Trabajo Práctico de Máquina – Versión N°1

Materia: Modelación Numérica.

Código: CB051.

Cuatrimestre: 1er cuatrimestre, 2024.

Docentes: Machiunas, Valeria - Rodríguez, Daniel.

Integrantes:

- Dominguez, Jonathan.
    - Nombre: Jonathan.
    - Apellido: Dominguez.
    - Padrón: 110057.
    - Carrera: Ingeniería en Informática.
    - Código de materia según SIU: CB051.
- Fanti, Gerónimo.
    - Nombre: Gerónimo.
    - Apellido: Fanti.
    - Padrón: 109712.
    - Carrera: Ingeniería en Informática.
    - Código de materia según SIU: CB051.

Número de grupo: Octave 03M.

Introducción

En este trabajo práctico estuvimos trabajando sobre una función catenaria para ver cómo varía la curva dependiendo de lo que hagamos. Hicimos un análisis analítico y experimental en cierto punto de la función y a su vez, la intentamos representar haciendo una maqueta real de la misma. Para luego realizar un código en el lenguaje Octave y evaluar los puntos que medimos de nuestra maqueta en el programa que se hizo para compararlo con la función original y ver el error cuadrático medio.

La primera parte consistió en trabajar la parte analítica y experimental. Mientras que en la segunda parte habla de la comparación de la maqueta y el modelo.

En la parte analítica se probó con 2 casos distintos en las mediciones, uno con error de inherente y otro sin él.

En el caso en el que hubo error inherente se tuvo que hacer los cálculos teniendo en cuenta que teníamos un error de ingreso, el cual al final del proceso se tiene que tener en cuenta en la cota de error.

Mientras que en el otro caso, si bien no hay error inherente en los valores iniciales, hay errores de redondeo en las cuentas. Como por ejemplo, en el método para hallar raíces que también se usó para el caso anterior. Y por ello hay que saber que existe la cota de error para ese método.

A su vez, se hicieron pequeñas perturbaciones para compararlo con el caso del error inherente.

El método para hallar raíces fue el de Newton-Raphson, ya que nos pareció el más adecuado para nuestro caso. Aparte de ser simple, en pocas iteraciones se llega a la raíz.

Mientras que en la segunda parte hicimos 10 mediciones en nuestras dos maqueta para notar como se ve la diferencia entre el gráfico de la catenaria y el de los puntos medidos.

De eso mismo se saca el error cuadrático medio para ver cuánto puede ser la variación entre el punto original y el estimado.

Al final de este trabajo haremos una breve conclusión sobre lo visto.

Primera parte

Comenzaremos verificando el caso particular de $y_0 = y_1$ e $x_0 = -x_1$ y tomando a $C_1 = 0$.

Por lo tanto, de estas ecuaciones:

$$y(x_0) = y_0 = \frac{\cosh(\mu x_0 + C_1)}{\mu} + C_2,\ y(x_1) = y_1 = \frac{\cosh(\mu x_1 + C_1)}{\mu} + C_2$$

Deducimos que por reemplazar e igualar

(1) $$\frac{\cosh(-\mu x_1)}{\mu} + C_2 = \frac{\cosh(\mu x_1)}{\mu} + C_2$$

(2) $$\frac{\cosh(-\mu x_1)}{\mu} = \frac{\cosh(\mu x_1)}{\mu}$$

(3) $$\cosh(-\mu x_1) = \cosh(\mu x_1)$$

Esta igualdad es válida por la propiedad de paridad del coseno, $\cosh(x) = \cosh(-x)$

Y por otro lado la fórmula de L,

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

Se reemplaza,

(2) $$L = \frac{\operatorname{senh}(\mu x_1) - \operatorname{senh}(-\mu x_1)}{\mu}$$

Por propiedad de paridad del seno, $\operatorname{senh}(-x) = -\operatorname{senh}(x)$

(3) $$L = \frac{\operatorname{senh}(\mu x_1) + \operatorname{senh}(\mu x_1)}{\mu}$$

(4) $$L = \frac{2\operatorname{senh}(\mu x_1)}{\mu}$$

Dándonos como resultado esta última fórmula.

Como siguiente ejemplo, vamos a usar los siguientes valores:

- P = (110057 + 109712) /2 = 109884,5
- x_0 = -P / 3500 = -31,4 y x_1 = 31,4
- $y_0 = y_1$ = P / 2100 = 52,3
- L = P / 1300 = 84,5
- $C_1 = 0$

Asumiendo que todos los valores son exactos no habría cota de error para el ingreso.

