Universidad Nacional de Ingeniería
Facultad de Ciencias

Física Computacional
CC063

Interpolacion y extrapolacion numerica y Ajuste de datos

Prof: J. Solano
2012-I

Introducción

La interpolación y extrapolación numérica es quizás una de las herramientas más utilizadas en aplicaciones numéricas a la Física.

La situación frecuente es de una función f de un conjunto de puntos $x_1,...,x_n$, donde falta una forma analítica.

La función f puede representar algunos puntos de datos del experimento o el resultado de una computación a gran escala de una magnitud física que no se puede convertir en una forma analítica sencilla.

A continuación, puede ser necesario evaluar la función f en algún punto x en el conjunto de datos $x_1,...,x_n$, pero donde x se diferencia de los valores tabulados. En este caso se trata de una interpolación. Si x está fuera nos quedamos con el problema más preocupante, de extrapolación numérica.

A continuación nos centraremos en dos métodos de interpolación y extrapolación, es decir, la interpolación polinómica y la extrapolación y el enfoque de qubic spline interpolación.

Introducción

Como ejemplo, considere una fuente radiactiva y un detector, el cual cuenta el número de desintegraciones. Con el fin de determinar la vida media de esta fuente, se cuenta el número de desintegraciones $N_0,N_1,N_2,...,N_k$ en instantes $t_0,t_1,t_2,...,t_k$. En este caso t sería la variable independiente, que se espera elegir de tal manera que sea adecuado para su problema. Sin embargo, lo que se mide es un conjunto discreto de pares de números (t_k,N_k) en el rango de (t_0,t_k).

Con el fin de extraer información de un experimento, nos gustaría ser capaces de encontrar una función analítica que nos daría N para cualquier punto arbitrario elegido t. A veces encontrar una función analítica es imposible, o aun si la función fuese conocida, consume demasiado tiempo calcularla o sólo nos interesa una pequeña región local de la variable independiente.

Introducción

Para ilustrar este punto, asumiremos que la fuente radiactiva es ^{241}Am, un emisor. Su vida media es de $\tau_{1/2}$=430 años. Es evidente que no se puede determinar $\tau_{1/2}$ mediante la medición de la misma.

Como el decaimiento es muy lento probablemente mediremos la actividad durante un período de tiempo largo, por ejemplo todos los lunes durante algunos meses. Después de cinco meses pararías y mirarías los datos.

Una pregunta que queremos responder es: ¿cuál era la actividad el Miércoles de la tercera semana? Como este día está dentro del rango (t_0,t_k) utilizarías técnicas de interpolación para determinar este valor.

Si, por el contrario, desea conocer la actividad de ocho meses desde la finalización de la medición, extrapolarias a este punto desde la anterior serie de medidas.

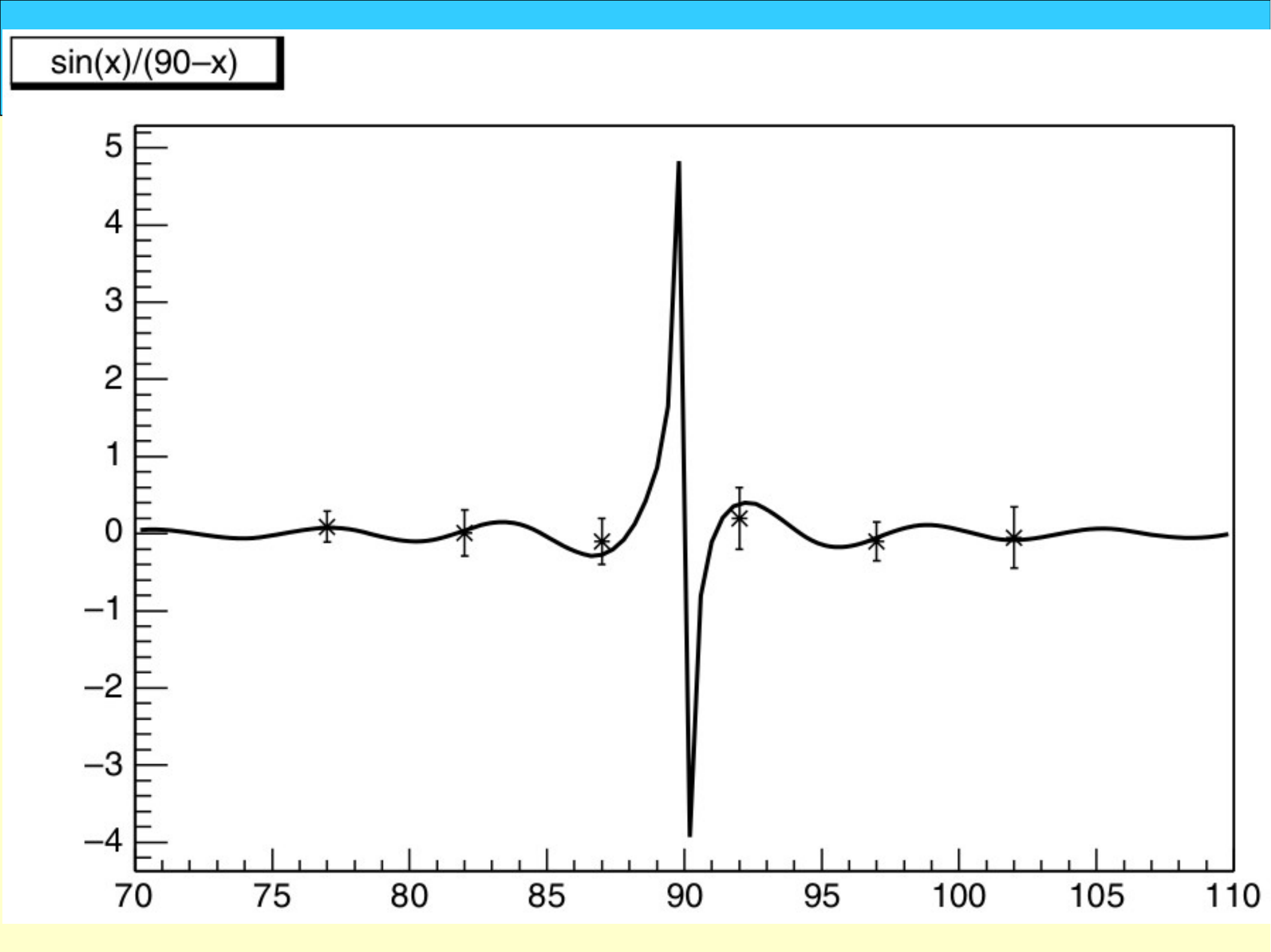
Interpolación y Extrapolación

La idea de la interpolación es seleccionar una función g(x) tal que $g(x_i)=f_i$ para c/pto i de datos y que esta función sea una buena aproximación para cualquier otro x que se encuentre entre los puntos de datos originales.

Pero, ¿qué podemos considerar como una buena aproximación a los datos originales, si no tenemos la función original? Debido a que los puntos de datos puede ser interpolados por un número infinito de funciones, se debe tener algún criterio o guía para seleccionar una función razonable.

En matemática hay teoremas de análisis de funciones, incluyendo interpolación con análisis de errores. Como regla estos métodos se basan en "suavidad" de las funciones interpoladas.

Obs: Debido a que se miden puntos discretos, hay que tener mucho cuidado con el espaciamiento de la variable independiente. Si estos puntos están demasiado lejos, se perderá la información en el medio y su predicción a partir de la interpolación sería inutil (ver fig).


