Subject: Compute area between curves
Posted by frankosuna on Sun, 12 Oct 2008 20:16:03 GMT
View Forum Message <> Reply to Message

Dear IDLers,

How can I compute the area between two curves if I have the x and y
values for both?

Subject: Re: Compute area between curves
Posted by mankoff on Sun, 12 Oct 2008 20:57:56 GMT
View Forum Message <> Reply to Message

On Oct 12, 4:16 pm, frankosuna <frankos...@gmail.com> wrote:
> Dear IDLers,
>
> How can I compute the area between two curves if I have the x and y
> values for both?

total(c1) - total(c0)?

  -k.

Subject: Re: Compute area between curves
Posted by frankosuna on Sun, 12 Oct 2008 21:14:14 GMT
View Forum Message <> Reply to Message

Thanks :)

Subject: Re: Compute area between curves
Posted by jameskuyper on Sun, 12 Oct 2008 21:17:46 GMT
View Forum Message <> Reply to Message

frankosuna wrote:
> Dear IDLers,
>
> How can I compute the area between two curves given two functions?

Are the curves closed? That is, do you create the complete curve by
drawing a line from the final <x,y> pair to the first <x,y> pair?

Are the x values the same for the two curves? Are they evenly spaced?

Note: you don't need to post the same question multiple times, this is a newsgroup, not a chat room. Your message will stay up indefinitely. As a general rule, you might have to wait 24 hours or more before getting an answer.

Subject: Re: Compute area between curves
Posted by frankosuna on Sun, 12 Oct 2008 21:35:22 GMT

View Forum Message <> Reply to Message

I am trying to calculate how much of an error there is between two rings. I have two images each with a ring pictured in these two images.

This is from a wireframe generated from a 3d model. The edge is selected by the user and region growing is used to extract that ring.

http://frankosuna.googlepages.com/wireframe2.jpg

This other one is from an edge detected image of Saturn. All the rings are edge detected and the same ring clicked on the first image is clicked on this one. Region growing is done to extract just that one ring.

http://frankosuna.googlepages.com/edgeDetect2.jpg

I have the x and y values for each pixel that make up the ring

Subject: Re: Compute area between curves
Posted by ben.bighair on Sun, 12 Oct 2008 22:59:28 GMT

View Forum Message <> Reply to Message

On Oct 12, 5:35 pm, frankosuna <frankos...@gmail.com> wrote:
> I am trying to calculate how much of an error there is between two
> rings. I have two images each with a ring pictured in these two
> images.
>
> This is from a wireframe generated from a 3d model. The edge is
> selected by the user and region growing is used to extract that ring.
>
> http://frankosuna.googlepages.com/wireframe2.jpg
>
> This other one is from an edge detected image of Saturn. All the
> rings are edge detected and the same ring clicked on the first image
> is clicked on this one. Region growing is done to extract just that
> one ring.
>

> http://frankosuna.googlepages.com/edgeDetect2.jpg
>
> I have the x and y values for each pixel that make up the ring

Hi,

Given that you have the [x,y] coordinates for the rings, why don't you
try using IDL's built-in INT_TABULATED which should be fine for your
task.

As James points out, you'll need to know how the two curves are
registered in the images. If you can't "line up" comparable x
locations then you have a serious problem.

Cheers,
Ben

Subject: Re: Compute area between curves
Posted by ben.bighair on Sun, 12 Oct 2008 22:59:40 GMT
View Forum Message <> Reply to Message

On Oct 12, 5:35 pm, frankosuna <frankos...@gmail.com> wrote:
> I am trying to calculate how much of an error there is between two
> rings. I have two images each with a ring pictured in these two
> images.
>
> This is from a wireframe generated from a 3d model. The edge is
> selected by the user and region growing is used to extract that ring.
>
> http://frankosuna.googlepages.com/wireframe2.jpg
>
> This other one is from an edge detected image of Saturn. All the
> rings are edge detected and the same ring clicked on the first image
> is clicked on this one. Region growing is done to extract just that
> one ring.
>
> http://frankosuna.googlepages.com/edgeDetect2.jpg
>
> I have the x and y values for each pixel that make up the ring

Hi,

Given that you have the [x,y] coordinates for the rings, why don't you
try using IDL's built-in INT_TABULATED which should be fine for your
task.

As James points out, you'll need to know how the two curves are

registered in the images. If you can't "line up" comparable x
locations then you have a serious problem.