Partimos de esta ecuación: $L = \frac{2\operatorname{senh}(\mu x_1)}{\mu}$, $\mu \neq 0$

Como habíamos demostrado arriba, debido a $x_0 = -x_1$, llegamos a esta expresión.

Reemplazamos los valores de L, x_1 y C_1, para hallar μ.

(1) $$84,5 = \frac{2senh(31,4\mu)}{\mu}$$

(2) $$0 = \frac{2senh(31,4\mu)}{\mu} - 84,5$$

De esta manera obtenemos la expresión de F(μ) = 0. Y esto nos permite el uso de un método para hallar raíces. En nuestro caso vamos a usar Newton – Raphson.

(3) $$F(\mu) = \frac{2senh(31,4\mu)}{\mu} - 84,5$$

Así nos queda definida la F. Y tenemos que usar la fórmula de Newton-Raphson:

(4) $$\mu_{n+1} = \mu_n - \frac{F(\mu_n)}{F'(\mu_n)}$$

(5) $$F'(\mu) = \frac{2(31,4\mu\, cosh(31,4\mu) - senh(31,4\mu))}{\mu^2}, \ \mu \neq 0$$

Que partiendo de μ_0 vamos iterando hasta llegar a nuestra solución.
En nuestro caso tomamos $\mu_0 = 0,1$ ya que vimos gráficamente que está cerca de una raíz y empezamos a iterar.

1ra iteración: $\mu_1 \cong 0,0705559200869$ 2da iteración: $\mu_2 \cong 0,0515556306013$
3ra iteración: $\mu_3 \cong 0,0445818694224$ 4ta iteración: $\mu_4 \cong 0,0437584299992$
5ta iteración: $\mu_5 \cong 0,0437477876854$ 6ta iteración: $\mu_6 \cong 0,0437477859265$
7ma iteración: $\mu_7 \cong 0,0437477859265$

Como la 6ta y 7ma iteración nos dieron los mismos decimales, damos por hecho que en $0,0437477859265$ tenemos una raíz.
A su vez, $-0,0437477859265$ también es raíz debido a que estamos trabajando con un seno.

Nuestra cota de error de NR para los errores de truncamiento es:
$|\alpha - \mu_7| \leq |\mu_7 - \mu_6| \cong 0$

Esto quiere decir que los errores del NR son únicamente de redondeo.
No podemos expresar el μ bien redondeado ya que no calculamos los errores de redondeo.
Por esto mismo decimos que nuestro μ vale $0,0437477859265$ y que no podemos dar por seguro que ese sea su valor por lo mencionado.

Para hallar el valor de C_2 despejo y reemplazo en la ecuación: $y_1 = \frac{cosh(\mu x_1 + C_1)}{\mu} + C_2$

(1) $$C_2 = y_1 - \frac{cosh(\mu x_1 + C_1)}{\mu}$$

(2) $$C_2 = 52,3 - \frac{cosh((0,0437477859265)(31,4))}{0,0437477859265}$$

(3) $$C_2 \cong 4,2628855657535$$

No podemos expresar a C_2 bien redondeado ni tampoco hallar su cota de error, dado que no tenemos el error de μ. Y en base a esto no podemos ver la propagación del error ni asegurar la certeza del mismo.

Y con esto procedemos a buscar el valor de y cuando x = 0.

$$y(0) = \frac{cosh\,(0)}{0,0437477859265} + 4,2628855657535 \cong 27,1211852218798$$

Donde tampoco podemos expresar a y bien redondeado ni hallar su cota de error, por el mismo tema del μ y ahora también el de C_2.

Ahora continuaremos con otro ejemplo donde vamos a trabajar con las mismas mediciones de arriba, pero vamos a tomarlas como que están bien redondeadas a un decimal.
Dándonos así una cota de errores inherentes de: $0,5\,.\,10^{-1}$.

Y vamos a tener que hallar la derivada de nuestra función, $y(x) = \frac{cosh\,(\mu x)}{\mu} + C_2$
pero la vamos a hacer depender de $(x,\ \mu,\ C_2)$ debido a que necesitamos ver la propagación de los errores.

La forma de hallar el error de y es:

$$\left|\varepsilon_y\right| \leq \left|y'_x(x, \mu, C_2)\right| . \left|\varepsilon_x\right| + \left|y'_\mu(x, \mu, C_2)\right| . \left|\varepsilon_\mu\right| + \left|y'_{C_2}(x, \mu, C_2)\right| . \left|\varepsilon_{C_2}\right|$$

Sabemos que $\left|\varepsilon_x\right| = 0,5\,.\,10^{-1}$, por lo que nos resta $\left|\varepsilon_\mu\right|$ y $\left|\varepsilon_{C_2}\right|$.
Más abajo definiremos las derivadas con respecto a cada variable.

