OBLIGATORIO 2

MÉTODOS NUMÉRICOS

Curso 2017.

DMEL, CenUR LN

Universidad de la Republica

UdelaR.

Profesor: José Vieitez

Sebastián Castro

Grupo:

- Juan Ferrand
- Luciano Acosta

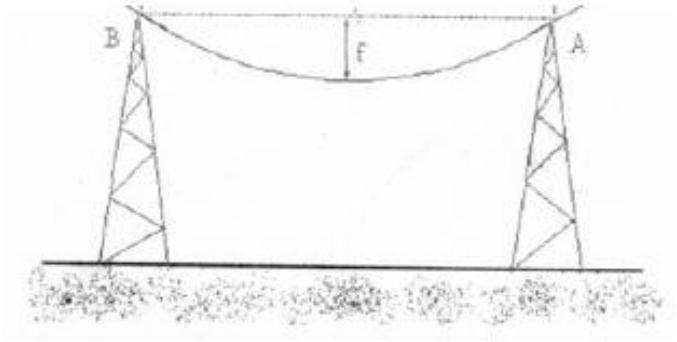
Contenido

Descripción del Problema

Un cable homogéneo entre dos puntos a igual altura sometido a su propio peso y separados una distancia D adopta la forma (aproximada) de una catenaria, i.e.: la función coseno hiperbólico:

$$y(x) = a\cosh\left(\frac{x}{a}\right) = a * \left(\frac{e^{x/a} + e^{-x/a}}{2}\right)$$

Con $-D/2 \leq X \leq D/2$



Ejercicio N° 1:

Probar que la longitud del cable es L= $2a * \sinh\left(\frac{D}{2a}\right)$, siendo $\sinh\left(\frac{x}{a}\right) = \left(\frac{e^{x/a} - e^{-x/a}}{2}\right)$ el seno hiperbólico de x/a.

Para hallar la longitud de la curva utilizamos:

$$\int_a^b \sqrt{1 + (f'(x))^2}$$

Con f: [a, b]→$\mathscr{R}$

Nuestra función está definida en el intervalo $[-D/2\ , D/2] \rightarrow \mathscr{R}$

$$y'(x) = a * \sinh\left(\frac{x}{a}\right) * \frac{1}{a} = \sinh\left(\frac{x}{a}\right)$$

Sustituyendo y' en la ecuación anterior nos queda:

$$L = \int_{-D/2}^{D/2} \sqrt{1 + \left(\sinh\left(\frac{x}{a}\right)\right)^2} = \int_{-D/2}^{D/2} \sqrt{1 + sinh^2(x/a)}$$

Utilizando que:

$$cosh^2(x/a) - sinh^2(x/a) = 1 \rightarrow 1 + sinh^2(x/a) = cosh^2(x/a)$$

Sustituyendo $cosh^2(x/a)$ por $1 + sinh^2(x/a)$ en la integral anterior tenemos:

$$\int_{-D/2}^{D/2} \sqrt{cosh^2(x/a)}$$

$$\rightarrow \int_{-D/2}^{D/2} \sqrt{cosh^2(x/a)} = \int_{-D/2}^{D/2} cosh(x/a) = a * sinh(x/a) \begin{cases} D/2 \\ -D/2 \end{cases}$$

$$= a * \sinh\left(\frac{D}{2a}\right) - a * \sinh\left(\frac{-D}{2a}\right) = a * \sinh\left(\frac{D}{2a}\right) + a * \sinh\left(\frac{D}{2a}\right)$$

$$\rightarrow \text{L} = 2a * \sinh\left(\frac{D}{2a}\right)$$

Ejercicio N° 2:

Se llama flecha f de la catenaria, a la diferencia entre el punto más alto del cable (donde está sujeto el cable a los postes) y el punto más bajo del cable (la “barriga" del cable que corresponde al punto medio entre los postes).

Mostrar que $f = a * (\cosh(D/2a) - 1)$.

Para calcular el valor de la flecha f, necesitamos hallar el punto en el que la derivada de la función y(x) (y’(x)) vale 0, debido que en este punto se encuentra el mínimo de la función en el intervalo $[-D/2\ , D/2]$.