Cheers,
Ben

Subject: Re: Compute area between curves
Posted by frankosuna on Mon, 13 Oct 2008 01:31:26 GMT
View Forum Message <> Reply to Message

On Oct 12, 3:17 pm, James Kuyper <jameskuy...@verizon.net> wrote:
> frankosuna wrote:
>> Dear IDLers,
>
>> How can I compute the area between two curves given two functions?
>
> Are the curves closed? That is, do you create the complete curve by
> drawing a line from the final <x,y> pair to the first <x,y> pair?
>
> Are the x values the same for the two curves? Are they evenly spaced?
>
> Note: you don't need to post the same question multiple times, this is a
> newsgroup, not a chat room. Your message will stay up indefinitely. As a
> general rule, you might have to wait 24 hours or more before getting an
> answer.

The curves are not closed... I posted some images of the actual rings
I am trying
to compare. They look like parabolas. The rings might differ in shift
and slight rotation from each other. So because the rings might be
shifted, they probably have different x-values. Also the x-values are
continuous..meaning that once the ring starts.. there is an (x,y)
value until the end of the ring. When I was extracting the (x,y)
location of every pixel that makes up the rings I noticed that a lot
of x values had multiple y's. To fix this I used the MIN x value for
that group in order to be able to compute the area. I'm not sure if
that's bad or good.

I am using INT_TABULATED and TSUM currently and get very close values
for the most part. I compute the area under the curve for both rings
and then subtract but I'm not sure if that is correct either.

Thanks,

Frank

Subject: Re: Compute area between curves
Posted by Craig Markwardt on Mon, 13 Oct 2008 10:15:56 GMT
View Forum Message <> Reply to Message

frankosuna <frankosuna@gmail.com> writes:

> On Oct 12, 3:17ï¿½pm, James Kuyper <jameskuy...@verizon.net> wrote:
>> frankosuna wrote:
>>> Dear IDLers,
>>
>>> How can I compute the area between two curves given two functions?
>>
>> Are the curves closed? That is, do you create the complete curve by
>> drawing a line from the final <x,y> pair to the first <x,y> pair?
>>
>> Are the x values the same for the two curves? Are they evenly spaced?
>>
>> Note: you don't need to post the same question multiple times, this is a
>> newsgroup, not a chat room. Your message will stay up indefinitely. As a
>> general rule, you might have to wait 24 hours or more before getting an
>> answer.
>
> The curves are not closed... I posted some images of the actual rings
> I am trying
> to compare. They look like parabolas. The rings might differ in shift
> and slight rotation from each other. So because the rings might be

Sorry, you are not giving enough information. I.e. your problem is
not well defined enough. The "area" under the curves assumes we know
what "under" means. One definition could be under=Y, another under=X.
And I presume the best "under" might actually be some kind of radial
coordinate. Until you know which one you mean, it's difficult to
comment.

But assuming it's the radial version, i.e. centered on the planet,
then why not transform your X-Y curves to be R-PHI curves, with R and
PHI measured from the planet center. Then you could resample to a
uniform PHI grid, and compute the whatever difference you want,
straightforwardly.

Good luck,
Craig

--
-- --------------
Craig B. Markwardt, Ph.D. EMAIL: cbmarkwardt+usenet@gmail.com
-- --------------

Subject: Re: Compute area between curves
Posted by jameskuyper on Mon, 13 Oct 2008 13:24:27 GMT
View Forum Message <> Reply to Message

frankosuna wrote:
> On Oct 12, 3:17 pm, James Kuyper <jameskuy...@verizon.net> wrote:
>> frankosuna wrote:
>>> Dear IDLers,
>>> How can I compute the area between two curves given two functions?
>> Are the curves closed? That is, do you create the complete curve by
>> drawing a line from the final <x,y> pair to the first <x,y> pair?
>>
>> Are the x values the same for the two curves? Are they evenly spaced?
>>
>> Note: you don't need to post the same question multiple times, this is a
>> newsgroup, not a chat room. Your message will stay up indefinitely. As a
>> general rule, you might have to wait 24 hours or more before getting an
>> answer.
>
> The curves are not closed... I posted some images of the actual rings
> I am trying
> to compare. They look like parabolas.

Given the geometry involved, I think it's more likely that they are
sections of an ellipse, not parabolas.

