3. Regularization

Since we got a nice taste of what regularization is, let's see a generic problem of the type that is usually handled with regularization. The simplest such problem is smooth interpolation in one dimension. This is an important problem, not because it is such a good way to interpolate, but because it can be generalized in many different ways and form the basis of many very applied algorithms like "snakes", "membranes", and "thin plates". Despite their silly names these are very important and very applied techniques (I do not want to scare you from going to the doctor) in medical imaging and other areas.

The problem is best explained by Fig. 3.1. We want to find a line that goes "near" near these points and is "smooth". Both the words "near" and "smooth" admit different interpretations but we aim for the simplest. For the "near" we have only one choice if we want to keep things simple and this is least squares. For the "smooth" we have more choices but all of them involving a form of least squares. We can minimiza the square of the first derivative, the square of the second or a weighted sum of the two. We can go to the third derivative or beyond but this is not very common. Finally we have to decide how much emphasis we put on the nearness or the smoothness which is controlled by the parameter λ in the expression

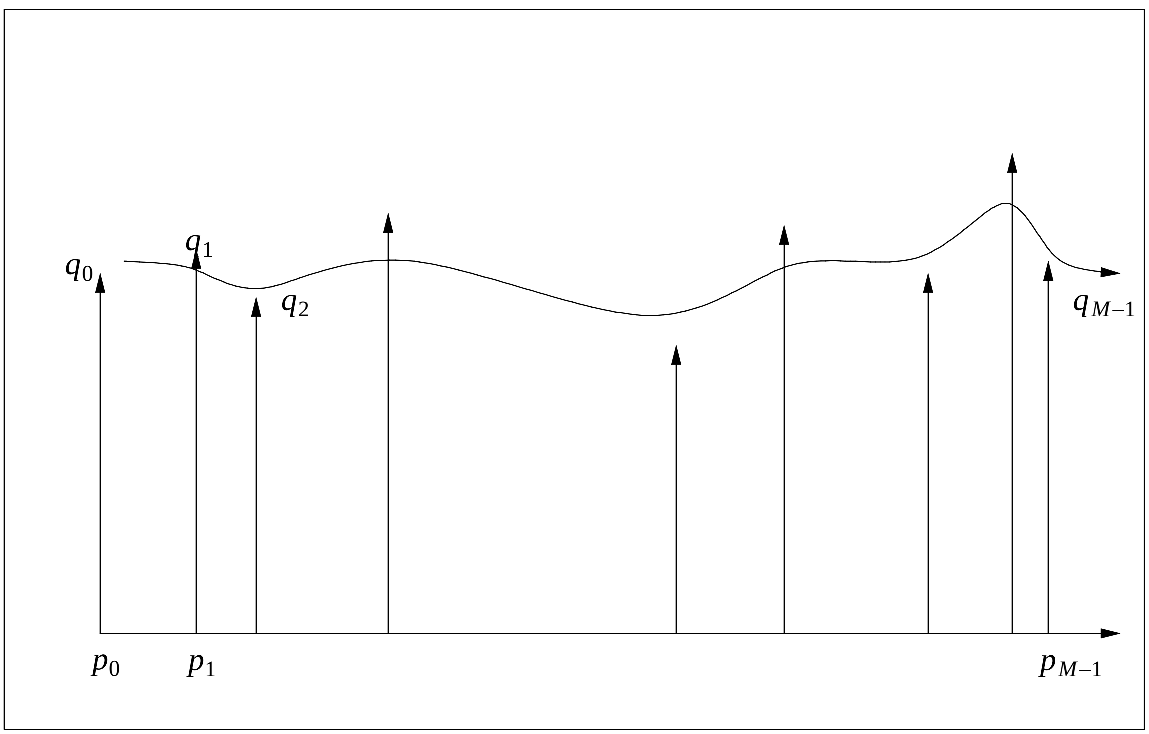

Figure 3.1: The question is how to find a line that passes "near" all these points and is "smooth". The word "smooth" can have different meanings as can the word "near".

$$S = S_{ph} + \lambda S_{sm}$$

where S is the expression we want to minimize, S_{ph} represents the physical constraint, that relates to the nearness and S_{sm} represents the smoothness.