Luego necesitamos evaluar la función y(x) en el punto más alto del intervalo para poder calcular la diferencia entre estos dos puntos.

$$y'(x) = \sinh\left(\frac{x}{a}\right) = 0 \Leftrightarrow x = 0 \text{ porque:}$$

$$\sinh(x) = \frac{e^{x/a} - e^{-x/a}}{2}$$

$$e^{x/a} = e^{-x/a} \Leftrightarrow x = 0$$

$$\frac{e^{0/a} - e^{-0/a}}{2} = \frac{e^0 - e^0}{2} = \frac{1-1}{2} = 0$$

$$y(0) = a * \frac{e^0 + e^0}{2} = a * \frac{2}{2} = a$$

Los puntos más altos son los extremos de los intervalos, elegimos x = $D/2$

$$\text{y } (D/2) = a * \cosh\left(\frac{D}{2a}\right)$$

$$\rightarrow \text{y } (D/2) - y(0) = a * \cosh\left(\frac{D}{2a}\right) - a$$

$$\rightarrow f = a * (\cosh(D/2a) - 1).$$

Ejercicio N° 3:

Sea D = 440m y L = 442:5m. Implementar en Octave el método de Newton-Raphson y el de la secante para calcular a con error relativo menor que 10^{-4}.

$$0 = 2a * \sinh\left(\frac{D}{2a}\right) - L$$

$$\rightarrow 0 = 2a * \sinh\left(\frac{440}{2a}\right) - 442.5$$

Método de Newton-Raphson:

El Método de Newton-Raphson (N-R) es utilizado para encontrar aproximaciones de los ceros o raíces de una función de manera iterativa. También puede ser usado para encontrar el máximo o mínimo de una función, encontrando los ceros de su primera derivada.

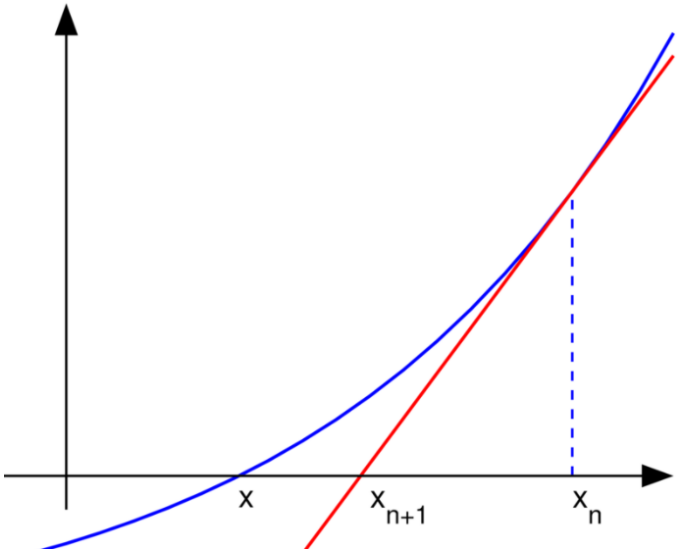


Figura N°1: Representación Gráfica de una iteración del Método de N-R

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$$

$$\rightarrow a_n = a_{n-1} - \frac{f(a_{n-1})}{f'(a_{n-1})}$$

$$f(a) = 2a * \sinh\left(\frac{440}{2a}\right) - 442.5$$

$$f'(a) = 2a * \sinh\left(\frac{440}{2a}\right) - \frac{440\cosh\left(\frac{440}{2a}\right)}{a}$$

Para la implementación de este método se realizó un módulo llamado Newton-Raphson (ver en Anexos) brindándole como condición de "parada" (condición para que el algoritmo deje de iterar, considerando que se consiguió la precisión deseada) la siguiente expresión:

$$\frac{|x_n - x_{n-1}|}{|x_n|} \leq 10^{-4}$$

Las entradas de este módulo son una Semilla, el Largo del Cable (L) y una variable que llamamos "Err" que refleja la precisión que deseamos tener; Obteniendo el siguiente Resultado:

Newton_Raphson(seed,err,L)

seed = 200, err = 10^{-4} , L = 442.5

Valor de "a" conseguido = **1192.53980124362**

Método de la Secante:

El Método de la Secante es un método para encontrar los ceros de una función de forma iterativa. Es una variación del método de Newton-Raphson donde en vez de calcular la derivada de la función en el punto de interés, se aproxima la pendiente a la recta que une la función evaluada en el punto de estudio y en el punto de la iteración anterior.

Este método es de especial interés cuando el coste computacional de derivar la función de estudio y evaluarla es demasiado elevado.