 The rings might differ in shift
> and slight rotation from each other. So because the rings might be
> shifted, they probably have different x-values. Also the x-values are
> continuous..meaning that once the ring starts.. there is an (x,y)
> value until the end of the ring. When I was extracting the (x,y)
> location of every pixel that makes up the rings I noticed that a lot
> of x values had multiple y's. To fix this I used the MIN x value for
> that group in order to be able to compute the area. I'm not sure if
> that's bad or good.

I'm not sure I follow that. You describe those groups as having the same
 x value, so MIN(x) doesn't make sense. If you meant MIN(y), that would
make more sense, but taking the average makes more sense than using the
minimum.

> I am using INT_TABULATED and TSUM currently and get very close values
> for the most part. I compute the area under the curve for both rings
> and then subtract but I'm not sure if that is correct either.

That approach should produce reasonable results, if you use the average
y value rather than the minimum y value. However, this approach a
assumes that for any given value of x, a vertical line at x intersects
each curve only once. Since I suspect that these are actually sections

of an ellipse, it's entirely possible that some images will violate that
condition. If you had an image of the entire ellipse, any given x value
would cross 0, 1, or 2 times, depending upon the value of x. If you have
such an image, the simplest approach would be to break the images into
two sections, breaking them at the point where the tangent to the curve
is vertical. Calculate the area between the curves separately for each part.

A more general approach would would work regardless of the shapes of the
two curves. Just connect the two curves to create a single combined
curve that starts by listing all the points on one curve in clockwise
order, then continues by listing all of the points of the other curve in
counter-clockwise order. As a result, the combined curve encloses the
area that lies between the two curves. Then use POLY_AREA to calculates
the area enclosed by the combined curve. If you've got lots of data
points and a very thin area between the two curves, you'll almost
certainly need to do the calculations in double precision in order to
get useful results; otherwise the algorithm used will be dominated by
round-off errors. Note that you should connect the two curves in such a
way that the combined curve goes around the area between them in a
counter-clockwise direction. If you do it in the wrong order, you'll get
the negative of the correct area.

If you are taking this approach, there's no need to remove cases where
you have multiple y values for the same value of x. You'll get more
accurate results by keeping all of those points, rather than by
replacing them with the average value of y.

Subject: Re: Compute area between curves
Posted by Craig Markwardt on Tue, 14 Oct 2008 04:42:50 GMT

View Forum Message <> Reply to Message

James Kuyper <jameskuyper@verizon.net> writes:
> A more general approach would would work regardless of the shapes of
> the two curves. Just connect the two curves to create a single
> combined curve that starts by listing all the points on one curve in
> clockwise order, then continues by listing all of the points of the
> other curve in counter-clockwise order. As a result, the combined
> curve encloses the area that lies between the two curves. Then use
> POLY_AREA to calculates the area enclosed by the combined curve.
...

James, I had that thought as well, but I believe POLY_AREA will not
work as expected. When a polygon's edges self-intersect, then the
polygon is no longer "simple." In that case, the POLY_AREA method
will compute the *signed* total area. Polygonal segments where the
path traverses clockwise will contribute in a positive sense, and

counter-clockwise in the negative sense. The result will not be the
'total' area as we commonly expect, but some kind of non-intuitive
'net' area.

I still think the original questioner doesn't really know what he
needs yet.

Craig

--

Craig B. Markwardt, Ph.D.   EMAIL: cbmarkwardt+usenet@gmail.com
 ----

Subject: Re: Compute area between curves
Posted by mystea on Tue, 14 Oct 2008 05:35:35 GMT

View Forum Message <> Reply to Message

Hi everyone,

I am also working on a topic where I need to numerically calculate an
integral
of a tabulated function. However, what I need is an indefinite
integral, namely,
the area under a curve as a function of x-coordinate.

