# Planetary Sciences Assignment 5
# Image Formation in Gravitational Lensing

Kiran L
SC17B150

7th December 2020

## 1 Introduction

[1]According to Einstein's general theory of relativity, the path of the light from a distant star that passes by a massive object (lens) between the source and the observer is bent. The bending angle is typically very small, and the effect is known as microlensing. The light from a source located directly behind the lens is bent such that it appears to come from a circular region known as the Einstein ring . The Einstein ring is generally so small that it is unresolvable. However, the lens magnifies the light from the source by a substantial factor when it passes closer to the line of sight than the radius of the Einstein ring, $R_E$ , which is given by

$$R_E = \sqrt{\frac{2R_s D_{LS} D_L}{D_s}} \tag{1}$$

where,

- Rs is the schwarzschild radius
- $D_L$ is distance to the lens
- $D_S$ is distance to the source
- $D_{LS}$ is distance between source and lens

Microlensing is used to investigate the distribution of faint stellar and substellar mass bodies within our galaxy. The brightness of the source can increase several fold for a few months during a microlensing event, and the pattern of brightening can be used to determine (in a probabilistic manner) properties of the lens. If the lensing star has planetary companions, then these less massive bodies can produce characteristic blips on the observed light curve provided the line of sight passes within the planet's Einstein ring. Under favorable circumstances, planets as small as Earth can be detected. Microlensing events are only observed when the source and lens are very well aligned, so millions of potential source stars need to be monitored regularly to have a high probability of

detecting planets using this technique.

Microlensing provides information on the mass ratio and projected separation of the planet and star. This technique is capable of detecting systems with multiple planets and/or more than one star. The properties of individual microlensing planets (especially orbital eccentricities and inclinations) can often only be estimated in a statistical sense because of the many parameters that influence a microlensing light curve. However, additional information about the planets may be deduced under some special circumstances, such as very high magnification microlensing events and microlensing events that are viewed from two well-separated (of order 1 AU apart) locations. Follow-up observations of planets detected via microlensing (using other techniques because a given system's chance of producing a second observable microlensing event is exceedingly small) are very difficult because of the faintness of these distant systems. But when light from the lensing star can be distinguished from that of the source star, the mass of the planet's host star can be determined spectroscopically. Careful monitoring of many microlensing events is providing a very useful data set on the distribution of planets within our galaxy.

"Mass tells space how to curve.
space tells mass how to move"

In this assigment, we use the following given lens-source system properties to simulate the gravitational microlensing event and obtained plots of variation of amplitude (source solid angle in sky) and the trajectory of source, major and minor images with the lens position fixed:

1. Distance to Lens = 10 pc
2. Distance to Source = 500 pc
3. Mass of Lens = 100 $M_{\odot}$

# 2 Results

## 2.1 Case (a)

Source position starting at (-2.0,0.3) and ending at (2.0,0.3) in incremental steps of 0.1. Here the source's trajectory will be parallel to the X-axis.

