Subject: highpass spatial filtering (lon,lat,time)
Posted by Teddy Allen on Sat, 12 Aug 2017 03:49:46 GMT

View Forum Message <> Reply to Message

Gulp, after wrestling with this task for the entire day, I am reduced to admitting my IDL programming deficiencies and am thus seeking help / advice / encouragement / comfort / healing from other more knowledgeable IDL users (which is basically global population - 1).

My goal is to compute and plot the 1-10 day frequency (highpass) filtered variance of 250hPa geopotential height (z). I have a time series of 9028 days across the Caribbean at 0.5 deg spatial resolution ([lon,lat,time], z =
[201,101,9028]).

I know how to compute the highpass filtered time series (inverse FFT) for one grid cell (code below):

```
PRO FOURIER_FILTER_250Z, type
n   = 9028        ;Number of samples in time signal

x = INDGEN(9028)  ;Compute independent coordinate

IF (N_ELEMENTS(type) EQ 0) THEN TYPE = 'lowpass'     ;Default filter type
k = [LINDGEN(n/2 + 1), REVERSE(-(1 + LINDGEN(n/2 - 1)))] ;compute wavenumbers


filter =  FLTARR(n)                                                    ;Define filter array
CASE STRUPCASE(type) OF
'LOWPASS'  : q = WHERE(ABS(k) LT  902, count)  ;Find low frequences

'HIGHPASS' : q = WHERE(ABS(k) GT 902, count)   ;Find high frequences

'BANDPASS' : q = WHERE((ABS(k) GT 902) AND (ABS(k) LT 9028), count) ;bandpass
frequencies
   ELSE    : MESSAGE, 'Filter type must be specified.'  ;Default function
ENDCASE

;Create filter
IF (count gt 0) THEN filter[q] = 1.0
;Compute Fourier transform (z is a 1-D 9028 array of 250hPa geo ht)
g  = FFT(z)
;Filter the signal
gg = filter*g
;Compute the inverse FFT, which is the time filtered time series at one grid

ggg = FFT(gg, /INVERSE)
```

ggg is my highpass filtered time series at one grid cell from which I can now compute the variance. But, I get into all sorts of trouble when I try to loop the above through lon and lat to calculate the highpass time series at each longitude and latitude grid cell. Does anybody have experience in using IDL to compute highpass filtering across a spatial domain over many days (as in 30 years worth)? Basically, I want to compute ggg for all grid cells across my domain. Sounds simple, right? What am I missing? The below spirals out of control and is filled with errors.

```
g = dcomplexarr(201,101,9028)
gg = dcomplexarr(201,101,9028)
ggg = dcomplexarr(201,101,9028)
IF (count gt 0) THEN filter[q] = 1.0     ;Create filter
   for i = 0,200 do begin
     for j = 0,100 do begin
      for k = 0,9027 do begin
          g[i,j,k]  = FFT(z)             ;Compute Fourier transform
         gg[i,j,k] = filter*g[i,j,k]     ;Filter the signal
         ggg[i,j,k] = FFT(gg, /INVERSE)
      endfor
     endfor
   endfor
```

Fingers (and toes) crossed.
Thank you,
teddy

Subject: Re: highpass spatial filtering (lon,lat,time)
Posted by Helder Marchetto on Mon, 14 Aug 2017 06:32:47 GMT