The procedure int_tabulated only calculates the definite integral,
given tabulated
f and its x-coordinates x. Let's say both f and x are double array of
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])
```

I thought it will work but not quite! Turns out that in general, the
result
integral will not be monotone even if f are always positive. I
believe this has something
to do with the algorithm of int_tabulated? How can I produce a
consistent calculation of
the indefinite integral?

Best Regards,
Gene

On Oct 13, 9:42 pm, Craig Markwardt
<craigm...@REMOVEcow.physics.wisc.edu> wrote:
> James Kuyper <jameskuy...@verizon.net> writes:
>> A more general approach would would work regardless of the shapes of
>> the two curves. Just connect the two curves to create a single
>> combined curve that starts by listing all the points on one curve in
>> clockwise order, then continues by listing all of the points of the
>> other curve in counter-clockwise order. As a result, the combined
>> curve encloses the area that lies between the two curves. Then use
>> POLY_AREA to calculates the area enclosed by the combined curve.
>
> ...
>
> James, I had that thought as well, but I believe POLY_AREA will not
> work as expected. When a polygon's edges self-intersect, then the
> polygon is no longer "simple." In that case, the POLY_AREA method
> will compute the *signed* total area. Polygonal segments where the
> path traverses clockwise will contribute in a positive sense, and
> counter-clockwise in the negative sense. The result will not be the
> 'total' area as we commonly expect, but some kind of non-intuitive
> 'net' area.
>
> I still think the original questioner doesn't really know what he
> needs yet.
>
> Craig
>
> --
> -- --------------
> Craig B. Markwardt, Ph.D. EMAIL: cbmarkwardt+use...@gmail.com
> -- --------------

Subject: Re: Compute area between curves
Posted by jameskuyper on Tue, 14 Oct 2008 10:16:58 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
> James Kuyper <jameskuyper@verizon.net> writes:
>> A more general approach would would work regardless of the shapes of
>> the two curves. Just connect the two curves to create a single
>> combined curve that starts by listing all the points on one curve in
>> clockwise order, then continues by listing all of the points of the
>> other curve in counter-clockwise order. As a result, the combined
>> curve encloses the area that lies between the two curves. Then use
>> POLY_AREA to calculates the area enclosed by the combined curve.
> ...
>

> James, I had that thought as well, but I believe POLY_AREA will not
> work as expected. When a polygon's edges self-intersect, then the
> polygon is no longer "simple."

As I understand it, the curves involved are sections of two
non-intersecting ellipses, with the smaller enclosed entirely in the
larger one. Connecting the curves as I suggest would create a simple
closed curve, with no intersections.

>   In that case, the POLY_AREA method
> will compute the *signed* total area. Polygonal segments where the
> path traverses clockwise will contribute in a positive sense, and
> counter-clockwise in the negative sense. The result will not be the
> 'total' area as we commonly expect, but some kind of non-intuitive
> 'net' area.

In a sense, a 'net' area is precisely what we want, and the fact that
this is the case seems quite intuitive to me. If the OP had two complete
ellipses, then as I understand it, what he wants is the area of the
larger ellipse minus the area of the smaller ellipse. If he were to
follow my suggestion with two full ellipses, that's precisely the
quantity that POLY_AREA should calculate.

Subject: Re: Compute area between curves
Posted by jameskuyper on Tue, 14 Oct 2008 11:26:49 GMT
View Forum Message <> Reply to Message

mystea wrote:
> Hi everyone,
>
> I am also working on a topic where I need to numerically calculate an
> integral
> of a tabulated function. However, what I need is an indefinite
> 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
> f and its x-coordinates x. Let's say both f and x are double array of
> 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.

> I thought it will work but not quite! Turns out that in general, the
> 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?

Subject: Re: Compute area between curves
Posted by Craig Markwardt on Tue, 14 Oct 2008 16:59:28 GMT
View Forum Message <> Reply to Message

James Kuyper <jameskuyper@verizon.net> writes:

> Craig Markwardt wrote:
>> James Kuyper <jameskuyper@verizon.net> writes:
>>> A more general approach would would work regardless of the shapes of
>>> the two curves. Just connect the two curves to create a single
>>> combined curve that starts by listing all the points on one curve in
>>> clockwise order, then continues by listing all of the points of the
>>> other curve in counter-clockwise order. As a result, the combined
>>> curve encloses the area that lies between the two curves. Then use
>>> POLY_AREA to calculates the area enclosed by the combined curve.
>> ...
>> James, I had that thought as well, but I believe POLY_AREA will not
>> work as expected. When a polygon's edges self-intersect, then the
>> polygon is no longer "simple."

>
> As I understand it, the curves involved are sections of two
> non-intersecting ellipses, with the smaller enclosed entirely in the
> larger one. Connecting the curves as I suggest would create a simple
> closed curve, with no intersections.

Assuming the poster knows what he wants to do, he said,
: I am trying to calculate how much of an error there is between two
: rings. I have two images each with a ring pictured in these two
: images.
[And then goes on to describe how the two traces are computed by
different methods.] In my mind, the two traces are measures of
essentially the *same* phenomenon, and he's trying to measure the
areal difference between these two different representations of the
same curve. I assumed this was some attempt to estimate the
uncertainty of some modeling method.

In fact, if you look at the image links the original poster provides,
the curves *are* intersecting. There is primarily a translation
offset, which causes them to intersect near the apex. So again, I'm
left with the quandry that either, (a) POLY_AREA isn't providing
what's needed, or (b) the poster needs to understand what he *really*
wants to do.

>> In that case, the POLY_AREA method
>> will compute the *signed* total area. Polygonal segments where the
>> path traverses clockwise will contribute in a positive sense, and
>> counter-clockwise in the negative sense. The result will not be the
>> 'total' area as we commonly expect, but some kind of non-intuitive
>> 'net' area.
>
> In a sense, a 'net' area is precisely what we want, and the fact that
> this is the case seems quite intuitive to me. If the OP had two
> complete ellipses, then as I understand it, what he wants is the area
