

Practica 8

Esta practica consta de 5 ejercicios. Aquí escribiré los códigos fuente(en azul) y comentare los resultados obtenidos al ejecutarlos(en negro). Los archivos *.cpp y los archivos *.exe están en el CD adjunto.

Ejercicio 1;

En este primer ejercicio trabajaremos con una función $f(x)$ trigonométrica modulada por una exponencial;

$$f(x) = a \sin(x) + b \exp(-x)$$

a) Primero la representaremos con Gnuplot para los valores de $a = 2$ y $b = 20$.

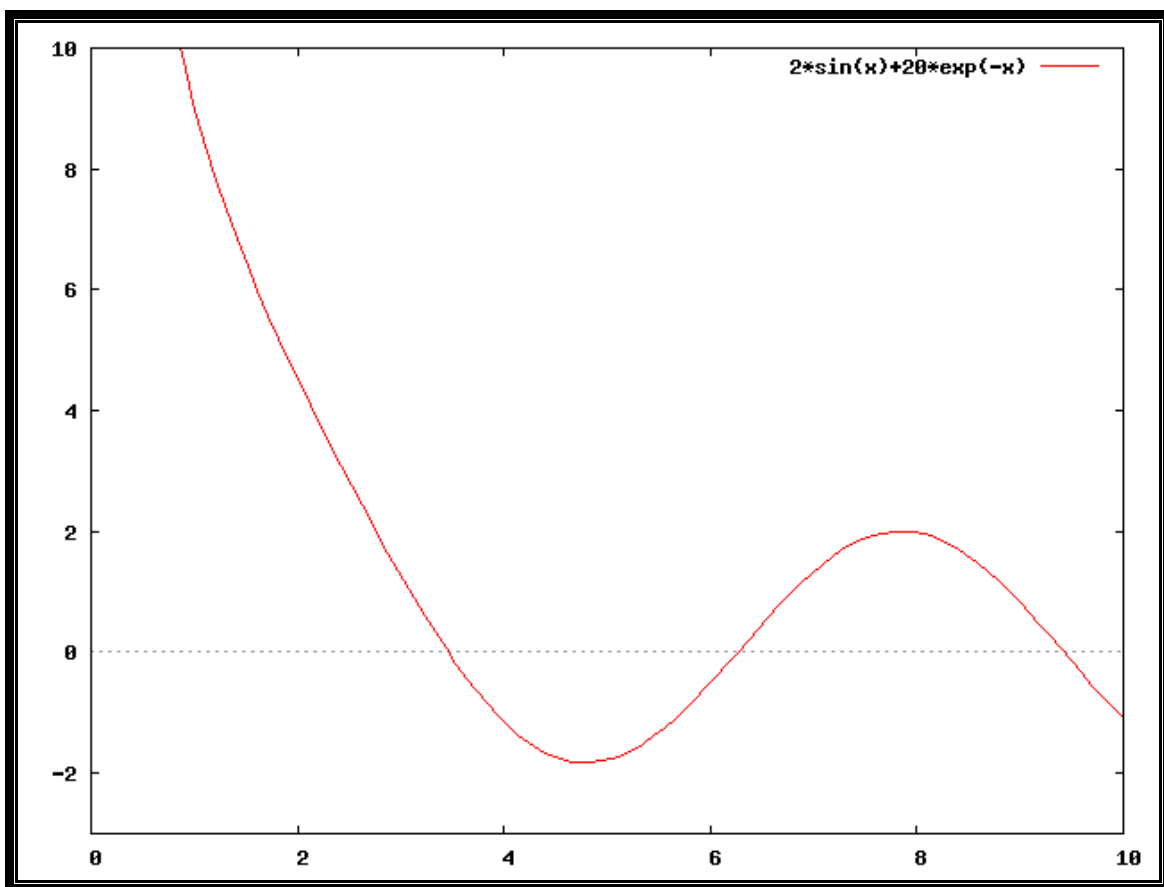


Fig 1; función $f(x) = 2 \sin(x) + 20 \exp(-x)$.

b) Ahora generamos 30 puntos con el programa xgendat.cpp en el intervalo $[0,10]$ con un error de los puntos $s = 1$ y un error de dispersión, e_d , nulo. Representamos el resultado con Gnuplot.

