

# **ANALISIS DE SIMULACION DE ONDAS GRAVITACIONALES GENERADAS EN UNA COLISIÓN**

**BH-NS**

David Santiago Merchan  
Ciro Alejandro Gelvez  
Sebastian Anaya



# DESCRIPCION DEL SISTEMA FÍSICO

## Tipos de objetos del sistema estudiado:

w.metadata.object\_types

Tipo de objetos: **BHNS** (Black Hole - Neutron Star)

Otros tipos: **NSNS, BBH**

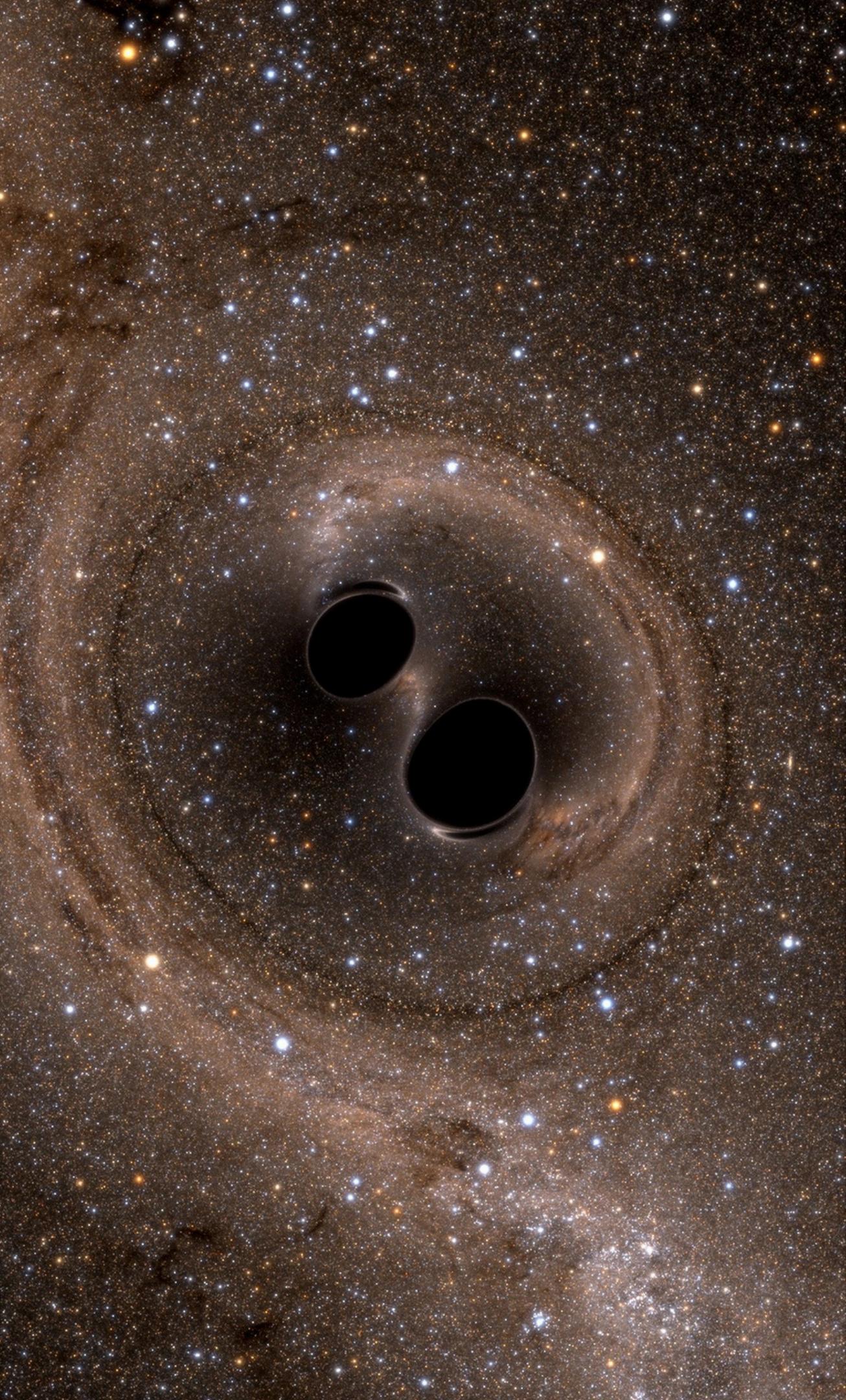
## Masas de los objetos antes de colisionar:

Masa BH: w.metadata.initial\_mass1

Masa NS: w.metadata.initial\_mass2

Masa BH: **8.4 [Msol]**

Masa NS: **1.4 [Msol]**



# DESCRIPCION DEL SISTEMA FÍSICO

**¿Los objetos giran? (Spin)**

**Espín BH:**

w.metadata.initial\_dimensionless\_spin1

**Espín NS:**

w.metadata.initial\_dimensionless\_spin2

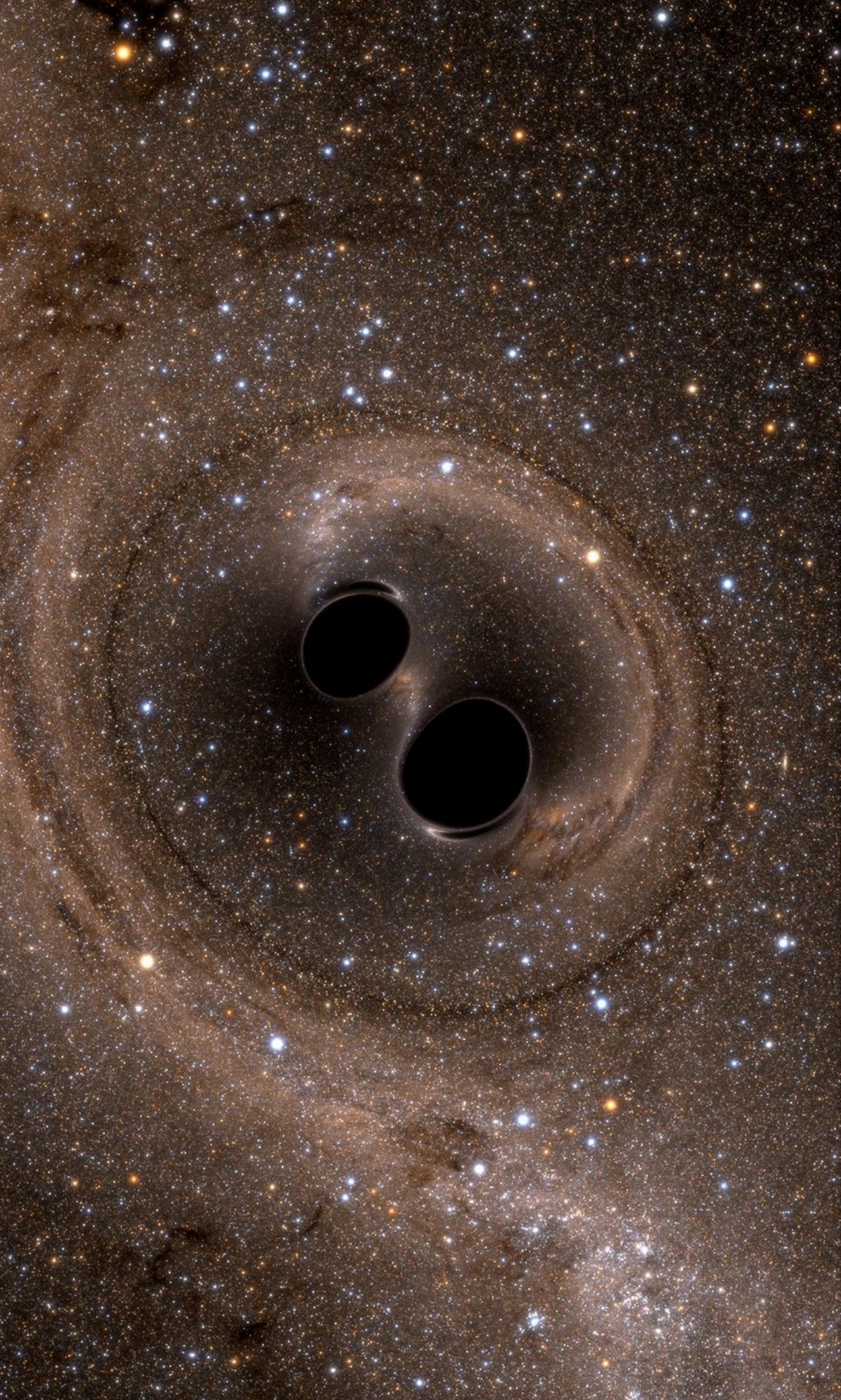
**Espín remanente:**

w.metadata.remnant\_dimensionless\_spin

**Espín BH:** [-2.17854e-11, -1.15479e-11, 7.88081e-08]

**Espín NS:** [0.0, 0.0, 0.0]

**Espín remanente:** [-5.33167e-06, -1.04486e-06, 0.37292]



# DESCRIPCION DE METADATA DEL SISTEMA

**¿Cuantos datos tiene la simulacion?**

**Tiempo de referencia:**

```
t0 = w.metadata.reference_time
```

**Filtro para evitar la “Junk Radiation”**

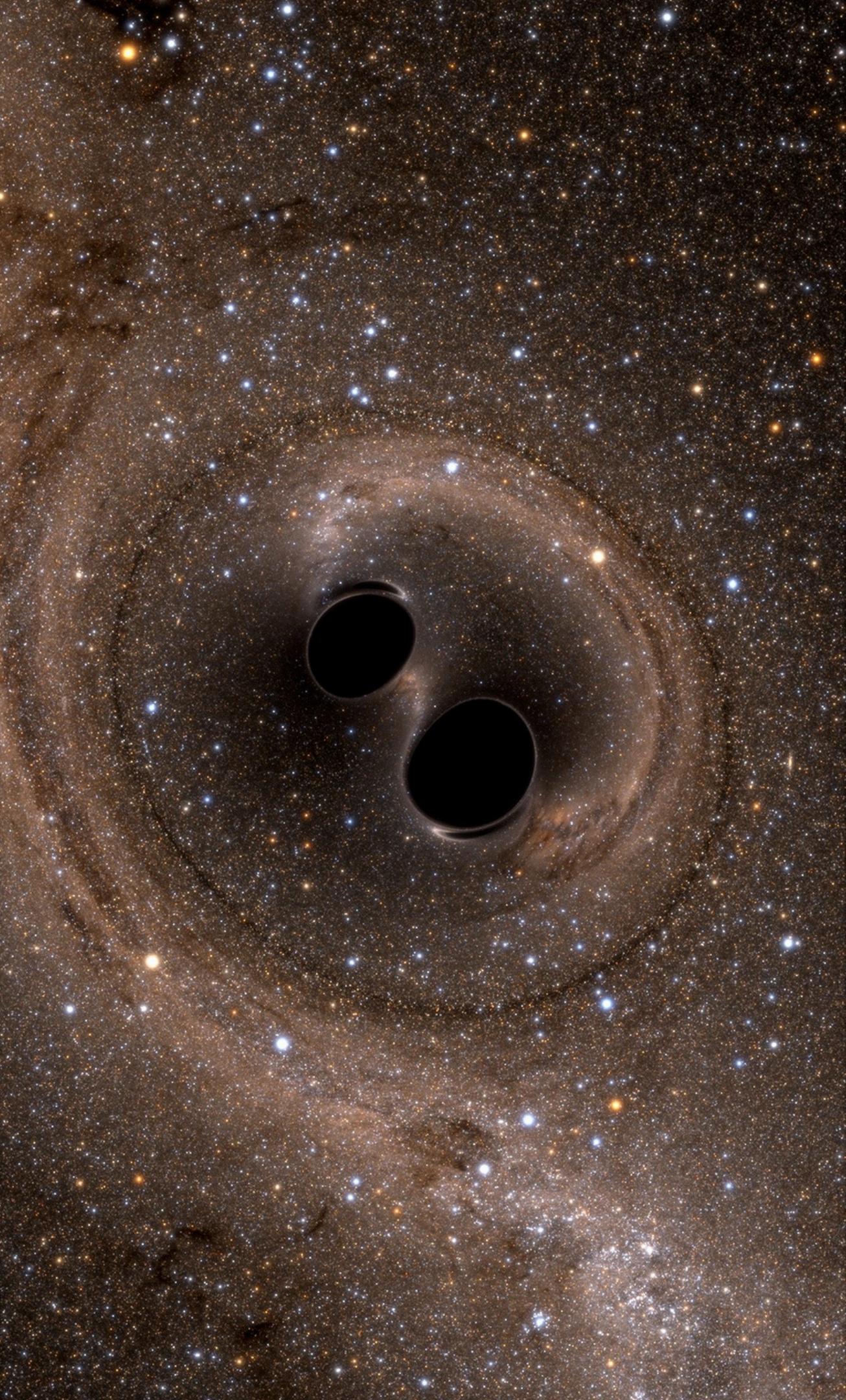
```
mask = w.t >= t0
```

```
t = w.t[mask]
```

**Cantidad de datos desde  $t$  confiable:**

```
N = t.size
```

```
N = 17508
```



# DESCRIPCION DE METADATA DEL SISTEMA

## Máximos y mínimos de los datos

Conversion de unidades temporales M a ms:

$$M_{tot} = w.metadata.initial\_mass1 + w.metadata.initial\_mass2$$

$$M\_to\_ms = M_{tot} * 4.925490947e-3$$

Tiempo inicial (min) y final(max) de los datos:

$$t[min, max] = t.min(), t.max()$$

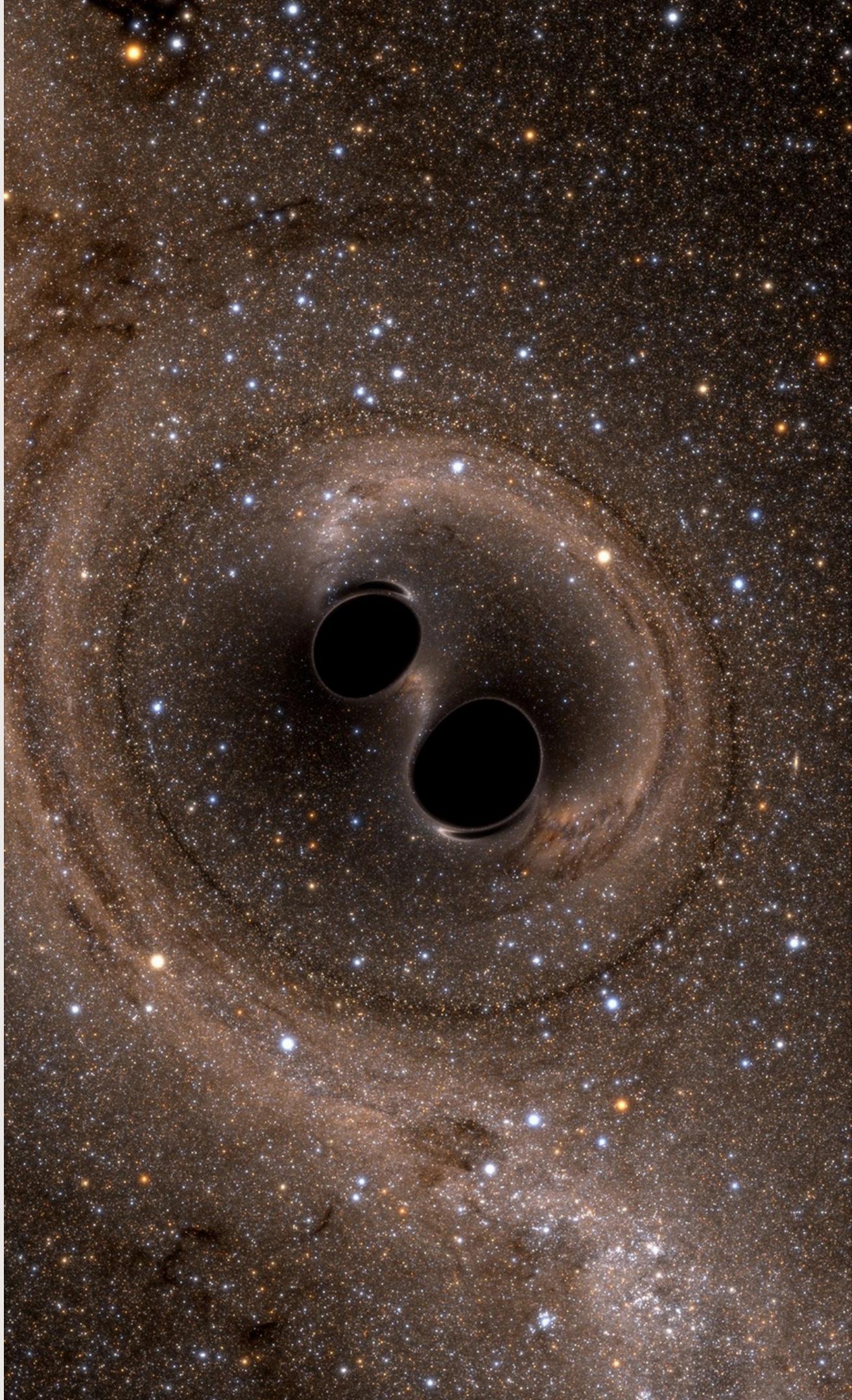
$$t[min, max] = [300.06179, 2458.45163] [M]$$

Tiempo total de los datos:

$$\text{tiempo total [ms]} = (t.max() - t.min()) * M\_to\_ms$$

$$\text{tiempo total [ms]} = 104.18507 \text{ [ms]}$$

La “Junk Radiation” dura aproximadamente **14.5 ms (300M)**



# DESCRIPCION DE METADATA DEL SISTEMA

## Máximos y mínimos de los datos

Fase maxima y minima de los datos:

fase[min, max] = phi.min(), phi.max()

**[-2.86413,189.26996]** [rad]

Fase total acumulada:

delta\_phi = phi\_max - phi\_min

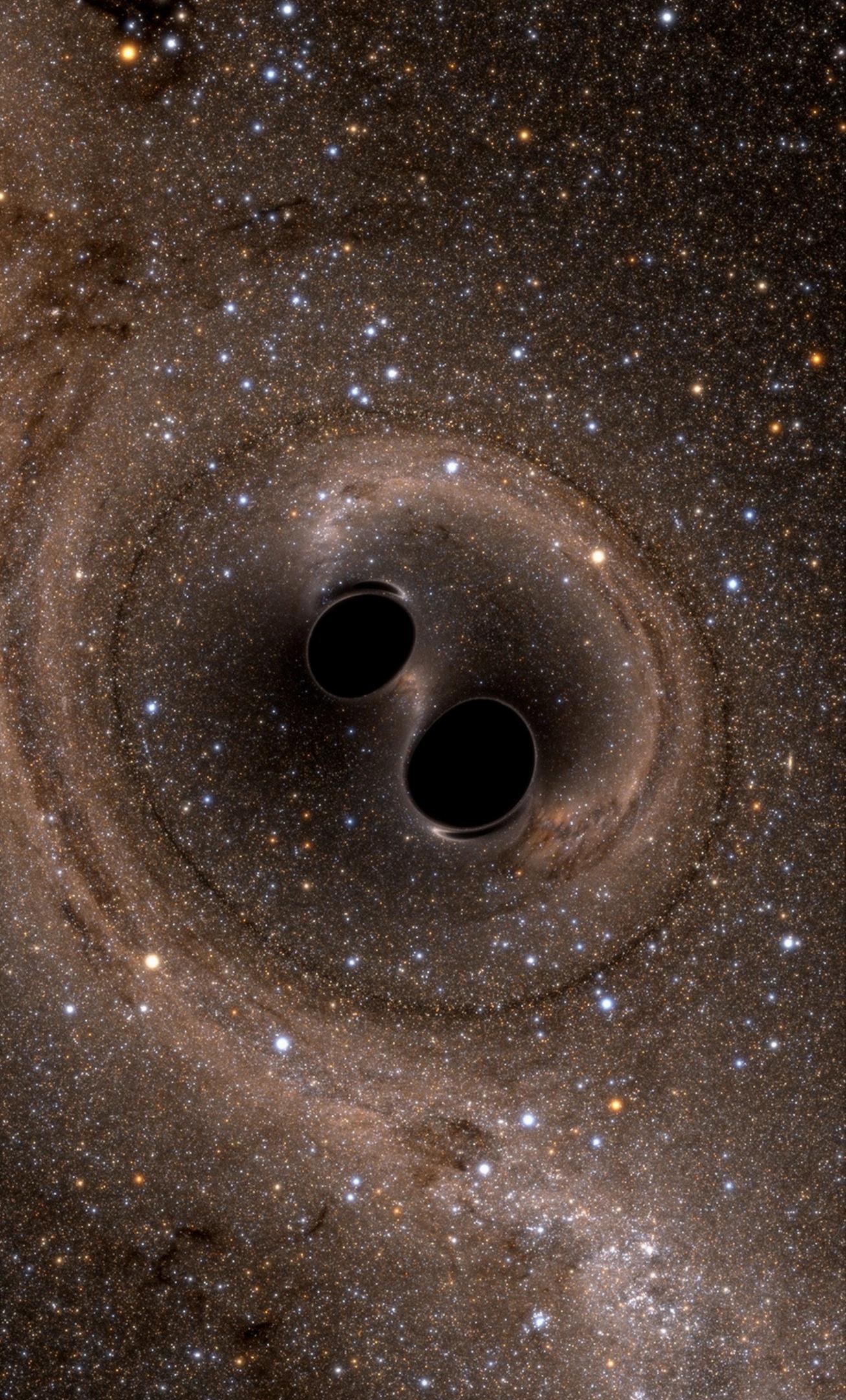
**192.13409** [rad]

Numero de ciclos:

N\_ciclos = delta\_phi / (2\*np.pi)

**30.57909**

Desde  $t_0$  hasta  $t.\max$ , se realizaron aproximadamente **15** orbitas  
alrededor del centro de masa comun entre los dos objetos



# DESCRIPCION DE METADATA DEL SISTEMA

## ¿Los datos tienen un paso constante?

Diferencia entre el dato  $i+1$  e  $i$ :

$dt = np.diff(t) [M]$

El paso entre datos es constante?