> of the larger ellipse minus the area of the smaller ellipse. If he
> were to follow my suggestion with two full ellipses, that's precisely
> the quantity that POLY_AREA should calculate.

Yes, assuming they don't intersect (which they do).

Craig

--
-- --------------
Craig B. Markwardt, Ph.D. EMAIL: cbmarkwardt+usenet@gmail.com
-- --------------

Subject: Re: Compute area between curves
Posted by jameskuyper on Tue, 14 Oct 2008 17:21:15 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
> James Kuyper <jameskuyper@verizon.net> writes:
>
>
>> Craig Markwardt wrote:
>>> James Kuyper <jameskuyper@verizon.net> writes:
>>>> A more general approach would would work regardless of the shapes of
>>>> the two curves. Just connect the two curves to create a single
>>>> combined curve that starts by listing all the points on one curve in
>>>> clockwise order, then continues by listing all of the points of the
>>>> other curve in counter-clockwise order. As a result, the combined
>>>> curve encloses the area that lies between the two curves. Then use
>>>> POLY_AREA to calculates the area enclosed by the combined curve.
>>> ...
>>> James, I had that thought as well, but I believe POLY_AREA will not
>>> work as expected. When a polygon's edges self-intersect, then the
>>> polygon is no longer "simple."
>>
>> As I understand it, the curves involved are sections of two
>> non-intersecting ellipses, with the smaller enclosed entirely in the
>> larger one. Connecting the curves as I suggest would create a simple
>> closed curve, with no intersections.
>
> Assuming the poster knows what he wants to do, he said,
> : I am trying to calculate how much of an error there is between two
> : rings. I have two images each with a ring pictured in these two
> : images.
> [And then goes on to describe how the two traces are computed by
> different methods.] In my mind, the two traces are measures of
> essentially the *same* phenomenon, and he's trying to measure the
> areal difference between these two different representations of the
> same curve. I assumed this was some attempt to estimate the
> uncertainty of some modeling method.

I traced the message I was responding to back to the original message,
in which he said that he was looking for the area between the curves.
The message you're referring to was on a different branch of this
discussion, and I missed the implications of the text you cite. Now
that I've re-read it in light of what you've said, I agree with your
interpretation. I can think of two or three bad ways to measure the
error between the two curves, but I can't come up with any good ways.

> In fact, if you look at the image links the original poster provides,
> the curves *are* intersecting. There is primarily a translation
> offset, which causes them to intersect near the apex.

As displayed on my screen, I only saw one curve; perhaps I don't have
enough resolution to resolve the two curves clearly.

Subject: Re: Compute area between curves
Posted by mystea on Wed, 15 Oct 2008 05:35:09 GMT
View Forum Message <> Reply to Message

On Oct 14, 4:26 am, James Kuyper <jameskuy...@verizon.net> wrote:
> mystea wrote:
>> Hi everyone,
>
>> I am also working on a topic where I need to numerically calculate an
>> integral
>> of a tabulated function. However, what I need is an indefinite
>> 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
>> f and its x-coordinates x. Let's say both f and x are double array of
>> 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.
>
>> I thought it will work but not quite! Turns out that in general, the
>> 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?

I tried "tsum" and suddenly every problem was solved!

I found that the warning in the int_tabulated help file must be taken
very seriously:

Warning:
Data that is highly oscillatory requires a sufficient number of
samples for an accurate integral approximation.

My data was not oscillatory. However, I tried to find its first
derivative using "deriv" and
found that its first derivatives are oscillatory.

So the motto is: thou shalt not use int_tabulated when the result
from deriv is oscillatory. huh?

Subject: Re: Compute area between curves
Posted by Craig Markwardt on Wed, 15 Oct 2008 11:21:53 GMT
View Forum Message <> Reply to Message

jameskuyper@verizon.net writes:

> Craig Markwardt wrote:
>
>> In fact, if you look at the image links the original poster provides,
>> the curves *are* intersecting. There is primarily a translation
>> offset, which causes them to intersect near the apex.
>
> As displayed on my screen, I only saw one curve; perhaps I don't have
> enough resolution to resolve the two curves clearly.

I meant that, the single curves in each image intersect each other
when overlayed. I just did a simple image blink to see this.

Craig

--
 -- --------------

Craig B. Markwardt, Ph.D.   EMAIL: cbmarkwardt+usenet@gmail.com
-- --------------

Subject: Re: Compute area between curves
Posted by jameskuyper on Wed, 15 Oct 2008 11:44:35 GMT

View Forum Message <> Reply to Message

mystea wrote:
> On Oct 14, 4:26 am, James Kuyper <jameskuy...@verizon.net> wrote:
...
>> Could you give a simple example that demonstrates the problem you've seen?
>
> I tried "tsum" and suddenly every problem was solved!

I assume you're referring to
<http://idlastro.gsfc.nasa.gov/ftp/pro/math/tsum.pro>. tsum uses the
trapezoidal rule, which is the two-point Newton-Cotes formula;
INT_TABULATED uses the five-point formula. The higher-order formula
gives you more accurate results, so long as the data is tabulated at
sufficiently close intervals so that is it relatively smooth over any 5
consecutive data points. The lower order formula gives less accurate
results, but is more robust with respect to the errors that can be
created when the tabulated function isn't tabulated sufficiently
closely. This is the typical trade-off you get when comparing
higher-order numerical methods with lower-order ones.

> I found that the warning in the int_tabulated help file must be taken
> very seriously:
>
> Warning:
> Data that is highly oscillatory requires a sufficient number of
> samples for an accurate integral approximation.
>
> My data was not oscillatory. However, I tried to find its first
> derivative using "deriv" and
> found that its first derivatives are oscillatory.
>
> So the motto is: thou shalt not use int_tabulated when the result
> from deriv is oscillatory. huh?

I think that's the wrong conclusion. It's not that int_tabulated
shouldn't be used; it's that the data that was causing int_tabulated
problems shouldn't be used. If possible, you should get the data
tabulated more closely. If not, you should understand that any method
you use will give inaccurate results, though those results will be more
reasonable when you use a lower-order integration method rather than a
higher-order one.

I'd still like to see specific examples of data for which int_tabulated
has problems of this kind.

