Wednesday Hands-On Instructions

In this session, you will fit your planet light curves starting with the analytic parameters from Tuesday.

Quick Introduction to pyLIMA and the Sagan notebook

pyLIMA is the first microlensing modeling open source software. It is written in python. You can find a more complete documentation and download it here :

https://github.com/ebachelet/pyLIMA

This python notebook is based on pyLIMA and uses that package to explore and fit a (planetary) microlensing light curve. Using this notebook, you will:

1. Compare the microlensing model generated from your analytic estimation with the data.
2. Manually adjust with the parameters to see how they affect the model.
3. Fit a single lens model to the data (excluding the planetary perturbation).
4. Fit a planetary model to the full data set.

Opening the Notebook

1. Open a VNC connection to the Amazon cloud using the IP address given and your username and password.
2. Open an xterm by clicking
   Applications -> System Tools -> XTerm
3. use xrandr to set the VNC display to match the device screen size. It is important to maximize the screen size in order to maximize the space for displaying plots.
   - The command “xrandr” will list the available screen options.
   - As an example, if you screen is 1920x1080 pixels, use the command
     ```
     xrandr -s 1920x1080
     ```
     to maximize the size of the VNC display.
4. Guide to directories:
   - Your default directory is your home directory /nwhome/userxxx/
   - The /nwhome/userxxx/examples/
   - The data are located in /ssw/data/
5. Go to the examples directory, i.e.
   ```
   cd examples
   ```
6. Setup python using the command
   ```
   source activate astroconda
   ```

(if you miss this step, the notebook will load, but not run. Trust me, I tried it several times.)

7. Open the notebook using

   jupyter notebook SAGAN_binary_3.ipynb

   This opens a browser window with the notebook.

General notes on using the notebook:

- **The better the initial guess, the better the final fit will be.**
- Run each cell sequentially.
- To run a cell, **click** on it and then use the command

  Shift+Enter
- Places you may need or want to modify the code for your particular light curve are marked with

  ###**CHANGE****###**

  ###**ENDOFCHANGE****###**
- Read the comments. You may find them helpful.
- If for some reason, you have messed up your examples/ directory and want to start over with the original version, copy /ssw/pyLIMA-master/examples/ to your home directory, i.e.

  cp -r /ssw/pyLIMA-master/examples .

There are a total of 7 cells in the notebook. Below are instructions for each one.

Cell 1: Import the required libraries/modules

Just run this. There is nothing to modify.

Cell 2: Import and view the light curve file

Referring to the handout, adjust the file path for the file corresponding to your event. Specifically, change the line:

/ssw/data/WFIRST_SAGAN_X/WFIRST_binary_SAGAN.dat

where “X” → your group number and “WFIRST_binary_SAGAN.dat” → your filename. The document gl_planets_X.txt matches your event name to the proper filename. Note that the event name also encodes your group number, e.g.

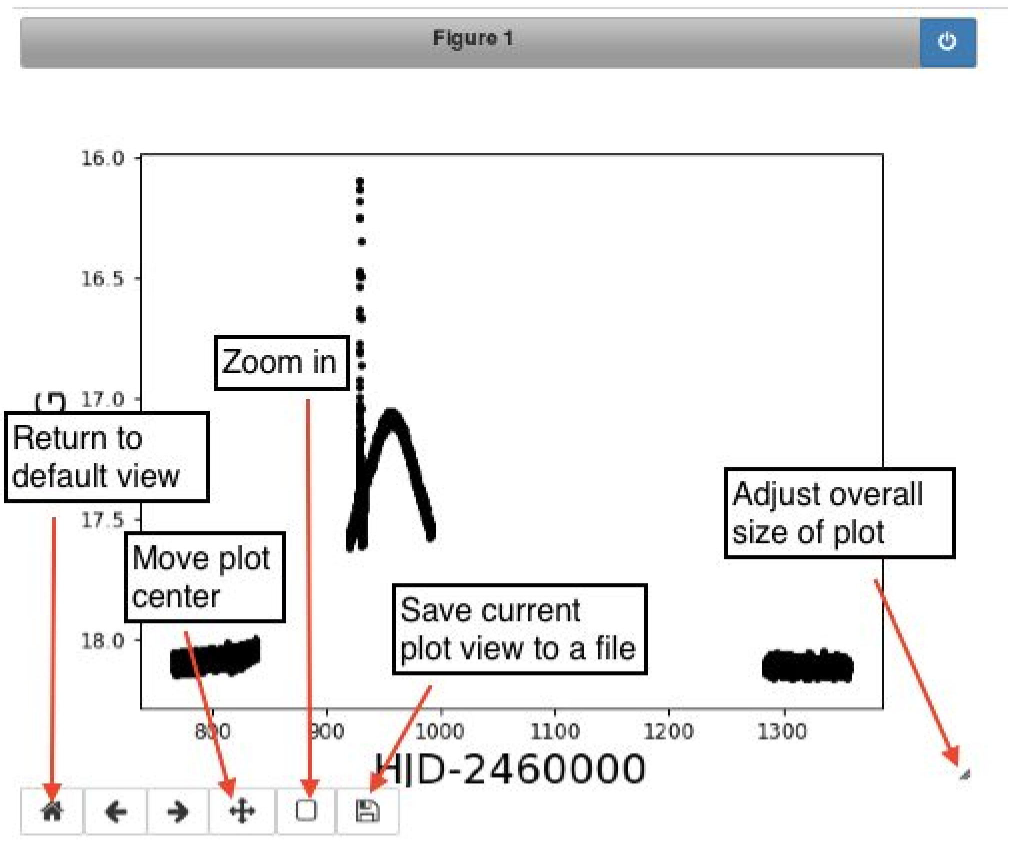
Suppose your event is ***Event_2_05***. Then, you are in group ***2***, so you would open the file ***gl_planets_2.txt***, and look for the line:

2_05 WFIRST_0915.dat GLPlanet

WFIRST_0915.dat is your data file. So you should modify the path to be
/ssw/data/WFIRST_SAGAN_2/WFIRST_0915.dat

Adjusting the plot

When you run this cell, it will produce a plot of the light curve data. You can adjust the plot using the features indicated on the image below:



Cell 3: Setting t_anomaly and delta_t

t_anomaly = the approximate midpoint of the anomaly. You can use the value of t_planet from Tuesday.
delta_t = the approximate duration of the anomaly.

