Segundo Parcial, Lab. Simulación

José Alejandro García González, carnet: 201804567

4/5/2021

1. Mínimos cuadrados

1.1. Planteamiento

Se toma el crecimiento de una planta tabulando la siguiente evolución:

Semana	Altura(m±0.01m)
1	0.01
2	0.03
3	0.09
4	0.13
5	0.19
6	0.22
7	0.37
8	0.43

Le solicitan que genere un programa el cual cumpla con las siguientes condiciones:

- Una grafica que compare los valores tabulados y la recta que mejor aproxima el comportamiento.
- Estimar la altura de la planta despues de 23 semanas.

1.2. Metodología

Para realizar el ajuste se usará el método de mínimos cuadrados, el cual hace un ajuste lineal a una serie de k datos, de la forma: $y = mx + b$, donde los parámetros m y b son calculados de la siguiente forma:

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

$$b = \frac{\sum_{k=1}^{n} y_k - m \sum_{k=1}^{n} x_k}{n} \tag{2}$$

Donde n es el número total de datos a los que se les hará el ajuste. Este se calculará por medio de un script en C, y luego se ingresara a un archivo ejecutable de genuplot para poder graficar los datos junto al ajuste calculado.

1.3. Abstracción del problema

Para nuestro caso, haremos el ajuste para $y(t) = mt+b$, es decir, pondremos a la altura dependiente del tiempo. De esta forma, a la altura le asignaremos la variable y (misma que en las ecuaciones (1) y (2)) y al número de semanas transcurridas le asignaremos la variable t, la cual sustituirá a la variable x en las ecuaciones (1) y (2).

Si bien la altura posee error, la propagación del mismo a m y b no afectara la veracidad del ajuste de manera significativa, pues es un error muy pequeño. Por ello, vamos a despreciarlo para realizar el ajuste, pero no para dar la respuesta a la segunda pregunta planteada. De forma adicional, imprimiremos los valores de m y b para ingresarlos a un script de Gnuplot para poder graficar el ajuste, además de incluir a los datos en un archivo de texto, también para que puedan ser graficados en la misma gráfica.

Variables de entrada:

- Lista con los tiempos ->arreglo float
- Lista con alturas ->arreglo float
- Archivo de texto con datos ->para script de Gnuplot

Variables de salida:

- *res* (respuesta pregunta) ->float
- m y b ->float
- Gráfica de puntos vs ajuste (en archivo .gp)

Notamos que las ecuaciones (2) y (1) tienen sumatorias en común, por lo que es posible resumir toda la expresión con dos funciones: una que calcule la sumatoria de una lista de datos, y una que calcule la sumatoria del producto de los elementos de dos listas, uno a uno.

Pseudocódigo:

1. Ingresar datos a archivo datos.txt (para usarse en archivo .gp)
2. Ingresar datos a las listas T[] y Y[]
3. Declarar funciones de sumatoria y sumatoria de una multiplicación.
4. Calcular m y b usando (1) y (2) respectivamente.
5. Usar m y b para calcular res.
6. Imprimir m, b y res en consola.
7. Ingresar el ajuste a archivo .gp.
8. Sentencia *system* para ejecutar archivo .gp de inmediato.

1.4. Resultados

A continuación los resultados del programa. El código se encuentra en los archivos Crecimientoplanta.c y graficador.gp dentro de esta misma carpeta.

Figura 1: Gráfica generada

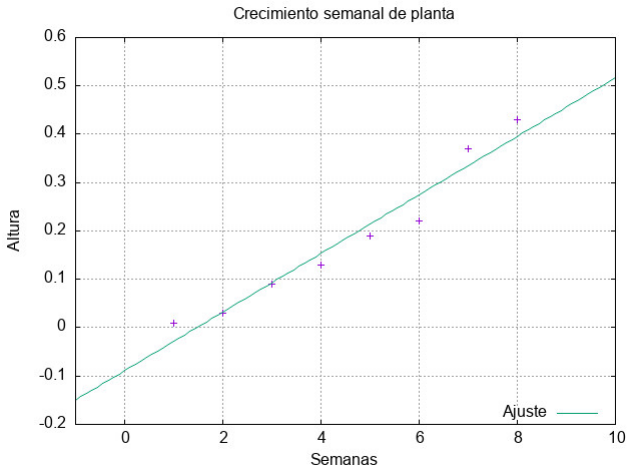
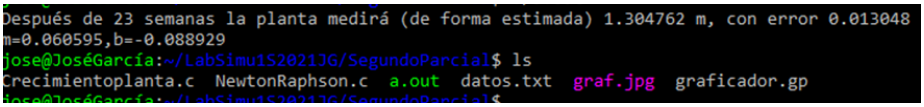


Figura 2: Respuesta a la pregunta



Como podemos ver en la figura 1, el ajuste calculado se aproxima bastante al comportamiento de los datos, siendo que la mayoría se sale de la gráfica solo por un pequeño rango. Para mayor seguridad de la veracidad de nuestro ajuste, se recomienda calcular el coeficiente de determinación para el ajuste hecho.