El error de μ va a estar compuesto por el inherente, el de truncamiento y el redondeo. Sabemos la existencia del error de redondeo, pero como no fue calculado. Entonces nuestro error de μ es: $\left|\varepsilon_{\mu}\right| = \left|\varepsilon_{\mu - ix,\, iL}\right| + \left|\varepsilon_{\mu-NR}\right|$, donde $\left|\varepsilon_{\mu-NR}\right|$ es la cota de NR.

Entonces para hallar el $\left|\varepsilon_{\mu-ix,\, iL}\right|$ debemos usar la misma fórmula de NR cambiando la $F(\mu) = 0$ para que dependa de $(x,\ \mu,\ L)$.

Defino $U(x,\ \mu,\ L) = \mu - \frac{F(x,\mu,L)}{F'_{\mu}(x,\mu,L)}$.

Donde $F(x,\ \mu,\ L) = \frac{2\, senh(\mu\, x)}{\mu} - L$ y $F'_{\mu}(x,\ \mu,\ L) = \frac{2(x\mu\, cosh(x\mu) - senh(x\mu))}{\mu^2}$

De este modo nos queda, $U(x,\ \mu,\ L) = \mu - \frac{\frac{2senh(x\mu)}{\mu} - L}{\frac{2(x\mu cosh(x\mu) - senh(x\mu))}{\mu^2}} = \mu - \frac{2\mu senh(x\mu) - L\mu^2}{2(x\mu cosh(x\mu) - senh(x\mu))}$

Procedemos a buscar sus derivadas.

$$U'_{x}(x,\ \mu,\ L) = \frac{2\mu^2 senh(x\mu)cosh(x\mu) + 2\mu^3 senh(x\mu)^2 - 2\mu^3 cos(x\mu)^2 - L\mu^4 senh(x\mu)}{2\mu^2 x^2 cosh(x\mu)^2 - 4\mu x senh(x\mu)cosh(x\mu) + 2senh(x\mu)^2}$$

$$U'_{L}(x,\ \mu,\ L) = \frac{\mu^2}{-2senh(x\mu) + 2\mu x cosh(x\mu)}$$

Y la forma de hallar el error es:

$$\left|\varepsilon_{\mu - ix,\, iL}\right| = \left|\varepsilon_{F}\right| \leq \left|U'_{x}(x,\mu,L)\right| .\left|\varepsilon_{ix}\right| + \left|U'_{L}(x,\mu,L)\right| .\left|\varepsilon_{iL}\right|$$

Donde $\left|\varepsilon_{iL}\right| = \left|\varepsilon_{ix}\right| = \left|\varepsilon_{iy}\right| = 0,5\ .\ 10^{-1}$, ya que están bien redondeados a un decimal.

Evaluamos todo en el punto $(31.4,\ 0.0437477859265, 84.5)$ y nos queda que:

$$\left|\varepsilon_{\mu-ix,iL}\right| \leq \left|U'_{x}(31.4,\ 0.0437477859265, 84.5)\right| .0,5\ .\ 10^{-1} + 0,5\ .\ 10^{-1} \cong 0,00024 \leq 0,5.10^{-3}$$

Por lo tanto $\left|\varepsilon_{\mu}\right| \leq \left|\varepsilon_{\mu-ix,iL}\right| + \left|\varepsilon_{\mu-NR}\right| \leq 0,5.10^{-3} + 0 \leq 0,5.10^{-3}$.

Continuamos con $\left|\varepsilon_{C_2}\right|$ donde de la función de y despejamos C_2 tal que, $C_2 = y - \frac{cosh\,(\mu x)}{\mu}$

Definimos C_2 como una función de $h(x,\ y,\mu) = y - \frac{cosh(\mu x)}{\mu}$

Sus derivadas son:

$$h'_{x}(x,\ y,\mu) = senh(\mu x)$$

$$h'_{y}(x,\ y,\mu) = 1$$

$$h'_{\mu}(x,\ y, \mu)\ = - \frac{x\mu senh(x\mu)\ - cosh(x\mu)}{\mu^2}$$

Donde el $\left|\varepsilon_{C_2}\right| \leq \left|h'_x(x,\ y, \mu)\right| . \left|\varepsilon_{ix}\right| + \left|h'_y(x,\ y, \mu)\right| . \left|\varepsilon_{iy}\right|\ + \left|h'_{\mu}(x,\ y, \mu)\right| . \left|\varepsilon_{\mu}\right|$.

