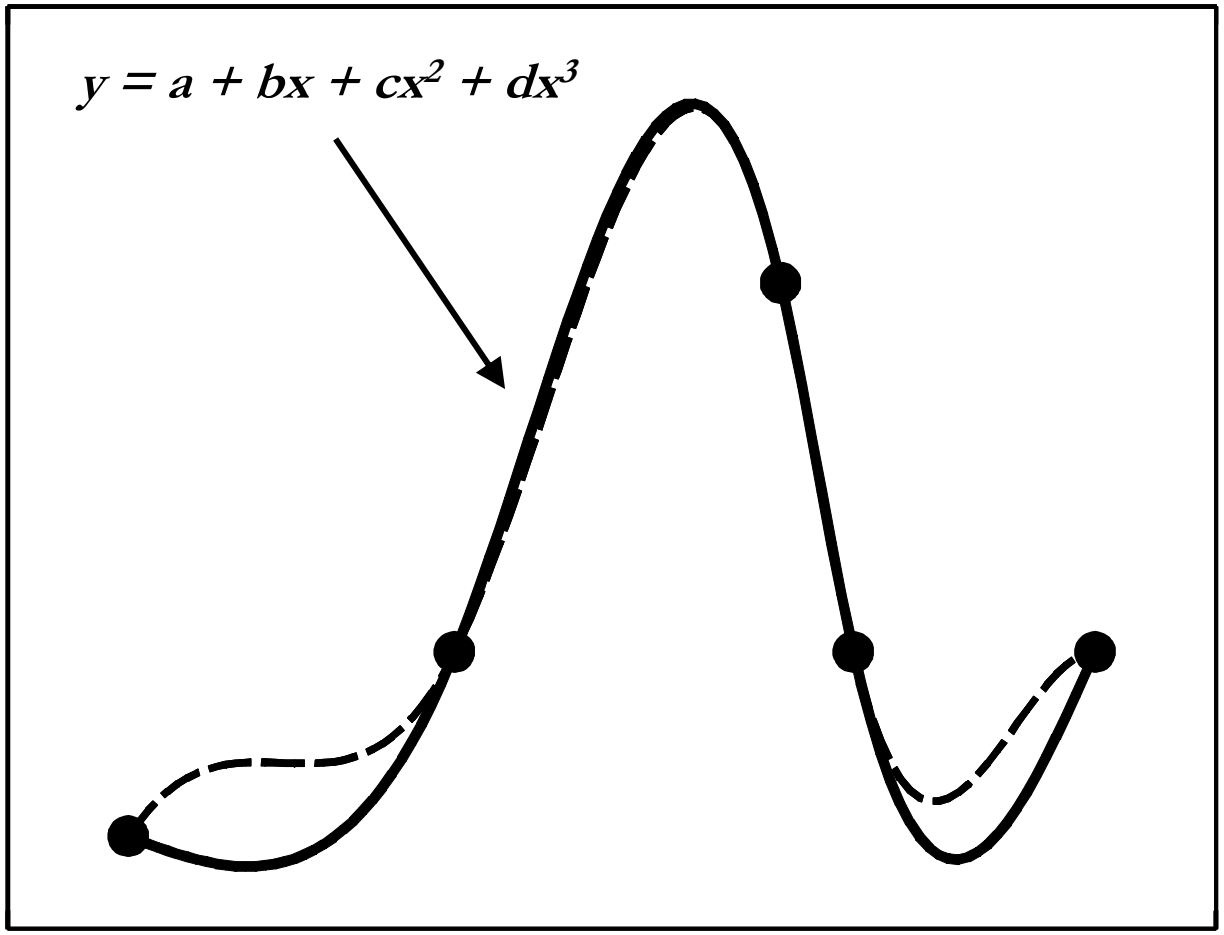
Introduction

The cubic spline is a standard mathematical function commonly used to interpolate a data set. (The *spline* is a pre-CAD draftsman's tool which can be bent into a smooth shape and subsequently holds that shape.) This lecture breaks our policy of not covering merely mathematical topics because we plan to use the cubic spline in a number of ways, many not found in standard textbooks:

