Escuela de Ciencias Físicas y Matemáticas – Laboratorio de Simulación

Segundo Examen Parcial, Dayrin Idalia Cardona Crúz, 201804753

Primer problema

1) Enunciado

Un atleta necesita pasar la evaluación física para poder ingresar a la selección de su equipo. Como mínimo debe poder realizar diez sentadillas con 250 kg de peso. 10 meses antes de la evaluación sufre un desgarre en el cuadrícep, derecho lo que le permitió obtener la siguiente evolución.

Semana	Peso (kg)
1	20
2	26
3	31
4	38
5	45
6	49
7	54

Solicitan generar un programa que cumpla con las siguientes condiciones:

- Una gráfica que compare los valores tabulados y la recta que mejor aproxima el crecimiento.
- Estimar el peso que logra levantar el atleta después de 5 meses, este logra pasar la prueba para ingresar al equipo.

2) Metodología

Para resolver este problema se utilizará el método numérico de mínimos cuadrados. El método de mínimos cuadrados se utiliza para encontrar las curvas que mejor aproximen un conjunto de pares. En este caso, los datos tabulados se van a aproximar a una recta, es decir, se hará una regresión lineal utilizando el modelo de mínimos cuadrados.

La pendiente de la recta se puede aproximar por medio de la siguiente ecuación:

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

El intersecto de la recta con el eje *x* se puede aproximar con:

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

Finalmente, el coeficiente de correlación se determina con:

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

El cuadrado del coeficiente de correlación, el coeficiente de determinación, indica que mientras este coeficiente se acerca más al 1, más se parece al modelo matemático en proceso, en este caso, la aproximación a una recta.

3) Variables de entrada y salida

Las variables de entrada son vectores de tipo entero, kilogramos y semanas, pero debido a la forma del modelo matemático para encontrar las variables de salida que utiliza fracciones, es preferible ponerlas como de tipo float desde mucho antes. Por lo tanto, las variables de salida son la recta *y/x*, la pendiente *m* de la recta y el intersecto *b* con el eje *x/t*, también son de tipo float.

4) Pseudocódigo

INICIO

void main()

Definir x[],y[]

Definir n

Definir prototipos de funciones: imprimirvector(), sumvector(), sumvectormul()

Definir b, m, r

Escribir vector t en pantalla tomando imprimirvector(t)

Escribir vector x en pantalla tomando imprimirvector(x)

Tomar m = (n*sumvectormul(t,x) - sumvector(t)*sumvector(x))/(n*sumvectormul(t,t)-sumvector(t)*sumvector(t))

Tomar b= (sumvector(x)-m*sumvector(t))/n;

Tomar r= (n*sumvectormul(t,x)-sumvector(t)*sumvector(x))/sqrt((n*sumvectormul(t,t)-sumvector(t)*sumvector(t))*(n*sumvectormul(x,x)-sumvector(x)*sumvector(x)));

Escribir "La recta que mejor aproxima es"

Escribir "x= mt + b" y "r"

Definir x1=m*20+b

Escribir x1 el peso que logra levantar el jugador tras 5 meses

Fin void main()

void imprimirvector(float a[])

Escribir "Los datos del vector son"

Desde i=0, i<tamaño del arreglo

Escribir a en la posición i

Aumentar i=i+1

Fin ciclo

Escribir \n

Fin imprimirvector(float a[])

float sumvectormul(float a1[], float a2[])

Definir variable de almacenamiento rep

Desde i=0, i<tamaño del arreglo

Tomar rep = rep + a1*a2 en la posición i

Aumentar i=i+1

Fin ciclo

Regresar rep

Fin float sumvectormul(float a1[], float a2[])

float sumvector(float a[])

Escribir “Los datos del vector son”

Desde i=0, i<tamaño del arreglo

Tomar rep = rep + a en la posición i

Aumentar i=i+1

Fin ciclo

Regresar rep

Fin sumvector(float a[])

FIN

5) Código

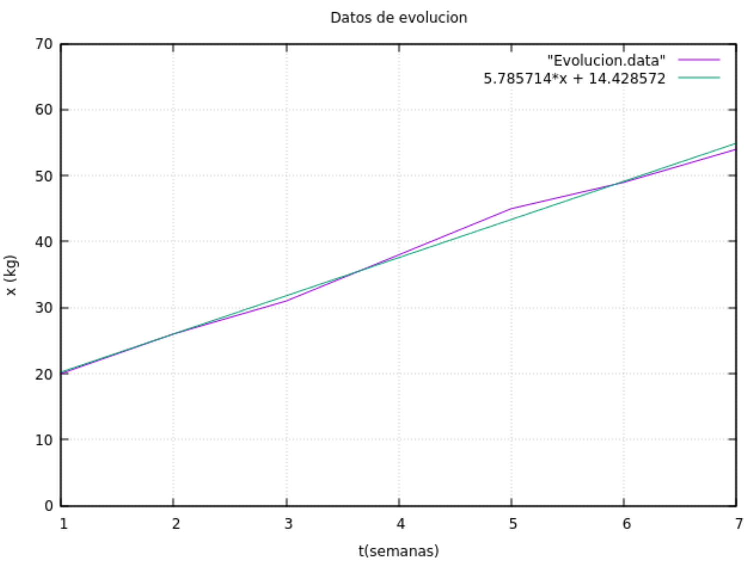
El código se puede encontrar en: https://github.com/DayrinCardona/LabSimu1S2021DC/blob/main/Segundoparcial/NewtonRaphson.c

