

Predicción de Eclipses Lunares y Solares

Martín Gutiérrez Donoso,¹ Vicente Honorato Rodríguez¹ and Constanza Soto Suárez¹

Departamento de Física, Universidad Técnica Federico Santa María, Av. Vicuña Mackenna 3939, Santiago, Chile.¹

Received May 2, 2024

ABSTRACT

En este trabajo se utiliza la librería de Python llamada Rebound para simular el movimiento de la Luna, la Tierra y el Sol con el fin de poder predecir futuros eclipses. Utilizando un método matemático simple donde se toman vectores entre los cuerpos y usando el teorema del coseno se define cuando los vectores son totalmente paralelos, es decir, en este contexto, hay un eclipse total. Con este código y el método utilizado se pueden predecir eclipses con una alta precisión y, con la suficiente capacidad computacional, se puede emular el movimiento por la cantidad de tiempo que se deseé. De esta forma, los próximos eclipses encontrados fueron: **Eclipses Lunares:** 14/03/2025 (*Lunar Total*), 03/03/2026 (*Lunar Total*), 28/08/2026 (*Lunar Parcial*); **Eclipses Solares:** 02/10/2024 (*Solar Anular*), 06/02/2027 (*Solar Anular*), 02/08/2027 (*Solar Total*), 26/01/2028 (*Solar Anular*).

Key words. Órbitas – Eclipses: Totales, Parciales, Anulares – Rebound

1. Introducción

Los eclipses consisten en fenómenos astronómicos en donde se ve obstruida la luz de un cuerpo celeste por el paso de un cuerpo "*eclipsante*". Por ejemplo, cuando la Luna se posiciona entre el Sol y la Tierra, es posible apreciar un eclipse solar. De manera análoga, si la Tierra se posiciona entre el Sol y la Luna, es posible apreciar un eclipse lunar.

Desde la Tierra, el Sol sigue una trayectoria aparente denominada elíptica. Esta forma un ángulo de inclinación de 5.2° con respecto al plano orbital de la Luna, por lo que no ocurre un eclipse Solar cada vez que la Luna orbita entre el Sol y la Tierra, ya que estos deben estar perfectamente alineados, al igual que para un eclipse lunar. El análisis geométrico de esta situación se hablará en la Sección 2.2.

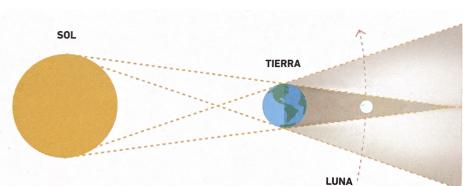


Fig. 1: Esquema visual de un eclipse lunar, donde la región sombreada clara es la penumbra y la región sombreada oscura es la umbra (Maza [1]).

Cuando ocurre un eclipse lunar, la Tierra crea un cono de sombra, como se aprecia en Fig. 1. Este se compone de una zona donde la Tierra eclipsa totalmente el sol, denominada *umbra*, y otra donde el Sol es eclipsado parcialmente, denominada *penumbra*.

Cuando la Luna transita a través del cono de sombra, y logra ser cubierta en su totalidad por la umbra, ocurrirá *eclipse total de Luna*. Si es cubierta parcialmente por la umbra, se generará un *eclipse parcial de Luna*. Y, si es cubierto únicamente por la penumbra en toda su trayectoria, se producirá un *eclipse penumbral de Luna*.

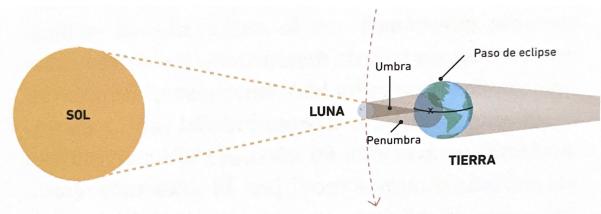


Fig. 2: Esquema visual de un eclipse solar, donde la región sombreada clara es la penumbra y la región sombreada oscura es el cono de la umbra (Maza [1]).

De manera similar, cuando ocurre un eclipse solar, la Luna también crea un cono de sombra, pero este impacta en la Tierra como se muestra en la Fig. 2. Este, al igual que en el eclipse lunar, está compuesto por la umbra y la penumbra. En el instante en que el Sol, la Luna y la Tierra se alinean, y el cono de la umbra toca la Tierra, se produce un *eclipse total de Sol*. Sin embargo, si el cono de la umbral se cierra antes de tocar la superficie de la Tierra, se genera un *eclipse anular de Sol*.