These variables are used when fitting a point lens light curve to mask out the data during the planetary anomaly. They are also used later when resampling the data prior to doing the full 2-body lens fit. After setting these values, run the cell as is. You can reassess them after running Cell 5.

Cell 4: Initial guess for the microlensing parameters

In this box, you define the parameters of the initial model light curve and create a pyLIMA "event.Event()" object. Then, the rest of the cell sets up the model light curve using your initial parameters and plots both the light curve and the caustics with the source trajectory.

Defining the parameters is probably the most important step in the entire notebook. If the initial guess is really bad (e.g., the source trajectory doesn't pass through the caustics), the fit will fail. The purpose of this box is to assess the initial parameters by examining the light curve (**black** points) and microlensing model (**blue** curve) in detail. Adjust the parameters manually to improve your initial guess. Specifically,

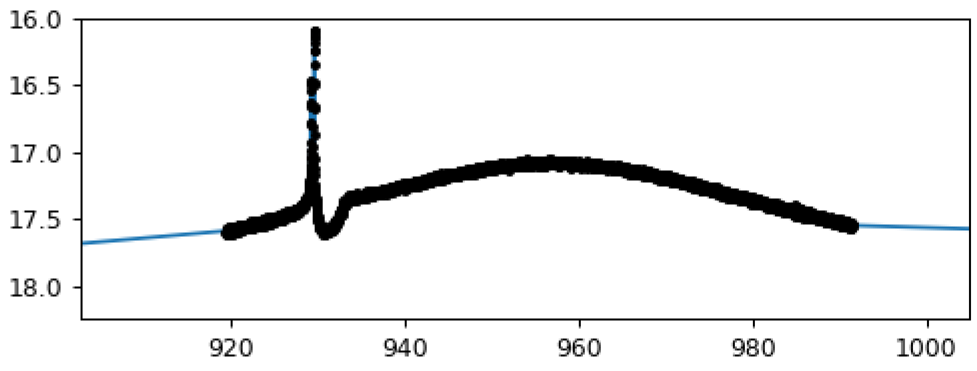
1. Set the values of **to, uo, tE, rho, s, q,** and **alpha** using your analytic estimates from the Tuesday sesion.
2. Run the cell using **Shift+Enter**.
3. Look at the output light curve. Does the model look "good"? Follow the questions below to answer this question.
4. Adjust the parameters one at a time to get a better input model. Adjust only one parameter at a time. Then re-run the cell and re-examine the light curve. You may want to take notes about what values you tried for each parameter.
5. (Optional) you can change the name of your event, i.e.
   your_event.name = 'WFIRST binary'
   You can call it anything you want, e.g. "Henrietta Levitt's favorite microlensing event"

Below are some common problems and how to solve them. They are given in the order in which they should be addressed. Your fit does not need to be perfect, but the closer it is, the better the final result. Spend no more than 10 minutes adjusting parameters. The reason we do numerical fits using computers is that it is hard to adjust the parameters by hand.

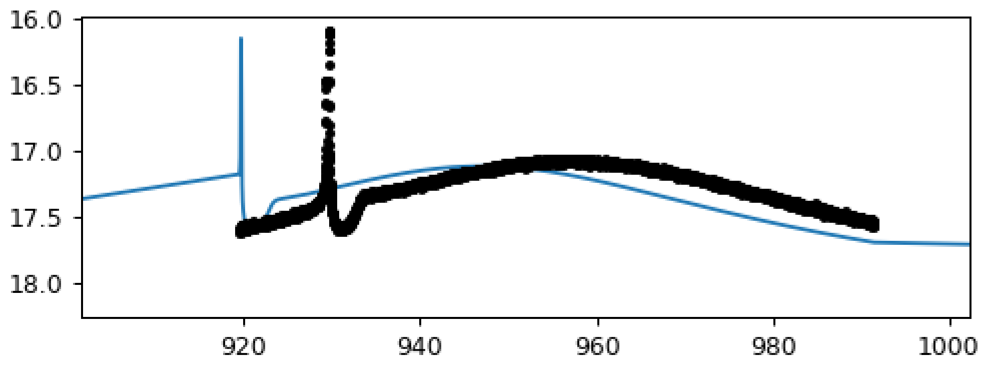
*Are the point-lens parameters (**to, uo, tE**) okay?*

The analytic estimates of these should be pretty good. For completeness, the pictures below show some possible adjustments. As seen from the pictures below, adjustments to **uo** may be degenerate with adjustments to **tE**.

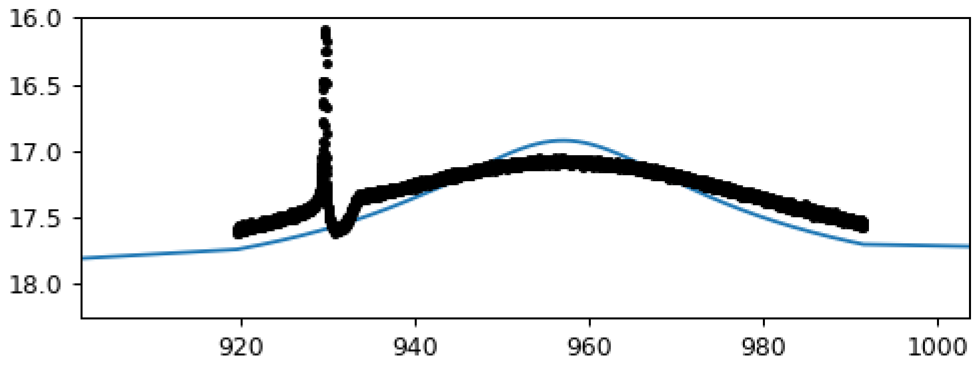
A good fit:



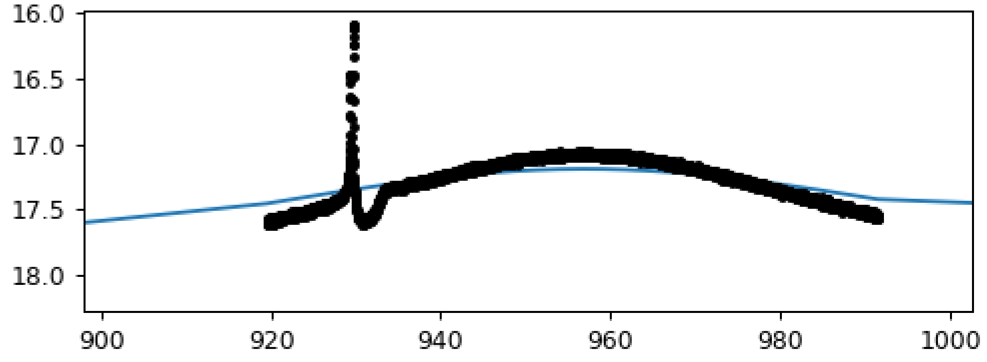
t0 is too small:



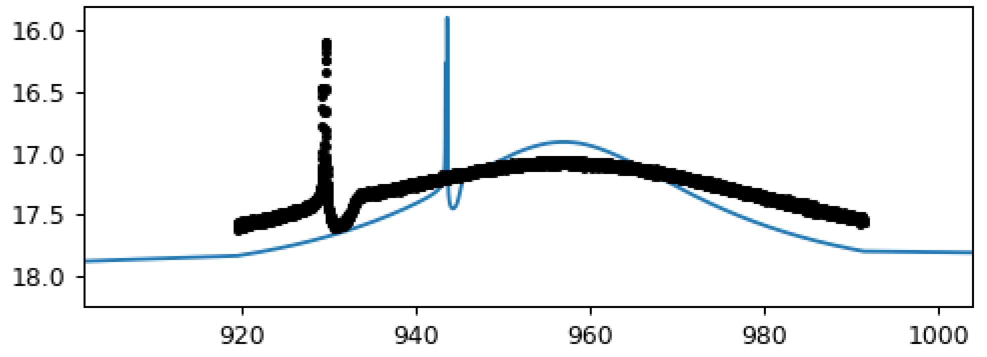
u0 is too small:



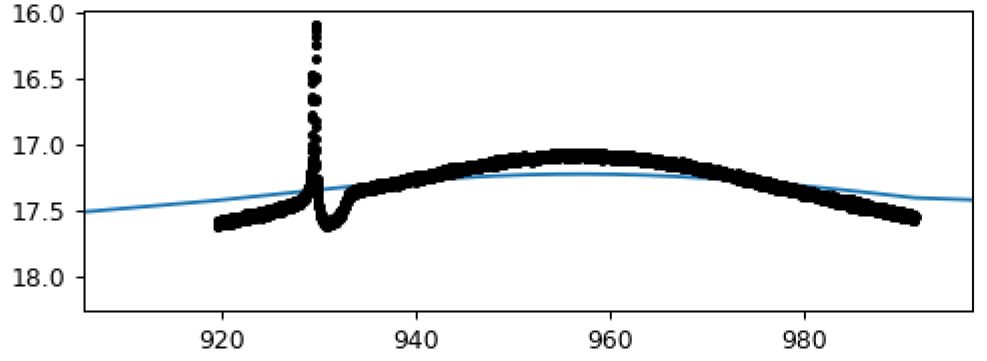
u0 is too large:



tE is too small:



tE is too large:



Is there a planetary perturbation?

1. Top panel: Zoom in on the planetary perturbation. Does the model have some kind of perturbation there?
2. Bottom panel: Does the source trajectory (**blue**) go through the caustics (**red**)?

If the answers are no, the problem is either with **alpha** or **s**. *Focus on the caustics and source trajectory: the source trajectory should go through the caustics.*

First, check **alpha** which affects the *angle of the source trajectory*:

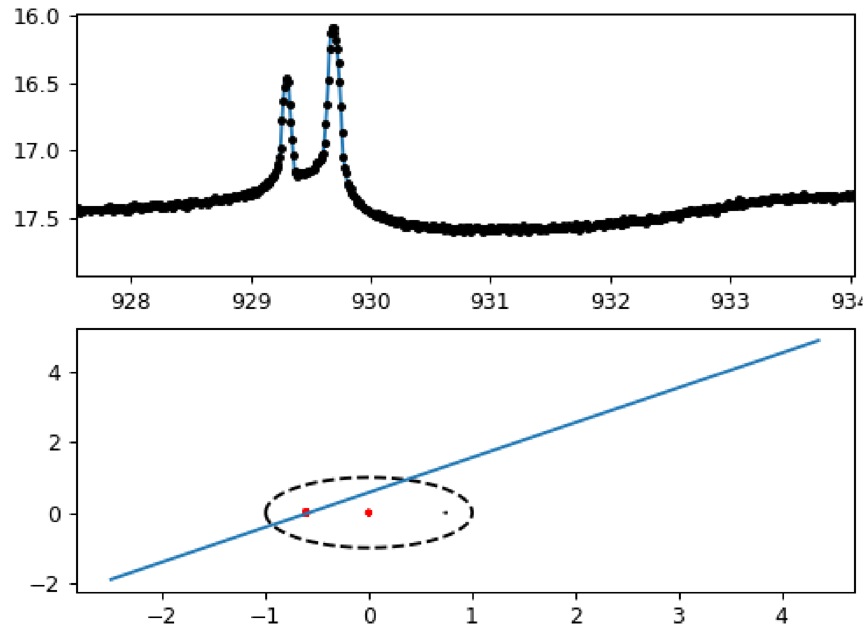
- **alpha** should be in RADIANS.
- Try inverting the sign of **alpha**
- Try adding π or subtracting π from **alpha**

Alternatively, the problem may be with **s**. **s** affects the *position of the caustics*.

