PHYS454 HW2

by Edward Sanchez

References: https://ssd.jpl.nasa.gov/horizons/manual.html
https://ssd.jpl.nasa.gov/horizons/manual.html#observer-table
https://ssd.jpl.nasa.gov/horizons/manual.html#obsquan
https://ssd.jpl.nasa.gov/horizons/tutorial.html
https://astroquery.readthedocs.io/en/latest/jplhorizons/jplhorizons.html

All files available in GitHub https://githu

Prove Kepler's second law numerically

In [1]:

```
# 1a. Use JPL Horizons Web app (https://ssd.jpl.nasa.gov/horizons/app.html#/) to retri
# about an object orbiting the sun. Try to show Kepler's 2nd law by numerically estima
# by the orbit during a short time duration and comparing that area to an equal time a
# location along the orbit.
from astropy.coordinates import SkyCoord
from astroquery.jplhorizons import Horizons
from astropy.time import Time

# Mercury has an orbital period of 88 days so I use a 90-day period
mercury_start_time = '2024-01-01T00:00:00.00'
mercury_end_time = '2024-03-29T23:59:59.00'

# Generate ephemeris for Mercury
# https://ssd.jpl.nasa.gov/horizons/tutorial.html
mercury = Horizons('199', location='@sun', epochs={'start':mercury_start_time,'stop':m
table_mercury = mercury.vectors(refplane='ecliptic')
coord_mercury = SkyCoord(table_mercury['x'].quantity, table_mercury['y'].quantity, tab
                         representation_type='cartesian', frame='icrs',obstime=mercury

print(coord_mercury)
```

```
<SkyCoord (ICRS): (x, y, z) in AU
    [(-0.27463037,  0.20036221,  4.15640302e-02),
     (-0.29593486,  0.17809692,  4.16986213e-02),
     (-0.31516174,  0.15457935,  4.15403085e-02),
     (-0.33227883,  0.13002522,  4.11037681e-02),
     (-0.3472754 ,  0.1046397 ,  4.04047825e-02),
     (-0.36015893,  0.07861603,  3.94598307e-02),
     (-0.37095213,  0.05213478,  3.82857501e-02),
     (-0.37969021,  0.02536349,  3.68994632e-02),
     (-0.38641843, -0.00154315,  3.53177634e-02),
     (-0.39119005, -0.02844298,  3.35571523e-02),
     (-0.39406447, -0.05520578,  3.16337205e-02),
     (-0.39510575, -0.08171261,  2.95630637e-02),
     (-0.39438133, -0.10785505,  2.73602286e-02),
     (-0.39196101, -0.13353441,  2.50396816e-02),
     (-0.3879161 , -0.15866096,  2.26152963e-02),
     (-0.38231873, -0.18315316,  2.01003539e-02),
     (-0.37524137, -0.20693692,  1.75075548e-02),
     (-0.36675641, -0.22994496,  1.48490376e-02),
     (-0.3569358 , -0.25211609,  1.21364039e-02),
     (-0.34585094, -0.27339468,  9.38074554e-03),
     (-0.33357242, -0.29373013,  6.59267598e-03),
     (-0.32017   , -0.31307633,  3.78236124e-03),
     (-0.30571254, -0.33139123,  9.59552223e-04),
     (-0.29026798, -0.34863648, -1.86638334e-03),
     (-0.27390339, -0.36477702, -4.68642954e-03),
     (-0.25668499, -0.37978079, -7.49189169e-03),
     (-0.23867826, -0.39361845, -1.02743669e-02),
     (-0.21994797, -0.40626311, -1.30257158e-02),
     (-0.20055834, -0.41769014, -1.57380356e-02),
     (-0.18057311, -0.42787697, -1.84036342e-02),
     (-0.16005568, -0.4368029 , -2.10150062e-02),
