PROBLEMA 1

La densidad media del satélite Titán es de 1,88 g/cm^3. Como es una densidad pequeña, esto nos sugiere que en su composición predominarán materiales poco densos como el agua helada, el amoniaco o el metano, en comparación con la cantidad de metales más densos o rocas, que habrá en menor cantidad.

In [ ]:
```python
# calcular la densidad de Titán
# Titán es una luna de Saturno
# su radio es 2576 km
# su masa es 1.35 x 10^23 kg
# la densidad se calcula como masa/volumen
# el volumen de una esfera es 4/3 pi r^3
import math
r = 2576e3
m = 1.35e23
v = 4/3 * math.pi * r**3
d = m/v
print("La densidad de Titán es", d, "kg/m^3")
```

```
La densidad de Titán es 1885.4177142288502 kg/m^3
```

In [ ]:
```python
# PROBLEMA 2
# Apartado 1
# El hielo sólido de agua pura tiene un albedo A ≈ 0.35.
# ¿Cuál es la distancia mínima al Sol a la que un cubo de hielo que rota ráp
# [Temperatura de sublimación del hielo en vacío Tsub ~ 200 K.]
import math
# Constantes
A = 0.35  # Albedo del hielo
sigma = 5.67 * 10 ** -8  # Constante de Stefan-Boltzmann en W/m^2 K^4
L = 3.828 * 10 ** 26  # Luminosidad del Sol en vatios (W)
Tsub = 200  # Temperatura de sublimación del hielo en K

# Cálculo de la distancia mínima al Sol
d = math.sqrt(L * (1 - A) / (16 * math.pi * sigma * Tsub**4))

# Imprimir el resultado
print(f"La distancia mínima al Sol a la que un cubo de hielo que rota rápida
#cambiar unidades de metros a UA
d = d / (1.496 * 10 ** 11)
print(f"La distancia mínima al Sol a la que un cubo de hielo que rota rápida

# Apartado 2
# ¿Entre qué dos planetas se encuentra esta distancia?
# Distancias medias al Sol de los planetas del sistema solar en m
distancias = {
    "Mercurio": 5.79 * 10 ** 10,
    "Venus": 1.08 * 10 ** 11,
    "Tierra": 1.496 * 10 ** 11,
    "Marte": 2.28 * 10 ** 11,
```

```
    "Júpiter": 7.78 * 10 ** 11,
    "Saturno": 1.43 * 10 ** 12,
    "Urano": 2.87 * 10 ** 12,
    "Neptuno": 4.50 * 10 ** 12,
    "Plutón": 5.91 * 10 ** 12
}
# Se encuentra entre Marte y Júpiter.
```

```
La distancia mínima al Sol a la que un cubo de hielo que rota rápidamente pe
rmanecería congelado es 233591047623.6936 metros.
La distancia mínima al Sol a la que un cubo de hielo que rota rápidamente pe
rmanecería congelado es 1.561437484115599 UA.
```

In [ ]:

```
# PROBLEMA 3
from astropy import units as u
#Calisto, the moon of Jupiter, rotates slowly and has low albedo (A=0.2).
#Calculate the temperature of the subsolar point of Calisto
#Knowing that the temperature of the subsolar point of the Earth is 395K.
TsubT=395
A=0.2
#The distance between Jupiter and the Sun is 5.2 AU
d_jupi=5.203*u.AU
#The distance between Jupiter and Calisto is 1.88e6 km
d_cal=1.883e6*u.km
d_calisto_sun=d_jupi-d_cal

TsubC=(TsubT*(1-A)**0.25)*(d_calisto_sun/(1*u.AU))**(-0.5)
print("La temperatura del punto subsolar de Calisto es",TsubC,"K")

#Apartado 2
#Do you expect to Calisto to retain an N2 atmosphere? use mu=7.1*(T_ex/1000K
#Knowing that the mass of Calisto is 1.08e23 kg and its radius is 2410 km.
masa=1.08e23
radio=2410e3
mu=7.1*(TsubC/1000)*(masa/5.97e24)**(-1)*(radio/6371e3)
print("El valor de mu es",mu)
#So Calisto can retain an N2 atmosphere because mu is smaller than the atomi

#Apartado 3
#Calisto can not ratain an He atmosphere because mu is greater than the atom
```

```
La temperatura del punto subsolar de Calisto es 163.97160121119916 K
El valor de mu es 24.343723102802674
```

Apartado 2

Calisto podría retener una atmósfera de N2 ya que mu es más pequeño que la masa atómica de N2

