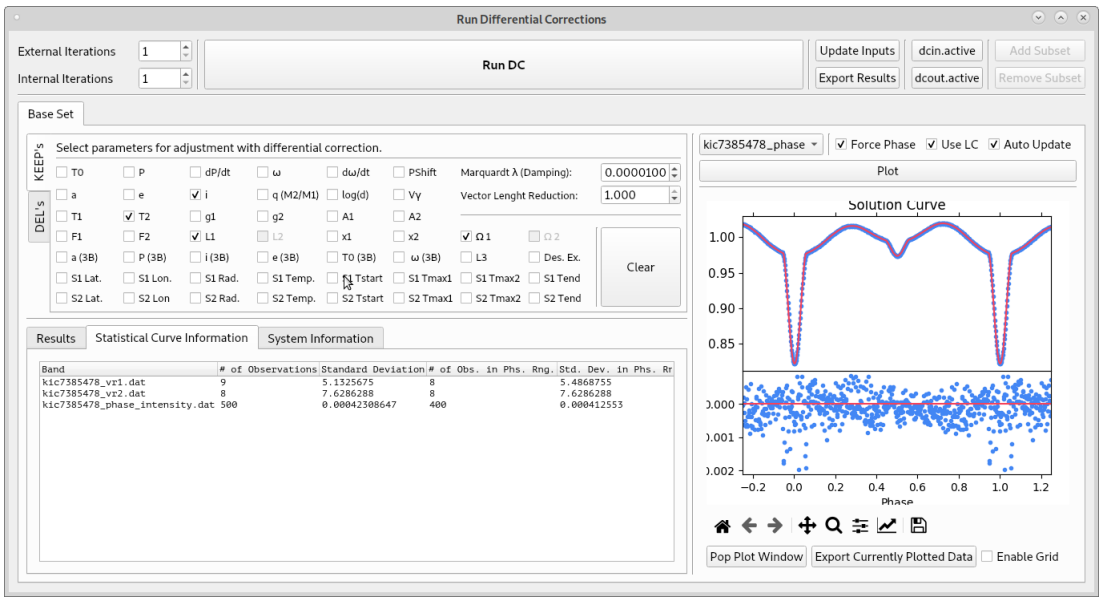

Figure 29: "Statistical Curve Information" tab in differential corrections window.

Switching to the second tab, ("Statistical Curve Information", Figure 29), user can see auxiliary data properties, which are produced by DC program. PyWD2015 reads these numbers directly from dcout file.

The third tab, "System Information" shows detailed information on the system and the components (Figure 30). Note that this tab shows information from output file of LC code, thus it is functional only if "Use LC" box is checked. If "Use LC" box is not checked then all values in this tab appear as "None" in black color on red background. Checking "Use LC" box and pressing plot button (or, alternatively running DC), user can see these values again.

