Estudio de los modos forzados del planeta 9

Este código está destinado al estudio de los modos forzados del planeta 9 (P9). Concretamente, se estudian los datos generados por el código "data_generator" que a su vez utiliza el código evosecular para calcular la evolución secular de los elementos orbitales de un conjunto de particulas con diferentes semi-ejes perturbadas por el P9. En este caso se consideró el intervalo de semi-ejes de 200 a 800 ua (los datos se encuentran divididos en 3 archivos).

1. Importación de paquetes y librerías

En esta sección se importan los paquetes y librerías necesarias.

In [1]:
```
import numpy as np # numpy
import pandas as pd # pandas
import warnings # esto es para sacar warnings molestos
import matplotlib.pyplot as plt # pyplot para gráficos
import matplotlib as mpl # esto es para que los gráficos sean más lindos
import seaborn as sns # esto también es para tener gráficos más bonitos
from scipy import interpolate # de acá se importan funciones para hacer interpolaciones con pol
```

In [2]:
```
warnings.filterwarnings("ignore")
sns.set_context("talk")
mpl.style.use("seaborn")
sns.set_context("paper",font_scale=1.5)
sns.set_style("whitegrid")
```

2. Lectura y formateo de los datos

En esta sección se leen los datos y formatean de modo que queden guardados en formato de un DataFrame, además se define el rango de semi-ejes a estudiar en aall.

In [3]:
```
amin = 200 # minimo semi-eje
amax = 800 # maximo semi-eje
da = 20 # paso en los semi-ejes
deltaa = [200,400,600]
aall = np.arange(amin,amax+da,da) # rango de semi-ejes a estudiar (CUIDADO!!! debe ser un subco
# se leen los datos de los distintos randos y se junta todo en un DataFrame data
alldata = []
for a in deltaa:
    dataa = pd.read_csv(f"evosecularout_{a}_{a+200}_{da}.sal",sep=" ").drop(["Unnamed: 0"],axis
    dataa = dataa[dataa["a"] != "a"] # se eliminan las filas con los header de cada integración
    dataa = dataa.astype(float) # se cambia el tipo de objeto de todos los elementos de data a
    alldata += [dataa]
data = pd.concat([alldata[0],alldata[1],alldata[2]],axis=0)
data
```

Out[3]:

	T	a	e	i	lnod	lper	de/dt	di/dt	dln/dt	dlp/dt	
0	0.000000e+00	200.0	0.300	0.1000	0.0000	0.0000	-1.779130e-15	-2.932640e-17	-1.522350e-07	4.149400e-08	
1	3.192000e+03	200.0	0.300	0.1000	359.9995	0.0001	-3.155900e-15	2.777250e-14	-1.522350e-07	4.149400e-08	
2	6.384000e+03	200.0	0.300	0.1000	359.9990	0.0003	-4.532680e-15	-1.398700e-14	-1.522350e-07	4.149400e-08	
3	9.576000e+03	200.0	0.300	0.1000	359.9985	0.0004	-5.896590e-15	2.772730e-14	-1.522350e-07	4.149400e-08	
4	1.276800e+04	200.0	0.300	0.1000	359.9981	0.0005	-7.273360e-15	-1.198910e-16	-1.522350e-07	4.149400e-08	

	T	a	e	i	lnod	lper	de/dt	di/dt	dln/dt	dlp/dt
...	...	...	...	...	...	...	...	...	...	...
1069152	1.999870e+09	800.0	0.293	0.0706	275.5933	3.1367	-5.559520e-11	-2.542790e-11	-3.115330e-07	1.045810e-08
1069153	1.999900e+09	800.0	0.293	0.0706	275.5854	3.1369	-5.559990e-11	-2.540850e-11	-3.115390e-07	1.045720e-08
1069154	1.999920e+09	800.0	0.293	0.0706	275.5774	3.1372	-5.560470e-11	-2.538910e-11	-3.115450e-07	1.045640e-08
1069155	1.999950e+09	800.0	0.293	0.0706	275.5694	3.1375	-5.560940e-11	-2.536480e-11	-3.115520e-07	1.045550e-08
1069156	1.999970e+09	800.0	0.293	0.0706	275.5615	3.1377	-5.561410e-11	-2.530610e-11	-3.115580e-07	1.045470e-08