Apatado 3

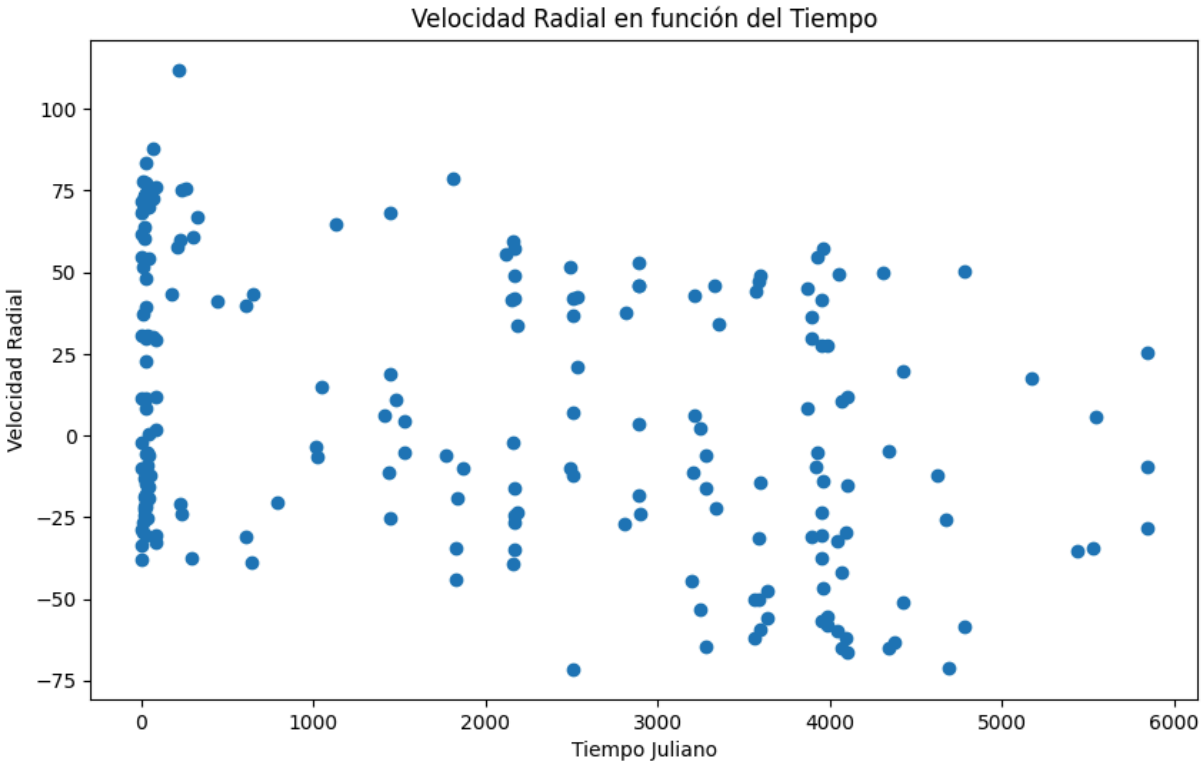
Calisto no podría retener una atmósfera de He porque mu es mayor que la masa atómica de He.