Evaluamos en el punto $(31.4,\ 52.3, 0.0437477859265)$

$\left|\varepsilon_{C_2}\right| \leq \left|h'_x(x,\ y, \mu)\right| . \left|\varepsilon_{ix}\right| + \left|h'_y(x,\ y, \mu)\right| . \left|\varepsilon_{iy}\right|\ + \left|h'_{\mu}(x,\ y, \mu)\right| . \left|\varepsilon_{\mu}\right| \cong 0,258\ \leq\ 0,5$

Ahora en $\left|\varepsilon_y\right|$ evaluamos el punto $(31.4, 0.0437477859265, 4.2628855657535)$.

Por ende, vamos a definir las derivadas de y:

$y'_x(x, \mu, C_2) = -\ senh(\mu x)$ $\quad$ $y'_{C_2}(x, \mu, C_2) = 1$ $\quad$ $y'_{\mu}(x, \mu, C_2) = \frac{x\mu senh(x\mu)\ - cosh(x\mu)}{\mu^2}$

Dando así:

$\left|\varepsilon_y\right| \leq \left|y'_x(x, \mu, C_2)\right| . 0,5\ .\ 10^{-1} + \left|y'_{\mu}(x, \mu, C_2)\right| . 0,5. 10^{-3}\ +\ 0,5 \cong 0.7\ \leq\ 1$

Esto es lo que nos da el error de y sin considerar el existente y no calculado error de redondeo. Para recordar tanto los valores de μ, C_2 e y, no se pueden expresar bien redondeados ni afirmar que esos son los valores reales, por todo lo que mencionamos previamente.

Como siguiente experimento sobre este tema vamos estimar el valor de μ de manera experimental, perturbando las constantes x_1 y L. Para esto primero se dejará fija una constante y se irá perturbando la otra de la siguiente forma:

1. x_1 fijo y $L\ +\ 0.1$
2. x_1 fijo y $L\ +\ 0.01$
3. x_1 fijo y $L\ +\ 0.001$
4. $x_1\ +\ 0.1$ y L fijo
5. $x_1\ +\ 0.01$ y L fijo
6. $x_1\ +\ 0.001$ y L fijo

De esta forma se podrá observar qué constante influye más en la cota de error final.
Usando nuestro código en octave, al reemplazar la función con los nuevos valores los resultados en bruto son los siguientes:

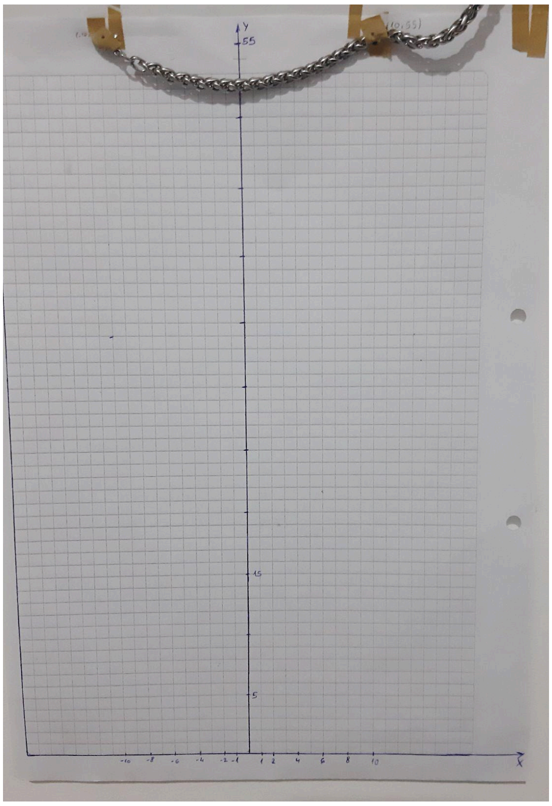
1. 0.04336154513500586
2. 0.04370906625489250
3. 0.04374391299840864

4. 0.04383593119233992
5. 0.04375312665615690
6. 0.04374483448283142

Por lo tanto al hacer este análisis experimental,concluimos que μ tienen una cota de error de $0,5 . 10^{-9}$, cota obtenida de restar el mayor y el menor valor de la raíces obtenidas al perturbar x_1 y L.

Segunda parte

A continuación vamos a mostrar nuestras dos maquetas que armamos para hacer nuestra comparación.



