Università degli Studi di Napoli Federico II

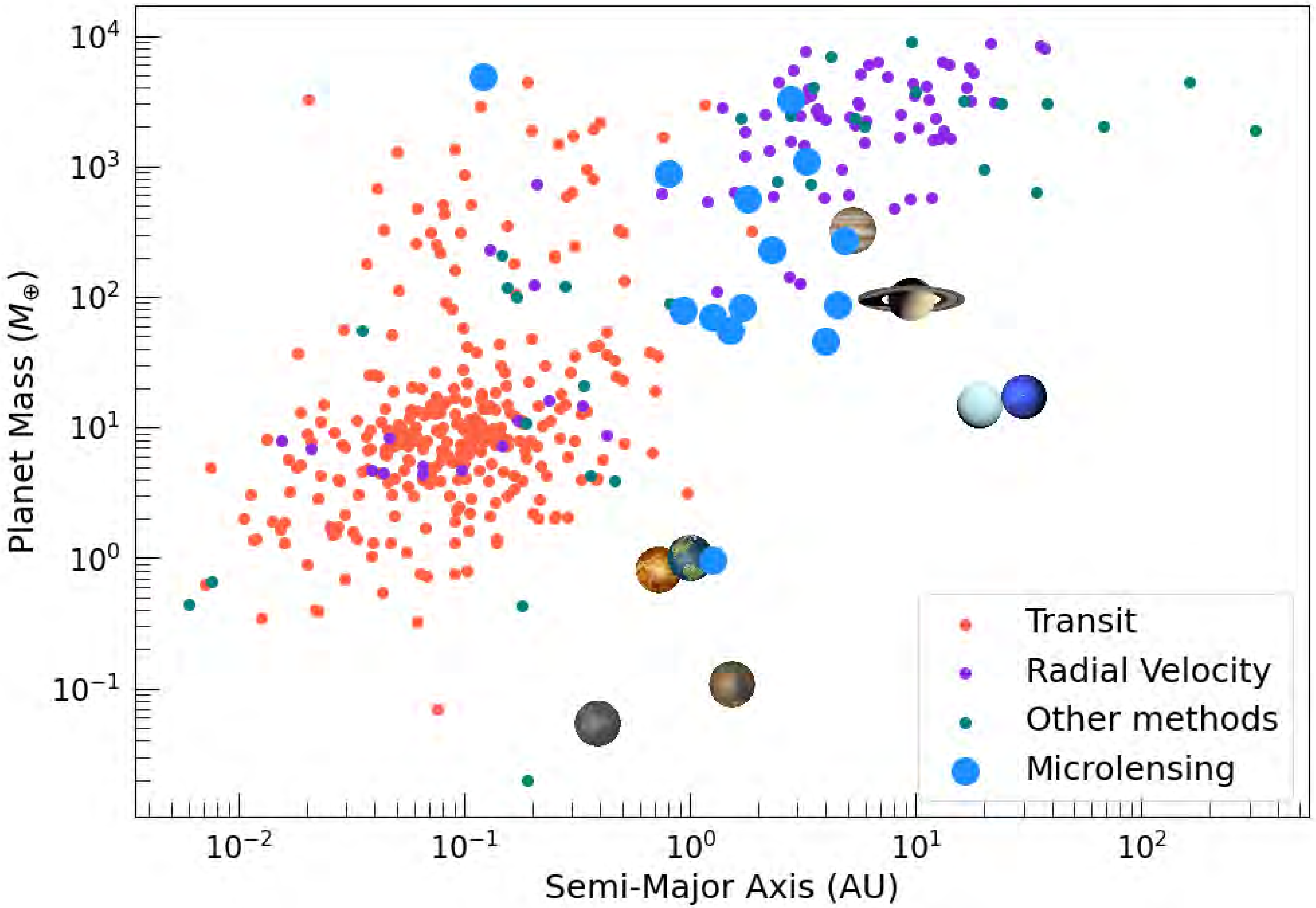
Università degli Studi di Salerno

Microlensing Conference 2024

A NEW CODE FOR MULTIPLE MICROLENSING EVENTS

Speaker: Vito Saggese, Ph.D. Student

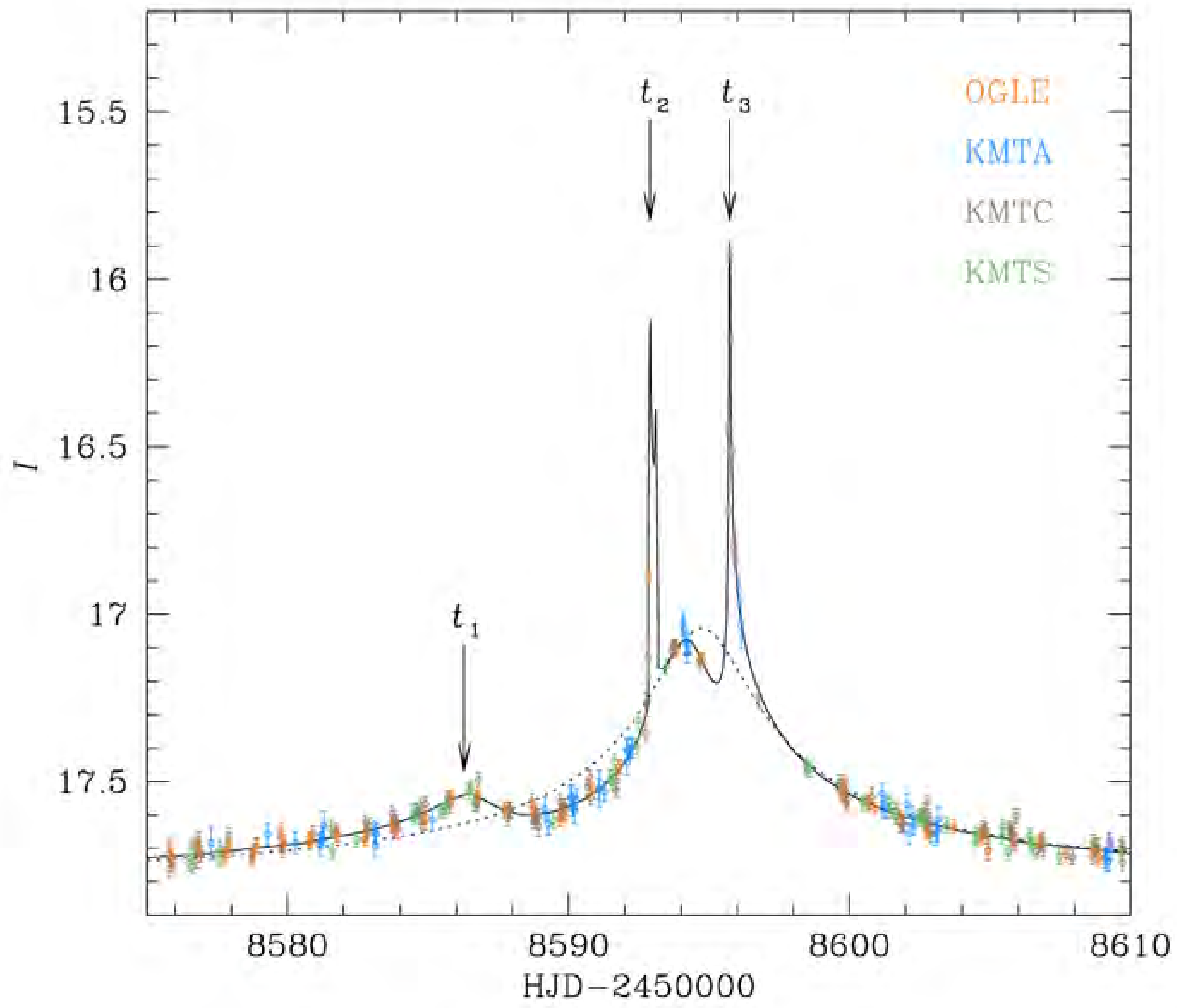
PLANETS IN MULTIPLE SYSTEMS



TRIPLE LENS SYSTEM

OGLE-2019-BLG-0468Lb,c

(C. Han et al. 2022)