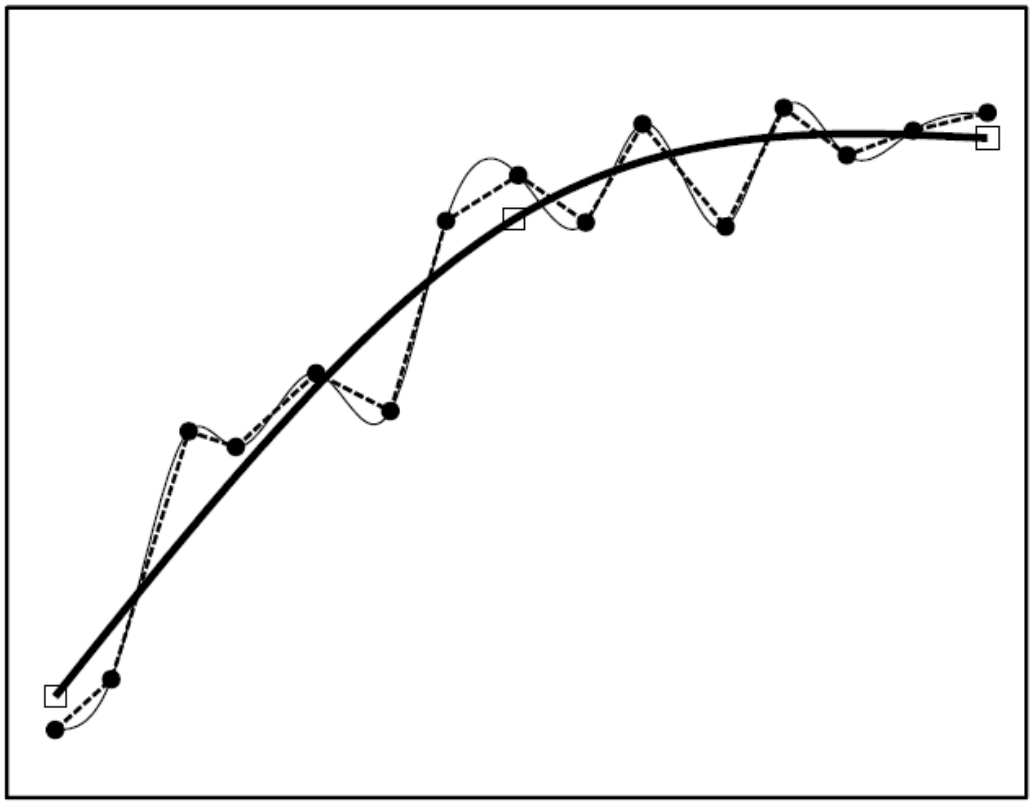
1. As an **interpolating** function (the normal use) in `RANGE`;
2. As a **fitting** function for pristine Bragg peaks (`BPW`);
3. As a **fitting** function for data sets with **corners** (`FitDD`);
4. As an **adjustable smooth curve** (contoured scatterer, `NEU`).

We will describe each of these applications in turn, beginning with a reminder of the difference between *interpolation* and *fitting*. We will not explain how to compute the polynomial coefficients that define a given cubic spline. For that, we refer the student to Numerical Recipes.



In this example, five arbitrary points (defining four *intervals*) are joined by a cubic spline. In each interval the four coefficients ***a...d*** are different. Their 16 values are fixed by requiring that the curve *pass through the points* and that its *slope be continuous* at each point (no 'corners'). At each endpoint we must therefore either *specify the first derivative* (dashed line) or require that the *second derivative be zero* ('natural' cubic spline, full line).

Interpolation vs. Fitting



The solid circles are input data, intentionally noisy. The linear and cubic spline *interpolation* lines pass through all the given points (illustrating a weakness of spline interpolation). The bold line, however, is a cubic spline *fit* obtained by adjusting the three points shown as hollow squares for least-squares deviation of the *bold line* interpolating them (with a cubic spline) from the *data points*. You will argue (correctly) that a low-order polynomial would do just as well here, but we will soon see cases where *any* single polynomial — even high order — fails of a good fit.