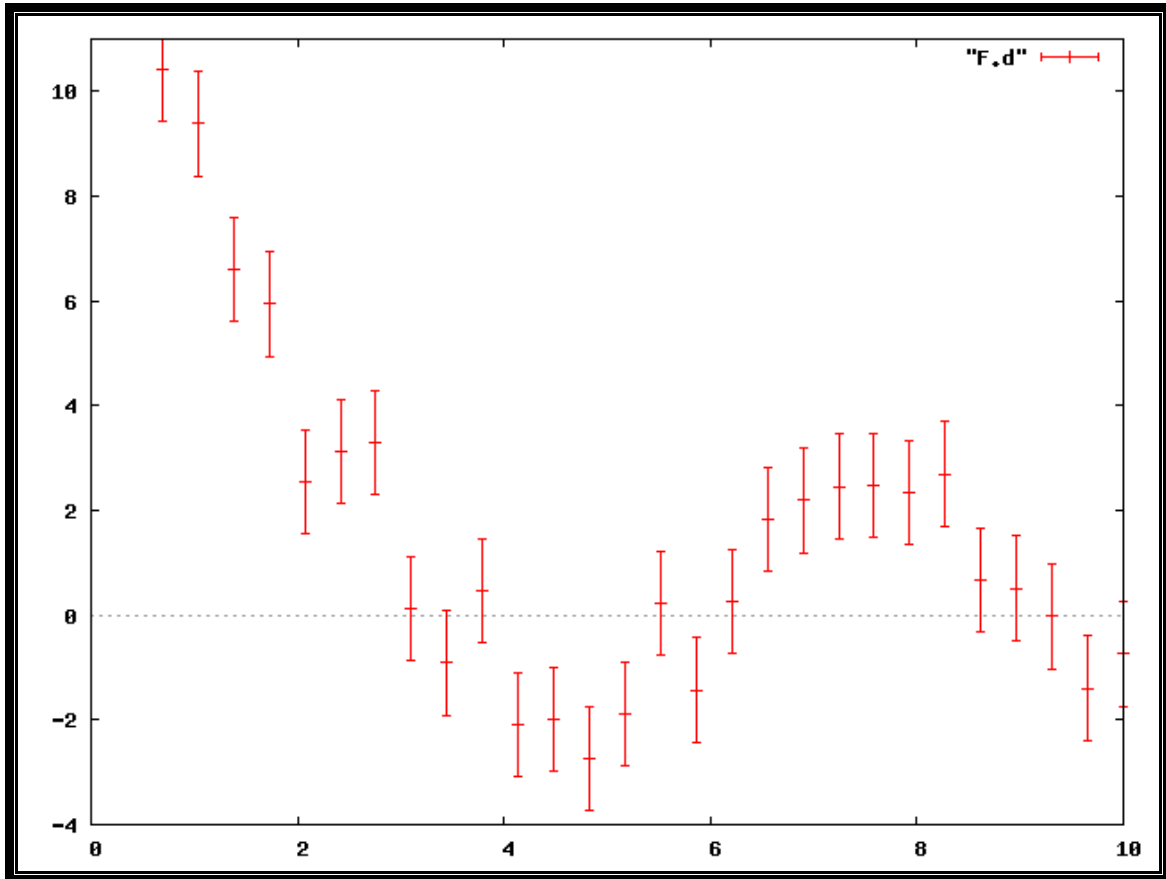


Fig 2; representación de los puntos generados con el programa xgendat.cpp para una $f(x) = 2 \sin(x) + 20 \exp(-x)$ con $S = 1$ y un error de dispersión nulo.

Ajustamos estos puntos a una función tipo $f(x) = a \sin(x) + b \exp(-x)$ para calcular el valor de los parámetros a y b con el programa xajuste.cpp. Obtenemos que;

$$a = 2.111541$$

$$b = 18.93166$$

$$\mathbf{c}^2 = 23$$

Grados de libertad $\mathbf{n} = 28$

$$\mathbf{c}^2/\mathbf{n} = 0.82$$

Calculamos la desviación típica de la distribución \mathbf{c}^2 a partir del valor de \mathbf{n} sabiendo que;

$$S(\mathbf{c}^2) = (2\mathbf{n})^{1/2} = 7.483314774$$

Ahora podemos valorar la calidad del ajuste en función del valor de la distribución \mathbf{c}^2 .