Ya que en nuestro análisis solo se enfoca en las órbitas respectivas del Sol, la Tierra y la Luna, los eclipses a predecir pueden ser lunar total o parcial, y solar total o anular.

2. Metodología

2.1. Rebound

Rebound es una librería de Python de libre acceso que tiene como enfoque simular la dinámica de colisiones, pero también sirve para resolver el problema clásico de N-Cuerpos. Dada su facilidad de uso y personalización, es muy útil para estudios de la física y la astrofísica.

El funcionamiento más a fondo de Rebound se encuentra explicado en (Rein & Liu, 2012 [3]), pero de manera general, para simular el efecto gravitatorio entre los cuerpos, calcula la aceleración entre cada partícula. Para una partícula i -esima se utiliza la Eq. 1, donde G es la constante de gravitación universal, m_j es la masa de la partícula j y r_{ji} es la distancia relativa entre las partículas j e i . El parámetro de ablandamiento gravitacional b es cero por defecto, pero se puede establecer como un valor finito para simulaciones en las que las colisiones físicas entre partículas no se resuelven y los encuentros cercanos pueden provocar grandes aceleraciones anti-físicas. La variable N_{active} especifica el número de partículas masivas en la simulación. Las partículas con un índice igual o mayor que N_{active} son tratadas como partículas de prueba. De manera predeterminada, se asume que todas las partículas tienen masa y contribuyen a la suma en Eq. 1 (Texto extraído de [3]).

$$a_i = \sum_{j=0}^{N_{active}-1} \frac{Gm_j}{(r_{ji}^2 + b^2)^{3/2}} \hat{r}_{ji} \quad (1)$$

2.2. Matemática Aplicada

En términos matemáticos existe una relación entre los lados de un triángulo (como se muestra en Fig. 3) y el coseno de uno de sus ángulos internos, conocida como el Teorema del coseno. Este teorema se puede expresar como muestra la Eq. 2, donde a, b y c son los lados del triángulo y " γ " es el ángulo opuesto a c .

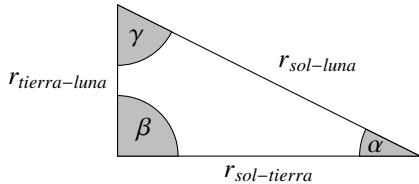


Fig. 3: Representación gráfica del triángulo formado en el sistema Sol-Tierra-Luna, utilizada en el teorema del coseno para la predicción de eclipses solares y lunares. Si $\beta = 180$ hablamos de eclipses lunares, si $\gamma = 180$ de eclipses solares.

$$c = \sqrt{a^2 + b^2 - 2ab \cos(\gamma)} \quad (2)$$

Al modificar la Eq. 2 utilizando los parámetros de distancia del sistema Sol-Tierra-Luna, se obtiene la distancia entre el Sol y la Luna, como muestra la Eq. 3.

$$r_{s-l}^2 = r_{s-t}^2 + r_{t-l}^2 - (2 \cdot r_{s-t} \cdot r_{t-l} \cdot \cos(\beta)) \quad (3)$$

Los sub-índices significan:
 $s - t$: Distancia del Sol a la Tierra
 $t - l$: Distancia de la Tierra a la Luna.

$s - t$: Distancia del Sol a la Tierra
 $t - l$: Distancia de la Tierra a la Luna.

De esta forma, despejando el coseno, la Eq. 3 queda como se muestra en Eq. 4.

$$\cos(\beta) = \frac{r_{s-t}^2 + r_{t-l}^2 - r_{s-l}^2}{2 \cdot r_{s-t} \cdot r_{t-l}} \quad (4)$$

Con esto se pueden ver los casos cuando el coseno sea igual a -1 , lo que implicaría ver cuando el ángulo sea de 180 grados, lo que indicaría estar en presencia de un eclipse Lunar. De la misma forma, se puede trabajar con el vector distancia entre el Sol y la Tierra, como se hizo en Eq. 3, obteniendo así la Eq. 5.

$$r_{s-t}^2 = r_{s-l}^2 + r_{t-l}^2 - (2 \cdot r_{s-l} \cdot r_{t-l} \cdot \cos(\gamma)) \quad (5)$$

Haciendo el mismo despeje realizado para obtener obtener la Eq. 4, finalmente obtenemos la expresión mostrada en Eq. 6.

$$\cos(\gamma) = \frac{r_{s-l}^2 + r_{t-l}^2 - r_{s-t}^2}{2 \cdot r_{s-l} \cdot r_{t-l}} \quad (6)$$

De manera análoga a lo explicado con la Eq. 4, si el coseno es igual a -1 entonces el ángulo es de 180 grados, lo que en este caso significa un eclipse Solar.