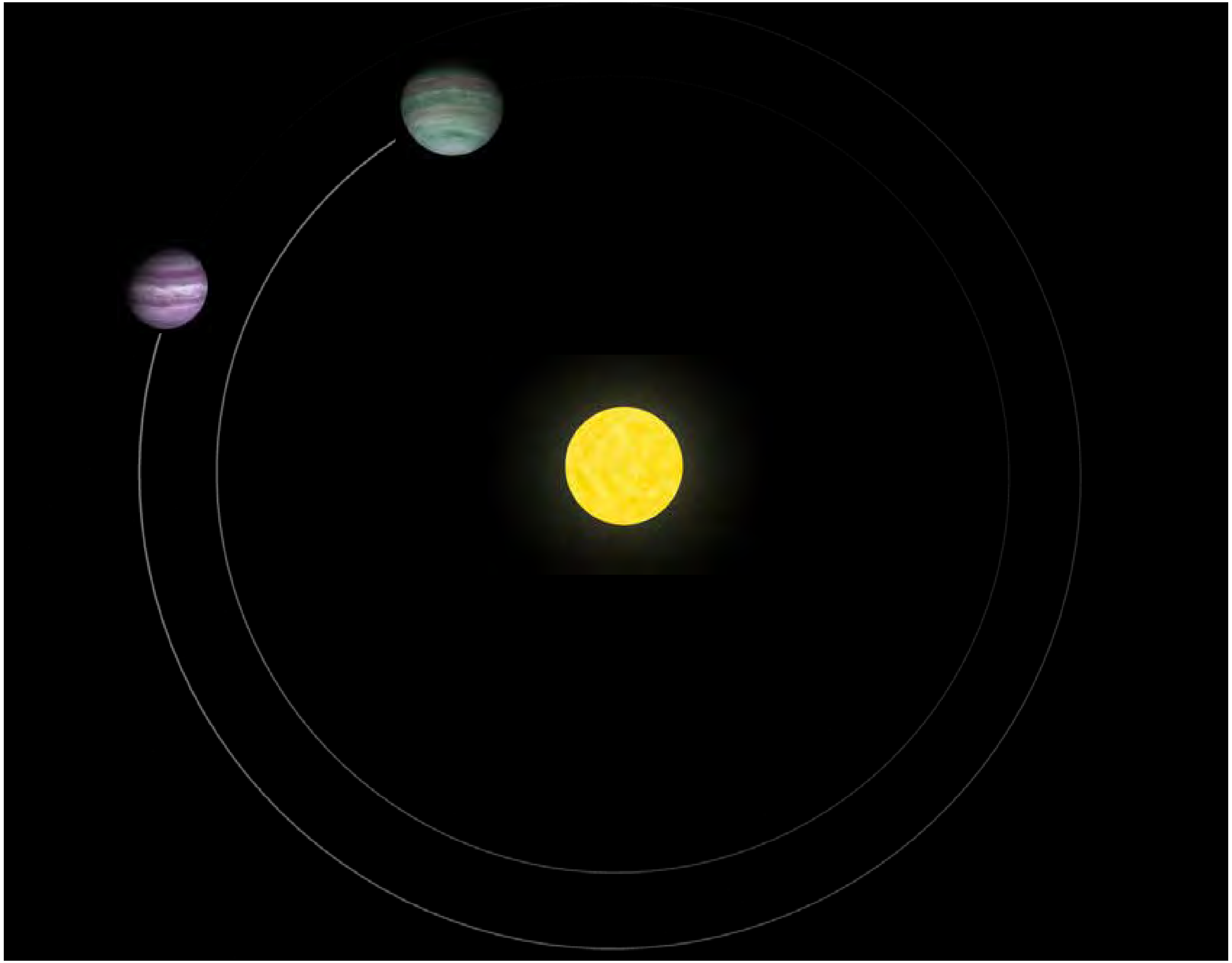


TRIPLE LENS SYSTEM

OGLE-2019-BLG-0468Lb,c

(C. Han et al. 2022)

Credit: NASA



MICROLENSING MAGNIFICATION

- Microlensing magnification can be calculated by:
    - **Inverse-ray-shooting** *(Wambsganns 1992, 1997; Bennett & Rhie 1996; Bennett 2010; Dong et al. 2009)*
    - **Contour integration** *(Schramm & Kayser, 1987; Dominik 1995; Gould & Gaucherel 1997; Dominik 1998)*

MICROLENSING MAGNIFICATION

- Microlensing magnification can be calculated by:
  - **Inverse-ray-shooting** *(Wambsganns 1992, 1997; Bennett & Rhie 1996; Bennett 2010; Dong et al. 2009)*
  - **Contour integration** *(Schramm & Kayser, 1987; Dominik 1995; Gould & Gaucherel 1997; Dominik 1998)*