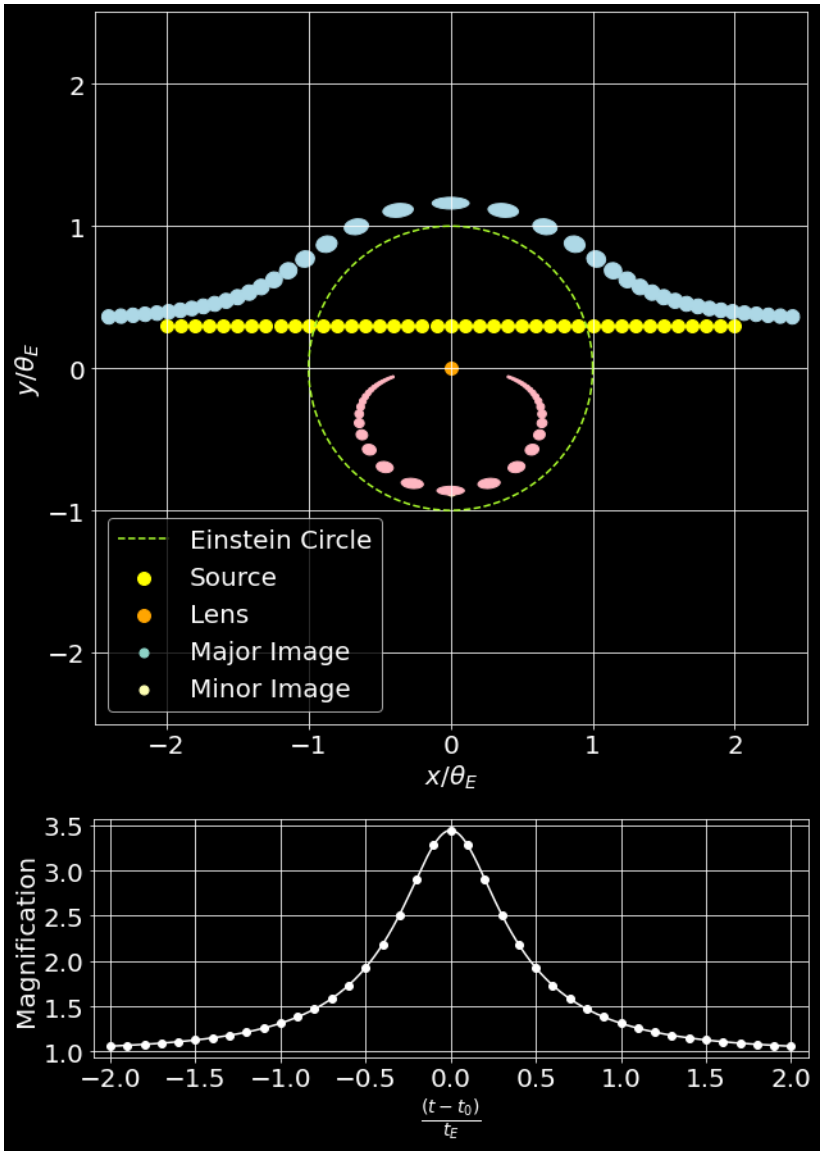


Figure 1: Case (a)

## 2.2 Case (b)

Source position starting at (-2.0,0.05) and ending at (2.0,0.05) in incremental steps of 0.1.

