Subject: Re: I need a bit of help....Convol and functions
Posted by James Kuyper on Mon, 02 Oct 2006 20:00:17 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
> 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:
>>
>> Sum[(-1)^n*cos(n*P(1)*X)*exp(-(2n)^2*P(2)*X), n ->0 to 10] 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 their 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 infinately
>> 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)*ex p(-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".

What you've written is indead the multiplication, and is therefore not a correct response to his request. The true mathematical convolution of f(x) and g(x) is y(x) = the integral over z from -infinity to +infinity of f(x-z)*g(z) dz. It's only possible to approximate the mathematical convolution numerically when you can approximate the infinite integral with acceptable accuracy by integrating over a finite range. That's why the numerical convolutions you've seen have involved small kernals.

Since the relevant integral covers arbitrarily large negative values of x, the exponential terms in the functions he wants to convolve worry me. I'm not sure the integral converges, though it's been more than a decade since the last time I did integrals like that. If it does converge, I suspect it can be convolved analytically for each value of n seperately.