Si nuestro valor de \mathbf{c}^2 se encuentra en el intervalo $[\mathbf{n} - (2\mathbf{n})^{1/2}, \mathbf{n} + (2\mathbf{n})^{1/2}]$ el ajuste es aceptable con 68% de probabilidad. En nuestro caso tenemos que este intervalo (teniendo en cuenta el valor de \mathbf{n}) es $[20.51, 35.48]$. Nuestro valor de \mathbf{c}^2 se encuentra dentro de ese intervalo. Un ajuste de estas características es aceptable siempre.

Podemos decir que la curva pasa a través de todos los puntos, incluyendo su barra de error..

- c) Al variar la semilla para inicializar el generador de números aleatorios también varía el resultado de ajuste. Lo que tenemos es una distribución probabilística de datos, el valor de χ^2 está, en el peor de los casos, dentro del intervalo $[n - 3(2n)^{1/2}, n + 3(2n)^{1/2}]$. Al variar el entero que inicializa el generador de números aleatorios variamos las condiciones iniciales de medida, es decir, podemos obtener cualquier valor de la función χ^2 dentro de ese intervalo lo que significa, a su vez, obtener un mejor o peor ajuste.

Por ejemplo, al calcular el valor de los parámetros a y b inicializando el generador de números aleatorios con un número al azar obtenemos que;

$$a = 1.8558544$$

$$b = 19.02807$$

$$\chi^2 = 50$$

$$\text{Grados de libertad } n = 28$$

$$\chi^2/n = 1.8$$

Podemos observar a simple vista que este ajuste es peor que el anterior

- d) Ahora ejecutaremos el programa xchi2.cpp y representaremos el resultado con Gnuplot, obtenemos que;