Maqueta 1, tamaño de la cadena: 21,00 unidades

Maqueta 2, tamaño de la cadena: 100,62 unidades

En la 1ra maqueta usamos los siguientes valores:

- x1 = 10.00
- y1 = 55.00
- L = 21.00

Y en la 2da usamos:

- x1 = 10.00
- y1 = 55.00
- L = 100.62

Para medir las ordenadas de los puntos se utilizó el software ImageJ, y se tomó como unidad arbitraria medio centímetro.

Los resultados de las mediciones de los 10 puntos son los siguientes (x, y):

{(-8, 54.009), (-6, 53.202), (-4, 52.680), (-2, 52.334), (-1, 52.298), (1, 52.256), (2, 52.379), (4, 52.673), (6, 53.202), (8, 54.061)}.

Para las distancia de los valores de x, procuramos que recorran desde x_0 hasta x_1 de manera equidistante (exceptuando en -1 y 1) y siempre en valores enteros para facilitar las mediciones. De esta manera se mide bien las posición de cada punto de la cadena, y no se evalúa solamente una parte como si pudiese ser si se tomaran todos los valores desde 0 a 10.

Como cota de error para las mediciones tomamos 2 unidades, $\Delta m = 2$, ya que debemos abarcar errores provenientes del instrumento de medida (regla milimetrada), errores del operador, de la longitud de la cadena, de la perspectiva de la imagen, de los puntos en donde se pegaron los extremos, del grosor de la cadena (se evaluó a ojo en el centro de la cadena) entre otros errores menores, además que se nota que hay un error importante en que la parte inferior de la cadena está desalineada del eje y.

Ahora vamos a presentar la comparación de la catenaria y nuestros puntos aproximados en un gráfico hecho dentro del mismo octave.
Para aclarar, el color azul hace referencia a la catenaria en los puntos mientras que el otro color representa nuestros puntos estimados de forma unida.

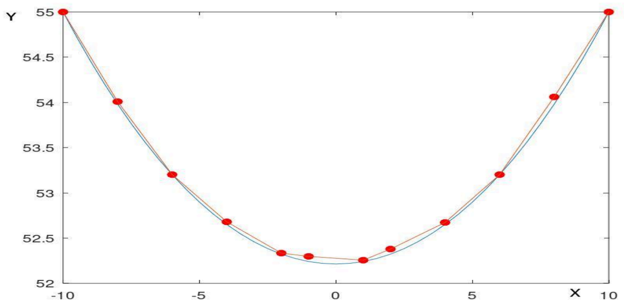


Gráfico 1, representa la catenaria con nuestra maqueta 1.

El error cuadrático medio en este ejemplo es: 0,035625

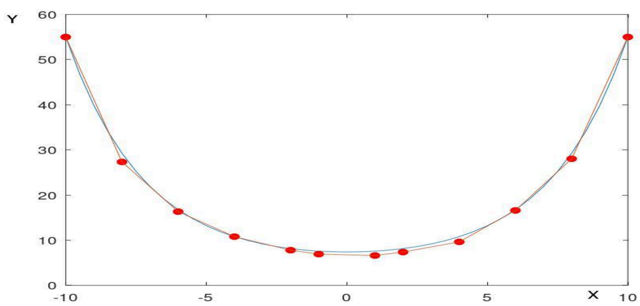


Gráfico 2, representa la catenaria con nuestra maqueta 2.

El error cuadrático medio en este ejemplo es: 0.85872

Conclusión

Como pudimos notar tanto los errores como los números no podían expresarse correctamente. Porque como ya mencionamos al no calcular pero considerar el error de redondeo estamos frente a una situación donde no podemos asumir que dichos valores son los reales y exactos. Por esto mismo al hacer las cuentas siempre se tiene en cuenta esta falta del error de redondeo.

Y por otro lado, si bien las comparaciones de los gráficos no son exactas, se ve que la maqueta llevada al programa es muy similar a la original. Es probable que con otro tipo de maqueta, por ejemplo usando en vez de una cadena un hilo podríamos haber reducido considerablemente el error de medición e inclusive se podría haber acercado más a la catenaria. Que a su vez el error cuadrático iba a ser menor.

Como opinión para cerrar, si uno quiere que la maqueta se aproxime al modelo lo mejor posible es necesario hacerla lo más precisa posible, usando el material adecuado y probablemente usando herramientas de medición más exactas más allá de una regla. Y siempre hay que tener en cuenta todos los posibles errores existentes.

Anexo I

En esta parte adjuntamos todos los scripts de octave con los que trabajamos.

