Subject: Re: Hankel (Fourier-Bessel) Transform
Posted by on Thu, 27 Mar 2014 15:12:11 GMT
View Forum Message <> Reply to Message

Den onsdagen den 26:e mars 2014 kl. 17:25:06 UTC+1 skrev Mats Löfdahl:
> Den tisdagen den 24:e juli 2001 kl. 21:03:48 UTC+2 skrev William Thompson:
>
>> Georg.Pabst@nrc.ca (Georg Pabst) writes:
>
>>
>
>>> Hi,
>
>>
>
>>> I'm looking for the Hankel (Fourier-Bessel) Transform, i.e.,
>
>>> int(0,infinity) f(t)*BesselJ(t*r)t dt being implemented in IDL.
>
>>
>
>>> There is a paper "Siegman A. 1980. Quasi fast Hankel transform. Opt.
>
>>> Lett. 1, 13-15" and one can also find the code in Fortran or C...
>
>>
>
>>> Thanks,
>
>>> Georg
>
>>
>
>> Here's an old program that I think might be what you need.
>
>>
>
>> Bill Thompson
>
>>
>
>>
>
>>
>
>>
>
>> FUNCTION HANKEL,F
>
>> ;

```
>
>>  ;  This function returns the Hankel transform of the argument.
>
>>  ;
>
>>   S = SIZE(F)
>
>>   IF S(0) NE 1 THEN BEGIN
>
>>    PRINT,'*** Variable must be a one-dimensional array, name= F, routine HANKEL.'
>
>>    RETURN,F
>
>>   ENDIF
>
>>  ;
>
>>   X = INDGEN(F)
>
>>   K = ( 2. * !PI / FLOAT(N_ELEMENTS(X)) ) * X
>
>>   SC = 0.*X + 1.
>
>>   IF N_ELEMENTS(SC) GT 3 THEN BEGIN
>
>>    SC(0) =  3.D0 /  8.D0
>
>>    SC(1) =  7.D0 /  6.D0
>
>>    SC(2) = 23.D0 / 24.D0
>
>>   ENDIF
>
>>  ;
>
>>   H = BES0( K # X )  #  ( K * F * SC )
>
>>  ;
>
>>   RETURN,H
>
>>   END
>
>
>
>  So this thread is from 2001 but it is all I found by googling for "Hankel transform IDL"...
```

I've spent some time trying to understand various matlab codes for calculating Hankel transforms

but they seem to require the function to be sampled at some exponential sampling. I need a code that you can call like the fft(), with equidistant sampling.

The OP was happy with porting a matlab code found at www.nmt.edu/~borchers/hankel.html, but that is a code that requires the name of a function that it then evaluates at points of its own choosing. It does not solve my problem.

> I tried to use the HANKEL function given above but I couldn't make it work. The X=INDGEN(F) makes no sense to me. Should it be X=INDGEN(n_elements(F))?

To be specific, X=INDGEN(F), where F is the input function. How is that supposed to work? IDL will accept a vector of dimension lengths, but that vector must be shorter than 8 elements and should anyway consist of integers > 0. That seems like rather limiting requirements.

> I tried that and as I can't find the BES0 function, I substituted beselj(K # X, 0). This gave me some output, but it completely failed my test of using the same function to compute the inverse Hankel transform and thus getting the original function back. So maybe my changes were all wrong.

Here is one test that demonstrates that the transform of the transform does not return the original function (not even times some constant - that would be OK of course):

```
r = findgen(101)/100.
f = r^3
h = hankel(f)
hh = hankel(h)

cgplot, r, hh, color = 'red'
cgplot, r, f, /over, color = 'blue'
```

But that is with my edits. Here is the edited function:

```
FUNCTION HANKEL,F
;
;  This function returns the Hankel transform of the argument.
;
  S = SIZE(F)
  IF S(0) NE 1 THEN BEGIN
     PRINT,'*** Variable must be a one-dimensional array, name= F, routine HANKEL.'
     RETURN,F
  ENDIF
;
 ;X = INDGEN(F)
  X = INDGEN(n_elements(F))
  K = ( 2. * !PI / FLOAT(N_ELEMENTS(X)) ) * X
  SC = 0.*X + 1.
  IF N_ELEMENTS(SC) GT 3 THEN BEGIN
```

```
    SC(0) =  3.D0 /  8.D0
    SC(1) =  7.D0 /  6.D0
    SC(2) = 23.D0 / 24.D0
  ENDIF
;
;  H = BES0( K # X )  #  ( K * F * SC )
  H = beselj( K # X, 0)  #  ( K * F * SC )
;
  RETURN,H
END
```

> So, does anyone know of a useful implementation for IDL? Or C?

Nothing? How about FORTRAN?

> In a response to the post above, Craig commented that it can be done with the discrete Fourier transform. That sounds easy. Is it?

One of the matlab functions I found does this. But it won't let me choose my own sampling points.