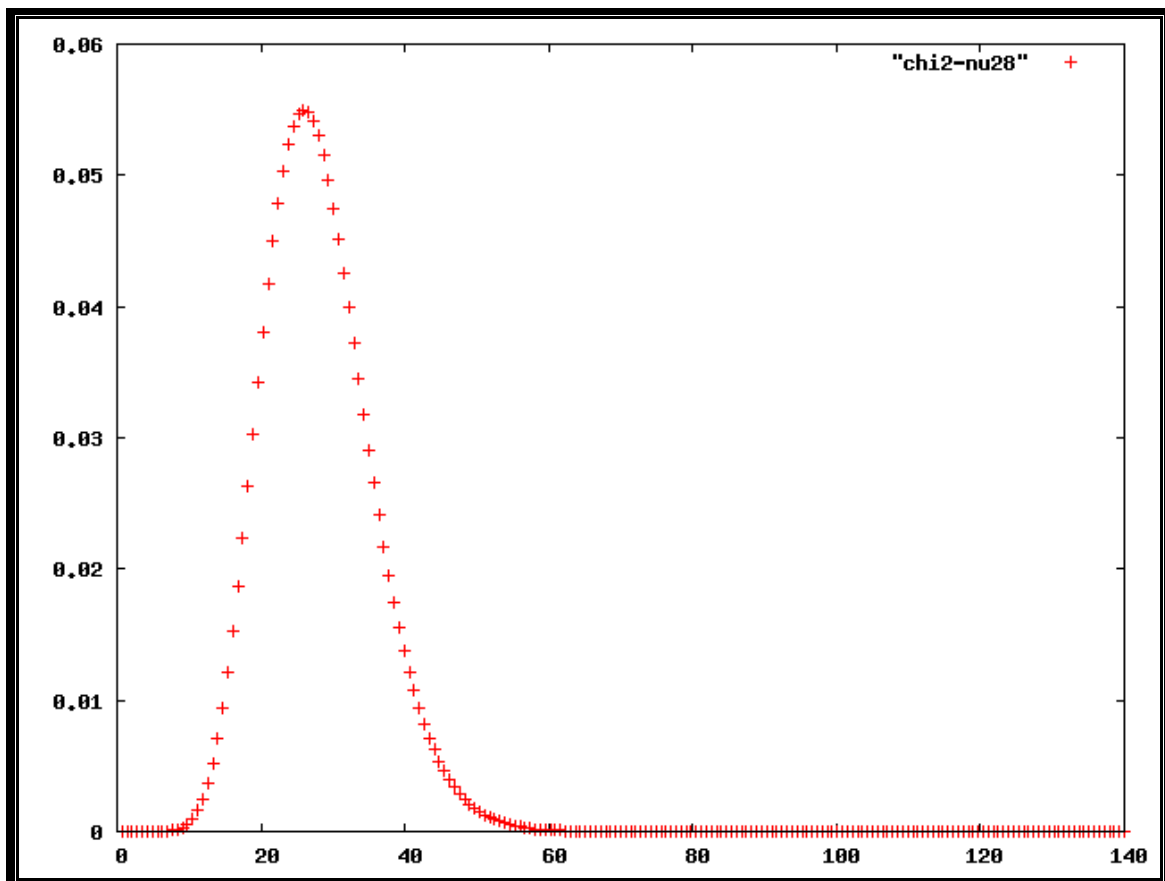


Fig 3; función χ^2 para $n = 28$.

Al obtener todos los valores posible de la función \mathbf{c}^2 obtenemos una distribución de valores alrededor de un máximo que en nuestro caso es el valor esperado de la función \mathbf{c}^2 , calculado a partir de la diferencia entre el numero de datos utilizados para realizar el ajuste menos el numero de parámetros del ajuste.

Podemos obtener un valor u otro de la función \mathbf{c}^2 en función de nuestra medida y por tanto un mejor o peor ajuste.

Ejercicio 2;

En este ejercicio trabajáremos con una función tipo $f(x)$;

$$f(x) = ax^2 + bx + c$$

Donde $a = 0.1$, $b = 5$ y $c = 0.5$. Generaremos 20 puntos con el programa xgendat.cpp en el intervalo $[0,40]$ con una $\mathbf{s} = 20$ y un error de dispersión $\mathbf{e}_d = 5$.

Primero representaremos la función con Gnuplot, obtenemos que;

