Resumen

El objetivo de este trabajo fue crear un graficador de funciones que pueda desplegar cualquier $f(x)$. Para hacerlo, se utilizó el lenguaje C++, y algunas librerías como: Cairo y Fparser.

1. Introducción

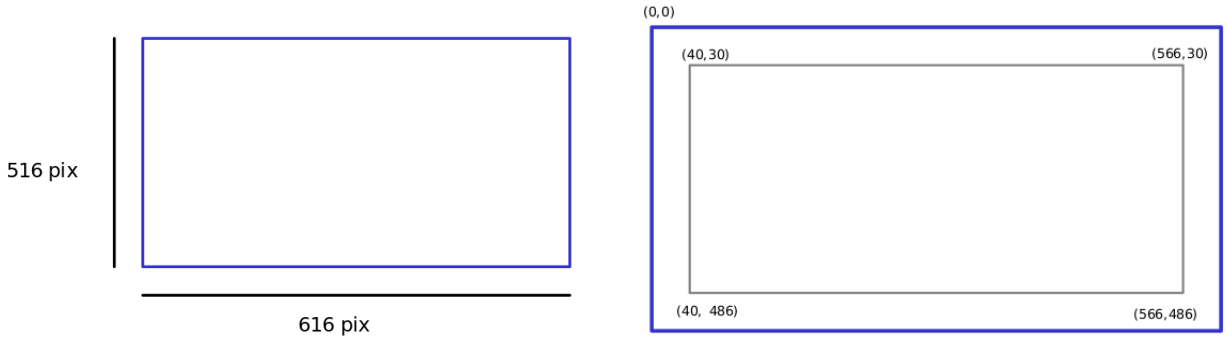
La gráfica de una función $f(x)$ nos permite observar su comportamiento en un dominio $a \leq x \leq b$, así, podríamos observar sus soluciones (en caso que existieran). Esto nos ayudaría con la aplicación de métodos numéricos que dependen de las condiciones iniciales, como es el método de bisección o el método de Newton-Rhapson.

Para hacer graficas de funciones, actualmente existen muchas librerías y aplicaciones. Por ejemplo: Geogebra, Matplotlib, Seaborn, Ggplot, ploty, etc., sin embargo, el resultado que se obtiene depende de los parámetros que se hayan ingresado al crearla, como la cantidad de líneas que hay en la cuadrícula, las cifras significativas que se muestran en los valores x, $f(x)$, y la calidad de salida de la imagen. Por estas razones, es que existen herramientas que nos permiten hacer estos gráficos, como la librería Cairo [1] y OpenGL [2]. Cairo es una librería de software libre que permite crear graficos vectoriales en dos dimensiones [3]. Está escrita en C, pero tiene versiones para usarse en C++, Python, Perl, entre otras, y es posible renderizar gráficos, textos, fuentes o formatos, es multiplataforma y tiene numerosas herramientas [4, 5].

Asimismo, para hacer una graficadora, es necesario tener una forma de entrada de una función, es decir, el usuario ingresa cualquier función en el dominio deseado, y se utiliza una herramienta que después le pasará a Cairo los puntos para graficarlos. Esta herramienta es la librería Fparser [6]. Es una librería en C++ que permite leer una función matemática de una cadena de caracteres, y puede evaluar la función en diferentes puntos; con esto, es posible crear una superficie con Cairo, evaluar la función con Fparser y hacer una conversión de puntos del dominio a pixeles en la gráfica.

2. Metodología

Se utilizó la paquetería Cairo para diseñar la superficie y sus detalles, como el color de fondo, los bordes, los valores de la función, la tipografía, y la gráfica misma. Primero se escogió una superficie de fondo de 516 pixeles por 616 pixeles (fig. 1a) y después se dibujó un borde (fig. 1b).



(a) Superficie inicial Cairo.

(b) Bordes a la superficie y coordenadas.

Figura 1: Dibujo en Cairo.