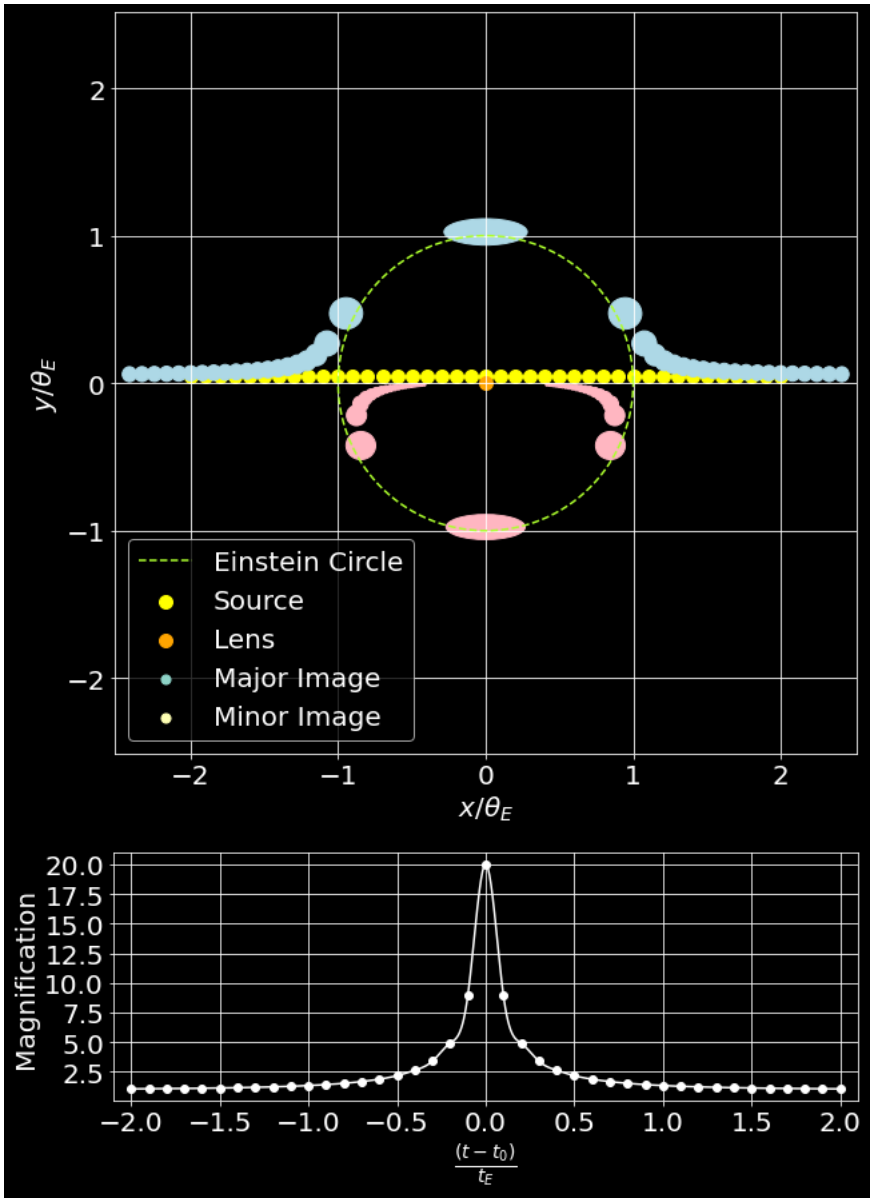


Figure 2: Case (b)

## 2.3 Case (c)

Source position starting at (-0.3,2.0) and ending at (-0.3,-2.0) in incremental steps of 0.1. Here the source's trajectory will be parallel to the Y-axis.

