# Informe de TPM2

*Facultad de Ingeniería, Universidad de Buenos Aires*

Materia: Modelación Numérica

Código: EB051

Curso: 06

Carrera: Ingeniería en Informática (ambos integrantes)

Cuatrimestre y año: 2C2024

## Integrantes


| Apellido y nombre | E-mail | Padrón |
|---|---|---|
| Riat Sapulia, Mateo | mriat@fi.uba.ar | 106031 |
| Reimundo, Martín | mreimundo@fi.uba.ar | 106716 |


Nombre del grupo: **Python01**

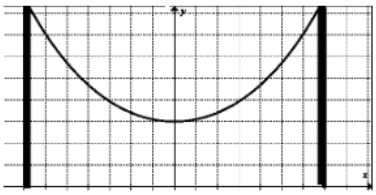
## Docentes

| Apellido y nombre | E-mail |
|---|---|
| Rodriguez, Daniel | drodriguez@fi.uba.ar |
| Machiunas, Valeria | vmachiunas@fi.uba.ar |

## Enunciado

**Análisis Numérico (75.12) – Modelación Numérica (CB051) - 2ºC2024**
**Trabajo Práctico de Máquina – Curso: Rodríguez**

Una catenaria es una curva generada por un cable *sin rigidez flexional*, suspendido de sus dos extremos y sometido a un campo gravitatorio.



La ecuación general de una catenaria, relacionando la posición x,y en cada punto, es la siguiente:

$$y(x)=\frac{\cosh(\mu x+C_1)}{\mu}+C_2 \qquad (1)$$

con las condiciones iniciales entre los parámetros:

$$y(x_0)=y_0=\frac{\cosh(\mu x_0+C_1)}{\mu}+C_2 \qquad (2)$$

$$y(x_1)=y_1=\frac{\cosh(\mu x_1+C_1)}{\mu}+C_2 \qquad (3)$$

$$\frac{senh(\mu x_1+C_1)-senh(\mu x_0+C_1)}{\mu}=L \qquad (4)$$

donde $x_0$, $y_0$ son las coordenadas de la posición de uno los extremos, $x_1$, $y_1$ de la posición del otro extremo y L es la longitud de la cuerda. Los parámetros $\mu$, $C_1$ y $C_2$ quedan determinados por las condiciones iniciales (ecuaciones 2, 3 y 4) .

**Primera parte (estudio del modelo de la catenaria)**

a) Verifique analíticamente que para el caso particular en que $y_0 = y_1$ e $x_0 = -x_1$ se cumplen las tres condiciones iniciales de los parámetros tomando $C_1 = 0$.

b) Suponga ahora que los extremos están en las posiciones $x_0 = -P/3500$, $x_1 = P/3500$, $y_0 = y_1 = P/2100$ y la longitud de la cuerda es P/1300 (u.a.) (redondee a un decimal y suponga dichos valores exactos; P es el promedio de los padrones del grupo). Obtenga los valores de los parámetros de la catenaria ($\mu$, $C_2$) y la posición de "y" correspondiente a x = 0, y expréselos bien redondeados. Para resolver la ecuación no lineal para $\mu$ implemente alguno de los métodos de resolución vistos en clase, y recuerde estimar las cotas de error que correspondan.

c) Repita el punto anterior pero ahora considere que los valores de las posiciones de los extremos de la cuerda y su longitud están bien redondeados (a un decimal). Para estimar las cotas de error propagado:

Primero: Estime la cota de error inherente de $\mu$ en forma experimental (perturbaciones experimentales) y analítica (propagándolo). Compare ambas estimaciones.

Segundo: Estime la cota de error inherente de $C_2$.

Tercero: Estime el valor de "y" y su cota de error cuando la posición de x = 0.

d) Estudie que parámetro ($x_1$, $y_1$ o L) tiene mayor peso en el error propagado de "y" cuando x = 0.



106031 – Riat Sapulia, Mateo 106716 – Reimundo, Martín

**Segunda parte (aplicación a una situación real)**

La ecuación de la catenaria es de utilidad en el análisis de situaciones ingenieriles como el diseño del tendido de cables de alta tensión, el tendido de cables que den soporte en puentes, y además su estructura, invertida, se emplea para diseñar arcos de puentes y en construcciones como arcos en iglesias (puede visualizar algunas de estas aplicaciones realizando una búsqueda en internet).
El problema de tomar los datos de una imagen o foto de una construcción existente, además del tema de la perspectiva de la foto -que altera las longitudes en la representación, complicando su estimación-, es la falta de información, por ejemplo, en una foto de torres de alta tensión tendrían que estimar la longitud del cable -que no es una línea recta-, y si bien es posible, eso excede el objetivo de este trabajo práctico.

Por lo tanto, se propone que armen unas "maquetas" con una cadena suspendida entre dos puntos de altura y separación conocidas. La cadena debe ser delgada y de eslabones pequeños. Deben suspenderla de forma tal que la cadena no toque el piso, ni la pared, debe quedar "en el aire", consigan que la cadena solo toque mínimamente los dos soportes. La maqueta debe tener (impresas o dibujadas con trazo fino) referencias en que permitan deducir distancias en horizontal y en vertical en la foto de la misma.
Deberán construir cuatro maquetas con las siguientes características: Si la longitud de la cadena es L (tengan cuidado al medirla), la separación entre los soportes deberá estar a una distancia de:

Caso 1: 0.8 L
Caso 2: 0.7 L
Caso 3: 0.5 L
Caso 4: 0.3 L

Para cada caso (cada separación), primero ajusten la curva catenaria aplicando lo estudiado en la primera parte de este trabajo práctico (*Nota: para este ajuste no se requiere la maqueta para eso*).

Para evaluar el modelo, en cada caso, coloquen una referencia en la maqueta para longitudes horizontales y verticales, realice 20 marcas entre $x_0$ y $-x_0$ *a distancias que decidan en el grupo*, y tomen una foto digital de frente, evitando el efecto de la perspectiva. Deben verse claramente: la cadena, las 20 marcas sobre una línea horizontal, la ubicación de los soportes, y dos referencias para poder estimar la escala en horizontal y en vertical en las fotos. Esas fotos deberán adjuntarse en la entrega del TP.

Sobre cada foto, midan las ordenadas de los 20 puntos de la cadena correspondientes a las marcas que hicieron, y regístrenlas. Pueden usar el software de procesamiento de imágenes **ImageJ** (tienen un breve tutorial por ejemplo en <https://www.ibv.csic.es/smicroscopia/Documentos/Primeros%20pasos%20en%20ImageJ.pdf> sobre descarga y primeros pasos al usarlo), o pueden medirlas con regla (y teniendo cuidado, tomen varias mediciones y promédienlas) sobre la foto impresa.
En cualquiera de los casos, estimen una cota de error inherente de las ordenadas de sus mediciones (expliquen su elección). Las 20 mediciones deben incluirse en una tabla (con ambas coordenadas).

En el siglo XV y XVI, tanto Leonardo Da Vinci como Galileo Galilei creyeron que la figura que describía una cuerda flexible suspendida de sus dos extremos era una parábola.

