Transit Light Curve Modeller – Version 9.6

Short user manual

Szilárd Csizmadia
Deutsches Zentrum für Luft- und Raumfahrt,
Institut für Planetenforschung

Berlin, Rutherfordstr 2., D-12489, Germany
E-mail: szilard.csizmadia@dlr.de

Version: 2016 December 19
Updated: 2019 Sep 12
Updated: 2019 Jan 17
Updated: 2018 Jan 5
Updated: 2017 Dec 7

Transit Light Curve Modeller, abbreviated TLCM, is a software package to analyze the light curves of transiting exoplanets. It can be used:

- to model and to fit transit light curves;
- to model and to fit occultation light curves (secondary transits);
- to analyze phase curves (out-of-transit variations), including beaming effect, reflection effect and ellipsoidal variation up to first order approximation;
- to analyze full light curves, i.e. simultaneous fit of transit, occultation (secondary transit) and phase curves;
- to fit simultaneously the observed light curves and radial velocity curves (either SB1 and SB2 systems);
- fitting either circular or elliptical orbits (but no mutual perturbations between possible additional objects in the system are taken into account);
- fitting the instrumental radial velocity shifts between two or more (up to five) different spectrographs;
- fitting RV-trends up to quadratic term
- to take contamination values to the observed values into account;
- to take red-noise effect to the light curve into account;
- to subtract a parabola from the light curve (e.g. caused by baseline-variation or spot);
- simulating light curves;
- test of grazing transits.

The code fits the light curves with quadratic limb darkening law, and the limb darkening coefficients can be different for the two objects considered.

The code now runs as a callable procedure. No X-interface is needed nor appears. If you want to follow the running of the code graphically, set ScreenPlot = 1.0 in the *config_ninit.dat* file. If you set ScreenPlot = 0.0, then no any window or graphical interface appears or pops up, so you can run it on another machine, cluster or in the background.

Prerequisites:

To run it, you need GDL (I tested, it worked for me; GDL is free and in runs under Linux/Unix systems) or IDL (it is not free, but it has Linux/Unix/Windows versions). It definitely runs over IDL 7.1 or GDL 0.99. (Quite probably it runs over IDL 6.4 as well.)

You can install CygWin under Windows, and then GDL can be run under WINDOWS via CygWin as well.

Installation

You need the following files and directories:

(1) You must create a working directory which can be called whatever you wish.

The code will recognize after start automatically, that the system is Linux or Unix or Windows. You do not have to set it.

In this working directory you have to have to subdirectories: *color*, *wv*, *starsdata* and *coyote.*

The content of the directory *color* should be:

beaming_factors.dat	exo_aux_printfooter.pro	oploterror.pro
claret2004_apsMotConst.dat	exo_aux_printheader.pro	readcol.pro
exo_aux_printheaderline.pro	remchar.pro	repchr.pro
colortable.pro	exo_aux_sysvars.pro	strnumber.pro
eclipse_data.sav	exo_aux_testpaths.pro	zparcheck.pro
exo_aux_errorcheck.pro	gettok.pro	greek.pro
exo_aux_filename.pro	exo_aux_getheader.pro	kepler.sav

The subroutines in the color directory were partially written by Thomas Fruth and Thomas Pasternacki. You may acknowledge them.

The content of the directory *wv* should be:

filterredwv.pro	solveredwv.pro	wv_fn_daubechies.pro
minfuncwv.pro	waveletlike.pro	wv_test.pro

The subroutines in the wv directory were written by J. Winn and J. Carter, please acknowledge them if you publish something with TLCM.

The content of *starsdata* directory should be:

starsdata.sav

The coyote directory is maybe already installed and you put it into your IDL path earlier. Then you do not need it and then delete or comment out (by putting ; in front of the lines) the following two lines from tlcm.pro:

```
textX = textX + '+' + directory + '/coyote/:'
textWIN = textWIN + '+' + directory + '/coyote/;'
```

If this directory is not installed in your machine, then you can download it from D. W. Fanning's wonderful page:

http://www.idlcoyote.com/catalyst/howtoinstall.html

In your working directory the following files should be there:

```
config_ninit.dat
pga3.pro
tlcm96.pro
tlcm.pro
```

and one-three input files: one photometry input file (mandatory) and one or two radial velocity files, depending if you have radial velocities for the primary and the secondary objects.

For running in the background you need the *start.sh* and *model_tlcm.sh*.

Some subroutines in some of the working directory were written by Sz. Csizmadia, please acknowledge me and cite Csizmadia (2011, 2015) or the forthcoming paper if you publish something with TLCM.