6968625 rows × 13 columns

3. Control de las integraciones

En esta sección se realizan gráficos de control para averiguar en que rangos la aproximaciones realizadas para el cálculo de la evolución secular fallan. En efecto, recordemos que el programa evosecular hace uso de la teoría semi-analítica para integrar la evolución secular de los elementos orbitales. Concretamente, en cada paso, este código se encarga de calcular numéricamente la función perturbadora secular (promediando su valor en las anomalías medias de P9 y de la partícula) y sus derivadas parciales respecto a los elementos orbitales para predecir los elementos orbitales de la órbita de la partícula en el siguiente paso. En este desarrollo, se promedia la función perturbadora R en un intervalo de tiempo Δt (que en este caso corresponde a 2 períodos Keplerianos de la partícula según explicado en la notebook data_generator). Es entonces claro que al realizar este cálculo se pierde información sobre la evolución por parte de las variables rápidas del sistema. Si bien esta evolución es despreciable cuando la partícula se encuentra lejos de P9 ($\frac{a}{a_9} >> 1$ o $\frac{a}{a_9} << 1$), las evoluciones de corto período ya no son despreciable cerca de la órbita del P9. Es entonces esperable que la evolución secular ya no sea representativa de la evolución real de la partícula cuando $a \approx a_9$. Es entonces necesario determinar en que intervalos de semi-ejes no es válido el uso de la teoría semi-analítica para estudiar la evolución secular con el fin de determinar los modos forzados.

Para determinar dicho intervalo de semi-ejes, recordemos que podemos escribir el Hamiltoniano $\mathcal{H}$ como:

$$\mathcal{H}(\vec{r}) = H + R(a, e, i, \varpi, \lambda, \lambda_9)$$

Donde H es el Hamiltoneano Kepleriano que depende del potencial central de la estrella, más el del planeta y de la energía cinética de la partícula. Ahora, notemos por un lado que H es constante y por otro que al ser la fuerza gravitatoria una fuerza conservativa, tenemos que R debe ser constante. Pero al promediar R, se eliminan las variables rápidas, luego, considerando únicamente la evolución secular, tenemos:

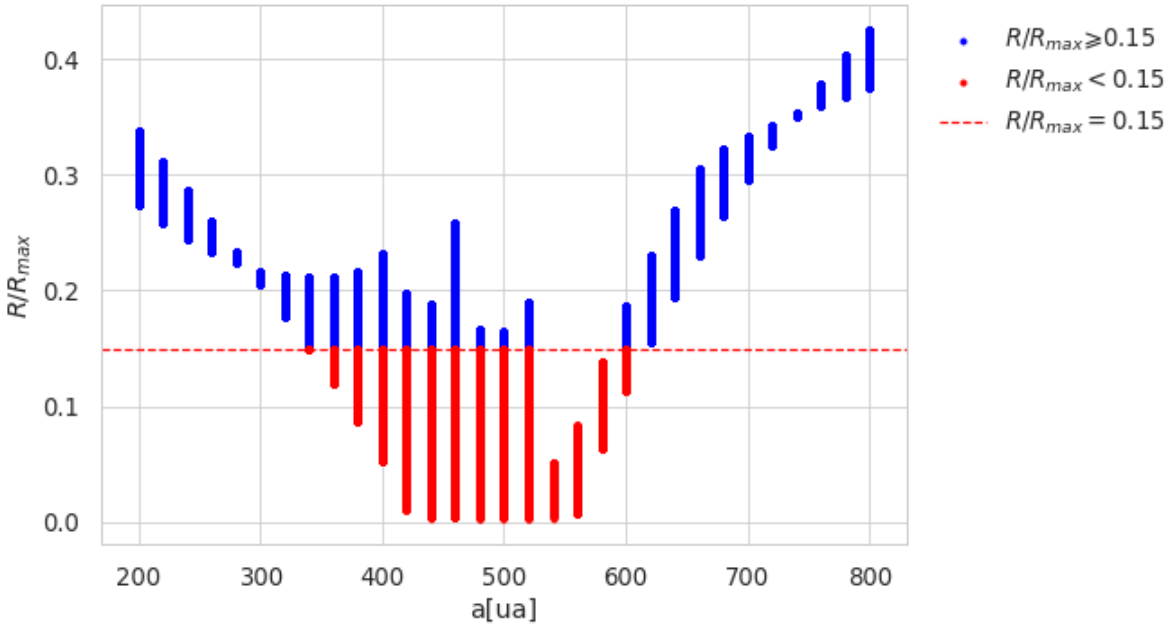
$$\mathcal{H}_{sec}(\vec{r}) = H + R_{sec}(a, e, i, \varpi)$$