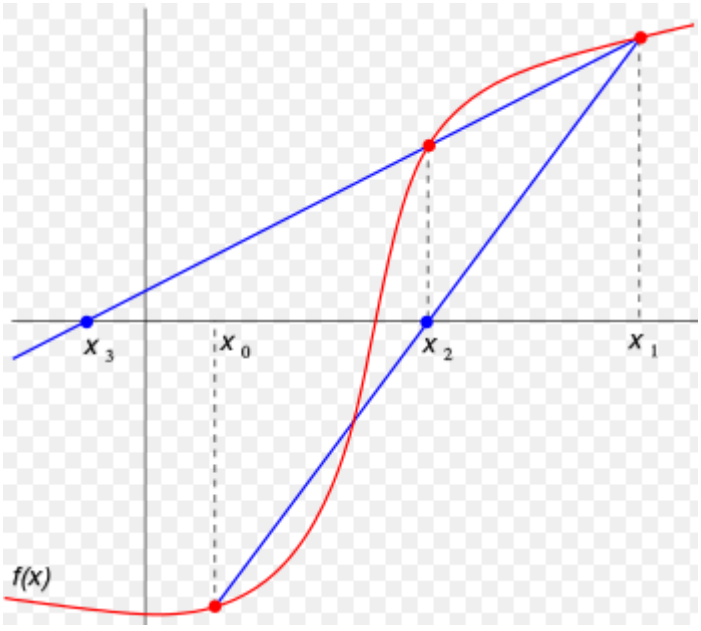


Figura N°2: Representación Gráfica de una iteración del Método de la Secante.

$$x_{n+1} = x_n - f(x_n) * [\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}]$$

También se puede escribir de la siguiente forma cuando $f(x_n)$ es muy cercano a $f(x_{n-1})$:

$$x_{n+1} = x_n - \frac{f(x_n)}{f(x_{n-1})} * [\frac{x_n - x_{n-1}}{(\frac{f(x_n)}{f(x_{n-1})} - 1)}]$$

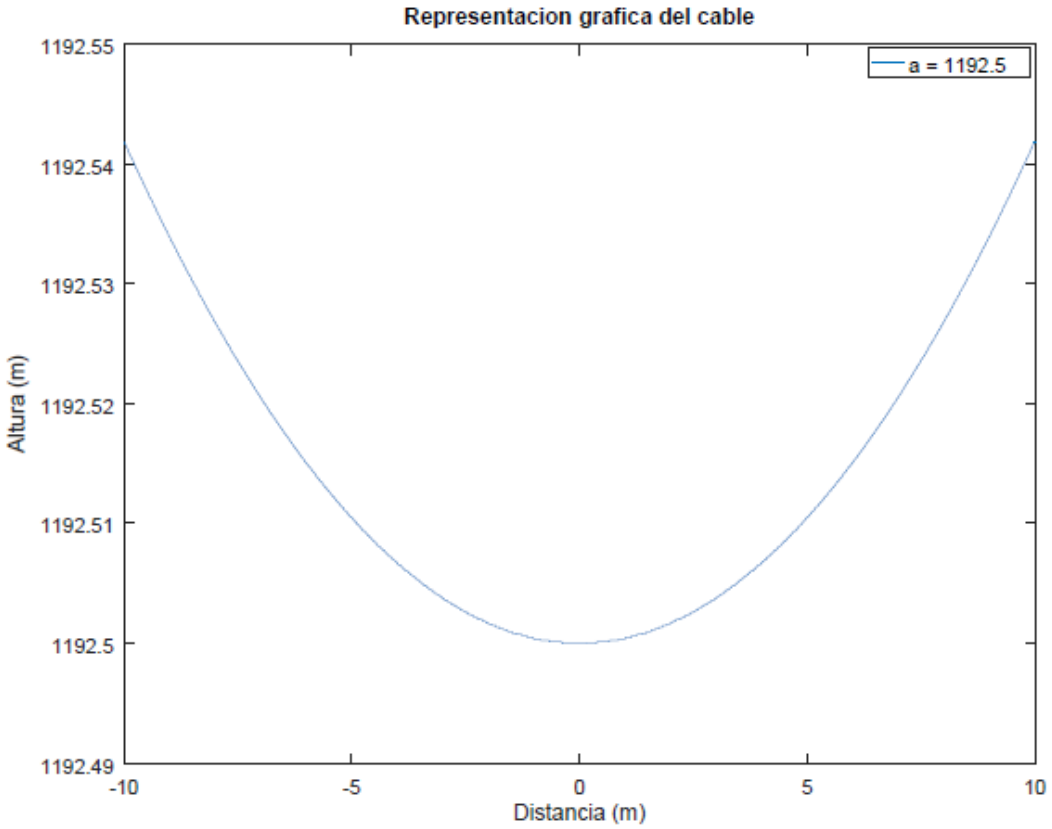
$$x_{n+1} = x_n + \frac{Sn\,(x_n - x_{n-1})}{(1 - Sn)} \; con\; Sn = \frac{f(x_n)}{f(x_{n-1})}$$

De igual manera que para el método de Newton-Rapshon, para la implementación del método de la secante se construyó un módulo de nombre "Metodo_Secante" (ver en Anexos), el cual cuenta con las siguientes entradas:

Metodo_Secante(seed1, seed2,err,L)

Seed1 = 200, seed2 = 400, err = 10^{-4} , L = 442.5

Valor de "a" conseguido = **1192.53959178179**



Ejercicio N°4:

Calcular la flecha f y hacer las modificaciones que correspondan para que el error relativo en la flecha sea menor que 10^{-4}.

$$f(a) = a * (\cosh(D/2a) - 1).$$

D = 440.