Let us now start with the physical contraint. We have to determine a function $f(x)$ that passes near the points $[x_i, y_i]$, or in more mathematical terms

$$f(p_i) - q_i$$

is small for all i. Since we know and love least squares, that's what we use:

$$S_{ph} = \sum_{i=0}^{M-1} (f(p_i) - q_i)^2. \tag{3.1}$$

3.1. Representation of an Arbitrary Function.

The next problem is how we represent function f. The choices are many. We could, for example approximate f with a polynomial. But we would need a very high degree poynomial for a function that tries to pass close to many points and such polynomials are difficult to handle. We could also represent f as a sum of sines and cosines, but this will lead to complicated equations that involve the inversion of large matrices with very feww zeros. The best method is to represent f by a set of regular points that it passes through and interpolate between these points with low degree polynomials. This leads to simple equations that involve the inversion of matrices that are mostly zeros. To make our life even easier we will interpolate between successive points with straight lines. So we define f as

$$f(x) = \begin{cases} 0 \le x < 1: & (1-x)y_0 + x\, y_1 \\ 1 \le x < 2: & (2-x)y_1 + (x-1)y_1 \\ 2 \le x \le 3: & (3-x)y_2 + (x-2)y_1 \\ \cdots & \cdots \\ k \le x \le k+1: & (k+1-x)y_k + (x-k)y_{k+1} \\ \cdots & \cdots \\ N-1 \le x < N: & (N-x)y_{N-1} + (x-N+1)y_N \end{cases} .$$

where y_0, y_1, etc are the unknown parameters and once we have them then we know function f. Now for every i the corresponding term in the summation of Eq. (3.1)

$$(f(p_i) - q_i)^2$$

if we set $k = \lfloor p_i \rfloor$, in other words k is the biggest integer that does not exceed p_i, becomes

$$((k+1-p_i)y_k + (p_i - k)y_{k+1} - q_i)^2.$$

3.2. Dealing with the Physical Constraint

To minimize S_{ph} we differentiate with respect to the unknowns y_l for $l = 0 \cdots N$, but for every l only the term that happens to have a y_k such that $k = l$ or a y_{k+1} such that

$y+1=l$ will be non zero. So each term in the summation of Eq. (3.1) will contribute one equation when $l=k$ and one more when $l=k+1$. The first differention then looks like

$$\frac{\partial}{\partial y_k}((k+1-p_i)y_k+(p_i-k)y_{k+1}-q_i)^2 =$$
$$2(k+1-p_i)((k+1-p_i)y_k+(p_i-k)y_{k+1}-q_i)$$

and finaly we get this equation

$$(k+1-p_i)^2 y_k+(p_i-k)(k+1-p_i)y_{k+1}=(k+1-p_i)q_i.$$

Similarly, when we differentiate with respect to y_{k+1} we get

$$(k+1-p_i)(p_i-k)y_k+(p_i-k)^2 y_{k+1}=(p_i-k)q_i.$$

All is nice and good, but what do we do with these two equations. Where do they fit and how do we find the y_k's. These two equations can be seen as the following system of linear equation written in matrix form

$$\begin{bmatrix} (k+1-p_i)^2 & (p_i-k)(k+1-p_i) \\ (p_i-k)(k+1-p_i) & (p_i-k)^2 \end{bmatrix}\begin{bmatrix} y_k \\ y_{k+1} \end{bmatrix}=\begin{bmatrix} (k+1-p_i)q_i \\ (p_i-k)q_i \end{bmatrix}.$$

We can write the same equation in an even better way by including all the y_k's and a big square matrix in which all but four of its elements are zero and similarly for the vector of the knowns on the right hand side

$$\begin{bmatrix} \cdot & & & & & & & \\ & \cdot & & & & & & \\ & & \cdot & & & & & \\ & & & (k+1-p_i)^2 & (p_i-k)(k+1-p_i) & & & \\ & & & (p_i-k)(k+1-p_i) & (p_i-k)^2 & & & \\ & & & & & \cdot & & \\ & & & & & & \cdot & \\ & & & & & & & \cdot \end{bmatrix}\begin{bmatrix} y_0 \\ \cdot \\ \cdot \\ y_k \\ y_{k+1} \\ \cdot \\ \cdot \\ y_N \end{bmatrix}=\begin{bmatrix} 0 \\ \cdot \\ \cdot \\ (k+1-p_i)q_i \\ (p_i-k)q_i \\ \cdot \\ \cdot \\ 0 \end{bmatrix}.$$