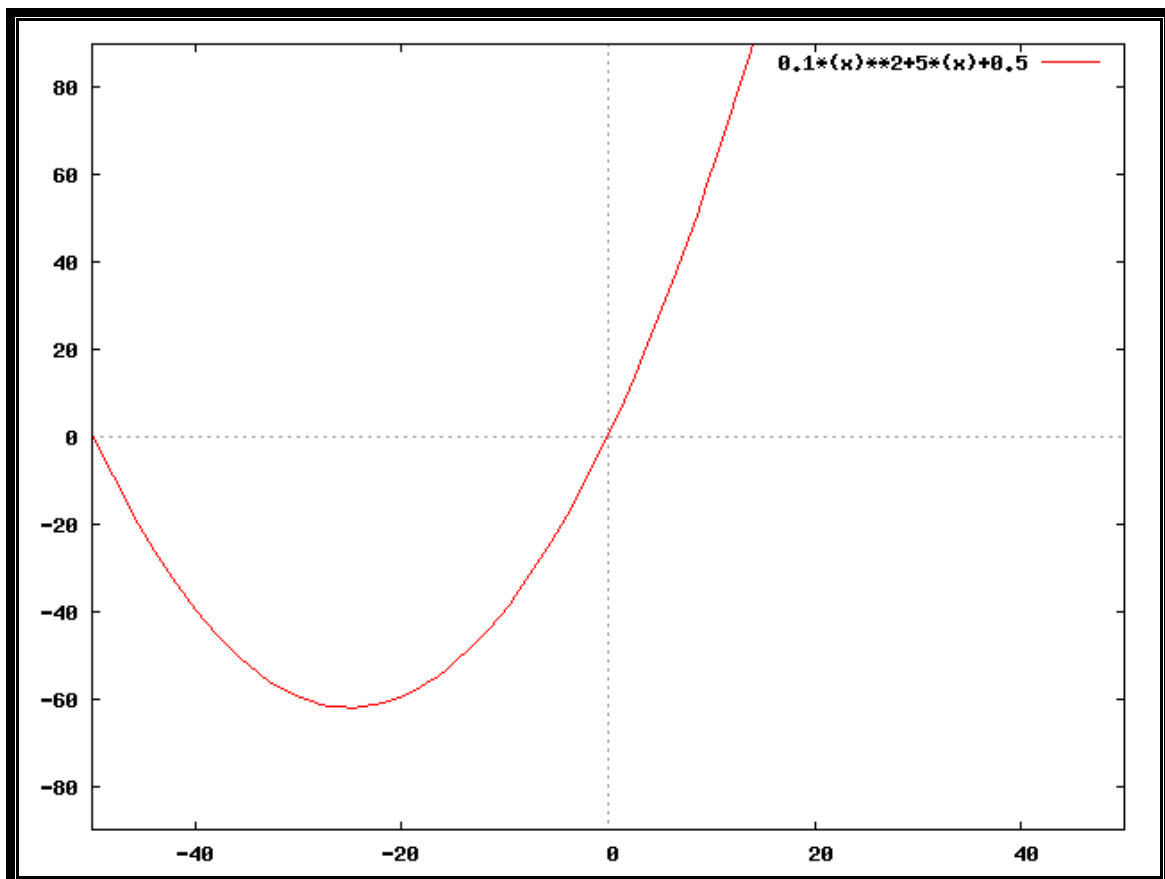


Fig 4; $f(x) = 0.1x^2 + 5x + 0.5$ en el intervalo $[-50,50]$, aunque el intervalo de trabajo será $[0,40]$, con esta representación podemos observar mejor el comportamiento global de la función.

Ahora ajustaremos el fichero *parabola.d* a dos modelos distintos, una parábola y una recta.

a) Parábola;

$$a = 0.092059084$$

$$b = 5.0439539$$

$$c = 8.9575301$$

$$c^2 = 14$$

Grados de libertad $n = 17$

Calculamos la desviación típica de la distribución c^2 a partir del valor de n sabiendo que;

$$s(c^2) = (2n)^{1/2} = 5.8309551895$$

$$c^2/n = 0.84$$

Otra forma de valorar la calidad del ajuste es comprobando que el valor de la magnitud $\chi^2/n \approx 1$. En este caso se trata de un ajuste aceptable.

b) Recta;

$$a = 8.7735242$$

$$b = -15.28048$$

$$c^2 = 25$$

Grados de libertad $n = 18$

Calculamos la desviación típica de la distribución c^2 a partir del valor de v sabiendo que;

$$s(c^2) = (2n)^{1/2} = 6$$

$$c^2/n = 1.4$$

En este caso el ajuste es de menos calidad, pero se puede calificar de aceptable bajo un punto de vista probabilístico. Si no conociéramos la forma explícita de la función $f(x)$ sería difícil tener que elegir entre uno de los dos ajustes. La razón de este parecido es obvia a la vista de la *figura 5*, el comportamiento de la parábola en ese intervalo es casi lineal, además contamos con una desviación típica y un error de dispersión suficientemente altos como para que sea muy difícil excluir a alguna de las dos representaciones.