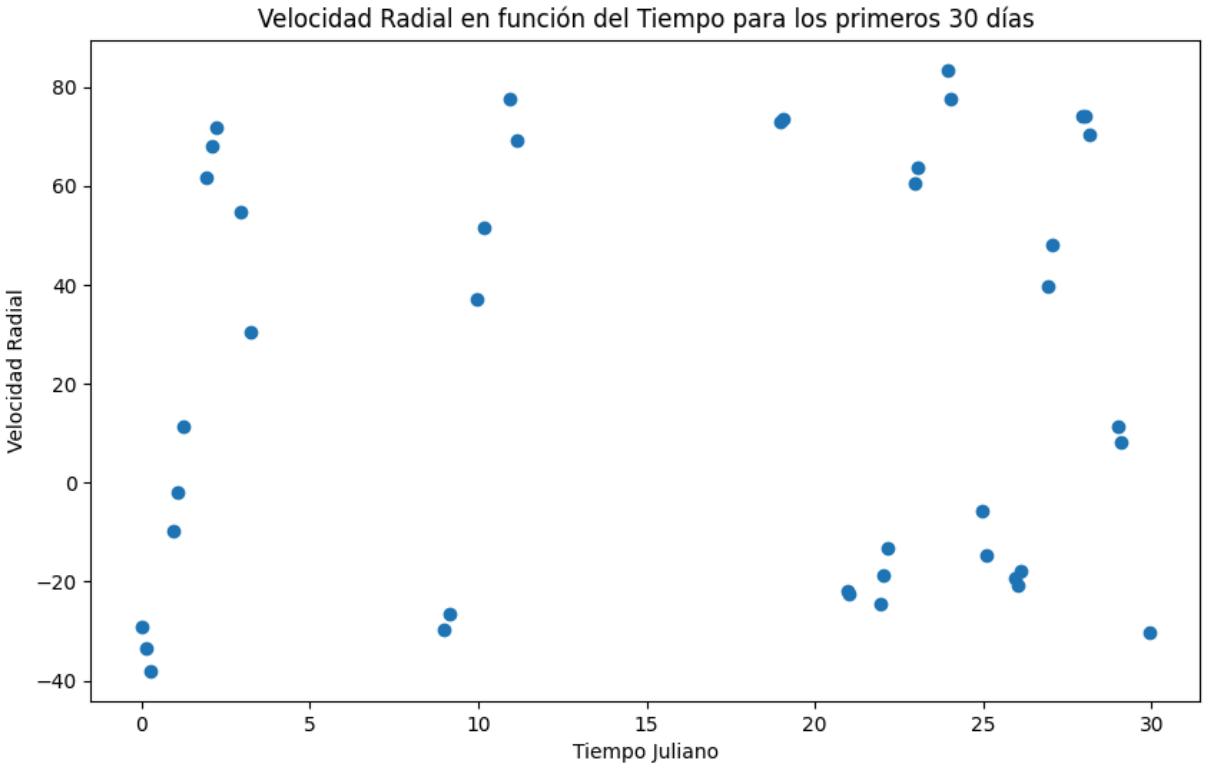
In [ ]:

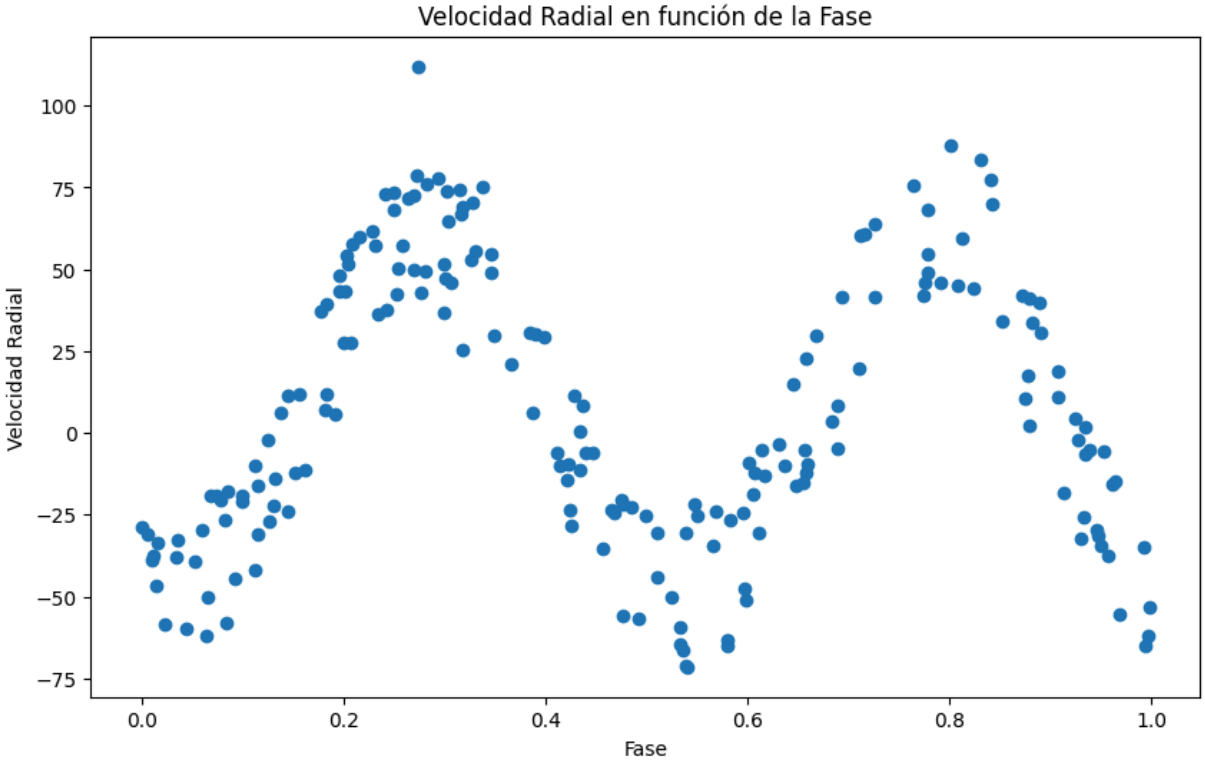
```
import pandas as pd
# PROBLEMA 4
# Apartado 1
# Cargar el archivo
data = pd.read_csv('UID_0113357_RVC_005.tbl', skiprows=22, delim_whitespace=
# Mostrar las primeras líneas de los datos
print(data.head())

# Apartado 2
# Redefinir el tiempo fijando el origen en la primera observación
data['Tiempo Juliano'] = data['Tiempo Juliano'] - data['Tiempo Juliano'].ilc
# Mostrar las primeras líneas de los datos
print(data.head())

# Apartado 3
import matplotlib.pyplot as plt
# Crear un gráfico de la velocidad radial en función del tiempo
plt.figure(figsize=(10, 6))
plt.plot(data['Tiempo Juliano'], data['Velocidad Radial'], 'o')
plt.title('Velocidad Radial en función del Tiempo')
plt.xlabel('Tiempo Juliano')
plt.ylabel('Velocidad Radial')
plt.show()

# Apartado 4
# Filtrar los datos para los primeros 30 días
data_30 = data[data['Tiempo Juliano'] <= 30]
# Crear un gráfico de la velocidad radial en función del tiempo para los pri
plt.figure(figsize=(10, 6))
plt.plot(data_30['Tiempo Juliano'], data_30['Velocidad Radial'], 'o')
plt.title('Velocidad Radial en función del Tiempo para los primeros 30 días'
plt.xlabel('Tiempo Juliano')
plt.ylabel('Velocidad Radial')
plt.show()

# Apartado 5
import numpy as np
# Definir el periodo
P = 4.230785  # días
T = 2 * P  # dos periodos
# Calcular la fase para cada observación
data['Fase'] = data['Tiempo Juliano'] / T - np.floor(data['Tiempo Juliano']
# Crear un gráfico de la velocidad radial en función de la fase
plt.figure(figsize=(10, 6))
plt.plot(data['Fase'], data['Velocidad Radial'], 'o')
plt.title('Velocidad Radial en función de la Fase')
plt.xlabel('Fase')
plt.ylabel('Velocidad Radial')
plt.show()

# Apartado 6 y 7
#Define the sinusoidal function
```