     (-0.13906925, -0.44444903, -2.35648097e-02),
     (-0.11767694, -0.4507981 , -2.60458449e-02),
     (-0.09594194, -0.45583443, -2.84510337e-02),
     (-0.07392766, -0.45954385, -3.07734008e-02),
     (-0.05169791, -0.46191364, -3.30060564e-02),
     (-0.02931702, -0.46293255, -3.51421799e-02),
     (-0.00685003, -0.4625908 , -3.71750054e-02),
     ( 0.01563714, -0.46088009, -3.90978086e-02),
     ( 0.03807752, -0.45779367, -4.09038952e-02),
     ( 0.06040292, -0.45332641, -4.25865914e-02),
     ( 0.08254372, -0.44747494, -4.41392363e-02),
     ( 0.10442872, -0.44023776, -4.55551766e-02),
     ( 0.12598494, -0.43161544, -4.68277637e-02),
     ( 0.14713742, -0.42161079, -4.79503540e-02),
     ( 0.16780908, -0.41022916, -4.89163124e-02),
     ( 0.18792048, -0.3974787 , -4.97190206e-02),
     ( 0.20738971, -0.38337074, -5.03518888e-02),
     ( 0.2261322 , -0.36792012, -5.08083746e-02),
     ( 0.24406058, -0.35114572, -5.10820075e-02),
     ( 0.2610846 , -0.33307094, -5.11664214e-02),
     ( 0.27711104, -0.31372433, -5.10553967e-02),
     ( 0.29204366, -0.29314021, -5.07429129e-02),
     ( 0.30578327, -0.27135946, -5.02232141e-02),
     ( 0.31822788, -0.24843038, -4.94908887e-02),
     ( 0.32927285, -0.22440958, -4.85409670e-02),
     ( 0.33881131, -0.19936301, -4.73690377e-02),
     ( 0.34673468, -0.17336712, -4.59713862e-02),
     ( 0.35293337, -0.14650996, -4.43451582e-02),
```

```
( 0.35729778, -0.11889249, -4.24885493e-02),
( 0.35971951, -0.09062979, -4.04010236e-02),
( 0.36009302, -0.06185233, -3.80835608e-02),
( 0.35831755, -0.03270711, -3.55389326e-02),
( 0.35429956, -0.00335863, -3.27720037e-02),
( 0.3479556 ,  0.02601035, -2.97900518e-02),
( 0.33921558,  0.0551986 , -2.66030968e-02),
( 0.32802661,  0.0839866 , -2.32242234e-02),
( 0.31435709,  0.11213767, -1.96698764e-02),
( 0.29820105,  0.13939982, -1.59601017e-02),
( 0.27958262,  0.16550886, -1.21187007e-02),
( 0.2585601 ,  0.19019263, -8.17326326e-03),
( 0.23522953,  0.2131765 , -4.15504161e-03),
( 0.20972722,  0.23419008, -9.86352091e-05),
( 0.18223086,  0.25297488,  3.95853485e-03),
( 0.15295878,  0.26929258,  7.97696511e-03),
( 0.12216726,  0.28293337,  1.19160039e-02),
( 0.09014563,  0.29372377,  1.57349363e-02),
( 0.05720938,  0.30153321,  1.93941535e-02),
( 0.02369172,  0.30627885,  2.28563214e-02),
(-0.01006591,  0.30792811,  2.60874623e-02),
(-0.0437237 ,  0.30649883,  2.90578637e-02),
(-0.07695284,  0.30205696,  3.17427557e-02),
(-0.10944382,  0.29471223,  3.41227176e-02),
(-0.14091325,  0.28461219,  3.61838116e-02),
(-0.17110871,  0.27193521,  3.79174636e-02),
(-0.19981179,  0.25688313,  3.93201344e-02),
(-0.22683933,  0.23967393,  4.03928359e-02),
(-0.252043  ,  0.22053502,  4.11405491e-02),
(-0.27530786,  0.19969726,  4.15715958e-02)]>
```

Observer & Vector Tables

If an observer or vector table has been requested, the @ symbol may be dropped; the Earth will be assumed if only an integer like 675 or a name fragment like Palom is input.