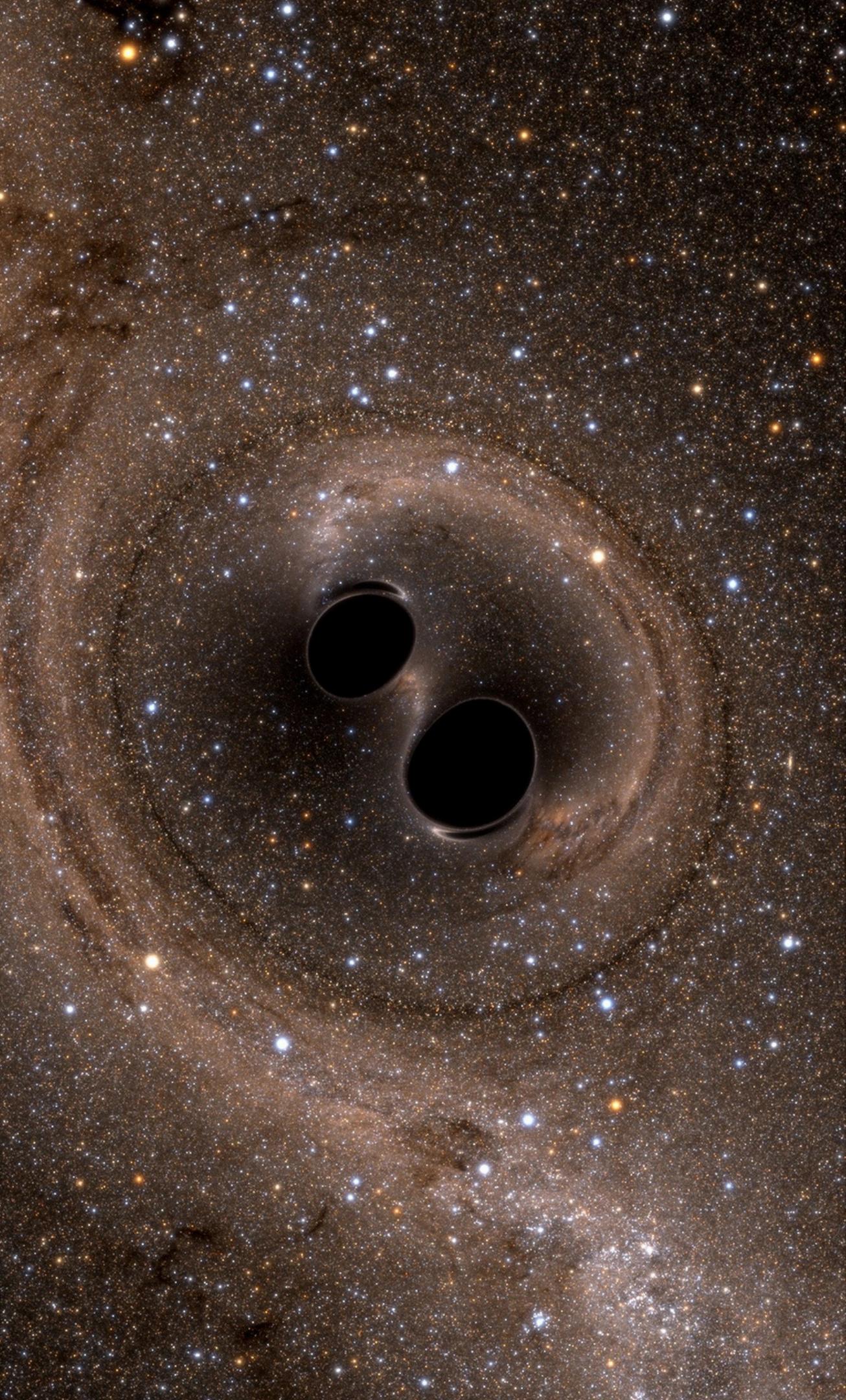
$paso\_const = np.allclose(dt, dt[0], rtol=1e-6, atol=1e-12)$

Paso constante = **False**

Hallamos la distancia entre datos min, mean, max:

$dt.min() * M\_to\_ms, dt.mean() * M\_to\_ms, dt.max() * M\_to\_ms$   
**[0.00246, 0.00595, 0.01971] [ms]**

El paso se puede considerar constante pero a tramos, es decir ***localmente*** pero **NO** de forma global



# INTERPOLACIÓN

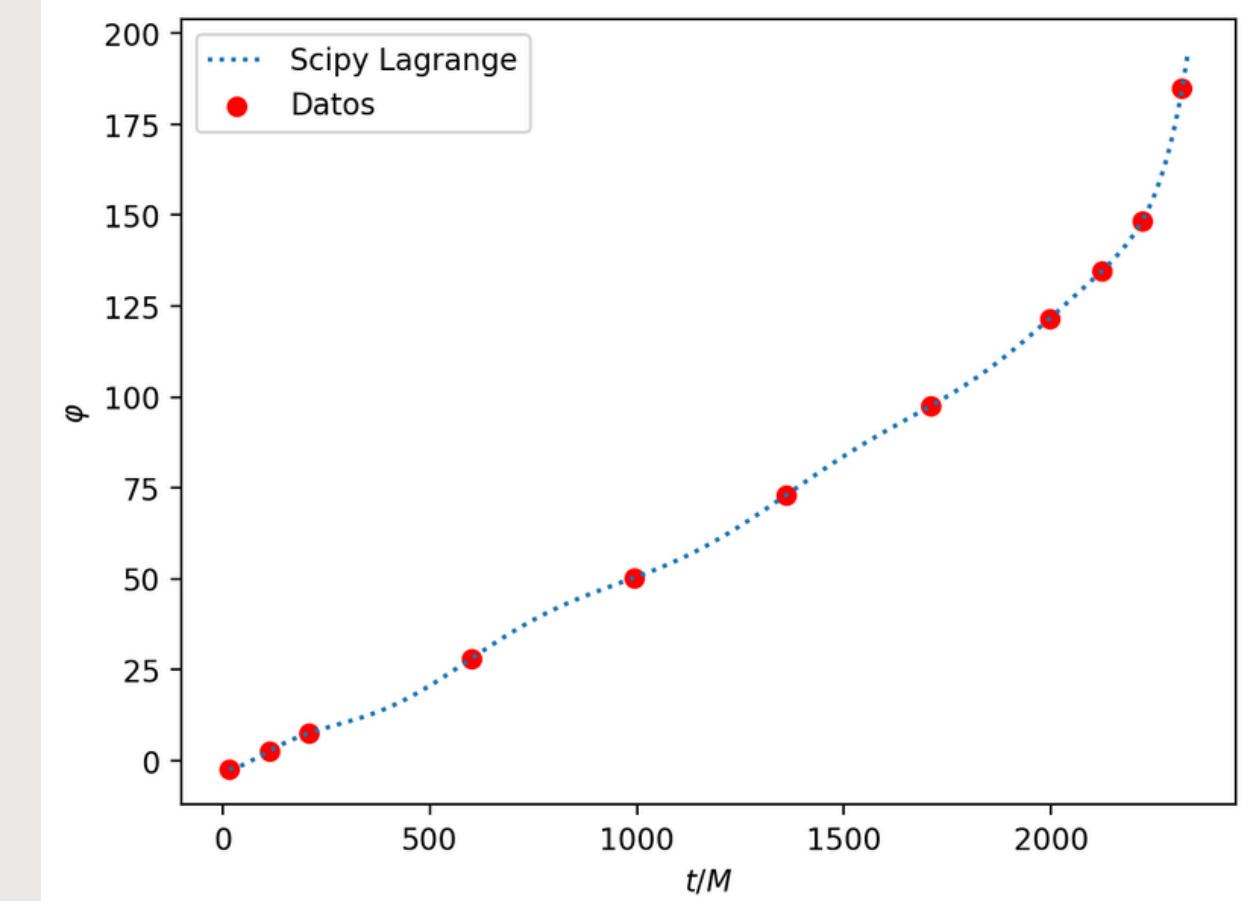
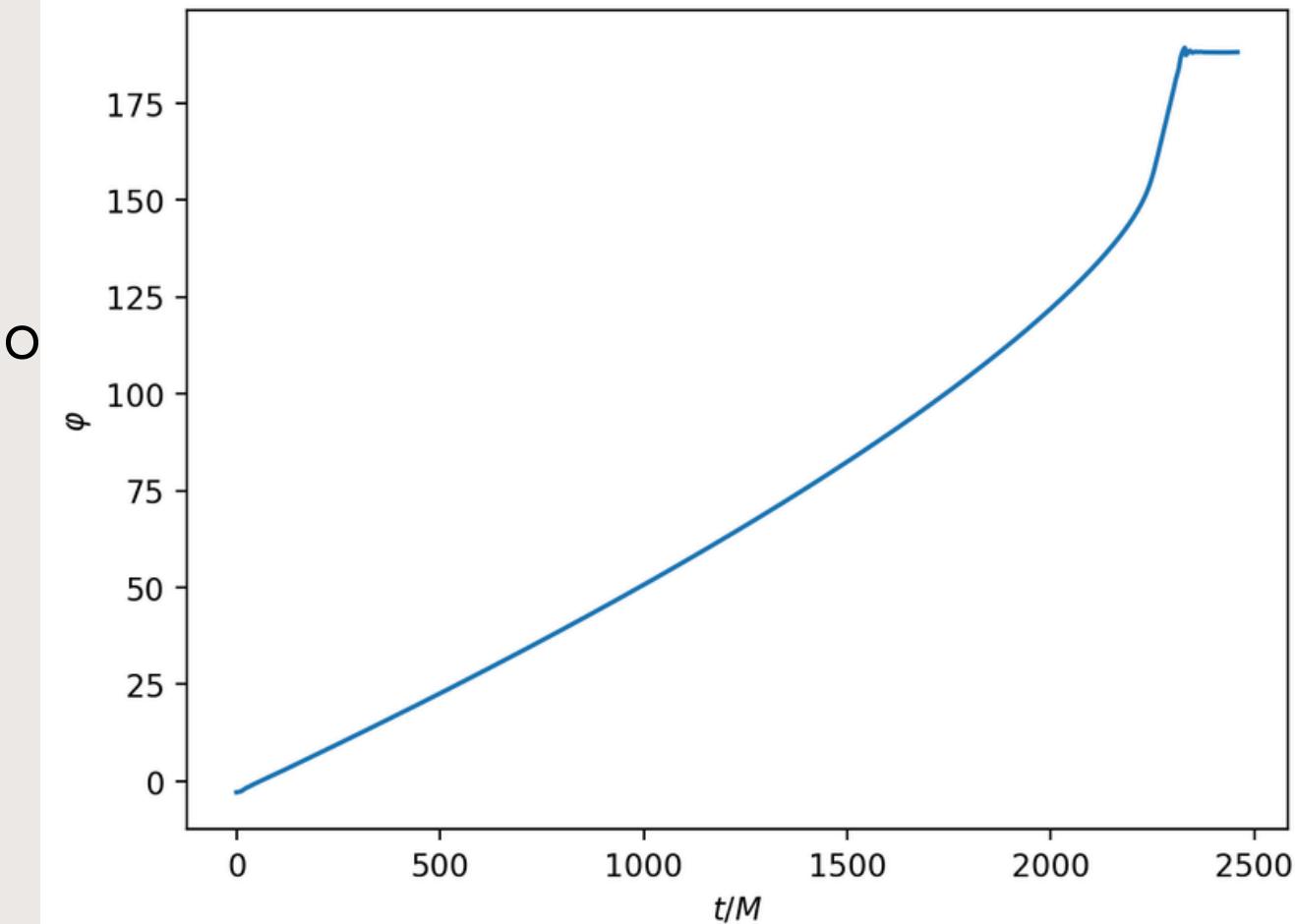
Se usa la interpolación para estimar nuevos puntos de datos dentro del rango de datos ya conocido, conectando los puntos de manera suave.