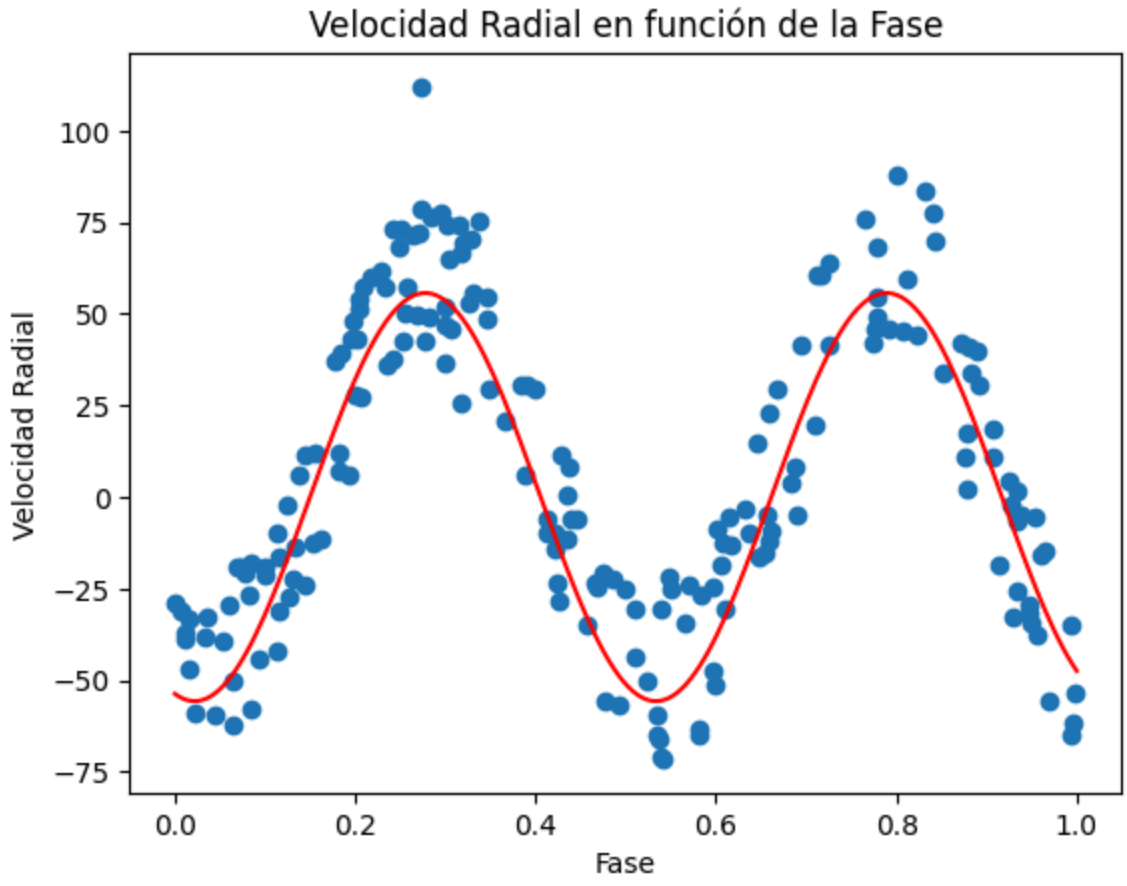
```
def f(x,A, B,C):
    return A*np.sin(2*np.pi*B*x+C)
#Realize the fit of the function
xx=np.linspace(0,1,len(data["Fase"]))
popt,pcov=curve_fit(f,data["Fase"],data["Velocidad Radial"])
A=popt[0]
B=popt[1]
C=popt[2]
#Apartado 7
#Plot the fit
plt.scatter(data["Fase"],data["Velocidad Radial"])
plt.plot(xx,A*np.sin(2*np.pi*B*xx+C), color="red")
plt.xlabel("Fase")
plt.ylabel("Velocidad Radial")
plt.title("Velocidad Radial en función de la Fase")
plt.show()

# Apartado 8
print(f"La variación de la velocidad radial observada se debe a la presencia

# Apartado 9
# Constantes
A=popt[0]
A=A*u.m/u.s
Mstar=1.2*to.M_sun
Mstar=Mstar.to(u.kg)
G=to.G
P=4.230785*u.day
P=P.to(u.s)
a=(A/2)*P
Mplanet=4*np.pi**2*(Mstar)**2*(a)**3/(G*P**2)
Mplanet=Mplanet**(1/3)
print("La masa del planeta es",Mplanet,"kg")

print(f"Las aproximaciones realizadas en este cálculo son que la órbita del
```

```
   Tiempo Juliano  Velocidad Radial  Error en la Velocidad Radial
0    2.450003e+06        -28.980785                      1.345139
1    2.450003e+06        -33.494823                      1.467521
2    2.450003e+06        -38.008003                      1.840782
3    2.450004e+06         -9.878883                      1.157139
4    2.450004e+06         -2.043795                      0.981048
   Tiempo Juliano  Velocidad Radial  Error en la Velocidad Radial
0         0.00000        -28.980785                      1.345139
1         0.13484        -33.494823                      1.467521
2         0.28840        -38.008003                      1.840782
3         0.95363         -9.878883                      1.157139
4         1.05979         -2.043795                      0.981048
C:\Users\recio\AppData\Local\Temp\ipykernel_2760\3984140423.py:5: FutureWarn
ing: The 'delim_whitespace' keyword in pd.read_csv is deprecated and will be
removed in a future version. Use ``sep='\s+'`` instead
  data = pd.read_csv('UID_0113357_RVC_005.tbl', skiprows=22, delim_whitespac
e=True, names=['Tiempo Juliano', 'Velocidad Radial', 'Error en la Velocidad
Radial'])
```