SPLINE.FOR Implementation

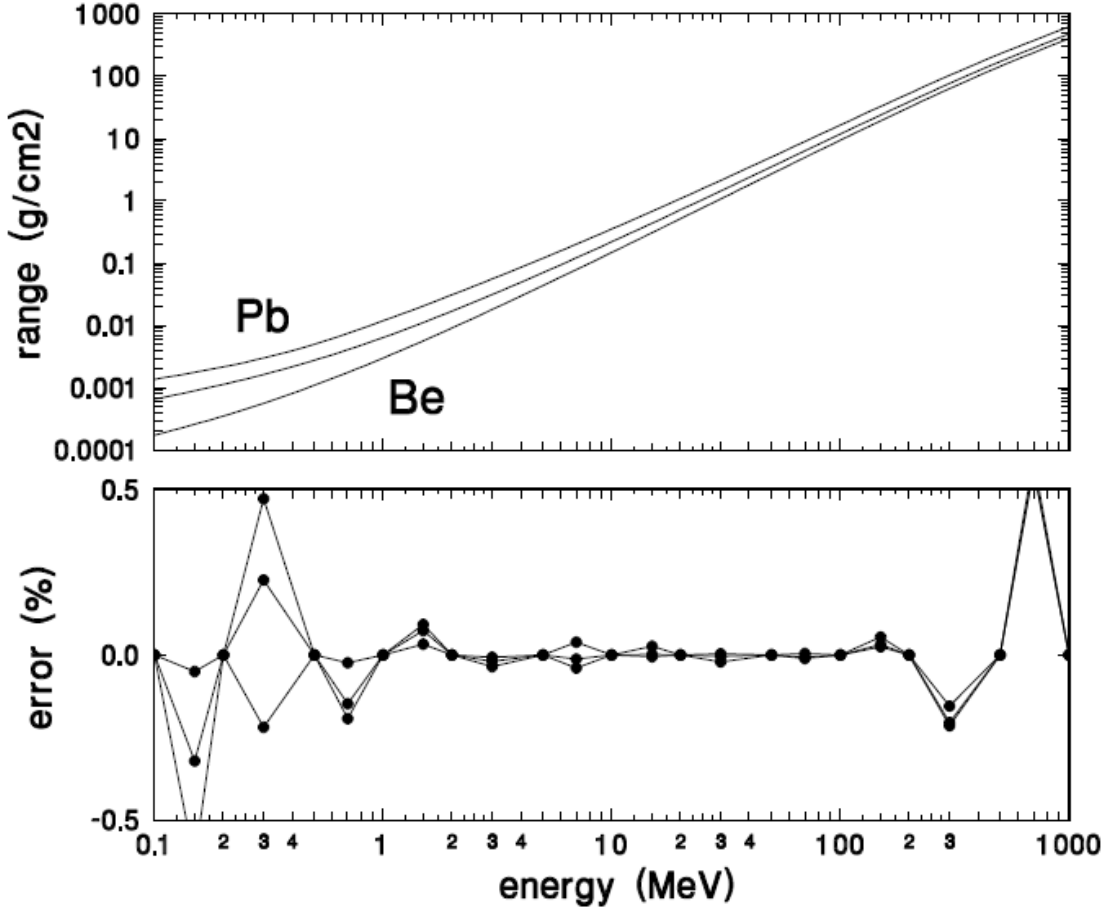
This differs slightly from Numerical Recipes' version. `InitSpline` sets up scratch array `s` (second derivatives) to go along with given arrays `x` and `y`. `InitSpline` does a fair amount of computation and should be invoked ***only when the x, y table changes***.

```
DIMENSION x(n),y(n),s(n)
DATA yp1,ypn/0.,1.E30/
          ...
i = InitSpline(n,x,y,s,yp1,ypn,xx,klo,khi)
          ...
ya = Spline(n,x,y,s,yp1,ypn,xa,klo,khi)
yb = Spline(n,x,y,s,yp1,ypn,xb,klo,khi)
```

`yp1` and `ypn` as given yield y' = 0 at one end and y'' = 0 ('natural' boundary) at the other (the non-physical value `1.E30` serving as a code here). The next line sets `ya` to the value of the spline at an arbitrary `xa` and so forth.

