Subject: Re: Compute area between curves
Posted by pgrigis on Thu, 16 Oct 2008 16:30:44 GMT
View Forum Message <> Reply to Message

mystea wrote:
> Hi James,
>
> I would like to thank you for your reply. It is truly helpful.
>
> Here is an example:
>
> x=dindgen(200)+1
> y=x^(-2)
> q=dblarr(200)
> for i=1,199 do q[i]=int_tabulated(x[0:i],y[0:i])
>
> plot,q
> plot,y
> plot,deriv(y)
>
> plot,sort(q)
>
> x2=x
> x2[60:199]=x2[60]+x2[60]*dindgen(140)
> y2=x2^(-2)
> q2=dblarr(200)
> for i=1,199 do q2[i]=int_tabulated(x2[0:i],y2[0:i])
> plot,q2
> plot,deriv(y2)
> plot,sort(q)
>
>
> As you can see, the results of integration is wiggling here.
> It's not necessary that the polynomial goes negative, its
> probably just because the routine is using different polynomials when
> new terms
> are introduced.

The reason for the wiggling is the sharp spike near 0, which is
sampled differently if you have a different numbers of elements.
If you divide your function in two ranges (say, 1 to 10 and 10 to 200)
the effect should be smaller. Again, let me stress that your problem
arise because of undersampling of the sharp peak near x=1.
As a matter of fact, if you were to integarate starting from the right
hand side (x=200)
instead of the left (x=1), you would get much better results, as the
peak would

not get in the way of all results but the latest ones (which however
are the
dominant factors in the integral).

Ciao,
Paolo

>
> I also figured out that the deriv routine went crazy under some
> circumstance, too.
> In these cases, deriv(y) should go to zero, yet it was oscillating
> crazy.
>
> What is the criteria for deriv to work? Is there a simple-minded
> deriv which doesn't
> go nuts?
>
> Best,
> Gene
>

>> mystea wrote:
>>> Hi everyone,
>>
>>> I am also working on a topic where I need to numerically calculate an
>>> integral

>>> integral, namely,
>>> the area under a curve as a function of x-coordinate.
>>
>> You can't calculate the true indefinite integral using numerical
>> methods; that's something that can only be done by using a symbolic math
>> program like Mathematica.
>>
>>> The procedure int_tabulated only calculates the definite integral,
>>> given tabulated

>>> length nl.
>>
>>> I tried the following fix:
>>
>>> integral=dblarr(nl)
>>> for i=1, nl-1 do integral[i]=int_tabulated(x[0:i],f[0:i])
>>
>> What you're getting by this method is not the indefinite integral, but a
>> tabulation of definite integrals. This can represent the indefinite
>> integral, in much the same sense that your x and f arrays represent the

>> function you want to integrate, but it is not the indefinite integral
>> itself.
>>

>>> result
>>> integral will not be monotone even if f are always positive.
>>
>> That should not be the case for the true integral of a function that is
>> always positive, assuming that the x values are sorted.
>>
>> However, numerical integration always produces no better than an
>> approximation. INT_TABULATED uses a " a five-point Newton-Cotes
>> integration formula", which is basically derived from fitting those five
>> points to a polynomial. The best-fit polynomial could go to negative
>> values within the range of integration, even if all of the data it is
>> being fitted to is positive; in that case, the integral could decrease
>> with increasing x, for some values of x. That seems unlikely, however,
>> if your function is tabulated with sufficient detail.
>>
>> Could you give a simple example that demonstrates the problem you've seen?