However, if you are trying to specify an observing site not on Earth, you MUST use the @ symbol for correct interpretation. For example, if an observer table as seen from the Sun is desired, it must be specified as @10 or @sun. Specifying 10 only will select the Caussols site.

For location 0 and ssb refer to solar system barycenter (SSB).

For id 1 = Mercury Barycenter and 199 = Mercury.

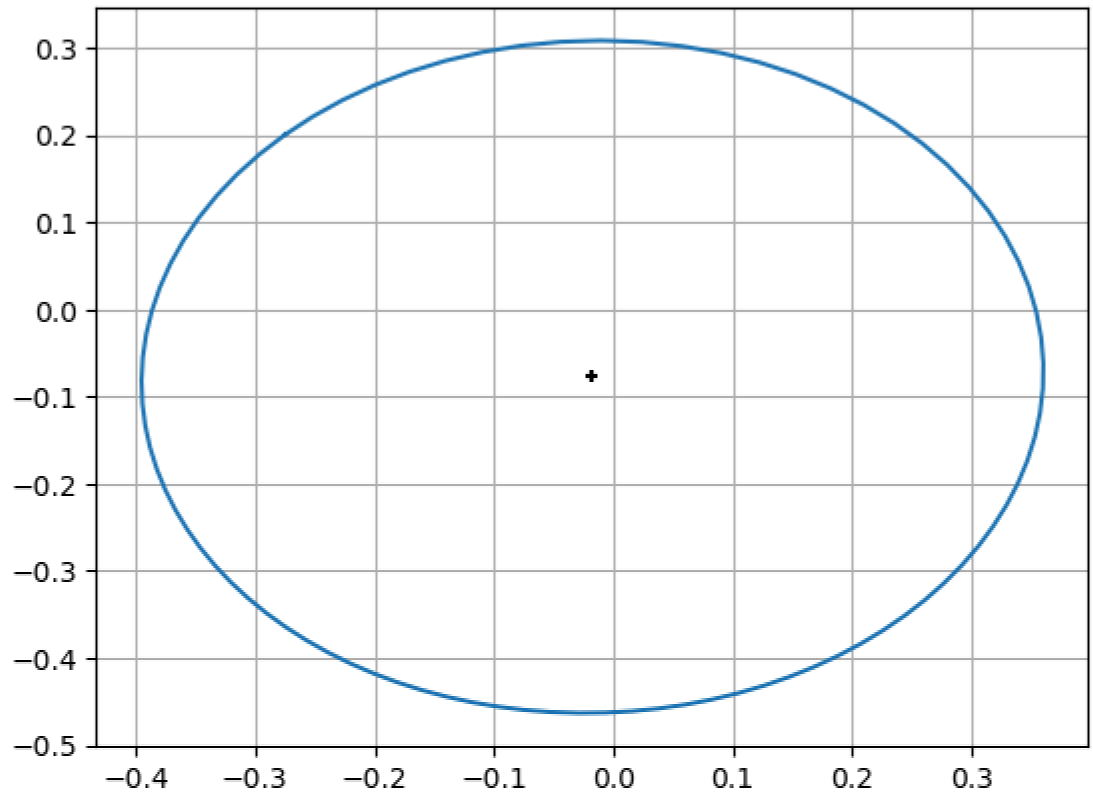
For Mercury and Venus, there is no difference between planet-center and system barycenter (1=199, 2=299) as far as Horizons selection is concerned because there is only the planet: no satellites, so no offset between planet center and planetary system center-of-mass.