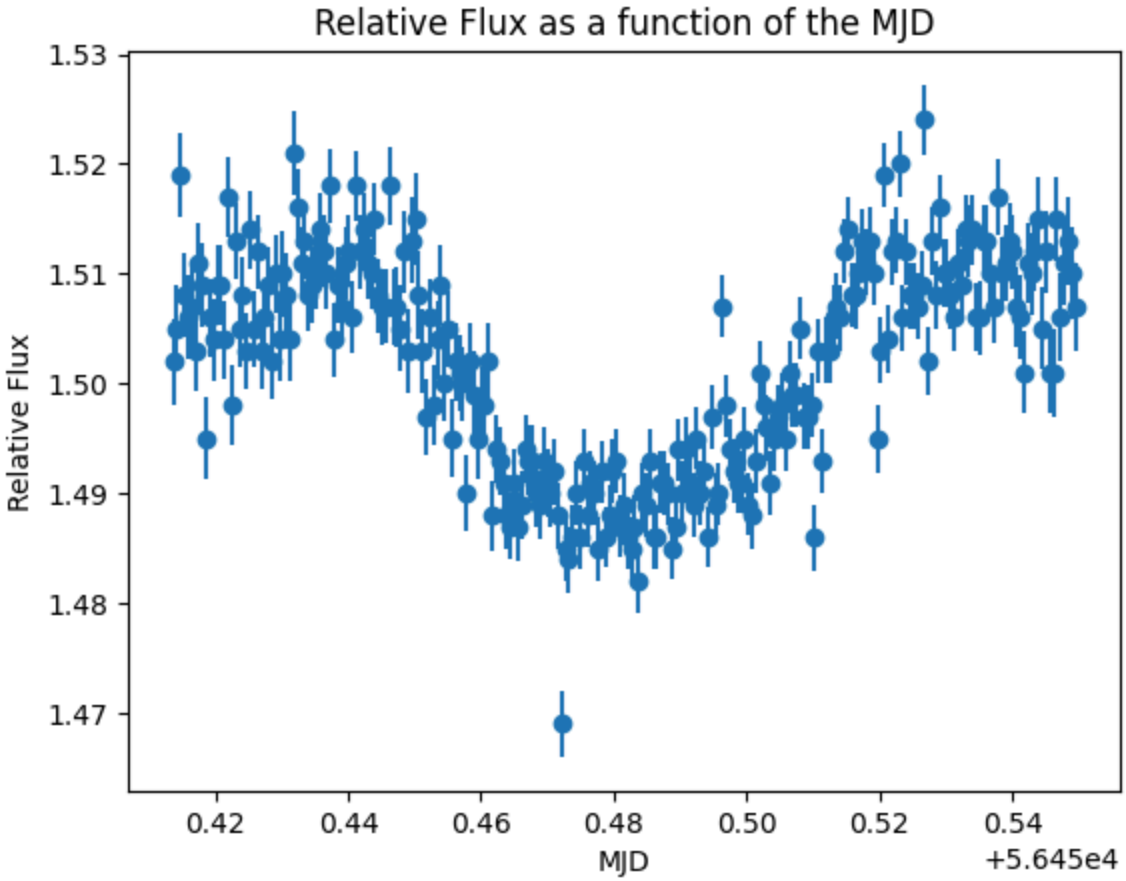


La variación de la velocidad radial observada se debe a la presencia de un p
laneta en órbita alrededor de la estrella. La curva ajustada representa la s
eñal de la velocidad radial causada por el planeta en órbita. Los parámetros
A, B y C del ajuste son importantes para caracterizar la órbita del planeta,
como la amplitud, el periodo y la fase de la señal.
La masa del planeta es 2.985741673613497e+27 kg kg
Las aproximaciones realizadas en este cálculo son que la órbita del planeta
está alineada de forma que se ve el máximo efecto de la velocidad radial y q
ue la masa de la estrella es mucho mayor que la masa del planeta, lo que per
mite simplificar la expresión de la masa del planeta. Además, l aecuación no
tiene en cuenta la excentricidad de la órbita del planeta, lo que podría afe
ctar a la precisión de la estimación de la masa del planeta.

