Subject: Problem discovered in bandpass_filter.pro
Posted by kagoldberg on Wed, 11 Nov 2015 18:58:54 GMT
View Forum Message <> Reply to Message

I just alerted Exelisvis to an error with BANDPASS_FILTER() on IDL 8.4.
I found that for 2D array, the high-frequency cutoff changes by sqrt(2) when the low frequency argument changes from 0 to 0.000001. The program uses different expressions to calculate the filter, based on the lowFreq argument.

Consider the following 2 cases.

**CASE 1
a = randomu(seed, 1000,1000) - 0.5
b = bandpass_filter(a, 0., 0.1, /ideal) ;--- lowFreq is zero
c = abs(fft(b))
window
tvscl, c

**CASE 2
a = randomu(seed, 1000,1000) - 0.5
b = bandpass_filter(a, 0.000001, 0.1, /ideal) ;--- changed lowFreq to something very small
c = abs(fft(b))
window
tvscl, c

Notice that the different lowFreq value here changes the HIGH frequency cutoff
in the output by sqrt(2) because there is an error in the way the function is coded.

In fact, the behavior of the function with lowFrequency NE 0 is incorrect
and leads to cutoff frequencies that are sqrt(2) smaller than they should be.

Say you have a 1000 pixel array, and you set
b = bandpass_filter(a, 0., 0.1, /ideal)

Here, we expect the high frequency cutoff to occur at 0.1 * 1000 = 100 cycles.
Instead, a quick test will show that the cutoff occurs at 70 cycles ~ 100/sqrt(2).
This occurs with /butterworth and /ideal, maybe /gaussian but I didn't test it.

I discovered it in the difference that occurs with filtered an array using a BUTTERWORTH() and 2 FFTs, versus just using BANDPASS_FILTER(... BUTTERWORTH=N)

Subject: Re: Problem discovered in bandpass_filter.pro
Posted by chris_torrence@NOSPAM on Fri, 13 Nov 2015 18:25:23 GMT
View Forum Message <> Reply to Message

On Wednesday, November 11, 2015 at 11:58:58 AM UTC-7, kagol...@lbl.gov wrote:
> I just alerted Exelisvis to an error with BANDPASS_FILTER() on IDL 8.4.

> I found that for 2D array, the high-frequency cutoff changes by sqrt(2) when the low frequency argument changes from 0 to 0.000001. The program uses different expressions to calculate the filter, based on the lowFreq argument.
>
> Consider the following 2 cases.
>
> **CASE 1
> a = randomu(seed, 1000,1000) - 0.5
> b = bandpass_filter(a, 0., 0.1, /ideal) ;--- lowFreq is zero
> c = abs(fft(b))
> window
> tvscl, c
>
> **CASE 2
> a = randomu(seed, 1000,1000) - 0.5
> b = bandpass_filter(a, 0.000001, 0.1, /ideal) ;--- changed lowFreq to something very small
> c = abs(fft(b))
> window
> tvscl, c
>
> Notice that the different lowFreq value here changes the HIGH frequency cutoff
> in the output by sqrt(2) because there is an error in the way the function is coded.
>
> In fact, the behavior of the function with lowFrequency NE 0 is incorrect
> and leads to cutoff frequencies that are sqrt(2) smaller than they should be.
>
> Say you have a 1000 pixel array, and you set
> b = bandpass_filter(a, 0., 0.1, /ideal)
>
> Here, we expect the high frequency cutoff to occur at 0.1 * 1000 = 100 cycles.
> Instead, a quick test will show that the cutoff occurs at 70 cycles ~ 100/sqrt(2).
> This occurs with /butterworth and /ideal, maybe /gaussian but I didn't test it.
>
> I discovered it in the difference that occurs with filtered an array using a BUTTERWORTH() and 2 FFTs, versus just using BANDPASS_FILTER(... BUTTERWORTH=N)

Hi,
Thanks for reporting this! I think the "Ideal" filter is actually okay, but there is something fishy with the Butterworth. Here's a different reproduce case:

```
; Ideal
a = randomu(seed, 1000,1000) - 0.5
b1 = bandpass_filter(a, 0., 0.4, /ideal)
c1 = abs(fft(b1)/fft(a))
b2 = bandpass_filter(a, 0.000001, 0.4, /ideal)
c2 = abs(fft(b2)/fft(a))

p = plot(c1[*,0], '2', dim=[800,800], yrange=[0,1.1], $
```

```
  layout=[1,2,1], title='Ideal')
p = plot(c2[*,0], /overplot, color='red')

; Butterworth
b1 = bandpass_filter(a, 0., 0.4, butterworth=20)
c1 = abs(fft(b1)/fft(a))
b2 = bandpass_filter(a, 0.000001, 0.4, butterworth=20)
c2 = abs(fft(b2)/fft(a))

p = plot(c1[*,0], '2', layout=[1,2,2], yrange=[0,1.1], $
  /current, title='Butterworth')
p = plot(c2[*,0], /overplot, color='red')
```

Notice that for the Ideal filter there is no significant difference between using lowFreq=0 and lowFreq=0.000001, but for Butterworth there is definitely a discontinuity. As an aside, for the Ideal case I believe that the sqrt(2) just comes from the fact that it is a "circle".

I'll take a look at the code and see what's up.
Thanks again,
Chris

Subject: Re: Problem discovered in bandpass_filter.pro
Posted by chris_torrence@NOSPAM on Tue, 01 Dec 2015 19:16:48 GMT
View Forum Message <> Reply to Message

Hi,

This turned out to not be a bug. Inside bandpass_filter (and bandreject_filter) we are using three different formulas: one for low-pass, one for high, and one for bandpass. For example, for the lowpass filter, you want the filter to include all of the frequencies near zero, and then gradually fall off as you get to the high-frequency cutoff. However, for the bandpass filter, *regardless* of the low-pass frequency (even if it is very close to 0), you want the response function to gradually increase from 0 to 1, and then back down to 0.

These formulas can all be found within:
Gonzalez, R.C., and R.E. Woods. Digital Image Processing, 3rd edition. Pearson Education, Upper Saddle River, New Jersey, 2008.

I've gone ahead and updated the IDL documentation to include the correct formulas, as well as some better graphs and examples.

Thanks for reporting this!

-Chris