sin(x)/(90−x)
5
4
3
2
1
0
−1
−2
−3
−4
70
75
80
85
90
95
100
105
110

Metodo de Interpolación de Lagrange

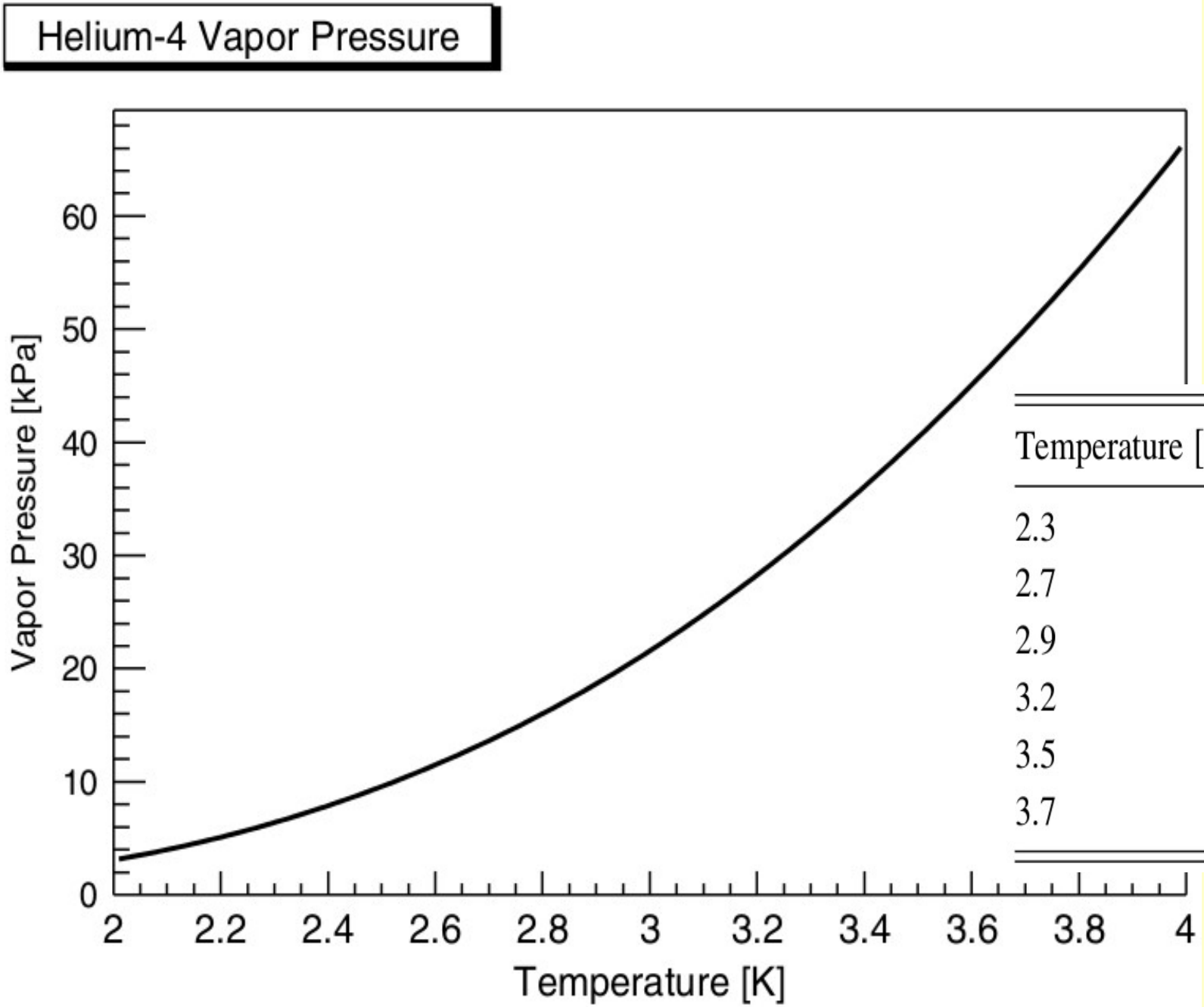
Como un primer paso en la interpolación nos fijamos en la interpolación de Lagrange. Esto nos ayudará a entender los principios de métodos de interpolación más complicadas como el algoritmo de Neville.

El método se basa en el hecho de que en un intervalo finito [a,b] una función f(x) siempre puede ser representada por un polinomio P(x). Nuestra tarea será encontrar este polinomio P(x), desde el conjunto de puntos $(x_i, f(x_i))$.

Si tenemos sólo dos de estos pares, la interpolación es clara y estamos seguros de que lo han utilizado en algun experimento de laboratorio antes, mirando hacia arriba los valores tabulados y determinando un punto entre dos valores indicados.

Echemos un vistazo a la presión de vapor de ^{4}He como una función de la temperatura (ver fig). En la literatura se encuentran valores de tabla como estos pero quieres saber la presion a 3^oK

Presión de vapor ^{4}He vs Temperatura



Temperature [K]	Vapor pressure [kPa]
2.3	6.38512
2.7	13.6218
2.9	18.676
3.2	28.2599
3.5	40.4082
3.7	49.9945

Metodo de Interpolación de Lagrange

La forma mas directa es hacer una interpolacion lineal $y=a+bx$ entre los dos puntos $(x_1, y_1(x_1))$ y $(x_2, y_2(x_2))$ dando:

$$b=\frac{y_2(x_2)-y_1(x_1)}{x_2-x_1}$$

$$a=y_1(x_1)-x_1\frac{y_2(x_2)-y_1(x_1)}{x_2-x_1}$$

Esto nos da:

$$y(x) = \frac{x-x_2}{x_1-x_2}y_1(x_1)+\frac{x-x_1}{x_2-x_1}y_2(x_2)$$

Usando la interpolacion da 21.871 kPa, mientras que la parametrizacion da 21.595 kPa.

Para mejorar el resultado se puede usar un polinomio de 2do grado y una interpolacion cuadratica (Ej: verificar).

$$y(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)}y_1(x_1)+\frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)}y_2(x_2)+\frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)}y_3(x_3)$$

Usando los puntos 2.8, 2.9 y 3.2 kPa, la presion para 3 ^{o}K es ahora 21.671 kPa.

El siguiente paso seria usar cuatro puntos y construir un polinomio de 3er grado.

Metodo de Interpolación de Lagrange

En general podemos escribir para la interpolacion en terminos de un polinomio:

$$P(x) = \sum_{k=1}^{N} \lambda_k(x) f(x_k)$$

donde:

$$\lambda_k(x) = \frac{\prod_{j=1 \neq k}^{N} (x - x_j)}{\prod_{j=1 \neq k}^{N} (x_k - x_j)}$$

que es la formula de la interpolacion de Lagrange. Este es un polinomio que tiene los valores $y_k(x_k)$ para k = 1, 2, … , N

Algoritmo de Neville

Buena manera de encontrar el polinomio de interpolación. En este método usamos interpolaciones lineales entre iteraciones sucesivas. El grado del polinomio será simplemente el número de iteraciones que hemos hecho.

Suponga que hay cinco mediciones de presión de vapor $P_i(T_i)$ en cinco diferentes temperaturas T_i y te gustaría encontrar la presión de vapor para una temperatura T intermedio. La primera versión es para determinar un polinomio lineal P_{ij} lineal entre puntos vecinos para todos los valores que tenemos:

$$P_{12} = \frac{1}{T_1 - T_2}((T - T_2)P(T_1) - (T - T_1)P(T_2))$$

$$P_{23} = \frac{1}{T_2 - T_3}((T - T_3)P(T_2) - (T - T_2)P(T_3))$$

$$P_{34} = \frac{1}{T_3 - T_4}((T - T_4)P(T_3) - (T - T_3)P(T_4))$$

$$P_{45} = \frac{1}{T_4 - T_5}((T - T_5)P(T_4) - (T - T_4)P(T_5))$$

Algoritmo de Neville

La siguiente iteracion sera nuevamente interpolacion lineal pero ahora entre esos puntos intermedios:

$$P_{123} = \frac{1}{T_1 - T_3}((T - T_3)P_{12} - (T - T_1)P_{23})$$

$$P_{234} = \frac{1}{T_2 - T_4}((T - T_4)P_{23} - (T - T_2)P_{34})$$

$$P_{345} = \frac{1}{T_3 - T_5}((T - T_5)P_{34} - (T - T_3)P_{45})$$

Nuestra interpolacion es ahora de grado 2. Podemos continuar hasta un polinomio de grado 4 que nos da el punto final P_{12345}

$$\begin{array}{lllll}
T_1 & P(T_1) & & & \\
 & & P_{12} & & \\
T_2 & P(T_2) & & P_{123} & \\
 & & P_{23} & & P_{1234} \\
T_3 & P(T_3) & & P_{234} & & P_{12345} \\
 & & P_{34} & & P_{2345} \\
T_4 & P(T_4) & & P_{345} & \\
 & & P_{45} & & \\
T_5 & P(T_5) & & &
\end{array}$$

```
//    Lagrange.cc: Interpolacion de Langrange de datos tabulados
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <cmath>
using namespace std;
double inter ( double xin[], double yin[], int end, double x);

int main() {
        double x, y;
        int  i,j,k; int end = 9;
        double xin[] = {0, 25, 50, 75, 100, 125, 150, 175, 200};  //Datos de entrada
        double yin[] = {10.6, 6, 45, 83.5, 52.8, 19.9, 10.8, 88.25, 4.7};

        FILE *output1;                                  // salvar datos en Lagr_input.dat
        output1 = fopen("Lagr_input.dat","w");
        for (k=0; k<9; k++)  fprintf(output1,"%f\t %f\n",xin[k],yin[k]);

        FILE *output2;                                  // salvar datos en Lagr.dat
        output2 = fopen("Lagr.dat","w");
        for ( k=0; k<=1000;k++){
                x = k*0.2 ;
                y = inter(xin, yin, end, x);
                fprintf(output2,"%f\t %f\n",x,y);
        }
        fclose(output1);
        fclose(output2);
}

double inter ( double xin[], double yin[], int end, double x) {
        double  lambda,y;  int i,j;
        y = 0.0;
        for ( i=1; i <= end; i++) {
                lambda = 1.0;
                for ( j=1; j<=end; j++) if (i != j)
                        {lambda *= ((x - xin[j-1])/(xin[i-1] - xin[j-1]));}
                        y += (yin[i-1] * lambda);
                }
        return y;
}
```