LENS EQUATION

$$\zeta = z - \sum_i \frac{m_i}{\bar{z} - \bar{a}_i}$$

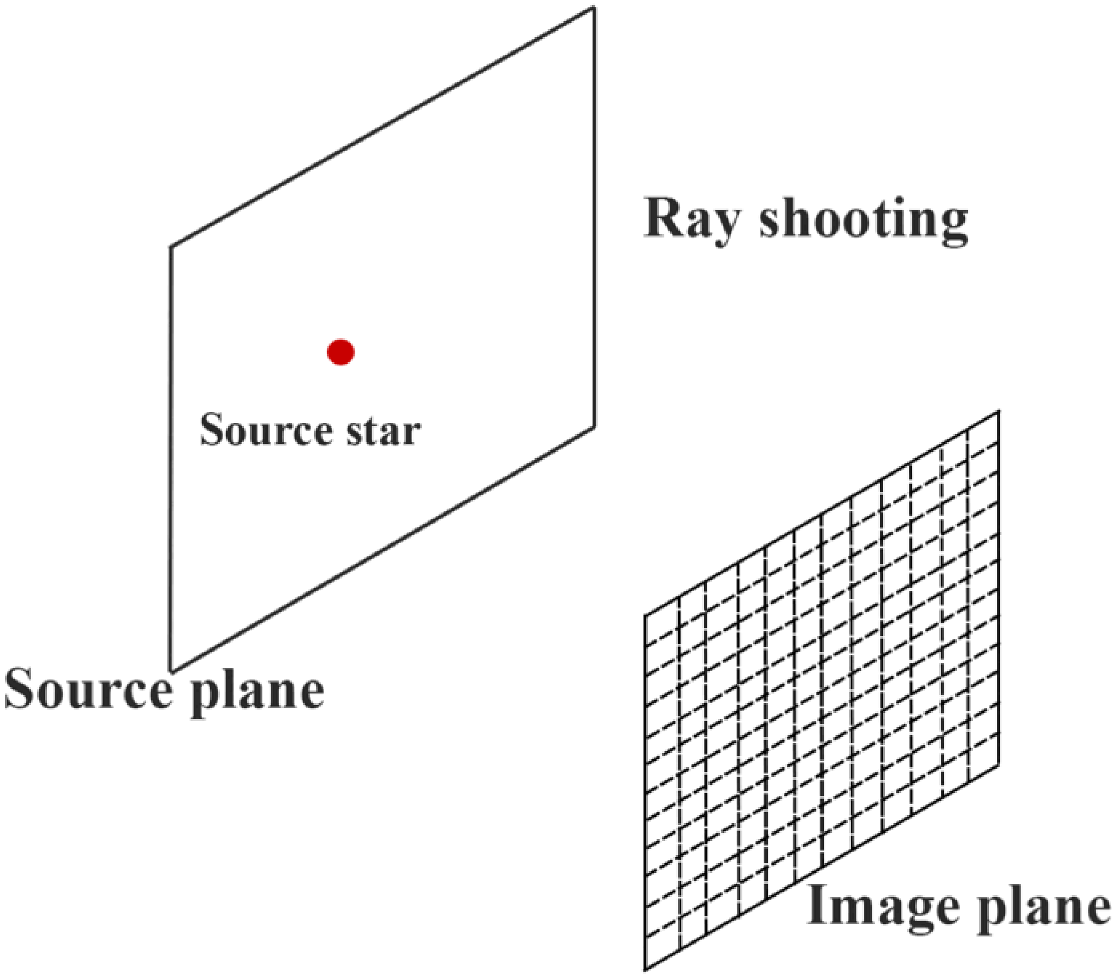
Source

Image

Lenses

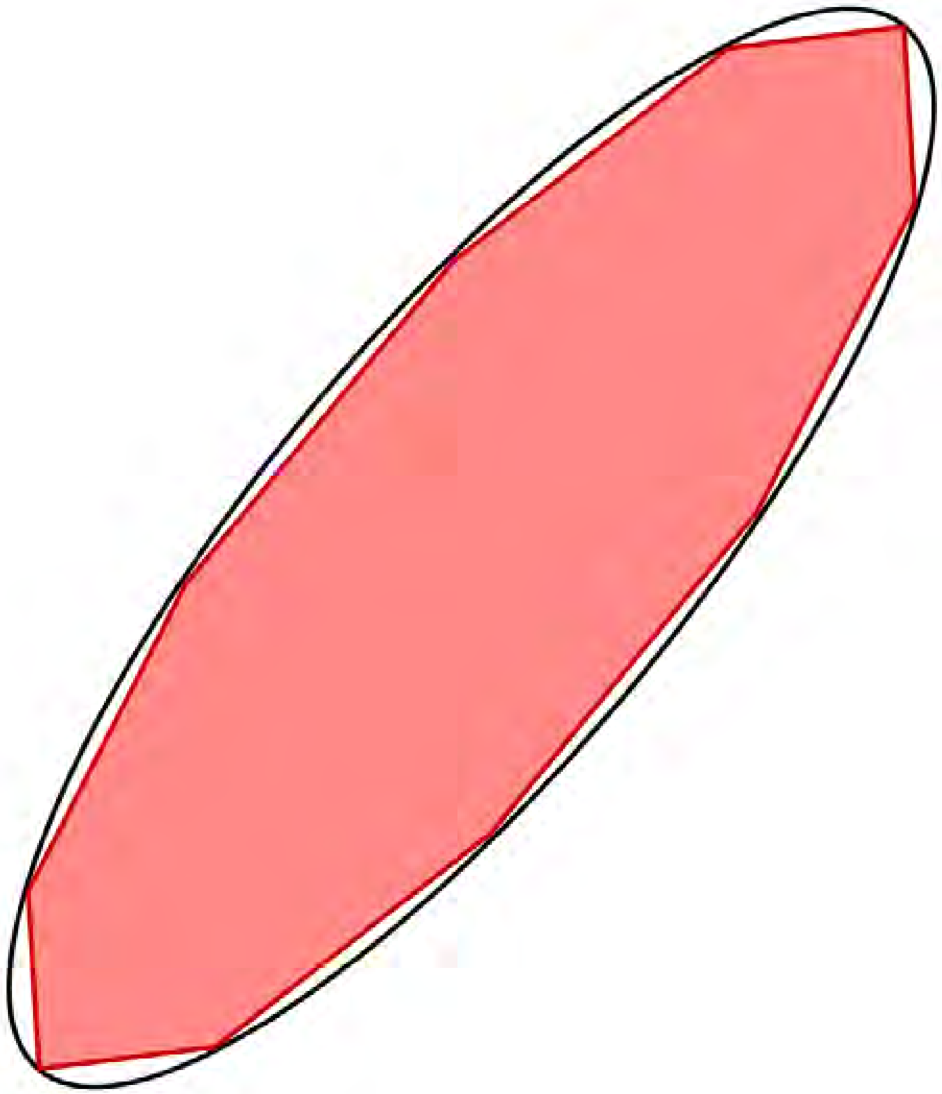
MICROLENSING MAGNIFICATION

- Microlensing magnification can be calculated by:
  - **Inverse-ray-shooting** *(Wambsganns 1992, 1997; Bennett & Rhie 1996; Bennett 2010; Dong et al. 2009)*
  - **Contour integration** *(Schramm & Kayser, 1987; Dominik 1995; Gould & Gaucherel 1997; Dominik 1998)*



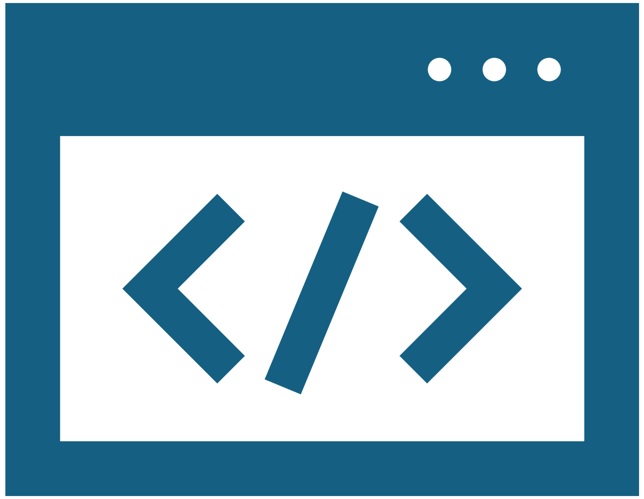
MICROLENSING MAGNIFICATION

- Microlensing magnification can be calculated by:
  - **Inverse-ray-shooting** *(Wambsganns 1992, 1997; Bennett & Rhie 1996; Bennett 2010; Dong et al. 2009)*
  - **Contour integration** *(Schramm & Kayser, 1987; Dominik 1995; Gould & Gaucherel 1997; Dominik 1998)*



VBBinaryLensing

- Computation by contour integration
- Public code
- Written in C++
- Importable in Python
- Used by most modelling platforms:
    - RTModel
      (http://www.fisica.unisa.it/GravitationAstrophysics/RTModel.htm)
    - pyLIMA
      (https://github.com/ebachelet/pyLIMA, Bachelet et al. 2018)
    - MulensModel
      (https://github.com/ebachelet/pyLIMA, Bachelet et al. 2018)
    - muLAn
      (https://github.com/muLAn-project/muLAn, Cassan & Ranc 2017)



https://github.com/valboz/VBBinaryLensing

MULTIPLE LESES

- **Lens equation**:

$$\zeta = z - \sum_i \frac{m_i}{\bar{z} - \bar{a}_i}$$

- **Associated polynomial** p(z) (degree $n^2 + 1$):

$$(\zeta - z)\prod_i\{(\bar{\zeta} - \bar{a}_i)\prod_j(z - a_j) + \sum_j[m_j\prod_{k\neq j}(z - a_k)]\} + [\prod_j(z - a_j)]\sum_i[m_i(\prod_{p\neq i}\{(\bar{\zeta} - \bar{a}_i)\prod_j(z - a_j) + \sum_j[m_j\prod_{k\neq j}(z - a_k)]\})] = 0$$

- **Image theorem** (Rhie 2001, Khavinson 2004):

  Minimum number of images = $\boldsymbol{n}$**+1**
  Maximum number of images = **5n-5**

Using **Newton's method**, starting from an initial guess, we can iteratively find a root of a polynomial p(z).
To find the next roots the polynomial is divided, **this introduces numerical noise!**

From **VBBinaryLensing** to **VBMicroLensing**

3 different options for mutiple lenses will be made public:

1. SINGLEPOLY
2. MULTIPOLY
3. NOPOLY

SINGLEPOLY

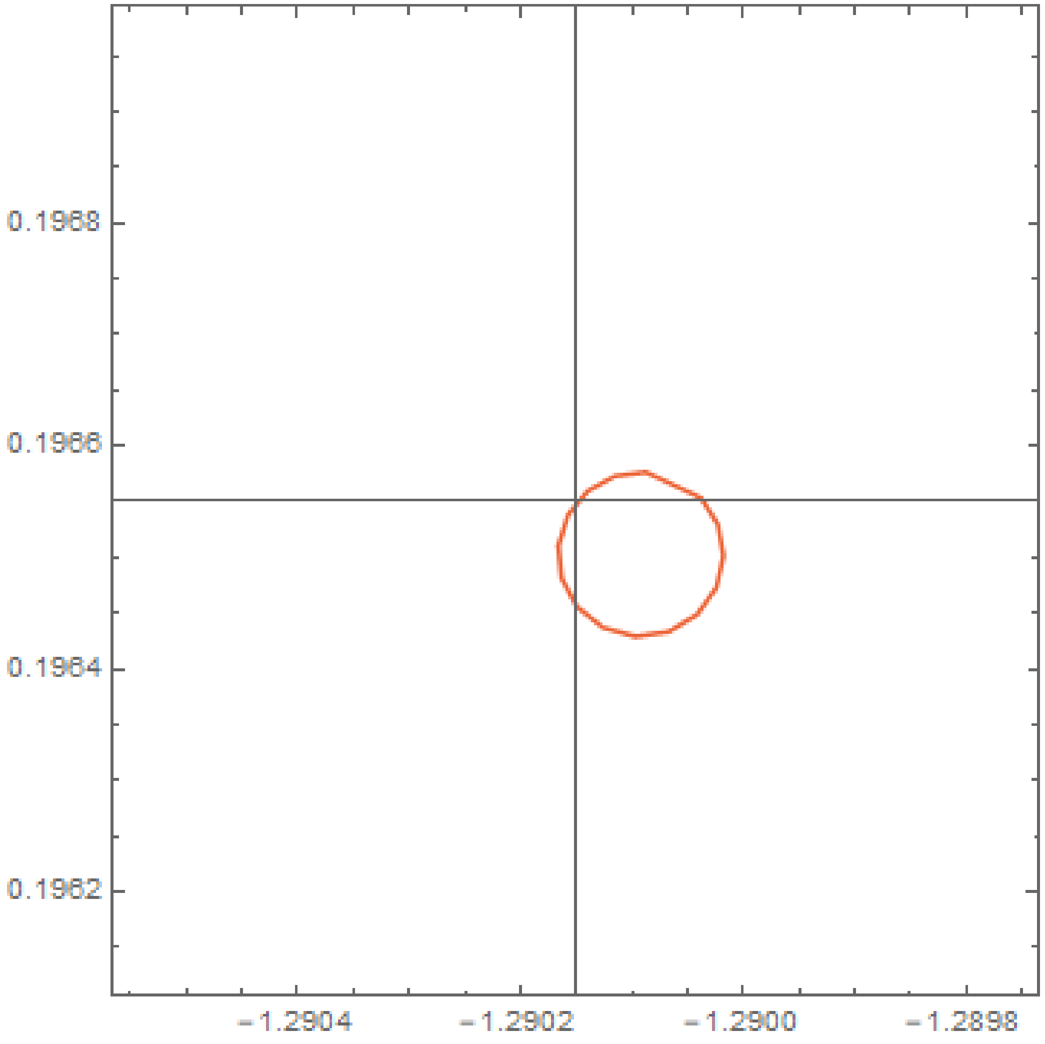
Classical $n^2 + 1$ order polynomial

- Accuracy loss for multiple systems
- Impractical for high n
- All real images are found
- Spourious images, useful to check for nearby folds
- Well-defined computational time