Se probaron cuatro métodos principales, todos de la biblioteca SciPy de Python.

## INTERPOLACIÓN DE LAGRANGE

Este método ajusta un único polinomio que pasa exactamente por un conjunto de puntos de datos.

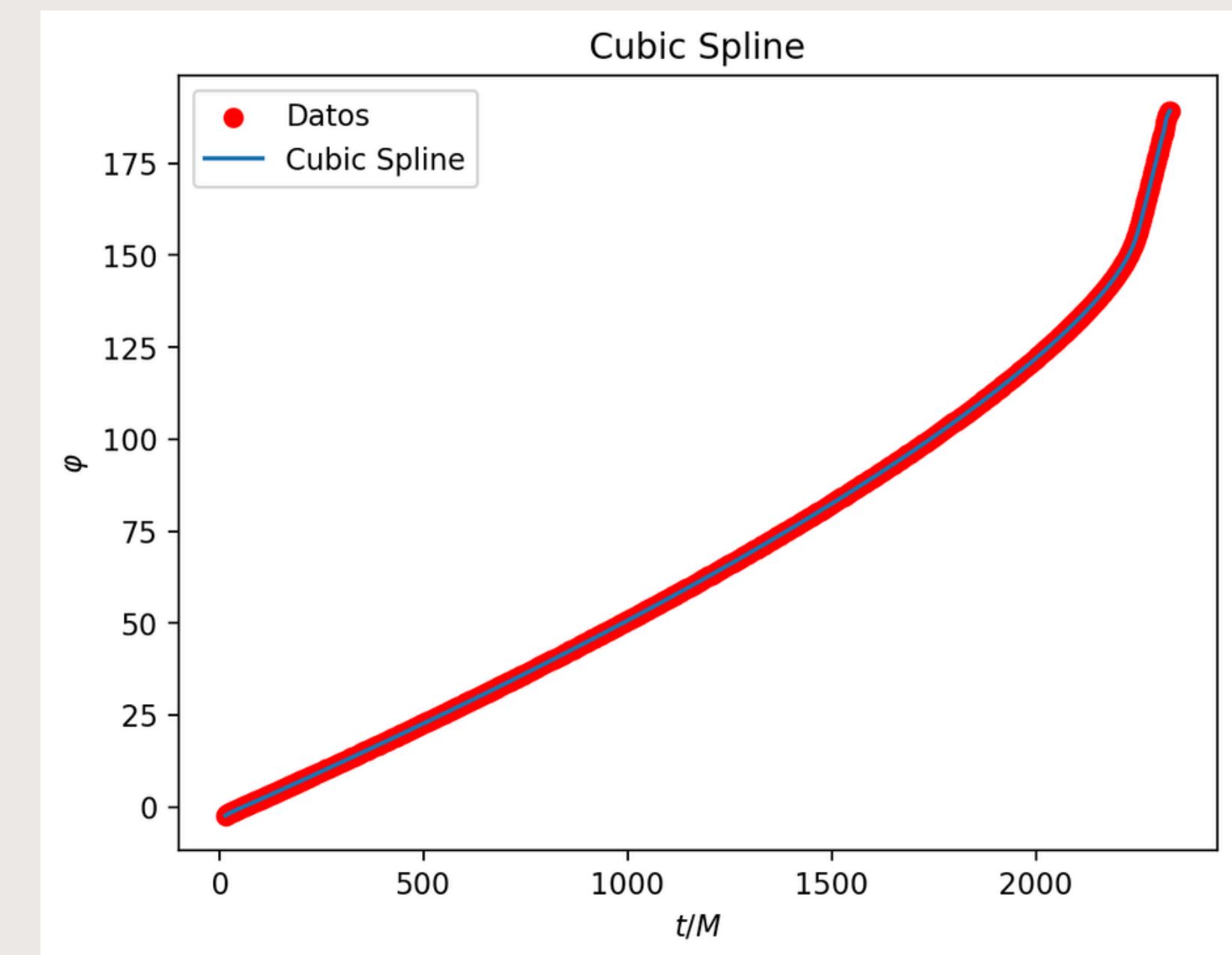
- Primero, se seleccionó un pequeño subconjunto de datos, solo 11 puntos.
- Se extrajeron los valores de tiempo ( $x_{\text{new\_La}}$ ) y los valores de la fase ( $y_{\text{new\_La}}$ ) de estos 11 puntos.
- Luego, se usó `scipy.interpolate.lagrange` para crear un objeto de interpolación. Este objeto representa el único polinomio que pasa por todos los puntos seleccionados.
- La curva resultante pasa precisamente a través de cada uno de los 11 puntos, creando una línea que los une.



# INTERPOLACIÓN DE SPLINE CÚBICO

Esta interpolación usa varios polinomios cúbicos para cada par de puntos de datos, asegurando que las curvas se unan de forma suave.

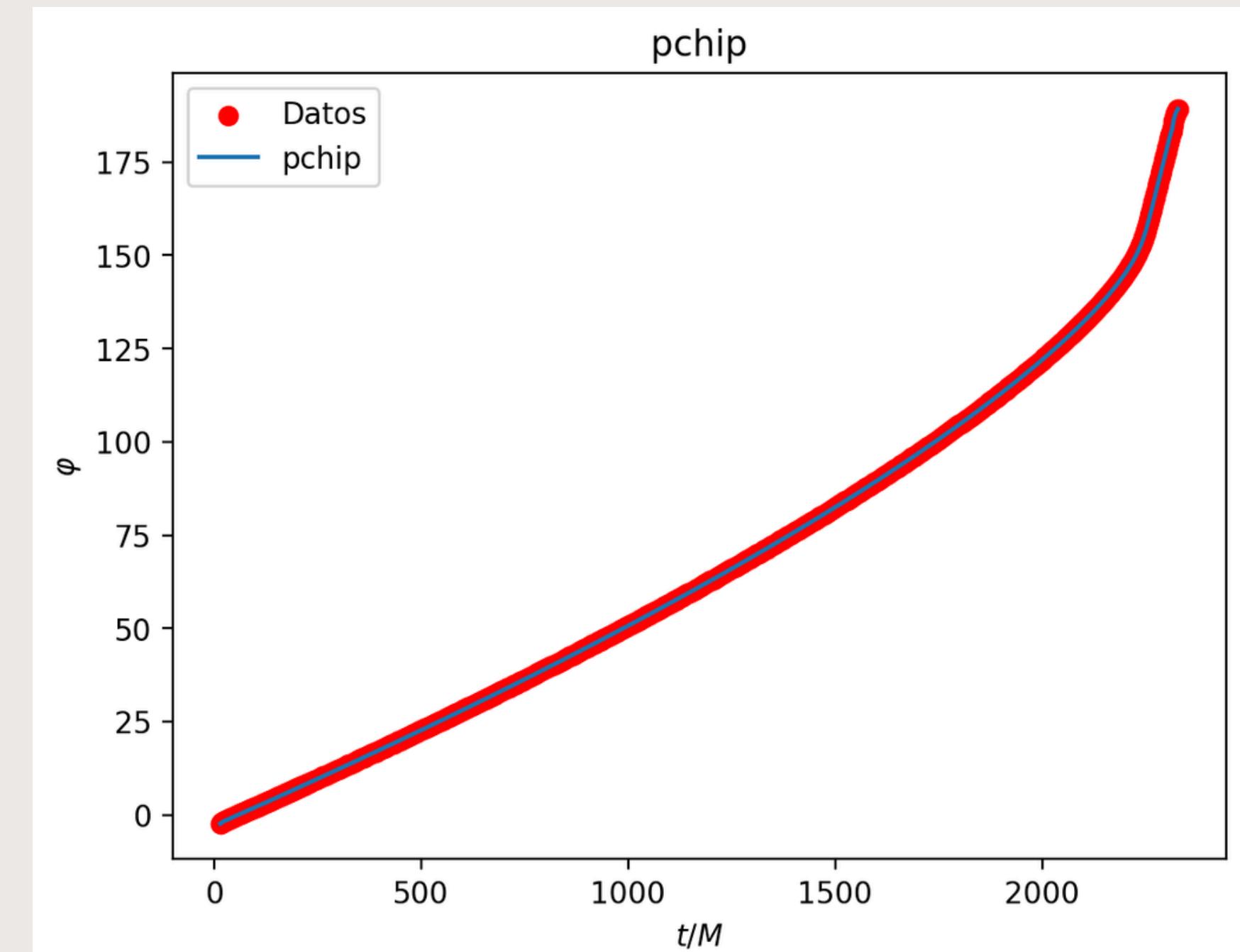
- Se seleccionó un conjunto más grande de 1014 puntos de la curva.
- Se extrajeron los valores de tiempo ( $x_{\text{new}}$ ) y fase ( $y_{\text{new}}$ ) de estos puntos.
- Se utilizó `scipy.interpolate.CubicSpline` para crear un objeto spline.
- Se especificó el parámetro `bc_type='natural'`, lo que indica que la segunda derivada en los puntos finales es cero, creando una curva más suave.
- La gráfica muestra los puntos de datos originales y la curva de spline cúbico que se ajusta muy bien a ellos. La curva parece superponerse perfectamente debido a la gran cantidad de puntos.