Segundo, se utilizó la librería Fparser para llamar la función, y finalmente, se hizo una conversión de los valores de x y f(x) a pixeles en C++ para poder graficarlos en Cairo. Esta se hizo utilizando la ecuación de la recta y una transformación que mapea cada punto del dominio a un pixel, el cambio de valores de x a pixeles en el dominio $a \leq x \leq b$ fue con la ec. 1, y con la ec. 2 para los pixeles de y:

$$\text{pixel x} = \left(\frac{\text{valor máximo pixel eje x} - \text{valor mínimo pixel eje x}}{\text{b-a}} \right) * (\text{valor x} - \text{a}) + \text{valor mínimo pixel eje x}, \tag{1}$$

$$\text{pixel y} = \left(\frac{\text{valor máximo pixel eje y} - \text{valor mínimo pixel eje y}}{\text{mínimo f(x)- máximo f(x)}}\right) * (\text{valor f(x)} - \text{máximo f(x)}) + \text{valor mínimo pixel eje y}, \tag{2}$$

de está forma se unieron los puntos de los valores anteriores con los siguientes para hacer gráfica de la función.

3. Resultados

Se gráficaron tres funciones:

$$f(x) = x^3 + x^2 + 1, \tag{3}$$

y se guardaron en archivos ".svg"(Scalable Vector Graphics) [7], que tienen la propiedad de no perder la calidad bajo acercamientos consecutivos, es decir, se renderiza con cada ampliación, como en las fig. 2,4 y 6. Además, en las subfiguras (3a, 3b, 3c, , 5a, 5b, 5b, 7a, 7b, 7c) se muestran aumentos específicos para ver su calidad.

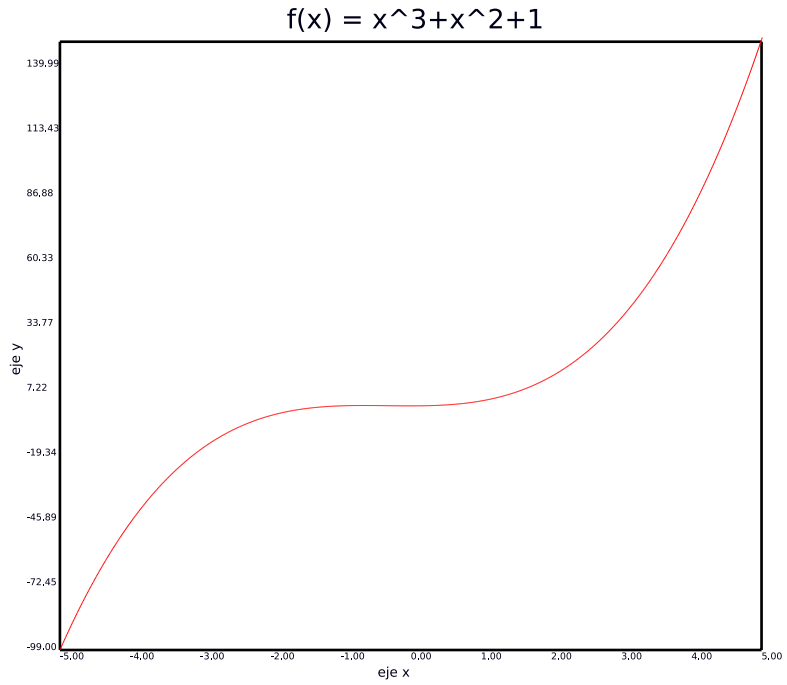
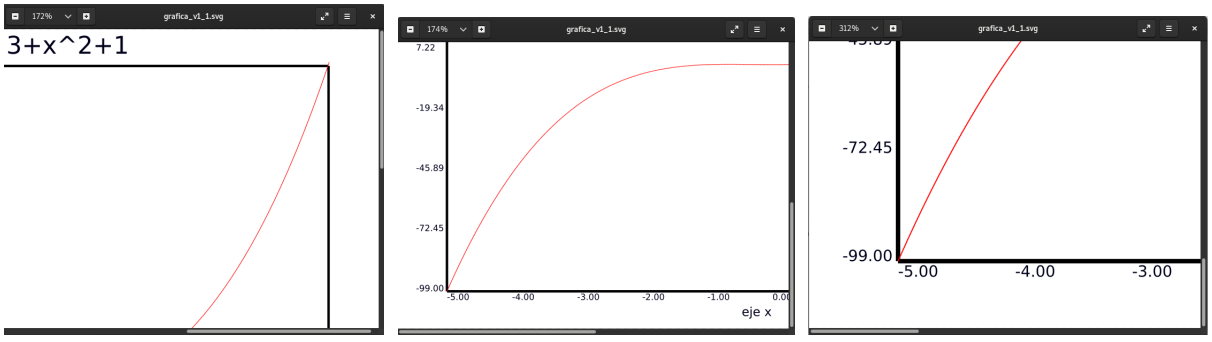