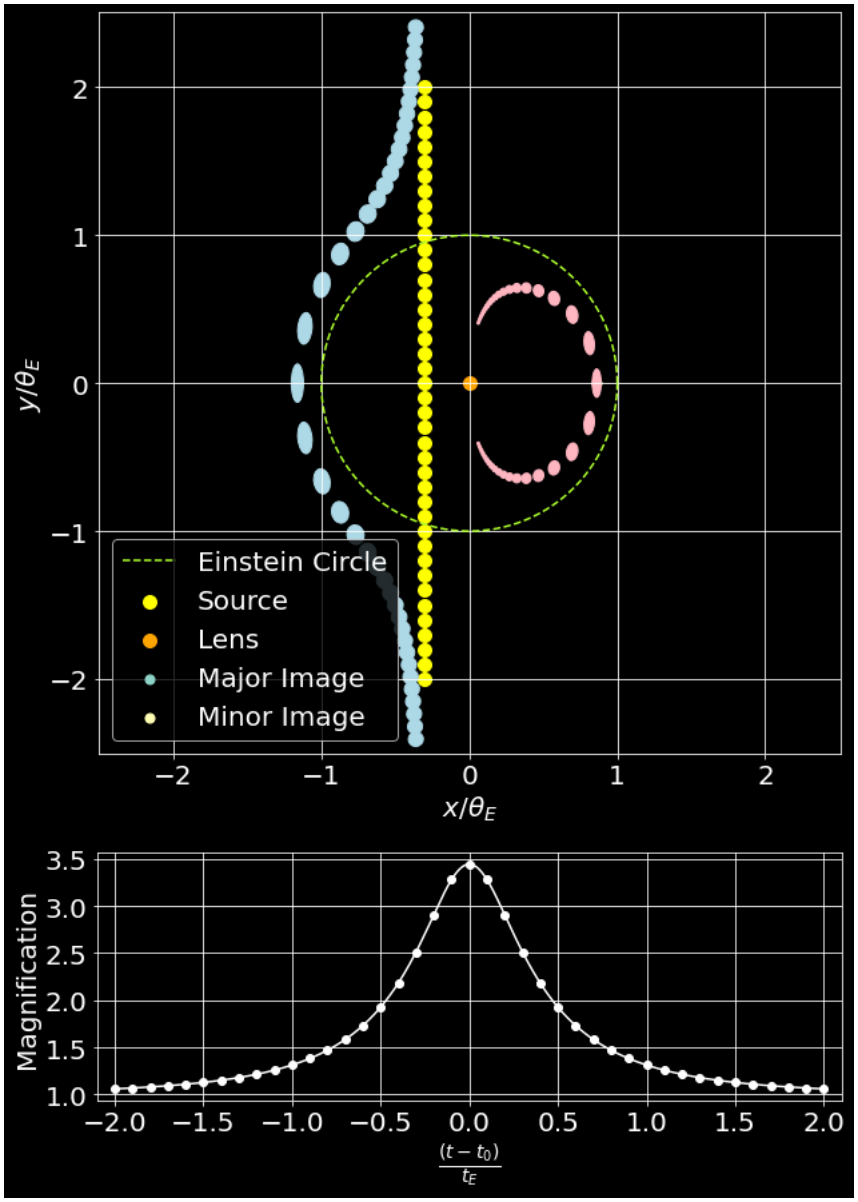


Figure 3: Case (c)

## 2.4 Case (d)

Source position starting at (-2.0,1.5) and ending at x=2.0. X coordinate should change in steps of 0.1 and Y coordinate should change in steps of -0.1.

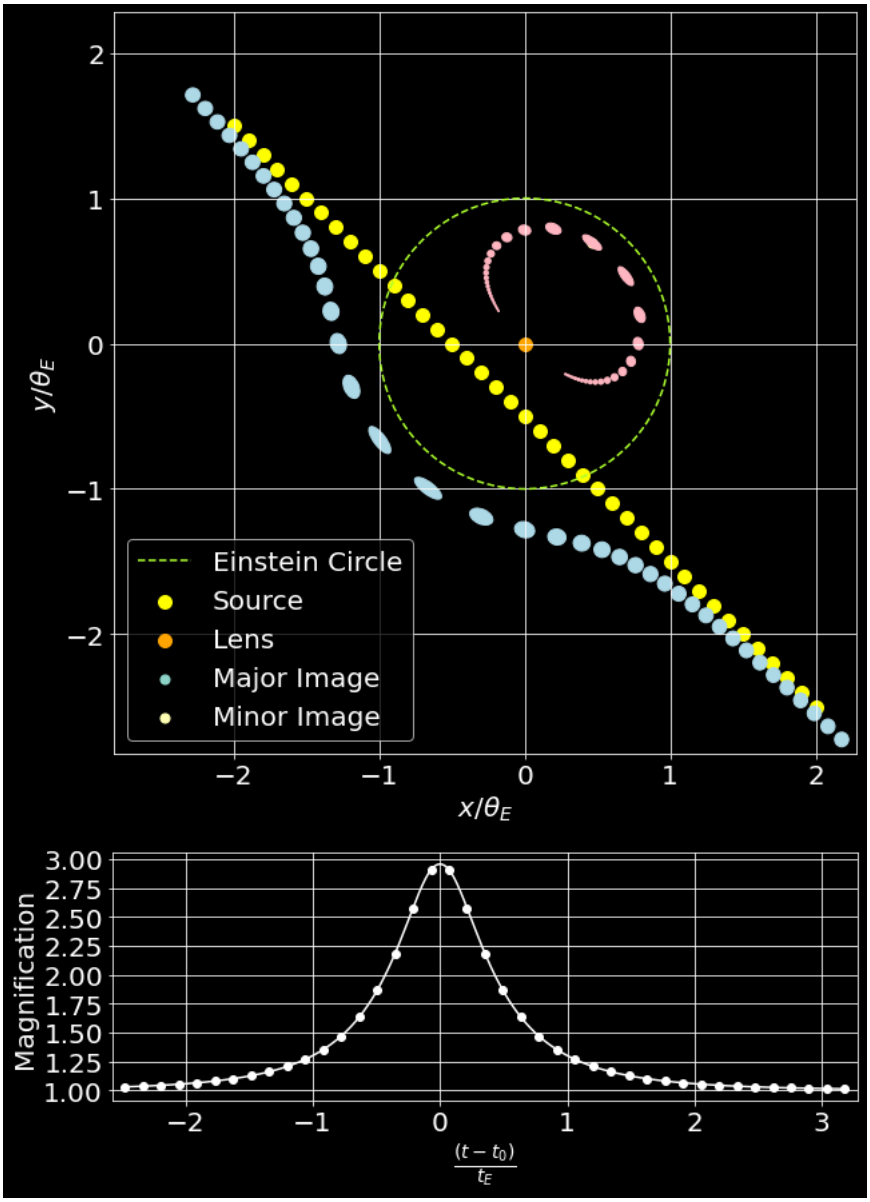


Figure 4: Case (d)