Start

When you started your IDL or GDL, then type the followings to your IDL / GDL command window and carry out them:

```
.r tlcm
tlcm, directory = [your directory's name - preferably with full path – where TLCM-files are]
```

(The directory where TLCM-files will be found is not necessarily equal to your working directory.)

After that you have two choices:

Type

tlcm_fit

and press Enter for modelling, or type

tlcm_test

to create test light / RV curves.

Actually, you can start with

```
tlcm_test, init='No'
```

or you can parametrize with

```
tlcm_fit, refl = 1, approx= 0, khi_overwrite = 'Yes', ntemp=100, init='Yes'
```

(or with init = 'No').

Meanings:
khi_overwrite = 'Yes': the khi.dat file will be overwritten instead of appended.

Ntemp=100: number of trials to produce light curves with perturbed parameters within the right solution.

init: at the first run of each IDL-session you must to initalize the stellar parameter library. If you run TLCm again in the same Idl-session., no further initiakization is needed, and you can save a few seconds with init = 'No'.

approx: there is a faster, but approximate method to calculate transit light curves from lookup-tables (approx=1) but it has smaller precision and inner contacts. If you wish to calculate more precisely, then use approx = 0.

refl: if this parameter is equal to 0, no phase curve is calculated, the out-of-transit part of the light curve is flat. Other choices: 1: simple cosinusoidal phase-variation, 2: lambertian phase variation, 3: Kane-Gelino (2011) phase function is used, 4: Kopal (1959) phase function is applied.

Both tlcm_test and tlcm_fit need the some preparation which is described hereafter.

Preparation of light curve modeling:

There should be at least one **light curve file as input**. It should contain four columns:

```
independent variable      | normalized flux    | uncertainty of normalized flux    |
exposure time
```

The independent variable can be either phase or time (in days). The exposure time should be in seconds if your independent variable is time in days. If your independent variable is phase, then exposure time should be given in: exposure time (days) / period (days) * 86400.0.

If you wish to model radial velocity data simultaneously with light curve (LC) data (only RV-data cannot be modelled by TLCM), then you have to prepare one file for the primary in case SB1 (one-lined spectroscopic binary) systems or two files: one for the primary and for the secondary in case of SB2 (double-lined spectroscopic binary) systems. The RV-input files have the following format:

```
time          RV-value (km/s)          Uncertainty (km/s)          Instrument identifier
```

If you fit simultaneously LC + RV data, then LC-file should use time-variable and not phase-variable. Be aware, that both time-scales (for RV and for LC) have the same zero-point, so all time-records are in HJD or BJD etc., but not a mixture of them!