Estudien que tan bueno sería este ajuste, comparado con el de la ecuación de la catenaria.
Para ello, con los 20 puntos medidos, más los dos extremos, realicen un ajuste por cuadrados mínimos lineales de los parámetros de una función cuadrática.

Grafiquen en un mismo gráfico (generado con Octave o Python) para cada juego de valores, los dos modelos obtenidos y los 22 puntos (los 20 puntos medidos, más los dos puntos de los extremos).
¿Cuál parece ajustar mejor mirando el gráfico? Expliquen su elección.

Luego calculen las ordenadas correspondientes según cada modelo obtenido, y para cada modelo calculen el error cuadrático medio de los 22 puntos. ¿Cuál modelo ajusta mejor, en el sentido de los mínimos cuadrados? ¿Era lo esperado? Discutan entre ustedes y escriban sus conclusiones.

(Fin del enunciado)

## Introducción

En el presente trabajo se resolverá numéricamente el problema de la catenaria explicado en el enunciado. Inicialmente, se estudiará el modelo teórico de manera analítica con el fin de comprender el funcionamiento de la ecuación original de la catenaria y qué significado tienen cada uno de sus parámetros. Luego, se pasará a un modelo práctico que consiste en un experimento que simula este fenómeno, se tomarán mediciones sobre distintos casos y se resolverá numéricamente con el objetivo de comparar los errores que existen entre como ajusta el modelo teórico, el de los distintos casos en la realidad, y el de un ajuste cuadrático por mínimos cuadrados.

Una vez realizado el trabajo, se espera haber puesto en práctica los conocimientos adquiridos en las unidades 1 (errores), 2 (ecuaciones no lineales) y 3 (ajuste de funciones), además de aprender a hacer los cálculos, operaciones y experimentos pertinentes en un lenguaje de programación donde se pueda realizar este análisis.



106031 – Riat Sapulia, Mateo 106716 – Reimundo, Martín

## Desarrollo

Se arrancó verificando la congruencia de las ecuaciones (2), (3) y (4) del enunciado, para las condiciones iniciales del mismo. Para ello, partiendo de la igualdad $y_0 = y_1$ se igualan las ecuaciones (2) y (3), donde mediante operaciones básicas (ver anexo 3) se llega a la igualdad

$$\cosh(\mu x_0) = \cosh\left(\mu(-x_0)\right)$$

Verificando la primera condición, es decir que $x_0 = -x_1$.
Luego, si se reescribe la ecuación (4) con esta noción, se obtiene tras operaciones básicas:

$$\frac{2\sinh(\mu x_1)}{\mu} = L$$

Ecuación utilizada con mucha frecuencia en la resolución del grupo para estimar la raíz de $\mu$ mediante el método de refinamiento de Newton-Raphson (igualando dicha función a 0).

Una vez verificadas las condiciones iniciales del enunciado, se pasó a un caso práctico en donde se cumplen las mismas, cuya tarea fue encontrar las raíces $\mu, C_2$. En primer lugar, como en ninguna de las ecuaciones presentadas existe una combinación lineal respecto a la variable a estimar (en este caso $\mu$), se probó con todas en cuál podría haber un método en el cual converja a una solución. Para $\mu$, ocurrió que la más apta resultó ser la ecuación (4) de donde, igualando a 0 la misma (esto se realiza restando $L$ a ambos lados de la igualdad) se logró hallar un $\mu$.
Graficando la función reescrita en el párrafo anterior, se observó que tenía una raíz doble cercana a 0 (referencia en anexo 3), por lo que con una semilla lo suficientemente cercana a una de las raíces (en este caso consideramos el lado derecho), resulta posible aplicar el método de Newton-Raphson, ya que esto asegura su convergencia. Inicialmente y como no hay observaciones realizadas sobre los datos de entrada del enunciado, se consideraron los valores como exactos (se desprecian errores inherentes), y el error resultante fue de truncamiento, que por método de Newton-Raphson, la cota resultante se expresa como $|x_n - x_{n-1}|$. A priori esto también dependería de la tolerancia máxima que se le impone al método de refinamiento. Analizando el número de iteraciones, se encontró que con 7 iteraciones se llegaba a una cota de $0.5 * 10^{-6}$, y con una más a $0.5 * 10^{-16}$, con lo cual se ganaban 10 decimales más de precisión y se decidió ir por esa tolerancia (no era posible establecer una más baja porque era el rango de la mantisa del ordenador).
Ya obtenido el $\mu$ y expresado bien redondeado, se procedió a hallar $C_2$ de la ecuación (3), que despejando queda como $C_2 = y_1\mu - \cosh(\mu x_1)$ (con las condiciones iniciales $C_1 = 0$). Para encontrar su cota, se analizan los errores que acarrean los distintos parámetros de la igualdad. En este caso, el único es $\mu$ que ingresa a esta ecuación con un error inherente. Para propagar dicho error se deriva $C_2$ respecto de $\mu$ y se multiplica por su cota correspondiente. Una vez realizados

estos cálculos, se pudo expresar $C_2$ bien redondeado acotando por $0.5 * 10^{-t}$, donde $t$ indica la cantidad de decimales significativos.

Por último en esta parte, se pidió hallar $y(x = 0)$. Para ello, se utilizó la ecuación original (1) de la catenaria (enunciado), y reemplazando los datos se llegó a la solución de manera inmediata, y para expresarlo bien redondeado se halló su cota con la misma idea conceptual que para $C_2$, pero en este caso las variables de entrada que poseían error inherente eran dos, en vez de una: $\mu$ y $C_2$. Por ende,

$$\left|\varepsilon_{y(x=0)}\right| \leq \left|\frac{\partial y}{\partial \mu}\right| * \left|\varepsilon_{\mu}\right| + \left|\frac{\partial y}{\partial C_2}\right| * \left|\varepsilon_{C_2}\right|$$

Siendo $e_{\mu}$ y $e_{C_2}$ las cotas halladas previamente de $\mu$ y $C_2$ respectivamente.

Posteriormente, se busca analizar qué ocurre en el mismo caso sabiendo que los valores de entrada $x_0,\ x_1,\ y_0,\ y_1,\ L$ están bien redondeados a un decimal, lo que significa que

$$\left|\varepsilon_{x_0}\right| = \left|\varepsilon_{x_1}\right| = \left|\varepsilon_{y_0}\right| = \left|\varepsilon_{y_1}\right| = |\varepsilon_L| = 0.5 * 10^{-1}$$

Es importante aclarar que los cálculos y los métodos empleados para llevarlos a cabo no se modifican, por consiguiente solo hay que analizar la modificación de los errores en cada caso.

Primero se calculó la propagación de errores inherentes para $\mu$. Por definición, se haría como fue hecho previamente para $C_2$, pero en diferente contexto ya que $\mu$ fue hallado por el método de Newton-Raphson. Es por ello que debe propagarse el error tomando el $\varphi_{n-1}$ y derivándolo (además de multiplicar por la cota correspondiente) en cada parámetro declarado con error inherente. Para este caso particular $\varphi_{n-1}$ se expresa como

$$\varphi_{n-1} = \mu_{n-1} - \frac{F(\mu_{n-1})}{F'_{\mu}(\mu_{n-1})}$$

Para la ecuación (4) despejada, no dependía de $x_0, y_0, y_1$. La cota de error resultante es entonces

$$|\varepsilon_u| \leq \left|\frac{\partial \varphi_{n-1}}{\partial x_1}\right| * \left|\varepsilon_{x_1}\right| + \left|\frac{\partial \varphi_{n-1}}{\partial L}\right| * |\varepsilon_L| + E_{Truncamiento}$$

Donde el error de truncamiento es el mismo que el anterior, dado que se estableció en el uso del método.