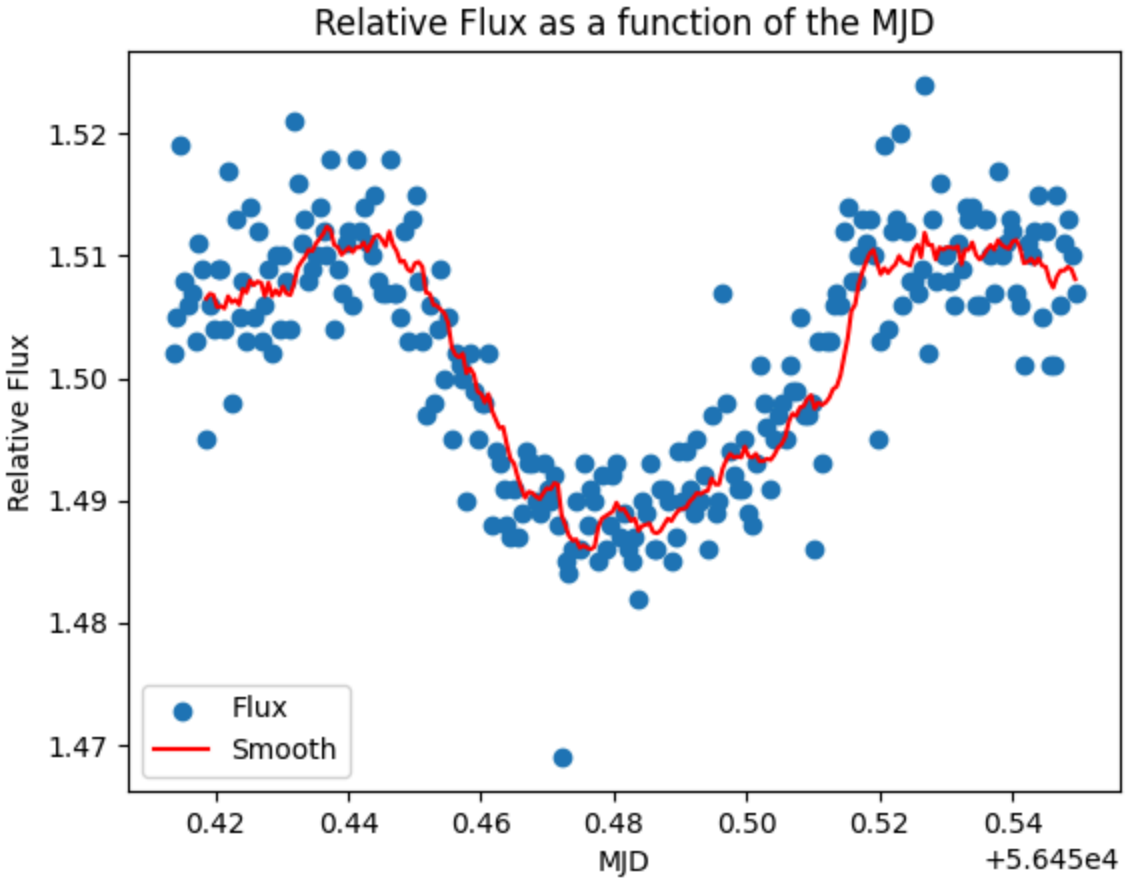
In [ ]:
```
# PROBLEMA 5
#Consider the light curve of the sistem TrES-2 taken by the telescope Oskar-
#The data is in the file tres2_data.dat, that has the columns: Modified Juli

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
from astropy import units as u
import astropy.constants as to

#Apartado 1
#Charge the data, knowing that the header occupies 2 lines.
data=pd.read_table("tres2_data.dat",sep="\s+",skiprows=2,header=None,names=[
print(data)

#Apartado 2
#Represent the relative flux as a function of the MJD
plt.errorbar(data["MJD"],data["Flux"],yerr=data["Error"],fmt="o")
plt.xlabel("MJD")
plt.ylabel("Relative Flux")
plt.title("Relative Flux as a function of the MJD")
plt.show()

#Apartado 3
#Explica brevemente a que se puede deber la variacion del flujo observada.
print("La variación del flujo observada se debe a la presencia de un exoplar

#Apartado 4
#To reduce the noise, smooth the light curve using a moving average of 10 da
def Moving_Average(data,window):
    return data.rolling(window=window).mean()
data["Smooth"]=Moving_Average(data["Flux"],10)
#Apartado 5
#Repeat the initial plot, but now adding the smoothed data
plt.scatter(data["MJD"],data["Flux"],label="Flux")
plt.plot(data["MJD"],data["Smooth"],label="Smooth", color="red")
plt.xlabel("MJD")
plt.ylabel("Relative Flux")
plt.title("Relative Flux as a function of the MJD")
plt.legend()
plt.show()
```