Interpolación Lineal

La idea es aproximar el valor de f en x, por un alinea recta pasando por los puntos x_n y x_{n+1} mas proximos a x:

$$g(x) = a_0 + a_1 x$$

donde a_0 y a_1 son coeficientes de funciones lineales, q se hallan del sistema de ecuaciones

$$g(x_j) = f_j = a_0 + a_1 x_j$$

$$g(x_{j+1}) = f_{j+1} = a_0 + a_1 x_{j+1}$$

Resolviendo el sistema para a_0 y a_1 la funcion g(x) toma la forma

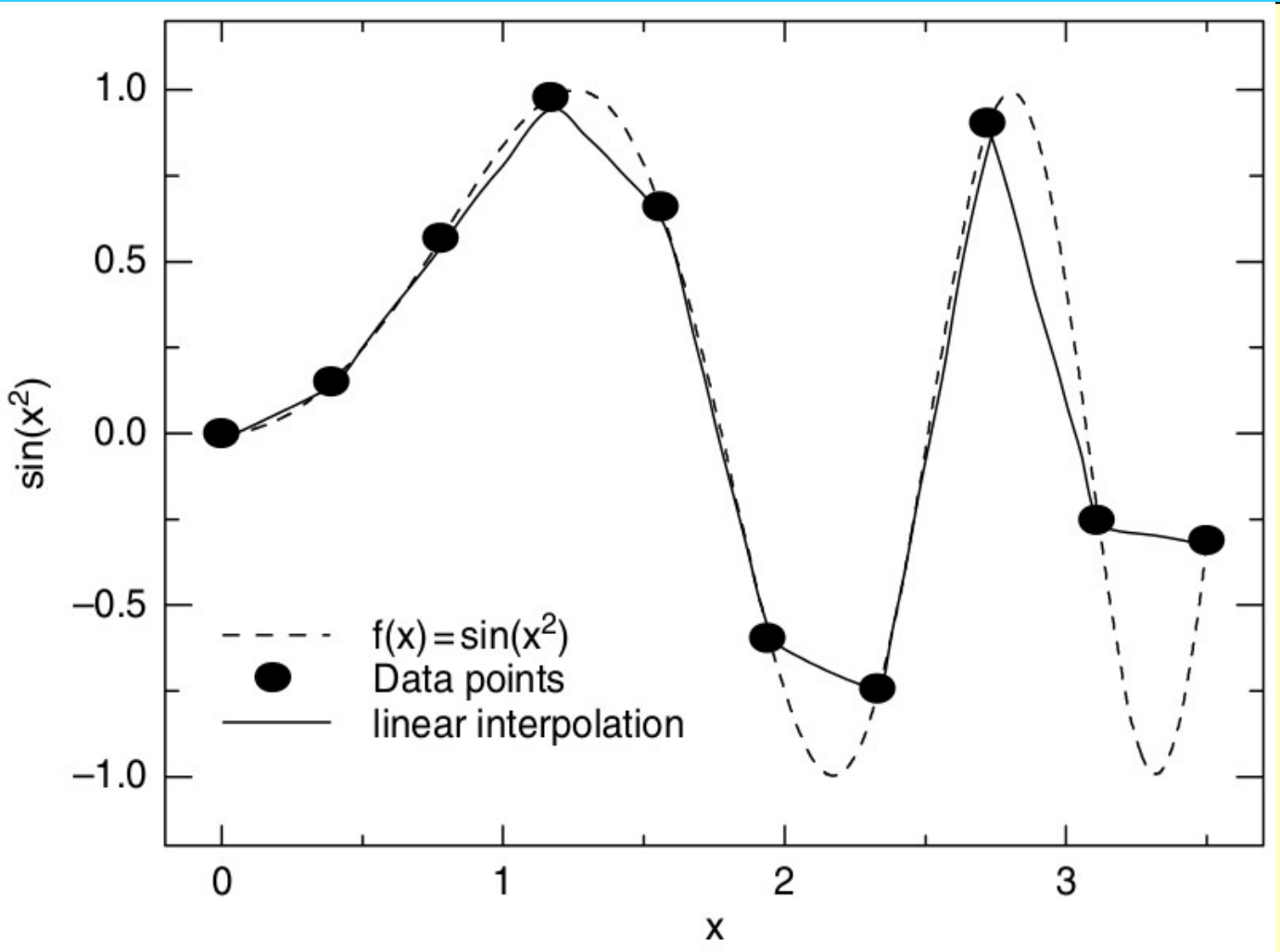
$$g(x) = f_j + \frac{x - x_j}{x_{j+1} - x_j}(f_{j+1} - f_j)$$

en el intervalo $[x_j, x_{j+1}]$.

Para mejor precision hay que tomar los puntos x_j y x_{j+1} mas cercanos a x. La interpolacion lineal tambien puede escribirse

$$g(x) = f_j \frac{x - x_{j+1}}{x_j - x_{j+1}} + f_{j+1} \frac{x - x_j}{x_{j+1} - x_j}$$

Interpolación Lineal (Ej: $f(x)=\sin(x^2)$)



IL: bien para funciones "suaves" donde las derivadas 2da y mayores son pequenhas.

Para c/intervalo diferentes coeficientes a_0 y a_1. Diferencia con ajuste de datos (fitting), que usa los mismo coeficientes en todo el intervalo $[x_1,x_n]$

Interpolación Polinomial

Metodo popular por su simplicidad:

$$g(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

La condicion que el polinomio g(x) pase por los puntos de la muestra $f_j(x_j)$

$$f_j(x_j) = g(x_j) = a_0 + a_1 x_j + a_2 x_j^2 + \ldots + a_n x_j^n$$

genera sistema de n+1 ecs lineales para determinar coeficientes aj. El numero de puntos de datos, menos uno, usados en la interpolacion define el orden de la interpolacion. Interpolacion lineal (o de dos puntos) es la interpolacion de 1er orden.

Del cap. de matrices se puede resolver un sist de ecs lineales, pero tambien hay otra forma, por interpolacion polinomial.

Consideremos interpolacion de tres puntos (2do orden) p/3ptos (x_j, f_j) con j, j+1, j+2:

$$f_j(x_j) = a_0 + a_1 x_j + a_2 x_j^2$$

$$f_{j+1}(x_{j+1}) = a_0 + a_1 x_{j+1} + a_2 x_{j+1}^2$$

$$f_{j+2}(x_{j+2}) = a_0 + a_1 x_{j+2} + a_2 x_{j+2}^2$$

Interpolación Polinomial (cont.)

Se resuelve el sistema para coeficientes a_0, a_1, a_2, dando:

$$g(x) = f_j \frac{(x-x_{j+1})(x-x_{j+2})}{(x_j-x_{j+1})(x_j-x_{j+2})} + f_{j+1} \frac{(x-x_j)(x-x_{j+2})}{(x_{j+1}-x_j)(x_{j+1}-x_{j+2})} + f_{j+2} \frac{(x-x_j)(x-x_{j+1})}{(x_{j+2}-x_j)(x_{j+2}-x_{j+1})}$$

Comparando con la interpolacion lineal, podemos escribir la interpolacion polinomial de grado n, a traves de n+1 puntos

$$g(x) = f_1 \frac{(x-x_2)(x-x_3)\ldots(x-x_{n+1})}{(x_1-x_2)(x_1-x_3)\ldots(x_1-x_{n+1})}$$

$$+ f_2 \frac{(x-x_1)(x-x_3)\ldots(x-x_{n+1})}{(x_2-x_1)(x_2-x_3)\ldots(x_2-x_{n+1})}$$

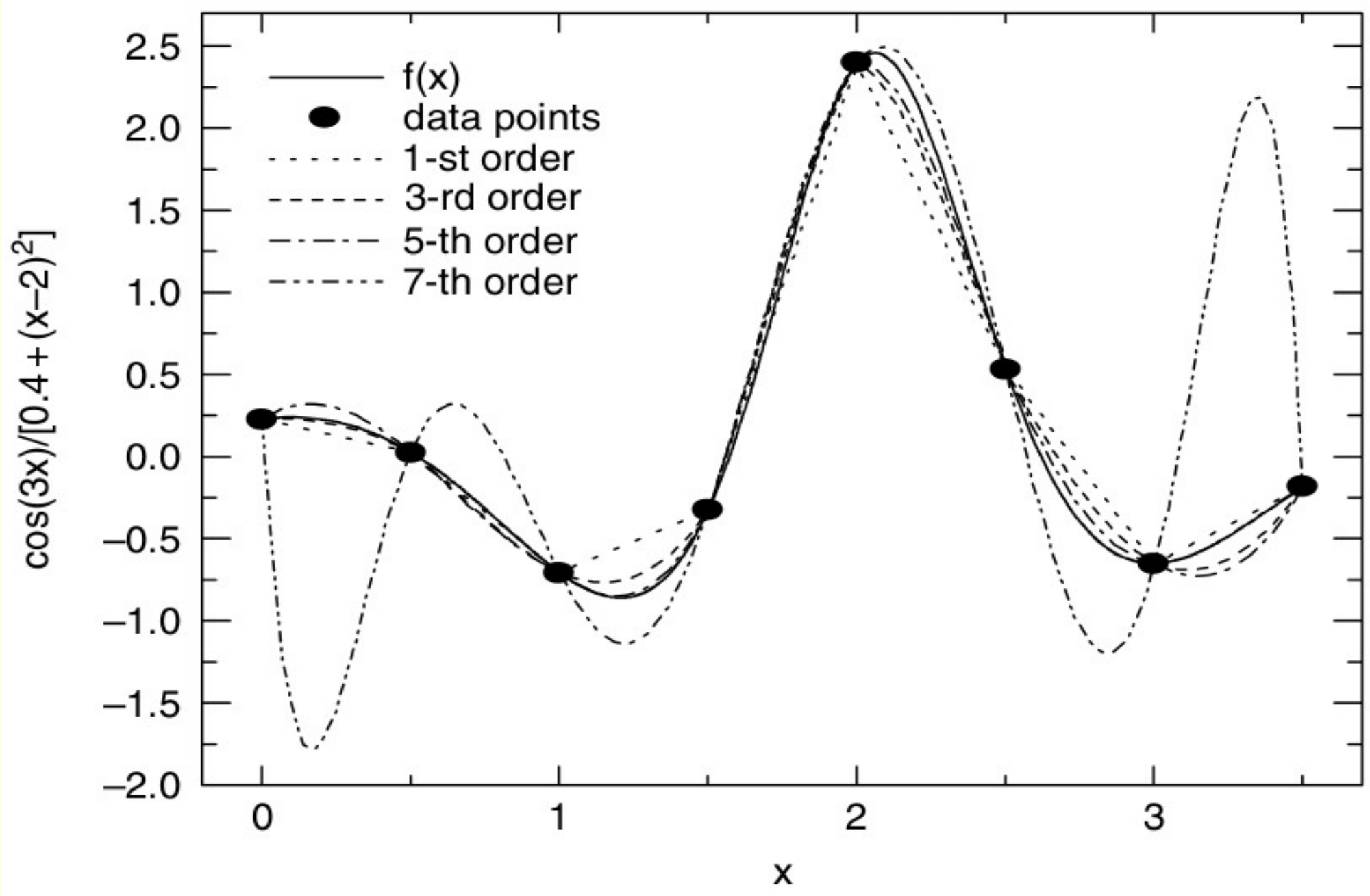
$$+ \ldots + f_{n+1} \frac{(x-x_1)(x-x_2)\ldots(x-x_n)}{(x_{n+1}-x_1)(x_{n+1}-x_2)\ldots(x_{n+1}-x_n)}$$