Figura 2: Gráfica de la función f(x)



(a) 1 (b) 2 (c) 3

Figura 3: Zoom a diversas partes de la gráfica f(x)

$$g(x) = \sin(10x) \tag{4}$$

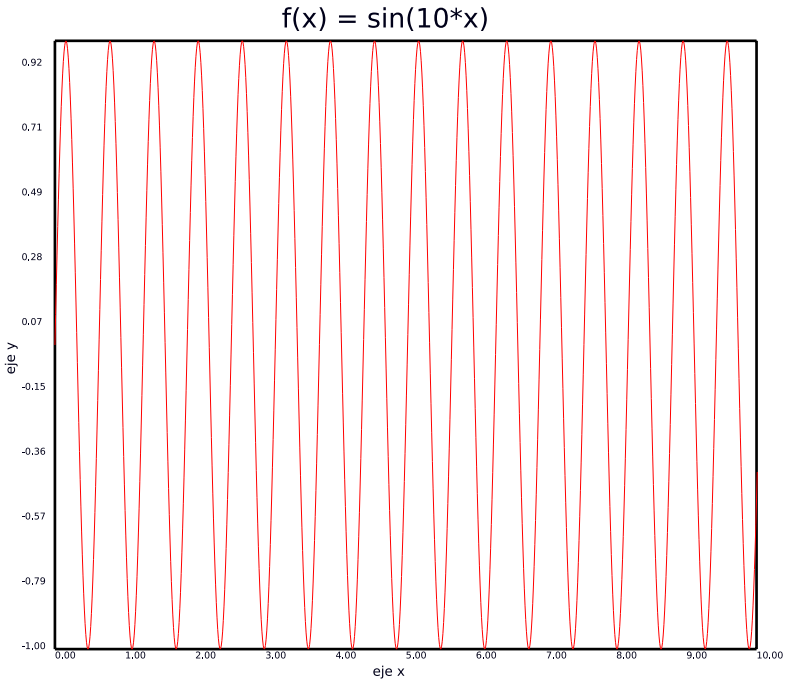
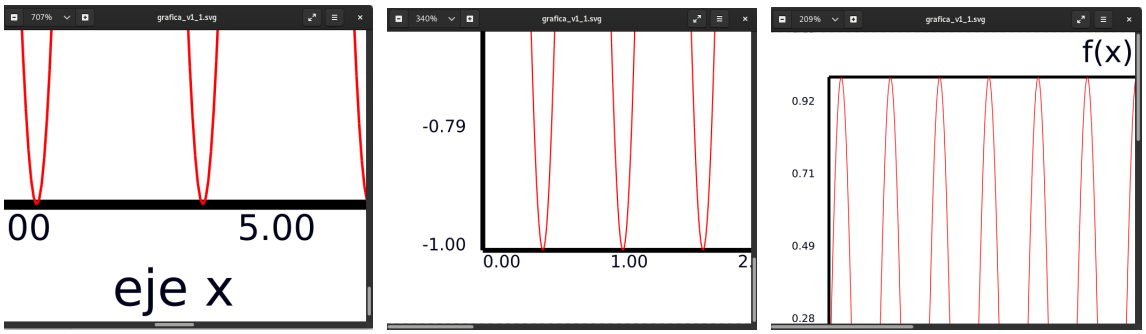


Figura 4: Gráfica de la función g(x).



