```
!pip install gefera
```

```
Collecting gefera
  Downloading gefera-0.1.tar.gz (2.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.3/2.3 MB 6.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from gefera) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from gefera) (1.11.4)
Requirement already satisfied: astropy in /usr/local/lib/python3.10/dist-packages (from gefera) (5.3.4)
Requirement already satisfied: pyerfa>=2.0 in /usr/local/lib/python3.10/dist-packages (from astropy->gefera) (2.0.1.4)
Requirement already satisfied: PyYAML>=3.13 in /usr/local/lib/python3.10/dist-packages (from astropy->gefera) (6.0.1)
Requirement already satisfied: packaging>=19.0 in /usr/local/lib/python3.10/dist-packages (from astropy->gefera) (24.0)
Building wheels for collected packages: gefera
  Building wheel for gefera (setup.py) ... done
  Created wheel for gefera: filename=gefera-0.1-cp310-cp310-linux_x86_64.whl size=591293 sha256=081ba9076a500367edb64bc7bfbbc21ac0e9907edb88d9c589
  Stored in directory: /root/.cache/pip/wheels/63/8d/41/bf4921831e168858b005547ed6f3f77c61b80532447d528f5e
Successfully built gefera
Installing collected packages: gefera
Successfully installed gefera-0.1
```

```
import numpy as np
import matplotlib.pyplot as plt
import gefera as gf

class MultiPlanetSystem:
    def __init__(self):
        self.planets = []
        self.orbits = None
        self.system = None
        self.default_values = {
            'a': 1.0,  # Default semi-major axis
            't': 0.0,  # Default time of transit
            'e': 0.00001,  # Default eccentricity
            'p': 5.0,  # Default orbital period
            'w': 90.0,  # Default argument of periastron
            'om': 0.0, # Default longitude of ascending node
            'b': 0.0,  # Default impact parameter
            'i': 0.0,  # Default inclination
            'u1': 0.2, # Default limb darkening coefficient 1
            'u2': 0.2,  # Default limb darkening coefficient 2
            'r': 0.01  # Default radius ratio
        }

    def add_planet(self, a=None, t=None, e=None, p=None, w=None, om=None, b=None, u1=None, u2=None, r=None):
        planet_params = {
            'a': a if a is not None else self.default_values['a'],
            't': t if t is not None else self.default_values['t'],
            'e': e if e is not None else self.default_values['e'],
            'p': p if p is not None else self.default_values['p'],
            'w': w if w is not None else self.default_values['w'],
            'om': om if om is not None else self.default_values['om'],
            'b': b if b is not None else self.default_values['b'],
            'u1': u1 if u1 is not None else self.default_values['u1'],
            'u2': u2 if u2 is not None else self.default_values['u2'],
            'r': r if r is not None else self.default_values['r']
        }
        self.planets.append(planet_params)

    def initialize_orbits(self):
        if len(self.planets) < 2:
            raise ValueError("At least two planets are required to initialize orbits.")

        primary_params = self.planets[0]
        o1 = gf.orbits.PrimaryOrbit(primary_params['a'], primary_params['t'], primary_params['e'], primary_params['p'], primary_params['w'], np.arccos
        self.orbits = [o1]
        for planet_params in self.planets[1:]:
            o2 = gf.orbits.ConfocalOrbit(planet_params['a'], planet_params['t'], planet_params['e'], planet_params['p'], planet_params['om'], planet_p
            self.orbits.append(o2)
        self.system = gf.systems.ConfocalSystem(*self.orbits)

    def simulate_lightcurve(self, t):
        flux = np.zeros_like(t)
        for i, planet_params in enumerate(self.planets):
            u1 = planet_params['u1']
