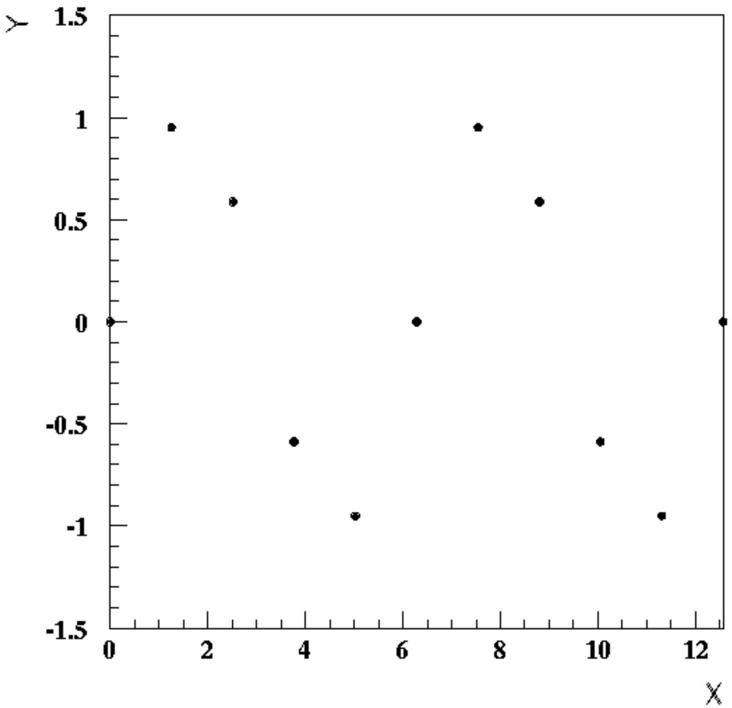
Interpolation, Smoothing, Extrapolation

A typical numerical application is to find a smooth parametrization of available data so that results at intermediate (or extended) positions can be evaluated.



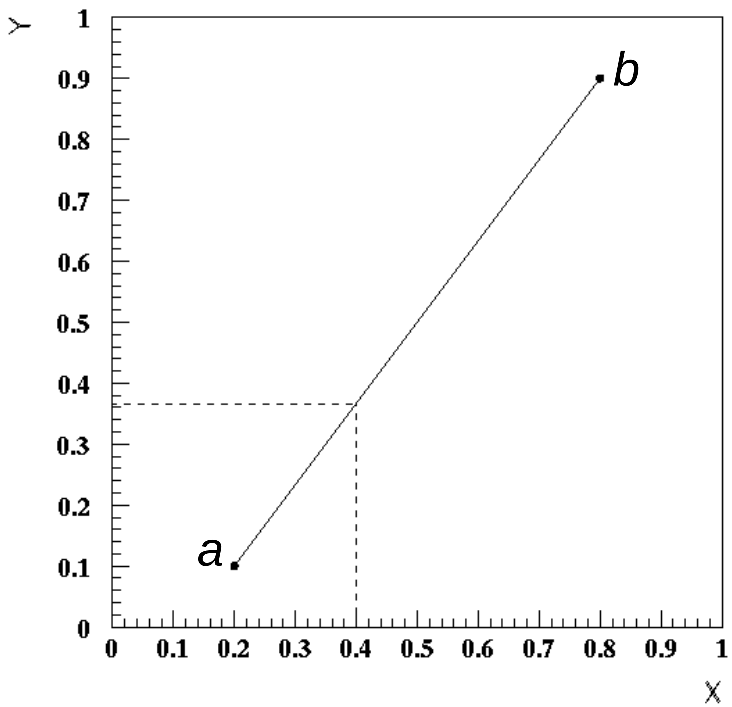
What is a good estimate for y for x=4.5, or x=15 ?