Archivo main_item_b.m

```
source("funciones.m");
%Defino las variables
L = 84.50;
x1 = 31.40;
y1 = 52.30;
c1 = 0;
u0 = 0.1;
max_iter = 1000;
tolerancia = 1e-14;

[u_final, error_u] = newton_raphson(u0,x1, L, f, df_du, tolerancia, max_iter);
disp(['Valor u: ', num2str(u_final, formatear(error_u))]);
disp(['Cota de error u: ', num2str(error_u)]);
disp('');

error_u_acot = calcular_mu(error_u);
u_red = redondear_numero(u_final, error_u_acot);
disp(['Valor u red: ', num2str(u_red, formatear(error_u_acot*10))]);
disp(['Cota de error u acot: ', num2str(error_u_acot)]);
disp('');

c2 = f_c2(u_red, x1, c1, y1);
error_c2 = abs(df_c2_du(u_red, x1, c1, y1)) * error_u_acot;
disp(['C2: ', num2str(c2, formatear(error_c2*10))]);
disp(['Cota de error C2: ', num2str(error_c2)]);
disp('');

error_c2_acot = calcular_mu(error_c2);
c2_red = redondear_numero(c2, error_c2_acot);
disp(['C2 red: ', num2str(c2_red, formatear(error_c2_acot*10))]);
disp(['Cota de error C2 acot: ', num2str(error_c2_acot)]);
disp('');

x_en_cero = 0;

y = f_y(u_red, x_en_cero, c1, c2);
error_y = abs(df_y_du(u_red, x_en_cero, c1, c2)) * error_u_acot + abs(df_y_dc2(u_red,
x_en_cero, c1, c2)) * error_c2_acot;
disp(['Y en X = 0: ', num2str(y, formatear(error_y*10))]);
disp(['Cota de error Y en X = 0: ', num2str(error_y)]);
disp('');

error_y_acot = calcular_mu(error_y);
y_red = redondear_numero(y, error_y_acot);
disp(['Y en X = 0 red: ', num2str(y_red, formatear(error_y_acot*10))]);
disp(['Cota de error Y en X = 0 acot: ', num2str(error_y_acot)]);
```

Archivo redondear_numero.m

```
function num_redondeado = redondear_numero(num, cota)
    % Formatear el número con la cantidad de decimales de la cota,
    % se multiplica por 10 para que el mu, me defina bien la cantidad de decimales
    num_formateado = sprintf(formatear(cota*10), num);

    % Convertir el número formateado nuevamente a un número
    num_redondeado = str2double(num_formateado);
end
```

Archivo newton_raphson.m

```
%{
f: función para la cual se busca la raíz
df: derivada de la función
u: estimación inicial
max_iter: número máximo de iteraciones
```

```
tolerancia: criterio de convergencia
%}
function [raiz, error] = newton_raphson(u, x, L, f, df, tolerancia, max_iter)
    u1 = u - f(u, x, L) / df(u, x, L);
    error = abs(u1 - u);
    if error < tolerancia || max_iter == 0
        raiz = u1;
        return;
    end
    [raiz, error] = newton_raphson(u1, x, L, f, df, tolerancia, max_iter-1);
end
```

Archivo main_parte_2_grafico_1.m

```
source("funciones.m");
%Defino las variables
L = 21.00;
x1 = 10.00;
y1 = 55.00;
c1 = 0;
u0 = 0.1;
max_iter = 1000;
tolerancia = 1e-14;
x = -x1 : 0.5 : x1;
puntos_x = [-10, -8, -6, -4, -2, -1, 1, 2, 4, 6, 8, 10];
puntos_y = [55.000, 54.009, 53.202, 52.680, 52.334, 52.298, 52.256, 52.379, 52.673,
53.202, 54.061, 55.000];

[u_final, error_u] = newton_raphson(u0,x1, L, f, df_du, tolerancia, max_iter);
error_u_acot = calcular_mu(error_u);

u_red = redondear_numero(u_final, error_u_acot);
c2 = f_c2(u_red, x1, c1, y1);

error_c2 = abs(df_c2_du(u_red, x1, c1, y1)) * error_u_acot;
error_c2_acot = calcular_mu(error_c2);

c2_red = redondear_numero(c2, error_c2_acot);

y = f_y(u_red, x, c1, c2_red);
plot(x, y, '-');
hold on;

plot(puntos_x, puntos_y, 'ro', 'MarkerFaceColor', 'r');
hold on;

plot(puntos_x, puntos_y);
hold off;

ecm = calcular_ecm(f_y, puntos_x, puntos_y, u_red, c1, c2_red);
disp(['Error cuadrático medio: ', num2str(ecm)]);
```