que es la formula clasica de Lagrange para interpolacion polinomial.

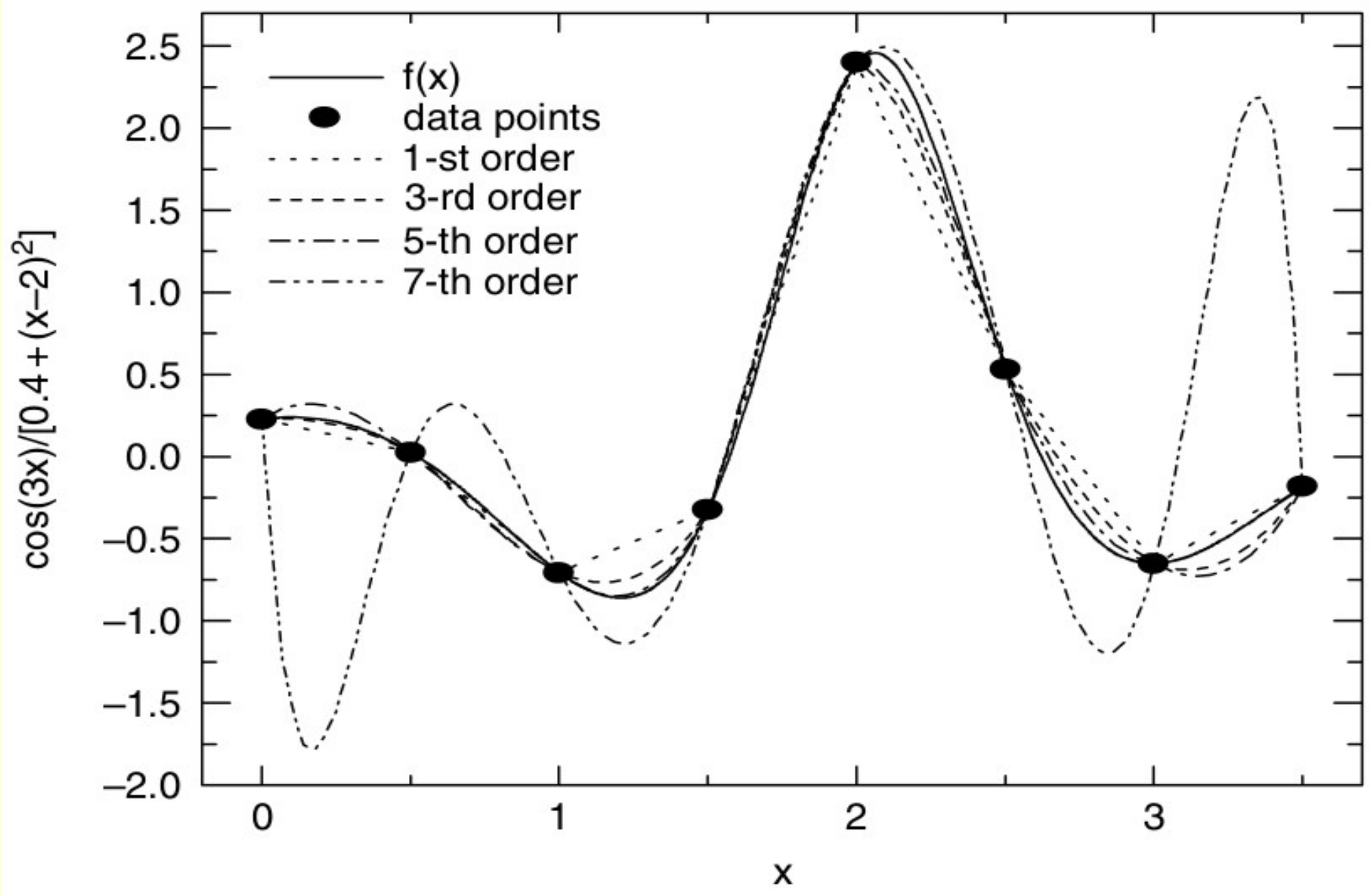
En la figura se ven aplicaciones de interpolacion polinomial de 1ro, 3er, 5to y 7mo orden. En el 7mo orden se ven oscilaciones.

Regla: no usar interpolaciones de orden grande. Limite practico 5to orden (caso contrario usar otro metodo)

Interpolación Polinomial
(Ej: $f(x)=\cos(3x)/[0.4 + (x-2)^2]$)



Interpolación Polinomial
(Ej: $f(x)=\cos(3x)/[0.4 + (x-2)^2]$)



Interpolación de Spline

Problema en interpolacion polinomial: discontinuidad de las derivadas en los puntos x_j.

Mejora: **interpolacion de spline**

Usa informacion de todos los puntos de datos (informacion no local) para garantizar total "smoothness" en la funcion interpolada a cierto orden de derivadas. Es como asegurar una tira de material elástico (metal o regla de plástico) entre nudos (o clavos). La forma final es bastante suave.

Splines cubicos son el metodo mas popular. La funcion interpolada en intervalo $[x_j, x_{j+1}]$:

$$g(x) = f_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

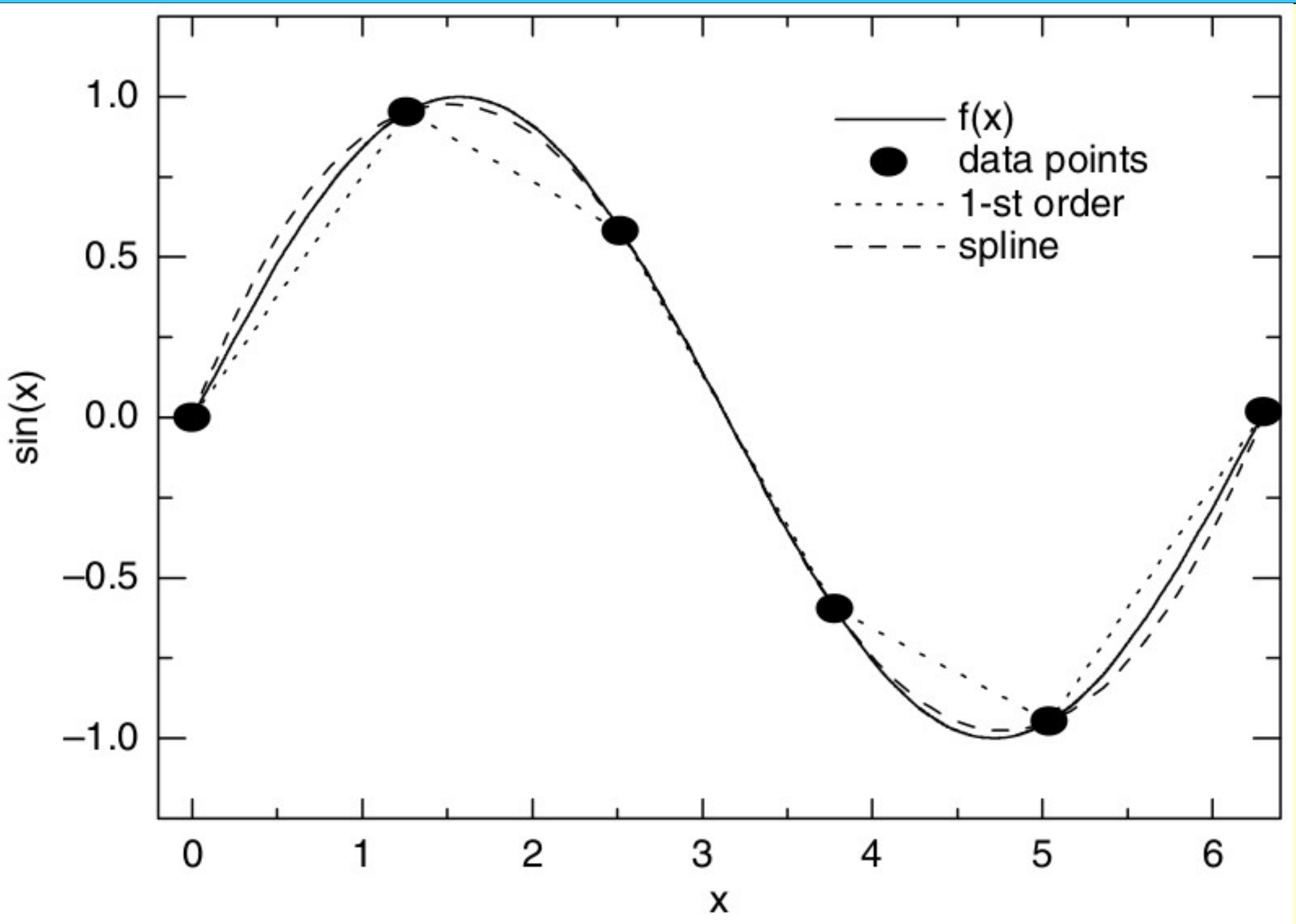
Para c/intervalo se necesita un conjunto de 3 parametros: b_j, c_j, d_j. Como hay n-1 intervalos, se tiene 3n-3 ecs para derivar los coeficientes para j = 1,...,n-1. El hecho que $g_j(x_j) = f_j(x_j)$ impone (n-1) ecs. La idea central para interpolación spline es la idea de que la función interpolada g(x) tiene 1ra y 2da derivadas continuas en c/u de los n - 2 puntos interiores x_j:

$$g'_{j-1}(x_j) = g'_j(x_j)$$

$$g''_{j-1}(x_j) = g''_j(x_j)$$

Estas condicines imponen 2(n-2) ecs resultando en n-1+2(n-2) = 3n-5 ecs para los coeficientes. Otras 2 condiciones: exigir f'' = 0 en los extremos.

Spline cubico (Ej: f(x)=sin(x))



```c
/* SplineAppl.c  Ajuste de Spline Cubico a datos
arreglo de entrada (input) x[n], y[n] =  funcion de tabulacion y(x), x0 < x1,
output yout para dados xout, con yp1 and ypn: 1st derivadas en los extremos,
evaluadas internamente, y2[n]: arreglo de 2da derivadas
(colocando yp1 o  ypn >0.99e30 produce un spline natural) */

#include<stdio.h>
#include<math.h>
#define n 9
#define np 15
#define Nfit 3000 // entra el numero deseado de puntos a ajustar

main()   {

        FILE *output1, *output2;
        output1 = fopen("Spline.dat","w");      // salvar datos en Spline.dat
        output2 = fopen("Input.dat","w");
  // datos de entrada, colocar tus propios datos aqui!
        double x[] = {0., 0.12, 0.25, 0.37, 0.5, 0.62, 0.75, 0.87, 0.99};
        double y[] = {108.6, 16.,845.,883.5,52.8, 19.9, 10.8, 88.25, 84.7};
        double xout, yout;
        double y2[9];
        int i, klo,khi,k ;
        double h,b,a,p,qn,sig,un,yp1,ypn, u[n];
                                                              // entrada
        for (i = 0;i<n;i++) fprintf (output2,"%f\t%f\n",x[i],y[i]);
                                                  // 1st derivada en punto inicial
        yp1 =  (y[1]-y[0])/(x[1]-x[0])
                - (y[2]-y[1])/(x[2]-x[1]) + (y[2]-y[0])/(x[2]-x[0]);
                                                  // 1st derivada en punto final
        ypn = (y[n-1]-y[n-2])/(x[n-1]-x[n-2]) - (y[n-2]-y[n-3])/
                        (x[n-2]-x[n-3]) +(y[n-1]-y[n-3])/(x[n-1]-x[n-3]);
       // prueba de spline natural
        if (yp1 > 0.99e30) y2[0]= u[0]= 0.0;
        else {
                y2[0] =  (-0.5);
                u[0] = (3.0/(x[1]-x[0]))*((y[1]-y[0])/(x[1]-x[0])-yp1);
        }
```

```
                                        // bucle de descomposicion;
        for (i = 1; i<= n-2; i++) {
                sig = (x[i]-x[i-1])/(x[i+1]-x[i-1]);
                p = sig*y2[i-1]+2.0;
                y2[i] = (sig-1.0)/p;
                u[i] = (y[i+1]-y[i])/(x[i+1]-x[i])-(y[i]-y[i-1])/(x[i]-x[i-1]);
                u[i] = (6.0*u[i]/(x[i+1]-x[i-1])-sig*u[i-1])/p;
        }
                                            // prueba para natural
        if (ypn > 0.99e30) qn = un = 0.0;
        else   {                        // tambien evaluar segunda derivada
                qn = 0.5;
                un = ((3.0)/(x[n-1]-x[n-2])) *(ypn-(y[n-1]-y[n-2])/(x[n-1]-x[n-2]));
        }
        y2[n-1] = (un-qn*u[n-2])/(qn*y2[n-2]+1.0);
        for (k= n-2;k>= 0;k--) { y2[k] = y2[k]*y2[k+1]+u[k]; }
                 // Termina initializacion de apline, Comienza ajuste (fit) *spline*
        for ( i = 1; i<=  Nfit; i++ ) {         // loop sobre valores xout
                xout = x[0] + (x[n-1]-x[0])*(i-1)/(Nfit);
                klo = 0;                // algoritmo de biseccion para hallar lugar en tabla
                khi = n-1;
                while (khi-klo > 1) {
                        k = (khi+klo) >> 1;
                if (x[k] > xout) khi = k;
                else klo= k;
                }
                h = x[khi]-x[klo];
                if (x[k] > xout) khi= k;
                else klo= k;
                h = x[khi]-x[klo];
                a = (x[khi]-xout)/h;
                b = (xout-x[klo])/h;
                yout = (a*y[klo]+b*y[khi]+((a*a*a-a)*y2[klo]
                   +(b*b*b-b)*y2[khi])*(h*h)/6.0);
                fprintf (output1,"%f\t%f\n",xout,yout);    // formato gnuplot 3D
        }
        fclose(output1);
        fclose(output2);
        printf("datos almacenados en Spline.dat");
}
```

Interpolación de Spline

Usar:

gnuplot> plot 'Spline.dat' w l , 'Input.dat' w p

Problema: ajuste (fit) de un espectro experimental

Sección transversal para la dispersión de resonancia de un neutrón de un núcleo en la Tabla. 1ra fila es la energía, 2da fila la sección experimental, y 3ra fila, la sección transversal como se predijo teóricamente.

El problema consiste en determinar los valores de la sección transversal en valores de energía entre los que se miden por el experimento.

La predicción teórica es sólo para fines pedagógicos.

$E(MeV)$	0	25	50	75	100	125	150	175	200
σ_{exp}(mb)	10.6	16.0	45.0	83.5	52.8	19.9	10.8	8.25	4.7
σ_{th}(mb)	9.34	17.9	41.5	85.5	51.5	21.5	10.8	6.29	4.09

Se puede resolver el problema usando tecnicas de interpolacion, pero eso ignora la posibilidad de ruido (background) experimental en los datos.

Otra forma de ver el problema es comenzar con lo que creemos es la "correcta" descripcion teorica de los datos, y luego ajustar los parametros presentes en la teoria para obtener el mejor "fit" de los datos.

Es el mejor fit "estadistico" pero no necesariamente pasara por todos los puntos de los datos.

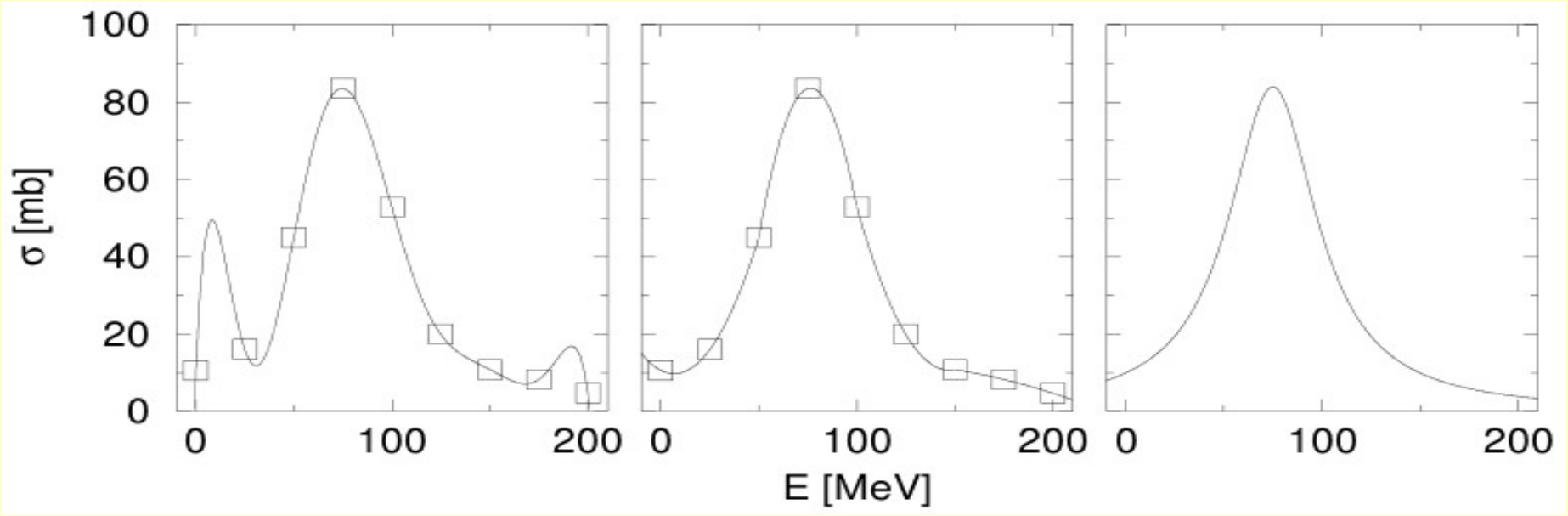
Problema: ajuste (fit) de un espectro experimental