View Forum Message <> Reply to Message

On Saturday, August 12, 2017 at 5:49:49 AM UTC+2, IDL filter novice wrote:
> Gulp, after wrestling with this task for the entire day, I am reduced to admitting my IDL programming deficiencies and am thus seeking help / advice / encouragement / comfort / healing from other more knowledgeable IDL users (which is basically global population - 1).
>
> My goal is to compute and plot the 1-10 day frequency (highpass) filtered variance of 250hPa geopotential height (z). I have a time series of 9028 days across the Caribbean at 0.5 deg spatial resolution ([lon,lat,time], z =
> [201,101,9028]).
>
> I know how to compute the highpass filtered time series (inverse FFT) for one grid cell (code below):
>

```
> PRO FOURIER_FILTER_250Z, type
> n   = 9028       ;Number of samples in time signal

> x = INDGEN(9028)  ;Compute independent coordinate
>  IF (N_ELEMENTS(type) EQ 0) THEN TYPE = 'lowpass'     ;Default filter type

>  k = [LINDGEN(n/2 + 1), REVERSE(-(1 + LINDGEN(n/2 - 1)))] ;compute wavenumbers

>
>  filter =  FLTARR(n)                                                    ;Define filter array
>  CASE STRUPCASE(type) OF
>  'LOWPASS'  : q = WHERE(ABS(k) LT  902, count)  ;Find low frequences

>  'HIGHPASS' : q = WHERE(ABS(k) GT 902, count)   ;Find high frequences

>  'BANDPASS' : q = WHERE((ABS(k) GT 902) AND (ABS(k) LT 9028), count) ;bandpass
frequencies
>    ELSE    : MESSAGE, 'Filter type must be specified.'  ;Default function
>  ENDCASE
>
>  ;Create filter
>  IF (count gt 0) THEN filter[q] = 1.0
>  ;Compute Fourier transform (z is a 1-D 9028 array of 250hPa geo ht)
>  g  = FFT(z)
>  ;Filter the signal
>  gg = filter*g
>  ;Compute the inverse FFT, which is the time filtered time series at one grid

>  ggg = FFT(gg, /INVERSE)
>
>
```

> ggg is my highpass filtered time series at one grid cell from which I can now compute the variance. But, I get into all sorts of trouble when I try to loop the above through lon and lat to calculate the highpass time series at each longitude and latitude grid cell. Does anybody have experience in using IDL to compute highpass filtering across a spatial domain over many days (as in 30 years worth)? Basically, I want to compute ggg for all grid cells across my domain. Sounds simple, right? What am I missing? The below spirals out of control and is filled with errors.

```
>
>  g = dcomplexarr(201,101,9028)
>  gg = dcomplexarr(201,101,9028)
>  ggg = dcomplexarr(201,101,9028)
>  IF (count gt 0) THEN filter[q] = 1.0     ;Create filter
>    for i = 0,200 do begin
>      for j = 0,100 do begin
>       for k = 0,9027 do begin
>           g[i,j,k]  = FFT(z)            ;Compute Fourier transform
>          gg[i,j,k] = filter*g[i,j,k]     ;Filter the signal
>          ggg[i,j,k] = FFT(gg, /INVERSE)
>       endfor
>      endfor
>    endfor
>
```

>
> Fingers (and toes) crossed.
> Thank you,
> teddy

Hi,
have you looked/tied the dimension keyword of FFT:

http://www.harrisgeospatial.com/docs/FFT.html#F_848155245_37 862

This should do the trick. Try setting dimension=3

```
g = fft(z, dimension=3)
gg = filter*g
ggg = fft(gg,/inverse,dimension=3)
```

I have not tested this, but should do the work...

Cheers,
Helder

Subject: Re: highpass spatial filtering (lon,lat,time)
Posted by Markus Schmassmann on Mon, 14 Aug 2017 09:29:47 GMT
View Forum Message <> Reply to Message

On 08/14/2017 08:32 AM, Helder wrote:
> On Saturday, August 12, 2017 at 5:49:49 AM UTC+2, IDL filter novice wrote:
>> Gulp, after wrestling with this task for the entire day, I am reduced to admitting my IDL programming deficiencies and am thus seeking help / advice / encouragement / comfort / healing from other more knowledgeable IDL users (which is basically global population - 1).
>>
>> My goal is to compute and plot the 1-10 day frequency (highpass) filtered variance of 250hPa geopotential height (z). I have a time series of 9028 days across the Caribbean at 0.5 deg spatial resolution ([lon,lat,time], z =
>>   [201,101,9028]).
>>
>> I know how to compute the highpass filtered time series (inverse FFT) for one grid cell (code below):
>>

```
>> PRO FOURIER_FILTER_250Z, type
>> n   = 9028        ;Number of samples in time signal
>> x = INDGEN(9028)  ;Compute independent coordinate
>> IF (N_ELEMENTS(type) EQ 0) THEN TYPE = 'lowpass'     ;Default filter type
>> k = [LINDGEN(n/2 + 1), REVERSE(-(1 + LINDGEN(n/2 - 1)))] ;compute wavenumbers
>>
>> filter =  FLTARR(n)                                                  ;Define filter array
>> CASE STRUPCASE(type) OF
>>  'LOWPASS'  : q = WHERE(ABS(k) LT  902, count)  ;Find low frequences
>>  'HIGHPASS' : q = WHERE(ABS(k) GT 902, count)   ;Find high frequences
>>  'BANDPASS' : q = WHERE((ABS(k) GT 902) AND (ABS(k) LT 9028), count) ;bandpass frequencies
>>      ELSE    : MESSAGE, 'Filter type must be specified.'  ;Default function
>>  ENDCASE
>>
>>  ;Create filter
>>  IF (count gt 0) THEN filter[q] = 1.0
>>  ;Compute Fourier transform (z is a 1-D 9028 array of 250hPa geo ht)
>>  g  = FFT(z)
>>  ;Filter the signal
>>  gg = filter*g
>>  ;Compute the inverse FFT, which is the time filtered time series at one grid
>>  ggg = FFT(gg, /INVERSE)
>>
>>
```