            u2 = planet_params['u2']
            r = planet_params['r']
            flux += self.system.lightcurve(t, u1, u2, r, 0.0 if i == 0 else self.planets[i-1]['r'], grad=False)
        return flux


f = open("/content/PS_2024.04.25_12.55.45.csv")
f= f.readlines(0)
# print(f)
```

```
f = f[147:]


list_all=[]
for i in f:
    list_all.append(i.split("\n"))

#print(list_all,end="\n\n\n\n")
word=[]
for i in list_all:
    for j in i:
        #print(j,"\n\n")
        s = j.split(",")
        #print(s)
        break
    word.append(s)

print(word[1])
```

```
['AU Mic b', 'AU Mic', '1', '1', '3', 'Transit', '2020', 'Transiting Exoplanet Survey Satellite (TESS)', 'Published Confirmed', '0', '<a refstr=CA
```

```
def find_partial_string_in_array(arr, target):
    indices = []
    for i, row in enumerate(arr):
        for j, item in enumerate(row):
            if target in item:  # Check if target is a substring of the current item
                indices.append([i, j])  # Append the index of the row containing the target string
    return indices


x= find_partial_string_in_array(word,"Kepler-51")

print(x)
for i in x:
    print(word[i[0]][i[1]])
```

```
[[1339, 0], [1339, 1], [1340, 0], [1340, 1], [1341, 0], [1341, 1], [1342, 0], [1342, 1], [1343, 0], [1343, 1]]
Kepler-51 b
Kepler-51
Kepler-51 c
Kepler-51
Kepler-51 d
Kepler-51
Kepler-511 b
Kepler-511
Kepler-511 c
Kepler-511
```

```
hostname=[]
orb_per=[]
for i in word:
    #print(i[1])
    if i[1] not in hostname:
        hostname.append(i[1])
    else:
        pass

    if i[13] not in hostname:
        orb_per.append(i[13])
    else:
        pass

orb_per=orb_per[1:]


exo_plnt_all=[]
hostname = hostname[1:]
for i in hostname:

    #print(i)
    x= find_partial_string_in_array(word,i)

    #print(x)
    exo_plnt_list=[]
    for j in x:
        exo_plnt_list.append(word[j[0]][j[1]])
    exo_plnt_all.append(exo_plnt_list)

print(hostname)
```

```
['AU Mic', 'CoRoT-20', 'CoRoT-24', 'CoRoT-7', 'K2-166', 'K2-16', 'K2-50', 'K2-168', 'EPIC 206024342', 'EPIC 206042996', 'EPIC 206317286', 'EPIC 21
```

```
planet={}
for key,value in zip(hostname, exo_plnt_all):
    if value not in planet.values():
        planet[key] = value
    else:
        pass
print(planet)
```

```
{'AU Mic': ['AU Mic b', 'AU Mic', 'AU Mic c', 'AU Mic', 'AU Mic d', 'AU Mic'], 'CoRoT-20': ['CoRoT-20 b', 'CoRoT-20'], 'CoRoT-24': ['CoRoT-24 b',
```

```
for i in hostname:
    value_to_check = i

    # Iterate over each key-value pair in the dictionary
    for key, value_list in planet.items():
        # Iterate over each item in the list associated with the current key
        for item in value_list:
            # Check if the value exists in the current item
            if item == value_to_check:
                value_list.remove(item)
        planet[key] = value_list
print(planet)
```

```
{'AU Mic': ['AU Mic b', 'AU Mic c', 'AU Mic d'], 'CoRoT-20': ['CoRoT-20 b'], 'CoRoT-24': ['CoRoT-24 b', 'CoRoT-24 c'], 'CoRoT-7': ['CoRoT-7 b'], '
```

```
import itertools

# Iterate over a copy of the dictionary because we'll modify it during iteration
for hostname, items in list(planet.items()):
    if len(items) == 1:  # If value list has a single element
        del planet[hostname]
    elif len(items) > 2:  # If value list has more than 2 elements, split into combinations of size 2
        combinations = list(itertools.combinations(items, 2))
        counter = 1
        for comb in combinations:
            planet[f"{hostname} (Combination {counter})"] = list(comb)
            counter += 1
        del planet[hostname]

print(planet)
```

```
{'CoRoT-24': ['CoRoT-24 b', 'CoRoT-24 c'], 'EPIC 206042996': ['EPIC 206042996 b', 'EPIC 206042996 c'], 'EPIC 206317286': ['EPIC 206317286 b', 'EPI
```

```