2.3. Sistema Sol-Tierra-Luna

Para comprender el sistema, posicionamos el Sol al centro de la simulación (tomando posiciones x, y, z). Si bien, este también orbita en torno a un centro de masa, al ser considerablemente más masivo que la Tierra, decidimos que es una buena aproximación tomar el Sol como centro de masa. Esto no ocurriría si agregamos más planetas masivos, como Júpiter, ya que alterarían significativamente la órbita del Sol.

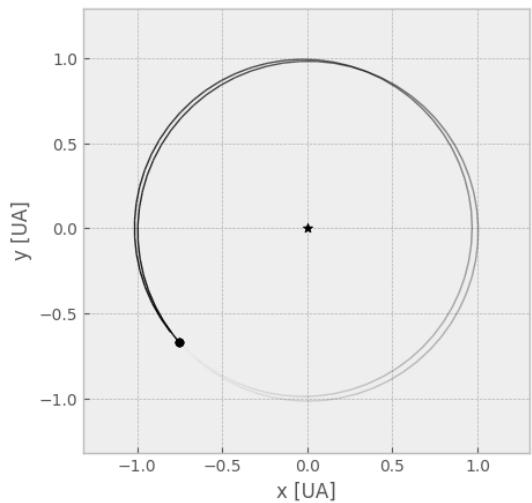


Fig. 4: Gráfica ilustrativa de la simulación para un sistema de 3 cuerpos con el Sol al centro, la Tierra y la Luna orbitando.

En Fig. 4 se logra diferenciar la órbita terrestre de la lunar de manera muy leve, esto se debe principalmente a que la distancia Sol-Tierra es considerablemente mayor a la distancia Tierra-Luna.

3. Predicción de Eclipses

En primer lugar, se define un vector temporal que va desde $t = 0$ (actualidad) hasta $t = 4$ (cuatro años a futuro), en la simulación. Con lo que se pueden obtener gran parte de las posiciones de las órbitas de cada cuerpo, en distintos tiempos. Como ejemplo se tiene la Fig. 5, donde para un tiempo arbitrario sin eclipse, se tiene un triángulo en las tres dimensiones, formadas por las posiciones de cada cuerpo.

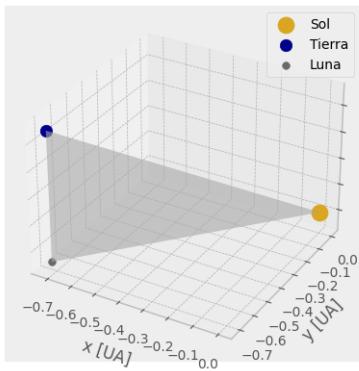


Fig. 5: Representación gráfica del triángulo formado entre Sol, Tierra y Luna cuando no hay ningún tipo de eclipse, obtenida de la data entregada por la simulación "Rebound", tras integrar el tiempo. Similiar al de Fig. 3.

De esta misma Fig. 5, podemos hacer la comparación con Fig. 3, donde los ángulos α , β , γ , corresponden a los ángulos que se encuentran en los vértices correspondientes al Sol, Tierra y Luna, respectivamente.

Esta representación es bastante útil para hacerse una idea de cómo se aplica el teorema del coseno en la predicción de eclipses solares y lunares, mediante las Eq. 4 y Eq. 6, dado que, como se ha comentado anteriormente, con valores para $\cos(\beta) = -1$ se tienen eclipses lunares y con valores para $\cos(\gamma) = -1$ eclipses solares.

Con la condición planteada anteriormente de los ángulos, se aplica este Teorema para cada tiempo, generando múltiples sistemas como el de Fig. 5, pero con distintas distribuciones. Sin embargo, no hay resultados con valores exactos de -1 , por lo que se establece una condición para tener fechas más precisas en la predicción, la cual consiste en que $-1.0001 < \cos(\beta) < -0.9999$ para eclipses lunares y $-1.0001 < \cos(\gamma) < -0.9999$ para eclipses solares.

