Problema 2

Enunciado

Utilizando un método numérico, encuentre una raíz de la ecuación

$$f(x) = \frac{\sin(x)}{x}.$$

Metodología

Para resolver este problema elegí implementar el método de Newton-Raphson. Este método se utiliza para aproximar raíces de una función mediante la siguiente relación:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}.$$

Para atacar nuestro problema necesitamos definir algunos parámetros. Entre ellos está la **tolerancia** del método. Es decir, cuán cerca de la solución verdadera tiene que estar nuestra aproximación para considerarla suficientemente buena. También debemos definir un **número finito de iteraciones** en las cuales nuestro programa hará lo posible por acercarse tanto cuanto pueda a la solución. Por otro lado, debemos definir un **valor inicial** para x_i, fácilmente notamos que nuestra función se hace cero en $x = \pi$ entonces tomamos $x_0 = 2$. Finalmente, debemos aprovechar que la derivada de nuestra función queda en términos de coseno y la función misma:

$$f'(x) = \frac{\cos(x)}{x} - \frac{\sin(x)}{x^2} = -\frac{1}{x}(\cos(x) - f(x)).$$

La relación que compararemos con la tolerancia para validar nuestra aproximación es

$$x_{i+1} - xi = -f(x_i)/f'(x_i),$$

dado que el lado derecho de la ecuación se hace más pequeño conforme nos acercamos a la solución. Entonces, mientras más pequeña sea la diferencia entre x_{i+1} y x_i, más cerca de la solución estaremos.

Variables de entrada y salida

Este programa no está diseñado para interactuar con el usuario, sin embargo, podemos considerar que hay variables de entrada para la función empleada para el método N-R. → reales: x_0, tolerancia
→ enteros: iteradores, numero máximo de iteraciones
← cadena de caracteres con números: x (raíz de la función)
* Variables locales: valor anterior de x (x_i), valor actual de x x_{i+1}, diferencia ($x_{i+1} - x_i$), iteración actual.

Pseudocódigo

Listing 1: Pseudocódigo del método Newton-Raphson

```
definir tolerancia , xinicial , iteradormaximo
definir funcion(float x){
    retorna f(x)
}

definir derivada(float){
    retorna f'(x)
```

```
}

definir metodoNR(tolerancia, xinicial, iteradormaximo, mas variables){
    float xnuevo
    iniciar iterador =0
    xinicial=x0
    diferencia=|x_nuevo-xinicial|
    mientras(diferencia > tolerancia y iterador<iteradormaximo){
        diferencia=|x_nuevo-x_inicial|
        x_inicial=x_nuevo
        x_nuevo=x_inicial-funcion(x_inicial)/derivada(x_inicial)
        iterador++
    }
    apuntar solucion en memoria
    apuntar iteracion en memoria

}
```

El código de este programa está disponible **aquí.**

Resultados

El solucionador encontró un cero en $x = 3.14159$ con una tolerancia de 0.0001 y un valor inicial de $x_0 = 2$ después de 4 iteraciones.

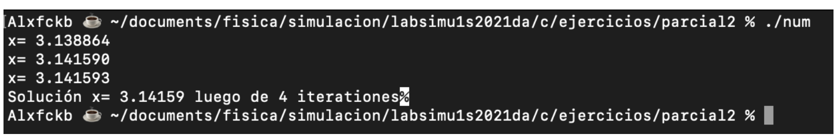

Figure 1: Captura de pantalla del resultado obtenido utilizando el método Newton-Raphson.

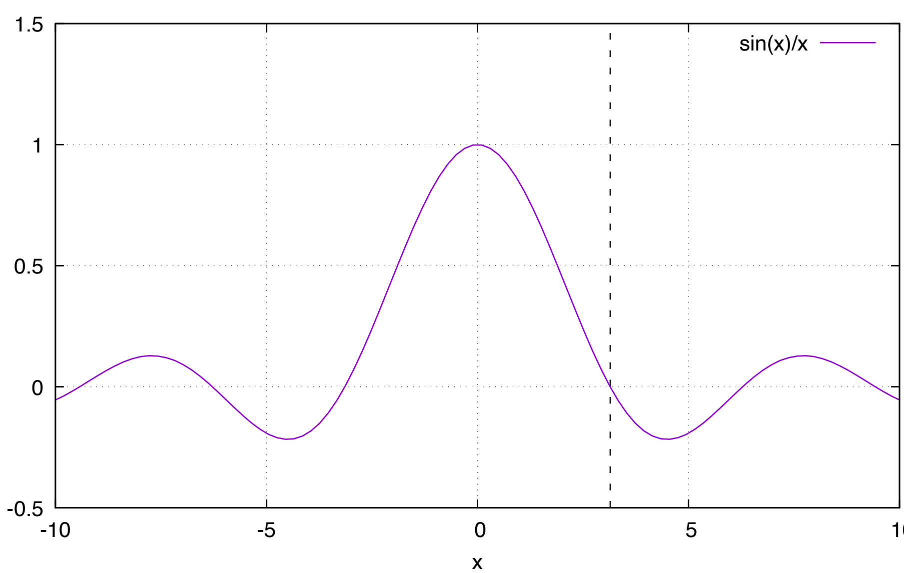

Figure 2: Gráfica de $Sin(x)/x$ con solución en $x = \pi$.

Problema 1

Enunciado

Se presentan los valores obtenidos de forma experimental de presión y volumen, con una incerteza de 0.1 pulgadas cubicas y 0.2 lbin.