Sabemos (o intuimos) q la seccion transversal (cross section) resonante, como funcion de energia debe ser descrita por una funcion Breit-Wigner

$$\sigma = \frac{\sigma_0}{(E - E_r)^2 + \gamma^2/4}$$

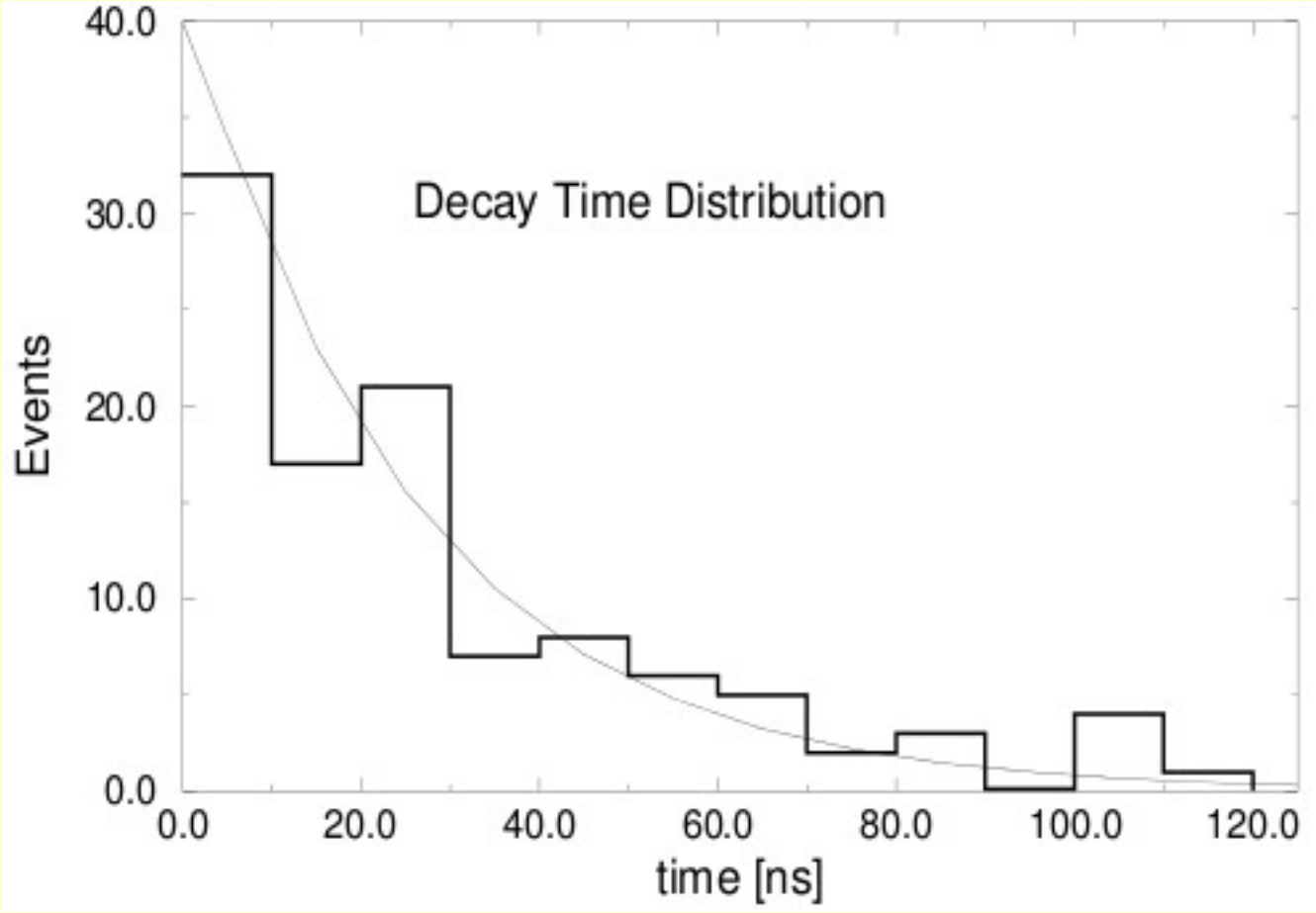
Ajuste de datos (data fitting) puede ser global o local. En ajuste global una sola f(x) es usada para representar el conjunto entero de datos.

Usar con cuidado porque la funcion puede oscilar sin sentido fisico en especial si c/dato contiene error (incerteza) experimental o la funcion no es correcta para los datos. Probl. en fig: polinomio grado 8, series de polinomios grado 3, y curva teorica de la seccion transversal. Hallar E_r, γ.



Problema: ajuste (fit) de decaimiento exponencial

Figura presenta datos de los numeros de decaimientos ΔN del meson π como funcion del tiempo. Nota: el tiempo ha sido "binned" en intervalos (cajas, "bins") de $\Delta t = 10ns$, y la curva suave da la ley teorica de decaimiento exponencial. El problema es deducir el tiempo de vida media $\tau_{1/2}$ del meson p a partir de esos datos (la medida experimental conocida es 2.6×10^{-8} seg).



Modelo: decaimiento exponencial

Empezamos con un número N^0 de partículas radiactivas en el tiempo t=0 que puede decaer a otras partículas. Si se espera un tiempo Δt corto, entonces un número pequeño ΔN de partículas se desintegran espontáneamente, es decir, sin influencias externas.

Esta descomposición es un proceso estocástico, lo que significa que hay un elemento de azar involucrado y las fluctuaciones son de esperar. La ley básica de la naturaleza de decaimiento espontáneo es que el número ΔN que se desintegra en intervalo de tiempo Δt es proporcional al número de partículas presentes N(t) en ese momento y al intervalo de tiempo.

$$\Delta N(t) = -\frac{1}{\tau} N(t) \Delta t$$

donde τ es el tiempo de vida medio de la partícula. Esta ecuación se puede organizar en una ecuación para la tasa de decaimiento promedio

$$\frac{\Delta N(t)}{\Delta t} = -\lambda N(t)$$

Modelo: decaimiento exponencial

Si el número de desintegraciones ΔN es muy pequeño en comparación con el número de partículas N, y si nos fijamos en los intervalos de tiempo infinitamente pequeño, entonces la ecuación diferencial anterior se convierte en la ecuación diferencial

$$\frac{dN(t)}{dt} \simeq -\lambda N(t)$$

Esta ecuación diferencial tiene una solución exponencial del número,

$$N(t) = N_0 e^{-t/\tau}$$

así como una solución exponencial de la velocidad de desintegración

$$\frac{dN(t)}{dt} = -\frac{N_0}{\tau} e^{-t/\tau} = \frac{dN}{dt}(0) e^{-t/\tau}$$

Esta es la formula teorica que deseamos ajustar (fitar) a los datos.el resultado de ese ajuste es el mejor valor para el τ.

Teoria de probabilidad (distribución binomial)

El campo de Estadística es un intento de utilizar la Matemática para describir eventos, tales como lanzamientos de moneda, en el que hay un elemento de casualidad o azar. Un bloque de construcción básico de las estadísticas es la función de **distribución binomial**,

$$P_B(x) = \binom{N}{x} p^x (1-p)^{N-x} = \frac{N!}{(N-x)!x!} p^x (1-p)^{N-x}$$

donde $P_B(x)$ es la probabilidad de que el evento independiente ("caras") se producirá x veces en los N ensayos. Aquí p es la probabilidad de que ocurra un evento individual, por ejemplo, la probabilidad de "caras" en cualquier lanzamiento es p = 1/2.