Las fechas obtenidas a partir de este método se encuentran informadas en la Tabla 1 y en la Tabla 2, cuyos valores para el tiempo tuvieron que convertirse a fechas, ya que el vector de tiempo viene dado desde cero, hasta cuatro en unidades de año.

En específico, de Tabla 1, se tienen los eclipses lunares que se aproximan hasta el 2026, cuya clasificación fue obtenida de (Espacio Profundo [2]). La razón de haber obtenido distintos tipos de eclipses puede deberse, principalmente, a dos aspectos importantes de considerar. Primero, que no se considera la distancia entre los vectores para asegurar que la luna pase por la umbra, para tener solamente eclipses lunares totales. Segundo,

falta de precisión en la condición planteada para el coseno de β .

Fecha	Tipo de Eclipse
14/03/2025	Lunar Total
07/09/2025	Lunar Total
03/03/2026	Lunar Total
28/08/2026	Lunar Parcial

Table 1: Tabla con próximos eclipses Lunares, según su tipo. Valores obtenidos con el teorema del coseno. $\cos(\beta) \approx -1$, representado en Fig. 3. Todas las fechas fueron comparadas con (Espacio Profundo [2]), lo que permitió el conocimiento del tipo.

De manera similar se obtiene Tabla 2, donde se observan tres eclipses solares anulares y uno total, cuya clasificación puede deberse a las mismas dos razones planteadas anteriormente, pero para el ángulo γ .

Fecha	Tipo de Eclipse
02/10/2024	Solar Anular
06/02/2027	Solar Anular
02/08/2027	Solar Total
26/01/2028	Solar Anular

Table 2: Tabla con próximos eclipses Solares, según su tipo. Valores obtenidos con el teorema del coseno. $\cos(\gamma) \approx -1$, representado en Fig. 3. Todas las fechas fueron comparadas con (Espacio Profundo [2]), lo que permitió el conocimiento del tipo.

Para una mejor comprensión de los resultados, se tiene la Fig. 6, que muestra la variación del coseno de β (recordando que es el ángulo subtendido entre la distancia sol-tierra y distancia tierra-luna), en función del transcurso del tiempo (en fechas), señalando en línea roja el siguiente eclipse lunar que ocurrirá en el año 2025. También se aprecia en la región inferior de la gráfica que, según la condición utilizada para el ángulo, ese valle en el coseno es el correcto para informar el eclipse, mas no los demás. Este análisis se extiende para las siguientes fechas.

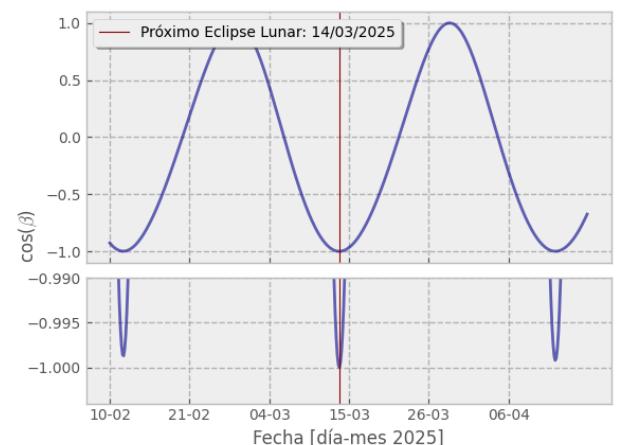


Fig. 6: Gráfica sobre el coseno del ángulo subtendido entre la distancia sol-tierra y distancia tierra-luna ($\cos(\beta)$) en función del tiempo en formato de fechas.

En Fig. 7 se tiene el mismo análisis realizado anteriormente, pero aplicado a eclipses solares según el ángulo correspondiente, indicando que el siguiente eclipse solar anular ocurriría en el presente año. Además, se observan que los valles en el coseno son más cercanos a -1 , que en Fig. 6, sin embargo, no fueron considerados por la condición aplicada.