(a) 1 (b) 2 (c) 3

Figura 5: Zoom a diversas partes de la gráfica g(x)

$$h(x) = \frac{1}{x+1} \tag{5}$$

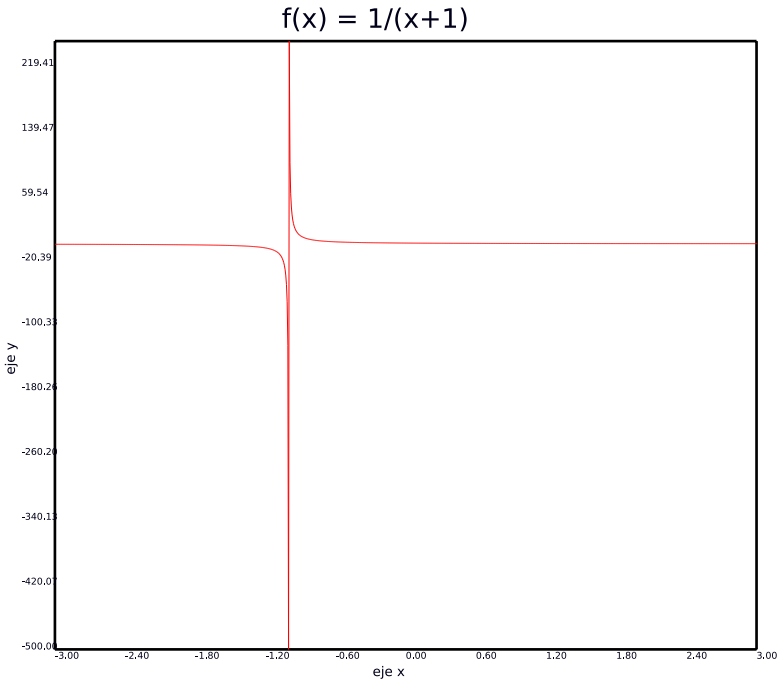
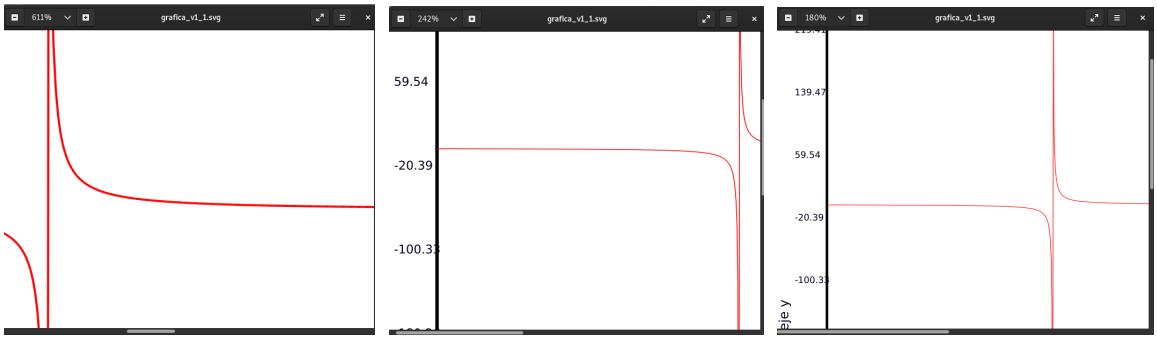


Figura 6: Grafica de la función h(x)



(a) 1 (b) 2 (c) 3

Figura 7: Zoom a diversas partes de la gráfica h(x)

4. Reporte de los resultados

Para poder ejecutar el archivo, fue necesario descargar la librería Fparser, e instalar algunas paqueterías, y también utilizar ”flags”para utilizar Cairo, y ejecutar otros dos archivos de la carpeta de Fparser .