2. Newton-Raphson

2.1. Planteamiento

Utilizando un método numérico, encuentre una raíz de la ecuación:

$$f = \sqrt{\frac{x}{2}} - 1 \tag{3}$$

Debe de realizar la gráfica de la ecuación y comparar el resultado obtenido con el programa realizado en C.

2.2. Metodología

Para calcular la raíz de la ecuación (3) vamos a usar el método de Newton Raphson, el cual establece una recursividad que se mueve por todo el eje x desde un punto inicial buscando la raíz del polinomio mediante el siguiente algoritmo:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4}$$

Donde $f'(x)$ es la derivada de la función a la que se le busca la raíz. Además, este método posee una manera de definir la presición del candidato a raíz que encuentra:

$$|x_{n+1} - x_n| \tag{5}$$

2.3. Abstracción del problema

Derivando (3) :

$$\frac{df}{dx} = f'(x) = \frac{1}{4}\sqrt{\frac{2}{x}} \tag{6}$$

Por ser una función que lleva raíces cuadradas, esta no está definida en los negativos. Por ello debemos hallar una manera de evitar que esto suceda en nuestro programa, además de prevenir también un valor $x = 0$, pues eso indefine (6). Con estas consideraciones, dividiremos el problema en tres funciones: una para (3), otra para (6), y una última para (4). Además, limitaremos la cantidad de iteraciones del programa, para evitar que se quede haciendo cálculos indefinidamente por no alcanzar la presición deseada.

Variables de entrada:

- x_0, punto de partida ->float
- *pres*, presición ->float

Variables de salida:

- *sol*, raíz encontrada ->float
- *iter*, iteraciones hechas ->int
- Mensaje de error ->Impreso en consola

Pseudocódigo:

1. Definir las máximas iteraciones (*maxiter*) como una constante dentro del programa.
2. Solicitar x_0 y *press* al usuario.
3. Si $x_0 < 0$ volver a solicitar el valor hasta que $x_0 > 0$.
4. Declarar un contador i tipo int.
5. Mientras $i < maxiter$:
   a) Valuar (4).
   b) Valuar $dif = |x_{i+1} - x_i|$.
   c) Si $dif < pres$, ir a paso 5.d, sino ir a paso 5.e
   d) Devolver el resultado de (4).
   e) Sumarle uno al contador y repetir todo el paso 5.
6. Imprimir resultado y número de iteraciones, o mensaje de error en su defecto.

2.4. Resultados

El programa calcula la raíz en $x = 2$ con total presición, y tal y como se puede ver en la gráfica adjunta, esta es la raíz de la función. Se probó con diversos puntos iniciales. El código se encuentra en el archivo NewtonRaphson.c dentro de esta misma carpeta.

Figura 3: Gráfica de la función

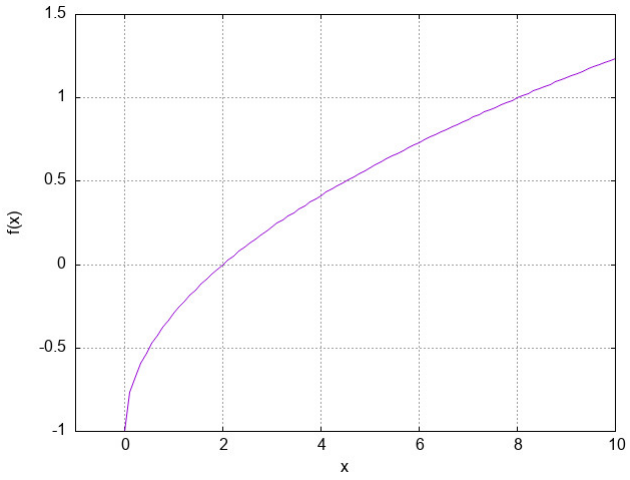


Figura 4: Respuesta 1

```
*****Método de Newton-Raphson para buscar raices*****
Ingrese valor inicial del método (mayor a cero):
3
Ingrese la presición de la respuesta:
0.001
La solución es: 2.000000, después de 4 iteraciones
```

Figura 5: Respuesta 2

```
*****Método de Newton-Raphson para buscar raices*****
Ingrese valor inicial del método (mayor a cero):
7
Ingrese la presición de la respuesta:
0.001
La solución es: 2.000000, después de 5 iteraciones
```

Figura 6: Respuesta 3

```
*****Método de Newton-Raphson para buscar raices*****
Ingrese valor inicial del método (mayor a cero):
0.01
Ingrese la presición de la respuesta:
0.001
La solución es: 2.000000, después de 6 iteraciones
```

Figura 7: Mensaje de error