Subject: Re: curve fiting issue Posted by Wasit. Weather on Fri, 12 Dec 2008 06:32:01 GMT View Forum Message <> Reply to Message

```
On Dec 11, 10:02 am, Jeremy Bailin <astroco...@gmail.com> wrote:
> On Dec 10, 10:21 am, Elkunn < Wasit. Weat... @gmail.com > wrote:
>
>
>
>
>
   On Dec 10, 7:54 am, Jeremy Bailin <astroco...@gmail.com> wrote:
>>> On Dec 10, 12:12 am, Elkunn <Wasit.Weat...@gmail.com> wrote:
>>>> Hello,
>>>> I have a data array like this. The 1st value a[0] is somewhat
>>> contaminated and need to be removed.
>>> a=[0.0382000, 0.3919000, 0.3843000, 0.3880000, 0.3720000, 0.4221000,
>>> 0.5966000, 0.8063000, 0.7955000
>>> 0.8022000,0.7941000,0.8149000,0.8170000,0.7212000,0.7299000,
0.7644000,0.773--8000,0.7574000
>>> 0.6756000, 0.6122000, 0.5646000, 0.5595000, 0.5151000]
>>>> I want to remove 1st pixel and replace it by a predicted value from
>>>> curve fitting and smooth the overll data, return them back into
>>> spatial domain. I do not have error values. Is there any simpler
>>> method to do that?
>>>> Thanks a lot!
>
>>>> Elkuun
>>> And do you expect whatever the curve you fit to have compact support
>>> (probably a good idea for most applications, but I have no clue about
>>> in your case) or to depend on all of the other data?
>>> -Jeremy.- Hide quoted text -
>>> - Show quoted text -
>> Thanks for your reply!
>> This is NDVI data of one pixel over one year. I think Gussian curve
>> fit works for that. Some pixels has no clouds, then I just need to
>> smooth the curve, but one for like this, I want to remove the cloud
>> pixel, then predict its value from least-square fitting, then smooth
>> the whole curve.
```

```
>> Thank you!
> Hmmm. Well, you can certainly fit a Gaussian to that, if you have good
> reason to think that it should be a good parameterization... but it
> doesn't *look* that good to me (of course, I have no idea what the
  errors on those points are). It would look something like this:
>
> na = n elements(a)
> gfit = gaussfit(lindgen(na-1)+1, a[1:*], gparms, nterms=3)
> ; replace a[0] with value from Gaussian. x=0 at this point, which
> makes it simpler.
> a[0] = gparms[0] * exp(-0.5*(gparms[1]/gparms[2])^2)
>
> If you want to add more terms, just increase nterms and add gparms[3]
> at the end. For example, to add a linear term (possibly a good idea,
> but I don't know what you'd theoretically expect):
>
  gfit = gaussfit(lindgen(na-1)+1, a[1:*], gparms, nterms=5)
 a[0] = gparms[0] * exp(-0.5*(gparms[1]/gparms[2])^2) + gparms[3]
 -Jeremy.- Hide quoted text -
>
> - Show quoted text -
Thank you Jeremy,
```

Indeed, my data for this point is not work well with Gaussian Fit. But, this helps me to understand the "fit" procedure. I probabely use poly fit, or SDVFIT with user supplied functions.

Cheers!