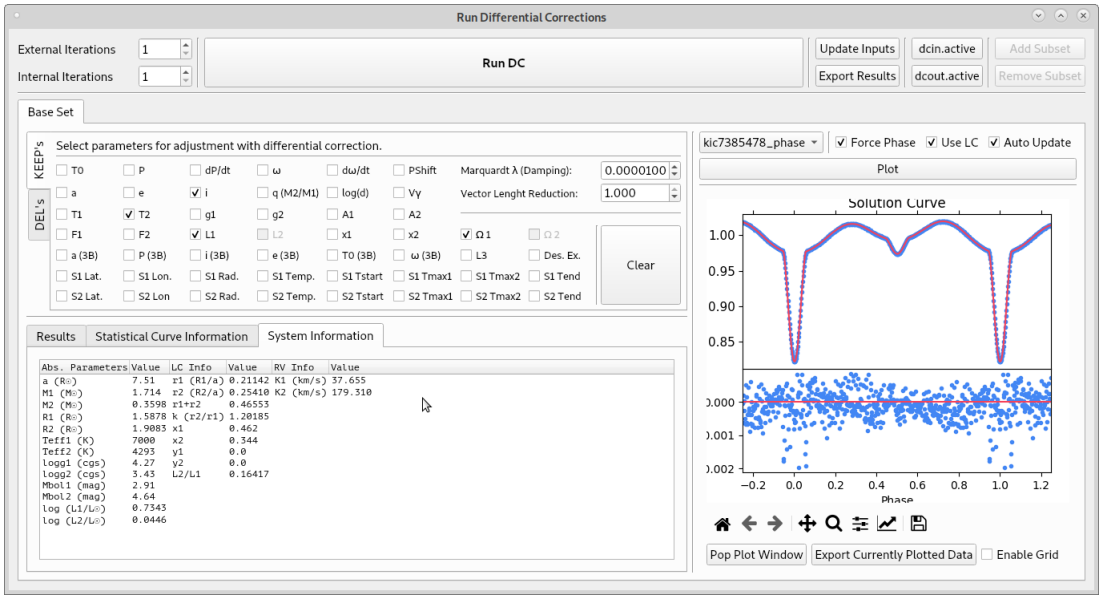

Figure 30: "System Information" tab in differential corrections window.

When plotting data and model, users can put grids on the plot window by checking "Enable Grid" box below the plot window. User can also decide to use LC program to produce model, or just using model data points computed by DC program. Checking

"Auto Update" box is useful when external iteration is set to a number larger than 1. In this case, plot window is automatically updated after each external iteration and user can trace successive iteration results instantly from plot window. Assume that the user sets external iteration to 10 and starts DC run. If the model diverge after the 4th iteration, the user can easily notice that and stop the whole process by clicking "Abort (Iteration # of #)" button. "Pop" and "Export" buttons have the same functionalities as described in light/velocity curve plotting window, but here serve for DC related data.

After a single iteration (external and internal iteration numbers are 1), if the user desires to use output values as new input for the next iteration, "Update Inputs" button can be clicked and output values of previous iteration are set as new input values for the next iteration. In this case, values of all related parameters are updated in corresponding windows ("System" tab, curve properties window, etc.)[11]. Pressing buttons "dcin.active" and "dcout.active" opens corresponding files of the very latest DC run. After reaching global solution "Export Results" button can be clicked to save iteration results as plain text or as LaTeXtable. Subset buttons are inactive since they have not been adopted yet.

After reaching a global solution, users can see star positions in yz–plane and draw the Roche geometry of the system. For this purpose "Star Positions / Roche Lobes" button under LC2015 tab is clicked. This opens a new window, which includes two tabs. "Single Plot" tab is used to plot star positions and Roche geometry for a given phase. If "Critical Roche Potentials" box is not checked and "Plot" button is pressed, PyWD2015 only draws star positions, which is the output of the LC program with MPAGE=5 option (left panel of Figure 31). If "Critical Roche Potentials" box is checked and "Plot" button is pressed, then PyWD2015 fixes the phase to 0.25 automatically, internally computes inner and outer critical Roche potentials and draws them together with the components. Here, components are drawn as shaded (right panel of Figure 31).

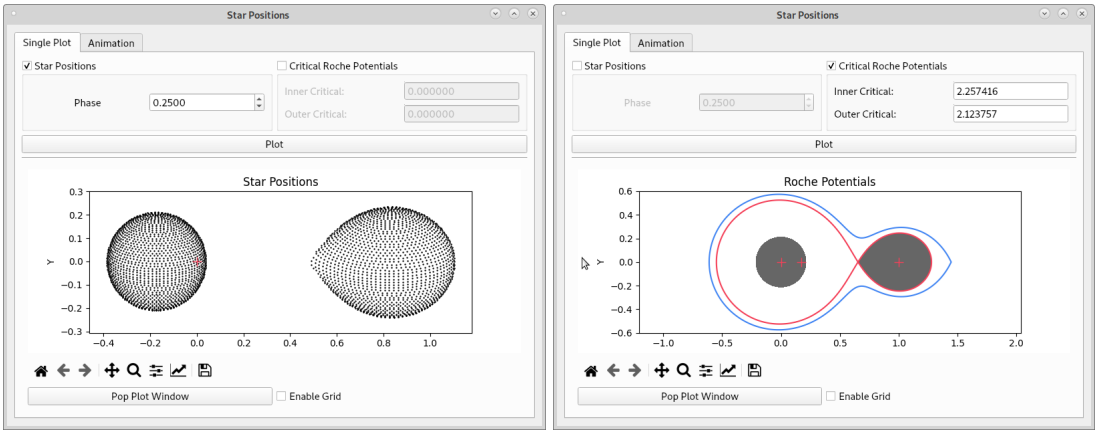

Figure 31: Left panel shows star positions for selected phase (0.25 in this case). Right panel shows Roche geometry of the system, together with inner and outer critical (dimensionless) potential values.