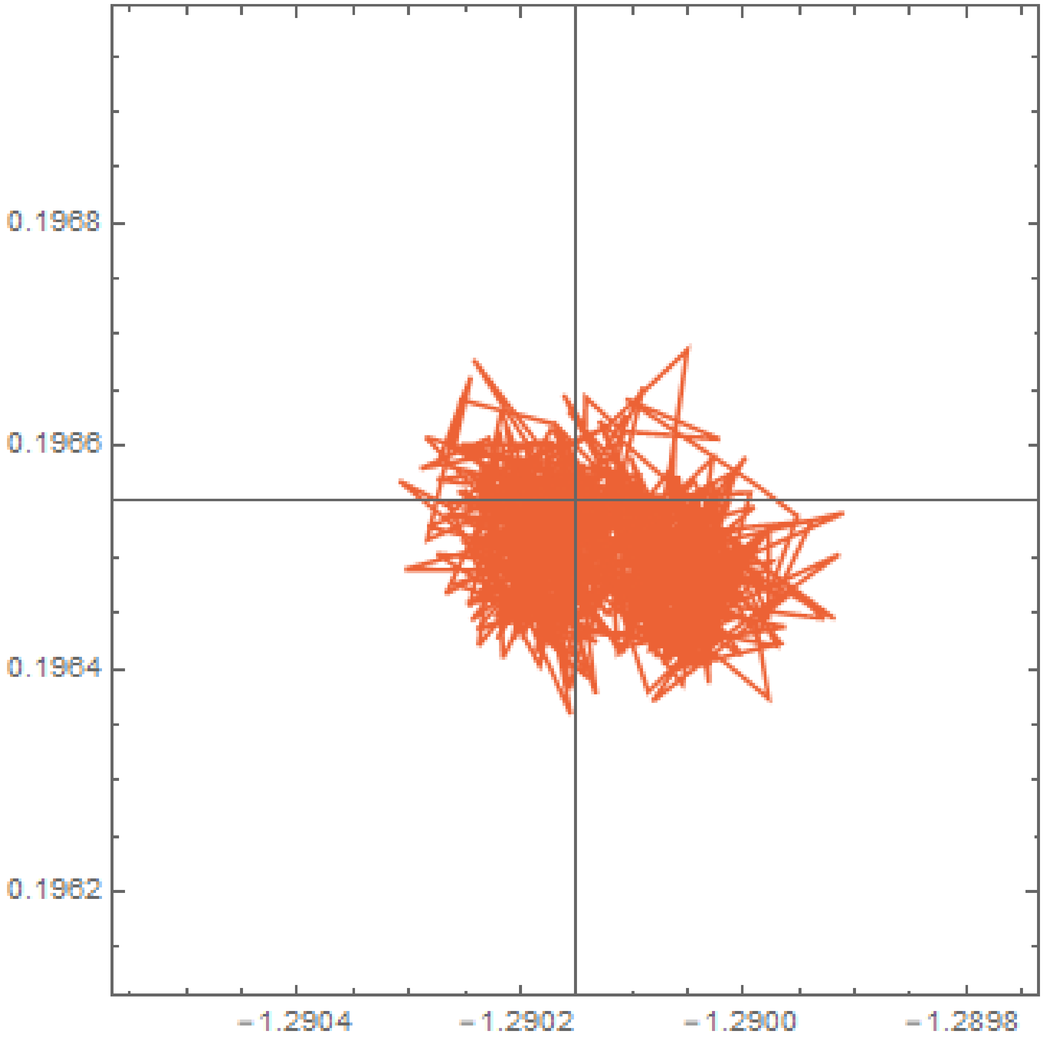
NUMERICAL ACCURACY

Planetary image

n=3



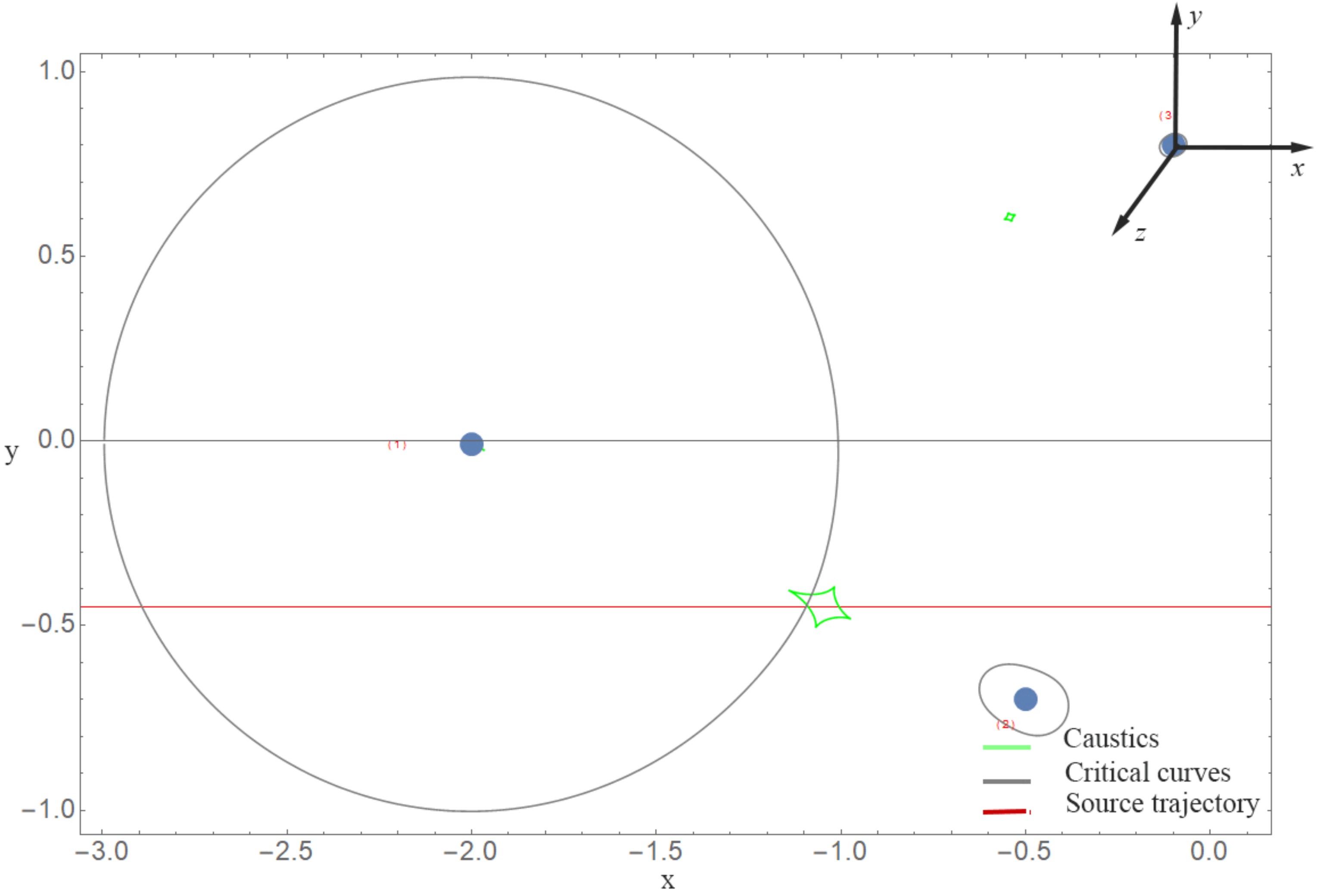
n=4



MULTIPOLY

- The roots with greater accuracy are the first ones found: the images located in proximity of the lens in the center of the reference frame.

- The other roots of the polynomial can be found changing the reference system.



MULTIPOLY

- Accurate roots
- All real images are certainly found
- Spourious images, useful to check for nearby folds
- Longer Computational time

NOPOLY

NEWTON-LIKE METHOD:

Lens equation $\zeta = z - \sum_i \frac{m_i}{\bar{z}-\bar{a}_i}$

- If we are close enough to a root z_0, we can write $z_0 = z + \epsilon$ and expand to first order in ϵ.