This is an equation that has an enviable property. It is tridiagonal, ie only three diagonals have non zero elements. Now notice that for every p_i we get one $k=\lfloor p_i \rfloor$ and one such equation as above. Then all we have to do is add these equations together and since the matrices are all tridiagonal, we get a tridiagonal matrix in the end. The matrix might not be invertible, but then we are not done yet. We still have the smoothing term. But before we go there, in case anyone is curious what would happen if instead of linear interpolation we used quadratic or, even better, cubic, the matrix would have five or seven non zero diagonals. These are not as easy to invert as tridiagonal, but still better than full dense matrices.

3.3. Dealing with the Smoothness Constraint

The smoothness component of the least squares has to take the form of the square of the first derivative. Everything else would be too much work for our sensitive minds. So

$$S_{sm} = \sum_{i=0}^{N-1} (y_{i+1} - y_i)^2$$

where we approximate the first derivative with a finite difference. We could use something fancier but then we are discussing the basic principles, not showing off our tolerance to mathematical brutality. Differentiating with respect to the unknowns y_l we get

$$\frac{\partial}{\partial y_l} \sum_{i=0}^{N-1} (y_{i+1} - y_i)^2 =$$

$$\sum_{i=0}^{N-1} \frac{\partial}{\partial y_l} (y_{i+1} - y_i)^2 =$$

$$\sum_{i=0}^{N-1} 2\left(\frac{\partial (y_{i+1} - y_i)}{\partial y_l} (y_{i+1} - y_i) \right)$$

$$2 \sum_{i=0}^{N-1} (\delta(i+1-l) - \delta(i-l))(y_{i+1} - y_i)$$

$$2 \sum_{i=0}^{N-1} \delta(i+1-l)\, (y_{i+1} - y_i) - 2 \sum_{i=0}^{N-1} \delta(i-l)\, (y_{i+1} - y_i)$$

where

$$\delta(n) = \begin{cases} n = 0: & 1 \\ n \neq 0: & 0 \end{cases}$$

which means that from the each summation the only terms that survive are the ones where $i+1 = l$ or $i = l-1$ and $i = l$ respectively. So the derivative of S_{sm} with respect to the unknowns y_l becomes

$$2(y_l - y_{l-1}) - 2(y_{l+1} - y_l)$$

and since this is equated to zero we finally get

$$-y_{l-1} + 2y_l - y_{l+1} = 0.$$

It is hard not to notice that it is essentially the second derivative as we could have guessed if we remembered anything at all from the previous section. Anyway, if we write it in matrix form as above, for every y_l we get

$$\begin{bmatrix} \cdots & -1 & 2 & -1 & \cdots \end{bmatrix} \begin{bmatrix} y_0 \\ . \\ . \\ y_{l-1} \\ y_l \\ y_{l+1} \\ . \\ . \\ y_N \end{bmatrix} = 0$$

Since we get this single equation for y_l, if we combine the equation for all the y_l's we get

$$\begin{bmatrix} 2 & -1 & & & & & & \\ -1 & 2 & -1 & & & & & \\ & -1 & 2 & -1 & & & & \\ & & . & . & . & & & \\ & & & . & . & . & & \\ & & & & -1 & 2 & -1 & \\ & & & & & -1 & 2 & -1 \\ & & & & & & -1 & 2 \end{bmatrix} \begin{bmatrix} y_0 \\ . \\ . \\ y_{l-1} \\ y_l \\ y_{l+1} \\ . \\ . \\ y_N \end{bmatrix} = \begin{bmatrix} 0 \\ . \\ . \\ . \\ \\ . \\ . \\ . \\ 0 \end{bmatrix}$$

which involves another tridiagonal matrix. We can add the two tridiagonal matrices, and the corresponding vectors of knowns (the one of them is conveniently all zeros) and get the final tridiagonal system of equations to solve. The solution is particularly fast because mathematicians, being very smart, have invented nice methods to solve them fast. Unfortunately, when we go to two dimensions, we are not so lucky. The mathematicians that worked on that particular kind of matrices (tridiagonal with fringes) were not as smart and we do not have that good solutions.