# 3 Observations and Inferences

| Sl. No. | **Observation** | **Inference** |
|---|---|---|
| 1. | We observe from Figure 1 and 2 that the closer the trajectory of the source to the lens, higher is the peak magnification. | Thus, we infer that (qualitatively) the total amplification is inversely proportional to the source lens separation. |
| 2. | The closer the trajectory of the source to the lens, lesser is the time for which the amplification is significant ($\gg 1$, say $A > 2$), as seen from bottom plots of amplification vs time since transiting the lens in Figure 1 and 2. | The FWHM of the amplitude variation curve with time since transit is inversely proportional to the minimum distance of the source to the lens. |
| 3. | We observe from plots in Figure 1 and 3 that the amplification vs time since transit plots are identically same for the relative proper motion of the source and lens along x- and y- axis. | Since most of the observations and inferences of the system are based on the light curves, which are insensitive to the direction of relative proper motion between the source and the lens, we infer that the unless the lens is bright enough to be observed and its properties studied through follow-up observations, the direction of motion bears no significance the in the observation of the microlensing event to study the properties of the lensing exoplanet system. |
| 4. | The direction of relative proper motion between the source and the lens does not bear any significance unless the source surface as well as the major and minor images are well resolved (which, in most cases is not true, due to all sources other than the Sun being point sources, effectively). | In case of them being resolved with the lens not very bright (relatively) from the image of the system at the time of minimum separation between the source and lens, we can infer the direction of relative proper motion between source and lens by observing the direction of elongation of the major and minor images. The direction of elongation indicates the direction of relative proper motion. |
| 5. | We observe from Figure 4 that the sampling of the position of the source relative to the lens is such that we miss the peak of the total magnification curve. | The sampling of the source periodically with sufficient frequency is necessary to capture the peak of the light curve of microlensing event. If ill sampled (as shown in Case (d), in Figure 4), then to obtain the peak amplitude requires us to use interpolation methods, which in-turn induces some degree of uncertainty in the peak magnification. |

| | | |
|---|---|---|
| 6. | When lens is not visible, the minimum separation between the source and the lens can be estimated based on the separation of the major and minor images from the source. This information along with the information of the direction of relative proper motion between the source and the lens, if available (with the images being well resolved to determine the direction of elongation, which is parallel to the direction of relative proper motion) can be used to observe the lens at other wavebands in which it is brighter than its surrounding background. | As an example, Brown dwarfs which are effectively invisible in visible waveband, can be observed in near IR ($\approx 1.55\mu m$), if the situation favours as described above. Thus, such a follow-up observations in other wavebands can help us infer more about the exo-planet system. Though all this seems possible theoretically, practical observation includes various hindrances such as - atmospheric absorption, telescope-emission in IR, background limited observations, and various alike. |

# 4 Conclusion

In this exercise, we have explored the phenomenon of gravitational microlensing by simulating the trajectory of the source and images with stationary lens. Based on the simulations of the trajectory of the source during microlensing event, the total magnification plots as well as the approximate shape of the major and minor images, various observations are made and inferences drawn. We infer that closer the trajectory to the lens higher is the peak of light curve. Also, for sources whose surface and images are un-resolved (non-extended or point-like) during microlensing event, the direction of proper motion is not discernible and all inferences about the lens is based on the light-curve along. Using the same, it is possible to infer the presence of exoplanet by detecting secondary blips in the light (amplification) curves. The width of the light curve gives an estimate of lens mass. This along with the duration of transit gives an estimate of rate of relative proper motion between source and lens. Closer the trajectory of source to the lens, higher should be the rate of sampling of the source brightness, in order to capture the peak amplification, since the FWHM of the light curve reduces drastically with decrease in minimum separation between the source and lens. The rate of sampling can be decided based on the time it takes for the brightness to double, faster it double higher should be the sampling rate. Owing to the finite observations time per source, high sampling can be achieved by taking observations from multiple observatories. Also, taking multi-waveband observations helps in detecting the lens in (at least) one of the wavebands, which is useful for follow-up observations of the lens (exoplanet system) and infer its properties. Though this is possible in theory, practical implementation requires one to overcome atmospheric absorption, air-glow, telescope emission, and various alike issues.