If you request orbital elements of the Earth (399) with respect to Sun (10), the resulting elements will contain short-period oscillations due to the Earth’s motion with respect to the Earth-Moon barycenter, as well as the Sun’s motion with respect to the solar system barycenter. Unless these short period motions are desired, you might want to instead request 3 with respect to 0 (Earth-Moon barycenter with respect to solar system barycenter).

https://ssd.jpl.nasa.gov/horizons/manual.html

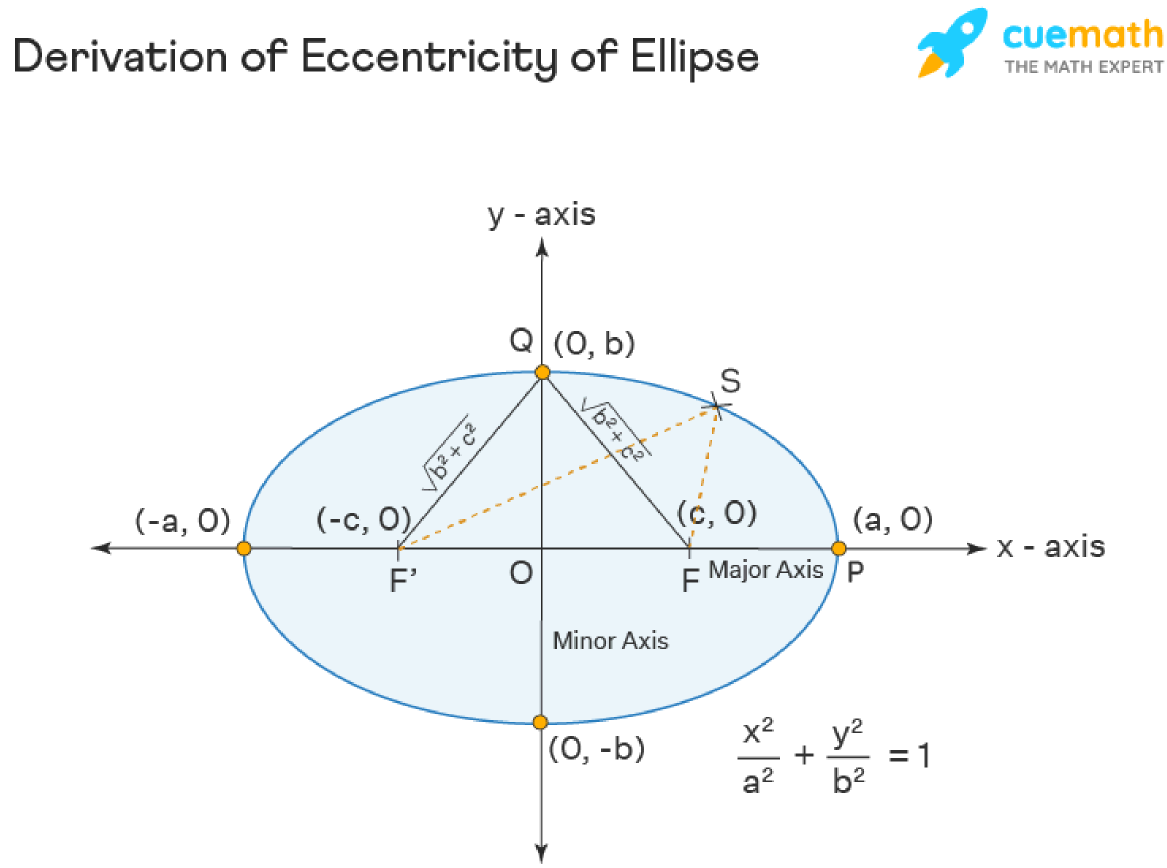
In [2]:
```
import matplotlib.pyplot as plt

# Solar system barycenter at x = 0, y = 0
ssb_xcenter = (max(table_mercury['x']) + min(table_mercury['x'])) / 2
ssb_ycenter = (max(table_mercury['y']) + min(table_mercury['y'])) / 2
print (ssb_xcenter, ssb_ycenter)

fig1 = plt.figure(1)

def plot1():
    plt.plot(table_mercury['x'], table_mercury['y'])
    plt.scatter(ssb_xcenter, ssb_ycenter, s=25, marker='+', color='k')
    plt.grid()

plot1()
plt.show()
```

```
-0.01750636558649063 -0.07750222055156711
```