Una observación relevante fue que en este punto no resultó tan sencillo saber cómo propagar el error para $\mu$ dado que no se sabía si tomar la ecuación (4) original en vez de la operada, en donde también tomaba protagonismo el $x_0$ y eso hacía que aparezca como término en la cota de error inherente. El grupo logró decantarse por esta última opción de la ecuación (4) dado que la raíz se aproximaba de la misma manera, pero comparando los errores esta opción obtuvo una disminución del error original.

La manera recientemente descrita en la cual se obtuvo la cota de $\mu$ fue de manera analítica, pero también se analizó sobre la cota de forma experimental. Para ello, se realizaron pequeñas perturbaciones al $\mu$ medido a ambos lados del mismo y se graficó para tener una noción (ver en

sección de resultados). Efectivamente, se encontró una perturbación con la cual la cota era aún inferior a la descubierta analíticamente (se comparó realizando la diferencia).

Con la cota ya hallada de $\mu$, se efectuó la propagación de errores inherentes de manera similar para $C_2$. Como este valor se encontró con un despeje, no resultó propagarlo en base a un $\varphi$, y por definición en la ecuación de $C_2$ se propagó como

$$|\varepsilon_{C_2}| \leq \left|\frac{\partial C_2}{\partial \mu}\right| * |\varepsilon_u| + \left|\frac{\partial C_2}{\partial x_1}\right| * |\varepsilon_{x_1}| + \left|\frac{\partial C_2}{\partial y_1}\right| * |\varepsilon_{y_1}|$$

Despreciando los errores de otro índole.

De manera similar se procedió para $|\varepsilon_{y(x=0)}|$ y la ecuación de dónde fue obtenida (catenaria):

$$|\varepsilon_{y(x=0)}| \leq \left|\frac{\partial y}{\partial \mu}\right| * |\varepsilon_u| + \left|\frac{\partial y}{\partial x}\right| * |\varepsilon_x| + \left|\frac{\partial y}{\partial C_2}\right| * |\varepsilon_{C_2}|$$

En realidad no se conoce el valor de $\varepsilon_x$, ya que los errores inherentes que se conocen acerca de $x$ es solo para los extremos $x_0, x_1$ según lo establecido en el enunciado, pero de igual manera en este caso se evalúa $x = 0$, por lo que se anula la derivada de la catenaria en esa posición, ergo ese término de la cota, por lo que finalmente quedaría

$$|\varepsilon_{y(x=0)}| \leq \left|\frac{\partial y}{\partial \mu}\right| * |\varepsilon_u| + \left|\frac{\partial y}{\partial C_2}\right| * |\varepsilon_{C_2}|$$

A criterio del grupo, se decidió expresarlo bien redondeado y como intervalo a estos dos últimos casos ($y(x = 0),\ C_2$) debido a que las cotas no eran lo suficientemente aptas (aún siendo las menores que se han experimentado para estos casos) para despejar un $t > 0$ de $0.5 * 10^{-t}$.

Tras todo este análisis abstracto se logró la idea de entender la manera en que una catenaria funciona, qué parámetros tiene y como analizar sus errores. Ahora se desarrollará lo mismo pero inversamente, es decir, a partir de un caso real se simulará el ejercicio (ver figuras de casos en anexo 2).

En primera medida, hubo un debate sobre qué materiales elegir para modelar la catenaria: con qué soportarlo, sobre qué plataforma ubicar los soportes, cómo ubicar los ejes, entre otras. La idea es construir un modelo real que se aproxime lo mejor posible al teórico, por lo que mientras más se minimicen las diferencias, mejor sería.

La cadena utilizada mide 50cm y cada eslabón tenía un espesor de 0,3cm. Como soportes se colocaron agujas, y la cadena enganchada en esas hizo que se desprecie 0,5cm de cada extremo, por lo que la longitud final considerada en el proyecto fue de 49cm.

Las agujas se pusieron sobre 4 hojas cuadriculadas A4, las cuales ofrecían como beneficio la escala; cada 2 unidades de cuadrado se forma 1cm.

Antes de colocar los ejes, para que sea lo más parecido al sistema de coordenadas que se aprecia en el enunciado, se priorizó colocar la cadena sobre arriba del eje x, por lo que todas las ordenadas queden positivas. El eje y se intentó ubicar de manera que interseccione perpendicularmente con el vértice formado por la cadena.

Luego, para cada caso se colgó la cadena de los soportes a una distancia entre ellos de 0,8, 0,7 y 0.3 veces la longitud de la cadena. Tomando 20 puntos entre el intervalo definido por las posiciones de los soportes sobre el eje x (en los gráficos anotados como $x_0, x_{21}$ y en el código como $x_0, x_1$), y siendo la distancia entre los puntos consecutivas definida por el grupo, se pudieron tomar los mismos puntos en los casos de 0,8 y 0,7 veces la longitud (cada 1,5cm) y cada 0,5cm en el caso 0,3 veces la longitud. Otra aclaración importante fue que la altura $y_0$ e $y_1$ se tomó en todos los casos por igual (24cm), dado que el eje x se mantuvo en la misma altura.
Con los puntos ya marcados, se tomaron fotos de cada maqueta y con el software recomendado por la cátedra ImageJ, se midieron la proyección de los puntos sobre la cadena (como convención, se tomaron por encima de ella) y con estos datos fue posible pasar a la implementación en código. Como cota de error inherente sobre las ordenadas, se tomó 0,3cm que corresponde al espesor, esto ya que si bien se decidió tomar la proyección por encima de la cadena, también podría haberse tomado por debajo, y el caso hipotético planteó esta cota del espesor (distancia entre esa hipotética ordenada y la medida).
En Python, se comenzó inicializando los valores conocidos $L, y_0, y_1$ y la cantidad de puntos por caso $n$. También en cada maqueta se conocen la posición de los soportes sobre el eje x $(x_0,\ x_{21})$ en las fotos, $(x_0,\ x_1)$ en el código.
A esta altura para modelar la ecuación de la catenaria se necesitaba hallar $\mu, C_2$ para cada uno de estos casos. En base a como se construyó la maqueta en cada situación, se cumplen las condiciones iniciales del enunciado (2), (3) y (4), por lo que resultó posible encontrar los valores de la misma manera que en la parte anterior.
Ya con $\mu, C_2$ obtenidos, se planteó la catenaria para cada caso (evaluando en sus respectivos puntos) y se guardaron los datos en un arreglo. Posteriormente, se realizó un ajuste por cuadrados mínimos de manera tal que la catenaria se aproxime por una cuadrática, o sea:

$$f^*(x) = c_0\varphi_o + c_1\varphi_1 + c_2\varphi_2$$

Donde $\varphi_o = 1,\ \varphi_1 = x,\ \varphi_2 = x^2$ y se hallaron los coeficientes por dicho método.
Una vez obtenida $f^*(x)$, se evaluó para cada caso y en los respectivos puntos de cada uno, y también se almacenaron estos datos en un arreglo.
Finalmente, para analizar la comparación de las distintas estimaciones, se calculó el error cuadrático medio entre las mediciones de las fotos y la catenaria, y entre las mediciones de las fotos y el ajuste cuadrático. Además, para tener una idea más intuitiva, se graficaron las funciones (imágenes en sección de resultados).

$$Error\ cuadrático\ medio = \sqrt{\frac{\sum_{i=0}^{21} \omega_k \left(f^*(x_k) - f(x_k)\right)^2}{n}}$$

Reemplazando el ajuste previo, o la catenaria, según el error cuadrático medio que se desee hallar. Por convención, $\omega_k = 1, n = 22$

Como conclusiones sobre las comparaciones, en todos los casos de manera analítica, el error cuadrático medio dio menor en los ajustes cuadráticos, y es también visible en los gráficos el hecho de que se aproxima más a los puntos de las fotos por lo que resultó más precisa.

