

---

Subject: Re: Minimization: Determine a constant across data sets

Posted by [Craig Markwardt](#) on Tue, 10 Apr 2012 18:24:16 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

On Tuesday, April 10, 2012 10:58:52 AM UTC-4, justinr...@gmail.com wrote:

> On Thursday, April 5, 2012 10:42:40 PM UTC-4, Justin wrote:

>> Hi all!

>>

>> I have several data sets that follow the form:

>>

>>  $data = A * e^{(-t/t_0)} + y$

>>

>> I suspect EVERY data set to have the same  $t_0$ , but different A & y values. (ie:

(A1,y1),(A2,y2)...) )

>> I can use MPFITFUN to fit A, $t_0$  and y, but the routine determines a least chi-squared such that  $t_0$  is different for every data set.

>>

>> It would seem simple enough to fix  $t_0$  in MPFITFUN if I knew what the value was beforehand, but I don't :) Is there a way to minimize  $t_0$  across several data sets such that A & y are allowed to vary, but  $t_0$  is tied to every data set?

>>

>> I tried doing this in grids, but it was very computationally time consuming to search an unknown gridspace of  $t_0$  in an double for loop.

>>

>>

>> Thanks!

>> Justin

>

>

>

> On Thursday, April 5, 2012 10:42:40 PM UTC-4, Justin wrote:

>> Hi all!

>>

>> I have several data sets that follow the form:

>>

>>  $data = A * e^{(-t/t_0)} + y$

>>

>> I suspect EVERY data set to have the same  $t_0$ , but different A & y values. (ie:

(A1,y1),(A2,y2)...) )

>> I can use MPFITFUN to fit A, $t_0$  and y, but the routine determines a least chi-squared such that  $t_0$  is different for every data set.

>>

>> It would seem simple enough to fix  $t_0$  in MPFITFUN if I knew what the value was beforehand, but I don't :) Is there a way to minimize  $t_0$  across several data sets such that A & y are allowed to vary, but  $t_0$  is tied to every data set?

>>

>> I tried doing this in grids, but it was very computationally time consuming to search an unknown gridspace of  $t_0$  in an double for loop.

```
>>
>>
>> Thanks!
>> Justin
>
> Awesome, I think I got it. Took a while to get the dimensions to agree.
>
> result=mpfitfun('myfun',x,y(*,*),1.,guess,bestnorm=chisq)
>
> FUNCTION myfun, X, P
> ;create the function from the inputs
> s=size(p)
> cols=(s(1)-1)/2
> model=dblarr(cols,n_elements(x))
> for i=0, cols-1 do begin
> model(i,*)=[P[i]* EXP(-x/P[cols])+P[i+cols+1] ]
> endfor
> RETURN, model
```

Now you are an advanced MPFIT user :-)

---