# INTERPOLACIÓN PCHIP

PCHIP (Polinomio de Interpolación Cúbico de Hermite a Trozos) preserva la forma de los datos, evitando la creación de picos o valles que no existen en los puntos originales. También se usaron 1014 puntos para este método.

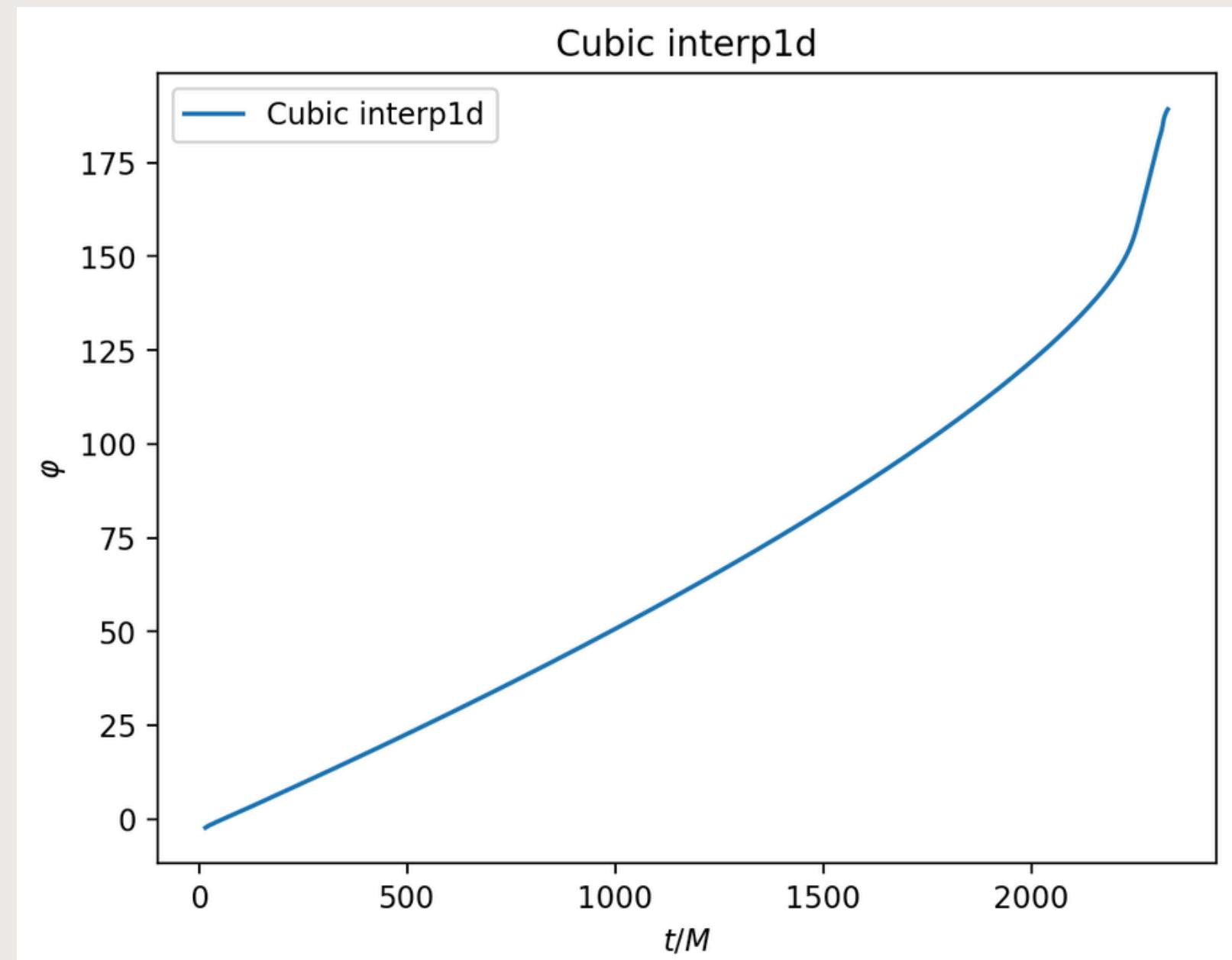
- De forma similar al Spline Cúbico, se usaron los mismos 1014 puntos.
- Se extrajeron los valores de tiempo ( $x_{\text{new}}$ ) y fase ( $y_{\text{new}}$ ).
- Se usó `scipy.interpolate.PchipInterpolator` para crear el objeto de interpolación.
- Este método es conocido por preservar la forma de la curva original, evitando oscilaciones adicionales.
- El gráfico resultante es casi indistinguible del spline cúbico, lo que confirma que ambos métodos se ajustan de manera muy precisa y suave a los datos.



# INTERPOLACIÓN DE CUBIC interp1d

interp1d es una función de la biblioteca SciPy diseñada para realizar interpolaciones en una dimensión. El argumento clave para esta interpolación es kind="cubic", que le indica a la función que utilice una técnica de spline cúbico. Esto significa que utiliza polinomios de tercer grado para conectar de manera suave cada par de puntos de datos.