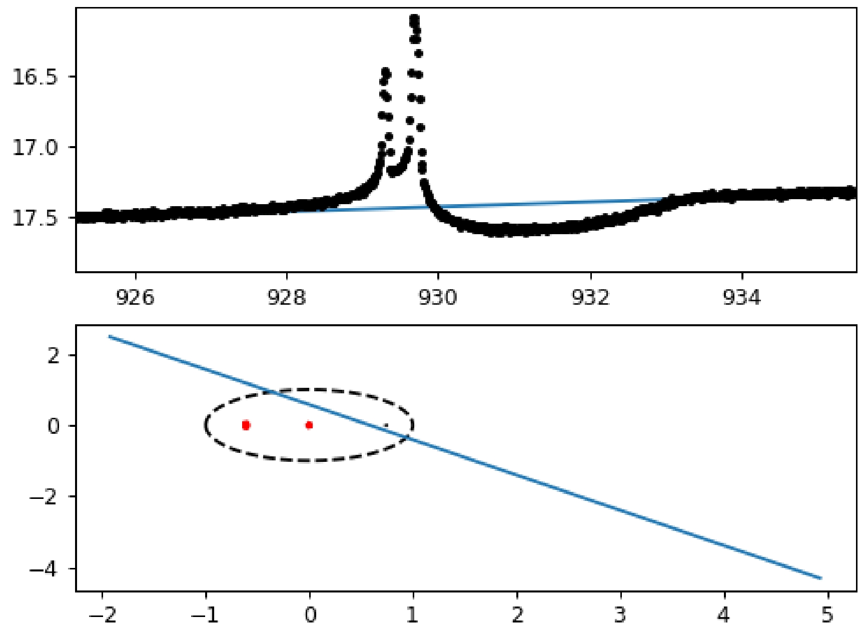
The diagrams starting on the next page show possible adjustments to **alpha** and **s**. For **s**, the diagrams showa minor image perturbation. For a major image perturbation, **s** may need to be changed in the opposite direction (because the caustics are on the opposite side of the lens star). In either case, try both increasing and decreasing **s** if necessary.

The exact timing of the caustic crossing in the model depends on both of these parameters simultaneously (and also **tE** and **uo**). Once you have a planetary perturbation, changing the timing of this perturbation requires changing both **s** and **alpha** in concert. It is not necessary to optimize this, just to make sure there is a perturbation of the right form within a few days of the perturbation in the data.

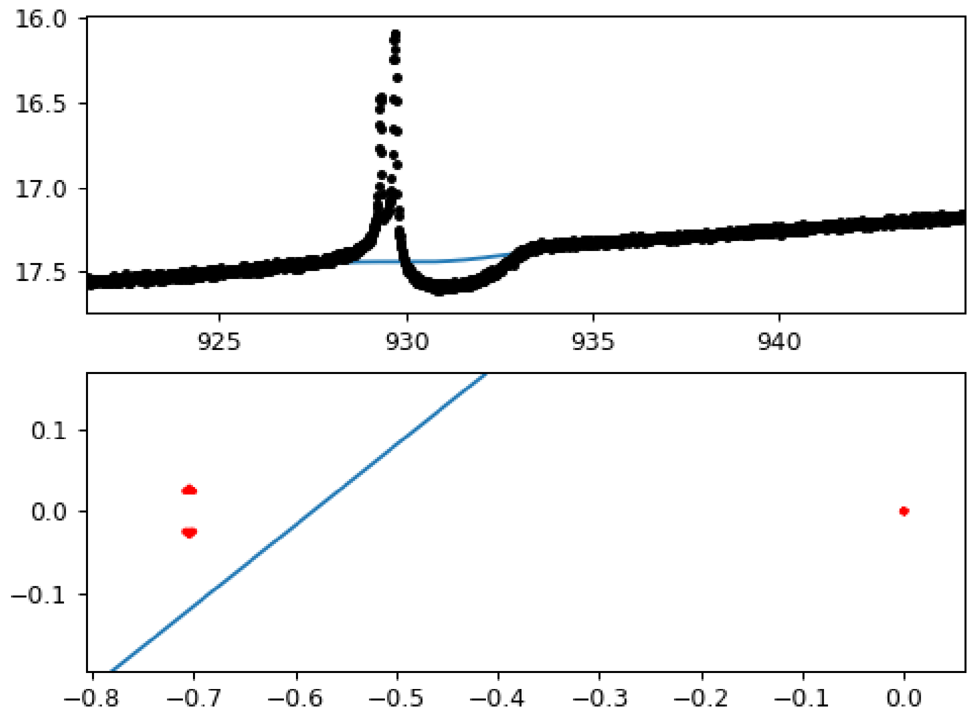
Good alpha (source trajectory goes through the caustics):



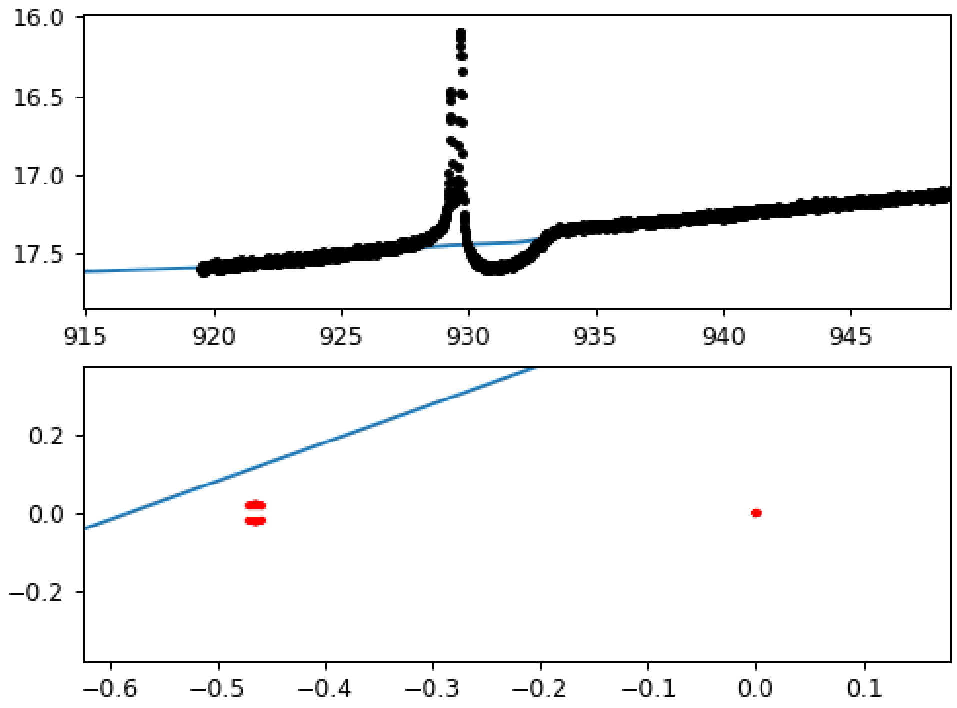
Bad alpha (source trajectory on the opposite side of caustics):