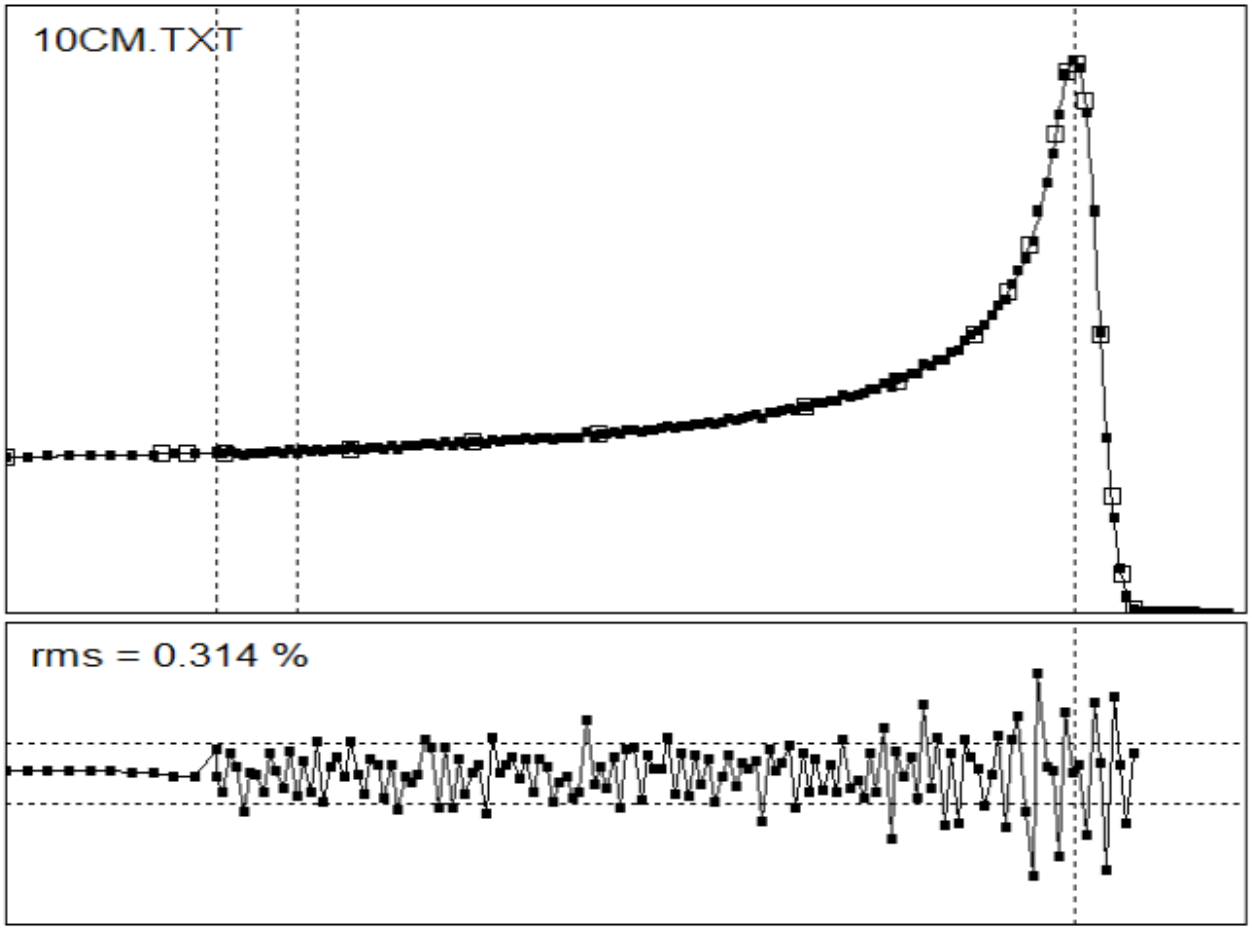
One virtue of the cubic spline as a fitting function is that it extrapolates reasonably well just *outside* the defined region whereas high order polynomials tend to go berserk there.

Example: Cubic Spline Interpolation (RANGE)



We express the range-energy relation in any given material by cubic spline *interpolation* of log(R) as a function of log(T), equivalent to using a variable power law. Only 13 values need be entered from 0.1 to 1000 MeV and, as the lower panel shows, the accuracy is ~0.1% from 3 to 300 MeV.

Example: Cubic Spline Fit (BPW)



Bragg peaks are described by a cubic spline *fit* to measured data (BPW). The adjustable points are the hollow squares. The lower panel shows the accuracy of the fit. The left end (*not measured* because of wall thickness) is a guess obtained by linear extrapolation. Bragg peaks are extremely hard to fit with polynomials!