Por lo cual, para que la teoría secular semi-analítica sea aplicable, se debe cumplir como consecuencia de esta que $R_{sec}(a, e, i, \varpi) \approx R_{sec}(a_0, e_0, i_0, \varpi_0)$ en todo instante, siendo el sub-índice 0 indicador de los elementos orbitales iniciales de la partícula. Ahora, es importante recordar de la ecuación planetaria de Lagrange para la evolución del semi-eje que en el caso secular no hay variación de dicho elemento orbital, luego, la función perturbadora secular debería mantenerse cuasi constante para un semi eje dado sin variar el valor de este.

3.1. Control estudio de la evolución de R/R_{max}

En base a estos argumento, para estimar el nivel de certeza de las integraciones, se grafica a continuación el cociente R/R_{max} como función del semi-eje a de la partícula (puntos azules) donde R_{max} corresponde al máximo valor de R calculado para cada valor de semi-eje. Si durante un paso de la integración existe un par (t, λ, λ_9) tal que $R|_{(\lambda,\lambda_9)} << R_{max}$ entonces la variación en la función perturbadora durante este paso es muy grande, luego, las perturbaciones por variables rápidas pueden ser demasiado importantes durante este paso y no son tomandas en cuenta por la R_{sec} calculada en dicho paso por lo cual no es válido el uso del método secular semi-analítico. El límite inferior elegido para este cociente está indicado por la línea punteada roja que indica el valor $R/R_{max} = 0.15$. Los puntos rojos corresponden a los valores que caen en la región en la que no vale el método.

In [4]:
```
plt.scatter(data[data['R/Rmax'] >= 0.15]["a"],data[data['R/Rmax'] >= 0.15]["R/Rmax"],color="blu
plt.scatter(data[data['R/Rmax'] < 0.15]["a"],data[data['R/Rmax'] < 0.15]["R/Rmax"],color="red",
plt.axhline(0.15,color="red",linestyle="--",label=r"$R/R_{max}=0.15$")
plt.xlabel("a[ua]")
plt.ylabel(r"$R/R_{max}$")
plt.legend(loc=1,bbox_to_anchor=(1.35,1))
plt.savefig(f"RRmax_a_{amin}_{amax}_{da}.png")
```



In [5]:
```
alims = [data[data['R/Rmax'] < 0.15].a.min(),data[data['R/Rmax'] < 0.15].a.max()] # se hallan l
alims
```

Out[5]:
```
[340.0, 600.0]
```

Notamos por un lado que el cociente R/R_{max} es decreciente desde $a = 200ua$ hasta $a = 500ua$ a partir de donde se vuelve creciente. Esto se debe al hecho que en las cercanías del P9 la perturbación de este último se vuelve demasiado importante y no se puede despreciar la evolución de variables rápidas.

Por otro lado, notamos que la dispersión en los valores del cociente como función del semi-eje se vuelve mínima entorno a los valores $a \in \{300, 740\}$ y se hace máxima en la vecindad del P9.

Con el criterio $R/R_{max} = 0.15$, tenemos que la región de validez del uso de la teoría secular semi-analítica es $\mathcal{V} = \{a : a \in [200, 340] \bigcup [600, 800]\}$