>> ggg is my highpass filtered time series at one grid cell from which I can now compute the variance. But, I get into all sorts of trouble when I try to loop the above through lon and lat to calculate the highpass time series at each longitude and latitude grid cell. Does anybody have experience in using IDL to compute highpass filtering across a spatial domain over many days (as in 30 years worth)? Basically, I want to compute ggg for all grid cells across my domain. Sounds simple, right? What am I missing? The below spirals out of control and is filled with errors.

```
>>
>>  g = dcomplexarr(201,101,9028)
>>  gg = dcomplexarr(201,101,9028)
>>  ggg = dcomplexarr(201,101,9028)
>>  IF (count gt 0) THEN filter[q] = 1.0     ;Create filter
>>      for i = 0,200 do begin
>>        for j = 0,100 do begin
>>         for k = 0,9027 do begin
>>             g[i,j,k]  = FFT(z)            ;Compute Fourier transform
>>            gg[i,j,k] = filter*g[i,j,k]     ;Filter the signal
>>            ggg[i,j,k] = FFT(gg, /INVERSE)
>>         endfor
>>        endfor
>>      endfor
>>
>>
```

>> Fingers (and toes) crossed.
>> Thank you,
>> teddy
>
> Hi,
> have you looked/tied the dimension keyword of FFT:
>
>   http://www.harrisgeospatial.com/docs/FFT.html#F_848155245_37 862
>

```
>  This should do the trick. Try setting dimension=3
>
>  g = fft(z, dimension=3)
>  gg = filter*g
>  ggg = fft(gg,/inverse,dimension=3)
>
>  I have not tested this, but should do the work...
>
>  Cheers,
>  Helder
>
```

whereby

```
filter_1d=dblarr(1,1,n)
filter_1d[q]=1.
filter=rebin(filter_1d,[201,101,n],/sample)
```

Subject: Re: highpass spatial filtering (lon,lat,time)
Posted by Dick Jackson on Thu, 17 Aug 2017 17:14:13 GMT