In [3]:
```
# https://ssd.jpl.nasa.gov/planets/approx_pos.html
# Keplerian elements and rates
# Table 1
# Mercury semi-major axis (AU) and eccentricity
mercury_sma = 0.38709927
mercury_ecc = 0.20563593

# Calculate semi-minor axis in AU
b = mercury_sma * (1 - mercury_ecc**2)
print(f'The eccentricity of Mercury is e = {mercury_ecc:.5f}.')
print(f'The semi-major axis is {mercury_sma:.5f} AU and the semi-minor axis is {b:.5f}
```

```
The eccentricity of Mercury is e = 0.20564.
The semi-major axis is 0.38710 AU and the semi-minor axis is 0.37073 AU.
```



In [4]:

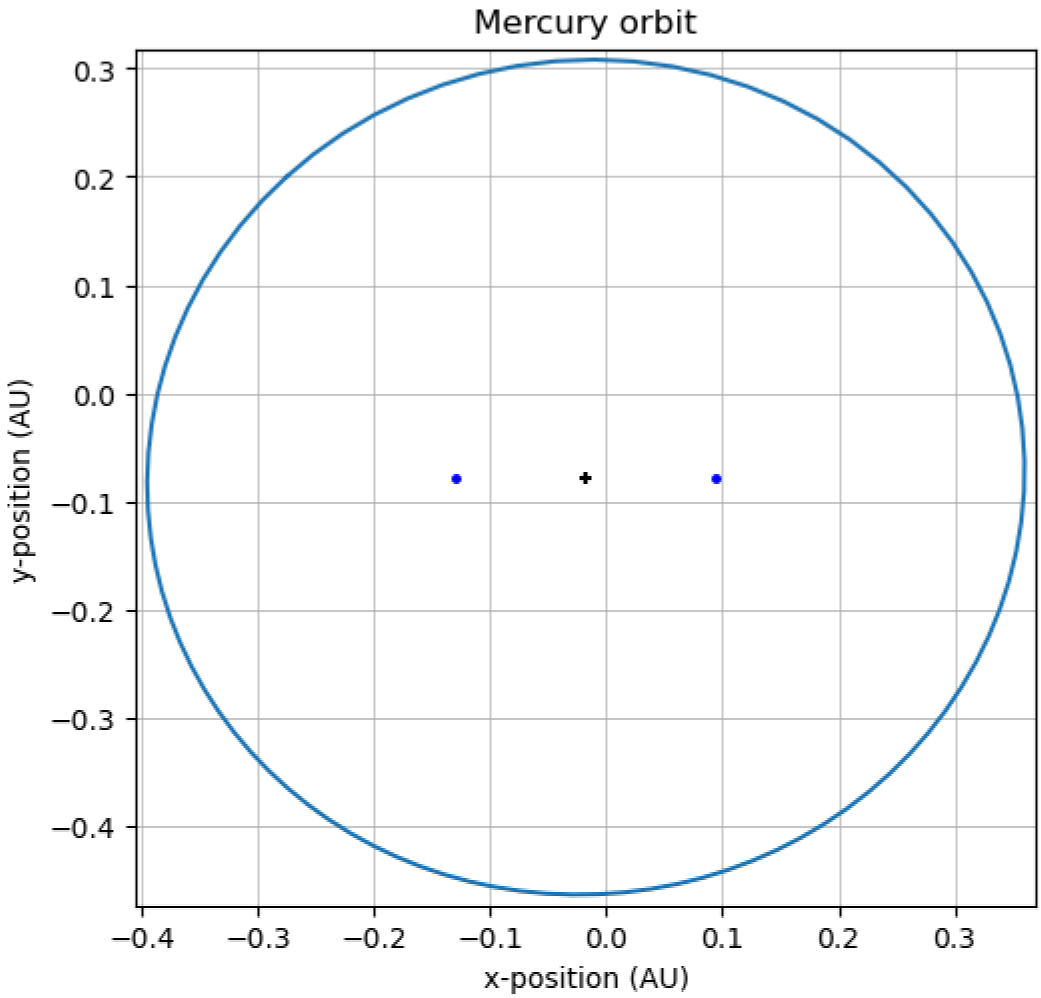
```python
import numpy as np

# Finding the ellipse foci
c = np.sqrt(mercury_sma**2 - b**2)
left_focus = ssb_xcenter - c
right_focus = ssb_xcenter + c
print(f'The two focii lie on ({left_focus:.2f}, {ssb_ycenter:.2f}) and ({right_focus:.
```

```
The two focii lie on (-0.13, -0.08) and (0.09, -0.08).
```