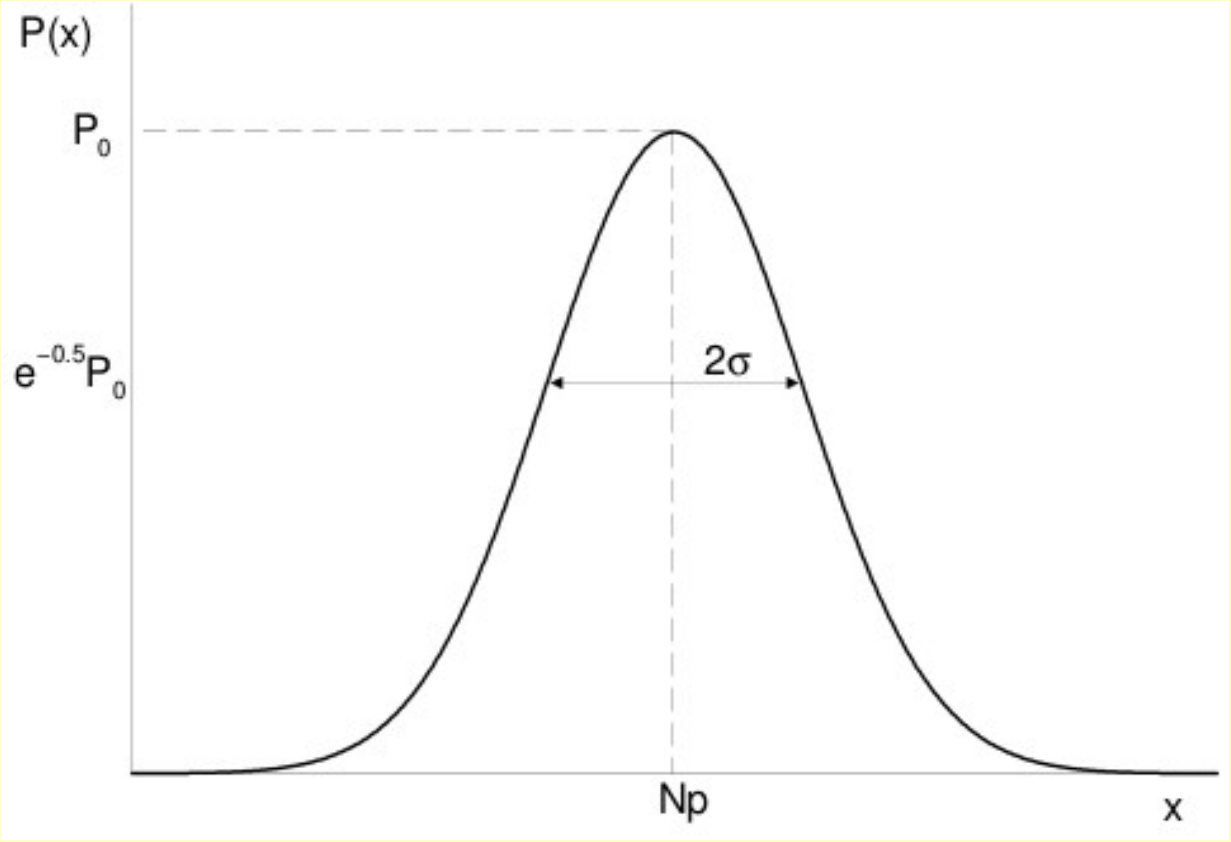
La variable N es el número de ensayos o experimentos en los que ese evento puede ocurrir, por ejemplo, el número de veces que voltear la moneda. Por moneda voltear, la probabilidad de éxito p y la probabilidad de fallo (1-p) son ambos 1/2, pero en general p puede ser cualquier número entre 0 y 1.

Teoria de probabilidad (distribución de Gauss)

Para mayor comodidad de cálculo, los factoriales en la distribución binomial son usualmente eliminados al considerar el límite en el que el número de ensayos de N→∞. En las **estadísticas de Gauss** o **normal**, la probabilidad p de una prueba individual (caras) sigue siendo finita cuando N→∞:

$$P_G(x) = \lim_{N\to\infty, p\neq 0} P_B(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

Esto produce la función de la fig, donde μ≡ <x> es la media y σ la varianza



Teoria de probabilidad (distribución de Gauss)

Esas constantes estan relacionadas con las otras por

$$\mu = Np\,, \quad \sigma = \sqrt{Np(1-p)}$$

La **distribución de Gauss** es buena aproximación a distribución binomial incluso para N tan pequeño como 10. Repitiendo, se describe un experimento en que se realizan N mediciones de la variable x. El promedio de estas mediciones es μ, y el "error" o incertidumbre en μ es σ. Como ejemplo, en N=1000 lanzamientos de moneda, la probabilidad de una cara es p = 1/2 y el promedio de caras μ debe ser Np = N/2 = 500.

Como se muestra en la figura, la distribución de Gauss tiene una anchura

$$\sigma = \sqrt{Np(1-p)} \propto \sqrt{N}$$

de modo que la distribución se hace más y más amplia a medida que más se hagan mas mediciones. Sin embargo, la anchura relativa, cuya inversa da una indicación de la probabilidad de obtener el promedio μ, disminuye con N:

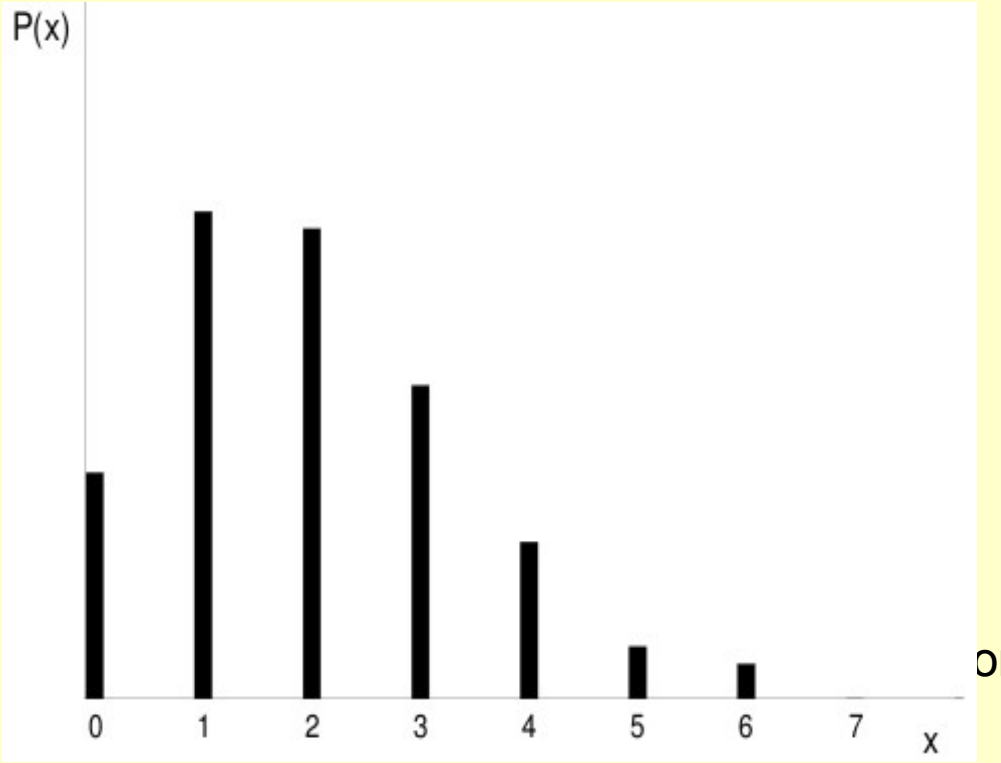
$$\frac{\text{width}}{N} \propto \frac{\sqrt{N}}{N} = \frac{1}{\sqrt{N}} \to 0, \quad (N \to \infty)$$

Teoria de probabilidad (distribución de Poisson)

Otro límite de la distribución binomial es la **distribución de Poisson**. Aqui, el número de ensayos N→∞, sin embargo, la probabilidad de un suceso individual p→0 de tal manera que el producto Np permanece finito:

$$P_P(x) = \lim_{N\to\infty, p\to 0} P_B(x) = \frac{\mu^x e^{-\mu}}{x!}$$

La distribución de Poisson describe experimentos de desintegración radiactiva o intercambios telefonicos donde puede haber un gran número de ensayos (cada μseg, cuando el contador está encendido), pero una baja probabilidad por evento (un decaimiento o llamada telefónica) que ocurren en este μseg.



La distribución de Poisson es bastante asimétrica para μ pequeña, y de esta forma muy diferente de una distribución de Gauss. Para μ>>1, la distribución de Poisson se aproxima a una distribución gaussiana.

Metodo de Minimos Cuadrados

Varias definiciones de “buen ajuste” (good fit):

- Si los datos contienen errores (incertezas), entonces el "mejor" ajuste en el sentido estadístico no necesariamente pasa por los puntos de datos.

- Sólo para el caso más simple de un ajuste de mínimos cuadrados lineal, podemos escribir una solución de forma cerrada para evaluar y obtener el ajuste.

Problemas mas realisticos suelen resolverse por procedimientos de búsqueda de **ensayo y error** (mediante subrutinas de bibliotecas como SLATEC).

Imagínese que se han medido N_D valores de la variable dependiente y (# decs/t) en función de la variable independiente x: $(x_i, y_i \pm \sigma_i)$, $i = 1,..., N_D$.

Nuestro objetivo es determinar qué tan bien una función matemática $y = f(x)$ (también llamada teoría) puede describir estos datos. Alternativamente, si la teoría contiene algunos parámetros o constantes, nuestro objetivo puede ser visto como determinar las mejores valores para estos parámetros. Suponemos que la función modelo f (x) contiene, además de la dependencia funcional de x, una dependencia adicional a M_P parámetros $\{a_1, a_2, ..., a_{MP}\}$. Para la función de decaimiento exponencial, el parámetro es el tiempo de vida τ.

Metodo de Minimos Cuadrados

Señalamos esto como $f(x)=f(x;\{a_1,a_2,...,a_{Mp}\})=f(x;\{a_m\})$

Nótese de que los parámetros $\{a_m\}$ no son variables, en el sentido de los números leídos desde un metro, sino más bien son partes del modelo teórico tales como el tamaño de una caja, la masa de una partícula, o la profundidad del de un pozo de potencial.