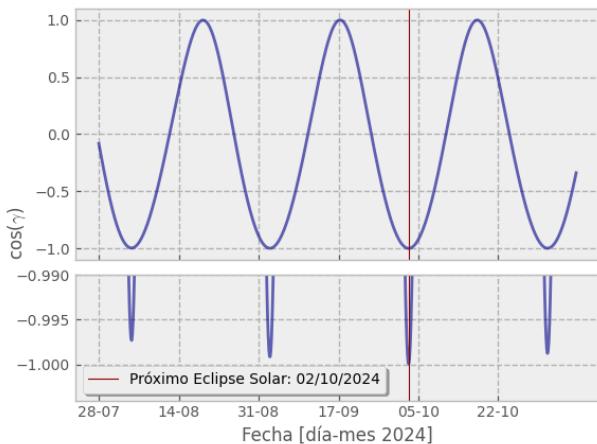


Fig. 7: Gráfica sobre el coseno del ángulo subtendido entre la distancia sol-luna y distancia tierra-luna ($\cos(\gamma)$) en función del tiempo en formato de fechas.

4. Conclusión

Simulamos el sistema Sol-Tierra-Luna, utilizando la librería de Python *Rebound*, con la finalidad de determinar las fechas en que ocurrirán eclipses lunares y solares. Esto fue posible mediante el desarrollo matemático mostrado en la Sección 2.2. Obtuvo cuatro fechas para eclipses lunares, entre los años 2025 y 2026, donde tres son eclipses totales y uno parcial (Tabla 1), y cuatro fechas para eclipses solares, entre los años 2024 y 2028, donde tres son anulares y uno total (2). No realizamos una clasificación propia de eclipses (si corresponde a anular, total o parcial) ya que, con el estudio realizado, no se determina la forma del cono de sombra. Detectar si la trayectoria de la Luna pasa por la umbra o la penumbra para los eclipses lunares, y si el cono de umbra se cierra antes de tocar la tierra para los eclipses solares, mejoraría la precisión de los resultados.

Los valores de las condiciones definidas para identificar eclipses, correspondiente al valor de $\cos(\beta)$ para los lunares, $\cos(\gamma)$ para los solares y la resolución (Sección 4), fueron asignados para que los resultados sean de la forma día-mes-año. Si considerábamos una resolución mayor o utilizábamos un rango más grande para el valor de los cosenos, el programa entregaría fechas repetidas, ya que tomaría en cuenta la hora y minuto del día en el que ocurre un eclipse. Por otro lado, esto también ocasionó que no se identificaran un eclipse parcial de luna, ni los eclipses parciales de sol.

References

- [1] J. Maza. *Eclipses*. Fuerza de colección. Grupo Planeta - Chile, 2019. ISBN: 9789563605839. URL: <https://books.google.cl/books?id=li6PDwAAQBAJ>.
- [2] Espacio Profundo. *Calendario de Eclipses Solares y Lunares*. 2019. URL: https://www.espacioprofundo.com/eclipses.html/calendario_de_eclipses_solares_y_lunares/eclipse-solar-anular-del-2-de-octubre-de-2024/.
- [3] H. Rein and S. F. Liu. “REBOUND: an open-source multi-purpose N-body code for collisional dynamics”. In: *A&A* 537, A128 (Jan. 2012), A128. doi: [10.1051/0004-6361/201118085](https://doi.org/10.1051/0004-6361/201118085). arXiv: [1110.4876 \[astro-ph.EP\]](https://arxiv.org/abs/1110.4876).

Anexo

Código para la simulación

Como se menciona previamente, para trabajar con Rebound se utiliza el lenguaje computacional Python, en este caso gracias a la interfaz proporcionada por Google Colab.

Listings

1	Importando librerías necesarias	4
2	Inicio de simulación	4
3	Creacion de imagen	5
4	Propiedades de los cuerpos	5
5	Iteraciones para obtener posiciones	5
6	Vectores de cada distancia	5
7	Teorema del coseno para Beta	5
8	Teorema del coseno para Gamma	6
9	Fechas de los eclipses Solares	6
10	Fechas de los eclipses Lunares	6
11	Gráfica de triángulo entre los cuerpos	6
12	Lista vacía para agregar fechas	6
13	Gráfico para eclipses Solares	6
14	Gráfico para eclipses Lunares	7

#librerias a utilizar

```
!pip install rebound
#activar esta linea de codigo si no se tiene
#instalada la libreria rebound en su propio
#computador

import rebound
#libreria de la simulacion

import numpy as np
#libreria para el trabajo matematico

import matplotlib.pyplot as plt
#libreria para las graficas

from mpl_toolkits.mplot3d.art3d import
    Poly3DCollection
#libreria para generar superficies en 3d (de
    matplotlib)

from matplotlib import gridspec
#para el grid de las graficas

plt.style.use("bmh")
```