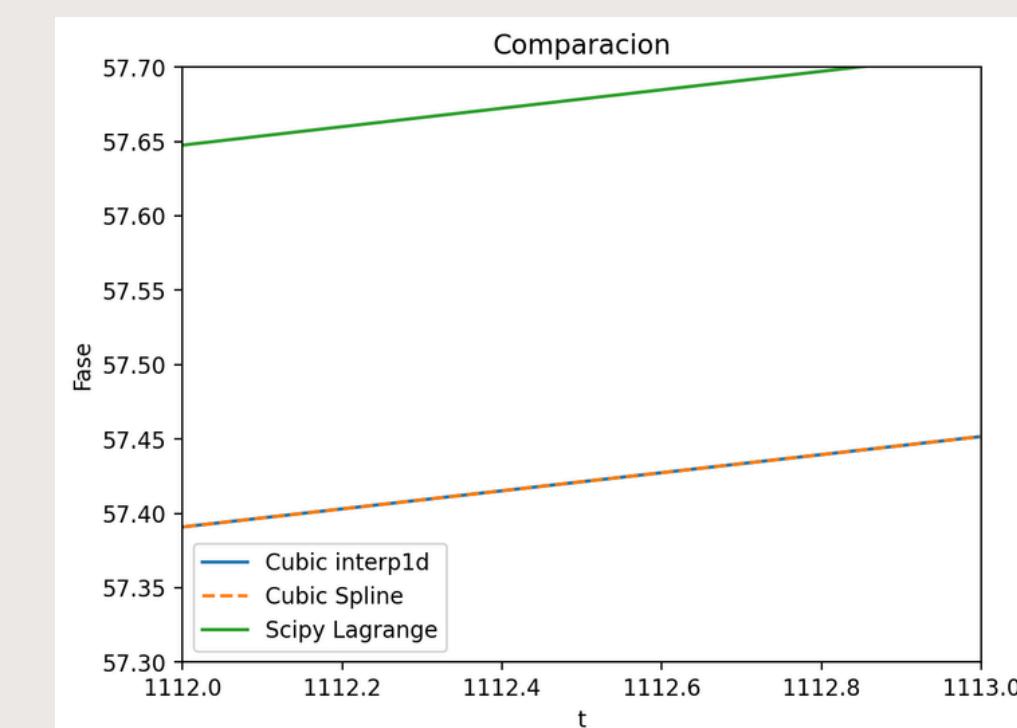
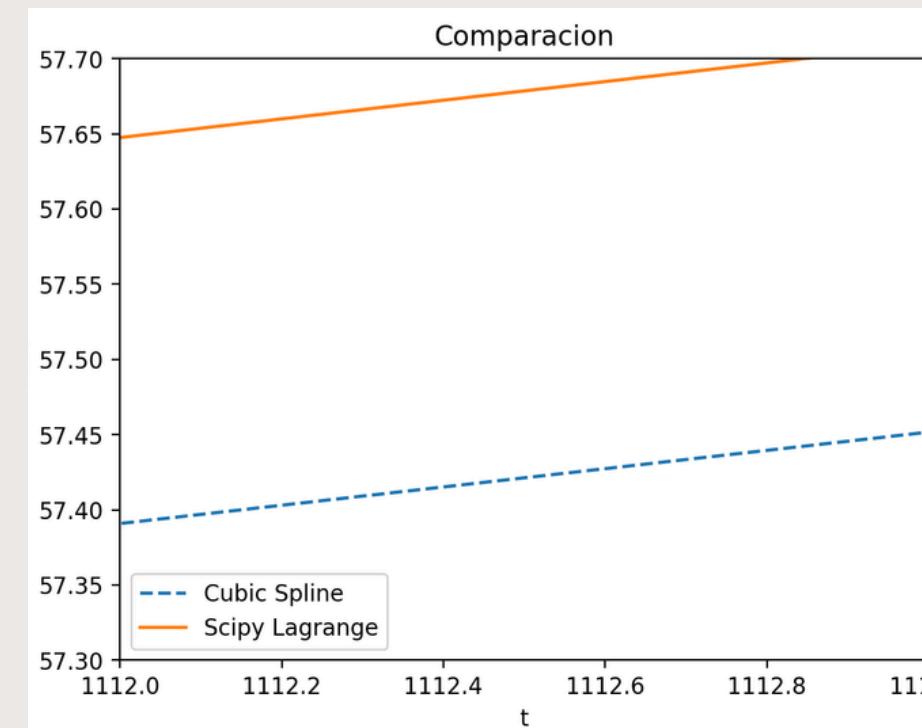
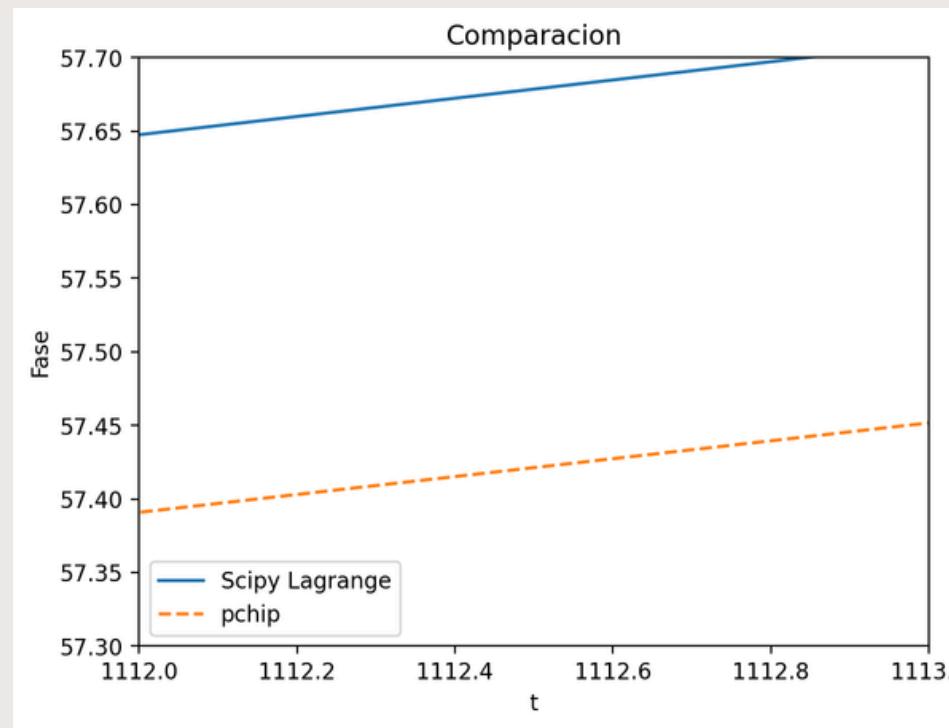
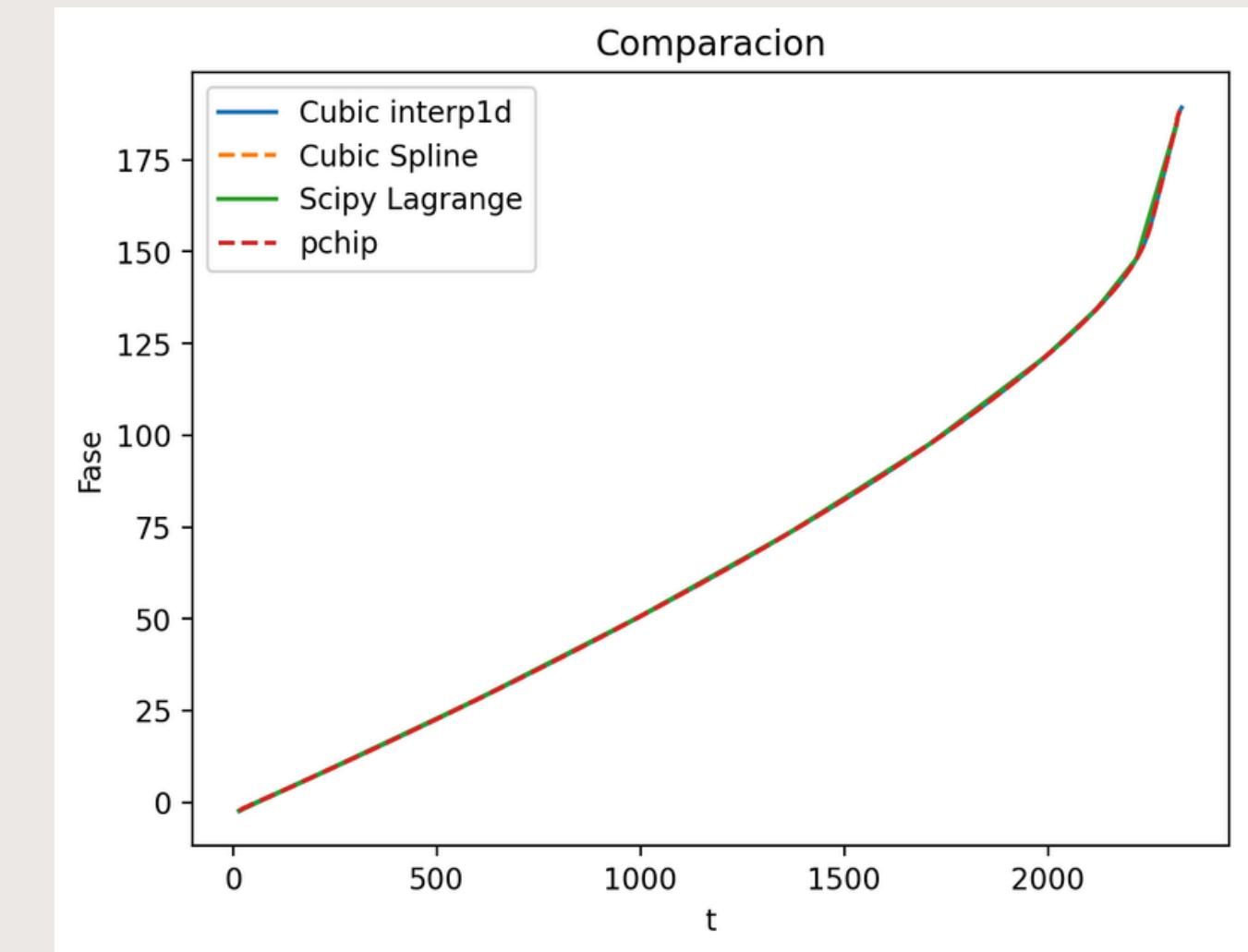
- Primero, se importó la función interp1d de la biblioteca `scipy.interpolate`.
- Se creó un objeto de interpolación llamado `f_cubic` utilizando los datos de entrenamiento (`x_new` y `y_new`) y especificando el tipo de interpolación cúbica.
- Luego, este objeto `f_cubic` se usó para calcular los valores de interpolación y obtener la curva final, que se guardó en la variable `mag_interp_cubic`.
- El objeto `f_cubic` también se utilizó para predecir valores en el conjunto de prueba (`x_test1`) y así calcular el (MSE)



# COMPARACIÓN DE METODOS

MSE lagrange 1.3215866353  
MSE spline 0.0000477590  
MSE Cubic 0.0000477246  
MSE pchip 0.0000477469

Para determinar la precisión, se calculó el Error Cuadrático Medio (MSE). Los métodos de Spline Cúbico, PCHIP en interp1d arrojaron un MSE muy bajo (casi cero), indicando un ajuste casi perfecto a los datos. En contraste, Lagrange mostró un error significativamente más alto, lo que la hace menos ideal para representar la curva de manera precisa, especialmente si se necesitan estimar puntos intermedios con exactitud.



# DATOS DE ENTRENAMIENTO

Los datos de entrenamiento ( $x_{\text{new}}$  y  $y_{\text{new}}$ ) se crearon seleccionando un subconjunto de los datos originales de la simulación.

- Se utilizó el comando `w_2_2.t[300:19558:19]` para seleccionar los valores de tiempo ( $x_{\text{new}}$ ) y `fphy[300:19558:19]` para los valores de fase ( $y_{\text{new}}$ ).
- El rango `[300:19558:19]` significa que se tomaron los datos desde el índice 300 hasta el 19558, con un paso de 19, lo que garantiza que no se utilicen todos los puntos de la simulación.
- Con esta selección, se obtuvieron 1014 puntos de datos para el entrenamiento.
- Para la interpolación de Lagrange, que requiere menos puntos, se seleccionó un subconjunto aún más pequeño de 11 puntos de los datos de entrenamiento.

# DATOS DE PRUEBA

Los datos de prueba ( $x_{\text{test1}}$  y  $y_{\text{test1}}$ ) se crearon para evaluar qué tan bien funcionan los modelos de interpolación con puntos que no se usaron durante el entrenamiento. Esto sirve para verificar que los modelos no se ajustan demasiado a los datos de entrenamiento y puedan generalizar correctamente.

- En  $x_{\text{test1}}$  se calculan los puntos medios entre cada par consecutivo de nodos de entrenamiento, generando un nuevo conjunto de prueba, usando  $0.5*(x_{\text{new}}[:-1] + x_{\text{new}}[1:])$ .
- En  $y_{\text{test1}}$  se obtienen los valores correspondientes de la señal original `fphy`, utilizando `searchsorted` para localizar los índices en `w_2_2.t`.