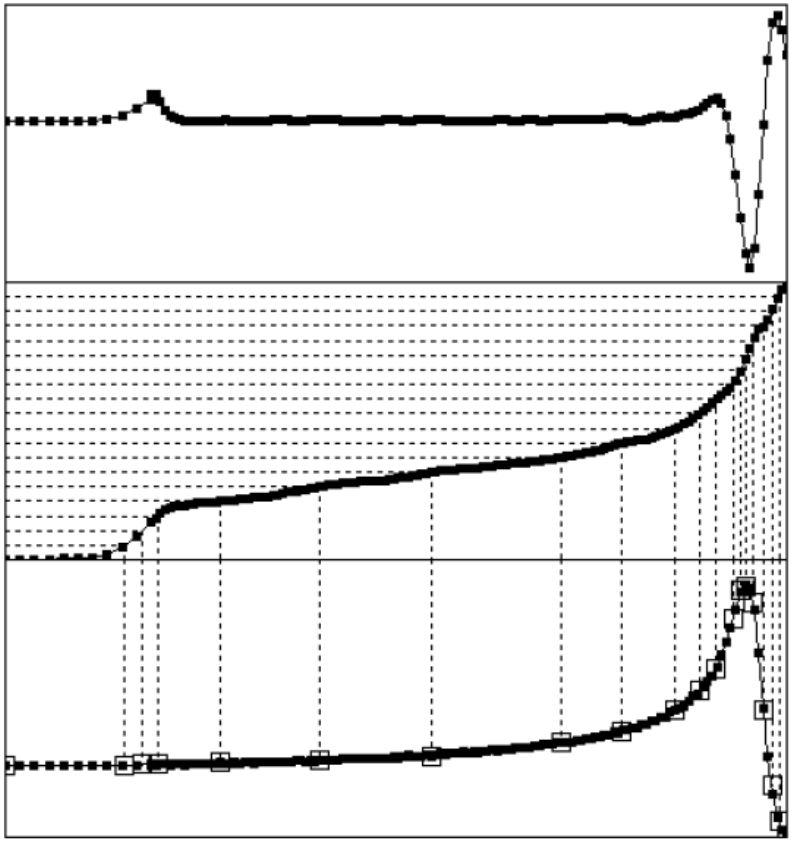
BPW Standard Output File

Once the measured Bragg peak has been fit, the fit parameters (namely the hollow squares of the previous slide) can be output in a standard form. In this example only 20 output points are used to characterize 152 raw data points. The distilled version is actually better for most purposes because experimental noise has been averaged out.

```
put a comment here ...
249.0  cm from source to Bragg peak
20   0.10000E+31   0.10000E+31  # pts, yp1, ypn
 0.0000    2.1855    2.7645    3.1580    5.1416    6.8809    8.7197  cm H2O
11.8539   13.2038   14.3375   14.8641   15.1743   15.5595   15.7307
15.8623   15.9955   16.2209   16.3838   16.5357   16.7100
 0.2824    0.2860    0.2870    0.2878    0.2954    0.3072    0.3217  rel dose
 0.3732    0.4204    0.5041    0.5818    0.6669    0.8650    0.9784
 0.9948    0.9272    0.5038    0.2085    0.0663    0.0065
```

The application using this file must be able to correct the Bragg peak to a standard condition — the depth-dose that would have been measured under the same circumstances with an *infinite* source distance — because the source distance at 'application time' will usually not be the same as it was at 'Bragg measurement time'. To do this, the application needs to know the effective source distance at Bragg measurement time.