s too small (source trajectory passes inside the caustics):



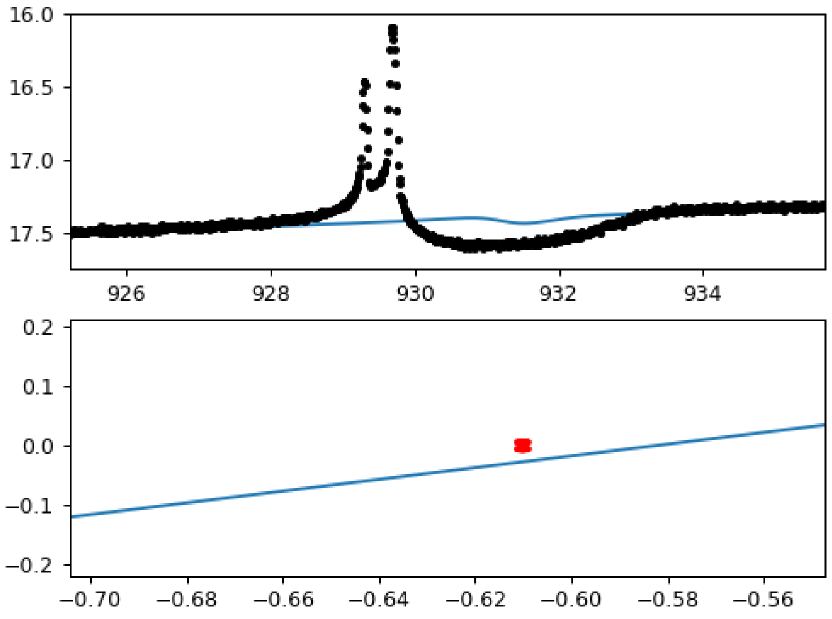
s too large (source trajectory passes outside the caustics):



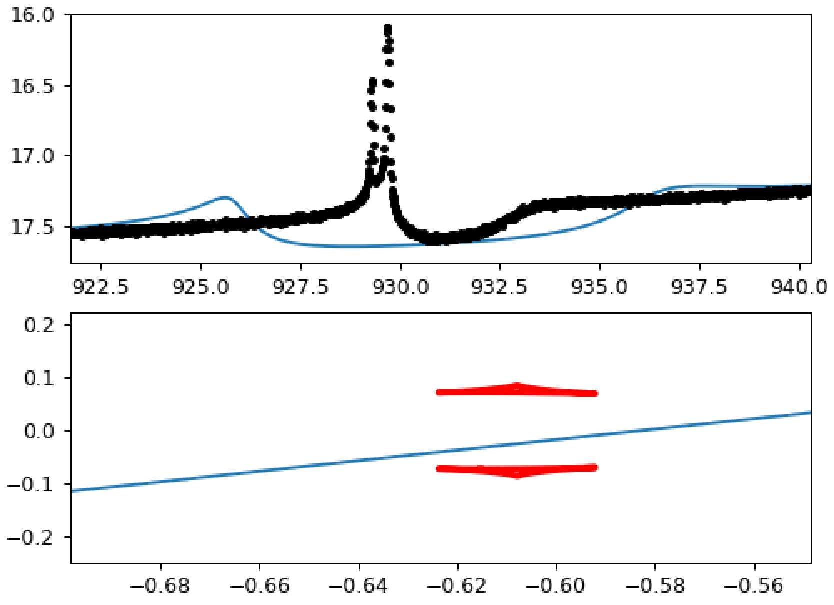
Is the planetary perturbation the right size?

The size/duration of the planetary perturbation is affected by the size of the *caustics*, which in turn depends on the value of **q**.

q too small:



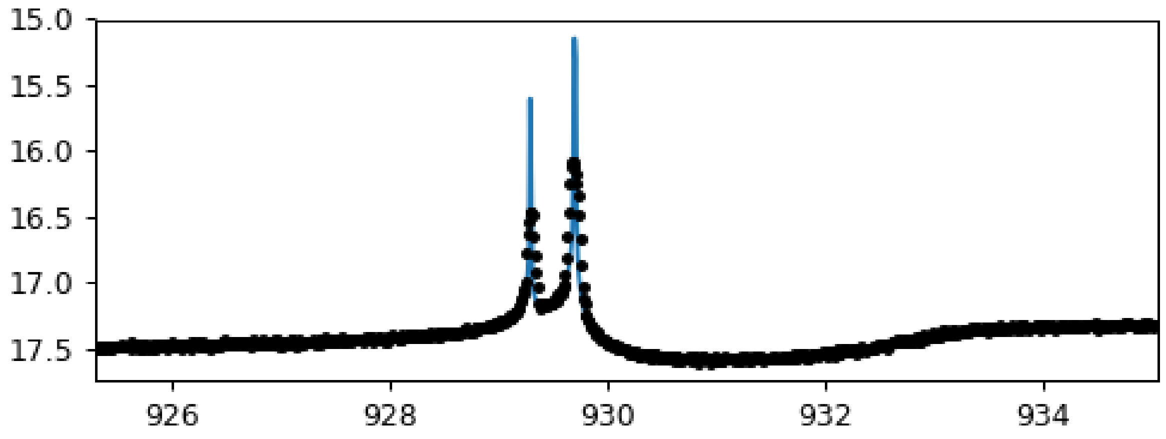
q too large:



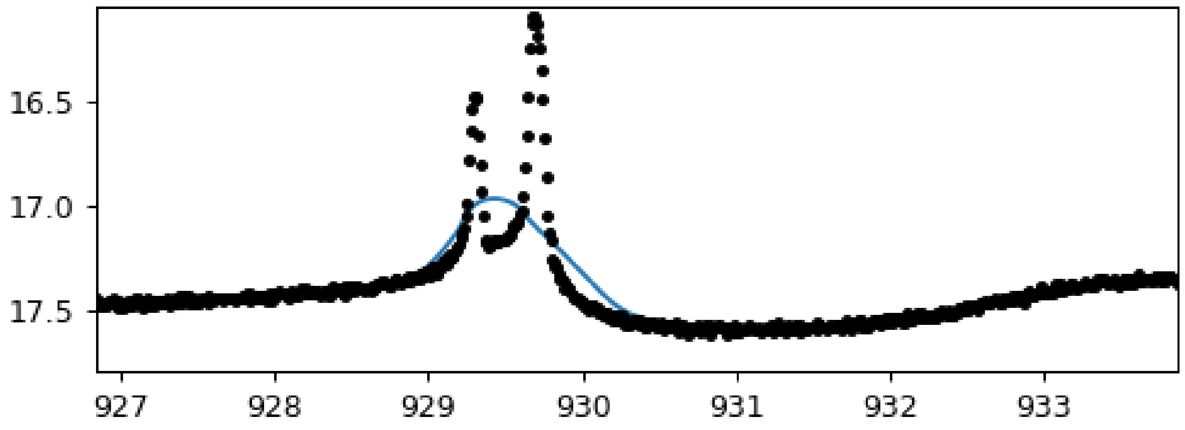
Is the amplitude of the planetary perturbation close to correct? Does the planetary perturbation have the right number of features?

The finite source effect, controlled by **rho** (the source size), affects how rounded the caustic crossings are and also the maximum magnification of the caustic crossing.

rho too small (model is peakier than the data):



rho too large (caustic crossings have merged because rho is bigger than the caustic size):



Cell 5: PSPL Fit

This cell fits a PSPL model to the light curve data. First, it excludes the planetary perturbation from the fitting using your values of **t_anomaly** and **delta_t** from Cell 3, i.e. it sets

```
        good = np.where((WFIRST_data[:,0]<t_anomaly-delta_t) |
                        (WFIRST_data[:,0]>t_anomaly+delta_t))[0]
```

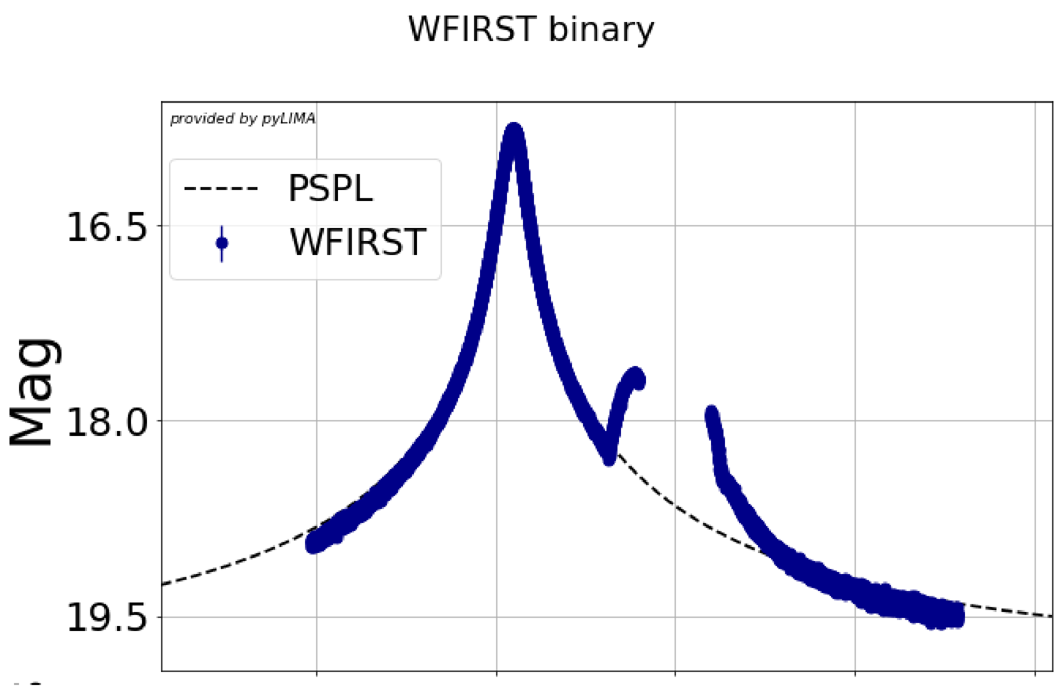
The key here is to assess whether or not the planetary perturbation has been sufficiently excluded. If the perturbation is large and not properly excluded from the fitting, it could skew the PSPL fitting. If too much of the light curve is excluded, the PSPL fit may fail due to insufficient information. These problems are more likely to arise if the planetary perturbations are large.

