

Re: Non-linear least squares using piece-wise approximation....

I've not explained this very well. It is similar to a problem I had posted sometime ago:

http://groups.google.com/group/sci.math.num-analysis/browse_thread/thread/5c3046a643bcd710/8160bd6b58

In the original problem:

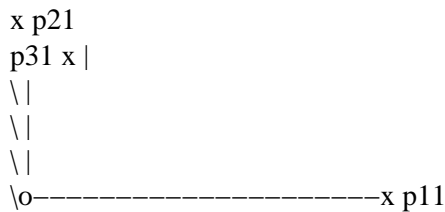
"I have a set of m points controlled linearly by n variables. The variables move the points along line segments.

For a single point I have:

$$p' = p_0 + \sum_{(i=1 \text{ to } n)} \{ (p_i - p_0) * t \}$$

where p_i is the position of the point when the i th variable has a value of 1. p_0 is the rest position of the point when the i th variable

is 0. In this case p_{0i} is the same so I am using the notation p_0 . The simple diagram below illustrates how a single point might be affected by three different variables.



but for different points the movement directions are different!?

My goal is to find values for the variables t that minimizes the distance between the point and its target position. By defining d_i as the delta of $(p_i - p_0)$ the problem can be stated in matrix-vector form:

$$\begin{bmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \dots & d_{2n} \\ \dots & \dots & \dots & \dots \\ d_{m1} & d_{m2} & \dots & d_{mn} \end{bmatrix} z = [p_{10} \ p_{20} \ \dots \ p_{m0}]'$$

$$X = \begin{bmatrix} \dots \\ \dots \\ \dots \end{bmatrix} y = [t_{p1} \ t_{p2} \ \dots \ t_{pm}]'$$

d_{21} is the direction the point p_2 moves when $t(1)=1$
 d_{m1} is the direction the point p_m moves when $t(1)=1$

Re: Non-linear least squares using piece-wise approximation....

X is a $m \times n$ matrix containing the n delta values for the m points.
z is a m -vector containing the original position of each point
y is a m -vector containing the target positions for each point.
t is a n -vector containing the variables to be optimized.

The problem can now be stated in matrix-vector terms as:

$$y = z + (X * t)$$

and solved as a least-squares problem."

What's different now, is that each of the n controls no longer moves the m points along line segments, but rather along some non-linear path by some unknown function. For a single point, I have:

$$p'_i = \sum_{k=1, \dots, n} f_{ik}(x_k)$$

$$\text{low}(k) \leq x(k) \leq \text{up}(k) \quad k=1, \dots, n$$

Additionally there are mutual exclusivity constraints e.g. x_0 and x_7 cannot both be non-zero.

My initial idea was that I could approximate the nonlinear paths with piece-wise linear ones. I'm thinking this could be solved using a constrained, nonlinear least-squares solver, but I'm not clear as to how I would specify the mutex constraints?

Thanks!

yes, I understood this well, at least following your notation:
for the different points, the nonlinear paths are different, so you have $m \times n$ paths
but in your linear example, the paths are given by the d_{ij}

here you say the paths are unknown???
the question is: how to represent such an unknown path: with piecewise linear functions this becomes even more involved (e.g. using SOS2)
my idea was to represent those paths by a finite set of unknown points in space, interpolating these by a cubic spline, then to minimize the overall goal with the $x(i)$ = positioning variable and the points representing the paths simultaneously

the modelling of the mutual exclusion constraints is already given in my previous post:

say $x(1)$, $x(3)$ and $x(7)$ cannot be nonzero simultaneously:

writing this as $x(1) * x(7) = 0$

Re: Non-linear least squares using piece-wise approximation....

$$x(1)*x(3)=0$$

$$x(3)*x(7)=0$$

("complementarity constraints")

is all but a good idea: it not only introduces nasty nonlinearity but also singularities in cases where more than one $x(j)$ is zero.

better write this as

$$y(1)*x(1)+y(3)*x(3)+y(7)*x(7) \text{ in } [0,1] \text{ (if you don't know the required outcome)}$$

$$y(1)+y(2)+y(3)=1 \text{ one variable nonzero at most}$$

$y(1), y(3), y(7)$ either 0 or 1 (zero-one-variables)

but this brings you in the field on mixed zero one nonlinear optimization.

possible, but hard

hth

peter

.