
Subject: Re: I need a bit of help....Convol and functions
Posted by [JD Smith](#) on Mon, 02 Oct 2006 17:14:58 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Sun, 01 Oct 2006 03:42:34 -0700, D.Kochman@gmail.com wrote:

> So, I'm fairly new to IDL (and an organic chemist so programming is not my
> forte), but I'm chunking my way through it. I'm currently in the process
> of modifying a program that fits exponential decays given the impulse
> response function and the decay curve.
>
> I have to remodel it to fit another much more complex function than an
> exponential decay, however with a similar number of parameters. I've
> already fixed the GUI, and changed all the references to the widgets,
> along with adjusting the appropriate arrays. It now compiles after many
> hours of debugging and displays itself appropriately with a dummy function
> with the appropriate amount of parameters.
>
> I'm stuck with now implementing the function itself, any help on the
> implementation will be **highly** appreciated.
>
> The function is an infinite sum convolved with an exponential decay. I've
> done modeling with the sum, and it converges fairly rapidly, and I can
> limit it to 10 terms or so and still get accuracy to 6 decimal places.
>
> Approximately it is:
>
> $\text{Sum}[(-1)^n \cos(nP(1)X) \exp(-(2n)^2 P(2)X), n \rightarrow 0 \text{ to } 10] \text{ convol}$
> $\exp[-X/P(4)]$
>
> **whew**
>
> anyways, I've been working through the documentation on convol, and I find
> it a bit cryptic. I have very few clues how to implement this function in
> code. I'm guessing the first portion (the sum portion) needs to be
> recursively defined in a for loop. Is this the case, or is there a
> shortcut with a sigma type function built in?
>
> However, how do I easily convolve the two functions if they are functions
> and not arrays? Should I just go to fourier space?
>
> Thanks for any help. I don't expect anyone to code this for me, just a
> gentle (or violent) shove in the appropriate direction will be infinitely
> helpful.

For the sum, you can use a total over an array:

`nx=n_elements(X)`

```
n=rebin(transpose(lindgen(11)),nx,11)
func=total((-1.)^n*cos(n*P[1]*X)*exp(-(2.*n)^2*P[2]*X),2)*exp(-X/P[4])
```

Are you sure it's really a convolution? When the kernel size is the same as the thing its convolving, I almost always think "multiplication" not "convolution". Otherwise, it has no scale. If it occurred over some fixed scale X_0 , then a convolution would be more understandable.

For speed, you could cache the "n" array above to avoid having to recreate it, assuming nx doesn't change.

JD