## Resultados

Parte 1)

b) Valores bien redondeados:

- $\mu = 0.0452347114575708$
- $C_2 = 0.1877466921834858$
- $y(x=0) = 22.29466343327906$

c)

- $|\varepsilon_\mu|\ (analítica) = 4.912575795925581 * 10^{-5} \rightarrow bien\ redondeado = 5 * 10^{-5}$
- $|\varepsilon_\mu|\ (experimental) = 3.7389579801427036 * 10^{-6} \rightarrow bien\ redondeado = 4 * 10^{-6}$
- $|\Delta\varepsilon_\mu| = 4.5386799979113107 * 10^{-5}$
- $\mu\ (bien\ redondeado\ por\ analítica) = 0.0452$
- $|\varepsilon_{C_2}| = 0.0647017779296913 \rightarrow bien\ redondeado = 0.07$
- $C_2 = 0\ bien\ redondeado$ ó $0.1877466921834858 \pm 0.07\ como\ intervalo$
- $|\varepsilon_{y(x=0)}| = 0.08871031044939455 \rightarrow bien\ redondeado = 0.09$
- $y(x=0) = 22\ bien\ redondeado$ ó $22.29466343327906 \pm 0.1\ como\ intervalo$

Parte 2)

Caso 0.8L: Valores bien redondeados

- $\mu = 0.06034313978532292$
- $C_2 = -0.3366139483485733$
- $ECM_{catenaria} = 5.369972573010284$
- $ECM_{cuadratica} = 0.09547278207066552$

Caso 0.7L: Valores bien redondeados

- $\mu = 0.08830641188186822$
- $C_2 = -0.26408122686673563$
- $ECM_{catenaria} = 2.813739754120727$
- $ECM_{cuadratica} = 0.21835433294394332$

Caso 0.3L: Valores bien redondeados

- $\mu = 0.4078014209306157$
- $C_2 = -0.2538203665716541$
- $ECM_{catenaria} = 1.180772193276597$
- $ECM_{cuadratica} = 1.0866821557099977$

Gráficos:

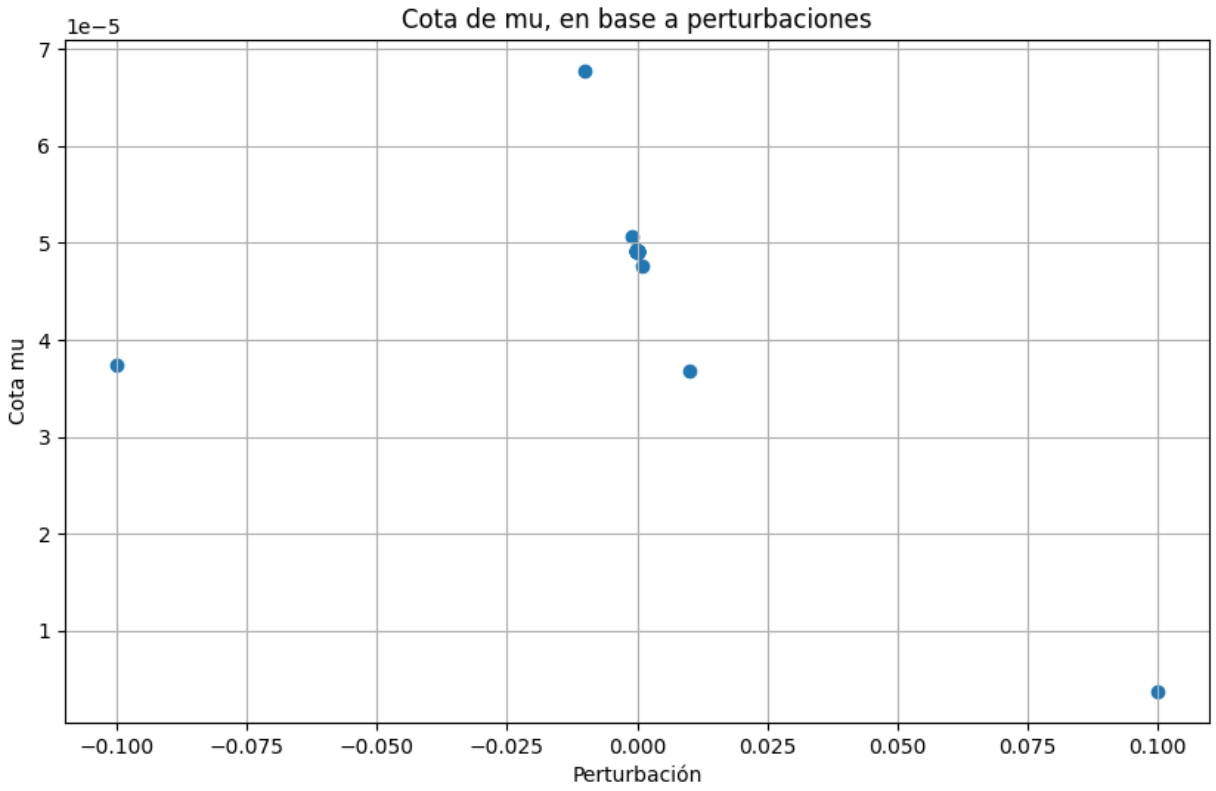



Figura 1: Cota de $\mu$ en función de las distintas perturbaciones

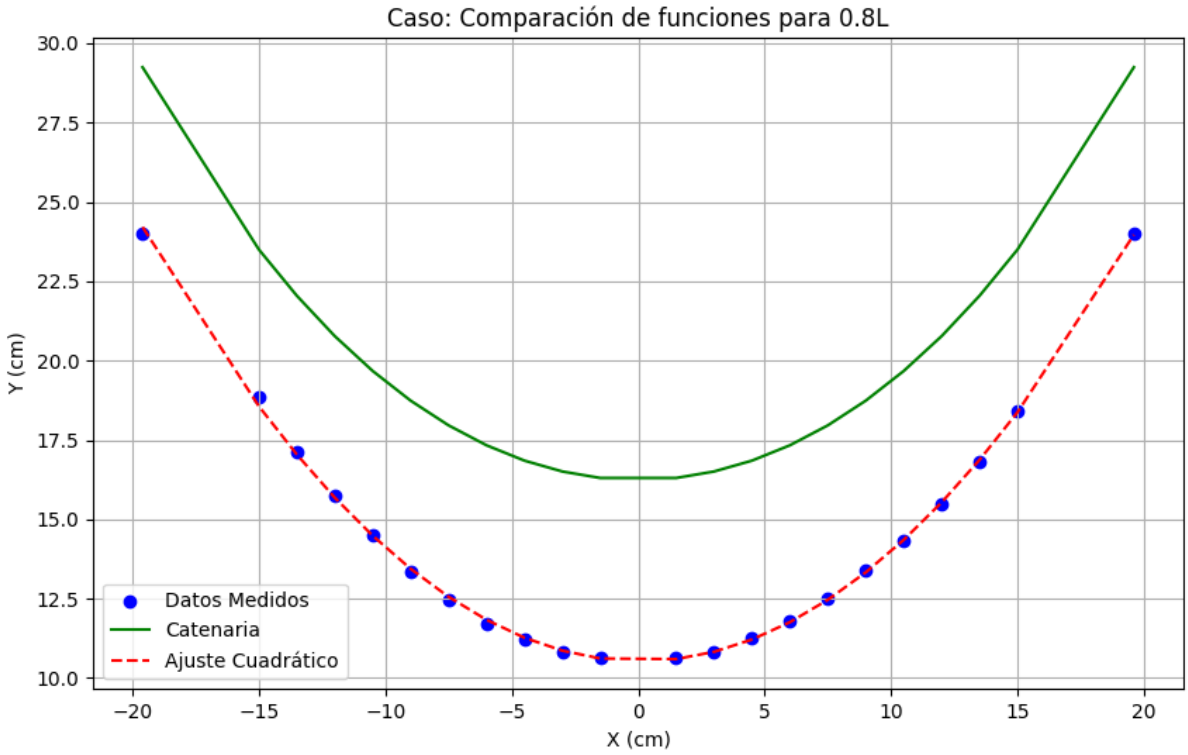



Figura 2: Comparación de estimaciones para 0.8L

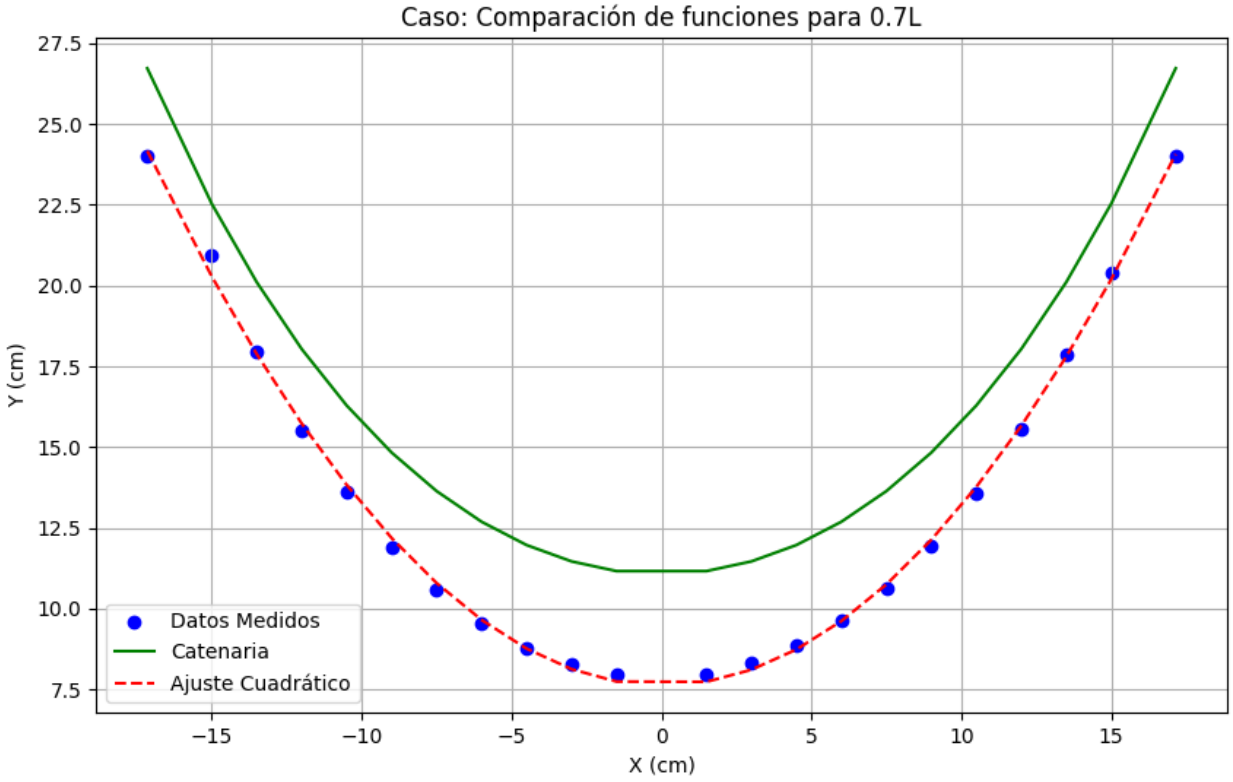



Figura 3: Comparación de estimaciones para 0.7L

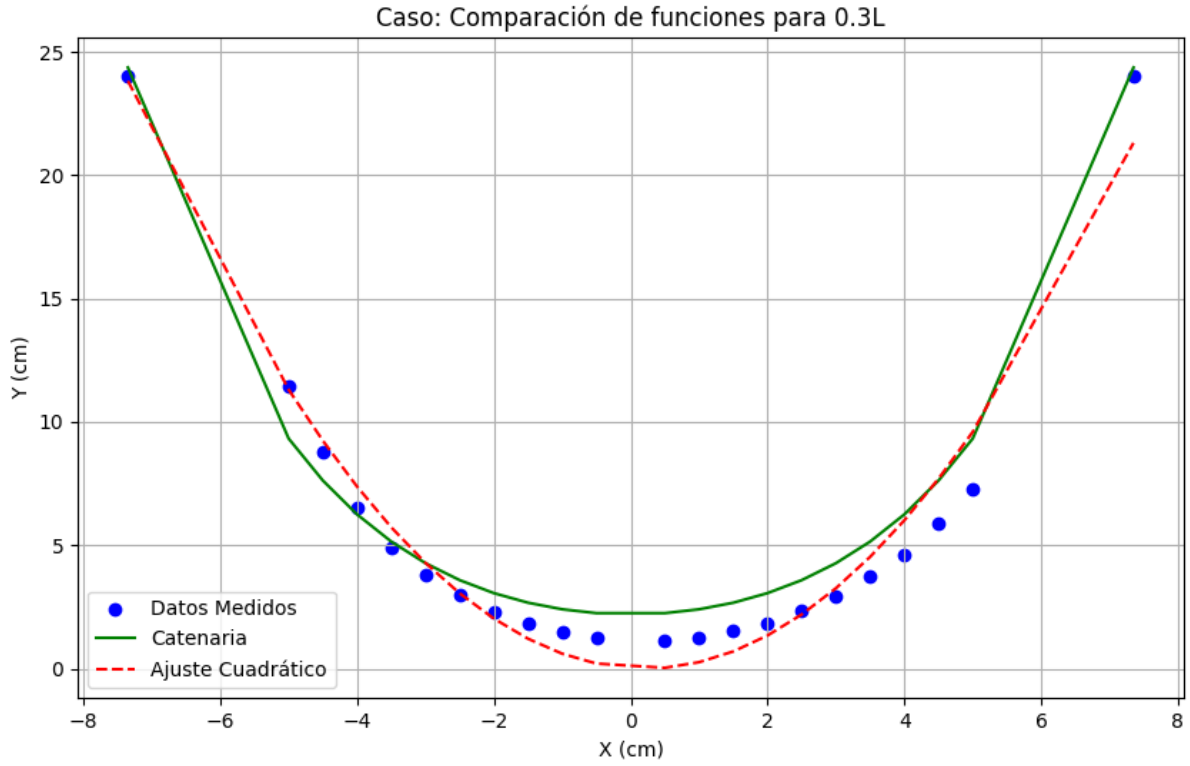



Figura 4: Comparación de estimaciones para 0.3L

## Conclusiones

- A la hora de realizar el trabajo, surgieron complicaciones en el grupo sobre cómo relacionar lo visto en clase con el problema, y aún más, sobre cómo plasmarlo en el código.
- Otro conflicto fue el hecho de encontrar $\mu$. Si tenía sentido físico, de qué ecuación corresponde mejor, entre otros pensamientos.
- Se aprendió que algunas veces resulta más preciso transformar el problema original a uno más equivalente, como ocurrió con $\mu$.
- Se lograron profundizar los temas de las unidades 1, 2 y 3
- Hacer el mismo análisis a mano resulta más engorroso, y gracias a las herramientas de la computadora se pueden realizar gráficos para visualizar mejor los conceptos.

## Anexo I

**primera-parte.py**

```
import math
import numpy as np
from matplotlib import pyplot as plt

#LA PARTE A) DE ESTA PRIMERA PARTE FUE REALIZADA EN HOJA. LEER INFORME PARA MÁS DETALLES.

#--------------------------- ITEM (B) ---------------------------
#Mediciones EXACTAS
P1 = 106716
P2 = 106031
P = (P1 + P2) / 2
x0 = -P/3500
x1 = P/3500
y0 = y1 = P/2100
L = P/1300

#Por Newton-Raphson, hallamos mu igualando la ecuación 4) a 0 (ver anexo), dado que en este caso se cumplen las condiciones
def f(mu):
    return ((2*np.sinh(mu*x1)) / mu) - L

# Derivada de f(mu)
def df(mu):
    return ((2*mu*x1*np.cosh(mu*x1)) - (2*np.sinh(mu*x1))) / mu**2

tolerancia = 0.5e-16 #el menor error de truncamiento que este ordenador permite debido a su mantisa (16 decimales)

def newton_raphson(semilla):
    x0 = semilla
    x1 = x0 - f(x0) / df(x0)

    while abs(x1-x0) > tolerancia:
```

```
        x0 = x1
        x1 = x0 - f(x0) / df(x0)

    return x1

mu = newton_raphson(0.1)
mu = round(mu, 16) #para redondear a 16 decimales significativos

#Cálculo del C2
C2 = y1*mu - (np.cosh(mu*x1)) #despejando de la ecuación (3) del enunciado

#propagación de errores c2 -> mu tiene error inherente para su cálculo

dc2_mu = y1 - (x1*np.sinh(mu*x1)) #derivada respecto a mu de C2
ec2 = float(f"{(abs(dc2_mu) * abs(tolerancia)):.1g}" ) #cota de C2 ya
redondeada
t = math.ceil(-math.log10(2 * ec2)) #decimales significativos

C2 = round(C2, t)
#para la ecuacion original, mu y C2 como datos de entrada tienen error
inherente
def catenaria(x):
    return ((np.cosh(mu*x))/mu) + C2

def df_catenaria_mu(x):
    return ((x*mu*np.sinh(mu*x)) - np.cosh(mu*x))/mu**2

#y la derivada respecto de C2 es 1 -> se propaga su error como es de entrada
y_x_0 = catenaria(0)
e_y = (abs(df_catenaria_mu(0)) * tolerancia) + 1*ec2
t = math.ceil(-math.log10(2 * e_y))
y_x_0 = round(y_x_0, t)
print("------------- ITEM B) -------------")
print("Valor de mu: ", mu)
print("Valor de C2: ", C2)
print("Valor de y(x=0): ", y_x_0)
```

```
#-------------------------- ITEM (C) --------------------------