6) Respuestas

Los datos tabulados y la recta que mejor aproxima el crecimiento fueron comparados mediante una gráfica realizada en Gnuplot. Para generar ambas, se utilizaron los comandos:

```
gnuplot> reset
gnuplot> set title "Datos de evolucion"
gnuplot> set xlabel "t(semanas)"
gnuplot> set ylabel "x (kg)"
gnuplot> set grid
gnuplot> set data file separator ","
                  ^
         unrecognized option - see 'help set'.

gnuplot> set datafile separator ","
gnuplot> plot [x=0:7] [y=0:100] "Evolucion.data" with points
QApplication: invalid style override passed, ignoring it.
gnuplot> plot [x=0:7] [y=0:100] "Evolucion.data" with lines
gnuplot> plot [x=0:10] [y=0:100] "Evolucion.data" with lines
gnuplot> plot [x=0:10] [y=0:100] 5.785714*x + 14.428572
gnuplot> plot [x=0:10] [y=0:100] "Evolucion.data" with lines, 5.785714*x + 14.428572 with lines
gnuplot> plot [x=0:8] [y=0:100] "Evolucion.data" with lines, 5.785714*x + 14.428572 with lines
gnuplot> plot [x=1:7] [y=0:70] "Evolucion.data" with lines, 5.785714*x + 14.428572 with lines
```



En donde la recta morada representa los datos tabulados y la recta azul representa la recta que mejor aproxima el crecimiento:

$x = 5.785714t + 14.428572$

O:

$y = 5.785714x + 14.428572$

El peso que el jugador logra levantar después de 5 meses, es decir, 20 semanas, es 130.142853 kg. Sí logra pasar la prueba del equipo en 10 meses.

Segundo problema

1) Enunciado

Utilizando un método numérico, encuentre una raíz de la ecuación

$$f(x) = \arcsin(x)$$

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

2) Metodología

Para resolver este problema se utilizará el método numérico de Newton Raphson. La solución iterativa de este problema está dado por:

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

Donde *f* es continua en el intervalo *[a,b]* y pertenece a los reales.

Para hallar las soluciones a la ecuación utilizando el método de Newton Raphson hay que escoger una x inicial relativamente cercana al cero de la ecuación, el cual será el valor supuesto del cero. Arcsin(x) solamente tiene una raíz, cuando x=0, y esto es fácilmente observable a partir de la gráfica. Además, la gráfica revela que arcsin(x) tiende a infinito cuando alcanza los puntos -1 y 1, donde los números están en radianes. Entonces desde el inicio es posible escoger un punto entre -1 y 1 pero no igual a -1 o 1 para poder hallar la raíz de la ecuación. Así, el método de Newton Raphson linealiza la función por la recta tangente a ese valor supuesto. Finalmente, se realizan sucesivas iteraciones hasta que el método haya convergido lo suficiente.

3) Variables de entrada y salida

Los datos de entrada x_inicial, tolerancia e iter, son de tipo double para asegurar la precisión suficiente de los datos de salida, los cuales serán xS y Aiter. "xS" será la solución o la raíz de la función arcsin(x).

4) Pseudocódigo

INICIO

void main ()

    Definir f(x), f'(x) y NR

    Definir x_i, tolerancia, xS

    Definir iter, Aiter

    Leer valor x_inicial, tolerancia, iteraciones

    Newton-Raphson(x_inicial, tolerancia, iter, &Aiter, &xS)

        Definir x_i, x, dif

        Definir i=1

        Sustituir x_i = x0

        Tomar x=x_i-f(x_i)/f'(x_i)

        Tomar dif=x-x_i

        Si dif < 0 entonces

```
                    dif = -dif
                    de lo contrario
                    dif = dif
            Fin Si
            Mientras dif > tolerancia y i<=maxi pasos siguientes:
                    Sustituir x_i = x
                    tomar x=x_i– f(x_i)/f'(x_i)
                    Tomar dif=x-x_i
                    Si dif < 0 entonces
                            dif = -dif
                            de lo contrario
                            dif = dif
                     Fin Si
                     Aumentar i=i+1
            Fin Mientras
            Guardar en la memoria *sol = x
            Guardar en la memoria *iterac=i
        Fin Newton-Rapshon(x_inicial, tolerancia, iter, &Aiter, &xS)
        "Si Aiter==iter entonces
                Escribir "No hay solución después de iter iteraciones"
                De lo contrario
                Escribir "Luego de Aiter iteraciones, la solución es"
        Fin Si
fin void main()
float f(float x)
        Devolver asin(x)
fin f(float x)
float df(float x)
```