# 5 Program

```
############################
### Importing Libraries ###
############################

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse
from astropy import units as u
from astropy import constants as const
import pylab as py
from matplotlib import animation, rc
#from IPython.display import HTML
from matplotlib.animation import FuncAnimation
from scipy.interpolate import interp1d
from matplotlib import gridspec
import sys

############################
###### User Inputs ########
############################

x0,x1,y0,y1 = float(sys.argv[1]),float(sys.argv[2]),float(sys.argv[3]),float(
    sys.argv[4])
step = float(sys.argv[5])
savename= sys.argv[6]
scale = float(sys.argv[7]) # 0.0075 or 0.0025 for given set of paths

############################
#######  Constants  #######
############################

DL = 10*u.pc
DS = 500*u.pc
DLS = DS - DL
ML = 100*u.M_sun
Rs = (2*const.G*ML)/const.c**2
theta_E = np.sqrt((2*Rs*DLS)/(DL*DS))

############################
#######  Variables  #######
############################

x = []
y = []
if(x0<x1):
  x = np.arange(x0,x1+step,step)
elif(x0>x1):
  x = np.arange(x1,x0+step,step)[::-1]
if(y0<y1):
  y = np.arange(y0,y1,step)
elif(y0>y1):
  y = np.arange(y1,y0+step,step)[::-1]
if(len(x) == 0 and len(y) != 0):
  if(x0 == x1):
    x = x0*np.ones_like(y)
elif(len(y) == 0 and len(x) != 0):
  if(y0 == y1):
    y = y0*np.ones_like(x)

a = x[np.where(np.round(y,8)==0)]
b = y[np.where(np.round(x,8)==0)]

if(len(a)==0 and len(b)==0):
  exit()
elif(len(a)==0):
```

```
    a = 0
    b = b[0]
    u0 = np.abs(b)
  elif(len(b)==0):
    b = 0
    a = a[0]
    u0 = np.abs(a)
  else:
    a = a[0]
    b = b[0]
    u0 = np.abs(a*b)/np.sqrt(a**2 + b**2)

  u_t = np.sqrt(x**2 + y**2) # u(t) wiht motion arbitrary direction
  A_plus = (u_t**2+2)/(2*u_t*np.sqrt(u_t**2+4))+0.5
  A_minus = (u_t**2+2)/(2*u_t*np.sqrt(u_t**2+4))-0.5
  A = A_plus+A_minus
  theta_S = u_t*theta_E
  theta_I_plus = 0.5*(theta_S + np.sqrt(theta_S**2+4*theta_E**2))
  theta_I_minus = 0.5*(theta_S - np.sqrt(theta_S**2+4*theta_E**2))
  v_plus = theta_I_plus/theta_E
  v_minus = theta_I_minus/theta_E
  theta = np.arange(0,np.pi*2,0.01)
  x_circ = np.cos(theta)
  y_circ = np.sin(theta)
  theta_S = u_t*theta_E
  v_plus = theta_I_plus/theta_E
  v_minus = theta_I_minus/theta_E
  x_plus, y_plus = v_plus.value*(x/u_t),v_plus.value*(y/u_t)
  x_minus, y_minus = v_minus.value*(x/u_t),v_minus.value*(y/u_t)
  t_arr = np.ones_like(A_plus+A_minus)
  t_arr[:np.argmin(np.abs(u_t-u0))] *= -1
  t_arr = t_arr*np.sqrt(u_t**2-u0**2)
  eccs = np.abs(u0/u_t)

  ############################
  #######  Plotting  ########
  ############################

  eps = 0.1
  ft = 20
  plt.style.use('dark_background')
  fig = plt.figure(figsize=(10, 15))
  gs = gridspec.GridSpec(2, 1, height_ratios=[3, 1])
  ax0 = plt.subplot(gs[0])
  plt.scatter(x,y,s=100,label="Source",c='yellow')
  ax0.plot(x_circ,y_circ,'--',c='greenyellow', label="Einstein Circle")
  plt.scatter(0,0,s=100,label="Lens",c='orange')
  idx = np.argmin(u_t)