#A priori, los cálculos serían los mismos. Solo se modifican sus errores
print("------------- ITEM C) -------------")
COTA_X0 = COTA_X1 = COTA_Y0 = COTA_Y1 = COTA_L = 0.05

def cota_error_propagado_mu(mu_medido, x1_medido):
    mu_anterior = mu_medido - tolerancia
    dphi_x1 = (((mu_anterior**2)*(np.sinh(2*mu_anterior*x1_medido)))-
(mu_anterior*x1_medido*(L*mu_anterior*np.sinh(mu_anterior*x1_medido)+2))) /
(2*np.sinh(mu_anterior*x1_medido)-
(mu_anterior*x1_medido*np.cosh(mu_anterior*x1_medido**2)))**2
    dphi_l =
(mu_anterior**2)/((2*x1_medido*mu_anterior*np.cosh(mu_anterior*x1_medido))-
(2*np.sinh(x1_medido*mu_anterior)))
    cota = COTA_X1

    return (cota * abs(dphi_x1) + cota * abs(dphi_l))

cota_error_mu = cota_error_propagado_mu(mu, x1) + tolerancia #error total =
error inh + error trunc (se desprecian err redondeo)
t = math.ceil(-math.log10(2 * cota_error_mu)) #decimales significativos
cota_error_mu_red = round(cota_error_mu, t)
print("Cota de error propagado de mu (analítica):", cota_error_mu)
perturbaciones = []
cotas_exp = []
for i in range(1,16):
    delta_mu = 0.1**i
    perturbaciones.append(delta_mu)
    perturbaciones.append(-delta_mu)
    cotas_exp.append(cota_error_propagado_mu(mu+delta_mu, x1))
    cotas_exp.append(cota_error_propagado_mu(mu-delta_mu, x1))

cota_min_exp = np.min(cotas_exp)
perturbaciones = np.ravel(perturbaciones)
cotas_exp = np.ravel(cotas_exp)
plt.figure(figsize=(10, 6))
plt.scatter(perturbaciones, cotas_exp)
```

```
plt.xlabel('Perturbación')
plt.ylabel('Cota mu')
plt.title('Cota de mu, en base a perturbaciones')
plt.grid(True)
plt.show()
print("Cota de error propagado de mu (experimental)", cota_min_exp)
dif_cota_mu = abs(cota_error_mu - cota_min_exp)
print("Diferencia entre cota analítica y experimental:", dif_cota_mu)
mu_redondeado = round(mu, t-1)
print("Valor de mu bien redeondeado:", mu_redondeado)

#c2 = f(mu, x1, y1) = y1 - cosh(mu * x1)/mu

def cota_error_total_c2(mu_medido, x1_medido):
    df_mu = (np.cosh(x1_medido*mu_medido) -
(mu_medido*x1_medido*np.sinh(mu_medido*x1_medido))) / mu_medido**2
    df_x1 = - mu_medido*np.sinh(mu_medido * x1_medido)
    df_y1 = 1
    return abs(cota_error_mu*df_mu) + abs(COTA_X1*df_x1) + abs(COTA_Y1 * df_y1)

cota_error_c2 = cota_error_total_c2(mu_redondeado, x1)
print("Cota de error de C2:",cota_error_c2)
#erc2 = (cota_error_c2/C2)*100 -> Por si se desea obtener error relativo de C2.
Da alrededor de 35%
print("C2 bien redondeado:", round(C2)) #t=0 y es un decimal, por lo que
redondea a 0
print("C2 expresado como intervalo: ", C2, "±", math.ceil(cota_error_c2 * 100)
/ 100)

#el error inherente de mu (dato de entrada para esta ecuación) es el obtenido
anteriormente, por lo que se reutiliza esa variable
def cota_total_error_y(mu_medido, x_medido):
    df_mu = df_catenaria_mu(x_medido)
    df_x = mu_medido*np.sinh(mu_medido * x_medido)
    df_c2 = 1
    return abs(cota_error_mu*df_mu) + abs(cota_error_c2 * df_c2) +
abs(COTA_X1*df_x)
```

```
cota_error_y = cota_total_error_y(mu_redondeado, 0)
print("Cota de error de y(x):", cota_error_y)
print("Valor de y(0) bien redeondeado:", round(y_x_0))
print("El valor expresado como intervalo", y_x_0, "±", round(cota_error_y, 1))
```

