Subject: Re: IDL hilbert() function
Posted by lecacheux.alain on Tue, 04 Nov 2008 13:45:56 GMT
View Forum Message <> Reply to Message

On 3 nov, 20:54, "R.G. Stockwell" <notha...@noemail.com> wrote:
> <ed.schm...@gmail.com> wrote in message
>
> news:f43242be-734b-4098-8bc1-3ab0691270d9@c22g2000prc.google groups.com...
> 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
>  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
>
>  A very long time ago, I modified the code (it is below), check it out.
>  I did run into a number of problems with hilbert in both IDL and MATLAB.
>  Matlab was just flat out wrong (it expanded the time series to be a power of
>  2,
>  but did not expand the convolving kernel properly).  After many arguements
>  in email
>  with a matlab developer, they finally acquiesced (but simply making the
>  wrong behaviour
>  a keyword option (rolls eyes)).
>
>  Cheers,
>  bob
>
>  Here is the code:
>
>  FUNCTION HILBERT,X,D, ANALYTIC = a   ; performs the Hilbert transform of
>  some data.
>         ON_ERROR,2           ; Return to caller if an error occurs
>         Y=FFT(X,-1)          ; go to freq. domain.
>         N=N_ELEMENTS(Y)
>
>         I=COMPLEX(0.0,-1.0)
>
>         IF N_PARAMS(X) EQ 2 THEN I=I*D
>         N2=ceil(N/2.)-1              ; effect of odd and even # of elements
>                              ; considered here.
>         y(0) = complex(0,0)  ; zero the DC value (required for hilbert
>  trans.)
>         Y(1)=Y(1:N2)*I       ; multiplying by I rotates counter c.w. 90 deg.
>       if (n mod 2) eq 0  then Y(N2+1) = complex(0,0)     ; don't need this
```

```
>         N2=N-N2
>    Y(N2)=Y(N2:N-1)/I
>         Y=float(FFT(Y,1))  ; go back to time domain
>         if keyword_set(a) then y = complex(x,y)
>   RETURN,y
>
>  END- Masquer le texte des messages précédents -
>
>  - Afficher le texte des messages précédents -
```

Thank you all for answering: I was a little bit confused in finding
both IDL (6.4)
and Matlab (5.4) in error. But you confirmed that !
Thanks for the code. Using IDL, a simpler one might be :

```
function Hilbert, x, DOUBLE=double
  n = N_elements (x)
  m = ((n mod 2) eq 0) ? [0, dblarr(n/2-1) + 1, 0, dblarr(n/2-1) -
1] : [0, dblarr((n-1)/2) + 1, dblarr((n-1)/2) - 1]
  return, fft (fft (x, -1, DOUBLE=double)*m*complex(0,1), 1,
DOUBLE=double)
  end

function Analytic, x, DOUBLE=double
  n = N_elements (x)
  m = ((n mod 2) eq 0) ? [1, dblarr(n/2-1) + 2, 1, dblarr(n/2-1)] :
[1, dblarr((n-1)/2) + 2, dblarr((n-1)/2)]
  return, fft (fft (x, -1, DOUBLE=double)*m, 1, DOUBLE=double)
  end
```

alx.