Tomamos el (χ^2) como indicador de qué tan bien una función f teórica reproduce los datos

$$\chi^2 \stackrel{\text{def}}{=} \sum_{i=1}^{N_D} \left(\frac{y_i - f(x_i;\{a_m\})}{\sigma_i} \right)^2 ,$$

donde la suma es sobre los N_D puntos experimentales $(x_i, y_i \pm \sigma_i)$. La definición es tal que los valores más pequeños de χ^2 son mejores fits, con $\chi^2 = 0$ que ocurre si la curva teórica pasó por el centro de cada punto de datos. Nótese también que la ponderación $1/\sigma_i^2$ significa que las mediciones con errores mayores contribuyen menos a χ^2.

Ajuste de mínimos cuadrados se refiere a ajustar a la teoría hasta que se encuentra un mínimo en χ^2, es decir, la búsqueda de una curva que produce el valor mínimo para los cuadrados sumados de las desviaciones de los datos de la función f(x). En general, este es el mejor ajuste posible o la mejor manera de determinar los parámetros en una teoría.

Metodo de Minimos Cuadrados

Los parámetros M_P $\{a_1,a_2,...,a_{Mp}\}$ que hacen que un valor extremo χ^2 son hallados resolviendo las M_P ecuaciones:

$$\frac{\partial \chi^2}{\partial a_m} = 0, \quad (m = 1, M_P)$$

$$\Rightarrow \sum_{i=1}^{N_D} \frac{y_i - f(x_i)}{\sigma_i^2} \frac{\partial f(x;\{a_m\})}{\partial a_m} = 0, \quad (m = 1, M_P)$$

Generalmente, la función $f(x;\{a_m\})$ tiene una dependencia suficientemente complicada en los valores a_m para esta ecuacion producir M_P ecuaciones no lineales simultáneas en los a_m valores. En estos casos, las soluciones se hallan por prueba y error a través del espacio de parámetros M_P-dimensional. Para estar seguro, cuando esa búsqueda se ha completado es necesario comprobar que el mínimo χ^2 has encontrado es global y no local. Una forma de hacerlo es repetir la búsqueda de una red completa de valores iniciales, y si se hallan diferentes mínimos, escoger el que tiene el menor χ^2.

Bondad del ajuste

Cuando las desviaciones de la teoría se deben a errores aleatorios y cuando estos son descritas por una distribución de Gauss, hay algunas reglas útiles para recordar. Un ajuste es bueno si el valor de χ^2 calculado con la definición dada antes es aproximadamente igual al número de grados de libertad (g.l.): $\boldsymbol{\chi^2 \approx N_D - M_P}$, donde N_D es el numero de puntos de datos y M_P el numero de parametros de la funcion teorica. Si χ^2 es mucho menor que esto no significa una gran teoria o una medidad precisa; probablemente tienes muchos parametros o has asignado errores (σ_i) que son muy grandes. De hecho, χ^2 muy pequeño puede indicar que se está ajustando la dispersión aleatoria de los datos en lugar de "perder" ~1/3 de las barras de error (como se espera para las estadísticas de Gauss). Si el χ^2 es mucho mayor que los g.l., la teoría no puede ser buena, puede que haya subestimado considerablemente sus errores, o puede tener errores que no son aleatorios.

Si usted cree que obtener un buen ajuste a los datos, pero no puede determinar el χ^2 porque no tienen los valores de los errores experimentales (tal vez asumido $\sigma_i = 1$), puedes conseguir un σ^2 aproximada para usar en el cálculo de χ^2.

Primero ajusta los datos con un valor arbitrario σ_i para. Luego calcula la varianza de tus datos,

$$\sigma^2_{\text{exp}} \overset{\text{def}}{=} \frac{1}{N_D} \sum_{i=1}^{N_D} [y_i - f(x_i)]^2$$

Finalmente usas σ_{exp} como un aproximado de σ_i y aplicas definicion de χ^2 para obtener un valor con significado fisico.

Implementación: ajuste de mínimos cuadrados

Las M_P ecuaciones simultáneas simplifican considerablemente si las funciones f(x;{am}) dependen linealmente sobre los valores a. Esto sucede, por ejemplo, cuando la función teorica f(x) es lineal:

$$f(x;\{a_1,a_2\}) = a_1 + a_2 x$$

En este caso (también conocido como regresión lineal) hay $M_P = 2$ parámetros, el a_2 pendiente y la intersección a_1. Notar que mientras que hay sólo dos parámetros para determinar, puede haber un número arbitrario N_D de puntos de datos a ajustar. Recuerde, una solución única no es posible a menos que el número de puntos de datos es igual o mayor que el número de parámetros.

Para el caso lineal, las ecuaciones de minimización de χ^2 son dos, y determina los parámetros en términos de todos los puntos de datos

$$a_1 = \frac{S_{xx}S_y - S_x S_{xy}}{\Delta}, \quad a_2 = \frac{S S_{xy} - S_x S_y}{\Delta}$$

$$S = \sum_{i=1}^{N_D} \frac{1}{\sigma_i^2}, \quad S_x = \sum_{i=1}^{N_D} \frac{x_i}{\sigma_i^2},$$

$$S_y = \sum_{i=1}^{N_D} \frac{y_i}{\sigma_i^2}, \quad S_{xx} = \sum_{i=1}^{N_D} \frac{x_i^2}{\sigma_i^2},$$

$$S_{xy} = \sum_{i=1}^{N_D} \frac{x_i y_i}{\sigma_i^2}, \quad \Delta = S S_{xx} - S_x^2.$$

Implementación: ajuste de mínimos cuadrados

Si conoce los errores σ_i en las mediciones experimentales de y_i, o ha determinado un σ aproximado, de la varianza de la muestra de su función de ajuste, la teoría le da una expresión para la varianza o incertidumbre en los parámetros que se deducen:

$$\sigma_{a_1}^2 = \frac{S_{xx}}{\Delta}, \quad \sigma_{a_2}^2 = \frac{S}{\Delta}$$

Es una medida de las incertidumbres en los valores de los parámetros ajustados, derivados de la incertidumbres y_i en los valores medidos y_i.

Una medida de la dependencia de los parámetros en cada otro viene dada por el coeficiente de correlación:

$$\rho(a_1, a_2) = \frac{\mathrm{cov}(a_1, a_2)}{\sigma_{a_1}\sigma_{a_2}}$$

$$\mathrm{cov}(a_1, a_2) = \frac{-S_x}{\Delta}.$$

Aquí $\mathrm{cov}(a_1,a_2)$ es la covarianza de a_1 y a_2, y desaparece si a_1 y a_2 son independientes. El coeficiente de correlación $\rho(a_1,a_2)$ se encuentra en el rango $-1 \leq \rho \leq 1$. Un ρ positivo indica que los errores en a_1 y a_2 es probable que tengan el mismo signo, ρ negativo indica signos opuestos.

Implementación: ajuste de mínimos cuadrados

Las soluciones anteriores analíticas para los parámetros son de la forma que se encuentra en los libros de estadísticas, pero no son óptimas para los cálculos numéricos porque cancelación sustractiva puede hacer que las respuesta sea inestable. Como se discutió en el Capítulo de errores e incertidumbres en los cálculos, una reordenación de las ecuaciones puede disminuir este tipo de error. Por ejemplo, mejores expresiones que miden los datos relativos a sus promedios:

$$
\begin{aligned}
a_1 &= y_{\rm av} - a_2 x_{\rm av}, \quad a_2 = \frac{S_{xy}}{S_{xx}}, \\
S_{xy} &= \sum_{i=1}^{N_d} \frac{(x_i - x_{\rm av})(y_i - y_{\rm av})}{\sigma_i^2}, \quad S_{xx} = \sum_{i=1}^{N_d} \frac{(x_i - x_{\rm av})^2}{\sigma_i^2} \\
x_{\rm av} &= \frac{1}{N} \sum_{i=1}^{N_d} x_i, \quad y_{\rm av} = \frac{1}{N} \sum_{i=1}^{N_d} y_i.
\end{aligned}
$$

```c
//   fit.c:    Ajuste de minimos cuadrados
#include <stdio.h>
#include <math.h>
#define datos 12        // numero de puntos de datos

main() {

        int i, j;
        double s, sx, sy, sxx, sxy, delta, inter, pendiente;
        double x[datos], y[datos], d[datos];
                                                    // datos de entrada
        for (i = 0; i<datos; i++) x[i] = i*10+5;
                y[0] = 382; y[1] = 187;  y[2] = 281; y[3] = 87;
                y[4] = 8;   y[5] = 6;    y[6] = 5;   y[7] = 28;
                y[8] = 2;   y[9] = 80.1; y[10] = 4; y[11] = 18;
                                                   // iniciar sumas a 0
        for (i = 0; i<datos; i++) d[i] = 1.;
                s = sx = sy = sxx = sxy = 0;
                                                   // calculando sumas
        for (i = 0;i<datos;i++) {
                s   +=         1 / (d[i]*d[i]);
                sx  +=      x[i] / (d[i]*d[i]);
                sy  +=      y[i] / (d[i]*d[i]);
                sxx += x[i]*x[i] / (d[i]*d[i]);
                sxy += x[i]*y[i] / (d[i]*d[i]);
        }
        delta = s*sxx-sx*sx;
        pendiente = (s*sxy-sx*sy) / delta;                  // calculando todo
        inter = (sxx*sy-sx*sxy) / delta;                    // coeficientes
        printf("intercepto = %f\t +/- %f\n", inter, sqrt(sxx/delta) );
        printf("pendiente = %f\t +/- %f\n", pendiente, sqrt(s/delta) );
        printf("correlacion = %f\n", -sx/sqrt(sxx*s));
}
```