To address these problems,

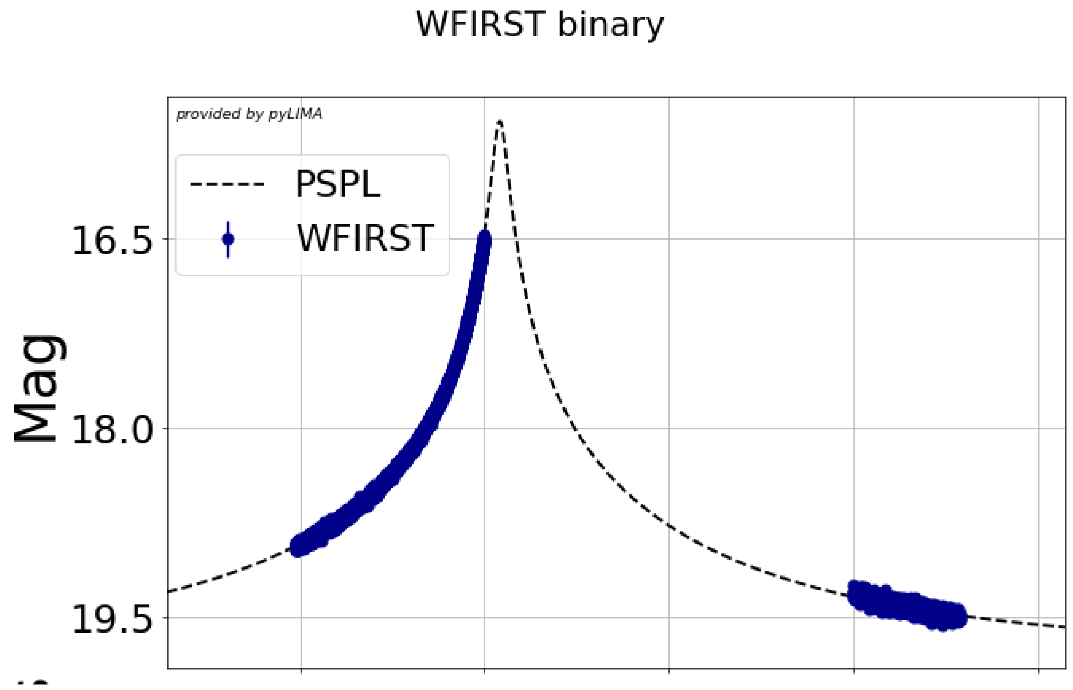
1. Do NOT change anything in this cell.
2. Instead, change **t_anomaly** and **delta_t** in ***Cell 3***
3. Re-run both Cells 3 and 4
4. Re-run this cell and re-evaluated

Also, keep in mind that the amount of the light curve that is excluded is **t_anomaly** + [**-delta_t**, **delta_t**] (so the width is TWICE the value of **delta_t**). Below are two examples that require changes to **delta_t**.

delta_t too small (perturbation still obvious):



delta_t too large (peak is lost unnecessarily; this will affect the accuracy of the fitted uo and tE):



Cell 6: Setting to, uo, tE based on PSPL fit

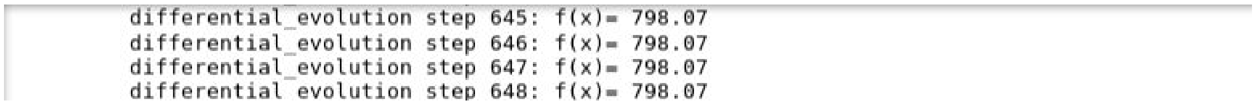
After checking the output from Cell 5, if the PSPL fit is good, just run this box with no changes.

Cell 7: Binary model fitting

Using **to, uo, tE** from the PSPL fit and **rho, s, q,** and **alpha** from Cell 4, this cell will fit a binary lens model (i.e. a 2-body lens model with a star and planet) to the full data set. Here is a brief description of what it does:

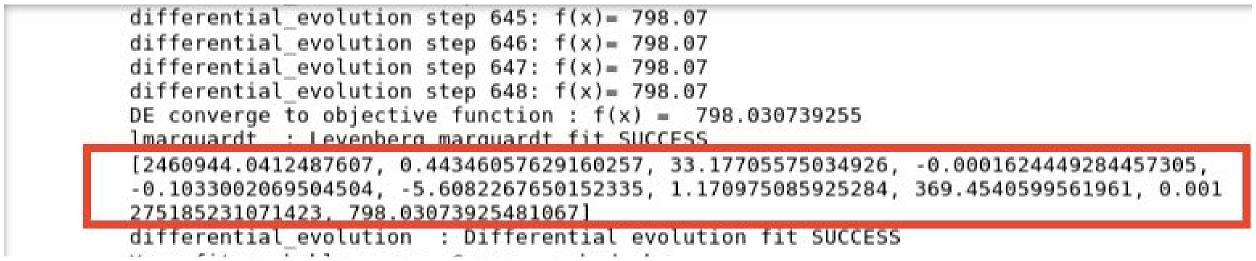
1. It extracts a subset of the data to speed up the computations. Outside of the anomaly, it uses every 32nd point. During the anomaly, it uses every Nth point where N = **anomaly_step_size**, which in turn depends on **delta_t**.
2. It sets the parameter boundaries based on the microlensing parameters described above and some default ranges. These are the lines "**binary_model.parameters_boundaries[0]**", etc. These boundaries have been deliberately set with restrictive ranges to ensure the code runs in a reasonable amount of time.

3. It will fit a binary model to the data using a differential evolution routine (specifically scipy.optimize.differential_evolution). While this is running, the code will produce a series of lines



```
differential_evolution step 645: f(x)= 798.07
differential_evolution step 646: f(x)= 798.07
differential_evolution step 647: f(x)= 798.07
differential_evolution step 648: f(x)= 798.07
```

4. It will produce the fitted values (highlighted in **red** below), the light curve of the data, and fitted model.



```
differential_evolution step 645: f(x)= 798.07
differential_evolution step 646: f(x)= 798.07
differential_evolution step 647: f(x)= 798.07
differential_evolution step 648: f(x)= 798.07
DE converge to objective function : f(x) =  798.030739255
lmarquardt  : Levenberg marquardt fit SUCCESS
[2460944.0412487607, 0.44346057629160257, 33.17705575034926, -0.0001624449284457305,
-0.1033002069504504, -5.6082267650152335, 1.170975085925284, 369.4540599561961, 0.001
275185231071423, 798.03073925481067]
differential_evolution  : Differential evolution fit SUCCESS
```

The recommended approach is to just **try running this box as written, with no changes**.