Primer debemos hallar el Error Absoluto que presenta la flecha con respecto a "a", para luego hallar el Error Relativo de "a" que nos asegura el Error Relativo en la flecha menor que 10^{-4}:

$$f = f(a) \ \rightarrow f'(a) = f\,(a + \Delta a) \ \approx f(a) + f'(a)\Delta a$$

$f(a)$ es el valor verdadero de la flecha

$f'(a)\Delta a$ es el Error que se comete en la flecha

$$\text{Error Relativo} = \left|\frac{f'(a)\Delta a}{f(a)}\right| \leq 10^{-4}$$

$$\left|\frac{f'(a)\Delta a}{f(a)}\right| = \left|\frac{\cosh\left(\frac{D}{2a}\right) - 1 - \frac{D}{2a}\sinh\left(\frac{D}{2a}\right)}{a\left(\cosh\left(\frac{D}{2a}\right) - 1\right)}\right| |\Delta a|$$

$$= \frac{|\Delta a|}{|a|}\left|1 - \frac{\frac{D}{2a}\sinh(\frac{D}{2a})}{\cosh\left(\frac{D}{2a}\right) - 1}\right| \leq \ 10^{-4}$$

$\frac{|\Delta a|}{|a|}$ = Error Relativo de "a"

$$ß = 1 - \frac{\frac{D}{2a}\sinh(\frac{D}{2a})}{\cosh(\frac{D}{2a}) - 1}$$

$$\rightarrow \frac{|\Delta a|}{|a|} \leq \frac{10^{-4}}{|ß|}$$

De esta manera se concluye que el error relativo de a tiene que ser menor que $\frac{10^{-4}}{|ß|}$ para asegurarnos que el error relativo en la flecha sea menor que 10^{-4}.

Utilizamos el módulo de Newton – Raphson pasándole este error relativo como parámetro y luego calculamos el nuevo valor de la flecha mediante el Módulo Cal_flecha (ver en Anexos):

$$\frac{10^{-4}}{|ß|} = 9.94363012156494 * 10^{-5} \geq 10^{-6}$$

Newton_Raphson (1000, 10^{-6},442.5) → **a = 1192.53980124467**

Cal_flecha (a) → **f = 20.3504411320577**

Ejercicio N°5:

Al cortar el cable se comete un error y ahora L = 443m. Recalcule de vuelta a y f. Estime el cambio relativo entre f y el cambio relativo entre L.
Comente el resultado.

A través del Método de la Secante calculamos nuevamente "a" para un largo L = 443m, obteniendo **a = 1088.81977919109**.

Con este nuevo valor del parámetro "a" calculamos el valor de la

flecha (f) dando como resultado **f = 22.3016190213117**.

Cambio relativo en el largo L:

$$\delta L = \frac{|L - L'|}{|L|} = \frac{|442.5 - 443|}{|442.5|} = 0.00112 = \mathbf{0.1129}\ \%$$

Cambio relativo en la flecha f:

$$\delta f = \frac{|f - f'|}{|f|} = \frac{|20.350441 - 22.301619|}{|20.350441|} = 0.09587 = \mathbf{9.59}\ \%$$

Se observa claramente como un cambio pequeño en el largo del cable provoca un aumento considerablemente mayor en el largo de la flecha, la variación cercana a un 0.1% en el largo del cable se vio reflejada en un aumento de casi el 9.6 % en la flecha.

Por dicho motivo este aspecto se debe tener en cuenta a la hora de realizar el diseño o la construcción de este tipo de estructuras, pudiendo este fenómeno causar el funcionamiento inadecuado de la obra.

Ejercicio N°6:

Sea a = 1 y D = 20. Interpolar con un polinomio P(x) que tome los mismos valores que y(x) para x = -10,-8,-6, …, 0, 2, …, 8,10. Usar el método de interpolación de Lagrange y el método de interpolación de Newton.

Graficar y(x), P(x) (calculado por Lagrange y Newton) y el error cometido en la interpolación Se puede usar la función cosh de Octave para la gráfica de y(x).