3.2. Control por estudio de la evolución de R_{sec}/R^0_{sec}

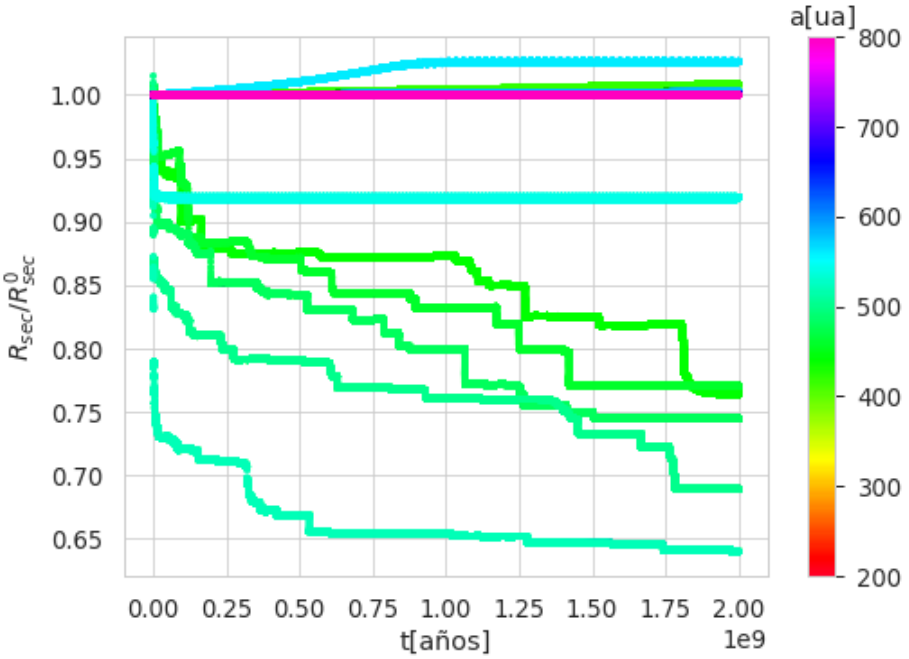
Es importante notar que este gráfico permite hallar en que situaciones lo calculado mediante la teoría secular semi-analítica se aleja demasiado de la realidad por omitir la evolución de variables rápidas. Ahora, para estudiar que tanto lo calculado se aleja de realida por errores de cómputo, es útil estudiar la evolución de R_{sec}/R^0_{sec} como función del tiempo, siendo R^0_{sec} la función perturbadora secular calculada a partir de las

condiciones iniciales. En efecto, los errores acumulativos de cómputo deberían inducir cambios en el valor de la R_{sec} asociada a cada semi-eje.

Es importante notar que se esperan diferencias importantes para los semi-ejes en los que no vale el uso de la teoría secular semi-analítica por culpa de la evolución de las variables rápidas y no de los errores de cómputo. Pero dichos casos deberían ser consistente con lo hallado en el estudio de la evolución de R/R_{max} .

A continuación se presentan las evoluciones de R_{sec}/R^0_{sec} como función del tiempo donde el mapa de colores indica el semi-eje asociado.

In [6]:
```
for i in np.arange(0,len(aall),1):
    a = aall[i]
    dataA = data[data["a"] == a].reset_index(drop=True)
    R0 = dataA["Rsec"].iloc[0]
    plt.scatter(dataA["T"],dataA["Rsec"]/R0,label=f"{a}",cmap=plt.cm.gist_rainbow,c=dataA["a"],
plt.xlabel("t[años]")
plt.ylabel(r"$R_{sec}/R_{sec}^0$")
cbar = plt.colorbar()
cbar.ax.set_title("a[ua]")
plt.savefig(f"R_t_{amin}_{amax}_{da}.png")
```



Notamos de forma general diferencias a la unidad importantes de parte de R_{sec}/R^0_{sec} para los valores de semi-ejes del entorno del P9, que concuerdan con los límites hallados anteriormente, por lo cual es claro que estas diferencias se deben a la evolución de variables rápidas. Luego, para los casos lejanos al P9 notamos que las diferencias crecen muy lentamente, lo debe ser dado a un error de cómputo acumulativo (esto se ve mejor en los gráficos de las regiones lejanas al P9, ver archivos "R_t_200_400_20.png" y "R_t_600_800_20.png"), pero estas diferencias son mínimas y despreciables. Podemos entonces concluir que se puede utilizar la teoría secular semi-analítica en la región $\mathcal{V}$ para hallar los modos forzados del P9.

4. Estudio de los modos forzados

En esta sección se estiman los modos forzados e_{for} del P9 para los distintos semi-ejes de la partícula considerada (no se buscan modos forzados en i porque asumimos caso plano). Para esto, recordemos que si vale la teoría secular entonces en el caso de un sólo planeta las trayectorias de las partículas consideradas en el plano (k,h) deberían ser cuasi circulares. Luego, e_{for} corresponde a la distancia del centro de dicha circunferencia al origen.