orb_per=[]
for i in word:


    if i[13] not in hostname:
        orb_per.append(i[13])
    else:
        pass


def find_orbital_period(csv_file, input_value):
    with open(csv_file, 'r') as file:
        file= file.readlines(0)
        file = file[147:]

        list_all=[]
        for i in f:
            list_all.append(i.split("\n"))

        #print(list_all,end="\n\n\n\n")
        word=[]
        for i in list_all:
            for j in i:
                #print(j,"\n\n")
                s = j.split(",")
                #print(s)
                break
            word.append(s)
        x= find_partial_string_in_array(word,input_value)
        #print(x[0][1])
        return(word[x[0][0]][11])



  # Assuming 'orbital_period' is the column header for the orbital period
```

```
def find_semi_major_axis(csv_file, input_value):
    with open(csv_file, 'r') as file:
        file= file.readlines(0)
        file = file[147:]

        list_all=[]
        for i in f:
            list_all.append(i.split("\n"))

        #print(list_all,end="\n\n\n\n")
        word=[]
        for i in list_all:
            for j in i:
                #print(j,"\n\n")
                s = j.split(",")
                #print(s)
                break
            word.append(s)
        x= find_partial_string_in_array(word,input_value)
        #print(x[0][1])
        if len(word[x[0][0]][15]) >0:
            return(word[x[0][0]][15])
        else:
            return


def find_eccentricity(csv_file, input_value):
    with open(csv_file, 'r') as file:
        file= file.readlines(0)
        file = file[147:]

        list_all=[]
        for i in f:
            list_all.append(i.split("\n"))

        #print(list_all,end="\n\n\n\n")
        word=[]
        for i in list_all:
            for j in i:
                #print(j,"\n\n")
                s = j.split(",")
                #print(s)
                break
            word.append(s)
        x= find_partial_string_in_array(word,input_value)
        #print(x[0][1])
        if len(word[x[0][0]][40])>0:
            return(word[x[0][0]][40])
        else:
            #print("yes")
            return


def find_inclination(csv_file, input_value):
    with open(csv_file, 'r') as file:
        file= file.readlines(0)
        file = file[147:]

        list_all=[]
        for i in f:
            list_all.append(i.split("\n"))

        #print(list_all,end="\n\n\n\n")
        word=[]
        for i in list_all:
            for j in i:
                #print(j,"\n\n")
                s = j.split(",")
                #print(s)
                break
            word.append(s)
        x= find_partial_string_in_array(word,input_value)
        #print(x[0][1])
        if len(word[x[0][0]][53])>0:
            return(word[x[0][0]][53])
        else:
            #print("yes")
            return
```

```
def find_transit_time(csv_file, input_value):
    with open(csv_file, 'r') as file:
        file= file.readlines(0)
        file = file[147:]

        list_all=[]
        for i in f:
            list_all.append(i.split("\n"))

        #print(list_all,end="\n\n\n\n")
        word=[]
        for i in list_all:
            for j in i:
                #print(j,"\n\n")
                s = j.split(",")
                #print(s)
                break
            word.append(s)
        x= find_partial_string_in_array(word,input_value)
        #print(x[0][1])
        if len(word[x[0][0]][69])>0:
            return(word[x[0][0]][69])
        else:
            #print("yes")
            return


def find_impacter_parameter(csv_file, input_value):
    with open(csv_file, 'r') as file:
        file= file.readlines(0)
        file = file[147:]

        list_all=[]
        for i in f:
            list_all.append(i.split("\n"))

        #print(list_all,end="\n\n\n\n")
        word=[]
        for i in list_all:
            for j in i:
                #print(j,"\n\n")
                s = j.split(",")
                #print(s)
                break
            word.append(s)
        x= find_partial_string_in_array(word,input_value)
        #print(x[0][1])
        if len(word[x[0][0]][61])>0:
            return(word[x[0][0]][61])
        else:
            #print("yes")
            return


def find_longitude_periastron(csv_file, input_value):
    with open(csv_file, 'r') as file:
        file= file.readlines(0)
        file = file[147:]

        list_all=[]
        for i in f:
            list_all.append(i.split("\n"))

        #print(list_all,end="\n\n\n\n")
        word=[]
        for i in list_all:
            for j in i:
                #print(j,"\n\n")
                s = j.split(",")
                #print(s)
                break
            word.append(s)
        x= find_partial_string_in_array(word,input_value)
        #print(x[0][1])
        if len(word[x[0][0]][85])>0:
            return(word[x[0][0]][85])
        else:
            #print("yes")
            return
```

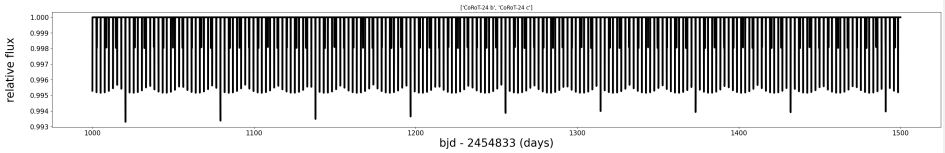
```
def find_radius_ratio(csv_file, input_value):
    with open(csv_file, 'r') as file:
        file= file.readlines(0)
        file = file[147:]

        list_all=[]
        for i in f:
            list_all.append(i.split("\n"))

        #print(list_all,end="\n\n\n\n")
        word=[]
        for i in list_all:
            for j in i:
                #print(j,"\n\n")
                s = j.split(",")
                #print(s)
                break
            word.append(s)
        x= find_partial_string_in_array(word,input_value)
        #print(x[0][1])
        if len(word[x[0][0]][99]) and len(word[x[0][0]][19])>0:
            # print(len(word[x[0][0]][99]),len(word[x[0][0]][19]))
            return(float(word[x[0][0]][19])/float(word[x[0][0]][99])*0.00916794)
        else:
            #print("yes")
            return None


def find_sr(csv_file, input_value):
    with open(csv_file, 'r') as file:
        file= file.readlines(0)
        file = file[147:]

        list_all=[]
        for i in f:
            list_all.append(i.split("\n"))

        #print(list_all,end="\n\n\n\n")
        word=[]
        for i in list_all:
            for j in i:
                #print(j,"\n\n")
                s = j.split(",")
                #print(s)
                break
            word.append(s)
        x= find_partial_string_in_array(word,input_value)
        #print(x[0][1])
        if len(word[x[0][0]][99])>0:
            return(word[x[0][0]][99])
        else:
            #print("yes")
            return


planet.pop('HD 39091')
```

```
     []
```

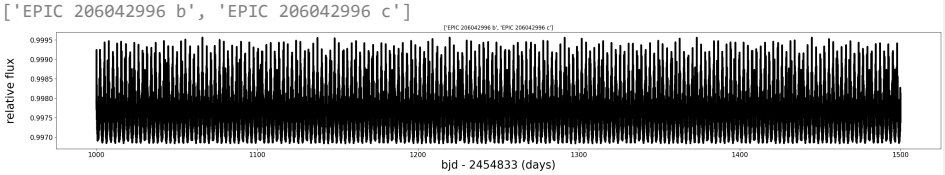
```
import os


# Define the parameters
t = np.linspace(1000, 1500, 10000)

# Create a directory to save figures and text files
output_folder = "final_plot_data"
os.makedirs(output_folder, exist_ok=True)

for host,planet_name in planet.items():
# # Iterate through the planet data and retrieve parameters
# planet_data = [
#     {"name": "Kepler-51", "items": ["Kepler-51 d", "Kepler-51 b"]}
# ]

# Iterate through planet data
    system = MultiPlanetSystem()
    # Prepare to save printed values into a text file
    output_file_path = os.path.join(output_folder, f"{host}_parameters.txt")
    with open(output_file_path, "w") as text_file:

        for item in planet_name:

            a = find_semi_major_axis("/content/PS_2024.04.25_12.55.45.csv", item)
            if a is None:
                a = None
```

```
        else:
            a = float(find_semi_major_axis("/content/PS_2024.04.25_12.55.45.csv", item))*244.99