- Let us define $L(z,\bar{z}) = \bar{\zeta} + \sum_i \frac{m_i}{z-a_i} - \bar{z}$

- We then have

$$0 = L(z_0,\bar{z}_0) = L(z,\bar{z}) - \epsilon \sum_i \frac{m_i}{(z-a_i)^2} - \bar{\epsilon}$$

- Coupling with the conjugate equation, we find

$$\epsilon = J^{-1}\left[\bar{L} - L\sum_i \frac{m_i}{(\bar{z}-\bar{a}_i)^2}\right]$$

- Images can be found iteratively with this **Newton-like approach**:

$$z_{k+1} = z_k + \epsilon; \qquad \epsilon = J^{-1}\left[\bar{L} - L\sum_i \frac{m_i}{(\bar{z}-\bar{a}_i)^2}\right]$$

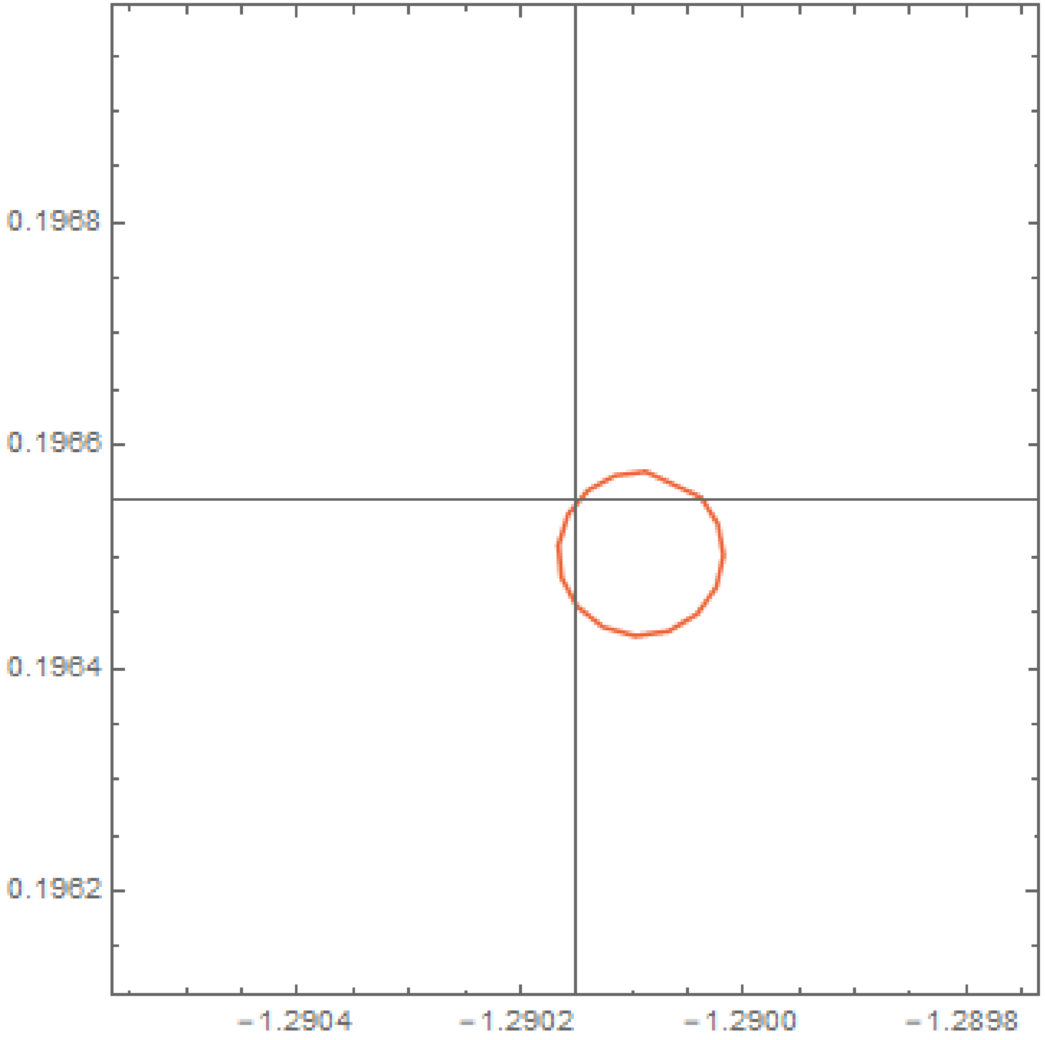
NOPOLY

- Very accurate roots
- Shorter computational time
- Scales linearly for high n
- Never sure that all images are found
- No Spourious images