# ERRORES ABSOLUTOS Y CUADRÁTICOS

Se calcularon errores cuadráticos y absolutos para cada método de interpolación obteniendo los siguientes resultados:

Errores cuadráticos medios:

MSE lagrange: 1.3215866353

MSE spline: 0.0000477590

MSE Cubic: 0.0000477246

MSE pchip: 0.0000477469

Errores absolutos medios:

MAE lagrange: 0.8132817767

MAE spline: 0.0054395924

MAE Cubic: 0.0054372313

MAE pchip: 0.0054383721

```
# valores de y_test
y_test_lagrange = poly_lagrange(x_test1)
y_test_spline = spline(x_test1)
y_test_cubic = f_cubic(x_test1)
y_test_pchip = pchip(x_test1)
# calcular el error cuadrático
SE_lagrange = (y_test1 - y_test_lagrange)**2
SE_CS = (y_test1 - y_test_spline)**2
SE_cubic = (y_test1 - y_test_cubic)**2
SE_pchip = (y_test1 - y_test_pchip)**2
# mean square error np.mean() o .mean()
MSE_lagrange = SE_lagrange.mean()
MSE_CS = SE_CS.mean()
MSE_cubic = SE_cubic.mean()
MSE_pchip = SE_pchip.mean()

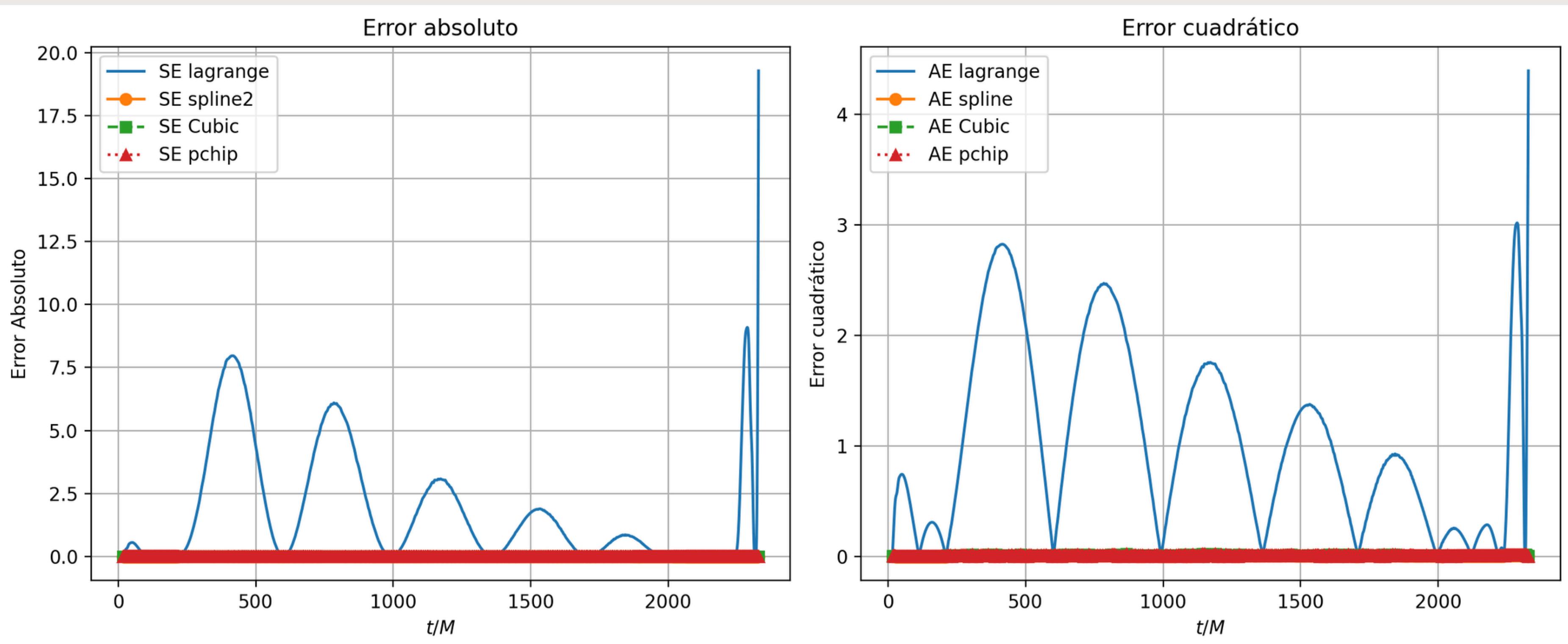
print(f'MSE lagrange {MSE_lagrange:.100f}')
print(f'MSE spline {MSE_CS:.100f}')
print(f'MSE Cubic {MSE_cubic:.100f}')
print(f'MSE pchip {MSE_pchip:.100f}')

# Error absoluto
AE_lagrange = np.abs(y_test1 - y_test_lagrange)
AE_CS = np.abs(y_test1 - y_test_spline)
AE_cubic = np.abs(y_test1 - y_test_cubic)
AE_pchip = np.abs(y_test1 - y_test_pchip)

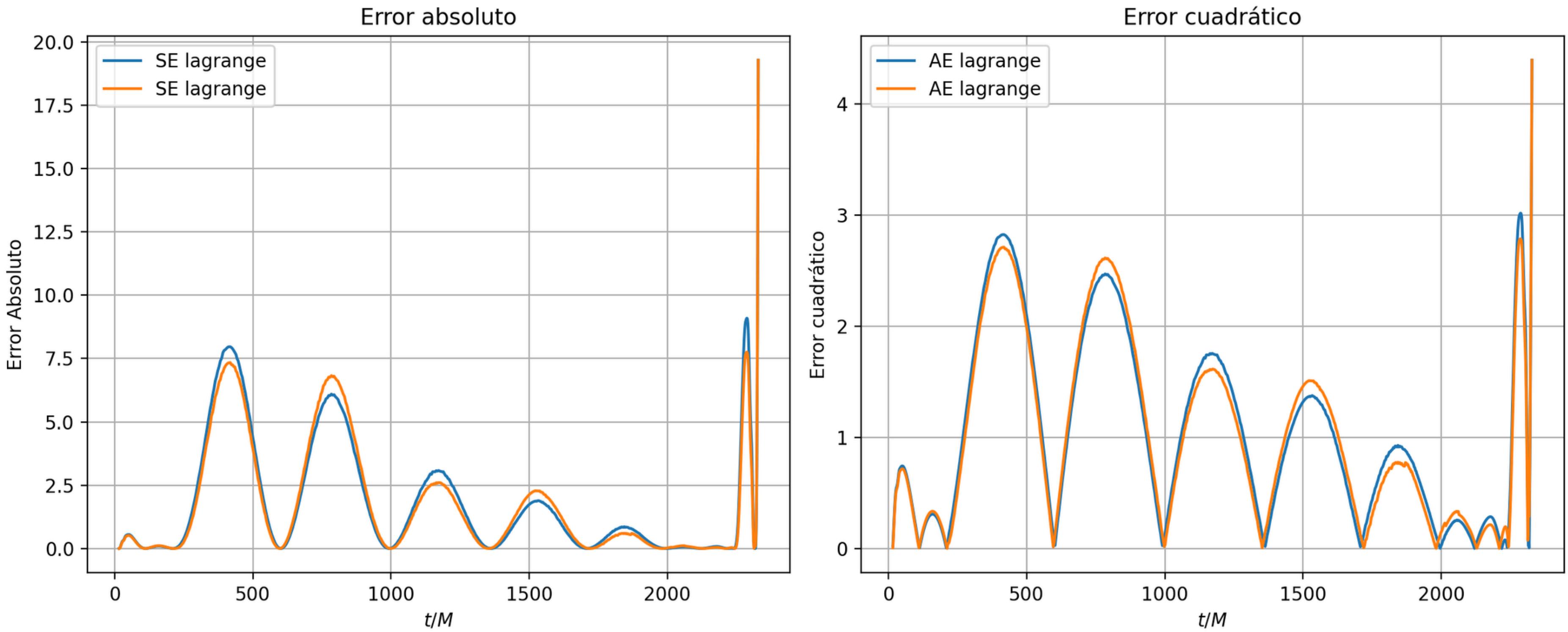
# Error absoluto medio
MAE_lagrange = AE_lagrange.mean()
MAE_CS = AE_CS.mean()
MAE_cubic = AE_cubic.mean()
MAE_pchip = AE_pchip.mean()

print(f'MAE lagrange {MAE_lagrange:.100f}')
print(f'MAE spline {MAE_CS:.100f}')
print(f'MAE Cubic {MAE_cubic:.100f}')
print(f'MAE pchip {MAE_pchip:.100f}'')
```