**segunda-parte.py**

```
import numpy as np
from matplotlib import pyplot as plt

#----------------- GENERALIDADES -----------------
#Las siguientes medidas y funciones valen para todos los casos del TPM (eg. todos los casos medimos la altura a 24cm)

L = 49   # Longitud de la cadena. Realmente mide 50cm, pero se consideró el espacio perdido en el soporte del primer eslabón de cada lado
n = 22  #Puntos considerando los extremos
y1 = y0 = 24.0 #altura del soporte

# Definimos la función f(mu) basada en la condición de longitud
def f(mu, x1):
    return ((2*np.sinh(mu*x1)) / mu) - L

# Derivada de f(mu)
def df(mu, x1):
    return ((2*mu*x1*np.cosh(mu*x1)) - (2*np.sinh(mu*x1))) / mu**2

# Newton-Raphson loop
tolerancia = 0.5e-16
def newton_raphson(semilla, soporte_derecha):
    mu0 = semilla
    mu1 = mu0 - f(mu0, soporte_derecha) / df(mu0, soporte_derecha)

    while abs(mu1-mu0) > tolerancia:
        mu0 = mu1
        mu1 = mu0 - f(mu0, soporte_derecha) / df(mu0, soporte_derecha)

    return mu1

#ecuación original
def catenaria(x):
    return ((np.cosh((mu_raiz*x)))/mu_raiz + C2)
```

```
#auxiliar para obtener valores de la ecuación original en cada caso
def obtener_soluciones_catenaria(puntos):
    y = []
    for punto in puntos:
        y.append(catenaria(punto))

    return y

#función que muestra ajuste y comparaciones según el caso
def resultados_soluciones(puntos, soluciones_caso, titulo):
    soluciones_catenaria = np.matrix(obtener_soluciones_catenaria(puntos))
    puntos = np.matrix(puntos)

    fi0=np.matrix(np.ones(n))
    fi1=puntos
    fi2=np.power(puntos, 2)
    M=(
        (np.inner(fi0,fi0)), (np.inner(fi0,fi1)), (np.inner(fi0,fi2)),
        (np.inner(fi1,fi0)), (np.inner(fi1,fi1)), (np.inner(fi1,fi2)),
        (np.inner(fi2,fi0)), (np.inner(fi2,fi1)), (np.inner(fi2,fi2))
    )
    M=np.array(M).reshape((3,3))
    M=np.matrix(M).reshape((3,3))

    b=(np.inner(fi0, soluciones_caso), np.inner(fi1, soluciones_caso),
np.inner(fi2, soluciones_caso))
    b=np.array(b).reshape(3,1)
    b=np.matrix(b).reshape(3,1)

    c=np.linalg.inv(M)*b

    soluciones_cuadratica = c[0,0]*fi0+c[1,0]*fi1+c[2,0]*fi2
    dif=soluciones_caso - soluciones_catenaria
    ecm=np.sqrt(np.inner(dif, dif) / n)
    print("Coeficientes del ajuste:", c) #Coeficientes de ajuste cuadrático
c0*fi0 + c1*fi1 + c2*fi2 respectivamente
    print("ECM Catenaria:", ecm[0,0])
```

```
    dif=soluciones_caso - soluciones_cuadratica
    ecm=np.sqrt(np.inner(dif, dif) / n)
    print("ECM Cuadrática:", ecm[0,0])

    puntos = np.ravel(puntos)
    soluciones_catenaria = np.ravel(soluciones_catenaria)
    soluciones_cuadratica = np.ravel(soluciones_cuadratica)

    # Gráficos para comparar ajuste, catenaria y datos medidos
    plt.figure(figsize=(10, 6))
    plt.scatter(puntos, soluciones_caso, color='blue', label='Datos Medidos')
    plt.plot(puntos, soluciones_catenaria, color='green', linestyle='-',
label='Catenaria')
    plt.plot(puntos, soluciones_cuadratica, color='red', linestyle='--',
label='Ajuste Cuadrático')
    plt.xlabel('X (cm)')
    plt.ylabel('Y (cm)')
    plt.title(titulo)
    plt.legend()
    plt.grid(True)
    plt.show()

#----------------- CASO 0.8L -----------------
print("----------------- CASO 0.8L -----------------")
x0 = -19.6
x1 = 19.6

puntos = np.array([x0, -15.0, -13.5, -12.0, -10.5, -9.0, -7.5, -6.0, -4.5, -
3.0, -1.5, 1.5, 3.0, 4.5, 6.0, 7.5, 9.0, 10.5, 12.0, 13.5, 15.0, x1])
soluciones_foto = np.array([
    y0,
    18.864, 17.129, 15.749, 14.508, 13.355, 12.468, 11.722, 11.203, 10.836,
10.659,
    10.635, 10.826, 11.256, 11.775, 12.498, 13.387, 14.313, 15.467, 16.826,
18.397,
    y1
])
```

```
#calculo de mu y c2
mu_raiz = newton_raphson(0.1, x1)
C2 = y1*mu_raiz - (np.cosh(mu_raiz*x1)) #despejando de la ecuación (3)

# Resultados
print("Valor de mu:", mu_raiz)
print("Valor de C2", C2)
resultados_soluciones(puntos, soluciones_foto, 'Caso: Comparación de funciones
para 0.8L')

#CASO 0.7L
print("----------------- CASO 0.7L -----------------")
x0 = -17.15
x1 = 17.15
puntos = np.array([x0, -15.0, -13.5, -12.0, -10.5, -9.0, -7.5, -6.0, -4.5, -
3.0, -1.5, 1.5, 3.0, 4.5, 6.0, 7.5, 9.0, 10.5, 12.0, 13.5, 15.0, x1])
soluciones_foto = np.array([
    y0,
    20.951, 17.932, 15.504, 13.615, 11.892, 10.566, 9.522, 8.775, 8.254, 7.944,
    7.972, 8.311, 8.846, 9.636, 10.623, 11.952, 13.559, 15.560, 17.846, 20.413,
    y1
])

#calculo de mu y c2
mu_raiz = newton_raphson(0.1, x1)
C2 = y1*mu_raiz - (np.cosh(mu_raiz*x1)) #despejando de la ecuación (3)

# Resultados
print("Valor de mu:", mu_raiz)
print("Valor de C2", C2)
resultados_soluciones(puntos, soluciones_foto, 'Caso: Comparación de funciones
para 0.7L')

#----------------- CASO 0.3L -----------------

# Definimos las constantes según nuestro caso del modelo
```

```
print("----------------- CASO 0.3L -----------------")
x0 = -7.35
x1 = 7.35
puntos = np.array([x0, -5.0, -4.5, -4.0, -3.5, -3.0, -2.5, -2.0, -1.5, -1.0, -
0.5, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, x1])
soluciones_foto = np.array([
    y0,
    11.464, 8.760, 6.519, 4.898, 3.809, 2.975, 2.309, 1.815, 1.469, 1.241,
    1.119, 1.278, 1.525, 1.846, 2.343, 2.911, 3.761, 4.593, 5.880, 7.287,
    y1
])

#calculo de mu y c2
mu_raiz = newton_raphson(0.1, x1)
C2 = y1*mu_raiz - (np.cosh(mu_raiz*x1)) #despejando de la ecuación (3)

# Resultados
print("Valor de mu:", mu_raiz)
print("Valor de C2", C2)
resultados_soluciones(puntos, soluciones_foto, 'Caso: Comparación de funciones
para 0.3L')
```