"Animation" tab is used to create nice animation of the system, which show positions of components through an orbital cycle. In this tab, it is also possible to draw components for a given phase, instead of creating an animation (Figure 32).

In case of successive iterations (i.e. external iteration is greater than 1) it is possible to trace variation of an adjustable parameter and its standard error through these iterations.

[11] Note that function of "Update Inputs" button has no undo action.

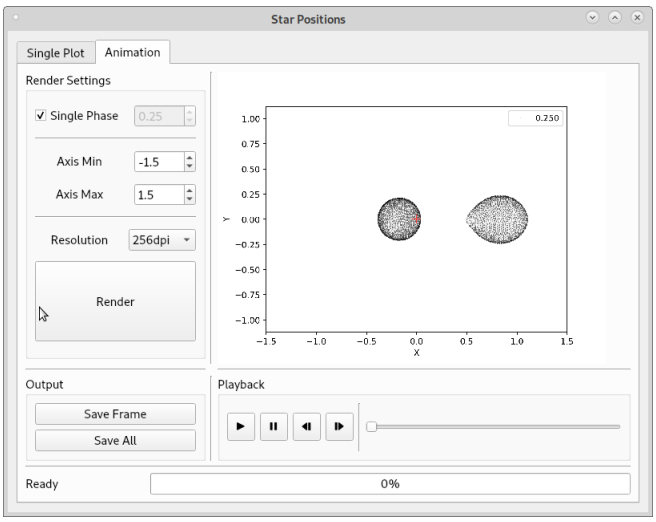

Figure 32: Animation tab.

Clicking "Solution History" in DC2015 tab opens a new window (Figure 33). When user prepares initial parameters, sets external iterations to, say 20, and runs DC, PyWD2015 keeps all successive iteration output values in memory. At the end of iterations, user can open "Solution History" window, choose an adjustable parameter by a left mouse click and press "Plot" button to observe how parameter value and its standard error change through 20 iterations. Checking "Auto" box enables plot window to be updated automatically after each iteration, so, after 20 iterations, if user wants to do 10 more iterations, output values of these additional iterations will be added to plot window automatically, after each individual iteration is completed. Although not listed in "Solution History" window, mean residual for input values and predicted mean residual values are also plotted inside plot window (lower panel). These plots help user to trace convergence of the model and if model converges to a local or global minimum in solution space. User can use "Export" button to save all data and standard errors into a plain text file.

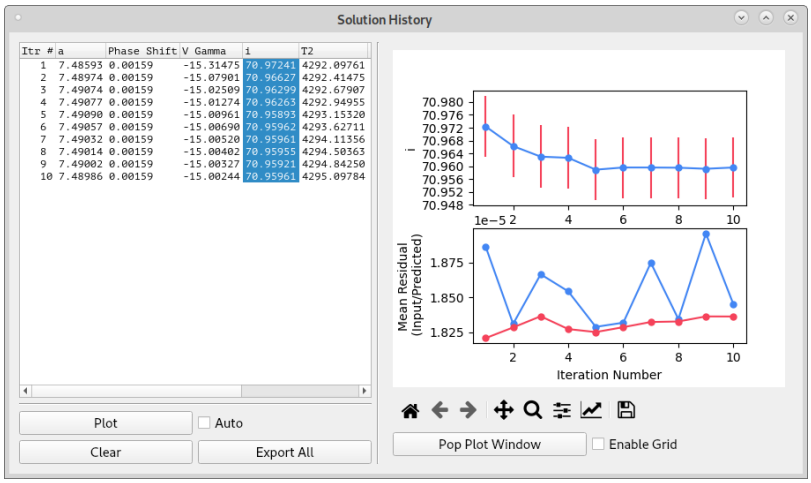

Figure 33: Solution history window with a plotted parameter.

If user wants to clear all results, "Clear" button can be clicked. Note that, after successive iterations, if user decides to keep any adjustable parameter fixed or if user adds a new parameter to the adjustable parameter list and then continues iterations, "Solution History" window will be removed from memory, hence users should take care of this property.

7 Caveats and notes

In this section, some notes and important caveats are given. These may partly be useful for avoiding unwanted mistakes. Some of them can be useful for saving time during analysis.