Método de Interpolación de Lagrange:

Base de Lagrange:

$$Li(x) = \frac{\sum_{j=1, j\neq i}^{n}(x - x_j)}{\sum_{j=1, j\neq i}^{n}(x_i - x_j)}$$

$$\{L1, L2, \dots, Ln\}\ base\ de\ R^{n-1}[x]$$

Polinomio Interpolante:

$$P(x) = \sum_{i=1}^{n} y_i * L_i(x)$$

Donde $y_i = f(x_i)$

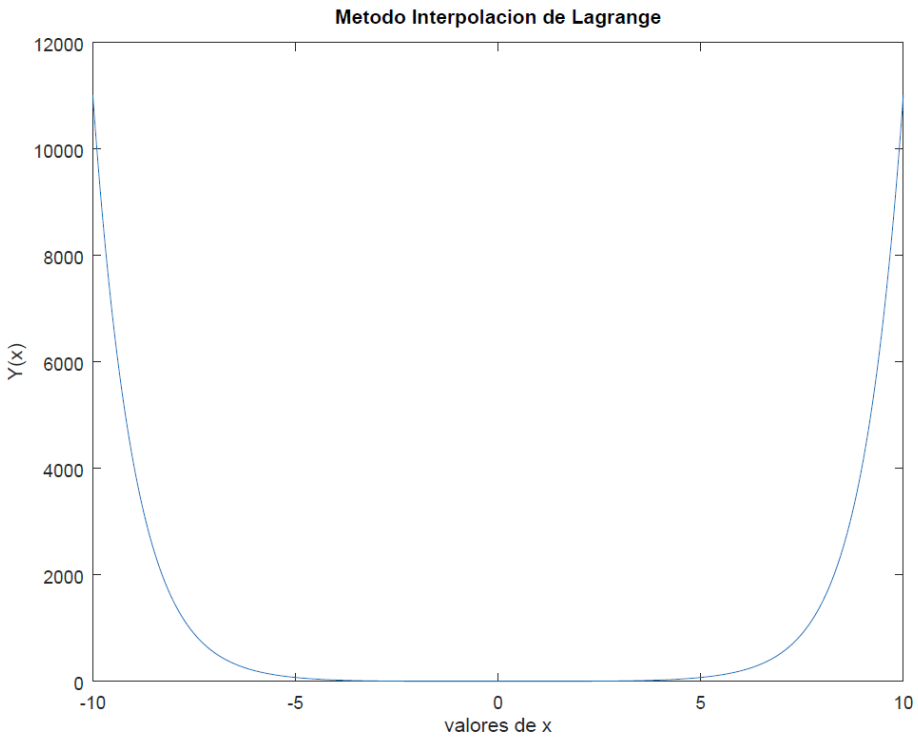
Para la implementación de este Método de Interpolación se realizó un módulo llamado “interpolación_lagrange” (ver en anexos), este módulo recibe el vector “x” con los puntos donde será evaluado el polinomio interpolante $P(x)$. Primero definimos un vector “p” que contiene todos los puntos que serán utilizador para calcular el elemento $L_i(x)$ de la base de Lagrange. El polinomio está dividido en la suma de varios productos, utilizando dos for loop logramos conseguir el mismo resultado. Utilizamos el primer for para calcular la suma de los $y_i * L_i(x)$.

El cálculo de y_i es $\cosh(x_i)$ donde x_i es la variable en la posición i del vector “p”. Luego calculamos $L_i(x)$ utilizando un segundo for. El resultado de este cálculo lo guardamos en la variable “sub_res”, a continuación, en una variable “res” guardamos el producto de $\cosh(x_i)$ por “sub_res” más los valores que tenía anteriormente la variable “res”.

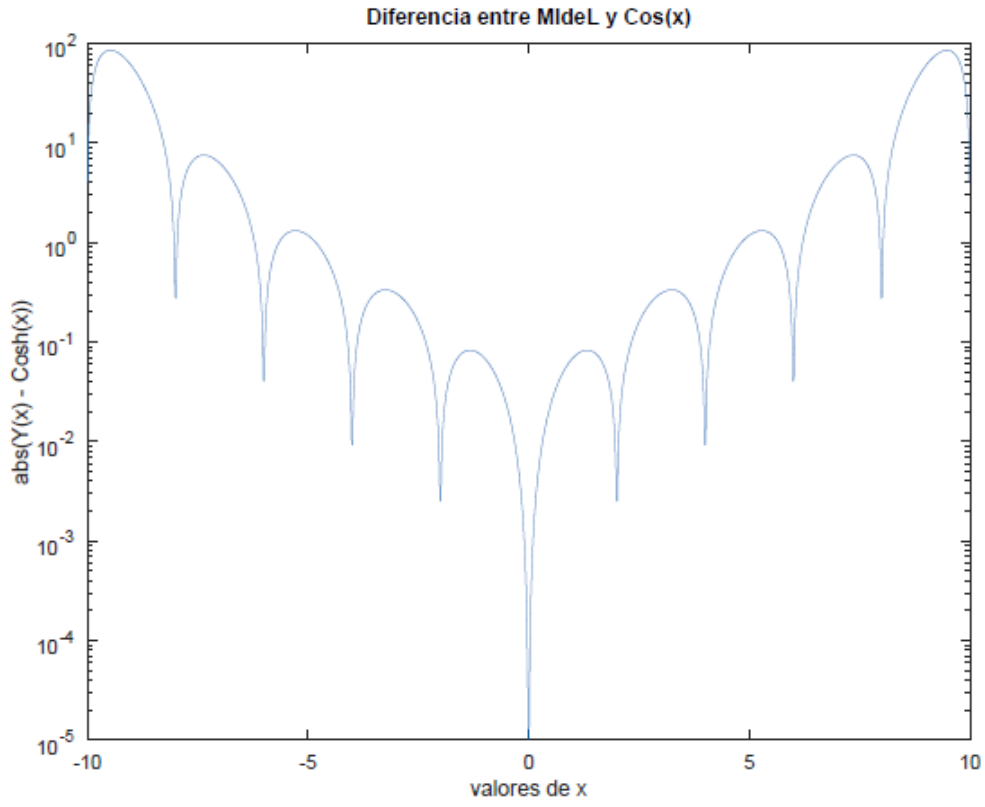
Al utilizar el programa “interpolación_lagrange” brindándole como datos de entrada:

x = -10,-8,-6, …, 0, 2, …, 8,10.

Se obtuvieron los siguientes resultados:



Cuando comparamos el resultado obtenido con el método de interpolación de Newton con la función definida en octave de $\cosh(x_i)$ con $x_i \in [-10 : .01 : 10]$ obtenemos la siguiente diferencia:



Como se puede observar, en los extremos la diferencia de los valores es de 10^2, errores bastante significantes comparados con los obtenido a medida que los valores de "x" se acercan a 0; también podemos observar que el error en cada intervalo aumenta cuando los valores se encuentran en la mitad de dicho intervalo y disminuyen en los extremos.

Esto se debe a que en los extremos de los intervalos el polinomio toma el valor de la función en dichos puntos, mientras que en los puntos internos de estos intervalos es una aproximación a la función.

Método de Interpolación de Newton:

Base de Newton:

$$\{1\,,(x-x_1)\,,(x-x_1)(x-x_2)\,,\dots,(x-x_1)(x-x_2)\dots(x-x_{n-2})(x-x_{n-1})\}$$

$$base\ de\ R^{n-1}[x]$$

Polinomio Interpolante:

$$P_0(x) = f_0(x_0)$$

$$P_i(x) = P(x)_{i-1} + f_i[x_0, x_1, \dots, x_i] * (x - x_0)\dots(x - x_{i-1})$$

Donde tenemos que $f_i[x_0, x_1, \dots, x_i] = \frac{f_{i-1}(x_1,x_1,\dots,x_i) - f_{i-1}(x_0,x_1,\dots,x_{i-1})}{x_i - x_0}$

Para la implementación de este Método de Interpolación se realizaron varios módulos: "interpolación_newton", "interpolación_aux", "polinomio", y "producto" (ver en anexos).

El módulo "interpolación_newton" recibe como parámetros: el vector con los valores donde será evaluada la función $\cosh(x)$; siendo estos los valores conocidos que utilizaremos para luego aproximar el polinomio. También recibe los puntos que serán utilizados para evaluar el polinomio generado mediante el método de interpolación. Este módulo se asegura que el vector este ordenado antes de llamar a la función "polinomio" pasándole como variable el vector ordenado y los puntos para evaluar el polinomio.

La función “polinomio” evalúa el polinomio en el punto y nos devuelve el resultado. Es una función recursiva que depende de la función “interpolación_aux” y “producto”. Tiene un caso base que es el $P_0(x)$ y la llamada recursiva

