Examen parcial 2

Nombre: Jorge Alejandro Rodriguez Aldana

Carné: 201804766

Problema 1

> En un experimento se presentan los valores obtenidos de velocidad con respecto al tiempo, con una incerteza de 0.1 s:
>
> | Velocidad | Tiempo |
> |---|---|
> | 2.1 | 1 |
> | 3 | 3 |
> | 5.2 | 8 |
> | 7.1 | 11 |
> | 9.2 | 14 |
> | 10.1 | 18 |
>
> Se le solicita que genere un programa el cual cumpla con las siguientes condiciones:
>
> - Una gráfica que compare los valores tabulados y la recta que mejor aproxima el comportamiento.
> - Obtenga la aceleración aproximada que determina su modelo. Estimar ls velocidad si se toma la medición a los 15 segundos.

Documentación para el usuario

En una terminal desde esta carpeta ejecutar el archivo Exe.sh:

```
    ./Exe.sh
```

Metodología

Para este problema solo podíamos seleccionar el método de mínimos cuadrados, así que fue el implementado. Además del programa en C, se programaron otros archivos para automatizar la ejecución para este problema específico.

Fue conveniente separar el código en C en dos archivos, aunque casi son un copy-paste del mismo código, uno de ellos tiene como salida un texto explicito para insertar en gnuplot.

Plan

Según cada parte solicitada, planifiqué lo siguiente:

- Una gráfica que comparara los valores
  - Tabular los datos en un archivo de texto, y agregar una columna con el error de tiempo.
  - Plotear los datos con Gnuplot
- Un ajuste de una recta que mejor se aproximara
  - Agregar una función a Gnuplot y hacerle un *fit*
  - Programar el método de mínimos cuadrados
- Obtener la aceleración aproximada
  - La aceleración es igual a la pendiente estimada
- Estimar la velocidad en $t = 15s$
  - La velocidad dependiente del tiempo está dada por la ecuación de la recta, solo hace falta valuarla en $t = 15s$

Variables de entrada

```
int n              // Cantidad de elementos en las listas
double v           // Lista de velocidades
double t           // Lista de tiempos
double err         // Error de la medicion de t
double tfinal      // Tiempo en el que se desea conocer la velocidad
```

Variables de salida

```
double m              // La pendiente
double b              // El corrimiento
double Dm             // El error de m
double Db             // El error de b
```

Funciones

```
double ProdDSum             // Funcion que suma los elementos de dos lists y los
multiplica
double SumDProd             // Funcion que multiplica elemento a elemento de dos
listas y suma los productos
double Sum                  // Funcion que suma los elementos en una lista
double m                    // Funcion que calcula la pendiente
double b                    // Funcion que calcula el corrimiento
double Dm                   // Funcion que calcula el error de m
double Db                   // Funcion que calcula el error de b
```

Pseudocódigo general

Programa 1, grafica

```
Calcula la pendiente
Calcula el corrimiento
Imprime la función
```

Programa 2, pregunta directa

```
Calcula la pendiente
Calcula el corrimiento
Calcula el error de la pendiente
Calcula el error del corrimiento
Calcula la velocidad a los 15s
Imprime la pendiente +/- el error
Imprime el corrimiento +/- el error
Imprime la velocidad a los 15s
```

Pseudocódigo explicito

```
main():
    print(m()±Dm())          // Valua e imprime la pendiente y su error
    print(b(m())±Dp())       // Valua e imprime el corrimiento y su error
    v = m()*15+b(m())        // Valua la funcion en 15s
    print(t)                 // Imprime este valor
```

```
m(): m                  // Ecuación
b(): b                  // Funcion que calcula el corrimiento
```

Con:

$$m = \frac{n\sum_{k=1}^{n}(x_k y_k) - \sum_{k=1}^{n} x_k \sum_{k=1}^{n} y_k}{n\sum_{k=1}^{n} x_k^2 - \left(\sum_{k=1}^{n} x_k\right)^2}$$

$$b = \frac{n\sum_{k=1}^{n} y_k - m\sum_{k=1}^{n} x_k}{n}$$

Problema 2

> Utilizando un método numérico, encuentre una raíz de la ecuación
>
> ```
> f(x) = e^{-x^2/2}-0.5
> ```
>
> Debe de realizar la gráfica de la ecuación y comparar el resultado obtenido con el programa realizado en C.

Documentación para el usuario

En una terminal desde esta carpeta ejecutar el archivo Exe.sh):

```
    ./Exe.sh
```

Metodología

Para realizar este problema utilicé el método de bisección, por que, aunque se ve más hambriento de poder, también es más simple, lo que evita código más complejo, y con esto, la documentación también es más sencilla de hacer. En mi caso, no tengo problema con la potencia que va requerir para computarlo relativamente rápido.

Plan

Comenzar realizando la gráfica para determinar el intervalo inicial a darle al programa. Y después proceder a correr el programa con este intervalo.

Variables de entrada

```
double FMin          // Menor valor en el intervalo original
double FMax          // Mayor valor en el intervalo original
```

Variables de salida

```
double r            // Raíz del polinomio
```

Otras variables

```
double MinErr   // Error mínimo admitido
double Min      // Menor valor en el intervalo actual
double Max      // Mayor valor en el intervalo actual
double Med      // Variable para almacenar el valor medio actual entre Min y Max
double r        // [r]oot Varialbe para almacenar el valor actual de la raiz
double l        // [l]ast Varialbe para almacenar el valor anterior de la raiz
```

Funciones

```
void main()             // Funcion inicial
double eval()           // Funcion que realiza las comparaciones evaluaciones
double err()            // Funcion que calcula el error
double f()              // Funcion que emula la funcion matematica
```

Pseudocódigo general

```
Calcula el valor medio actual
Revisa el error de ese valor
Si el error es mayor a 1, repite
```

Pseudocódigo explicito

```
main():
    Min = FMin          // Asigna el intervalo inicial al intervalo variable
    Max = FMax          // Asigna el intervalo inicial al intervalo variable
    l = FMin            // Guarda el valor anterior de la raiz en l
    r = eval()          // Guarda el valor acual de la raiz en r

    while (err()>1)     // Loop para iterar mientras el error no sea menor a 1
        l = r           // Guarda el valor anterior de la raiz en l
        r = eval()      // Guarda el valor acual de la raiz en r
    print(r)            // Imprime la raiz con error menor al 1% al usuario
```

```
eval():
    Med = (Min+Max)/2       // Evalua el valor medio entre Min y Max actual
    if (f(Med)*f(Min) > 0)  // Compara el signo
        Min = Med           // Remplaza el minimo si tiene el mismo signo
    else
        Max = Med           // Remplaza el maximo si tiene l mismo signo

    return Med              // Regresa el valor medio actual
```

```
err():
    return (r-l)*100/r     // Regresa el valor del error actual
```

```
f(x):
    return exp(-x^2/2)−0.5  // Regresa el valor de la funcion evaluada en x
```