Archivo main_parte_2_grafico_2.m

```
source("funciones.m");
%Defino las variables
L = 100.62;
x1 = 10.00;
y1 = 55.00;
c1 = 0;
u0 = 0.1;
max_iter = 1000;
tolerancia = 1e-14;
x = -x1 : 0.5 : x1;
puntos_x = [-10, -8, -6, -4, -2, -1, 1, 2, 4, 6, 8, 10];
puntos_y = [55.000, 27.342, 16.332, 10.796, 7.792, 6.921, 6.616, 7.376, 9.637, 16.597,
28.042, 55.000];

[u_final, error_u] = newton_raphson(u0,x1, L, f, df_du, tolerancia, max_iter);
error_u_acot = calcular_mu(error_u);

u_red = redondear_numero(u_final, error_u_acot);
```

```
c2 = f_c2(u_red, x1, c1, y1);

error_c2 = abs(df_c2_du(u_red, x1, c1, y1)) * error_u_acot;
error_c2_acot = calcular_mu(error_c2);

c2_red = redondear_numero(c2, error_c2_acot);

y = f_y(u_red, x, c1, c2_red);
plot(x, y, '-');
hold on;

plot(puntos_x, puntos_y, 'ro', 'MarkerFaceColor', 'r');
hold on;

plot(puntos_x, puntos_y);
hold off;

ecm = calcular_ecm(f_y, puntos_x, puntos_y, u_red, c1, c2_red);
disp(['Error cuadrático medio: ', num2str(ecm)]);
```

Archivo formatear.m

```
function formato = formatear(num)
  formato = ['%.', num2str(calcular_cant_decimales(num)), 'f'];
end
```

Archivo calcular_cant_decimales.m

```
function cant_decimales = calcular_cant_decimales(num)
    str_num = num2str(num);

    esta_en_notacion = strfind(str_num, "e");

    if esta_en_notacion != 0
      indice_signo = esta_en_notacion +1;
      if length(str_num) - indice_signo == 2
         cant_decimales = str2num([str_num(length(str_num)-1),
str_num(length(str_num))]);
      else
         cant_decimales = str2num(str_num(length(str_num)));
      end
    else
      indice_punto = strfind(str_num, ".");
      cant_decimales = length(str_num) - indice_punto;
    end
end
```

Archivo funciones.m

```
%Funcion a la que le buscamos la raiz u
f = @(u, x, L) ((2*sinh(u*x))/u - L);
%Funcion derivada de f con respecto a u
df_du = @(u, x, L) (2*(x*u*cosh(u*x)-sinh(u*x))/u^2);

%Funcion para hallar el C2
f_c2 = @(u, x, c1, y) (y - cosh(u * x + c1)/u);
%Funcion derivada de C2 con respecto a u
df_c2_du = @(u, x, c1, y) (- (x*u*sinh(x*u + c1) - cosh(x*u + c1))/u^2);

%Funcion caternaria
f_y = @(u, x, c1, c2) (cosh(u*x+ c1)/u + c2);
%Funcion derivada caternaria con respecto a u
df_y_du = @(u, x, c1,c2) (-(x*u*sinh(x*u + c1) - cosh(x*u + c1))/u^2);
%Funcion derivada caternaria con respecto a C2
df_y_dc2 = @(u, x, c1, c2) (1);

%Funcion del error cuadratico medio
function ecm = calcular_ecm(f, x_medido, y_estimado, u, c1, c2)
  y_absoluto = f(u, x_medido, c1, c2);

  % Inicializar la suma de los cuadrados de las diferencias
  suma_cuadrados_diff = 0;
  % Calculo la suma de los cuadrados de las diferencias
```

```
  for i = 1:length(y_absoluto)
      suma_cuadrados_diff = suma_cuadrados_diff + (y_absoluto(i) - y_estimado(i))^ 2;
  end

  ecm = sqrt(suma_cuadrados_diff / length(y_absoluto));
end
```