In [5]:

```python
# Foci location
foci_locations = np.array([[left_focus, ssb_ycenter],
                           [right_focus, ssb_ycenter]])

# Plot Mercury orbit with new information and more accuracy
fig2 = plt.figure(2, figsize=(round(15*mercury_sma,2), round(15*b,2)))

def plot2():
    plot1()
    plt.scatter(foci_locations[:, 0], foci_locations[:, 1], s=25, marker='.', color='b

    plt.title('Mercury orbit')
    plt.xlabel('x-position (AU)')
    plt.ylabel('y-position (AU)')
    plt.xlim(min(table_mercury['x'] - 0.01), max(table_mercury['x'] + 0.01))
    plt.ylim(min(table_mercury['y'] - 0.01), max(table_mercury['y'] + 0.01))
    plt.grid(linewidth=0.5)
```

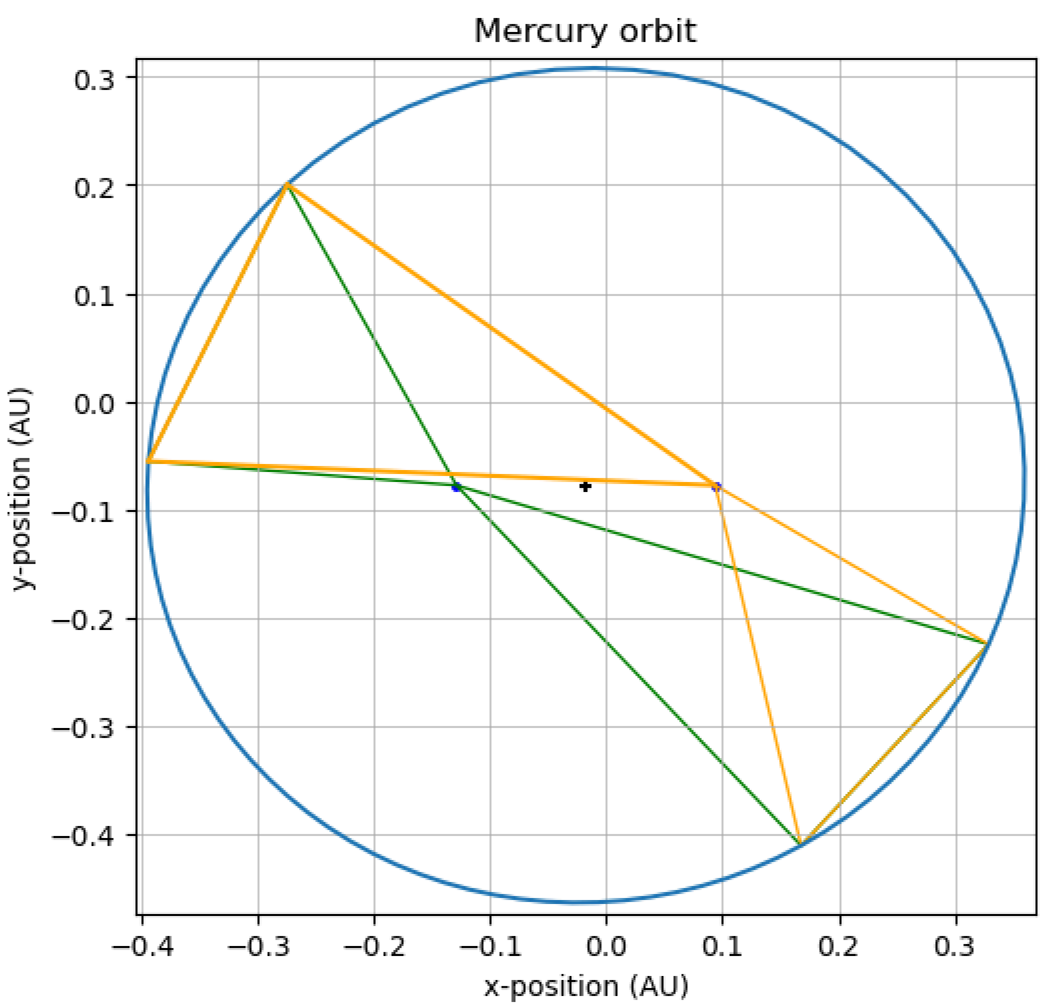
```
plot2()
plt.show()
```



In [13]:

```
# Visualization of triangles used for area calculations
import matplotlib.patches as patches

start_point = 0

# Defining the first triangle
vertices1 = [(left_focus, ssb_ycenter), (table_mercury['x'][start_point], table_mercur
             (table_mercury['x'][start_point+10], table_mercury['y'][start_point+10])]
triangle1 = patches.Polygon(vertices1, closed=True, edgecolor='g', facecolor='none')

triangle_base = np.array([[right_focus, ssb_ycenter],
                          [table_mercury['x'][start_point], table_mercury['y'][start_pc