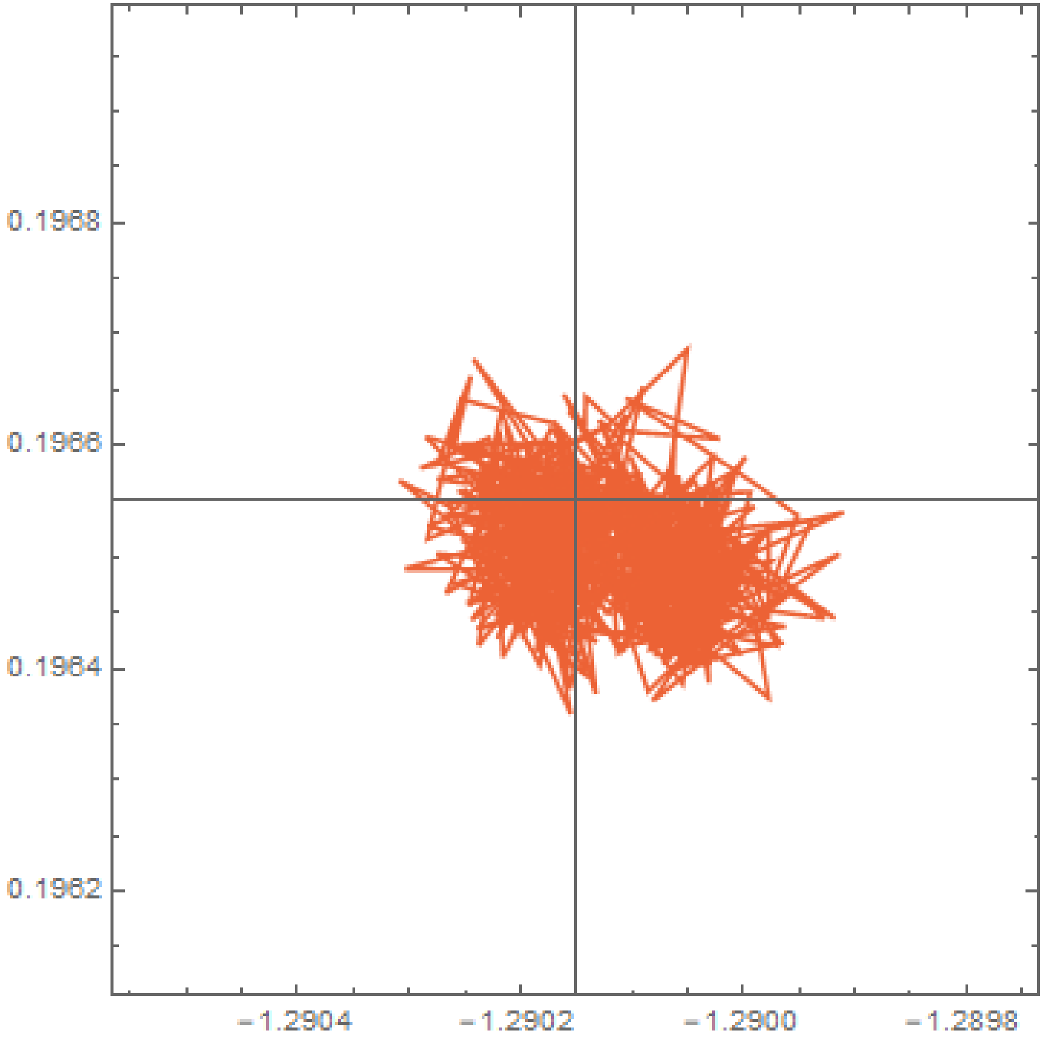
NUMERICAL ACCURACY

Planetary image

n=3



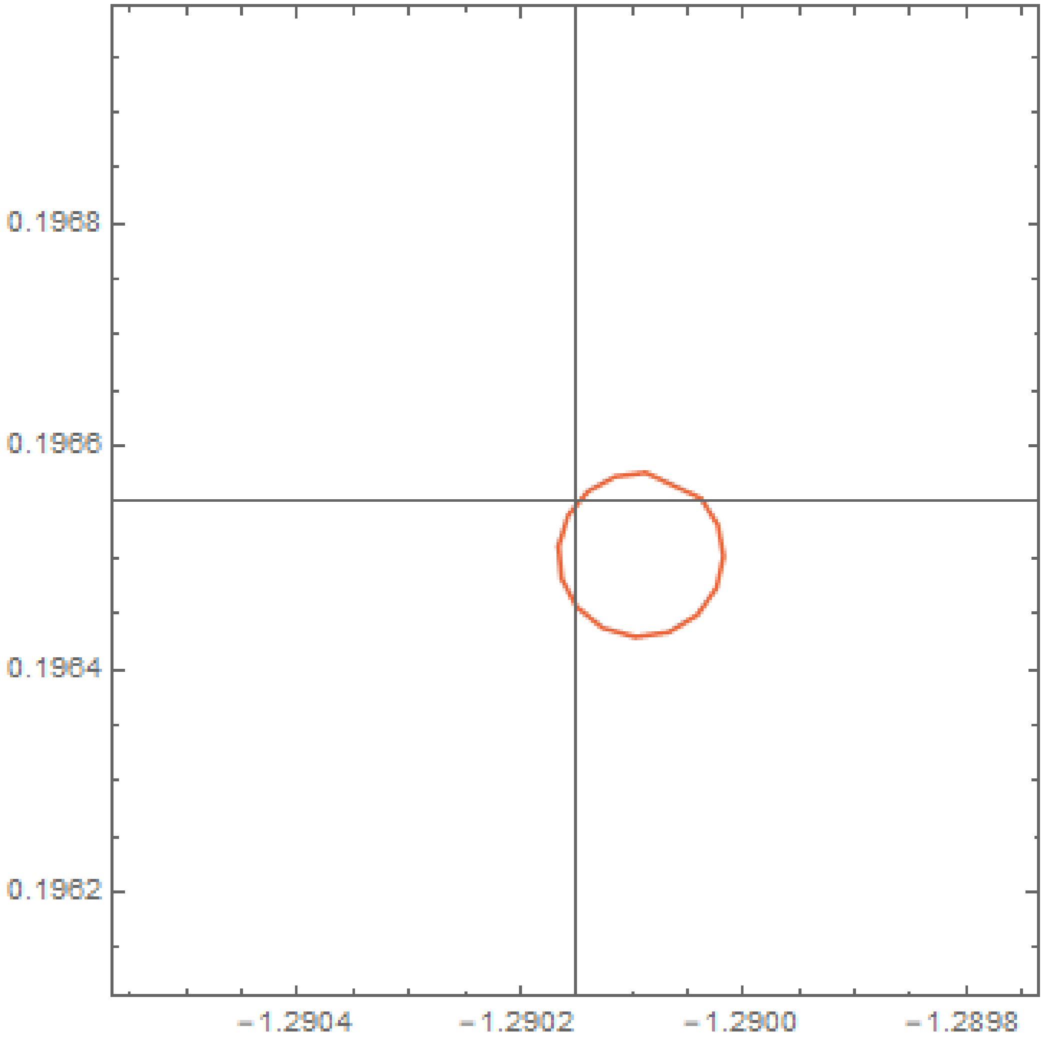
n=4 (polynomial)