config_ninit.dat

I think most of the lines are self-explanaible. Therefore I give only short, on-line explanations.

```
; This file contains the NINIT, nmax, Tchange etc. parameter
; write your number after the = character in the next lines
; TimeResolution can be: 1, 5, 9, 17, 33.
; LineType can be: dashed or solid. Curce color can be: red, blue, green, gray3
NINIT = 1000.0                           ; The population size of GA/Harmony Search
nmax = 100000.0                          ; Chain length for SA
Tchange = 0.98                           ; Controll parameter for SA
Albedofit = 0.0                          ; do you want to fit the Albedos?
Albedo1 = 0.0                            ; value/searching box center for A1
dAlbedo1 = 0.5                           ; searching box half-size for A1
Albedo2 = 0.0                            ; value/searching box center for A2
dAlbedo2 = 0.5                           ; searching box half-size for A2
qfit = 0.0                               ; do you want to fit mass ratio?
q = 0.0                                  ; value/searching box center for q
dq = 0.001                               ; searching box half-size for q
redNoiseFit = 0.0                        ; to fit (1) or not to fit(0) red noise?
sigma_r = 0.0                            ; red noise factor value/searching box center
dsigma_r = 0.04                          ; half-size for red noise searching box
sigma_w = 0.0                            ; white noise value/searching box center
dsigma_w = 0.04                          ; half-size for white noise searching box
TimeResolution = 1.0                     ; TimeResolution
RVOffsetFit = 1.0                        ; do you want to fit offsets betw. RV-instr.?
rv_offset = 0.0                          ; searching box center for rv-offset fit
drv_offset = 1.5                         ; half-size of searching box of rv-offset fit
RVTrendFit = 0.0                         ; Do you want to fit the RV-trend?
d1 = 0.0                                 ; linear term searching box center
dd1 = 0.01                               ; linear term half-size of the searching box
d2 = 0.0                                 ; quadratic term searching box center
dd2 = 0.0                                ; quadratic term half-size of the searching box (
symbolsize = 1.0                         ; increase/decrease the point-size in the Figure
overplot_fitline = 0.0                   ; you can plot the fit over the data (1) or below it (0)
curve_color = blue                       ; color of the fit without red noise
redNoise_curve = red                     ; color of the fit with red noise
LineType = dashed                        ; linetype of the red-noise fit-curve
rvDo = Yes                               ; do you fit the RV-curve of the primary?
primary_rv = rv_corot20b.dat             ; primary's rv-curve datafile
rv2Do = No                               ; do you fit the RV-curve of the secondary?
secondary_rv = secondary_rv.dat          ; secondary's rv-curve datafile
photometric_datafile = tomodel_c20b.dat  ; photometric data file
Parabola_fit = 0.0                       ; do you subtract a parabola from the data?
ca = 0.0                                 ; constant term – searching box center
dca = 0.01                               ; constant term – searching box half size
cb = 0.0                                 ; linear term – searching box center
dcb = 0.0                                ; linear term – searching box half size
```

```
cc = 0.0                              ; quadratic term – searching box center
dcc = 0.0                             ; quadratic term – searching box half size
parabolaPhaseShift = 0.0              ; mid-phase of the parabola – searching box center
deltaParabolaPhaseShift = 0.0         ; mid-phase if the parabola – searching box half-size
Grazing_test = 0.0                    ; grazing test parameter
t_eff_fit = 0.0                       ; fit effective temperature of the primar?
T_eff = 5880.0                        ; effective surface temperature of the primary in K
satellite = CoRoT                     ; satellite name (CoRoT, Kepler, TESS or other)
lambda = 600.0                        ; effective wavelength of the observations in nm
secLDFit = 0.0                        ; do you want to fit the secondary's limb darkening?
u2p = 0.0                             ; u+ for the secondary – searching box center
du2p = 0.0                            ; u+ for the secondary – searching box half-size
u2m = 0.0                             ; u- for the secondary – searching box center
du2m = 0.0                            ; u+ for the secondary – searching box half-size
semimajor_axis_fit = 1.0              ; 0: fix, 1: fit the scaled semi major axis
ars = 25.0                            ; Searching box center for the semi-major axis
dars = 24.0                           ; Half-size of the searching box for the semi-major axis
radiusratio_fit = 1.0                 ; 0: fix, 1: fit the radius ratio
rprs = 0.5                            ; Searching box center for radius ratio
drprs = 0.5                           ; Half-size of the searching box for the radius ratio
conjuction_param_fit = 1.0            ; 0: fix, 1: fit the conjuction parameter
b = 3.0                               ; Searching box center for the conjuction parameter
db = 3.0                              ; Half-size of the searching box for the conj. parameter
limb_dark_fit_u1p = 1.0               ; 0: fix, 1: fit $u_+$ parameter for the primary
u1p = 0.6                             ; Searching box center for the $u_+$ parameter (primary)
du1p = 0.6                            ; Half-size of the searching box for the  $u_+$ parameter
limb_dark_fit_u1m = 1.0               ; 0: fix, 1: fit the $u_-$ parameter for the primary
u1m = 0.2                             ; Searching box center for the $u_-$ parameter (primary)
du1m = 1.0                            ; Half-size of the searching box for  the $u_-$ parameter
omegadot_fit = 0.0                    ; 0: fix, 1: fit domega/dt parameter for the arg. of peri.
Omegadot = 0.0                        ; Searching box center / the change in the arg. of peri.