Options: if have a model, y=f(x), then fit the data and extract model parameters. Model then used to give values at other points.
If no model available, then use a smooth function to interpolate

Interpolation

Start with interpolation. Simplest - linear interpolation. Imagine we have two values of x, x_a and x_b, and values of y at these points, y_a, y_b. Then we interpolate (estimate the value of y at an intermediate point) as follows:

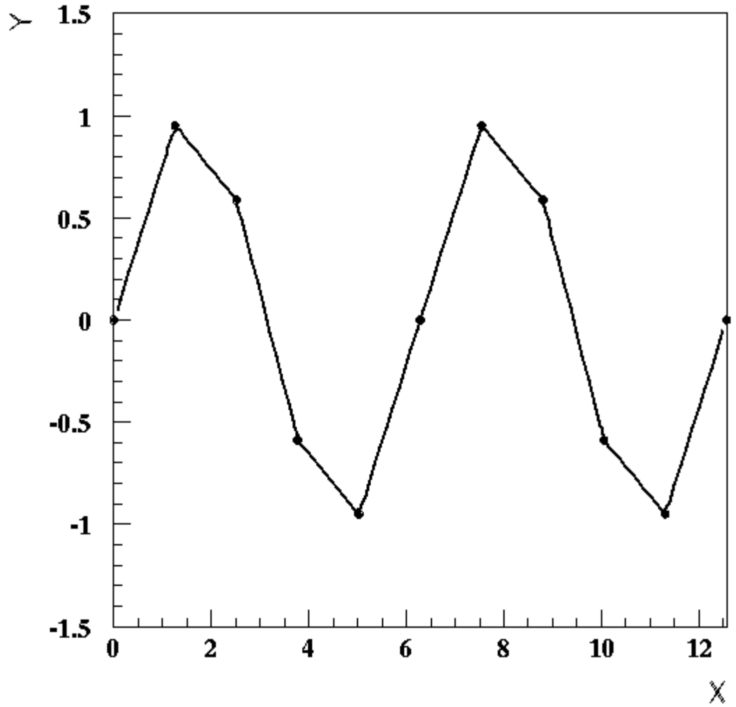
$$y = y_a + \frac{(y_b - y_a)}{(x_b - x_a)}(x - x_a)$$



Interpolation

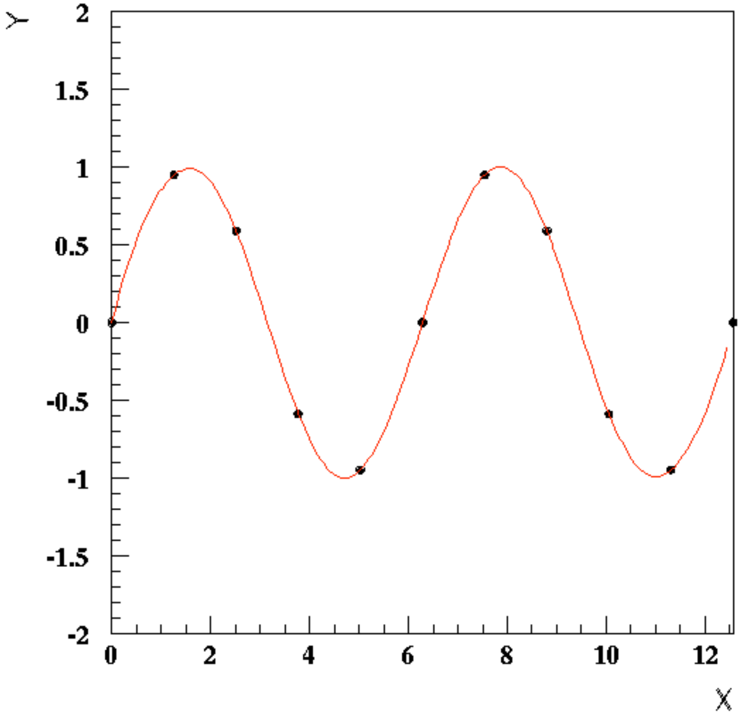
Back to the initial plot:

$$y = y_a + \frac{(y_b - y_a)}{(x_b - x_a)}(x - x_a)$$

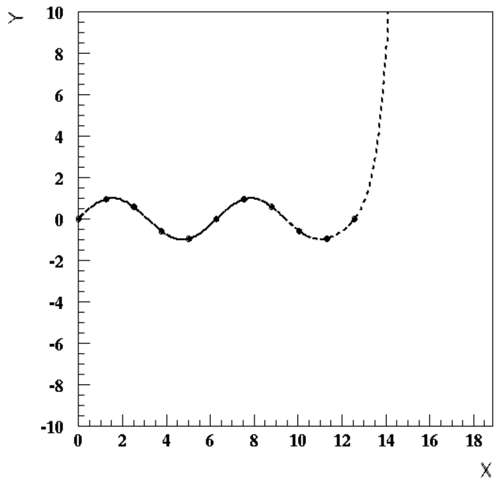


Not very satisfying. Our intuition is that functions should be smooth. Try reproducing with a higher order polynomial. If we have *n+1* points, then we can represent the data with a polynomial of order *n*.

Interpolation



Fit with a 10th order polynomial. We go through every data point (11 free parameters, 11 data points). This gives a smooth representation of the data and indicates that we are dealing with an oscillating function. However, extrapolation is dangerous !



Lagrange Polynomials

For $n+1$ points (x_i,y_i), with

$$i = 0,1,\cdots,n \qquad x_i \neq x_{j\neq i}$$

there is a unique interpolating polynomial of degree n with

$$p(x_i) = y_i \qquad i = 0,1,\cdots,n$$

Can construct this polynomial using the Lagrange polynomials, defined as:

$$L_i(x) = \frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}$$

Degree n (denominator is constant), and

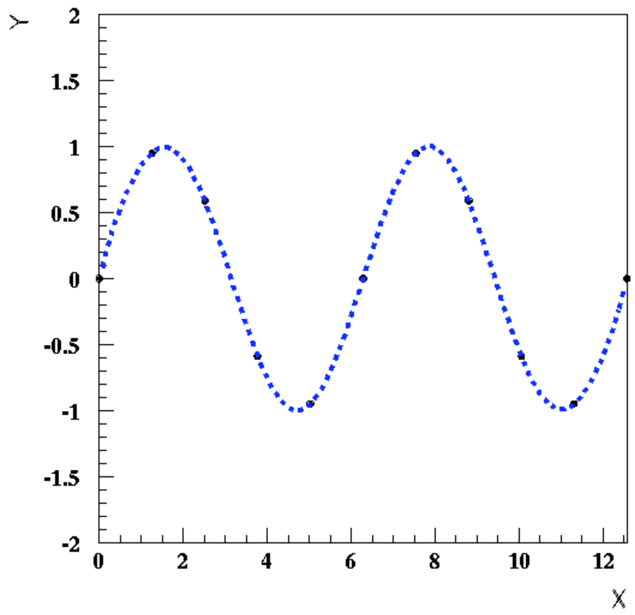
$$L_i(x_k) = \delta_{i,k}$$

Lagrange Polynomials

The Lagrange Polynomials can be used to form the interpolating polynomial:

$$p(x)=\sum_{i=0}^{n} y_i L_i(x)=\sum_{i=0}^{n} y_i \prod_{k=0,k\neq i}^{n} \frac{x-x_k}{x_i-x_k}$$

```
*
* Example: 10th order polynomial
    Lagrange=0.
*
    Do I=0,10
      term=1.
      Do k=0,10
        If (k.ne.I) then
          term=term*(x-x0(k))/(x0(I)-x0(k))
        Endif
      Enddo
      Lagrange=Lagrange+Y0(I)*term
    Enddo
```



Lagrange Polynomials

Error estimation of the interpolation/extrapolation:

Define $err(x) = f(x) - p(x)$ where $f(x)$ is original function,

$p(x)$ is interpolating function

Choose $\bar{x} \neq x_i$ for any $i = 0, 1, \cdots, n$

Now define

$$F(x) = f(x) - p(x) - \left(f(\bar{x}) - p(\bar{x})\right) \frac{\prod_{i=0}^{n} (x - x_i)}{\prod_{i=0}^{n} (\bar{x} - x_i)}$$

Now look at properties of *F(x)*

Inter- and Extrapolation Error

$F(x_i) = 0$ for all $i = 0, 1, \cdots, n$ and $F(\bar{x}) = 0$.