4.1. El método utilizado

Ahora, es importante notar que la posición de dicho centro no corresponde necesariamente a la posición media de los puntos de la trayectoria. En efecto si por ejemplo se integró durante un tiempo tal que la trayectoria de la particula en el plano (k, h) por corresponde a una vuelta completa y media entonces al tomar el promedio de las posiciones se obtendrá un punto un ligeramente corrido hacía el lado que tiene el doble de puntos que el otro. Por este motivo, para estimar el centro geométrico se diseñó una función que opera de la siguiente manera:

1. para cada punto $p_i = (x_i, y_i)$ se calcula su distancia a todos los demás puntos
2. se identifica el punto más alejado $p_{max} = (x_{max}, y_{max})$ de p_i
3. se calcula el punto $p_i^c = \frac{p_i + p_{max}}{2}$
4. se estima el centro como el promedio de los p_i^c Notemos que este método solamente es aplicable para una distribución cuasi circular de puntos.

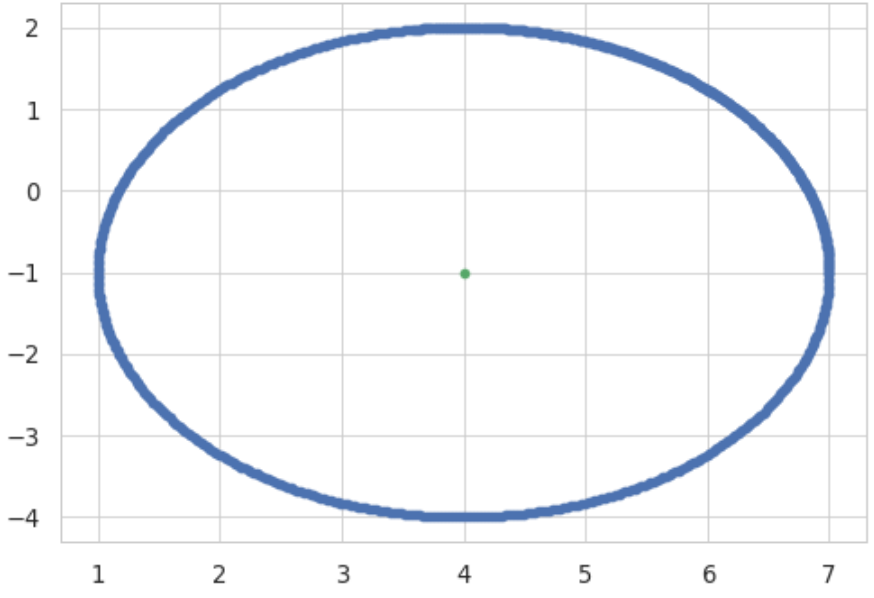
In [7]:
```
def center(circle):
    if circle.shape[0] > 100: # se usa una muestra aleatoria de 100 puntos por si hay demasiado
        circle = circle.sample(100)
    centerx = []
    centery =[]
    for i in np.arange(0,len(circle["x"]),1): # se calculan los pic de cada punto de la muestra
        p1 = circle.iloc[i]
        maxd = np.max(np.sqrt((p1[0]-circle.x)**2+(p1[1]-circle.y)**2))
        p2 = circle[np.sqrt((p1[0]-circle.x)**2+(p1[1]-circle.y)**2) == maxd].iloc[0]
        centerx += [(p1[0]+p2[0])/2]
        centery += [(p1[1]+p2[1])/2]
    centers = [centerx,centery]
    return [np.mean(centers[0]),np.mean(centers[1])] # se toma y devuelve el centro promedio
```

4.2. Breve validación del método

En esta sección se pone la función center() a prueba aplicandola a un conjunto de datos que describe una circunferencia centrada en p_0 con puntos que se solapan en ciertas regiones.