triangle_height = np.array([[table_mercury['x'][start_point], table_mercury['y'][start
                             [table_mercury['x'][start_point+10], table_mercury['y'][st
triangle_hypotenuse = np.array([[table_mercury['x'][start_point+10], table_mercury['y'
                                 [right_focus, ssb_ycenter]])

# Defining the second triangle
vertices2 = [(left_focus, ssb_ycenter), (table_mercury['x'][start_point+45], table_mer
             (table_mercury['x'][start_point+55], table_mercury['y'][start_point+55])]
triangle2 = patches.Polygon(vertices2, closed=True, edgecolor='g', facecolor='none')

vertices2b = [(right_focus, ssb_ycenter), (table_mercury['x'][start_point+45], table_m
             (table_mercury['x'][start_point+55], table_mercury['y'][start_point+55])]
```

```
triangle2b = patches.Polygon(vertices2b, closed=True, edgecolor='orange', facecolor='r

fig3 = plt.figure(3, figsize=(round(15*mercury_sma,2), round(15*b,2)))

def plot3():
    plot2()
    plt.plot(triangle_base[:, 0], triangle_base[:, 1], color='orange')
    plt.plot(triangle_height[:, 0], triangle_height[:, 1], color='orange')
    plt.plot(triangle_hypotenuse[:, 0], triangle_hypotenuse[:, 1], color='orange')
    ax = plt.gca()
    ax.add_patch(triangle1)
    ax.add_patch(triangle2)
    ax.add_patch(triangle2b)

plot3()
plt.show()
```

