Subject: Re: IDL hilbert() function
Posted by ed.schmahl on Mon, 03 Nov 2008 18:26:59 GMT
View Forum Message <> Reply to Message

On Nov 3, 2:16 am, Wox <s...@nomail.com> wrote:
> On Sat, 1 Nov 2008 02:14:54 -0700 (PDT), lecacheux.al...@wanadoo.fr
> wrote:
>
>> Is this function actually computing the Hilbert transform ?
>> The Hilbert transform is known to be idempotent, i.e. H(H(x)) = -x.
>> However, by applying the IDL function, one get for instance :
>> IDL> print, hilbert (hilbert (indgen(8)))
>> (6.00000, 0.000000)(7.00000, 0.000000)
>> (4.00000, 0.000000)(5.00000, 0.000000)
>> (2.00000, 0.000000)(3.00000, 0.000000)
>> (0.000000, 0.000000)(1.00000, 0.000000)
>
> It works for this:
> x=findgen(180)/90.*!pi
> plot,x,hilbert(hilbert(sin(x))),/xs
> oplot,x,-sin(x),psym=1
>
> Not for this (flips):
> plot,hilbert(hilbert((indgen(1000))))
> oplot,indgen(1000),psym=1
>
> I'm not sure why, but check the hilbert.pro in the IDL-lib directory
> to see how it's implemented.

Hello Hilbert fans,

Hilbert.pro is not a perfect implementation of the Hilbert function.
One of its failures is an inability to treat odd numbered arrays
properly. Try this:

```
 x1=findgen(9)-4      ; Evenly distributed around the origin
 y1=abs(x1)           ; An even function of x1
 h1=hilbert(y1)       ; This should be an ODD function of x1, i.e. H(-
x)=-H(x)
 plot,x1,y1
 oplot,x1,h1,psym=-6  ; But it's clearly not exactly anti-symmetric
about the origin!
```

The problem goes away if you use 10 values instead of 9:

```
 x2=findgen(10)-4.5   ; Again evenly distributed around the origin
 y2=abs(x2)           ; An even function of x2
 h2=hilbert(y2)       ; This should be an ODD function, i.e., H(-x)=-
```

H(x)
 plot,x2,y2
 oplot,x2,h2,psym=-6 ; ... and it is: h2(x2)=-h2(-x2) plus a constant

Looking at the source code suggests that an easy fix is possible.

Ed Schmahl