Choosing Initial Points for the Spline Fit



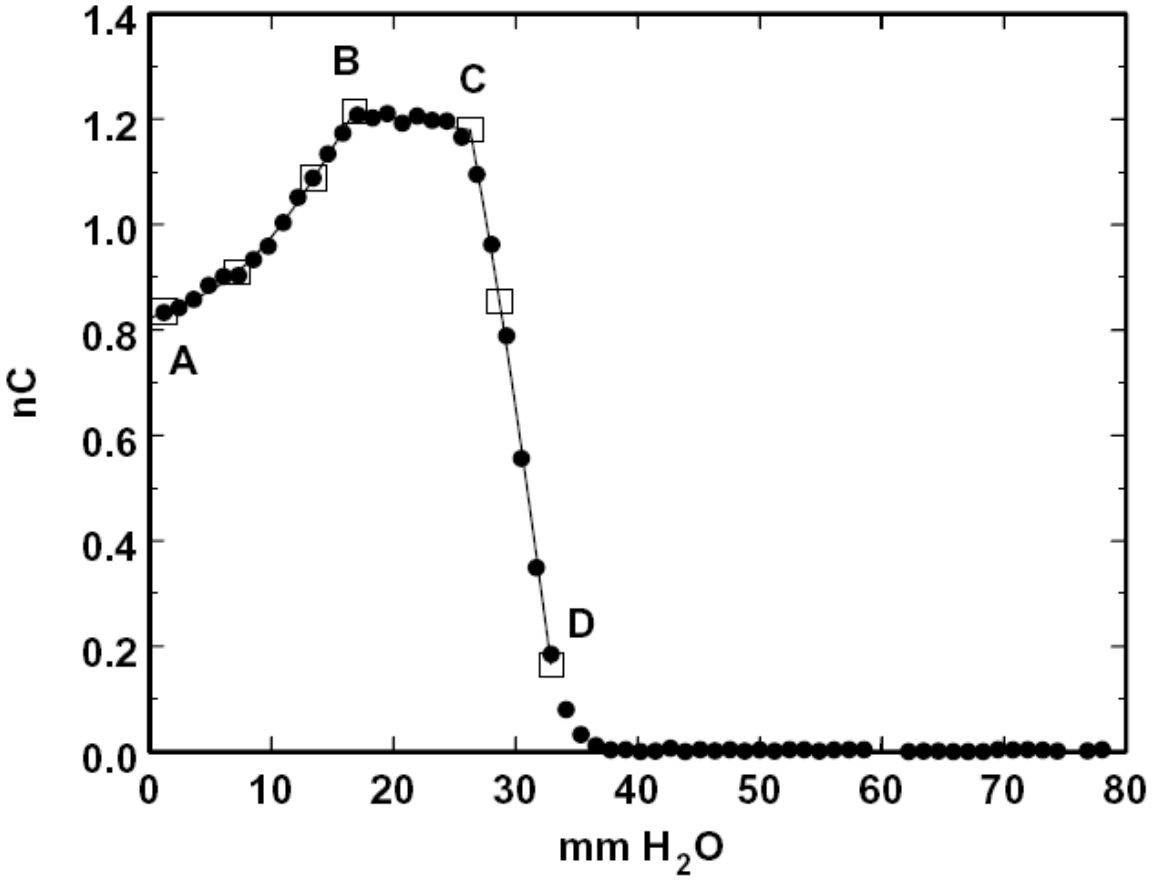
smoothed 2nd derivative

density function

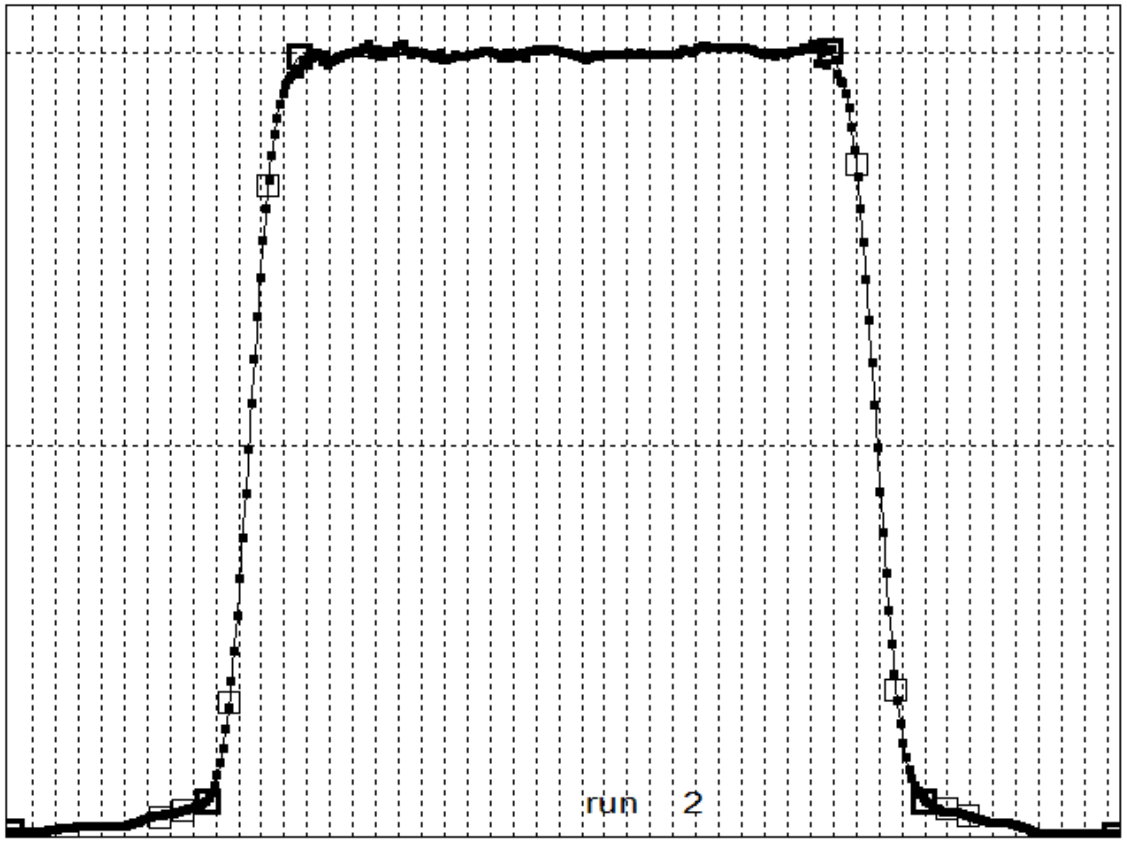
function to be fit

The hardest part of spline fitting is picking *initial points*. We don't want *uniform* spacing, because if we have enough points in the curvy parts we'll have far too many in the straight parts and the fit will start chasing noise. We project equally spaced points onto a density function which varies most rapidly where the absolute second derivative is greatest. See the `FitDD` User Guide for details.

The 'Broken Spline' Fit



A broken spline is a set of N connected natural cubic splines, single valued and continuous with a continuous first derivative