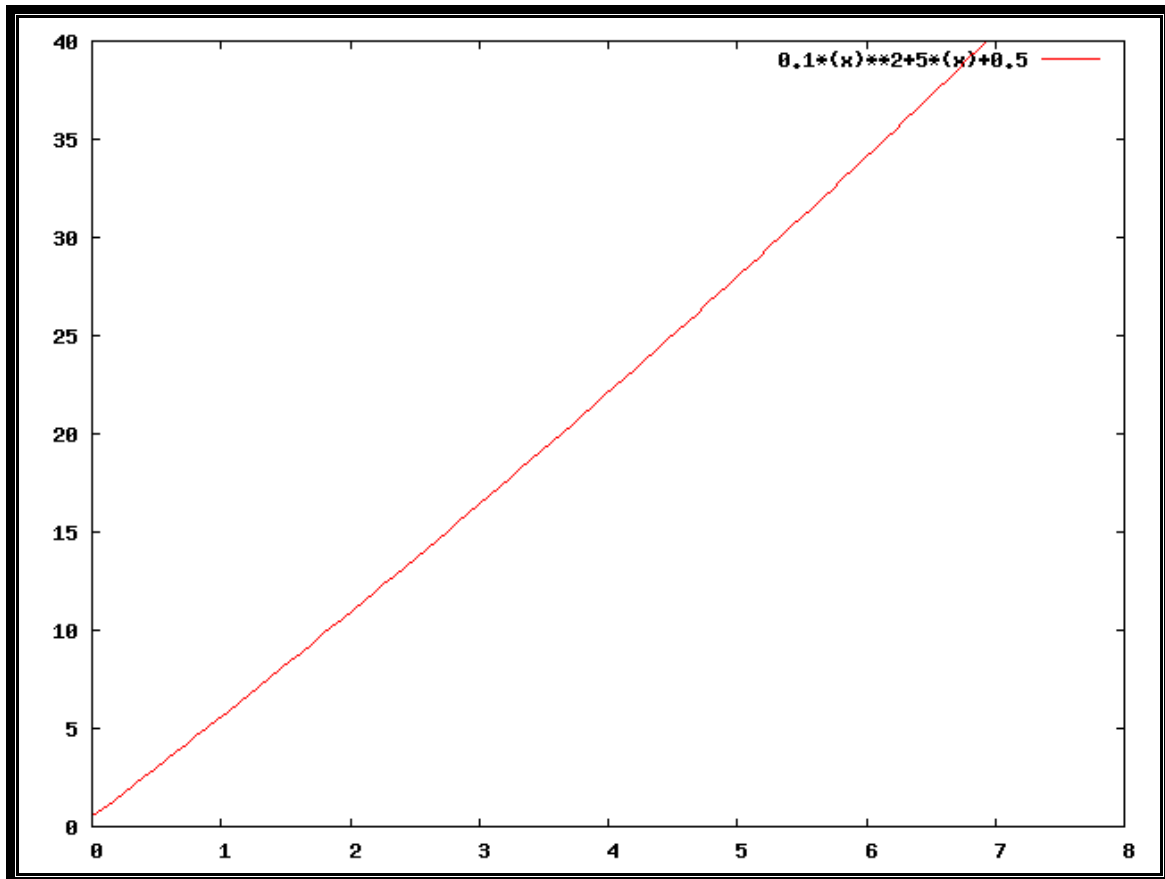


Fig 5; $f(x) = 0.1x^2 + 5x + 0.5$ en el intervalo $[0,40]$.

Ahora repetiremos el proceso anterior pero con una $s = 4$ y un error de dispersión $e_d = 1$.
 En este caso tenemos valores mas ajustados desde un principio, deberíamos poder obtener un ajuste diferenciado entre parábola y recta y así elegir entre uno de los dos.

c) Parábola;

$$a = 0.098411693$$

$$b = 5.0088023$$

$$c = 2.1914711$$

$$c^2 = 14$$

Grados de libertad $n = 17$

Calculamos la desviación típica de la distribución c^2 a partir del va lor de n sabiendo que;

$$s(c^2) = (2n)^{1/2} = 5.8309551895$$

$$c^2/n = 0.84$$

Se trata de un ajuste aceptable.

d) Recta;

$$a = 8.9957353$$

$$b = -23.719152$$

$$c^2 = 3.1E2$$

Grados de libertad $n = 18$

Calculamos la desviación típica de la distribución c^2 a partir del valor de n sabiendo que;

$$s(c^2) = (2n)^{1/2} = 6$$

$$c^2/n = 17$$

En este caso la magnitud c^2/n es mucho mayor que la unidad, este ajuste es del todo inaceptable. En este caso los ajustes difieren tanto, en comparación con los ajustes anteriores, debido a que el error de los puntos a partir de los cuales hemos ajustado una parábola o una recta eran mucho mas concretos. En los apartados *a)* y *b)* la desviación típica era muy grande, por tanto los dos modelos podían contener a esos puntos dentro del intervalo de error que se había propuesto.

Ejercicio 3;

La abundancia de una muestra de partículas elementales que se desintegra de acuerdo a una vida media t puede escribirse mediante una función exponencial de la siguiente forma;

$$N(t) = N_0 \exp(-t/t)$$

- a)* Primero representaremos gráficamente el conjunto de datos *tau.d* que se corresponde a un experimento ideal de la abundancia de una muestra de partículas elementales en función del tiempo.