When you get a ValueError...

The notebook will likely “throw an exception” (raise a ValueError). The most common type of error has the following form:

> **ValueError: [Upper/Lower] boundary for [parameter] too [small/large]! Make it [larger/smaller].**
> **Your boundaries: [X, X]**
> **Please change binary_model.parameters_boundaries[N].**

This happens because the boundaries (which will be used by the differential evolution routine) are hard boundaries. Therefore, this exception is thrown if the boundaries set in binary_model.parameters_boundaries do not include the true values. The exception states which parameter needs to be changed and in which direction.

The lines to modify are *one* of:

> **### to boundaries**
> **binary_model.parameters_boundaries[0] = [to-0.5,to+0.5]**
> **### uo boundaries**
> **binary_model.parameters_boundaries[1] = [uo*0.99 , uo*1.01]**
> **### tE boundaries**
> **binary_model.parameters_boundaries[2] = [tE-0.5,tE+0.5]**
> **### rho boundaries**
> **binary_model.parameters_boundaries[3] = [10**-4,10**-2]**

```
### log_10(s) boundaries
binary_model.parameters_boundaries[4] = [np.log10(s)-0.05,np.log10(s)+0.05]
### log_10(q) boundaries, no need to change this
#binary_model.parameters_boundaries[5] = [-5.60,-3.60]
### alpha boundaries
binary_model.parameters_boundaries[6] = [alpha-1.0,alpha+1.0]
```

Advice on changing binary_model.parameters_boundaries:

- If the problem is **to, uo**, or **tE**, review the output from Cell 5.
    a. Does your PSPL fit look okay? Most likely there are no glaring problems (e.g. the model does not go through the data).
    b. Is the peak of the lightcurve excluded from the PSPL fit (more likely to be a problem if your planetary perturbation is large and near the peak of the lightcurve)? The peak of the light curve contains most of the information about **uo** and **tE**. If it is excluded from the PSPL fit, this can lead to inaccurate values for **uo** and **tE**. If the peak is excluded, adjust **t_anomaly** and **delta_t** in Cell 3 to shift the range of the excluded data. (see examples and notes under Cell 5).
    c. If the PSPL fit seems okay, try adjusting binary_model.parameters_boundaries as described below.
- For everything except **uo**, start by changing the additive value by a factor of 2, e.g. (**to** - 0.5) → (**to** - 1.0)
- For **u0**, try changing 0.99 → 0.95 or 1.01 → 1.05
- Avoid making the width of the boundaries too large. If you need to change one of the boundaries by *more than twice* the amount above, also change the other boundary by the corresponding amount. The larger the width of the boundaries, the longer it will take the code to run. The initial widths of the boundaries were chosen so that the code will run in a reasonable amount of time. (e.g. increasing the width of the **log_s** boundary to +/- 0.5 could result in a run time of several hours or a fit that fails to converge).
- **u0**, **tE**, and **rho** are all positive quantities. Do not allow their lower boundaries to go below zero. (trying to fit negative **rho** is known to cause the fit to fail.)

ValueErrors that Do Not Fit the Template Above:

There are several special value errors for the boundaries of **log_s**:

1. **ValueError: log_s boundary too wide. Should not include log_s=0.**

Recall from Scott Gaudi's talk on Monday that the caustic structure has a significantly different form for log_s < 0, log_s ~ 0, and log_s > 0 (i.e. s < 1, s ~ 1, s > 1), which give the "close", "resonant," and "wide" caustic structures. The resonant caustic structures (log_s ~ 0) are extremely large and give extreme distortions to the light curve as a result. By design and by definition of being a "Gould & Loeb" -type planet, *your light curve does not have a resonant caustic*. If you include resonant caustic structures in your fit boundaries, the code will have a very hard time converging because those trials will be quite poor and are nowhere near the true minimum.

To fix this error:

1. Review Cell 4. Does your model generate a planetary perturbation in approximately the right part of the light curve? If not, go back to Cell 4, make adjustments, and re-run to this point.
2. Look at boundaries for log_s.
    a. What is the lower value?
    b. What is the upper value?
    c. Is your perturbation close (log_s < 0) or wide (log_s>0)?

   If your perturbation is close, both the upper and lower boundaries should have log_s < 0. If your perturbation is wide, both the upper and lower boundaries should have log_s > 0. Make adjustments as necessary.

2. **ValueError: Are you sure this is a close planet (s < 1)?**
3. **ValueError: Are you sure this is a wide planet (s > 1)?**

Both of these errors are related and indicate the same problem: on Tuesday, you incorrectly assessed whether the planet is a close or a wide planet (i.e. minor or major image perturbation).

1. Your planet should be similar to the other planets in your group. Consult with your other group members to get their opinion.
2. Go back to your Tuesday worksheet and rework the value of **s** and **alpha** assuming the planet perturbs the other image. Then, go back to Cell 4, fix the parameters and re-run up to this point.

Additional Notes:

- f(x) shown in the output is the chi2 of the fit. The final value of f(x) should be some number around ~700-900.
- The fitting process should take approximately 10-25 minutes.
- It is possible to speed up the fitting by fixing the values of **t0**, **u0**, and **tE**. Do this by setting their boundaries to [**t0**+0., **t0**-0.], etc. This will result in a faster, but less accurate fit. (Recommended only if your first fit failed and you are running out of time in the Hands-On session AND the PSPL fit is good.)

Known failure modes:

- If the fit is running and the f(x) value stops changing (e.g. decreases by <1 over 1000 steps), the fit is stuck. Interrupt it and restart.

Wrapping Up

For your presentations, you should save:

1. After the fit has run, SAVE your notebook.
2. The final fitted parameters and their uncertainties (especially the values for **log_s** and **log_q**). Either write this on a piece of paper or save it to a text file somewhere.
3. A plot of the light curve. In the bottom of the plot, there should be a “save” (floppy disk) icon. Or you could do a screenshot.
4. A zoom in of the planetary perturbation.

If you have extra time, start working on calculating the physical planet parameters for your group presentation.