I.e., $F(x)$ has $n+2$ zeroes

Rolle's theorem: There exists a ξ between $\bar{x}, x_0, x_1, \cdots, x_n$ such that

$F^{(n+1)}(\xi) = 0$

so,

$$0 = f^{(n+1)}(\xi) - \left(f(\bar{x}) - p(\bar{x})\right)\frac{(n+1)!}{\prod_{i=0}^{n}(\bar{x} - x_i)} \qquad \left(p^{(n+1)} = 0\right)$$

but $\bar{x}$ is arbitrary, so

$$e(\bar{x}) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(\bar{x} - x_i) \quad \text{for some } \xi \text{ between } \bar{x}, x_0, x_1, \cdots, x_n$$

Inter- and Extrapolation Error

Suppose we are trying to interpolate a sine function (our example). We have 11 data points (n=10). Then

$$f^{(n+1)}(x) = \frac{d^{11}\sin x}{dx^{11}} = -\cos x \qquad \text{so } f^{(n+1)}(\xi) \leq 1$$

$$e(\overline{x}) \leq \frac{\prod_{i=0}^{n}(\overline{x}-x_i)}{(n+1)!} \quad < \frac{(b-a)^{11}}{11!} \text{ for interpolation}$$

For extrapolation, the error grows as the power (n+1)

Splines

Assume have data $\{x_k, y_k\}$ with $k = 0, n$ i.e., $n+1$ points (also known as knots)

Define $a = x_0$, $b = x_n$ and arrange so that $x_0 < x_1 < \cdots < x_{n-1} < x_n$

A spline is polynomial interpolation between the data points which satisfies the following conditions:

1. $S(x) = S_k(x)$ for $x_k \le x \le x_{k+1}$ $k = 0, 1, \cdots, n-1$

2. $S(x_k) = y_k$ $k = 0, 1, \cdots, n$

3. $S_k(x_{k+1}) = S_{k+1}(x_{k+1})$ $k = 0, 1, \cdots, n-2$ i.e., $S(x)$ is continuous

Splines

Linear Spline:

$$S_k(x) = y_k + \frac{y_{k+1} - y_k}{x_{k+1} - x_k}(x - x_k)$$

Quadratic Spline:

$$S_k(x) = y_k + z_k(x - x_k) + \frac{z_{k+1} - z_k}{2(x_{k+1} - x_k)}(x - x_k)^2$$

z_0 has to be fixed, for example from requiring $S_k'(a) = z_0 = 0.$

then,

$$z_{k+1} = -z_k + 2\frac{y_{k+1} - y_k}{x_{k+1} - x_k}$$

Splines

The cubic spline satisfies the following conditions:

1. $S(x) = S_k(x) \quad \text{for } x_k \le x \le x_{k+1} \quad k = 0,1,\cdots,n-1$

   $S_k(x) = S_{k,0} + S_{k,1}(x - x_k) + S_{k,2}(x - x_k)^2 + S_{k,3}(x - x_k)^3$

2. $S(x_k) = y_k \quad k = 0,1,\cdots,n$

3. $S_k(x_{k+1}) = S_{k+1}(x_{k+1}) \quad k = 0,1,\cdots,n-2 \quad \text{i.e., } S(x) \text{ is continuous}$

4. $S'_k(x_{k+1}) = S'_{k+1}(x_{k+1}) \quad k = 0,1,\cdots,n-2 \quad \text{i.e., } S'(x) \text{ is continuous}$

5. $S''_k(x_{k+1}) = S''_{k+1}(x_{k+1}) \quad k = 0,1,\cdots,n-2 \quad \text{i.e., } S''(x) \text{ is continuous}$