1. If PyWD2015 crashes and terminates unexpectedly, users may suspect a few possible reasons. One reason might be long path names (or empty space character in file/folder names), which prevents FORTRAN codes (LC/DC) from running properly. Another reason might be unphysical input values (e.g. unphysical dimensionless Ω potential value), which crashes LC/DC codes. Corrupted LC or DC executables may also cause problem. All these possibilities may cause unexpected crash and termination of PyWD2015.

2. It is strongly recommended that users should not use empty space and/or non-ascii characters when naming files and folders. For instance,

   "C:\Users\user one\Desktop\PyWD2015"

   path would very likely cause a problem when running "LC" or "DC" during a "PyWD2015" session. The problem is very likely the empty space between "user" and "one". Instead of this naming, use of "user_one" will solve the issue. Or, alternatively, users may prefer to copy PyWD2015 folder directly under "C:\", which will very likely work flawlessly. This recommendation is also valid for Linux/Unix users. However, Linux/Unix users are probably more familiar with this issue and they very likely do not use empty space and/or non-ascii characters in file/folder naming.

3. Although explained in the text above, it is appropriate to stress again that plotted radial velocities in DC and LC widgets are always shown in absolute unit (km s^{-1}) even if they are provided as "scaled" (say divided by 100) in the input files. This is done by multiplying input velocities and computed velocities by "V Unit" number entered under "System" tab in the main window. User must keep in mind that conversion to absolute unit is done on-the-fly by PyWD2015, thus there is no modification in original data files or LC/DC input/output files. However, user can extract plotted data (which are in absolute km s^{-1}unit) into a text file, as defined in previous sections.

4. In absolute flux solution procedure, input light curve data can be in standardized magnitude or in absolute cgs flux unit. If user desires to do absolute flux solution, "Use CGS" box in the main window must be checked. Note that absolute flux solution attempt with normalized flux data will result in unphysical results.

   In absolute flux solution, user must also keep in mind that the logarithm of the distance ($log(d)$) can be adjusted and bandpass luminosities cannot be adjusted. Instead of bandpass luminosities, temperatures of the components can be adjusted, depending on the number of light curve defined in different bandpasses. In non-absolute flux solutions (i.e. "Use CGS" box is unchecked, thus IFCGS=0), bandpass luminosities are adjustable while $log(d)$ parameter is not adjustable. Current version of PyWD2015 applies these restrictions in DC widget. Therefore, depending on the status of "Use CGS" box, PyWD2015 locks check box of corresponding parameter in DC widget and removes it from adjustable parameter list. For further

details on absolute flux solution strategy, please see Wilson (2008) and documentation of the 2015 version of the WD code.

5. If provided light curve data are in actual standard magnitudes, then user should provide input bandpass luminosity of the primary component (L_1) as multiplied by 4π. For example, if user somehow estimates standardized magnitude of the primary component as $8\overset{\text{m}}{.}172$ (perhaps from total eclipse phases in light curves, if any), then input bandpass luminosity is computed as

   $L_1 = 10^{-0.4\times 8.172} \times 4 \times \pi = 0.006767$

   This number can be used as an initial value for corresponding bandpass luminosity.

6. It is strongly recommended to use separate data files for separate data sets when loading data into PyWD2015. In principle, user can prepare a single data file with multiple columns (e.g. time, U mag, weight, time, B mag, weight, etc..) and load the same file to PyWD2015 for each separate light curve by choosing proper columns. Curve properties window (Figure 21) allows this strategy.

   Although this is technically possible, this way leads to a side effect when printing dcout results in DC widget. Since DC widget relies on file names to print output (adjusted) bandpass luminosities under separate (correct) labels, single-file method leads to put a single label for chosen bandpass in DC widget and prints all bandpass luminosities under the same label. If user has data in UBV bandpasses defined in a single file, output three (UBV) bandpass luminosities are printed by DC widget under U label, instead of separate U, B and V labels. In order to avoid this confusion in modelling of multiple light curves, usage of separate data files with different names is advised for each light curve data.

   However, "column choose" option in curve properties window is particularly useful to choose proper data columns from a multi-column file for data in single bandpass (see long cadence or short cadence data files at *Kepler* eclipsing binary catalogue.)