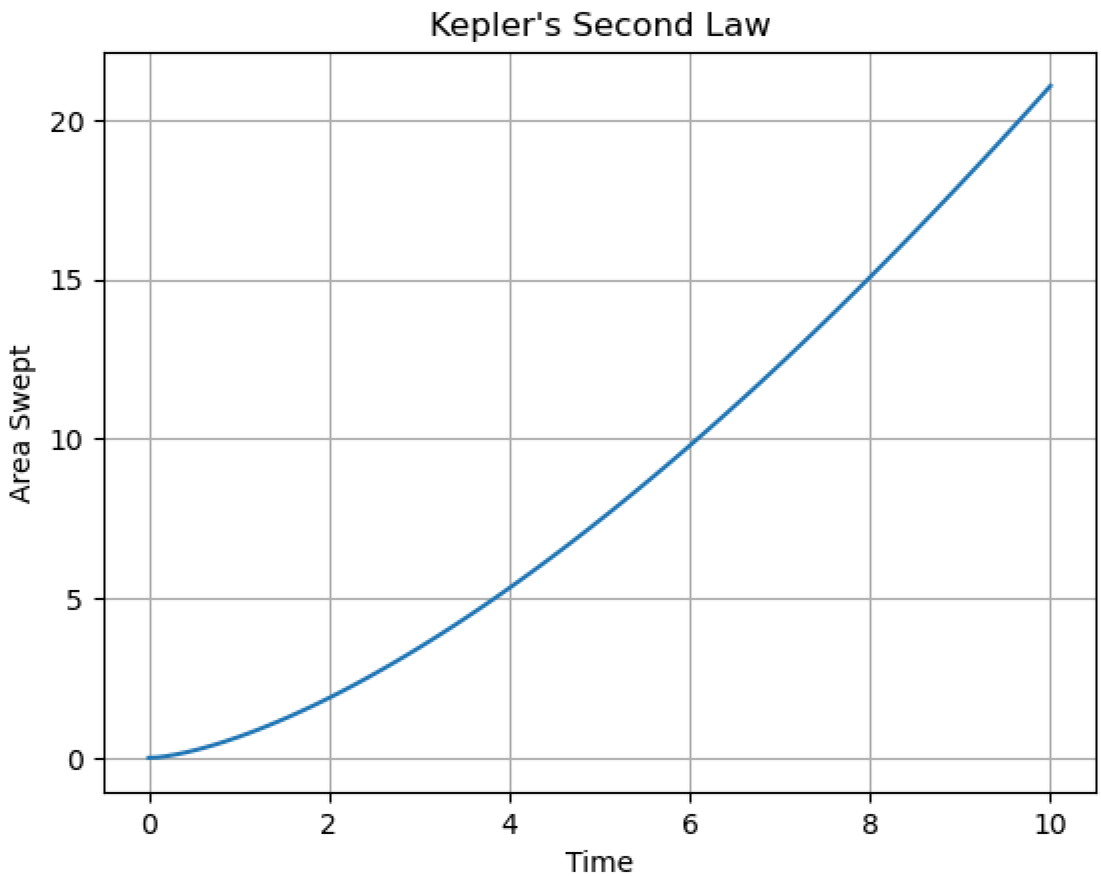


Those triangles do not look like they have equal areas...

Setting the center at (0,0)

In [7]:
```
# Function to calculate the swept area using triangles
def calculate_area(x, y):
    return 0.5 * np.sum((x[1:] - x[:-1]) * (y[:-1] + y[1:]))

# Starting parameters for first period
```

```python
time = np.linspace(0, 10, 100)  # Time intervals
radius = np.sqrt(time)  # Example radius (for demonstration purposes)

# Calculate areas swept by the line segment
areas = np.zeros_like(time)
for i in range(1, len(time)):
    area = calculate_area(time[:i+1], radius[:i+1])
    areas[i] = area

# Plotting
plt.plot(time, areas)
plt.xlabel('Time')
plt.ylabel('Area Swept')
plt.title("Kepler's Second Law")
plt.grid(True)
plt.show()
```




```
In [ ]:
```