
Subject: Re: Digital filter in IDL

Posted by [zawodny](#) on Mon, 04 Dec 1995 08:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

In article <DlxKGH.IMJ@hpl.hp.com> peter@hpl.hp.com (Peter Webb) writes:

> Meili Zhong (yafeng@leland.Stanford.EDU) wrote:

> : Does anyone know if there is a butterworth filter in IDL?

>

> : There is a filter called digital_filter in IDL. It is supposed not

> : to cause phase shift. But when I implemented this digital filter

> : in to my data, it does cause phase shift. Anybody tried this

> : routine?

>

> Butterworth filters, and their digital equivalents, are IIR filters, and

> as such have non-linear phase response, which cannot be corrected

> (exactly) by discarding samples. A useful trick is to pass your data

> through such filters once forwards, then again backwards, to cancel out

> the non-linearities. The major advantage of IIR filters is in reducing

> the number of computations required to achieve a certain sharpness of

> transition, so unless you are doing real-time filtering, the FIR filter

> produced by digital_filter should do the trick.

Here is what I use for a butterworth filter function:

```
function BUTTERWORTH,f,f0,q=q
```

```
; Routine to calculate the 2nd order Butterworth Filter function
```

```
; f Array of frequencies to evaluate the filter at (Hz).
```

```
; f0 The filter 3db point (Hz).
```

```
; q A damping factor. Butterworth => q = 1./sqrt(2.)
```

```
if(n_elements(q) eq 0) then q = 1./sqrt(2.)
```

```
fof0 = f/f0
```

```
mag = (1. - fof0^2)^2 + (fof0/q)^2
```

```
; Calculate the real and complex filter values
```

```
return, complex((1.-fof0^2.)/mag,-fof0/q/mag)
```

```
end
```

```
pro FILTER,a,fa,sps=sps,f0=f0,q=q
```

```
; Routine to apply a 2nd order Butterworth filter to a signal
```

```
; a Input signal vector to be filtered
```

```
; fa The output of the filter given "a" as input
```

```
; sps An alternate data sampling rate (default 64 sps)
```

```
; f0 An alternate filter 3db point (default 24.0 Hz)
```

```
; q A damping factor. Butterworth => q = 1./sqrt(2.)
```

```
if(n_elements(f0) eq 0) then f0 = 24. ; Filter 3db point (Hz)
```

```
if(n_elements(sps) eq 0) then sps = 64. ; Samples per second
if(n_elements(q) eq 0) then q = 1./sqrt(2.)
```

```
n = n_elements(a)
; Frequency array for FFT
f = (sps/n) * shift( findgen(n)-(n/2-1), n/2+1)
; Apply the filter
fa = float( fft( fft(a,-1) * butterworth(f,f0,q=q), 1))
return
end
```

Enjoy,

--

Dr. Joseph M. Zawodny	KO4LW	NASA Langley Research Center
E-mail: J.M.Zawodny@LaRC.NASA.gov		MS-475, Hampton VA, 23681-0001