domegadot = 0.05                      ; Half-size of the searching box for the change in peri.
third_light_fit = 0.0                 ; 0: fix, 1: fit the third light (related to contamination)
l3 = 0.0                              ; Searching box center for the third light
dl3 = 0.0                             ; Half-size of the searching box for the third light
esinw_fit = 1.0                       ; 0: fix, 1: fit the e sin(omega) parameter
esinw = 0.0                           ; Searching box center for e sin(omega) parameter
desinw = 1.0                          ; Half-size of the searching box for the e sin(omega)
ecosw_fit = 1.0                       ; 0: fix, 1: fit the e cos(omega) parameter
ecosw = 0.0                           ; Searching box center for e cos(omega) parameter
decosw = 1.0                          ; Half-size of the searching box for the e cos(omega)
Omega = 90.0                          ; Node of the longitude (only for plotting purposes)
intensity_ratio_fit = 0.0             ; 0: fix, 1: fit the surface brightness ratio
f = 0.0                               ; Searching box center for the surface brightness ratio
df = 0.5                              ; Half-size of the searching box for surf. bright. ratio
period_fit = 0.0                      ; 0: fix, 1: fit the orbital period of the transiting object
period = 9.24285                      ; Searching box center for the orbital period
dperiod = 0.0                         ; Half-size of the searching box for the orbital period
epoch_fit = 1.0                       ; 0: fix, 1: fit the epoch of the transits
epoch = 55266.0001                    ; Searching box center for the transit epoch
```

```
depoch = 0.05                          ; Half-size of the searching box for the transit epoch
gammaVelocity_fit = 1.0                ; 0: fix, 1: fit the systematic velocity
gammaVelocity = 0.0                    ; Searching box center for the syst. velocity, km/s
dgammaVelocity = 100.0                 ; Half-size of the s. box for the sys. vel. in km/s
Kampl_fit = 1.0                        ; 0: fix, 1: fit the radial velocity amplitude of the pri.
Kampl = 0.0                            ; Searching box center for the RV-amplitude in km/s
dKampl = 100.0                         ; Half-size of the search. box for RV-amplitude, km/s
height_fit = 1.0                       ; 0: fix, 1: fit the height parameter of the fluxes
height = 0.0                           ; Searching box center for the height parameter
dheight = 0.02                         ; Half-size of the searching box for the height par.
Ellipsoidal_fit = 0.0                  ; 0: no ellips. effect, 1: ellips. variability is present
empty = 0.0                            ; reserved for future use
empty = 0.0                            ; reserved for future use
empty = 0.0                            ; reserved for future use
empty = 0.0                            ; reserved for future use
empty = 0.0                            ; reserved for future use
wsgrid = 0.0                           ; 0: normal sky-projected distance calculation, 1: with
                                       ; approximation via Csizmadia & Pasternacki 2013
ScreenPlot = 1.0                       ; if it is zero, it does not make screen-plots
Z = 0.019                              ; metallicity of the primary
dZ = 0.005                             ; searching box (uncertainty) of the metallicity
nchain = 2.0                           ; number of chains for MCMC-run
thin = 1.0                             ; thinning parameter for MCMC (>=1)
Gaia_radius_primary = 1.0              ; estimated radius of the primary (e.g. from Gaia)
dGaia_radius_primary = -1.0        ; if this is negative, no Gaia-radius will be used as a prior
RMFit = 0.0                            ; 0: not to include, 1: to include the RM-effect into RV
VsinI1 = 0.0                           ; Searching box center for the VsinI of the primary
dVSinI1 = 0.0                          ; Half-size of the searching box for VsinI (primary)
VsinI2 = 0.0                           ; Searching box center for the VsinI of the secondary
dVSinI2 = 0.0                          ; Half-size of the searching box for VsinI (secondary)
beta1 = 0.0                            ; Searching box center for the beta-parameter (pri)
dbeta1 = 45.0                          ; Half-size of the searching box for the beta-param.
beta2 = 0.0                            ; Searching box center for the beta-parameter (sec.)
dbeta2 = 45.0                          ; Half-size of the searching box for the beta-param.
 u1rm = 0.0                            ; S. box center for u+ LD-coeff for the RM in the pri.
du1rm = 0.0                            ; Half-size of the searching box for u+ LD-coeff (RM)
u2rm = 0.0                             ; S. box center for u- LD-coeff for the RM in the pri.
du2rm = 0.0                            ; Half-size of the searching box for u- LD-coeff (RM)
 u3rm = 0.0                            ; S. box center for u+ LD-coeff for the RM in the sec.
du3rm = 0.0                            ; Half-size of the searching box for u+ LD-coeff (sec.)
 u4rm = 0.0                            ; S. box center for u- LD-coeff for the RM in the sec.
du4rm = 0.0                            ; Half-size of the searching box for u- LD-coeff (sec.)
logg_constraint = 0.0                  ; 0: do not use, 1: use the primary’s logg as prior
logg = 0.0                             ; logg of the primary.
logg_uncertainty = 0.0                 ; 1sigma unc. of the prim. logg, used as Gauss. prior
empty = empty                          ; reserved for future use
error_mode = 0.0                       ; 0: MCMC-error estimation, 1: bootstrap error estim.
jitter_rv = 0.0                        ; jitter of the RV-data, in km/s
jitter_phot = 0.0                      ; jitter of the photometric points
bootstrap_smaples = 1000.0             ; number of tests in the bootstrap analysis
```

```
qxfit = 0                                   ; 0: no extra baseline-fit, 1: extra baseline fit with qx.pro
nparams = 0                                 ; number of parameters for the extra baseline fit-curve
```