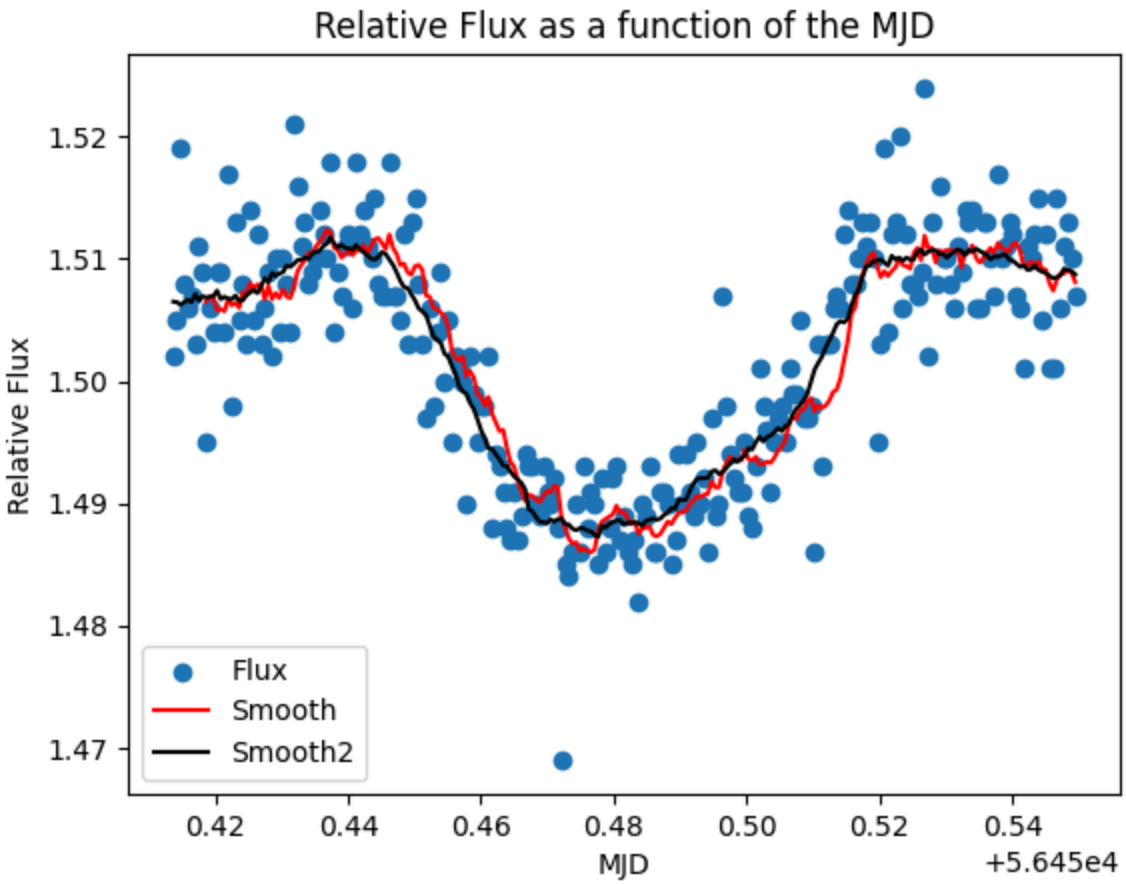
```python
#Apartado 6
def moving_median(data, window_size):
    N = len(data)
    avg = np.zeros(N)
    for i in range(N):
        start = max(0, i - window_size)
        end = min(N, i + window_size)
        avg[i] = np.mean(data[start:end])
    return avg
data["Smooth2"]=moving_median(data["Flux"],10)

plt.scatter(data["MJD"],data["Flux"],label="Flux")
plt.plot(data["MJD"],data["Smooth"],label="Smooth", color="red")
plt.plot(data["MJD"],data["Smooth2"],label="Smooth2", color="black")
plt.xlabel("MJD")
plt.ylabel("Relative Flux")
plt.title("Relative Flux as a function of the MJD")
plt.legend()
plt.show()

#Apartado 7
#Using the last plot, determinate the relation between the planet`s radius a
#El cociente entre ambos radios es igual a la raiz cuadrada del cociente ent
Cociente= np.sqrt(0.02/1.5)
print("El cociente entre el radio del planeta y el radio de la estrella es",

#Apartado 8
#Using the last plot, determinate the radius of the planet, knowing that the
P=2.47061*u.day
P=P.to(u.year)
Mstar=0.98
G=to.G
t=(0.48-0.44)*u.day
t=t.to(u.hour)
Rplanet=54000*u.km*(Mstar)**(1/3)*(P/(1*u.year))**(-1/3)*(t/(1*u.hour))
print("El radio del planeta es",Rplanet)
print("Las aproximaciones realizadas en este cálculo son que como la masa de
```

```
             MJD   Flux   Error
0    56450.41373  1.502  0.0039
1    56450.41427  1.505  0.0039
2    56450.41482  1.519  0.0039
3    56450.41537  1.508  0.0038
4    56450.41592  1.506  0.0038
..           ...    ...     ...
242  56450.54730  1.506  0.0040
243  56450.54784  1.511  0.0039
244  56450.54839  1.513  0.0040
245  56450.54894  1.510  0.0042
246  56450.54950  1.507  0.0040

[247 rows x 3 columns]
```



La variación del flujo observada se debe a la presencia de un exoplaneta que orbita la estrella. Cuando el planeta pasa por delante de la estrella, se pr oduce un eclipse y la luz que llega a la Tierra disminuye, lo que se traduce en una disminución del flujo observado.





El cociente entre el radio del planeta y el radio de la estrella es 0.115470
05383792516
El radio del planeta es 272271.61118188 km
Las aproximaciones realizadas en este cálculo son que como la masa de la est
rella es similar a la del Sol, se asume que el radio de la estrella también
es similar al del Sol. Además, se asume que el tránsito es total y que la es
trella es esférica, lo que simplifica el cálculo del radio del planeta.