Listing 1: Importando librerías necesarias

```
sim = rebound.Simulation() #Para iniciar la
                            simulacion

sim.G = 4*np.pi**2 #valor que le daremos a la
                    constante cosmologica G
```

```

sim.units = ('AU', 'yr', 'Msun') #unidades de
    medidas que queremos en la simulacion
    (Unidades Astronomicas para distancia,
    anos para el tiempo, Masas solares para la
    masa)

sim.add('Sun') #Nombre que recibe el sol en
    NAIFF (es de la NASA) para agregarlo a la
    simulacion
sim.add('399') #Nombre que recibe la tierra en
    NAIFF (es de la NASA) para agregarla a la
    simulacion
sim.add('301') #Nombre que recibe la luna en
    NAIFF (es de la NASA) para agregarla a la
    simulacion

sim.move_to_com() #No considera fuerzas
    externas y mueve el centro al frame de
    momento

```

Listing 2: Inicio de simulación

```

fig = rebound.OrbitPlot(sim) #figura
    ilustrativa del sistema generado (segun
    los tres cuerpos; Sol, Tierra, Luna)
plt.xlabel('x [UA]')
plt.ylabel('y [UA]')
plt.show()

```

Listing 3: Creacion de imagen

Con Listing 3 se genera la Fig. 4

```

sun = sim.particles[0] #propiedades del sol
earth = sim.particles[1] #propiedades del
    planeta tierra
moon = sim.particles[2] #propiedades de la luna

resolution = 10000 #resolucion que indicaria
    fisicamente la cantidad de intervalos de
    tiempo de la simulacion
time = np.linspace(0, 4, resolution) #tiempo
    que se usara en la simulacion
x_pos = np.zeros((3, resolution)) #Array de
    vectores en posicion x ( [sol, tierra,
    luna], len(resolution) )
y_pos = np.zeros((3, resolution)) #Array de
    vectores en posicion y ( [sol, tierra,
    luna], len(resolution) )
z_pos = np.zeros((3, resolution)) #Array de
    vectores en posicion z ( [sol, tierra,
    luna], len(resolution) )

```

Listing 4: Propiedades de los cuerpos

```

#iteraciones para agregar las posiciones de
    los cuerpos, segun el tiempo (time =
    np.linspace)
for i,t in enumerate(time):
    sim.integrate(t) #agregamos el tiempo a la
        simulacion
    x_pos[0][i] = sun.x #array de posiciones en
        eje x del sol en orden de las resolucion
        elegida
    y_pos[0][i] = sun.y #array de posiciones en
        eje y del sol en orden de las resolucion
        elegida

```

```

z_pos[0][i] = sun.z #array de posiciones en
    eje z del sol en orden de las resolucion
    elegida

x_pos[1][i] = earth.x #array de posiciones
    en eje x de la tierra en orden de las
    resolucion elegida
y_pos[1][i] = earth.y #array de posiciones
    en eje y de la tierra en orden de las
    resolucion elegida
z_pos[1][i] = earth.z #array de posiciones
    en eje z de la tierra en orden de las
    resolucion elegida

x_pos[2][i] = moon.x #array de posiciones en
    eje x de la luna en orden de las
    resolucion elegida
y_pos[2][i] = moon.y #array de posiciones en
    eje y de la luna en orden de las
    resolucion elegida
z_pos[2][i] = moon.z #array de posiciones en
    eje z de la luna en orden de las
    resolucion elegida