If nparams=o, then the files ends here. If not, then the searching box-sizes and their centers should be given here. An example for that can be found in the EXAMPLES directory in the distributed zip-file.

Results & Uncertainties

The latest version of TLCM estimates the uncertainties of the final parameters in five different ways and it gives the end-results from four different approaches. The user should choose among them.

Method I:
The first method simply gives the best χ^2 or lowest -logL fond by the last run of TLCM without error bars. Because of the nature of the random numbers, a re-run may produce different best fit.

Method II and III:
The second and third method utilizes a bootstrap-approach. This consists of the following steps:

(a) Take the best fit of HS+SA and calculate the residuals: $r_i = O_i - M_{best,i}$ where i is the index
of the observation, O is the observed value and M is the model.
(b) Select randomly 1/e fraction of all datapoints, and replace their residuals by another one randomly: $r_k' = r_s$ where k and s are randomly selected.
(c) Produce a new testlightcurve: $T_i = M_{best,i} + r_i'$.
(d) Fit T_i (this is done automatically by a shorter SA-chain in TLCM) and record the results – this is a bootstrap-iteration.
(e) Repeat steps (b-d) N_{sample} times. N_{sample} is set in the config-file by the user but the code takes at least $N_{sample} = 3$.

The goal of this exercise is to make the observational error distribution gaussian. At the end, you will have a distribution of the parameters. TLCM calculates the final solution and its error bars in two ways:

(I) It takes the MEAN and the STDDEV of all bootstrap-iteration – that is *Method II.*
(II) It fits a Gaussian to the histogram of the parameters (the histogram is calculated via the optimum binsize given by the Freedman-Diaconis rule) and the peak gives the result of the fit while from the width the 1sigma (68% C. I.) uncertainty is calculated – that is *Method III.*

Method IV.
Method IV performs an MCMC analysis after the HS+SA runs. There are at least two MCMC-chains – you can specify a bigger number of chains with changing the nchain=2 parameter in the config_ninit.dat file. The length of the chains are specified by the NINIT parameter in the same file. They can be any integer numbers with the constraints of nchain ≥ 2 and nchain ≥ 1. However, there is another limitation: the memory (RAM) of your computer. If you have a memory-problem message and the run stops, then you can do the followings:

- decrease the number of chains
- decrease the number of chain-length
- increase the thin parameter
- use a different computer.

The use of the thin-parameter can be justified from other point of view, too. If the convergence is slow or the mixing is not very good, then this usually occurs because the chain evolves too slowly. In such vase you can see a small, very thin chain which looks like a random walk-line in the screen. It may have quasiperiodic behaviour. If the mixing is good, then you see a very random jumping in the parameter. To avoid the quasiperiodic behaviour, you can record only every *n*th step only, eg. thin=n=10. Even thin = 100 is not unusual. This helps to get better mixing.

Method IV sorts the parameters in increasing order. The best and the median solution will be reported. The 1, 2 and 3 sigma error bars are calculated with checking the values of the solutions which are in the 15.87 – 84.13%, 2.275 – 97.725%, 0.135 – 99.865% quartiles, respectively. They contain 68.26%, 95.45% and 99.73% of all points around the median, respectively,

Method V
Method V uses the same MCMC chain as Method IV and calculated the marginalized likelihoods in the way given in Sajina et al. (2006).

In case of any method, the code will calculate the absolute dimensions of the system from the best solution and from the symmetrized error bars of Method II or IV (symmetrization means, that the downward and upward error bars are averaged). Note that when the median solution and the best solution are far from each other, then the posterior parameter distributions are likely asymmetric, and an unmanaged error source (e.g. red noise, unconsidered instrumental or astrophysical effect or parameter) may be still present in the data.

Important note on final parameter calculation. It is a well-known fact that in some cases the best solution (smallest found χ^2 or highest likelihood or minimum logL) does not coincide with the mean or with the median of the posterior distributions. Such an example is shown here in Figures 1 and 2 which was obtained by modelling jointly the RV+LC of CoRoT-20b without red noise (see later). Shall we use the best solution or the median solution from the posterior for the final reported values? The problem is unsolved (cf. Heller et al. 2019). Therefore, TLCM reports the absolute parameters obtained from the *best solution* and also the absolute parameters obtained from the *median solution*. When they disagree, then the user can choose which one will be used in the publication. Most of the investigators use the *median-solution*, as better representative of the probability distribution.