except at the joints or 'corners'. During optimization the initial break points migrate to the true corners of the data set. This identifies points such as B and C facilitating subsequent computation of such things as the 100% dose level and d_{80}. As the example shows, a natural cubic spline through 2 points (B and C) is a straight line. See the User Guide to `FitDD` for details.

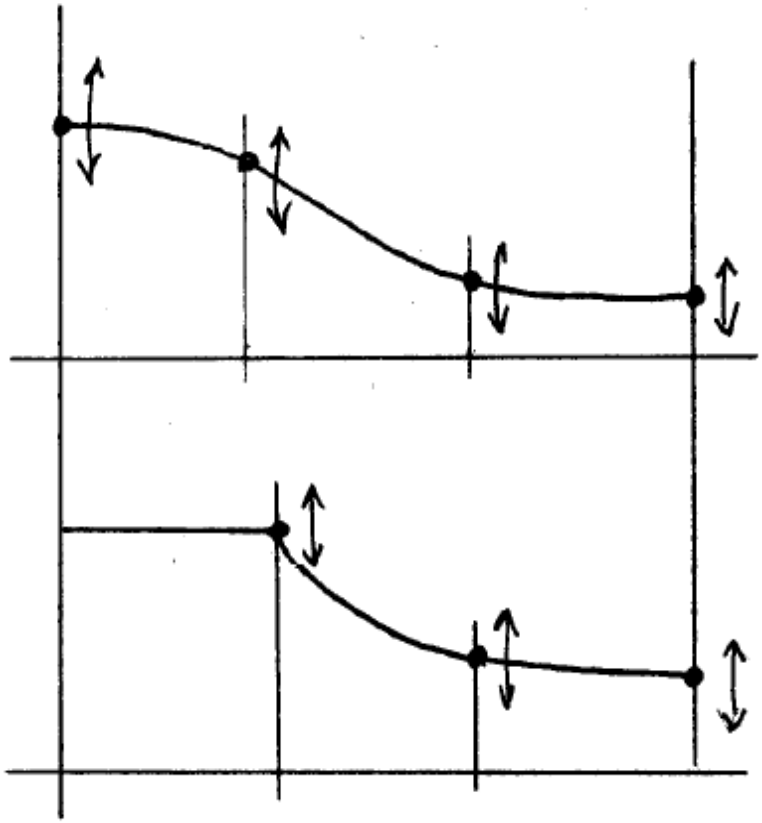


A 5-segment broken spline used to fit a transverse scan. The fitting process itself finds the four corners (bold squares). That in turn makes it easy to define the 100% dose level or the maximum dose `yPeak` and its position `xPeak`. Subsequently, the simple statement

```
xL90 = Rtfp (BSfunc, x1, xPeak, 0.9*d100, kErr)
```