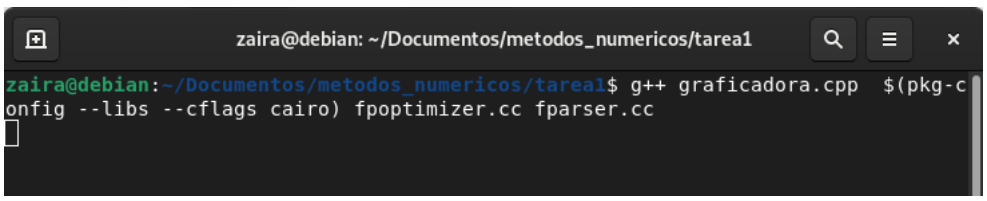


Figura 8: Caption

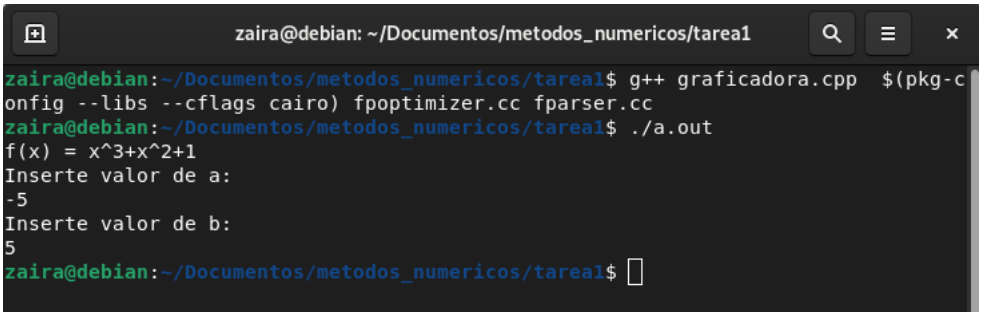
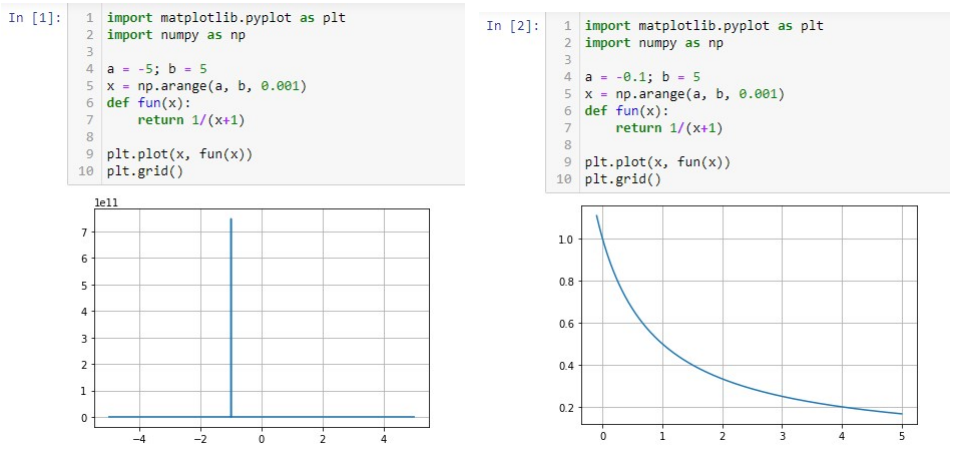


Figura 9: Caption

5. Conclusiones y discusiones

En este trabajo se utilizaron diversas librerías para hacer un graficador de funciones, y al comparar la gráfica de la ec. 5 con la graficadora desarrollada y con la librería Matplotlib de Python [8], ambas muestran la indeterminación con una línea recta, esto porque la pendiente no existe en el punto x = -1; en las fig. 6 y fig. 10b se suaviza después de este punto, y en la fig. 10a luce como la gráfica de la función delta de Dirac en todo el dominio.

Además, como trabajo a futuro se espera añadir diferentes funciones para que el usuario decida más su diseño, por ejemplo: la cuadrícula y la cantidad de cuadros en esta, la cantidad de valores que se mostrarán en los ejes, la cantidad de cifras significativas en los valores, y también, un botón de aumento para que el usuario decida dónde enforcarse.



(a) (b)

Figura 10: Gráfica de h(x) con Python

Referencias

[1] Cairo. https://www.cairographics.org/documentation/. Accessed: 2022-08-15.

[2] Opengl. https://www.opengl.org. Accessed: 2022-08-14.

[3] Wikipedia contributors. Cairo (graphics) — Wikipedia, the free encyclopedia, 2022. [Online; accessed 18-August-2022].

[4] The cairo graphics library. https://zetcode.com/gfx/cairo/cairolib. Accessed: 2022-08-14.

[5] Cairo graphics. https://sullivan.mathcs.wilkes.edu/courses/cs246-spring-2020/lectures/cairo/cairo.html. Accessed: 2022-08-15.

[6] Function parser for c++ v4.5.2. http://warp.povusers.org/FunctionParser/. Accessed: 2022-08-14.

[7] Wikipedia contributors. Scalable vector graphics — Wikipedia, the free encyclopedia, 2022. [Online; accessed 18-August-2022].

[8] Matplotlib. https://matplotlib.org/stable/index.html. Accessed: 2022-08-14.