## Anexo II

**primera-parte.py**

```
------------ ITEM B) ------------
  Valor de mu:  0.0452347114575708
  Valor de C2:  0.1877466921834858
Valor de y(x=0):  22.29466343327906
------------ ITEM C) ------------
Cota de error propagado de mu (analítica): 4.912575795925581e-05
Cota de error propagado de mu (experimental) 3.7389579801427036e-06
Diferencia entre cota analítica y experimental: 4.5386799979113107e-05
Valor de mu bien redeondeado: 0.0452
Cota de error de C2: 0.0647017779296913
C2 bien redondeado: 0
C2 expresado como intervalo:  0.1877466921834858 ± 0.07
Cota de error de y(x): 0.08871031044939455
Valor de y(0) bien redeondeado: 22
El valor expresado como intervalo 22.29466343327906 ± 0.1
```

**segunda-parte.py**

```
---------------- CASO 0.8L ----------------
Valor de mu: 0.06034313978532292
Valor de C2 -0.3366139483485733
Coeficientes del ajuste: [[ 1.05275216e+01]
 [-6.20736398e-03]
 [ 3.52954329e-02]]
ECM Catenaria: 5.369972573010284
ECM Cuadrática: 0.09547278207066552
---------------- CASO 0.7L ----------------
Valor de mu: 0.08830641188186822
Valor de C2 -0.26408122686673563
Coeficientes del ajuste: [[ 7.60906662e+00]
 [-3.00032964e-03]
 [ 5.61343595e-02]]
ECM Catenaria: 2.813739754120727
ECM Cuadrática: 0.21835433294394332
---------------- CASO 0.3L ----------------
Valor de mu: 0.4078014209306157
Valor de C2 -0.2538203665716541
Coeficientes del ajuste: [[ 0.01480042]
 [-0.17107422]
 [ 0.41749618]]
ECM Catenaria: 1.180772193276597
ECM Cuadrática: 1.0866821557099977
```

**Anexo III**

a) Considerando $y_0 = y_1$, $x_0 = -x_1$ y $C_1 = 0$:

(1) → $y(x_0) = y_0 = \frac{\cosh(\mu x_0)}{\mu} + C_2$

(2) → $y(x_1) = y_1 = \frac{\cosh(\mu x_1)}{\mu} + C_2$

. Igualando ambas → $\frac{\cosh(\mu x_0)}{\mu} + C_2 = \frac{\cosh(\mu x_1)}{\mu} + C_2$

. Multiplicando miembro a miembro por $\mu$ → $\cosh(\mu x_0) = \cosh(\mu x_1)$

. Reemplazando $x_0 = -x_1$ → $\cosh(\mu x_0) = \cosh(\mu(-x_0))$ ✓

(3) Reescribiendo → $\frac{\sinh(\mu x_1) - \sinh(-\mu x_1)}{\mu} = L$ (idem para (1) y (2))

Como $\sin(-\mu x_1) = -\sinh(\mu x_1)$ → $\frac{\sinh(\mu x_1) + \sinh(\mu x_1)}{\mu} = L$

→ $\frac{2\sinh(\mu x_1)}{\mu} = L$ ✓

Anexo bonus: despejo $\mu$

$\mu = \frac{2\sinh(\mu x_1)}{L}$ → Newton-Raphson para $\mu$

Figura 5: Análisis de a) primera parte

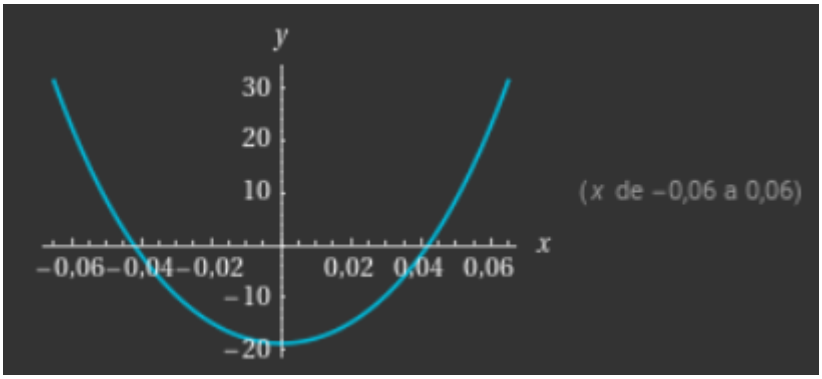



Figura 6: Gráfica de $\frac{2\sinh(\mu x_1)}{\mu} - L$

## Bibliografía


- Facultad de Ingeniería, Universidad de Buenos Aires. *Unidad 1, Módulo 1: Errores en el Cálculo Numérico*
- Facultad de Ingeniería, Universidad de Buenos Aires. *Unidad 1, Módulo 2: Representación de números, errores de redondeo*
- Facultad de Ingeniería, Universidad de Buenos Aires. *Unidad 1, Módulo 3: Gráfica de proceso, condición del problema y término de estabilidad*
- Facultad de Ingeniería, Universidad de Buenos Aires. *Unidad 2, Módulo 2: Métodos de punto fijo y de Newton-Raphson*
- Facultad de Ingeniería, Universidad de Buenos Aires. *Unidad 3, Módulo 1: Ajuste por cuadrados mínimos (lineal y no lineal)*
- González, H. *Análisis Numérico, Primer curso.* Nueva Librería