$$P(x)_{i-1} + f_i[x_0, x_1, \dots, x_i] * (x - x_0) \dots (x - x_{i-1})$$

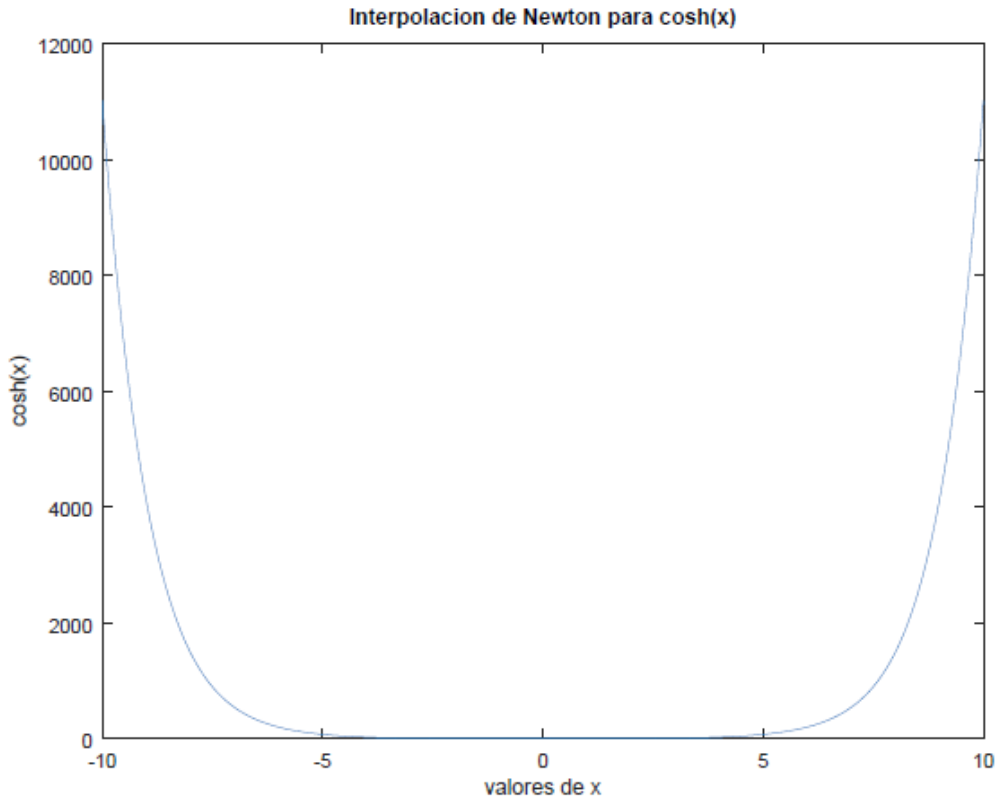
La función “interpolación_aux” es utilizada para calcular $f_i[x_0, x_1, \dots, x_i]$, una función recursiva con dos casos bases. El primer caso es el cálculo de $f_0[x_0] = f(x_0)$; el segundo caso es el cálculo de $f_1[x_0, x_1] = \frac{f_0(x_1) - f_0(x_0)}{x_1 - x_0}$ y el llamado recursivo en el cual se divide el vector $[x_0, x_1, \dots, x_i]$ en dos; el vector $x_1 = [x_0, x_1, \dots, x_{i-1}]$ y el vector $x_2 = [x_1, x_2, \dots, x_i]$ calculando con estos vectores $res = \frac{f_{i-1}(x_2) - f_{i-1}(x_1)}{x_i - x_0}$.

El módulo “producto” esta otra función auxiliar, lo que hace es calcular $(x - x_0) \dots (x - x_{i-1})$ mediante un for loop. Combinando estos módulos logramos obtener una aproximación a la función $\cosh(x)$. Para conseguir esto invocamos la función “interpolación_newton” brindándole como datos de entrada:

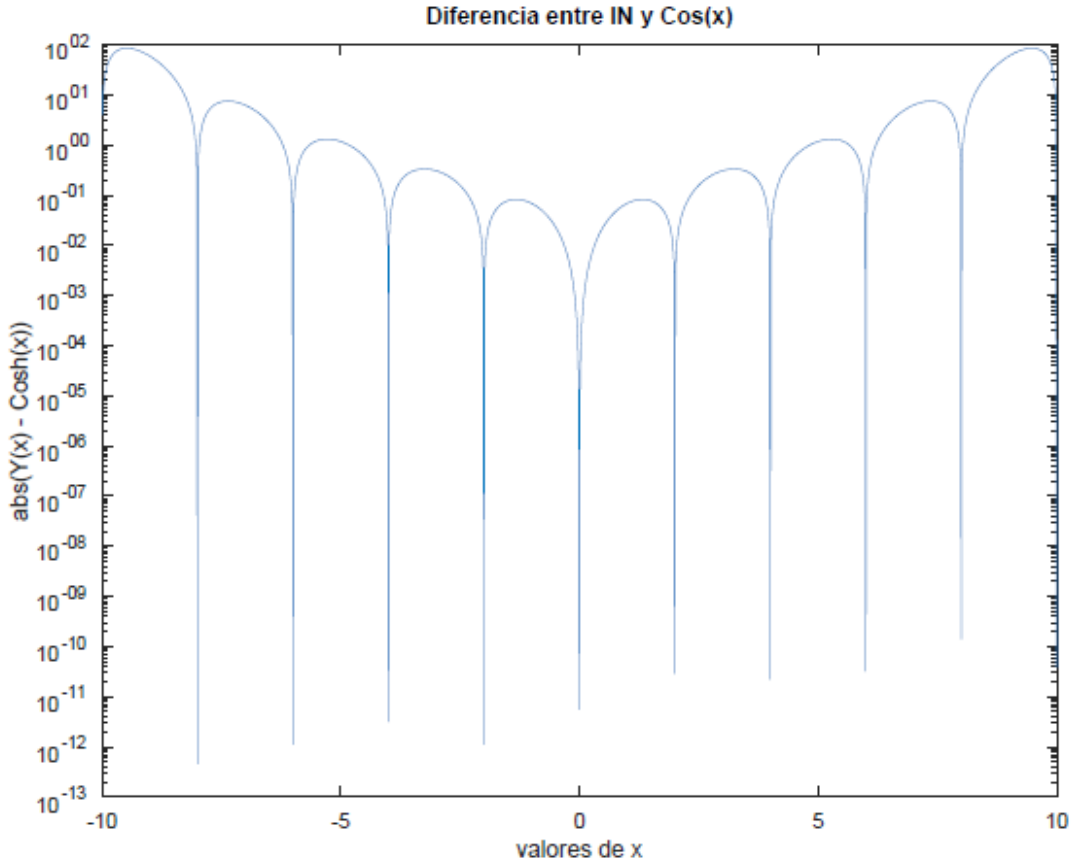
x = -10,-8,-6, …, 0, 2, …, 8,10.