  plt.scatter(x_plus[idx],y_plus[idx],s=50,label="Major Image")
  plt.scatter(x_minus[idx],y_minus[idx],s=50,label="Minor Image")
  for i in range(len(eccs)):
      rot = np.arctan(b/a)
      rot -= (b*x[i]-a*y[i])/u_t[i]
      A = scale*A_plus[i]
      b_plus = np.sqrt(A/np.pi  * np.sqrt(1+eps-eccs[i]**2))
      a_plus = A / np.pi / b_plus
      A = scale*A_minus[i]
      b_minus = np.sqrt(A/np.pi  * np.sqrt(1+eps-eccs[i]**2))
      a_minus = A / np.pi / b_minus
      ellipse_plus = Ellipse(xy=(x_plus[i],y_plus[i]), width=2*b_plus, height=2
      *a_plus, angle=(180/np.pi)*rot,color="lightblue",label="Major Image")
      ellipse_minus = Ellipse(xy=(x_minus[i],y_minus[i]), width=2*b_minus,
       height=2*a_minus, angle=(180/np.pi)*rot,color="lightpink",label="Minor
       Image")
      ax0.add_artist(ellipse_plus)
      ax0.add_artist(ellipse_minus)

  plt.legend(fontsize=ft,loc="lower left")
```

```
ax0.set_xlabel(r'$x/\theta_E$',fontsize=ft)
ax0.set_ylabel(r'$y/\theta_E$',fontsize=ft)
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
plt.grid()
lims = [0,0]
lims[0] = -step+np.amin([np.amin(y_plus),np.amin(y_minus),np.amin(x_plus),np.amin(x_minus)])
lims[1] = step+np.amax([np.amax(y_plus),np.amax(y_minus),np.amax(x_plus),np.amax(x_minus)])
ax0.set_ylim(lims[0],lims[1])
ax0.set_xlim(lims[0],lims[1])
ax0.set_aspect('equal')

ax1 = plt.subplot(gs[1])
spline_f = interp1d(x=t_arr,y= A_plus+A_minus, kind='cubic')
t_linspace = np.linspace(t_arr[0],t_arr[-1],1000)
y_linspace = spline_f(t_linspace)
plt.scatter(t_arr,A_plus+A_minus,c='white')
plt.plot(t_linspace,y_linspace,c='white')
plt.ylabel("Magnification",fontsize=ft)
plt.xlabel(r'$\frac{(t-t_0)}{t_E}$',fontsize=ft)
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
ax1.set_xlim(np.amin(t_arr)-0.1, np.amax(t_arr)+0.1)
plt.grid()
plt.savefig("{}1_black.png".format(savename),bbox_inches="tight")

############################
#######  Animating  #######
############################

def init():
    line0.set_data([], [])
    line1.set_data([], [])
    line2.set_data([], [])
    line3.set_data([], [])
    line4.set_data([], [])
    return line1,line2,line3,

def update(i):
    plt.cla()
    gs = gridspec.GridSpec(2, 1, height_ratios=[3, 1])
    ax0 = plt.subplot(gs[0])
    ax0.axis('square')
    ax0.set_ylim(lims[0],lims[1])
    ax0.set_xlim(lims[0],lims[1])
    ax0.plot(0,0,'o',markersize = 9, markerfacecolor = "orange",
     markeredgecolor ="orange", label="Lens")
    ax0.plot(x_circ,y_circ,'--',c='greenyellow', label="Einstein Circle")
    ax0.plot(x[i], y[i], 'o',color = 'yellow', markersize=10, markevery
     =10000, markerfacecolor = 'yellow',lw=2) # source
    plt.ylabel(r'$y/\theta_E$',fontsize=ft)
    plt.xlabel(r'$x/\theta_E$',fontsize=ft)
    plt.xticks(fontsize=ft)
    plt.yticks(fontsize=ft)
    plt.grid()

    rot = np.arctan(b/a)
    rot -= (b*x[i]-a*y[i])/u_t[i]
    A = scale*A_plus[i]
    b_plus = np.sqrt(A/np.pi  * np.sqrt(1+eps-eccs[i]**2))
    a_plus = A / np.pi / b_plus
    A = scale*A_minus[i]
    b_minus = np.sqrt(A/np.pi  * np.sqrt(1+eps-eccs[i]**2))