In [8]:
```
theta = np.arange(0,8,0.01) # rango de angulos de los datos ficticios
r = 3 # radio
p0 = [4,-1] # centro
circle = pd.DataFrame({"x": p0[0]+r*np.cos(theta),"y": p0[1]+r*np.sin(theta)}) # se contruyen l
c = center(circle) # se aplica center() a los datos sintéticos
plt.scatter(circle["x"],circle["y"]) # plot de control
plt.scatter(c[0],c[1])
```

Out[8]:
```
<matplotlib.collections.PathCollection at 0x7f40a3e11820>
```



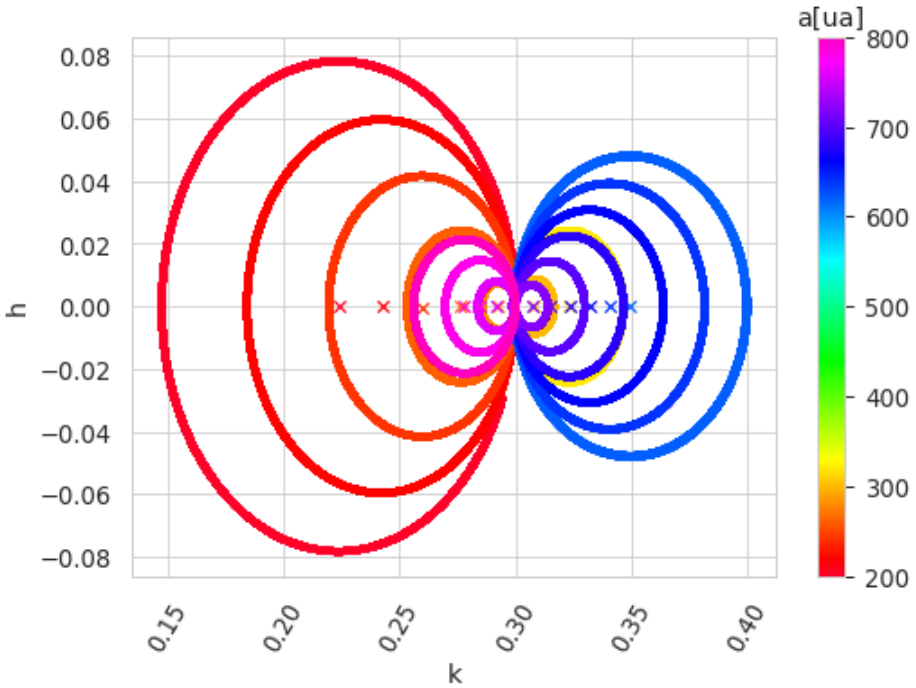
Notamos que el método logra obtener correctamente el centro geométrico de la distribución.

4.3. Aplicación a los datos reales

En esta sección se aplica el método utilizado a los datos reales. Se grafica el espacio (k, h) junto con los centros calculados (cruces). El mapa de colores indica el semi-eje.

In [9]:
```
cleaned = data[(data.a < alims[0]) | (data.a > alims[1])] # se filtran los datos con R/Rmax < 0
```

In [10]:
```
hkfor = []
efor = []

for a in cleaned.a.unique():
    dataA = cleaned[cleaned["a"] == a].reset_index(drop=True)

    k = dataA.e*np.cos(dataA.lper*np.pi/180)
    h = dataA.e*np.sin(dataA.lper*np.pi/180)

    circlekh = pd.DataFrame({'x':k,'y':h})
    centerskh = center(circlekh)
    hkfor += [centerskh]
    efor += [np.sqrt(centerskh[0]**2+centerskh[1]**2)]

    plt.scatter(k,h,cmap=plt.cm.gist_rainbow,c=dataA.a,vmin=data.a.min(),vmax=data.a.max(),s=10
    plt.scatter(centerskh[0],centerskh[1],cmap=plt.cm.gist_rainbow,c=dataA.a.iloc[0],vmin=data.
    plt.xlabel('k')
    plt.ylabel('h')
    plt.xticks(rotation = 60)

cbar = plt.colorbar()
cbar.ax.set_title("a[ua]")
plt.savefig(f'hkfor_{amin}_{amax}_{da}.png')
plt.show()
```



Notamos que las trayectorias de las partículas en estos espacios son cuasi-circulares para los casos de $a \in \mathcal{V}$, lo cual valida el análisis hecho para determinar $\mathcal{V}$. Notamos que en los casos cercanos a $a \sim 300ua$ y $a \sim 740ua$ que las trayectorias tienden a un punto. Ahora, recordemos que cuando estas trayectorias son circulares, se tiene que e es cuasi constante, luego tenemos cerca de $a = 300ua$ y $a = 740$ que ϖ es también cuasi constante, por lo cual deben existir cerca de estos puntos puntos de equilibrio. Esto se aprecian mejor en los diagramas de las regiones lejanas a P9 (ver archivos "hkfor_200_400_20.png" y "hkfor_600_800_20.png").

