Subject: Re: Slow object graphics when plotting multiple lines
Posted by lecacheux.alain on Mon, 04 Apr 2016 17:15:09 GMT
View Forum Message <> Reply to Message

Le lundi 4 avril 2016 18:51:01 UTC+2, Yngvar Larsen a écrit :
> On Monday, 4 April 2016 17:54:03 UTC+2, alx wrote:
>
>>> Other, similar, tools don't have these issues. (Yes, I mean matlab).
>
>> As far as I understand, the function PLOT(/OVERPLOT) was NOT designed as being a simple overplotting routine. When you add a new curve, the entire plot is actually modified (ranges, axes, etc...) at the expense of some slowness. I guess that it is a feature from Exelis.
>
> Yes, this "feature" makes this approach O(N^2), which scales terribly beyond a couple of hundred lines. I don't see why the view needs to be recalculated for each new object, at least not by default.
>
> Though this use is not really what you should do in IDL anyway since the overhead created by the loop itself makes even direct graphics slow when then number of lines becomes large (> 10000 or so).
>
>> If you want to do a "simple" overplotting (ala OPLOT), an efficient way is to combine POSITION, CURRENT and [XYZ]RANGE keywords, instead of using OVERPLOT.
>
> How? The documentation of PLOT() indicates something else (unless I misunderstood you):
>
> **********
> OVERPLOT
> Set this keyword to 1 (one) to place the graphic on top of the currently-selected graphic within the current window. The two graphics items will then share the same set of axes. If no current window exists, then this keyword is ignored and a new window is created.
> [...]
> Tip: If you want your graphic to have a new set of axes, you should use the CURRENT keyword instead.
>
> CURRENT
> Set this keyword to create the graphic in the current window with a new set of axes. If no window exists, a new window is created.
> [...]
> Tip: If you want your graphic to share the same axes as an existing graphic, you should use the OVERPLOT keyword instead.
> **********
>
> I think it should be possible to do something like
>
> p = PLOT(randn(seed,num_ points, num_lines))
>
> and get what you want without jumping through the hoops involved in constructing the

CONNECTIVITY matrix or adding fake missing data (NaN) to make it work. I don't see why not. PLOT() currently takes a vector as input, and if you try to apply it to a 2D array, it will flatten to a 1D array first. The latter property is _not_ documented, so the Harris people could easily add this functionality. It should be <10 lines of code. It would make the life easier for simple cases like this where all the curves have the same number of points.
>
>> The POLYLINE trick, when many curves are needed, looks like to me somewhat faster than Matlab or Python equivalents.
>
> Indeed. And, as I showed, nearly as fast as direct graphics for large number of lines. Which is quite impressive!
>
> POLYLINE is of course still very useful when plotting lots of curves with different number of points, e.g. a map.
>
> --
> Yngvar

> How? The documentation of PLOT() indicates something else (unless I misunderstood you):
IDL> pl = plot(/TEST)
IDL> pl1 = plot([50,100], [-0.5,0.5], COLOR='red', /CURRENT, POSITION=pl.POSITION, XRANGE=pl.XRANGE, YRANGE=pl.YRANGE)