    a_minus = A / np.pi / b_minus
    ellipse_plus = Ellipse(xy=(x_plus[i],y_plus[i]), width=2*b_plus, height=2
     *a_plus, angle=(180/np.pi)*rot,color="lightblue",label="Major Image")
```

```
        ellipse_minus = Ellipse(xy=(x_minus[i],y_minus[i]), width=2*b_minus,
        height=2*a_minus, angle=(180/np.pi)*rot,color="lightpink",label="Minor
        Image")
        ax0.add_artist(ellipse_plus)
        ax0.add_artist(ellipse_minus)

        line0.set_data(0,0)
        line1.set_data(x[i],y[i])
        line2.set_data(x_plus[i],y_plus[i])
        line3.set_data(x_minus[i],y_minus[i])
        line4.set_data(x_circ, y_circ)

        lgnd = ax0.legend((line4,line0,line1, line2, line3), ('Einstein Circle','
        Lens','Source', 'Major Image', 'Minor Image'), loc="best", fontsize=ft-
        2.5)
        lgnd.legendHandles[0]._legmarker.set_markersize(6)
        lgnd.legendHandles[1]._legmarker.set_markersize(6)
        lgnd.legendHandles[2]._legmarker.set_markersize(6)
        lgnd.legendHandles[3]._legmarker.set_markersize(6)

        ax1 = plt.subplot(gs[1])
        idx = np.where(t_linspace<=t_arr[i])
        plt.scatter(t_arr[:i+1],(A_plus+A_minus)[:i+1],c='white')
        plt.plot(t_linspace[idx],y_linspace[idx],c='white')
        plt.ylabel("Magnification",fontsize=ft)
        plt.xlabel(r'$\frac{(t-t_0)}{t_E}$',fontsize=ft)
        plt.xticks(fontsize=ft)
        plt.yticks(fontsize=ft)
        ax1.set_xlim(np.amin(t_arr)-0.1, np.amax(t_arr)+0.1)
        ax1.set_ylim(np.amin(A_plus+A_minus)-0.1, np.amax(A_plus+A_minus)+0.1)
        plt.grid()
        return (line0,line1,line2,line3,line4)

    fig = plt.figure(figsize=(10, 15))
    gs = gridspec.GridSpec(2, 1, height_ratios=[3, 1])
    ax0 = plt.subplot(gs[0])

    plt.rcParams["figure.figsize"] = (10,10)
    line0, = ax0.plot([], [], 'o',color = 'orange', markersize=10, markevery
        =10000, markerfacecolor = 'orange',lw=2) # lens
    line1, = ax0.plot([], [], 'o',color = 'yellow', markersize=10, markevery
        =10000, markerfacecolor = 'yellow',lw=2) # source
    line2, = ax0.plot([], [], 'o',color = 'lightblue', markersize=1, markevery
        =10000, markerfacecolor = 'lightblue',lw=2) # Major Image
    line2.set_animated(True)
    line3, = ax0.plot([], [], 'o',color = 'lightpink', markersize=1, markevery
        =10000, markerfacecolor = 'lightpink',lw=2) # minor image
    line3.set_animated(True)
    line4, = ax0.plot([], [], '--',color = 'greenyellow', markersize=10,
        markevery=10000, markerfacecolor = 'greenyellow',lw=2) # radius

    anim = FuncAnimation(fig, update, init_func=init, frames=np.arange(len(x)),
        interval=100, blit=True, repeat=True)
    #HTML(anim.to_html5_video())
    anim.save('{}1.gif'.format(savename), writer='pillow', fps=10)

    ############################
    ####### End of Code #######
    ############################
```

# References


[1] Jack J. Lissauer - Fundamental Planetary Science, Updated Edition, Physics, Chemistry and Habitability-Cambridge University Press (2019)