Devolver derivada de asin(x) = 1/sqrt(1-x*x)

fin df(float x)

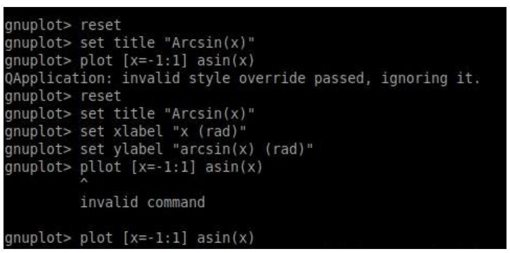
FIN

5) Código

El código del programa se puede encontrar en:
https://github.com/DayrinCardona/LabSimu1S2021DC/blob/main/Segundoparcial/Minimoscuadrados.c

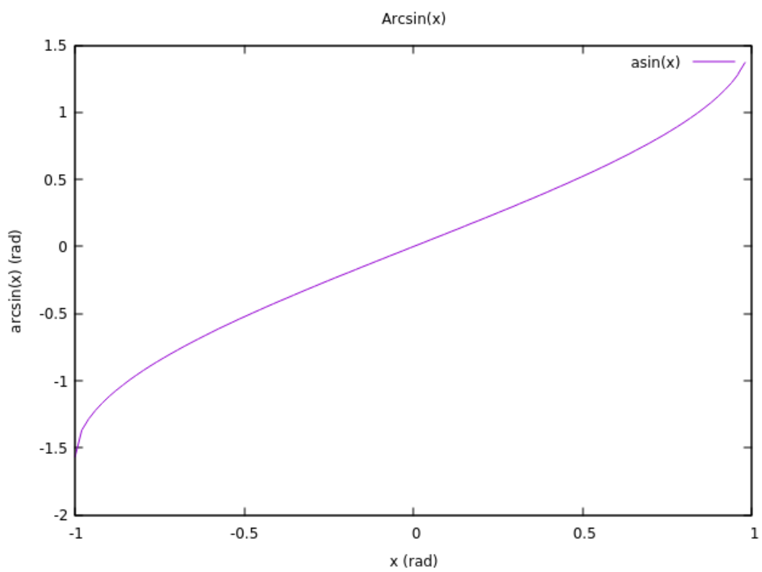
6) Resultados

Utilizando Gnuplot y los siguientes comandos se obtuvo una gráfica de la función asin(x), la cual viene en por defecto en el paquete de Gnuplot.



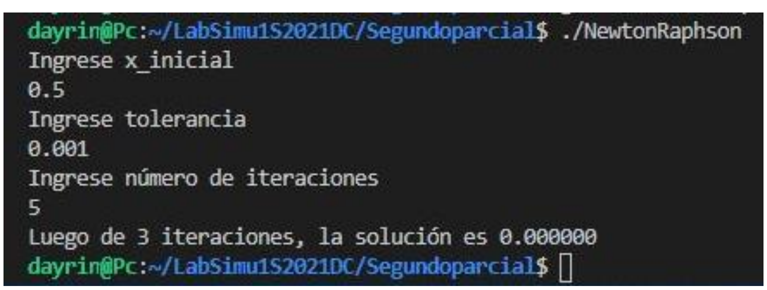
```
gnuplot> reset
gnuplot> set title "Arcsin(x)"
gnuplot> plot [x=-1:1] asin(x)
QApplication: invalid style override passed, ignoring it.
gnuplot> reset
gnuplot> set title "Arcsin(x)"
gnuplot> set xlabel "x (rad)"
gnuplot> set ylabel "arcsin(x) (rad)"
gnuplot> pllot [x=-1:1] asin(x)
         ^
         invalid command

gnuplot> plot [x=-1:1] asin(x)
```



A partir de la gráfica, es posible observar que solamente existe una raíz para la raíz en x=0, por lo tanto, cuando el programa pide x_inicial el usuario puede colocar cualquier valor cercano a 0 siempre y cuando esté dentro del intervalo cerrado -1 y 1.

Los resultados del programa para x_inicial = 0.5, tolerancia = 0.001 e iteraciones=5 es:



```
dayrin@Pc:~/LabSimu1S2021DC/Segundoparcial$ ./NewtonRaphson
Ingrese x_inicial
0.5
Ingrese tolerancia
0.001
Ingrese número de iteraciones
5
Luego de 3 iteraciones, la solución es 0.000000
dayrin@Pc:~/LabSimu1S2021DC/Segundoparcial$
```