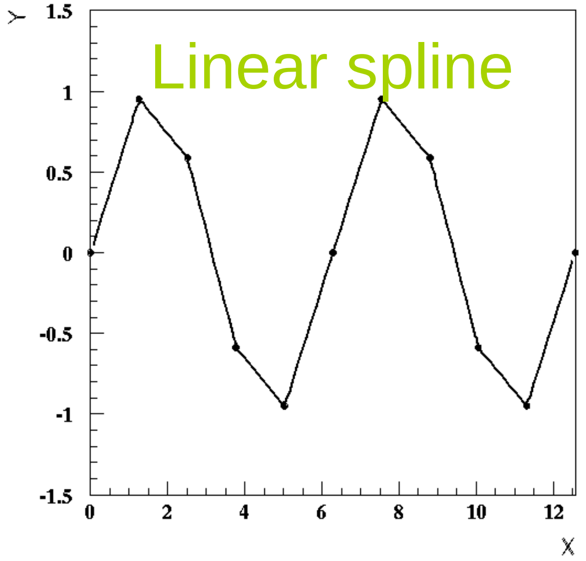
Splines

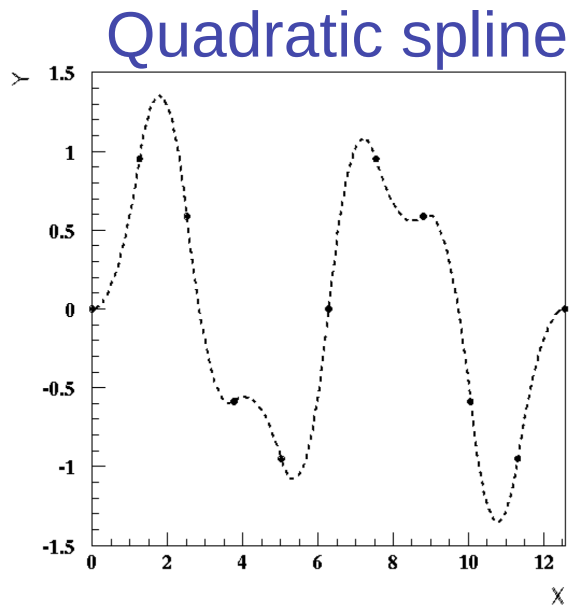
Need at least 3rd order polynomial to satisfy the conditions. Number of parameters is 4n. Fixing $S_k(x_k)=y_k$ gives n+1 conditions. Fixing $S_k(x_{k+1})=S_{k+1}(x_{k+1})$ gives an additional n-1 conditions. Matching the first and second derivative gives another 2n-2 conditions, for a total of 4n-2 conditions. Two more conditions are needed to specify a unique cubic spline which satisfies the conditions on the previous page:

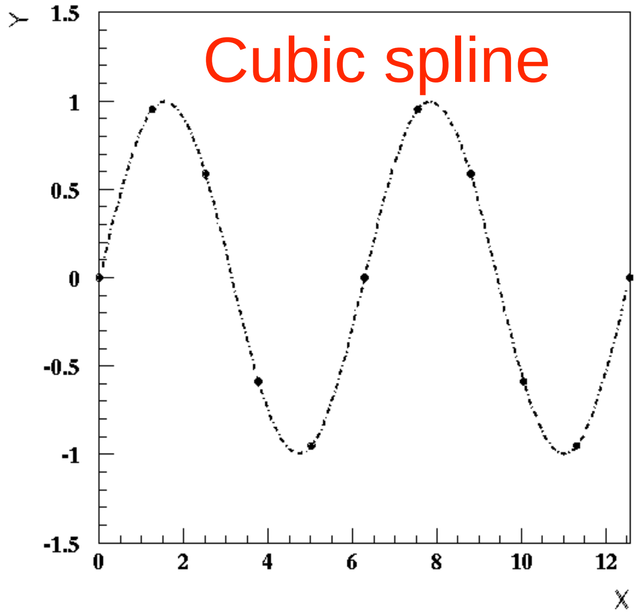
$$S''(a)=0 \quad S''(b)=0$$ Natural cubic spline

Can take other options for the boundary conditions

Splines


Linear spline
Y
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12
X


Quadratic spline
Y
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12
X


Cubic spline
Y
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12
X

Cubic Splines

The cubic spline is optimal in the following sense:

1. It is accurate to fourth order, and

$$\left|\mathrm{f(x)\text{-}S(x)}\right| \le \frac{5}{384} \max_{a \le x \le b} \left| f^{(4)}(x) \right| \cdot h^4 \qquad \text{where } h = \max_k \left| x_{k+1} - x_k \right|$$

2. It is the minimum curvature function linking the set of data points.

$$\text{Curvature is defined as } \left| \frac{f''(x)}{\left(1 + f'(x)^2\right)^{3/2}} \right| \approx \left| f''(x) \right|$$

$$\text{Cubic spline satisfies } \int_a^b \left[S''(x)\right]^2 dx \le \int_a^b \left[f''(x)\right]^2 dx$$

Any smooth interpolating function must have curvature at least as large as a cubic spline

Cubic Splines

Proof of 2.