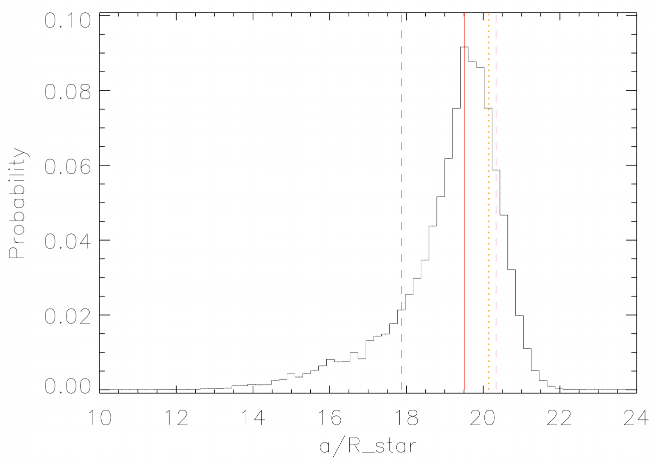

Figure 1. *The probability density distribution of a/R_s parameter for the joint RV+LC solution of CoRoT-20b (without red noise). The individual probabilities are defined as $p_i = exp(\chi^2_i \chi^2_{min})$ and in every bin the marginalized likelihood is calculated as $P = \Sigma p_i$ (lower < a/R_s < upper) , where the summation is extended for the binsize with limits denoted by 'lower' and 'upper'. Orange dotted line shows the place of the solution wih the best χ^2,solid red line is the peak of this distribution and red dashed lines are the 1sigma lower and upper limits calculated via the usual 16-85% quartule rule.*

The asymmetric distribution shown in Figure 1 was caused by some well-known correlation between the scaled semi-major axis a/R_s, and conjuction (impact) parameter b (see also Csizmadia et al. 2011) which is visualized in Figure 2.

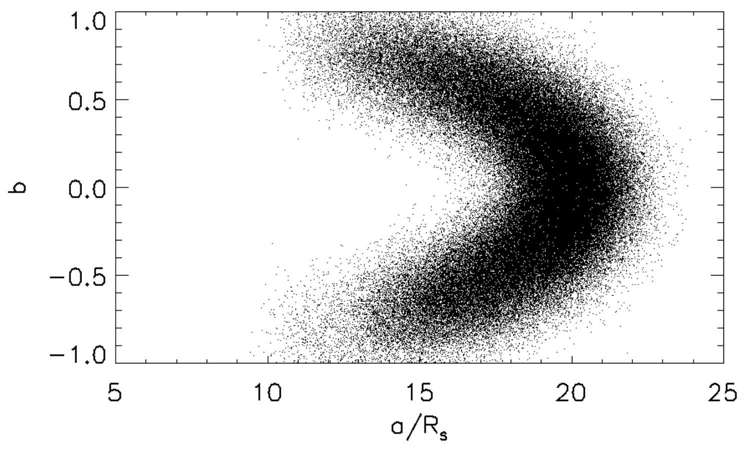

Figure 2. *Each dot represents one light curve solution in the 10 MCMC chains applied for for the joint RV+LC solution of CoRoT-20b (without red noise). Look at the boomerang-shape of the distribution. Such correlations exists especially when the ingress and egress parts of the light curve are not well defined by observations due to low Signal-to-Noise ratio, bad time-sampling, strong binning etc. (These parts of the light curves hold information about the inclination and impact parameter.) Cf. with Figure 3.*

An example

The above *config_ninit.dat* file was used to check the code on CoRoT-20b, using the lc and rv-data from Deleuil et al. (2012). For sake of comparison, I present Deleuil et al's results as well. I have chosen this example because CoRoT-20b is very eccentric as well as it shows only slight red noise-effect. To make your training easier, I re-distribute the rv- and lc-files together with TLCM in the EXAMPLES directory.

Table I presents the results of Deleuil et al. (2012) and my actual one. Notice that the light curve was managed slightly different way by me and by Deleuil et al., however, the results are in very good agreement.

Table I. Comparison of the light curve solutions of CoRoT 20b. No red noise was taken into account. Notations as usual. $\sigma = \sqrt{sigma_{Deleuil}^2 + sigma_{TLCM}^2}$ and the difference is expressed as the absolute difference of the two solutions in units if sigma. The results reported here are from the 'median solution'