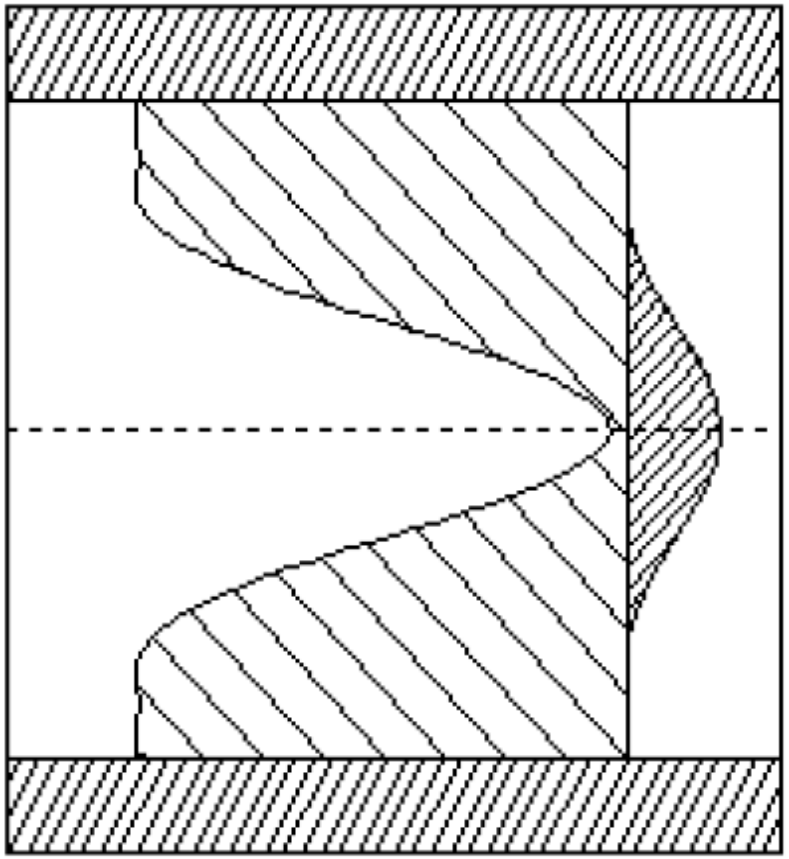
serves to find the left-hand 90% point (by the method of False Position) because `BSfunc` is a single continuous function valid over the entire transverse range.

The Cubic Spline as an Adjustable Smooth Curve



The cubic spline is a handy universal adjustable smooth curve. Here it is used to represent what will be, after optimization of the ***y***'s, the generic recipe for a contoured scatterer. The lower curve is a parametrization of the Siebers variant of the contoured scatterer. Contoured scatterers are not Gaussian in profile! Alternative parametrizations of the profile are possible: for instance, the Uppsala group uses a 9-segment distorted cosine curve.

A Compensated Contoured Second Scatterer



And here is the end product (to scale, thickness exaggerated). The profiles of both the lead dome and the energy-compensating plastic plate are cubic splines in thickness v. radius. For accuracy in compensation, far more points are used here than were used in the definition and optimization of the generic shape.

Summary

The cubic spline is a smooth function defined by the set of points through which it passes plus a derivative condition at each end. In each interval it is a cubic polynomial. Numerical Recipes explains how to find the polynomial coefficients. Our routines `InitSPLINE` and `SPLINE` are implementations of their method.

We use the cubic spline four ways:

1. As an interpolating function (`RANGE`);
2. As a fitting function, especially for Bragg peaks (`BPW`);
3. In a 'broken spline' version, as a fitting function (`FitDD`);
4. As a universal adjustable smooth curve (contoured scatterer, `NEU`).

The last three uses are uncommon. The cubic spline can have a complicated shape even in a single interval and therefore permits an excellent model independent fit to many kinds of data. A natural cubic spline between only two points degenerates to a straight line, another useful property. Unlike high order polynomials, the cubic spline can be made to extrapolate gracefully.