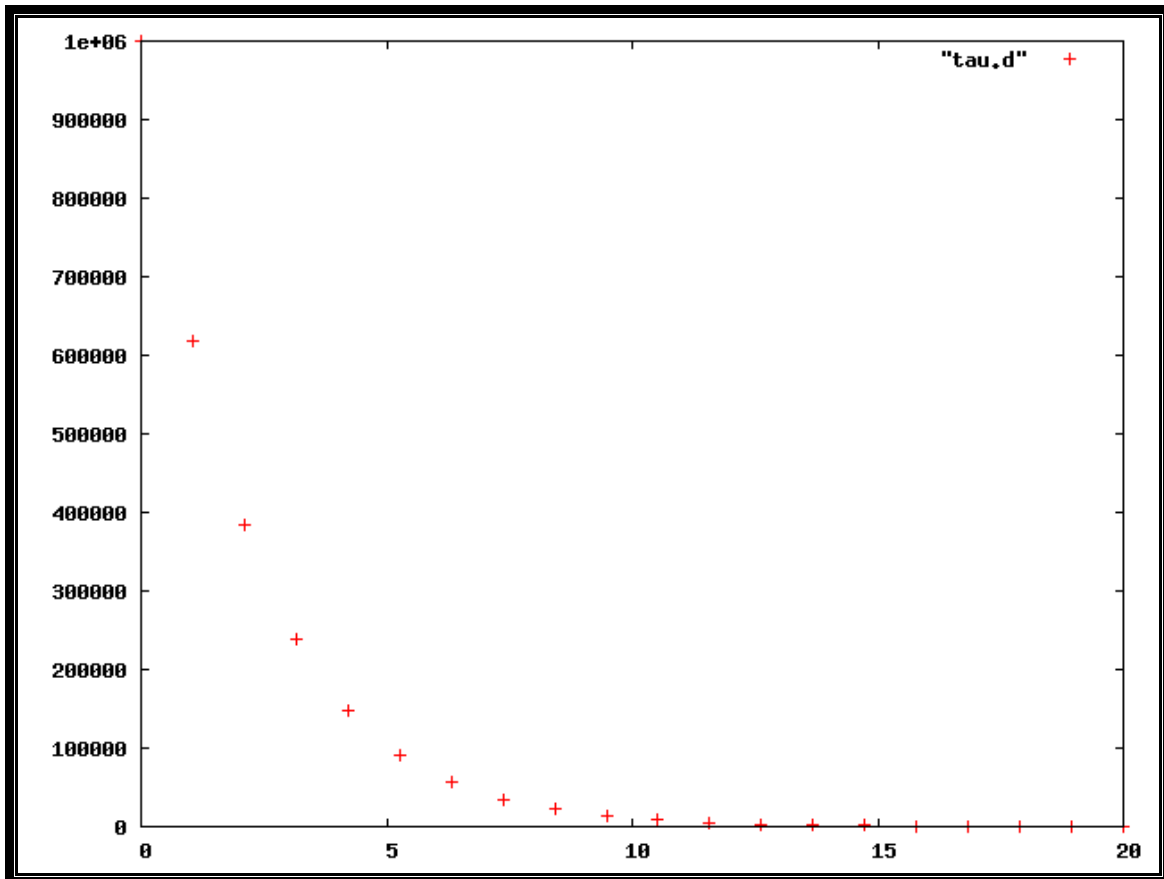


Fig 6; representación del fichero tau.d, se puede observar el comportamiento exponencial decreciente.

b) La función del problema no es lineal, pero se puede convertir en una recta en quedando de la siguiente forma;

$$y = at + b$$

Donde $a = -1/t$, $b = \ln N_0$ y la variable $y = \ln N(t)$.