Subject: Re: Compute area between curves
Posted by jameskuyper on Wed, 15 Oct 2008 12:05:35 GMT
View Forum Message <> Reply to Message

James Kuyper wrote:
> mystea wrote:
>> On Oct 14, 4:26 am, James Kuyper <jameskuy...@verizon.net> wrote:
> ...
>>> Could you give a simple example that demonstrates the problem you've
>>> seen?
>>
>> I tried "tsum" and suddenly every problem was solved!
>
> I assume you're referring to
> <http://idlastro.gsfc.nasa.gov/ftp/pro/math/tsum.pro>. tsum uses the
> trapezoidal rule, which is the two-point Newton-Cotes formula;
> INT_TABULATED uses the five-point formula. The higher-order formula
> gives you more accurate results, so long as the data is tabulated at
> sufficiently close intervals so that is it relatively smooth over any 5
> consecutive data points. The lower order formula gives less accurate
> results, but is more robust with respect to the errors that can be
> created when the tabulated function isn't tabulated sufficiently
> closely. This is the typical trade-off you get when comparing
> higher-order numerical methods with lower-order ones.

I can be more specific about this. According to
<http://en.wikipedia.org/wiki/Newton%E2%80%93Cotes_formulas>, the error
terms are

    trapezoidal (two-point) rule: -(h^3/12) *f2(zeta)
    Boole's (five-point) rule: -(8*h^7/945)*f6(zeta)

where 'h' is the spacing of the data, f2(zeta) is the second derivative
of the function being integrated, evaluated at some unspecified location
in the range of integration, and f6(zeta) is the sixth derivative,
evaluated at some unspecified and probably different location in the
range of integration.

Therefore, if f2_max is the maximum value of the magnitude of the second
derivative over the range of integration, and f6_max is the same thing
for the sixth derivative, then it is better to use TSUM if

    h > (315*f2_max/(32*f6_max))^0.25

if h is less than that limiting value, you'll get better results with
INT_TABULATED.

Subject: Re: Compute area between curves
Posted by frankosuna on Wed, 15 Oct 2008 20:37:53 GMT
View Forum Message <> Reply to Message

Ok, I will try to explain what is going on.

I have a 3D wireframe model of Saturn and its rings. This wireframe
(which is superimposed over an image) is shifted and rotated
to match an image. The wireframe contains rings of Saturn and the
image contains the edges to where those rings belong. Scientists
tweak the wireframe so that it matches those edges, thus correcting
spacecraft pointing vectors. This is called C-Smithing or Camera
Smithing.

http://frankosuna.googlepages.com/wireframe2.jpg
This image is taken from the wireframe. I used region growing so that
when the user clicks on a ring in the wireframe, it selects that ring
(so that they can use
that ring to match it to an edge) and create this image.

http://frankosuna.googlepages.com/edgeDetect2.jpg
This image is created by utilizing a saturn image that contains many
features from saturn(like the edges of its rings). This in fact is an
edge to a ring from Saturn.
Region growing is used when the user selects a point in the image(this
should be the corresponding ring from the wireframe). Based on that
point where the user clicks, it region grows
that section and creates this image.

I have an algorithm that tries to match two images of the same scene
but different perspectives meaning that one image might be scaled,
rotated, or shifted differently than the other image.

The values taken from this algorithm are used to correct the wireframe
so that we can match it to the corresponding edge and therefor
correcting the spacecraft pointing vectors.

I need to come up with some measurement to see how close the edge
detected ring from the image and the ring from the wireframe are. They
are ellipses and they do cross since we're trying to get them as close
as possible.

I hope this clears things up.

Thanks for taking time to respond with such detailed answers,

Frank

Subject: Re: Compute area between curves
Posted by pgrigis on Wed, 15 Oct 2008 20:56:10 GMT
View Forum Message <> Reply to Message

Maybe you could determine the ellipse paramteres
(half-axis lengths, center position, rotation angle)
and work with that 5 parameters? Should be much simpler.

Ciao,
Paolo

frankosuna wrote:
> Ok, I will try to explain what is going on.
>
> I have a 3D wireframe model of Saturn and its rings. This wireframe
> (which is superimposed over an image) is shifted and rotated
> to match an image. The wireframe contains rings of Saturn and the
> image contains the edges to where those rings belong. Scientists
> tweak the wireframe so that it matches those edges, thus correcting
> spacecraft pointing vectors. This is called C-Smithing or Camera
> Smithing.
>
> http://frankosuna.googlepages.com/wireframe2.jpg
> This image is taken from the wireframe. I used region growing so that
> when the user clicks on a ring in the wireframe, it selects that ring
> (so that they can use
> that ring to match it to an edge) and create this image.
>
> http://frankosuna.googlepages.com/edgeDetect2.jpg
> This image is created by utilizing a saturn image that contains many
> features from saturn(like the edges of its rings). This in fact is an
> edge to a ring from Saturn.
> Region growing is used when the user selects a point in the image(this
> should be the corresponding ring from the wireframe). Based on that
> point where the user clicks, it region grows
> that section and creates this image.
>
> I have an algorithm that tries to match two images of the same scene
> but different perspectives meaning that one image might be scaled,
> rotated, or shifted differently than the other image.

>
> The values taken from this algorithm are used to correct the wireframe
> so that we can match it to the corresponding edge and therefor
> correcting the spacecraft pointing vectors.
>
> I need to come up with some measurement to see how close the edge
> detected ring from the image and the ring from the wireframe are. They
> are ellipses and they do cross since we're trying to get them as close
> as possible.
>
> I hope this clears things up.
>
> Thanks for taking time to respond with such detailed answers,
>
> Frank

Subject: Re: Compute area between curves
Posted by Craig Markwardt on Thu, 16 Oct 2008 03:42:38 GMT
View Forum Message <> Reply to Message

frankosuna <frankosuna@gmail.com> writes:
>
> I need to come up with some measurement to see how close the edge
> detected ring from the image and the ring from the wireframe are. They
> are ellipses and they do cross since we're trying to get them as close
> as possible.

OK, it still sounds like the "best" way to do it is take your points
and re-sample them to the same grid, so they are directly comparable.
In principle you can re-sample in X, but I think it would be better to
convert the original points to R,PHI polar coordinates and sample to a
uniform PHI grid. At that stage, you can compute