```

Listing 5: Iteraciones para obtener posiciones

```

#define los vectores de cada distancia
pos_sol = np.array([x_pos[0], y_pos[0],
    z_pos[0]]) #vector de posiciones del sol
    en los distintos tiempos
pos_tierra = np.array([x_pos[1], y_pos[1],
    z_pos[1]]) #vector de posiciones de la
    tierra en los distintos tiempos
pos_luna = np.array([x_pos[2], y_pos[2],
    z_pos[2]]) #vector de posiciones de la
    luna en los distintos tiempos

dist_sol_tierra_vec = pos_tienda - pos_sol
    #vector distancia entre tierra y sol
dist_tienda_luna_vec = pos_luna - pos_tienda
    #vector distancia entre luna y tierra
dist_sol_luna_vec = pos_sol - pos_luna #vector
    distancia entre sol y luna

dist_sol_tienda = np.sqrt(
    dist_sol_tienda_vec[0]**2 +
    dist_sol_tienda_vec[1]**2 +
    dist_sol_tienda_vec[2]**2 ) #modulo de
    distancia entre sol tierra
dist_tienda_luna = np.sqrt(
    dist_tienda_luna_vec[0]**2 +
    dist_tienda_luna_vec[1]**2 +
    dist_tienda_luna_vec[2]**2 ) #modulo de
    distancia entre tierra luna
dist_sol_luna = np.sqrt(
    dist_sol_luna_vec[0]**2 +
    dist_sol_luna_vec[1]**2 +
    dist_sol_luna_vec[2]**2 ) #modulo de
    distancia entre sol luna

```

Listing 6: Vectores de cada distancia

```

#aplicamos teorema del coseno con el
    "coseno(beta)" despejado, para encontrar
    eclipses lunares:
cos_b = ( (dist_sol_tienda**2 +
    dist_tienda_luna**2 - dist_sol_luna**2) /
    (2 * dist_sol_tienda * dist_tienda_luna) )

```

```
#cos(beta) ; beta = angulo opuesto a la
distancia sol-luna
eclipses_lunares = np.where( ((cos_b <
-0.9999) & (cos_b > -1.0001)) )[0]
#posiciones en el array en donde hay
eclipses lunares (se eligio ese rango
porque no existe un valor de -1 exacto)
```

Listing 7: Teorema del coseno para Beta

```
#aplicamos teorema del coseno con el
"coseno(gamma)" despejado, para encontrar
eclipses solares:
cos_y = ( (dist_sol_luna**2 +
dist_tierra_luna**2 - dist_sol_tierra**2)
/ (2 * dist_sol_luna * dist_tierra_luna) )
#cos(gamma) ; gamma = angulo opuesto a la
distancia sol-tierra
eclipses_solares = np.where( ((cos_y <
-0.9999) & (cos_y > -1.0001)) )[0]
#posiciones en el array en donde hay
eclipses solares (se eligio ese rango
porque no existe un valor de -1 exacto)
```

Listing 8: Teorema del coseno para Gamma

```
import datetime #libreria para transformar de
ano a fechas
tiempos_eclipses_solares =
time[eclipses_solares] #tiempos en donde
hay eclipses solares

start_date = datetime.datetime.today() #
Fecha actual
fechas_eclipses_solares = [start_date +
datetime.timedelta(days=t * 365.25) for t
in tiempos_eclipses_solares]
#transformacion del tiempo a fechas

print("Fechas de eclipses solares:
(Dia-Mes-Ano)")
for date in fechas_eclipses_solares:
print(date.strftime("%d-%m-%Y"))
```

Listing 9: Fechas de los eclipses Solares

```
tiempos_eclipses_lunares =
time[eclipses_lunares] #tiempos en donde
hay eclipses lunares

start_date = datetime.datetime.today() #
Fecha actual
fechas_eclipses_lunares = [start_date +
datetime.timedelta(days=t * 365.25) for t
in tiempos_eclipses_lunares]
#transformacion del tiempo a fechas

print("Fechas de eclipse lunares:
(Dia-Mes-Ano)")
for date in fechas_eclipses_lunares:
print(date.strftime("%d-%m-%Y"))
```

Listing 10: Fechas de los eclipses Lunares

```
fig = plt.figure() #para graficar en 3d los
vertices del triangulo que se formaria
entre los cuerpos (EJEMPLO DE TRIANGULO
PARA UTILIZAR EL TEOREMA DEL COSENO!!!)
(sin eclipses)
ax1= fig.add_subplot(projection='3d')
ax1.scatter(x_pos[0][22], y_pos[0][22],
z_pos[0][22], label = 'Sol', s = 150,
color = 'goldenrod') #punto del sol
ax1.scatter(x_pos[1][22], y_pos[1][22],
z_pos[1][22], label = 'Tierra', s = 80,
color = 'darkblue') #punto de la tierra
ax1.scatter(x_pos[2][22], y_pos[2][22],
z_pos[2][22], label = 'Luna', s = 30,
color = 'dimgray') #punto de la luna
plt.xlabel('x [UA]') #nombre en el eje x de la
grafica
plt.ylabel('y [UA]') #nombre en el eje y de la
grafica

verts = [(x_pos[0][22], y_pos[0][22],
z_pos[0][22]), (x_pos[1][22],
y_pos[1][22], z_pos[1][22]),
(x_pos[2][22], y_pos[2][22],
z_pos[2][22])] #vertices para utilizarla
luego

srf = Poly3DCollection(verts, alpha=0.6,
facecolor='darkgray') #para crear region
coloreada

plt.gca().add_collection3d(srf) #para que
aparezca la region coloreada
plt.legend() #para que aparezca la leyenda en
el grafico
plt.show() #para mostrar la grafica
```