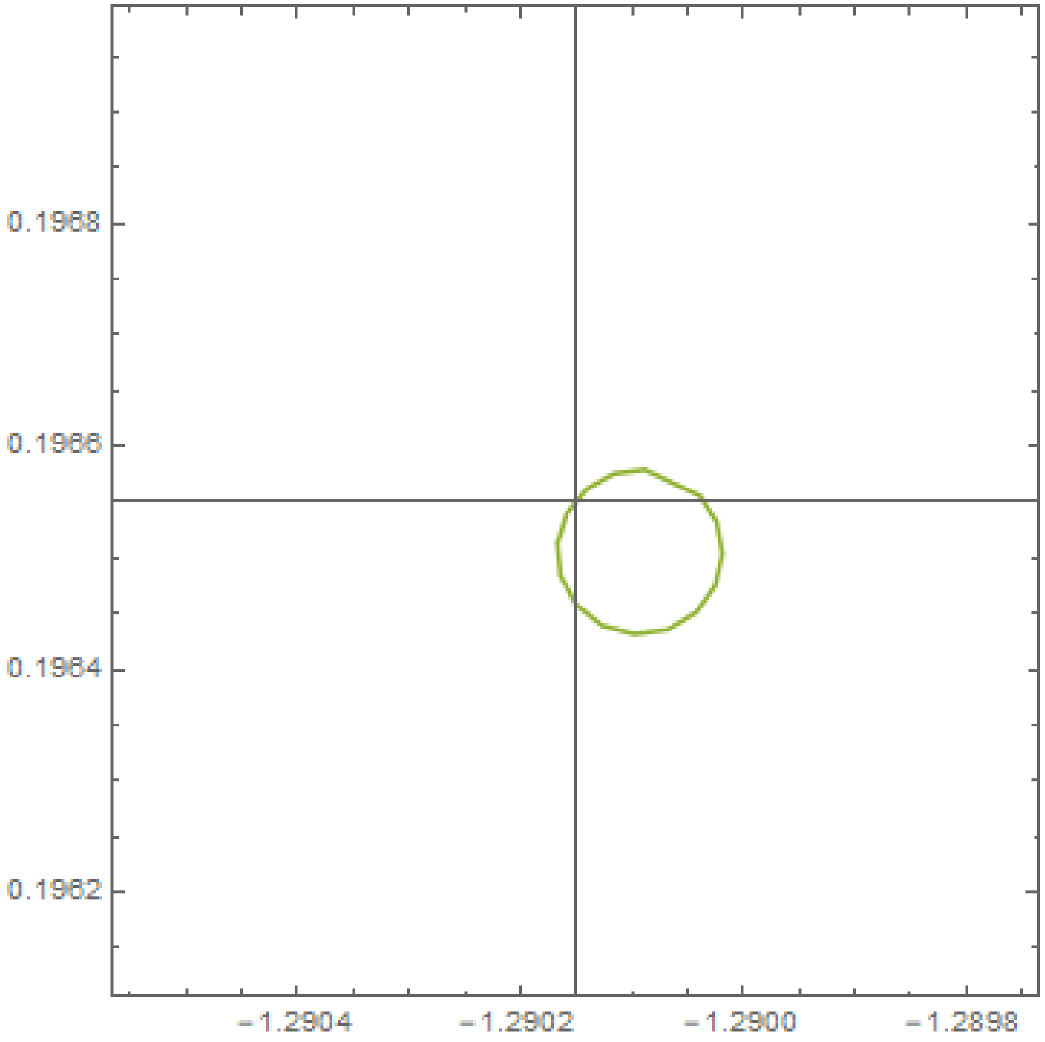
NUMERICAL ACCURACY

Planetary image

n=3

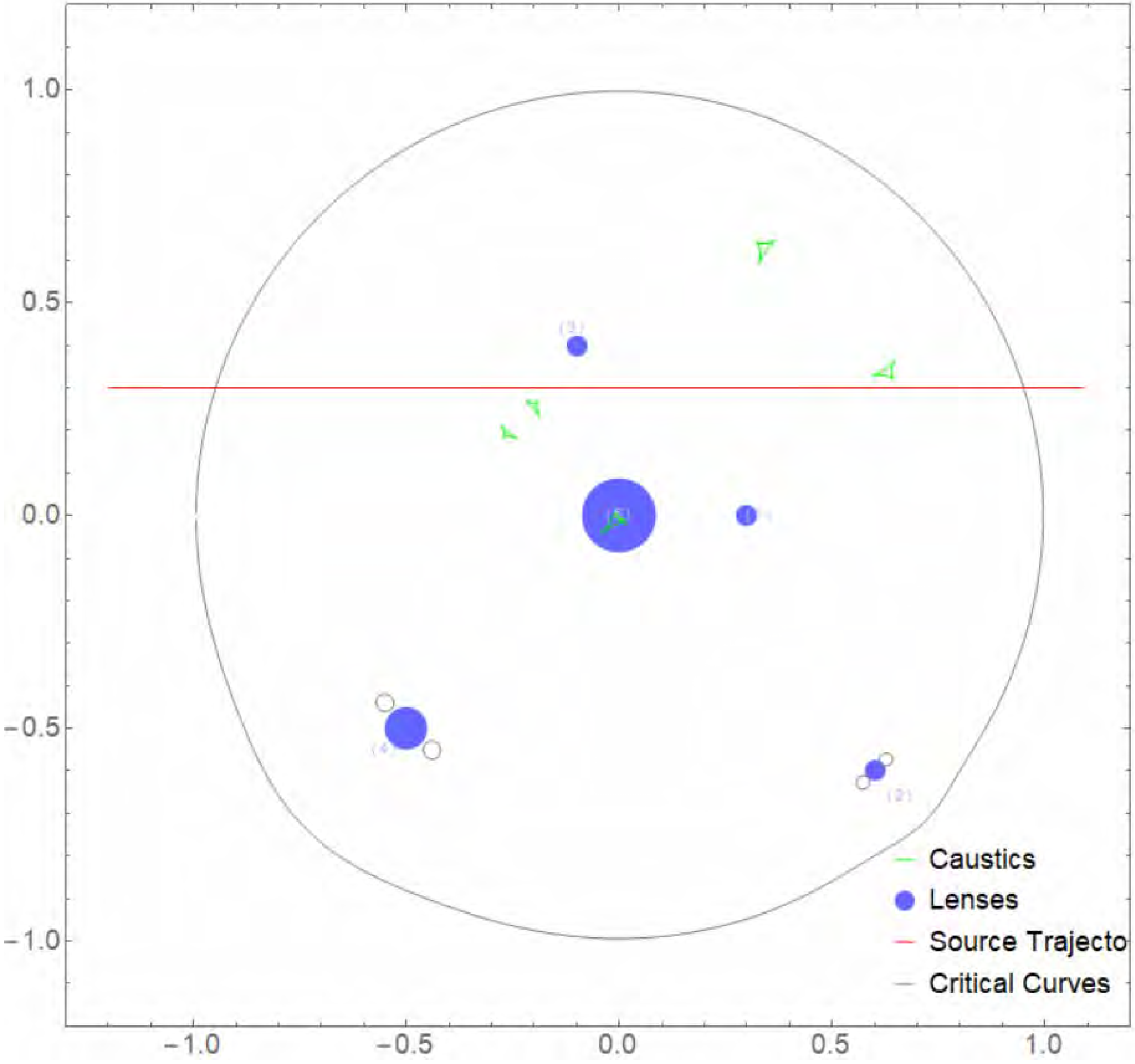


n=4

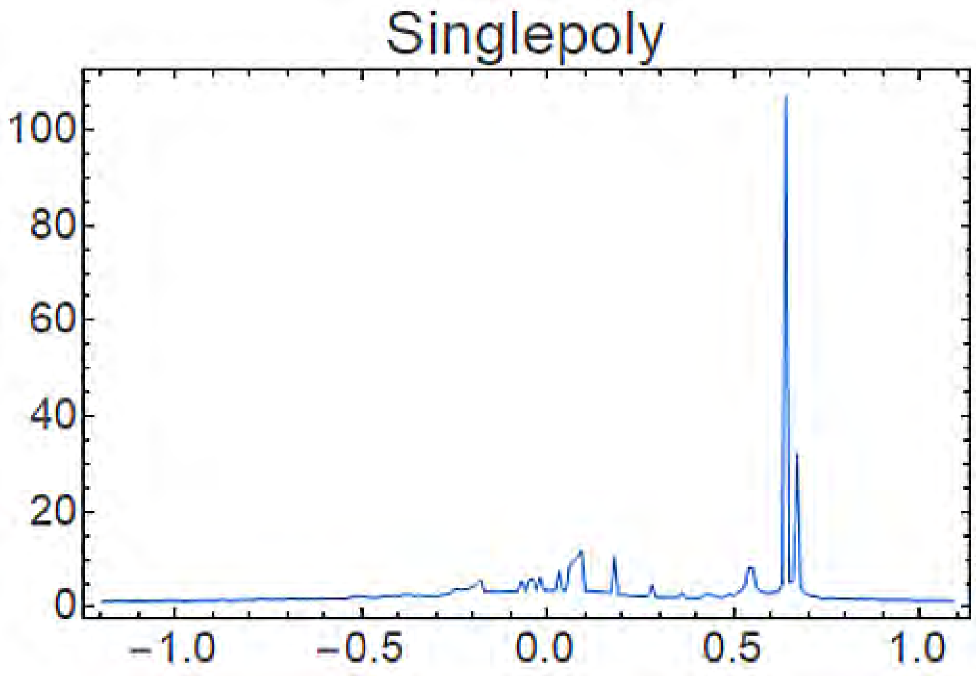


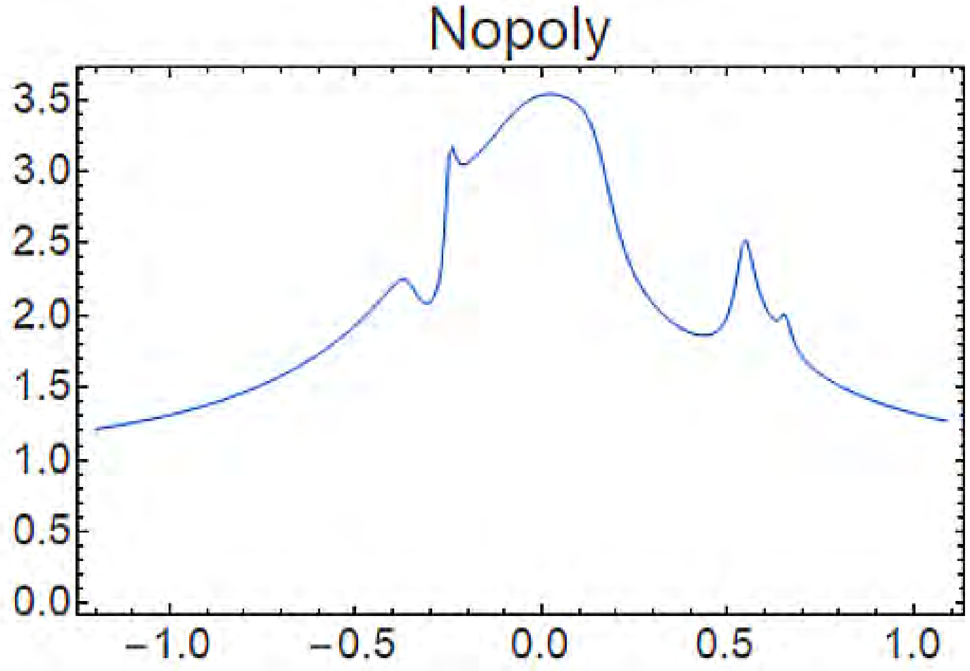
LIGHTCURVES

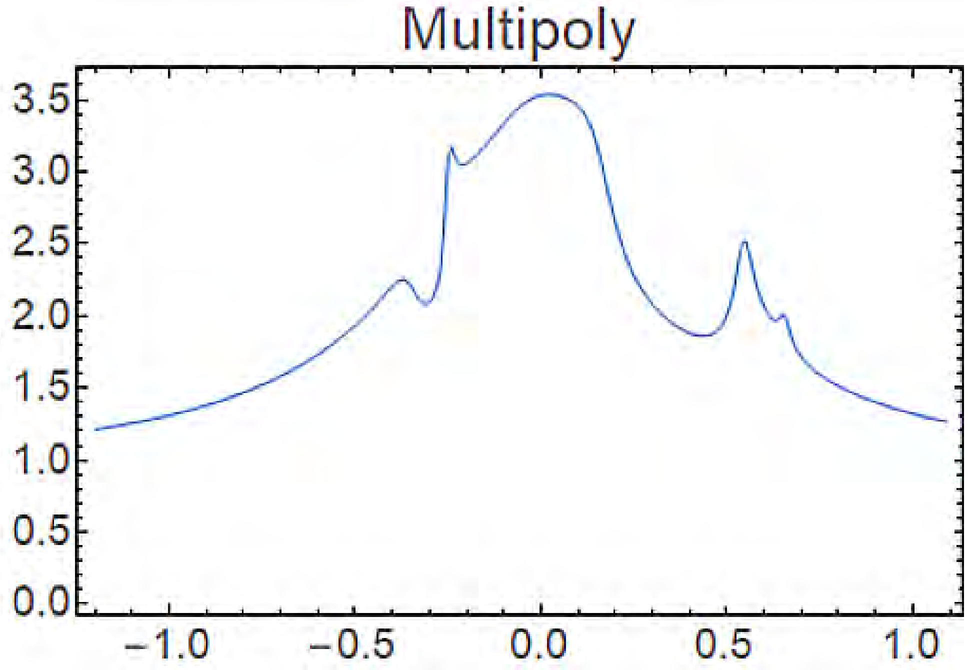
Configuration



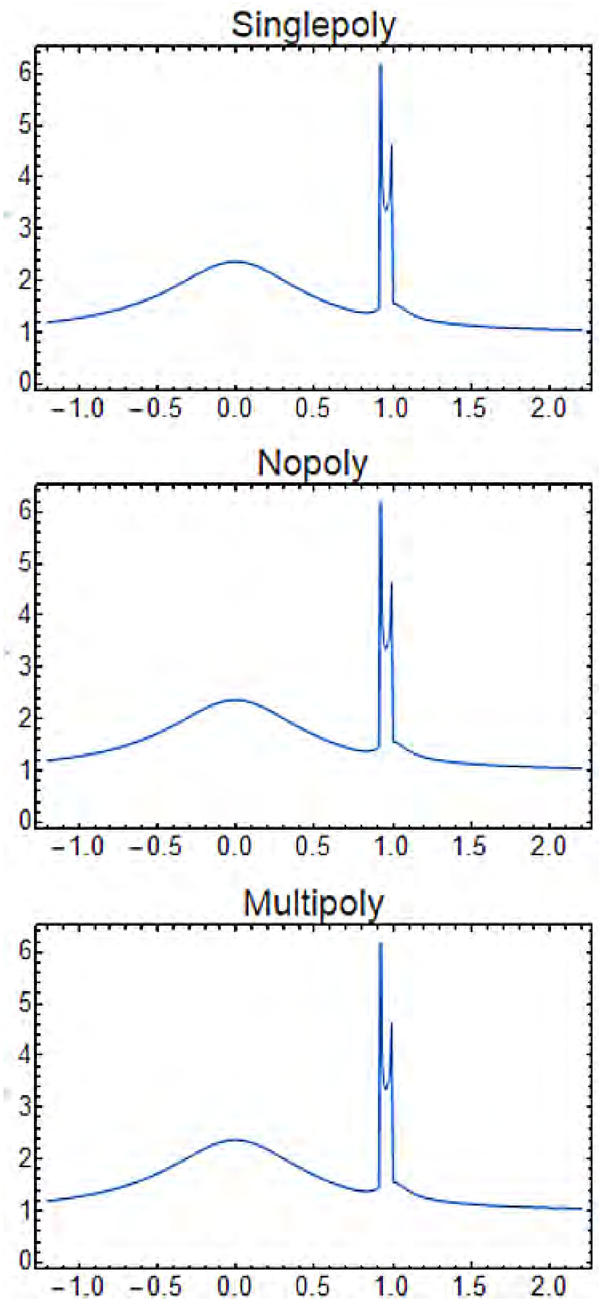
LIGHTCURVES

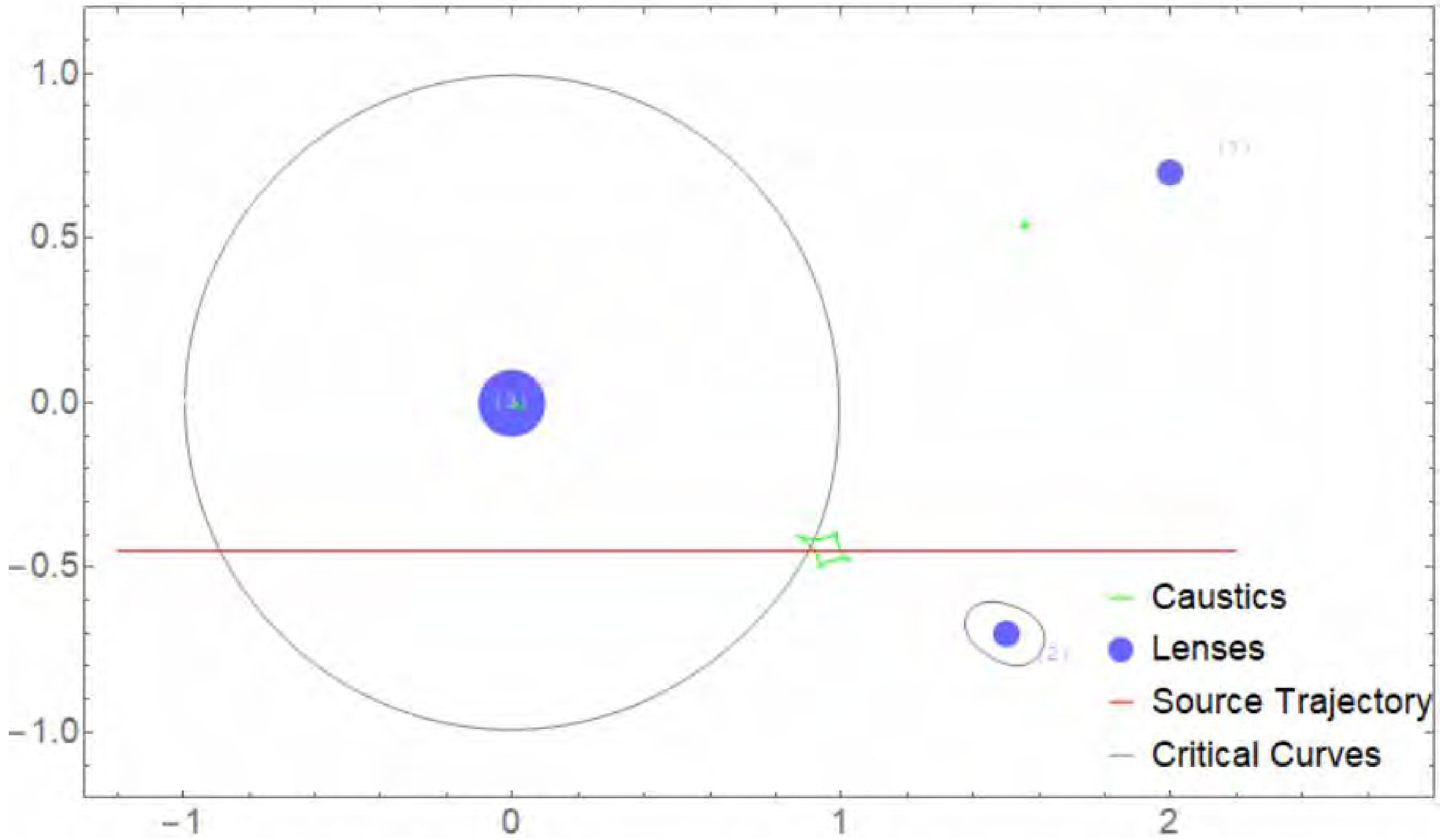




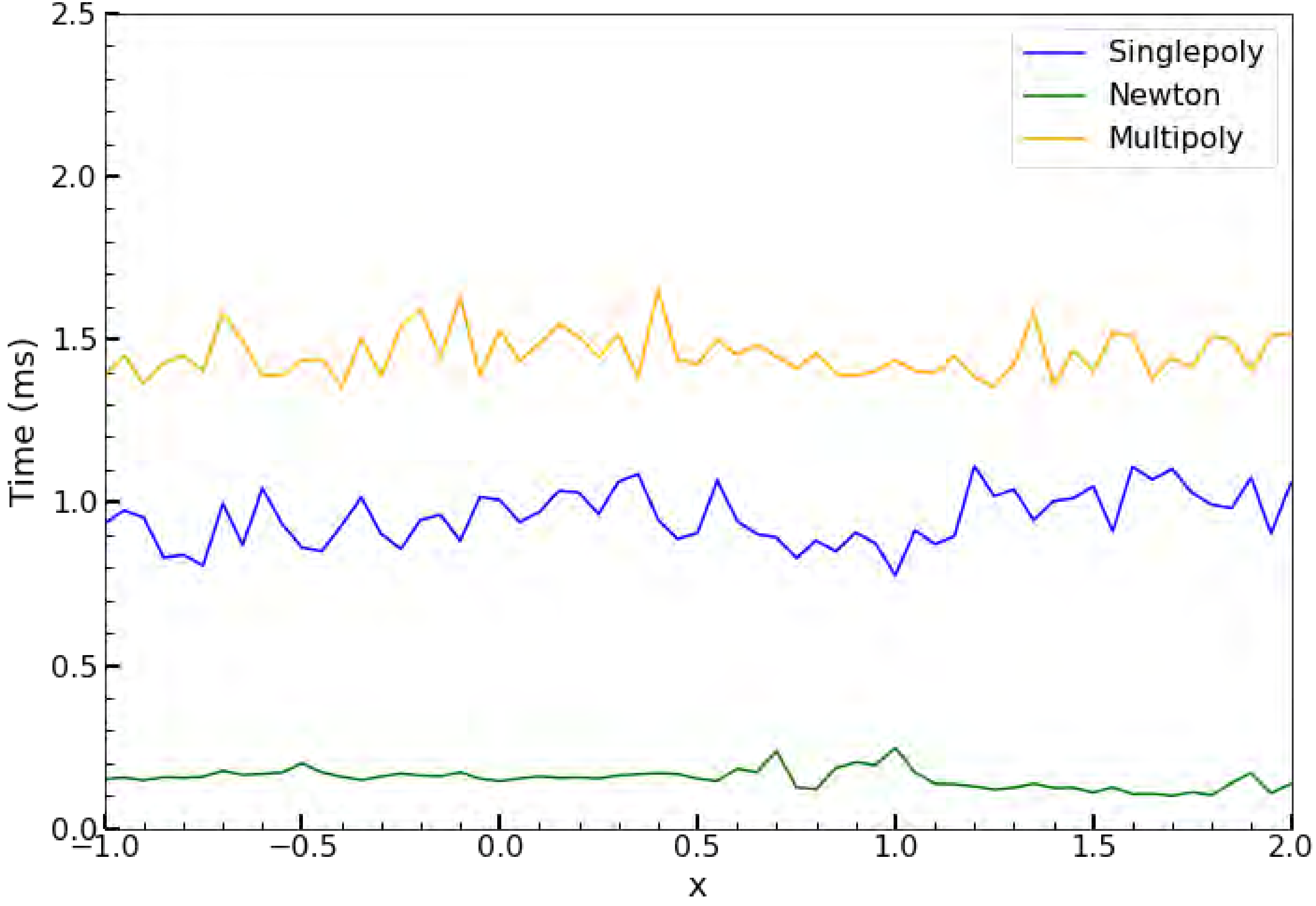


RUNTIME


Singlepoly
Nopoly
Multipoly


Caustics
Lenses
Source Trajectory
Critical Curves

RUNTIME


Singlepoly
Newton
Multipoly
Time (ms)
x
2.5
2.0
1.5
1.0
0.5
0.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

SUMMARY

	SINGLEPOLY	MULTIPOLY	NOPOLY
Accuracy for multi-planet systems	✗	✓	✓
All real images are certainly found	✓	✓	✗
Computational time	✗	✗	✓
Spurious images	✓	✓	✗

FUTURE PROSPETS

Project Infrastructure Team of Roman Galactic Exoplanet Survey;
The Pipeline will be based on VBBinarylensing and RTModel.

THANK YOU!

QUESTIONS?