View Forum Message <> Reply to Message

On Monday, 14 August 2017 02:29:50 UTC-7, Markus Schmassmann wrote:

```
>  On 08/14/2017 08:32 AM, Helder wrote:
>>  On Saturday, August 12, 2017 at 5:49:49 AM UTC+2, IDL filter novice wrote:
>>>  Gulp, after wrestling with this task for the entire day, I am reduced to admitting my IDL programming deficiencies and am thus seeking help / advice / encouragement / comfort / healing from other more knowledgeable IDL users (which is basically global population - 1).
>>>
>>>  My goal is to compute and plot the 1-10 day frequency (highpass) filtered variance of 250hPa geopotential height (z). I have a time series of 9028 days across the Caribbean at 0.5 deg spatial resolution ([lon,lat,time], z =
>>>    [201,101,9028]).
>>>
>>>  I know how to compute the highpass filtered time series (inverse FFT) for one grid cell (code below):
>>>
>>>  PRO FOURIER_FILTER_250Z, type
>>>  n   = 9028        ;Number of samples in time signal
>>>  x = INDGEN(9028)  ;Compute independent coordinate
>>>  IF (N_ELEMENTS(type) EQ 0) THEN TYPE = 'lowpass'     ;Default filter type
>>>  k = [LINDGEN(n/2 + 1), REVERSE(-(1 + LINDGEN(n/2 - 1)))] ;compute wavenumbers
>>>
>>>  filter =  FLTARR(n)                                                        ;Define filter array
>>>  CASE STRUPCASE(type) OF
>>>  'LOWPASS'  : q = WHERE(ABS(k) LT  902, count)  ;Find low frequences
>>>  'HIGHPASS' : q = WHERE(ABS(k) GT 902, count)   ;Find high frequences
>>>  'BANDPASS' : q = WHERE((ABS(k) GT 902) AND (ABS(k) LT 9028), count) ;bandpass
```

```
frequencies
>>>     ELSE    : MESSAGE, 'Filter type must be specified.'  ;Default function
>>>  ENDCASE
>>>
>>>  ;Create filter
>>>  IF (count gt 0) THEN filter[q] = 1.0
>>>  ;Compute Fourier transform (z is a 1-D 9028 array of 250hPa geo ht)
>>>  g  = FFT(z)
>>>  ;Filter the signal
>>>  gg = filter*g
>>>  ;Compute the inverse FFT, which is the time filtered time series at one grid
>>>  ggg = FFT(gg, /INVERSE)
>>>
>>>
>>>  ggg is my highpass filtered time series at one grid cell from which I can now compute the variance. But, I get into all sorts of trouble when I try to loop the above through lon and lat to calculate the highpass time series at each longitude and latitude grid cell. Does anybody have experience in using IDL to compute highpass filtering across a spatial domain over many days (as in 30 years worth)? Basically, I want to compute ggg for all grid cells across my domain. Sounds simple, right? What am I missing? The below spirals out of control and is filled with errors.
>>>
>>>  g = dcomplexarr(201,101,9028)
>>>  gg = dcomplexarr(201,101,9028)
>>>  ggg = dcomplexarr(201,101,9028)
>>>  IF (count gt 0) THEN filter[q] = 1.0     ;Create filter
>>>     for i = 0,200 do begin
>>>       for j = 0,100 do begin
>>>        for k = 0,9027 do begin
>>>            g[i,j,k]  = FFT(z)            ;Compute Fourier transform
>>>           gg[i,j,k] = filter*g[i,j,k]     ;Filter the signal
>>>           ggg[i,j,k] = FFT(gg, /INVERSE)
>>>        endfor
>>>       endfor
>>>     endfor
>>>
>>>
>>>  Fingers (and toes) crossed.
>>>  Thank you,
>>>  teddy
>>
>>  Hi,
>>  have you looked/tied the dimension keyword of FFT:
>>
>>   http://www.harrisgeospatial.com/docs/FFT.html#F_848155245_37 862
>>
>>  This should do the trick. Try setting dimension=3
>>
>>  g = fft(z, dimension=3)
>>  gg = filter*g
>>  ggg = fft(gg,/inverse,dimension=3)
>>
>>  I have not tested this, but should do the work...
>>
>>  Cheers,
>>  Helder
>>
>  whereby
>
>  filter_1d=dblarr(1,1,n)
>  filter_1d[q]=1.
>  filter=rebin(filter_1d,[201,101,n],/sample)
```

Hi all,

In case it's not clear, Helder's three lines of code (with Markus' code added before it) replace your entire last 13-line section.

Helder's method is clearly the IDL Way and that's what you should use (it will be *much* faster than your nested loops), but if you're wondering why your code didn't work (there's always room for learning from things like this!), you really only want two levels of loop, and it might work as follows:

(I'll assume there's a variable "allZ" that is a dblarr(201,101,9028) with your source data)

```
g = dcomplexarr(201,101,9028)
gg = dcomplexarr(201,101,9028)
ggg = dcomplexarr(201,101,9028)
IF (count gt 0) THEN filter[q] = 1.0     ;Create filter

for i = 0,200 do begin
  for j = 0,100 do begin
      z = allZ[i,j,*]                  ; Will be [1, 1, 9028]
      g[i,j,0]  = FFT(z)               ;Compute Fourier transform, lay result into g
      gg[i,j,0] = filter*g[i,j,k]      ;Filter the signal
      ggg[i,j,0] = FFT(gg, /INVERSE)
  endfor
endfor
```

Hope this helps!

Cheers,
-Dick

Dick Jackson Software Consulting Inc.
Victoria, BC, Canada --- http://www.d-jackson.com

Subject: Re: highpass spatial filtering (lon,lat,time)
Posted by Teddy Allen on Fri, 22 Sep 2017 01:47:07 GMT
View Forum Message <> Reply to Message

Thank you so much Helder, Markus, and Dick. Your tips and insights were a great help towards achieving a properly working code (the IDL way!). MUCH appreciated.
teddy