Subject: Re: poly_fit for less number of points
Posted by Yngvar Larsen on Tue, 07 May 2013 08:20:56 GMT
View Forum Message <> Reply to Message

On Monday, 6 May 2013 22:26:51 UTC+2, Craig Markwardt wrote:
> On Sunday, May 5, 2013 6:03:25 PM UTC-4, David Fanning wrote:
>
>> Craig Markwardt writes:
>
>>> This fit,
>>>   res = poly_fit(x-mean(x), y, 2)
>>> gives a smooth fit.
>
>> What is the principle here that made you think of this solution and
>> caused it to work?
>
> Well, not much. He asked for a smooth function and I gave it to him. Jeremy is right, there is a round-off problem that makes the original POLY_FIT(X,Y,2) not work. Subtracting the mean of X (and/or Y) is a classic solution to the problem of round-off.

Often you also want to _normalize_ the variation of X (and possibly also Y), such that X is bounded within [-1,1], e.g.

Xmid = 0.5*(min(X)+max(X))
Xinterval = max(X)-min(X)
Xnorm = 2*(X-Xmid)/Xinterval

This makes all powers of Xnorm to also be bounded within [-1,1], which is good for preservation of precision. In the particular example given by OP, all X values are bounded in an interval with a width of around 0.5, so normalization is not really important, just centering.

[Another option is the statistical normalization: Xnorm = (X-mean(X))/stddev(X)]

> As HeinzS intimates, there's really a lot more to the question, like, how smooth should the function be? what degree of polynomial is permissible? how close of an approximation should the function be? what is the tolerance for outliers? why are there no error bars on the data? what is the definition of a "good fit?"
>
> Without any of that information, I felt it worth to just provide *an* answer, even if it wasn't *the* answer to the original implicit question.

... but of course, what Heinz&Craig write here is the real issue.

--
Yngvar