VBBinaryLensing 2.0
A fast and robust code for light curve computation

Valerio Bozza
University of Salerno, Italy
**VBBinaryLensing**

- **VBBinaryLensing** is a code for the calculation of *microlensing light curves*.
  - Point-Source Point-Lens
  - Extended Source Point-Lens
  - Binary Source Point-Lens
  - **Extended Source Binary Lens**

- **Higher order effects** implemented:
  - Linear limb darkening
  - Annual and space parallax
  - Circular orbital motion

- C++ library, importable in Python, source **code public**.
- Tested on Windows, Linux, Mac OS.

- Essential core of **RTModel** ([http://www.fisica.unisa.it/GravitationAstrophysics/RTModel.htm](http://www.fisica.unisa.it/GravitationAstrophysics/RTModel.htm)).
- Incorporated in several high-level analysis frameworks:
  - **pyLIMA** *(Bachelet et al. 2017)*
  - **MulensModel** *(Poleski & Yee 2018)*
  - **MuLAN** *(Cassan & Ranc 2017)*
Magnification by Contour Integration

- VBBinaryLensing uses the **contour integration** method (*Schramm & Kayser 1987; Dominik 1993; Dominik 1995; Gould & Gaucherel 1997; Dominik 1998; Dong et al. 2006; Dominik 2007*).

- **Parabolic correction** greatly increases the accuracy.
- **Error control** is performed through several error estimators.
- **Sampling** is **optimized** where/when really needed to match the accuracy goal.
- **Limb darkening** accounted by repeating the calculation on multiple annuli.

(V. Bozza, MNRAS 408, 2188 (2010))
VBBinaryLensing 2.0: what’s new?

- The root solving routine is now Skowron & Gould.

- Relative precision goal and/or absolute accuracy goal.

- Extended source calculation performed only when really needed.

- Extended source-point lens calculation performed using pre-calculated tables.

- Light curve calculations on large datasets with a single call.
Skowron & Gould algorithm

- The binary lens equation is equivalent to a fifth degree complex polynomial.

Two ways for finding roots (Skowron & Gould arXiv:1203.1034)

- Use Newton’s method when close enough to the goal.

$z_{k+1} = z_k - \frac{1}{G} \\
G = \frac{p'(z_k)}{p(z_k)}$

- Use Laguerre’s method when still far away.

$H = G^2 - \frac{p''(z_k)}{p(z_k)} \\
z_{k+1} = z_k - \frac{n}{G \pm \sqrt{(n-1)(nH - G^2)}}$

- Speed-up of order 1.1-1.2 with respect to zroots (Numerical Recipes).
- More robust for extreme parameters.
- The original Fortran code has been translated to C++ by Tyler Heintz and Ava Hoag (Westminster College) during a summer studentship in Salerno.
Precision vs Accuracy

- If the **accuracy goal** is $\delta$, the calculation stops when the absolute error on the magnification is
  \[ \delta \mu < \delta \]

- If the **precision goal** is $\varepsilon$, the calculation stops when the relative error is
  \[ \frac{\delta \mu}{\mu} < \varepsilon \]

- At **high-magnification**, a millimag precision is sufficient.
- The calculation stops earlier and we get a significant **speed-up**.
PSBL vs ESBL

- Point-Source approximation works for most points in a microlensing light curve.
- Only regions around the caustics require the full Extended-Source calculation.
- In general, when modeling a specific event, the modeler defines time windows in which different codes are used.

- This approach is unpractical if we aim at modeling thousands of events automatically.
Tracking finite-source effects

- Pre-defined time windows are replaced by dynamical evaluation of specific metrics:

Point-Source calculation

Quadrupole below accuracy goal? or Ghost images far from critical?

- No: Extended-Source calculation
- Yes: Done
Finite-Source Metrics

- For each image, we can evaluate the **quadrupole** correction (*Pejcha & Heyrovsky 2009, Gould 2008, Cassan 2017*)

\[
Q = 2 \left| \frac{\text{Re}\left[3 \bar{f}^3 f''^2 - \left(3 - 3J + J^2 / 2\right) f''^2 + Jf^2 f''\right]}{J^5} \right|
\]

\[
Q_2 = \left| \frac{\text{Im}\left[3 \bar{f}^3 f''^2\right]}{J^5} \right|
\]

Additional term for better protection around cusps

- **Ghost images** provide us with an approximate distance to caustics:

\[
d_G = \left| \frac{J}{\nabla_\zeta J} \right|
\]

- Point-Source approximation is sufficient if

\[
(Q + Q_2)\rho^2 < \text{accuracy} \quad \text{AND} \quad d_G > \rho
\]
Safety maps

- The safety criterion \((Q + Q_2)\rho^2 < \text{accuracy} \quad \text{AND} \quad d_G > \rho\) tells us when to go to full Extended-Source calculation.
#include <stdio.h>
#include "VBBinaryLensingLibrary.h"

int main()
{
    VBBinaryLensing VBBL;
    double Mag, s, q, y1, y2, rho, a1, accuracy;
    s=0.8;  // separation
    q=0.1;  // mass ratio
    y1=0.01;  // source position
    y2=0.3;
    rho=0.01;  // source radius
    VBBL.a1=0.51;  // Linear limb darkening coefficient
    VBBL.Tol=1.e-2;  // Accuracy goal
    Mag=VBBL.BinaryMag2(s,q,y1,y2,rho);
    printf("Magnification = %lf\n",Mag);
    return 0;
}
Extended-Source-Point-Lens

- Extended-Source-Point-Lens has two regimes:

$$
\mu = \begin{cases} 
\frac{u^2 + 2}{u\sqrt{u^2 + 4}} f_{out}(\frac{\rho}{u}, \rho) & \text{if } u > \rho \\
\sqrt{1 + 4/\rho^2} f_{in}(\frac{u}{\rho}, \rho) & \text{if } u < \rho 
\end{cases}
$$

- Rather than calculating elliptic integrals, we evaluate the correction factors $f_{in}$ and $f_{out}$ by pre-calculated tables.
- This is much faster (and accurate) than any other methods.

- Limb darkening is obtained by dividing the source in annuli as in contour integration.
Outlook

- **VBBinaryLensing 2.0** achieves a significant speed-up by introducing new **decision trees** at several levels:
  - Skowron & Gould algorithm in root solving
  - Relative precision goal (great speed-up at high-mag)
  - Finite-Source used only when needed (great speed-up far from caustics).

- Code will still highly benefit from **continuous feedback** from the community.

- Extensive use within high-level analysis environments.

- Next step: **multiple lensing** with contour integration.