Parameter	Deleuil et al. (2012)	Results of TLCM	Difference [σ]
Orbital period	9.24285 ± 0.00030 d	9.24285 d (fixed)	-
Transit epoch	BJD 55 266.0001 ± 0.0014	55 265.99999 ± 0.00001	0.1σ
Periastron eopoch	BJD 55 265.79074	-	-
esinω	0.468 ± 0.017	-	-
ecosω	0.312 ± 0.022	-	-
$e^{1/2}$sinω	-	$0.628^{+0.023}_{-0.025}$	-
$e^{1/2}$cosω	-	$0.425^{+0.028}_{-0.029}$	-
e	0.562 ± 0.013	0.574 ± 0.020	0.5σ
ω	56.3 (+2.4,-2.3) deg	55.9 ± 2.0 deg	0.1σ
K	454 ± 9 m/s	458 ± 12 m/s	0.3σ
V_γ	60.623 ± 0.006 km/s	60.619 ± 0.008 km/s	0.4σ
HARPS-SOPHIE offset	93 ± 11 m/s	87 ± 20 m/s	0.3σ
SOPHIE-FIES offset	163 ± 20 m/s	168 ± 22 m/s	0.2σ
radius ratio R_p/R_s	0.0842 ± 0.0017	$0.0859^{+0.0039}_{-0.0031}$	0.5σ
impact parameter b	0.26 ± 0.08	$0.0^{+0.49}_{-0.48}$	within error bars
a/R_s	18.95 ± (+0.63, -0.73)	$18.75^{+1.56}_{-2.61}$	0.1σ
inclination	88.21 ± 0.53 deg	88.88 ± 3.7 deg	0.2σ
Stellar density	1.51 (+0.15, -0.17) g/cm^3	1.46 ± 0.49 g/cm^3	0.1σ
$M^{1/3}/R_s$ (solar units)	1.022 (+0.034, -0.039)	1.01 ± 0.11	0.1σ

code produced the following figures (Figures 3-4).

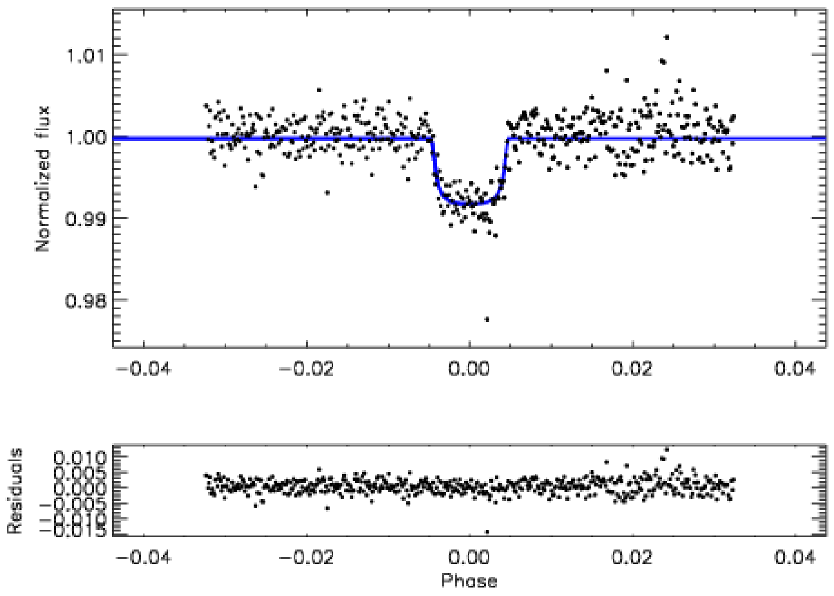

Figure 3. The upper panel shows the binned observations of CoRoT-20b, obtained by the CoRoT satellite (black points) and the fit – without red noise – by TLCM (blue curve). Lower panel shows the residuals.

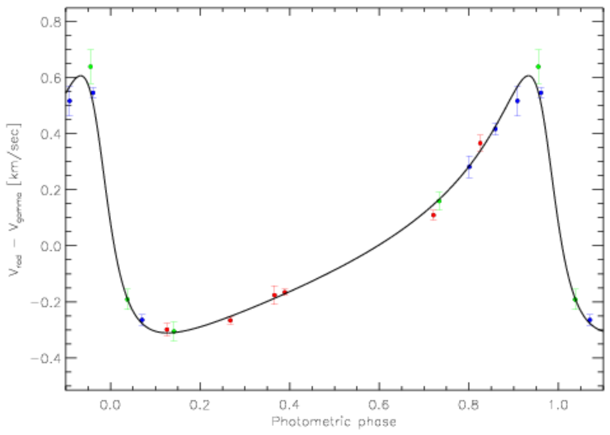

Figure 4. The radial velocity measurements and their error bars of CoroT-20b (taken from Deleuil et al. 2012) and the fit by TLCM (solid curve). Vertical lines are the observational error bars. Colours denote the different instruments. Red: HARPS-data. Green: FIES-data. Blue: SOPHIE-data.

The code was widely tested and validated on other objects, too. However, you have to take into account, that only the transit, occultation, parabola and RV-part are precise; the ellipsoidal and reflection and beaming effects are only approximate up to the first order to increase the speed. More term, more precise approximations can be added in the future when it is needed.