P = -10, -9.99,…,9.99,10

Se obtuvieron los siguientes resultados:



La grafica es idéntica a la que obtuvimos con el método de interpolación de Lagrange, pero cuando observamos la diferencia de interpolación de newton con la de octave vemos que se comportan de forma diferente.



En este caso, cuando los valores de la función se alejan de los extremos de los intervalos, vemos que los errores se mantienen del mismo orden que utilizando la interpolación de Lagrange, sin embargo, cuando los valores se aproximan a los extremos de los intervalos, el polinomio aproxima a la función con una mejor precisión.

Anexos:

```
function a_new =
Newton_Raphson(seed,err,L)


 a_old = seed;
 D = 440;
 da = 10;
while  ( da > err)
   f = 2*a_old*sinh(D/(2*a_old)) - L;
   df1 =2*sinh(D/(2*a_old));
   df2 = (D * cosh(D/(2*a_old)))/a_old;
   df=df1-df2 ;


   a_new = (a_old*df - f)/df;


   da = abs((a_old - a_new)/a_new);


   a_old = a_new;
 endwhile
```

```
function a = Metodo_Secante(seed1, seed2,err,L)


 D = 440; %L = 442.5;


 a = seed1; a_old = seed2;
  da = abs((a-a_old)/a)
  while (da > err)
    f = 2*a* sinh(D/(2*a)) - L;
    old_f = 2*a_old* sinh(D/(2*a_old)) - L;
    if (norm(f-old_f) < 1)
      s = f/old_f;
      num = ( s*(a - a_old) );
      den = (1-s);
      a_old = a;
      a_new = a + num/den;
      a = a_new;
    else
       num = (a - a_old);
       den = (f - old_f) ;
       a_new = a - f * (num/den);
       a_old = a;
       a = a_new;


     endif
    da = abs((a-a_old)/a)
  endwhile
```

```
function flecha = cal_flecha(a)
  D = 440
  x_p = -D/2
  flecha = a * ( cosh( D/(2*a) ) - 1 )
```

```
function res =  interpolacion_lagrange(x)
   u = [-10:2:10];
 res = 0;
 for i= 1 : length(u)
   f = cosh( u(i) );
    sub_res = 1;
   for j = 1:length(u);
     if j == i
       sub_res = sub_res;
       else
       sub_res = sub_res.* (x.- u(j))/(u(i) - u(j));
       endif
     endfor
     res = res + sub_res*f;
     endfor
```

```
function res =
interpolacion_newton(x_vect, punto)
  x = sort(x_vect);
  res = polinomio(x,punto);
```

```
function res = interpolacion_aux(x_vect)
  size = length(x_vect);
  if (size == 1)
    res = cosh(x_vect)  ;
   elseif (size == 2)
    res = (cosh(x_vect(2)) -
cosh(x_vect(1)))/(x_vect(2) - x_vect(1));
   else
      x1= x_vect(1:size-1);
      x2 = x_vect(2:size);
      res = (interpolacion_aux(x2)-
interpolacion_aux(x1))/(x_vect(size) -
x_vect(1));

    endif
```

```
function res = polinomio(x_vector,punto)
   size = length(x_vector);
   if size == 1
   res = interpolacion_aux(x_vector);
 else
   F = interpolacion_aux(x_vector);
   prod = producto(x_vector(1:size-1),punto);
      res = (F*prod) +
polinomio(x_vector(1:size-1),punto) ;
   endif
```

```
function res = producto(x_vect,punto)

  size = length(x_vect);
  res = 1;
  for i = 1:size
    res = res .* (punto - x_vect(i));
   endfor
```