7. If user sets "Designated Extinction" parameter ("Des. Ex." label in Figure 28) as adjustable (say for V bandpass) in magnitude or CGS based modelling, then DC widget ignores input "AEXTINC" values (entered by user from curve properties window, Figure 21). Instead, DC adopts entered "Designated Extinction" value for defined "Band#" number (i.e. bandpass) in "DC2015" tab (see Figure 8) and prints this number to dcout file. If there are other light curves defined in different bandpasses, extinction values for these light curves are automatically computed by DC program via relations given in Cardelli et al. (1989), and these numbers are printed in dcout file. If user desires to use entered extinction values in curve properties window, then both "Designated Extinction" and "Band#" values must be set to zero in "DC2015" tab[12]. In addition, user must re-enter original extinction values of each light curve from curve properties window[13]. In this case, DC program does not automatically compute extinction values and it only considers user defined extinction values entered in curve properties window.

8. Maximum value of "Band#" number ("LINKEXT" parameter) is 9 in DC2015 tab. It means that only Stömgren $uvby$ and Johnson $UBVRI$ bandpasses are accepted.

[12] At the same time, "Des. Ex." box in Figure 8 must be unchecked so that DC does not include "Designated Extinction" as adjustable parameter.

[13] This can be done from synthetic curve window in Figure 25, too.

This property comes from formatting rule defined in source code of DC program[14]. Furthermore, if user works normalized fluxes (thus, IFCGS=0; "Use CGS" box is unchecked), PyWD2015 disables "Designated Extinction" and "Band#" boxes and sets their value to zero. If user switches to absolute cgs fluxes or magnitudes, then PyWD2015 enables these boxes to edit.

9. During modelling of eclipsing binaries which show apsidal motion, user must be careful during plotting of synthetic and observed data in LC widget. Common practice for plotting observed and synthetic data with respect to the orbital phase is to set start and end phases to 0.0 and 1.0, respectively. This approach ignores integer cycle number and only considers orbital phase, which may lead to shifted synthetic curves in phase axis with respect to the observed one. However, user can get correctly computed model for each observation point in dcout file (under "Unweighted Observational Equations" label), which can be plotted in DC widget by unchecking "Use LC" box. DC computes "right phases" by considering integer cycle numbers as well. In such cases, user should either work with time based data and make all plots with respect to time (then, time based synthetic curve can be transformed to phase domain by user), or provide proper start and end phase values, including correct integer numbers.

10. During DC runs, if user encounters error message shown in Figure 34, it is very likely that something is wrong with dcout file and for that reason PyWD2015 fails to read dcout file properly. In this case, it is recommended to check dcout file carefully. Sometimes, numbers printed in different columns under "Unweighted Observational Equations" label in dcout file might be printed as contiguous (without any space between them) or a number in a column might be followed by "*" signs. In such cases, current version of PyWD2015 cannot read required numbers and columns correctly and prints the error message given in Figure 34. This error also causes to remove all light and radial velocity curves from DC widget (although they are still defined inside PyWD2015; this is Qt5 related issue) thus user can not choose observed data files from DC widget any more. Re-loading current project file again can restore light and velocity curve list in DC widget but user must be careful about that project re-loading resets all widgets and plot windows, i.e. everything printed or plotted in widgets disappear. Then, user should investigate input values and adjustable parameters in order to find the reason which leads to malformed dcout file.

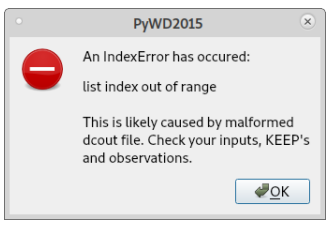

Figure 34: Error message, which indicates malformed dcout file.

11. DC program prints numbers under "Unweighted Observational Equations" label in dcout file as separate columns as long as column number does not exceed 23 under this label. This is because of the format defined in DC source code for writing

[14] Actually, DC only relies on these bandpasses in terms of interstellar extinction since extinction relations are more precisely defined for these bandpasses.

output values (see " 147 format(f13.5,22e14.5)" line in dc source code). If column number does not exceed 23, then DC program prints all computed numbers for a given observation as separate columns in a single row. However, if column number exceeds 23, then DC program does not print 24th column in the same row but it writes new value to a new line. In this case, computed values for a single observation are printed in two rows, hence PyWD2015 can not understand that this number belongs to the previous observation but it interprets that this number belongs to the next observation data point (which actually does not!). Eventually, PyWD2015 may produce weird plots in DC widget, although dcout file prints results successfully.