Es además interesante notar la asimetría en las curvas entre los casos más lejanos internos y externos. Es razonable pensar que esta asimetría se deba al hecho que es más fácil para un objeto orbitando una estrella

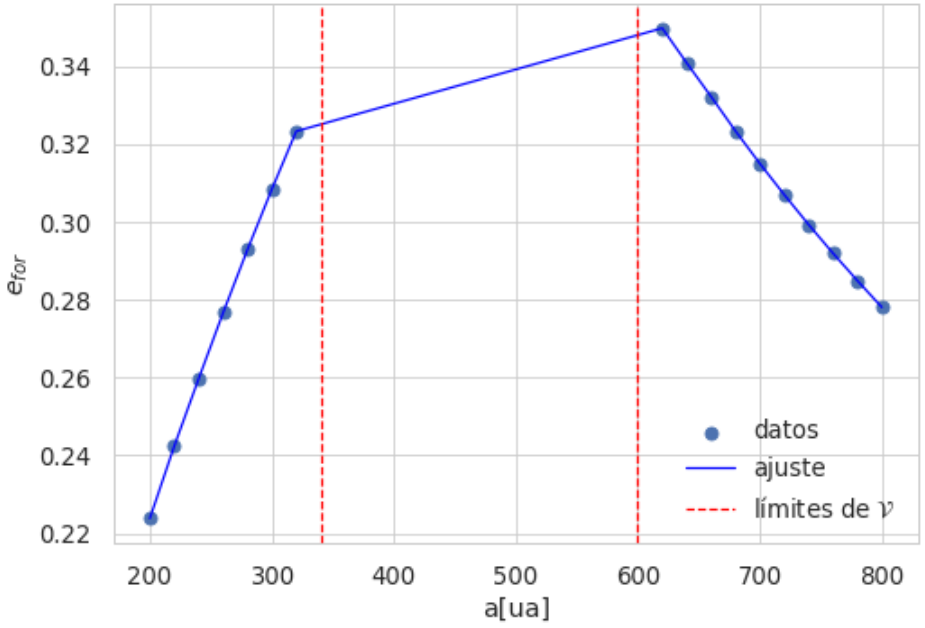
sentir los efectos de la perturbación de un perturbador externo que de uno interno en el caso que dicho objeto y su perturbador estén alejados, lo cual explicaría porque se notan modos libres mucho más importantes en el caso interno lejano que externo lejano.

Sabiendo que las trayectorias en el plano (k, h) son en efecto cuasi circulares, podemos calcular fácilmente los modos forzados.

4.4. Cálculo de los modos forzados

A partir de las posiciones de los centros de las trayectorias en el plano (k, h) se calculan los modos forzados como la distancia de dichos puntos al origen de sus planos respectivos. A continuación se grafican los modos forzados como función del semi-eje.

In [16]:
```
e_a_interp = interpolate.interp1d(cleaned.a.unique(),efor)
plt.scatter(cleaned.a.unique(),efor,s=50,label='datos')
plt.plot(cleaned.a.unique(),e_a_interp(cleaned.a.unique()),color='blue',label='ajuste')
plt.xlabel('a[ua]')
plt.ylabel(r'$e_{for}$')
plt.axvline(alims[0],color="red",linestyle='--',label= r'límites de $\mathcal{V}$')
plt.axvline(alims[1],color="red",linestyle='--')
plt.legend()
plt.savefig(f'efor_{amin}_{amax}_{da}.png')
```



Notamos que los modos forzados crecen cuando nos acercamos al semi-eje del P9, lo cual tiene sentido ya que más cerca nos encontramos del P9, más su perturbación domina. De forma general, encontramos además que estos modos son más importantes en la región externa de $\mathcal{V}$ que en la interna.