Así, si tenemos un fichero de datos de la variable $N(t)$ en función de t , solo tenemos que modificar la columna $N(t)$ poniendo el logaritmo neperiano de cada uno de los valores de $N(t)$ y ajustar ese fichero a una recta, obteniendo así los parámetros a y b a partir de los cuales podemos calcular los parámetros N_0 y t . Y eso hemos hecho en el programa xp8.cpp .Partiendo de la base del programa xajuste.cpp . El código de xp8.cpp queda de la siguiente forma;

```
#include <iostream>
#include <iomanip>
#include <string>
#include <mcldos.h>
#include <chi2.h>
```



```

RVector recta(double x){

    RVector ff(2);
    ff(1) =x;
    ff(2) = 1.;
    return ff;
}

int main ()
{
    string fdatos;
    cout << "fichero de datos a ajustar? "; cin >> fdatos;

    int m = 2;
    CDatos cd;
    cd.lee(fdatos);
    MCdos xmc(m);
    xmc.eDatos(cd);

    cout << "Este programa ajusta una recta modelo"
    <<" a un conjunto de "<<cd.dimension()<<" puntos" << endl;
    int imodel=1;
    if (imodel == 1)
        xmc.eModelo(recta);

    else
        error("", CEPROCESO);
    xmc.eInfo("S");
    RVector par = xmc.param();

    //Modificacion.Imprimo en pantalla los parametro a y b por separado.
    cout << "Parámetros a y b obtenidos por el ajuste a= " << par(1) <<endl;
    cout << "y b= " << par(2) <<endl;
    cout << "Ahora calcularemos los parametros tau y No a partir de estos " <<endl;    cout
    <<"parametros a y b sabiendo que a=-1/tau y No=ln b ." <<endl;
    cout <<"Obtenemos que el valor de tau es, tau= " << -1/(par (1)) <<endl;
    cout <<"Obtenemos que el valor de No es No= " << exp(par (2)) <<endl;

    return 0;
}

```

Obtenemos los siguientes resultados;

$$a = -53087.2 \text{ segundos}^{-1}$$

$$b = 983937$$

$$t = 1.88369_{E-5} \text{ segundos}$$

No podemos obtener el valor de N_0 , ya que para obtenerlo deberíamos poder calcular el valor de la exponencial de un numero muy grande. De todas formas el valor de N_0 es el valor de $N(t)$ cuando $t = 0$

que sabemos que es aproximadamente del orden de 10^{-6} (ver figura 6). A partir de estos resultados obtenemos que la vida media de la partícula es de, aproximadamente, $2 \cdot 10^{-5}$ segundos.

Se trata de una muestra de un mesón estable K_L^0 con una masa de $3.52 \cdot 10^{-6}$ eV y esta t corresponde a una desintegración en la cual, un K_L^0 decae en forma de un electrón, un positrón y un fotón;

$$K_L^0 \rightarrow e^+ e^- \gamma$$

Ejercicio 4;

En este ejercicio revisaremos las ecuaciones expuestas en la sección 11.5.1 y las modificaremos para que nos sirvan para ajustar cualquier función lineal de dos parámetros de la forma;

$$y = f_1(x)p_1 + f_2(x)p_2$$

Podremos calcular el vector de parámetros $P = (p_1, p_2)$ a partir de la relación;

$$P = F^{-1}W$$

Donde W es el vector (columna) de componentes $W = (\langle f_1(x)y \rangle, \langle f_2(x)y \rangle)$, donde

$\langle f_i(x)y \rangle = S f_i(x_j)y_j/(s_j)^2$ desde $i=1,2$ y $j=1 \dots m$, donde m es el número de puntos en el intervalo y F^{-1} es la inversa de la matriz F , formada por dos vectores columna F_1, F_2 ;

$$F_1 = (\langle f_1^2(x) \rangle, \langle f_1(x)f_2(x) \rangle) \\ F_2 = (\langle f_1(x)f_2(x) \rangle, \langle f_2^2(x) \rangle)$$

Donde $\langle f_i^2(x) \rangle = S f_i(x_j)f_i(x_j)/(s_j)^2$ desde $i=1,2$ y $j=1 \dots m$, donde m es el número de puntos en el intervalo y $\langle f_l(x)f_k(x) \rangle = S f_l(x_j)f_k(x_j)/(s_j)^2$ para $l, k=1,2$ y $l \neq k$.

No he conseguido implementar la función para comprobar el funcionamiento de este proceso simplificado de ajuste lineal.

Ejercicio 5;

En principio, debería comprobar el correcto funcionamiento del algoritmo anterior. Aunque no he conseguido programarlo, a partir del desarrollo de las ecuaciones de la sección 11.5.1 sabemos que el resultado debe ser el mismo.

Bibliografía;

Concepts of particle physics Volume I

Kurt Gottfried

Victor F. Weisskopf