There are a plenty of outputs:

output.eps: a black & white figure with the observed data, fit as a solid line and the residuals.

output1.eps: the same, but in a colored version; you can change the color of the fitted line in the *config_ninit.dat*.

results.dat: the final parameters, erro bars and some extra outputs calculated from the results.
residuals.dat: you may want to produce your own plots, therefore here you can find:
- orbital phase calculated via the program (or the same as your input if you put the phase in0
- your independent variable (time or phase) from the data file
- observed flux (from the data file)
- modelled flux
- observed flux - modelled flux (residuals, without rednoise)
- redNoise flux value
- observed-modelled-rednosie fluxes (residuals with rednoise)
- eclipse/transit flux for the primary
- eclipse/transit flux for the secondary

- beaming flux for the primary
- ellipsoidal flux for the primary
- reflection flux from the primary
- beaming flux for the secondary
- ellipsoidal flux for the secondary
- reflection flux from the secondary

These latter columns where the contributions from different effects are separated can be used for further analysis.

Additional notes.

When you plot (ScreenPlot = 1), then two selected parameters appear in a Figure. The diamonds represent the actual solutions, where are the individuals of the population. The red diamonds represent the 10% best ones (in χ^2 or -loglikelihood terms). The red diamond surrounded by a green one is the actual best solution.

Runtime and exposure time issues

The runtime strongly depends on the number of data points in the light curve. If you have many (>1000) it can be a good idea to bin them.

When you bin strongly, or the exposure time or your bin is longer than approximately 0.0001 part of the orbital period (typical for Kepler/K2 long cadence data), maybe you modify your light curve shape with smoothing it (e.g. Kipping 2010). Therefore, TLCM offers you a numerical integration over the exposure time: you can choose, 1, 5, 9, 17 or 33 subpoints inside an exposure and from that a numerical integrator calculates the observable intensity. (33 corresponds roughly 1 minute subexposures during a Kepler long cadence flux-measurement, and this will be summed up properly via a trapezoidal/Simpson-integration). For most of the Kepler-data, TimeResolution = 5 was found to be enough, so it means only 5 time longer run-time.

Grazing transits

It is awful to fit grazing transits (even for eclipsing binary stars), but many times we are obliged to do it. In this case you loose the 2nd and 3rd contact of the transits, so a lot of information is lost. In addition, close to the limb of the star, limb darkening is not well known, so fit is hard and degenerated.

To break down the degeneracy, TLCM is able to do a so-called semi-grid fit. If you do not want to do this, set grazing_test = 0.0 in the *config_ninit.dat* file. If you give any positive integer equal to or bigger than two, then TLCM will fix the conjuction parameter between b'-db' and b'+db' (both parameter defined in config_ninit.dat by the user) at

$$b'_i = b' - db' + 2.0 \times db' / grazing_{test} \times i, \qquad i = 0 \ldots grazing_{test} \quad .$$

This, of course, means a longer runtime, because if e.g. grazing_test = 21, then 21 different runs will be done with different conjuction parameteres (so, with different impact parameters). The best corresponding χ^2 value or -loglikelihood will be selected, and in the small vicinity a newer run will be done to refine the value of the conjuction parameter. The list of b' and the corresponding values of the goodness of the fit are also printed into a separate file called '*impact_test.dat*', and you can monitor how your object and fit behave and react to different inclinations (i.e. conjuction parameters).

Contact:

If you are in trouble when using TLCM, then pelase contact me via e-mail:
Szilard.Csizmadia@dlr.de

References

Csizmadia Sz. et al. 2011, A&A 531, A41
Csizmadia Sz. et al. 2013a, A&A 549, A9
Csizmadia Sz. et al. 2013b, ASPC 472, 147
Csizmadia Sz. et al. 2015, A&A 584, A13
Carter, J. & Winn, J. 2009, ApJ 704, 51
Deleuil, M. et al. 2012, A&A 538, A145
Geem, Z. W., Kim, J. H., & Lonatathan, G. V. 2001, Simulation, 76, 60
Heller, R., Rodenbeck, K., Bruno, G. 2019, A&A 624, A95
Kane, S. R. & Gelino, D. M. 2011, ApJ 729, 74
Kipping, D. M. 2010, MNRAS 408, 1758
Kopal, Z. 1959, Close Binary Systems, Publisher: Wiley
Mandel, K. & Agol, E. 2002, ApJ 580, L171
Russell, H. N. 1912, ApJ 35, 315
Sajina et al. 2006, MNRAS 369, 939