```
  TOTAL((R_MEAS - R_WIREFRAME)^2)
```

which is effectively a chi-square, which then also measures goodness
of your fit. [I.e. forget about integrals.]

Craig

--
 -- --------------
Craig B. Markwardt, Ph.D.   EMAIL: cbmarkwardt+usenet@gmail.com
 -- --------------

Subject: Re: Compute area between curves

Posted by Jeremy Bailin on Thu, 16 Oct 2008 12:18:42 GMT
View Forum Message <> Reply to Message

On Oct 15, 11:42 pm, Craig Markwardt
<craigm...@REMOVEcow.physics.wisc.edu> wrote:
> frankosuna <frankos...@gmail.com> writes:
>
>> I need to come up with some measurement to see how close the edge
>> detected ring from the image and the ring from the wireframe are. They
>> are ellipses and they do cross since we're trying to get them as close
>> as possible.
>
> OK, it still sounds like the "best" way to do it is take your points
> and re-sample them to the same grid, so they are directly comparable.
> In principle you can re-sample in X, but I think it would be better to
> convert the original points to R,PHI polar coordinates and sample to a
> uniform PHI grid. At that stage, you can compute
>   TOTAL((R_MEAS - R_WIREFRAME)^2)
> which is effectively a chi-square, which then also measures goodness
> of your fit.   [I.e. forget about integrals.]
>
> Craig
>
> --
> -- --------------
> Craig B. Markwardt, Ph.D.   EMAIL: cbmarkwardt+use...@gmail.com
> -- --------------

If the difference is just translational, I'd still just use an FFT to
cross-correlate the original images and find the peak...

-Jeremy.

Subject: Re: Compute area between curves
Posted by mystea on Thu, 16 Oct 2008 15:50:16 GMT
View Forum Message <> Reply to Message

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.

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

On Oct 14, 4:26 am, James Kuyper <jameskuy...@verizon.net> wrote:
> mystea wrote:
>> Hi everyone,
>
>> I am also working on a topic where I need to numerically calculate an
>> integral
>> of a tabulated function. However, what I need is an indefinite
>> 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
>> f and its x-coordinates x. Let's say both f and x are double array of
>> 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.
>
>> I thought it will work but not quite! Turns out that in general, the
>> 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?

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?

Subject: Re: Compute area between curves
Posted by jameskuyper on Fri, 17 Oct 2008 19:43:56 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)

For this function, the second derivative f2 = 6*x^(-4). The sixth
derivative f6 = 5040*x^-8. Within the range of integration, the
maximum values of f2 and d6 both occur at x =1, so f2_max = 6 and
f6_max=5040. Therefore, the maximum step size for which INT_TABULATED
gives better results than TSUM is