In [ ]:
```
```

Calculador de las evoluciones seculares

Este código tiene como fin calcular la evolución secular de un conjunto de particulas de distintos semi-ejes (listados en "aall") con excentricidad $e = 0.3$, inclinación $i = 0^o$ y longitud del perihelio $\varpi = 0^o$ utilizando el programa "Evosecular", bajo la perturbación de un planeta 9 de masa $M_9 = 5M_\oplus$, semi-eje $a_9 = 500ua$, excentricidad $e_9 = 0.4$, inclinación $i_9 = 0^o$ y argumento del perihelio $\varpi = 0^o$.

Importación de paquetes y librerías

En esta sección se importan los paquetes y librerías necesarias.

In [1]:
```
import numpy as np # numpy
import pandas as pd # pandas
from astropy import constants # de acá se calcula se obtiene G
import astropy.units as u # de acá se cambia G a unidades amigables
import warnings # esto es para sacar warnings molestos
import os # esto es para ejecutar líneas de comandos de la terminal (para ejecutar el evosecula
```

In [2]:
```
warnings.filterwarnings("ignore")
```

Constantes y variables

En esta sección se definen las constantes y variables necesarias a la ejecución de Evosecular. Concretamente se definen (todo en unidades de ua, años y $M_\odot$):

1. El rango de semi-ejes que se quiere explorar (aall)
2. La constante de proporcionalidad de la tercera ley de Kepler $C = \frac{G}{4\pi}$ mediante el uso de la librería Astropy (C)
3. El tiempo total de integración que se desea (T)

In [3]:
```
amin = 400
amax = 600
da = 20
aall = np.arange(amin,amax+da,da) # semi-ejes
C = constants.G.to('au3/(Msun*yr2)')*u.Msun/(4*np.pi)
print(C)
print(C.value)
C = C.value # constante Kepler
T = 2*10**9 # tiempo total de integración
```

```
3.141473988025521 AU3 / yr2
3.141473988025521
```

Cálculo de la evolución secular

En esta sección se calcula evolución secular para cada partícula. Para esto se modifica el archivo de entrada "evosecular.ent", cambiando en cada iteración:

1. El valor del semi-eje
2. El valor del paso de integración calculado como el entero más cercano a 2 veces el período orbital de la partícula asumiendo una órbita Kepleriana a partir de sus condiciones iniciales, esto es el entero más cercano a $\Delta t = 2\sqrt{\frac{a^3}{C}}$
3. El número de pasos como el entero más cercano a $\frac{T}{\Delta t}$

In [4]:
```
for i in np.arange(0,len(aall),1):
    a = float(aall[i])
    paso = float(round(2*np.sqrt(a**3/C)))
```

```
    Npasos = round(T/paso)
    with open("evosecular.ent", "r+") as f:
         old = f.read() # se lee el archivo de entrada

    old = old.split('\n')
    old[1] = str(a) + '    0.3  0.1     0.0    0.0' # se modifica el valor del semi-eje
    old[7] = str(paso) + '     ' + str(Npasos) # se modifica el paso y el número de pasos
    new = '\n'.join(old)
    with open('evosecular.ent', 'w+') as f:
        f.write(new) # se sobre-escribe el archivo de entrada

    os.system("chmod +x evosecular29ene22") # se da el derecho de ejecución del programa
    os.system("./evosecular29ene22") # se ejecuta el programa (CUIDADO!!! probablemente distint
    print(i) # se imprime el paso a modo de control (este paso es opcional)
```

```
0
1
2
3
4
5
6
7
8
9
10
```

Adaptación del archivo de salida

En esta sección se modifica el archivo de salida "evosecular.sal" para que tenga un formato más adaptado a la librería Pandas de Python y se guarda en un nuevo archivo "evosecularout.sal".

In [5]:
```
import re
with open ('evosecular.sal', 'r') as i_f, open (f'evosecularout_{amin}_{amax}_{da}.sal', 'w') a
    for line in i_f:
        o_f.write(re.sub('[\t ]+',' ', line))
with open (f'evosecularout_{amin}_{amax}_{da}.sal', 'r+') as i_f:
    i_f = i_f.read()
    i_f = i_f.replace(",", " ")
with open (f'evosecularout_{amin}_{amax}_{da}.sal', 'w+') as o_f:
    o_f.write(i_f)
```