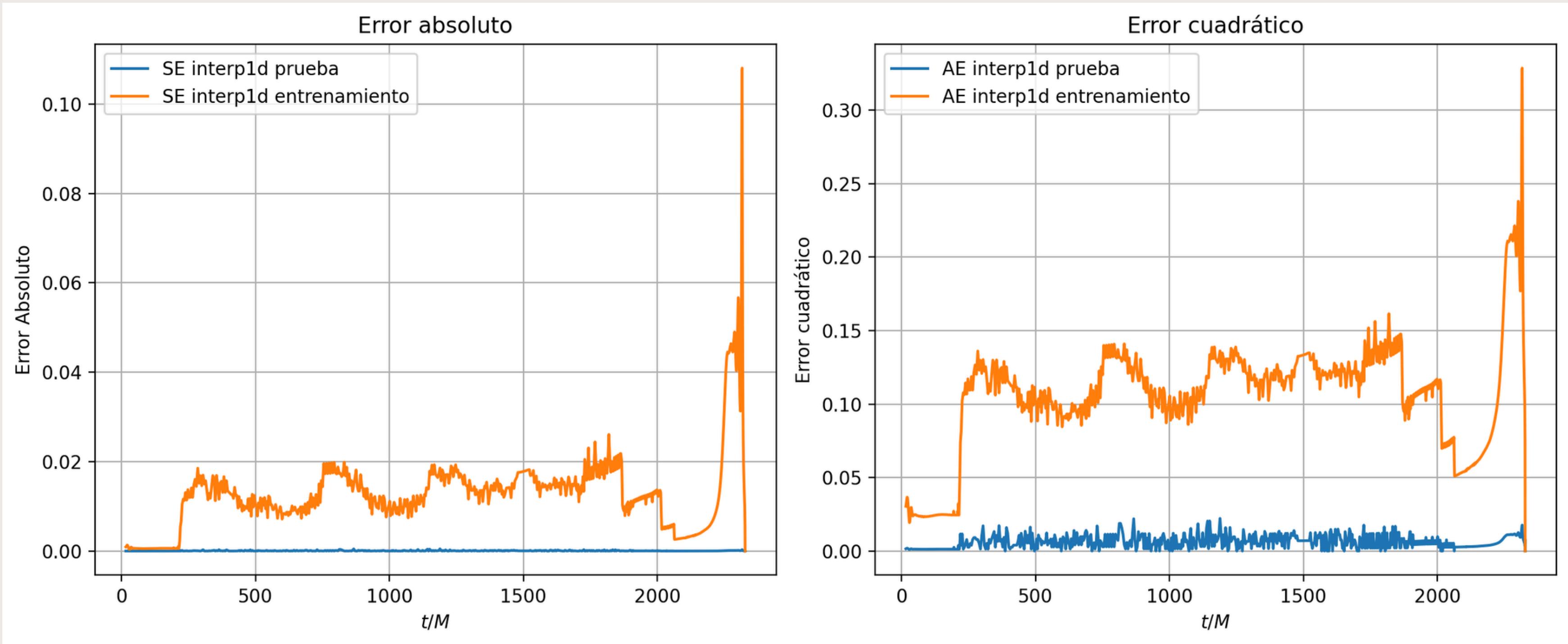
# GRÁFICA DE LOS ERRORES



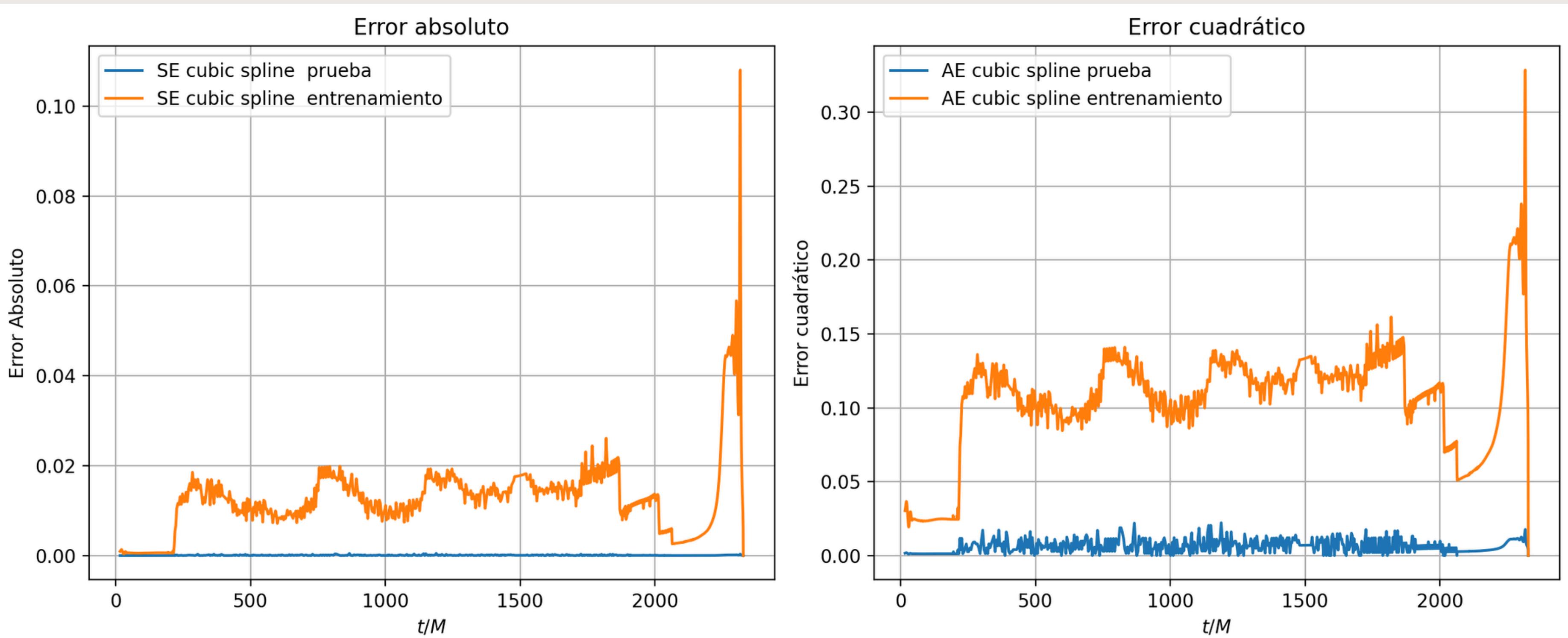
# COMPARACIÓN DE ERROR LAGRANGE



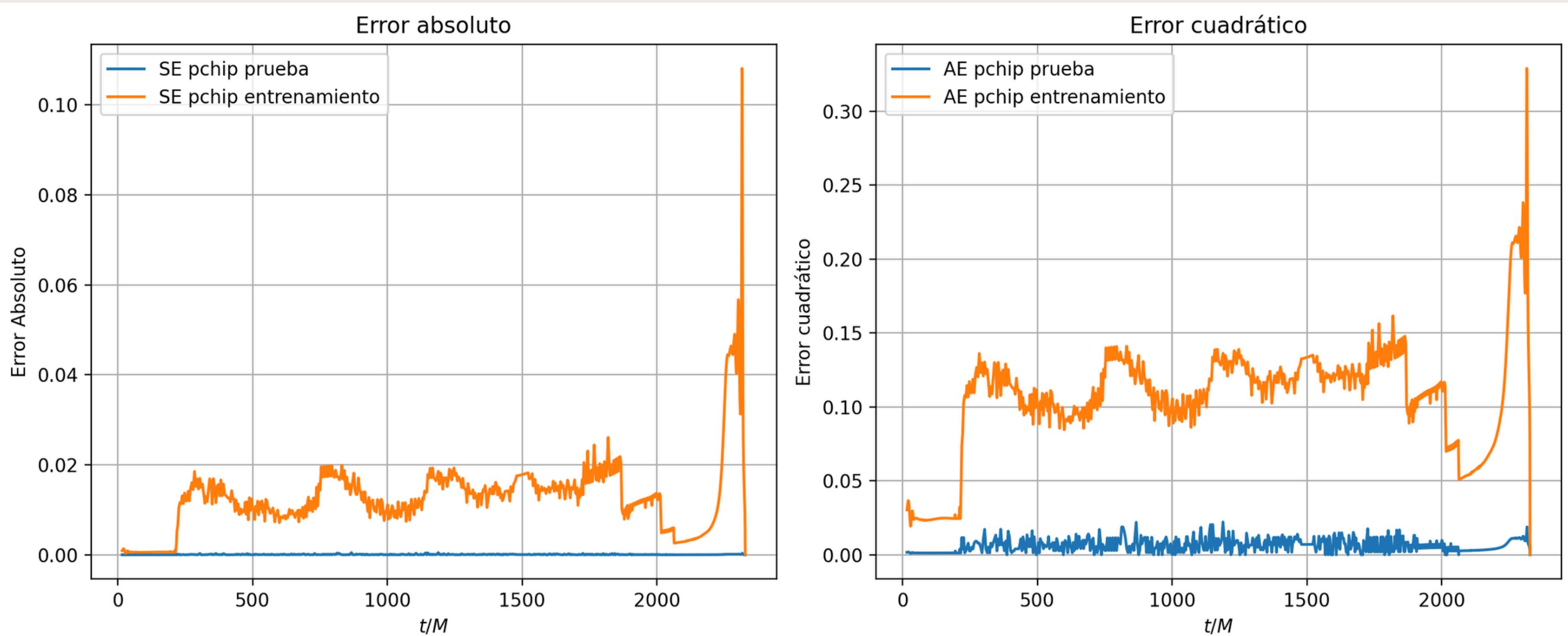
# COMPARACIÓN DE ERROR INTERP1D



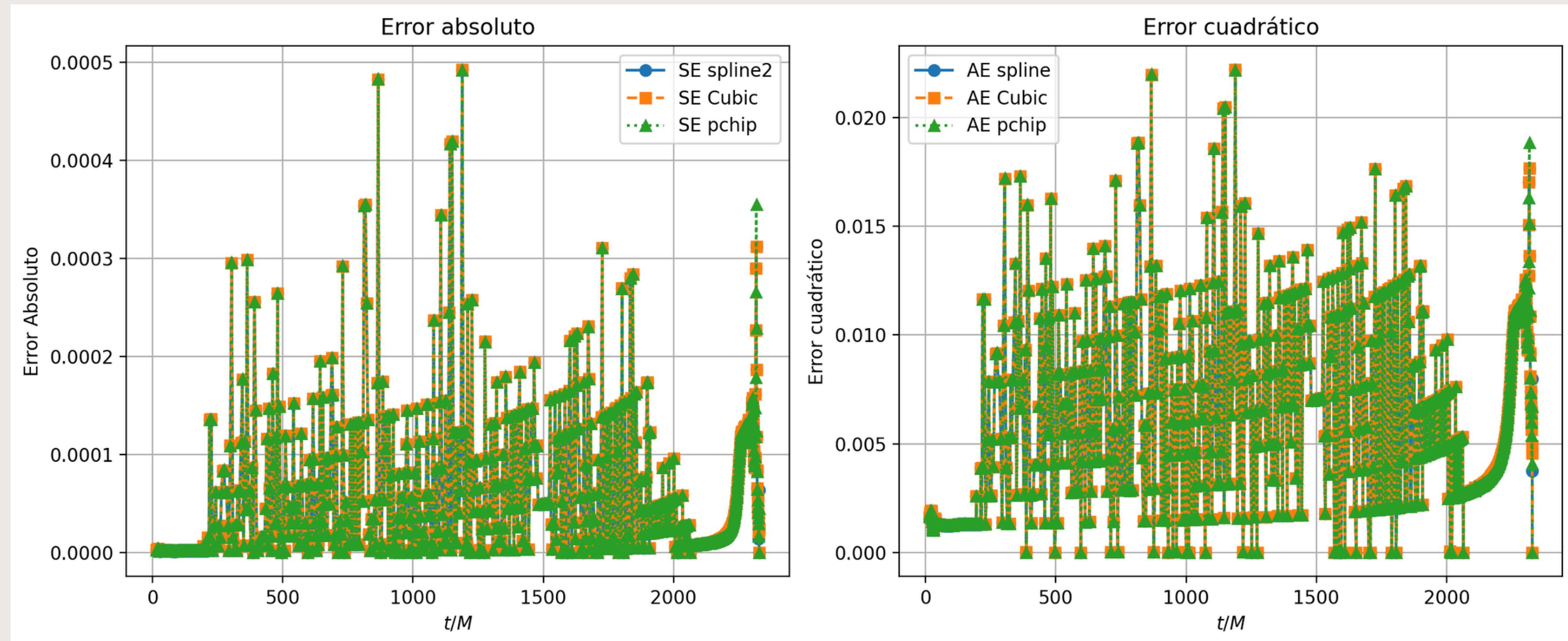
# COMPARACIÓN DE ERROR CUBIC SPLINE



# COMPARACIÓN DE ERROR PCHIP



# GRÁFICA DE LOS ERRORES



# MUCHAS GRACIAS



TURNO DE...

**PREGUNTAS (fáciles)**  
**COMENTARIOS (bonitos)**  
**CRÍTICAS (constructivas)**  
**APORTACIONES (monetarias)**

Y COMO DIJO MI EX, HASTA AQUÍ LLEGAMOS.