```
        (315*f2_max/(32*f6_max))^0.25 = 0.329
```

Your data is tabulated with a spacing of 1.0, which is more than 3
times too large to get better results with INT_TABULATED than with
TSUM. Try the following:

```
        x = 0.25*dindgen(800)+1.0
```

You get much nicer results that way.

> 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)
```

For the second portion of your data, the spacing is 61 units. The
maximum values of f2 and f6 occur in that region occur at x = 61:

```
        f2_max = 6*x[60]^(-4)
        f6_max = 5040*x[60]^(-8)
```

For this portion of the data, the minimum step size needed is

```
        (315*f2_max/(32*f6_max))^0.25 = 20.07
```

Once again, the spacing of your tabulated data is more than 3 times
too large for INT_TABULATED to give you better results than TSUM. The
moral of this story is that, if you can, you should sample your data
more closely. If you can't do that, you'll have to integrate it using
a more robust integration method, such as TSUM.

```
>  q2=dblarr(200)
>  for i=1,199 do q2[i]=int_tabulated(x2[0:i],y2[0:i])
>  plot,q2
>  plot,deriv(y2)
```

It would be more meaningful to use

```
        plot, x2, q2
        plot, x2, deriv(x2,y2)
```

However, give the range of values involved, you won't actually be able
to see anything on that second plot. For a more readable plot, use

```
        plot, x2, -deriv(x2,y2), /xlog,/ylog
```

For comparison, the analytical result is:

```
        oplot, x2, 2*x2^(-3)
```

Which seems to be a pretty accurate match.

...
> 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.

No, the issue is that the best-fit polynomial for under-sampled data isn't a particularly good fit. That polynomial has wiggles that don't match the shape of the function actually being tabulated.

> 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.

I didn't see any oscillations in deriv(y), or deriv(x2,y2). Could you explain that more precisely?