Start with algebraic identity $F^2 - S^2 = (F-S)^2 - 2S(S-F)$

Let $F = f''(x), \quad S = S''(x)$

then

$$\int_a^b [f''(x)]^2 dx - \int_a^b [S''(x)]^2 dx = \int_a^b [f''(x) - S''(x)]^2 dx - 2\int_a^b S''(x)[S''(x) - f''(x)]dx$$

$$\int_a^b [f''(x) - S''(x)]^2 dx \geq 0$$

$$\int_a^b S''(x)[S''(x) - f''(x)]dx = \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} S''(x)[S''(x) - f''(x)]dx$$

Now we use integration by parts to solve the integrals

Cubic Splines

Recall: $$\int_a^b u(x)v'(x)dx = u(x)v(x)\Big|_a^b - \int_a^b u'(x)v(x)dx$$

$$\int_a^b S''(x)\big(S''(x)-f''(x)\big)dx = \sum_{k=0}^{n-1}\left(S''(x)\big(S'(x)-f'(x)\big)\Big|_{x_k}^{x_{k+1}} - \int_{x_k}^{x_{k+1}} S'''(x)\big(S'(x)-f'(x)\big)dx \right)$$

The first term is

$$\sum_{k=0}^{n-1} S''(x)\big(S'(x)-f'(x)\big)\Big|_{x_k}^{x_{k+1}} = S''(b)\big(S'(b)-f'(b)\big) - S''(a)\big(S'(a)-f'(a)\big)$$

$= 0$ From the boundary conditions for natural cubic spline

$S'''(x)$ is a constant (since we have a cubic) and can be taken out of the integral, so

$$\int_a^b S''(x)\big(S''(x)-f''(x)\big)dx = -\sum_{k=0}^{n-1} S_k'''\int_{x_k}^{x_{k+1}}\big(S'(x)-f'(x)\big)dx$$

$$= \sum_{k=0}^{n-1} S_k'''\big(S(x)-f(x)\big)\Big|_{x_k}^{x_{k+1}}$$

but since $S(x_k) = f(x_k)$, $\quad = 0$

Cubic Splines

$$\int_a^b [f''(x)]^2 dx - \int_a^b [S''(x)]^2 dx \geq 0$$

We have proven that a cubic spline has a smaller or equal curvature than any function which fulfills the interpolation requirements. This also includes the function we started with.

Physical interpretation: a clamped flexible rod picks the minimum curvature to minimize energy - spline

Data Smoothing

If we have a large number of data points, interpolation with polynomials, splines, etc is very costly in time and multiplies the number of data. Smoothing (or data fitting) is a way of reducing. In smoothing, we just want a parametrization which has no model associated to it. In fitting, we have a model in mind and try to extract the parameters.

Data fitting is a full semester topic of its own.

A few brief words on smoothing of a data set. The simplest approach is to find a general function with free parameters which can be adjusted to give the best representation of the data. The parameters are optimized by minimizing chi squared:

$$\chi^2 = \sum_{i=0}^{n} \frac{(y_i - f(x_i;\vec{\lambda}))^2}{w_i^2}$$

Data Smoothing

$$\chi^2 = \sum_{i=0}^{n} \frac{(y_i - f(x_i;\vec{\lambda}))^2}{w_i^2}$$

$\vec{\lambda}$ are the parameters of the function to be fit

y_i are the measured points at values x_i

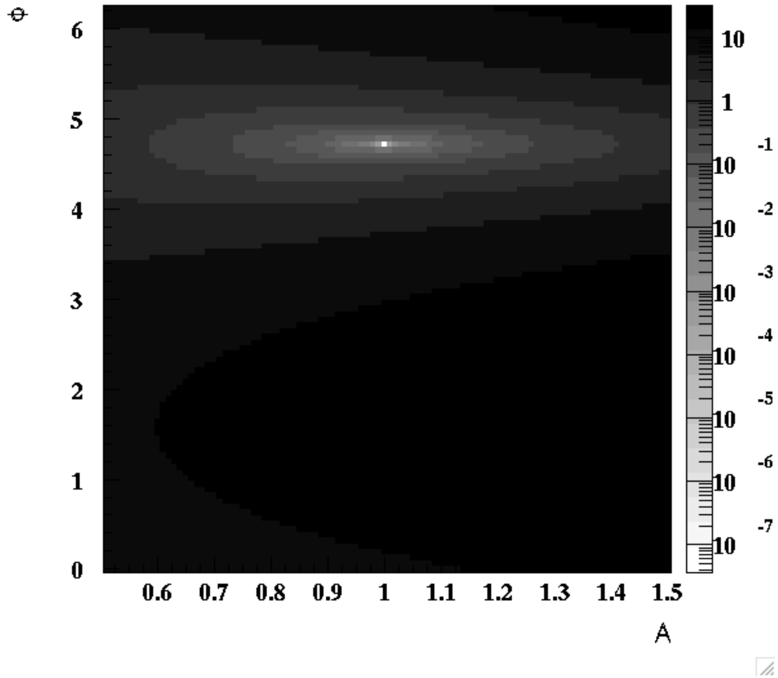
w_i is the weight given to point i

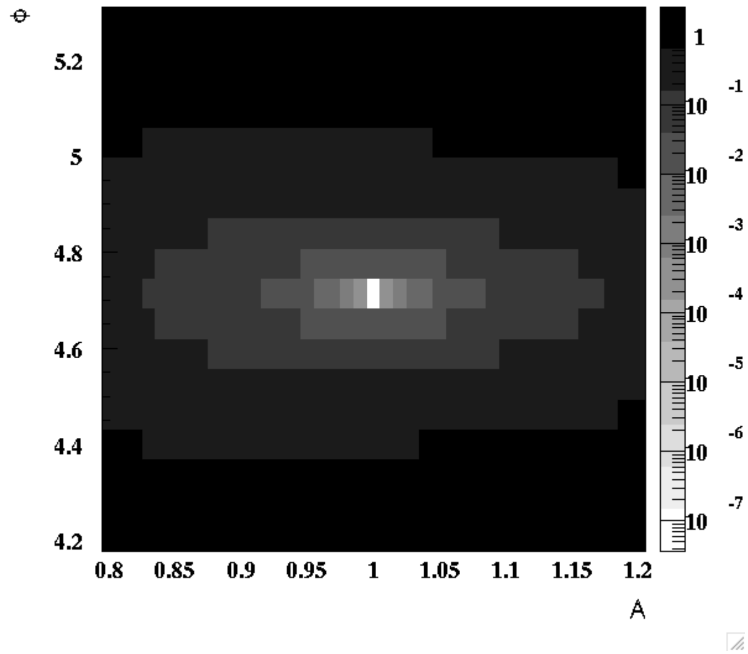
In our example, let's take $f(x; A, \vartheta) = A\cos(x + \vartheta)$

And set $w_i = 1 \quad \forall i$

Now we minimize χ^2 as a function of *A* and φ

Data Smoothing





Best fit for $A=1$, $\varphi=3\pi/2$

$$A\cos(x+\vartheta) = 1\cdot\cos(x+3\pi/2) = \cos x\cos 3\pi/2 - \sin x\sin 3\pi/2 = \sin x$$

Exercises

1. Calculate the Lagrange Polynomial for the following data:

x	y
0	1
0.5	0.368
1.	0.135
2.	0.018

2. For the same data, find the natural cubic spline coefficients. Plot the data, the lagrange polynomial and the cubic spline interpolations.
3. Smooth the data in the table with the function $f(x)=A\exp(-bx)$. What did you get for A,b ?