        # print(a)

        tv = find_transit_time("/content/PS_2024.04.25_12.55.45.csv", item)
        if tv is None:
            tv = None
        else:
            tv = float(find_transit_time("/content/PS_2024.04.25_12.55.45.csv", item)) - 2454833

        e = find_eccentricity("/content/PS_2024.04.25_12.55.45.csv", item)
        if e is None:
            e = None
        else:
            e = float(find_eccentricity("/content/PS_2024.04.25_12.55.45.csv", item))

        p = find_orbital_period("/content/PS_2024.04.25_12.55.45.csv", item)
        if p is None:
            p = None
        else:
            p = float(find_orbital_period("/content/PS_2024.04.25_12.55.45.csv", item))
        # print(p)

        w = find_longitude_periastron("/content/PS_2024.04.25_12.55.45.csv", item)
        if w is None:
            w = None
        else:
            w = float(find_longitude_periastron("/content/PS_2024.04.25_12.55.45.csv", item))

        b = find_impacter_parameter("/content/PS_2024.04.25_12.55.45.csv", item)
        if b is None:
            b = None
        else:
            b = float(find_impacter_parameter("/content/PS_2024.04.25_12.55.45.csv", item))

        r = find_radius_ratio("/content/PS_2024.04.25_12.55.45.csv", item)
        if r is None:
            r = None
        else:
            r = float(find_radius_ratio("/content/PS_2024.04.25_12.55.45.csv", item))
        print(r)

        om = 180 * np.pi / 180

        u1 = 0.6
        u2 = 0.2

        print(a, tv, e, p, w, om, b, u1, u2, r)
        system.add_planet(a, tv, e, p, w, om, b, u1, u2, r)
        # Write the parameters to the text file
        parameters_str = f"{a}, {tv}, {e}, {p}, {w}, {b}, {r}\n"
        text_file.write(parameters_str)

# Initialize orbits and system
system.initialize_orbits()
# print(t)
# Simulate the light curve
flux = system.simulate_lightcurve(t)

# Plot the simulated light curve
plt.figure(figsize=(30, 5))
plt.plot(t, flux + 1, color='k', linewidth=4)

plt.ylabel('relative flux\n', fontsize=25)
plt.xlabel('bjd - 2454833 (days)', fontsize=25)
plt.title("%s"%planet_name)
print("%s"%planet_name)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.tight_layout()
plt.subplots_adjust(top=0.9)
plt.show()

# Save the figure in the output folder
figure_file_path = os.path.join(output_folder, f"{host}_lightcurve.png")
# plt.savefig(figure_file_path)

# Show the figure
plt.show()
```

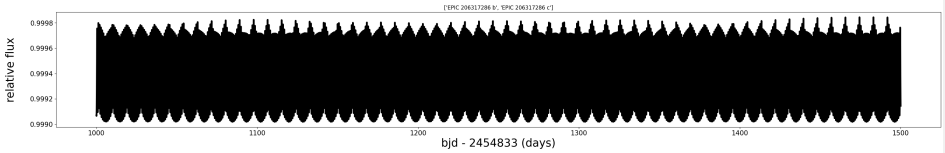
0.03944346279069768
13.71944 -2454830.15 0.0 5.1134 None 3.141592653589793 0.65 0.6 0.2 0.03944346279069768
0.05330197674418604
24.009020000000003 -2454828.2 0.0 11.759 None 3.141592653589793 0.3 0.6 0.2 0.05330197674418
['CoRoT-24 b', 'CoRoT-24 c']



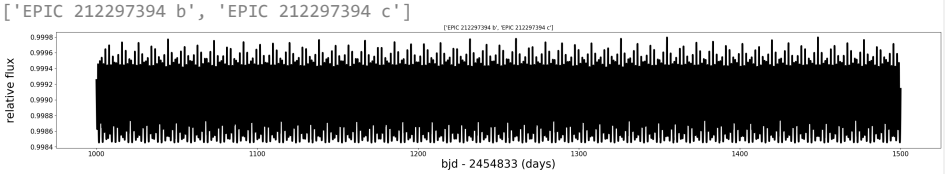
0.031170995999999996
None None None 5.2986 None 3.141592653589793 None 0.6 0.2 0.031170995999999996
0.022003055999999997
None None None 0.354884 None 3.141592653589793 None 0.6 0.2 0.022003055999999997
['EPIC 206042996 b', 'EPIC 206042996 c']



None
None None None 1.58252 None 3.141592653589793 0.57 0.6 0.2 None
0.02526008731578947
None None None 17.515472 None 3.141592653589793 0.268 0.6 0.2 0.02526008731578947
['EPIC 206317286 b', 'EPIC 206317286 c']



0.016960688999999998
None -2454831.12 None 2.289363 None 3.141592653589793 0.41 0.6 0.2 0.016960688999999998
0.026014029749999997
None -2454830.38 None 5.213965 None 3.141592653589793 0.39 0.6 0.2 0.026014029749999997
['EPIC 212297394 b', 'EPIC 212297394 c']



0.011787351428571428
None -2454829.54 None 15.28078 None 3.141592653589793 0.43 0.6 0.2 0.011787351428571428
0.021703694693877548
None -2454829.75 None 23.228555 None 3.141592653589793 0.43 0.6 0.2 0.021703694693877548
['EPIC 212587672 b', 'EPIC 212587672 c']