Volumen (in^3)	Presión ($\frac{lb}{in^3}$)
54.3	61.2
61.8	49.2
72.4	37.6
88.7	28.4
118.6	19.2
194.0	10.2

Metodología

Este problema presenta una expresión que relaciona cantidades de forma exponencial, dicha relación es:

$$PV^a = b.$$

La cual se puede linealizar fácilmente tomando el logaritmo de ambos lados. Así, obtenemos

$$\ln P = -a \ln V + \ln b.$$

Entonces definimos $y = \ln P$, $x = \ln V$ y $B = \ln b$. Obteniendo una relación lineal de la forma

$$y = -ax + B.$$

Una vez hecho esto, podemos aplicar el método de mínimos cuadrados para los logaritmos de P y V.

Debido a la forma matemática que tenemos para encontrar m y b por este método, es bastante conveniente trabajar de forma modular. Es decir, definir funciones generales que aparecen a menudo en nuestras expresiones útiles. De esta forma, nos ahorramos líneas de código y tiempo, dado que solamente quedaría ensamblar las funciones para m y b.

Variables de entrada y salida

Este programa no está diseñado para interactuar con un usuario, sin embargo, podemos considerar que hay variables de entrada para la función que calcula $\sum_i x_i$,$\sum_i x_i y_i$, m y b.

→ Variable global entera i (iterador para las distintas funciones del programa).
→ vectores con entradas tipo float x, y con los logaritmos de V y P, respectivamente.
→ Float (sumatorias realizadas).
→ Entero (longitud de nuestros vectores).
← Cadena de caracteres (Expresion de recta que mejor se aproxima).
← Cadena de caracteres (tabla en un archivo de texto para luego ser leido por gnuplot).
← Cadena de caracteres con numeros (P para $V = 100$).
* Variables locales (almacenan valores de sumatorias, de m y de b).

Pseudocódigo

Listing 2: Pseudocódigo del método mínimos cuadrados

```
float x[]={valores Log(V)}
float y[]={valores Log(P)}
int n = longitud de x[]
int i (iterador)
float suma(float x1[]){
    suma = 0
    para (i=0 hasta i<n con paso 1){
    sum = sum + x1[i]
    returnar sum
    }
}
float suma2(float x1[], float x2[]){
    sum2 = 0
    para (i=0 hasta i<n con paso 1){
    sum2 = sum2 + x1[i]*x2[i]
    returnar sum2
    }
}
float M(){
    float m
    m = (n*sum12(x,y)-suma(x)*suma(y)/(n*suma2(x,x)-(suma(x))^2)
    retornar m
}
float B(){
    float b
    b =(sum(y)-M()*sum(x))/n
}

void main(){
    imprimir("La curva que mejor aproxima es y=%fx+%b",M(),B())
    imprimir("El valor de P para V=100 es %f", valor)
}
```

El código de este programa está disponible **aquí.**

Resultados

La recta obtenida que mejor se aproxima a los datos dados es $y = -1.395324x + 9.639354$. La presión obtenida para $V = 100\text{in}^3$ es $P = 24.87\text{lb}/\text{in}^3$.

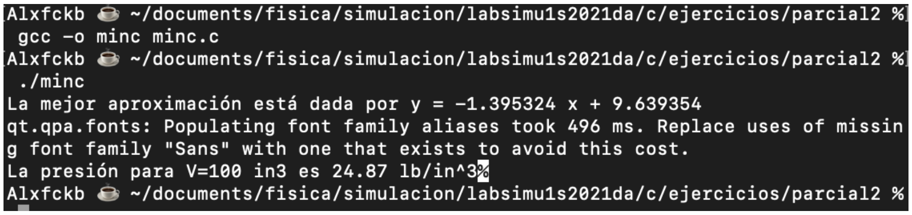

Figure 3: Programa compilado y ejecutado desde terminal.

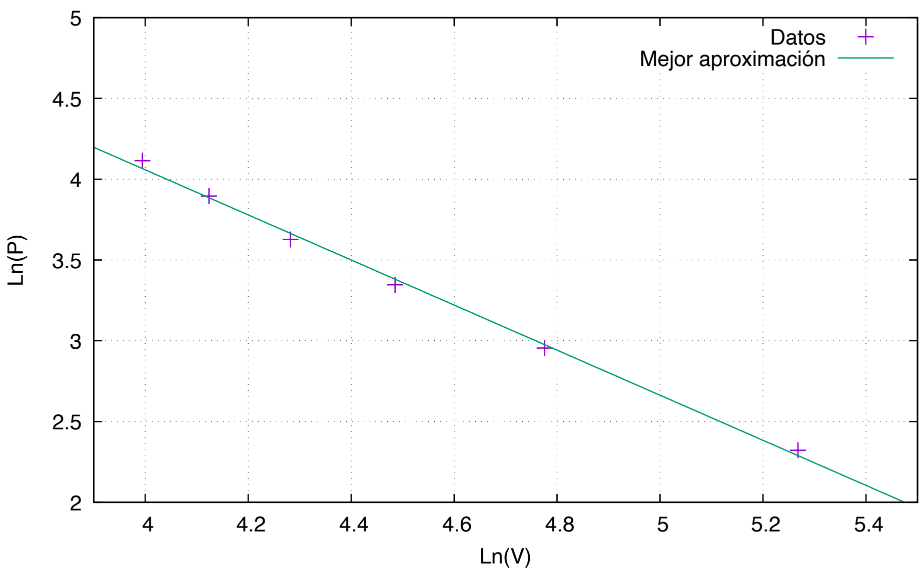

Figure 4: Gráfica de los datos dados y la curva que mejor se aproxima.