Archivo main_item_c.m

```
source("funciones.m");
%Defino las variables
L = 84.50;
x1 = 31.40;
valores_pertubados = [0.1, 0.01, 0.001];
y1 = 52.30;
c1 = 0;
u0 = 0.1;
max_iter = 1000;
tolerancia = 1e-14;
error_u_acot = 0;
resta_num_mayor_menor = 0;

disp('Valor de la raiz haciendo distintas perturbaciones en x1 y L');
disp('');

disp('Pertubaciones solo en x');
disp('');

for i = 1:length(valores_pertubados)
  x = x1 + valores_pertubados(i);
  [u_final, error_u] = newton_raphson(u0, x, L, f, df_du, tolerancia, max_iter);
  disp(['Valor actual de x1: ', num2str(x)]);
  disp(['Valor actual de L: ', num2str(L)]);
  disp(['Valor u: ', num2str(u_final, formatear(error_u))]);
  disp(['Cota de error u: ', num2str(error_u)]);
  disp('');
end

disp('Pertubaciones solo en L');
disp('');

for i = 1:length(valores_pertubados)
  l_p = L + valores_pertubados(i);
  [u_final, error_u] = newton_raphson(u0, x, l_p, f, df_du, tolerancia, max_iter);
  disp(['Valor actual de x1: ', num2str(x1)]);
  disp(['Valor actual de L: ', num2str(l_p)]);
  disp(['Valor u: ', num2str(u_final, formatear(error_u))]);
  disp(['Cota de error u: ', num2str(error_u)]);
  disp('');
end

resta_num_mayor_menor = 0.04383593119233992 - 0.04336154513500586;
error_u_acot = calcular_mu(resta_num_mayor_menor);
disp(['El error que tiene el calculo de la raiz mediante este medodo es: ',
num2str(error_u_acot*10)]);
```

Archivo calcular_mu.m

```
function mu = calcular_mu(cota)

    % Convierte el número a una cadena de caracteres
    str_num = num2str(cota);

    primer_decimal = 0;
    cant_decimales_cota = 0;

    % Se verifica si esta en notacion cientifica
    esta_en_notacion = strfind(str_num, "e");

    if esta_en_notacion == 0
```

```
        % Encuentra el índice del punto decimal
        indice_punto = strfind(str_num, '.');

        % Encuentra el primer dígito diferente de cero después del punto decimal
        for i = (indice_punto + 1):length(str_num)
            if str_num(i) ~= '0'
                primer_decimal = str2num(str_num(i));
                cant_decimales_cota = i - indice_punto;
                break;
            end
        end
    else
        primer_decimal = str2num(str_num(1));
        indice_signo = esta_en_notacion +1;
        if length(str_num) - indice_signo == 2
           cant_decimales_cota = str2num([str_num(length(str_num)-1),
str_num(length(str_num))]);
        else
           cant_decimales_cota = str2num(str_num(length(str_num)));
        end
    end

    % Calcula k según el primer decimal
    if primer_decimal < 5
        k = cant_decimales_cota;
    else
        k = cant_decimales_cota -1;
    end

    % Calcula mu
    mu = 5 * 10^(-k);
end
```

Anexo II

Resultados numéricos obtenidos al correr el programa.

main_item_b

```
Valor u: 0.04374778592650812
Cota de error u: 4.1633e-17

Valor u red: 0.0437477859265081
Cota de error u acot: 5e-17

C2: 4.2628855657517
Cota de error C2: 1.143e-14

C2 red: 4.2628855657517
Cota de error C2 acot: 5e-14

Y en X = 0: 27.1211852218737
Cota de error Y en X = 0: 7.6125e-14

Y en X = 0 red: 27.121185221874
Cota de error Y en X = 0 acot: 5e-13
```

main_item_c

```
Valor de la raíz haciendo distintas perturbaciones en x1 y L

Perturbaciones solo en x

Valor actual de x1: 31.5
Valor actual de L: 84.5
Valor u: 0.04336154513500586
Cota de error u: 9.0206e-17

Valor actual de x1: 31.41
Valor actual de L: 84.5
Valor u: 0.04370906625489250
Cota de error u: 6.245e-17

Valor actual de x1: 31.401
Valor actual de L: 84.5
Valor u: 0.04374391299840864
Cota de error u: 4.1633e-17

Perturbaciones solo en L

Valor actual de x1: 31.4
Valor actual de L: 84.6
Valor u: 0.04383593119233992
Cota de error u: 4.1633e-17

Valor actual de x1: 31.4
Valor actual de L: 84.51
Valor u: 0.04375312665615690
Cota de error u: 5.5511e-17

Valor actual de x1: 31.4
Valor actual de L: 84.501
Valor u: 0.04374483448283142
Cota de error u: 6.245e-17

El error que tiene el cálculo de la raíz mediante este método es: 5e-08.
```

main_parte_2_grafico_1

```
Error cuadrático medio: 0.3807
```

main_parte_2_grafico_2

```
Error cuadrático medio: 2.8838
```