Column number in question increases as the number of adjustable parameter increases, thus large number of adjustable parameters may lead to this situation. In time based solutions, columns start with "time, phased, observed, computed" quantities and ends with "residual" data. Ignoring these columns, 18 columns can be printed in a single row, which means 18 adjustable parameters in principle. However, user must consider that if there are light curve data defined in two different bandpasses, then an adjustable radiative property (e.g. L_1 or $\ell 3$ or a limb darkening coefficient) occupies two columns (one for each bandpass) instead of one.

Since this is related to the writing format mentioned above, a tiny change was made in source code of DC program in order to overcome this issue. The format line was replaced to " 147 format(f13.5,40f20.10)". Readers may notice that the number of columns was increase from 22 to 40, while decimal numbers were increased from 5 to 10. With this change, DC program can print 41 columns in total in a single row. Increase in decimal number is required to obtain more precise residuals, especially in the case of modelling light curves obtained from spacecrafts (e.g. $Kepler$, $TESS$ etc.). Such light curves have unprecedented precision and enables one to investigate sub-milimag variations in light curves.

Similar case is valid for printing numbers under "Weighted Observational Equations", "Normal Equations" and "Correlation Coefficients" labels in dcout file. Corresponding format lines in the source code were changed as shown below:

37 FORMAT(1X,11F12.7) $\rightarrow$ 37 FORMAT(1X,33F12.7) (For "Normal Equations" and "Correlation Coefficients")

149 format(20e14.5) $\rightarrow$ 149 format(38f20.10) (For "Weighted Observational Equations")

Provided DC executable file at PyWD2015 website is compiled from the source code which includes tiny modifications mentioned above. However, users may compile their own binary file from the original source code given in original ftp site of 2015 version of the Wilson – Devinney code. However, they must keep in mind that reading/plotting functions defined in PyWD2015 source code files were designed with respect to the mentioned changes above.

12. In modelling attempts with absolute CGS flux data (where "Use CGS" box checked in the main window), synthetic and observed data plotted in LC widget are rescaled, thus appear different from input numbers provided in data files. This is not an issue since LC widget multiplies both observation and model data by "XUNIT" and "CALIB" values defined in curve properties window. This multiplication is automatically done by DC program internally (independently from PyWD2015), thus plotting in DC widget by unchecking "Use LC" box shows correctly scaled

data by "XUNIT" and "CALIB" values. In LC widget, this multiplication is done by PyWD2015 on-the-fly.

13. In the existence of eclipse time data, it is recommended to set KSD value to 0 for eclipse data. Setting KSD to 1 or 2 may cause to print a warning message inside dcout file, which changes usual format of statistical information printed just above "Unweighted Observational Equations" label. As a result of this format change, PyWD2015 cannot read statistical information of eclipse time data properly and fails to print it in DC widget. However, please note that setting KSD to 1 or 2 prevents PyWD2015 only from proper printing of statistical information of eclipse time data in DC widget, but does not cause incorrect output values. When setting KSD to 0, user must not forget to enter proper "SIGMA" value for eclipse time data, otherwise DC run may end with "NaN" values.

14. If "Enable third body parameters" box (Figure 6) is unchecked, then PyWD2015 locks check boxes of third body parameters in DC widget and removes them from adjustable parameter list. In order unlock these boxes, "Enable third body parameters" box must be checked and proper initial values of third body parameters must be entered. Then, PyWD2015 allows to adjust these parameters.

References

Cardelli J. A., Clayton G. C., Mathis J. S., 1989, ApJ, 345, 245

Drilling J. S., Landolt A. U., 2000, Normal Stars. p. 381

Flower P. J., 1996, ApJ, 469, 355

Gray D. F., 2005, The Observation and Analysis of Stellar Photospheres

Güzel O., Özdarcan O., 2020, Contributions of the Astronomical Observatory Skalnate Pleso, 50, 535

Popper D. M., 1980, ARA&A, 18, 115

Tokunaga A. T., 2000, Infrared Astronomy. p. 143

Van Hamme W., Wilson R. E., 2007, ApJ, 661, 1129

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

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

Wilson R. E., 2008, ApJ, 672, 575

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

Wilson R. E., Van Hamme W., 2014, ApJ, 780, 151