Listing 11: Gráfica de triángulo entre los cuerpos

Con Listing 11 se genera la Fig. 5

```
tiempo_en_meses_lista= [] #lista vacia para
luego guardar las fechas
start_date = datetime.datetime.today() #
Fecha actual
tiempo_en_fecha = [start_date +
datetime.timedelta(days=t * 365.25) for t
in time] #variable con las fechas

for date in tiempo_en_fecha:

    tiempo_en_meses_lista.append(date.strftime("%d-%m"))
    #transformacion del tiempo a fechas
    (dia-mes)
```

Listing 12: Lista vacía para agregar fechas

```
fig = plt.figure() #para graficar las figuras
gs = gridspec.GridSpec(2, 1, height_ratios=[2,
1]) #para el grid

ax0 = plt.subplot(gs[0])
ax0.plot(time[600:1300], cos_y[600:1300],
color = 'darkblue', alpha = 0.6)
ax0.set_ylabel('cos($\gamma$)', loc = 'bottom')
ax0.axvline(tiempos_eclipses_solares[0], color
= 'darkred', lw = 0.8)
ax0.grid(lw=1)
```

```

ax1 = plt.subplot(gs[1], sharex = ax0)
ax1.plot(time[600:1300], cos_y[600:1300],
          color = 'darkblue', alpha = 0.6)
ax1.set_ylim(-1.004, -0.99)
ax1.axvline(tiempos_eclipses_solares[0], color
            = 'darkred', label = 'Proximo Eclipse
            Solar: 02/10/2024', lw = 0.8)
ax1.grid(lw=1)

plt.setp(ax0.get_xticklabels(), visible=False)

plt.xlabel('Fecha [dia-mes 2024]')
plt.subplots_adjust(hspace=.08)
ax1.set_xticks(time[600:1300],
               tiempo_en_meses_lista[600:1300])
plt.locator_params(axis='x', nbins=6)
plt.legend(shadow=True)
plt.show()

```

Listing 13: Gráfico para eclipses Solares

Con Listing 13 se genera la Fig. 7

```

fig = plt.figure()
gs = gridspec.GridSpec(2, 1, height_ratios=[2,
                                             1])

ax0 = plt.subplot(gs[0])
ax0.plot(time[1950:2400], cos_b[1950:2400],
          color = 'darkblue', alpha = 0.6)
ax0.set_ylabel(r'cos($\beta$)', loc = 'bottom')
ax0.axvline(tiempos_eclipses_lunares[0], color
            = 'darkred', label = 'Proximo Eclipse
            Lunar: 14/03/2025', lw = 0.8)
ax0.grid(lw=1)
ax0.legend(shadow=True, loc = 'upper left')

ax1 = plt.subplot(gs[1], sharex = ax0)
ax1.plot(time[1950:2400], cos_b[1950:2400],
          color = 'darkblue', alpha = 0.6)
ax1.set_ylim(-1.004, -0.99)
ax1.axvline(tiempos_eclipses_lunares[0], color
            = 'darkred', lw = 0.8)
ax1.grid(lw=1)

plt.setp(ax0.get_xticklabels(), visible=False)

plt.xlabel('Fecha [dia-mes 2025]')
plt.subplots_adjust(hspace=.08)
ax1.set_xticks(time[1950:2400],
               tiempo_en_meses_lista[1950:2400])
plt.locator_params(axis='x', nbins=6)
plt.show()

```

Listing 14: Gráfico para eclipses Lunares

Con Listing 14 se genera la Fig. 6.

El código se encuentra adjunto en el siguiente repositorio de GitHub: https://github.com/vahonorato/eclipse_predictions.