```
Deviation: 0.0236464 0.0236464
Deviation in dB: 0.4108557 -32.5247154
Extremal frequencies:
0.0000000 0.0312500 0.0625000 0.0878906 0.1000000
0.1500000 0.1617188 0.1871094 0.2183594 0.2496094
0.2828125 0.3160156 0.3492188 0.3824219 0.4156250
0.4507813 0.4839844
```

In the above, we have designed a lowpass filter with 32 taps with the edge of the passband at 0.1 Hz and the edge of the stopband at 0.15.

6.3 wfir — window method FIR filter design

`Wfir` is a Fortran program performing classical FIR filter design using the window method. There is nothing unusual about this program, so almost any DSP book will give an adequate explanation of the algorithm as well as the meaning of its arguments. Briefly, the program permits the design of lowpass, highpass, bandpass, and bandstop filters using any of a number of windows. The method is to first compute the impulse response of an ideal (brick wall) filter, and then window it with the selected window to make the impulse response finite.

The window method can also be implemented directly using Ptolemy block diagrams. See “FIR filter design” on page 3-138.

```
Enter weight factor for band 2:
     1
```

See comment above on weight.

```
Do you want x/sin(x) predistortion? (y/n):
     n
```

Note that the program will design a filter that predistorts to compensate for the effect of the zero-order hold in a D/A converter.

```
Enter name of coefficient output file (Sorry, no tilde-
expansion. Give path relative to your home directory):
     filter_taps
```

The name of the file in which to store the impulse response of the design. The resulting file can be used in an FIR star or a WaveForm star using the syntax `< filename` to read it. Note that the file will be stored in your home directory.

```
Executing ...


Finite Impulse Response (FIR)
Linear Phase Digital Filter Design
Remez Exchange Algorithm
Bandpass Filter
Filter length = 32


Impulse Response Coefficients:
h( 1) = 0.2199929E-02 = h( 32)
h( 2) = -0.1615105E-01 = h( 31)
h( 3) = -0.1167266E-01 = h( 30)
h( 4) = -0.5506404E-02 = h( 29)
h( 5) = 0.6444952E-02 = h( 28)
h( 6) = 0.1864344E-01 = h( 27)
h( 7) = 0.2227415E-01 = h( 26)
h( 8) = 0.1105212E-01 = h( 25)
h( 9) = -0.1328082E-01 = h( 24)
h( 10) = -0.3894535E-01 = h( 23)
h( 11) = -0.4806972E-01 = h( 22)
h( 12) = -0.2521652E-01 = h( 21)
h( 13) = 0.3334328E-01 = h( 20)
h( 14) = 0.1151020E+00 = h( 19)
h( 15) = 0.1945332E+00 = h( 18)
h( 16) = 0.2434263E+00 = h( 17)


Lower band edge: 0.0000000 0.1500000
Upper band edge: 0.1000000 0.5000000
Desired value: 1.0000000 0.0000000
Weight factor: 1.0000000 1.0000000
```

```
Enter name of input command file (press <Enter> for manual
entry, Sorry, no tilde-expansion. Give path relative to your
home or start-up directory):
      <Return>
```

You can put your responses in a file to avoid having to type them each time you execute the program. Here, I assume you are typing them interactively.

```
Enter filter type (1=Bandpass, 2=Differentiator, 3=Hilbert
transformer, 4=Half-band):
      1
```

Lowpass, Highpass and Bandpass filters are all called Bandpass.

```
Enter filter length (enter 0 for estimate):
      32
```

The number of taps. If you enter 0, the program figures out how many you need for your specification.

```
Enter sampling rate of filter:
      1
```

Use 1Hz.

```
Enter number of filter bands:
      2
```

There will be a passband and a stopband.

```
Enter lower band edge for band 1:
      0
```

For a lowpass filter, this should be 0 for d.c.

```
Enter upper band edge for band 1:
      0.1
```

Upper edge of the passband.

```
Enter desired value for band 1:
      1
```

Specify unity gain in the passband.

```
Enter weight factor for band 1:
      1
```

Using 1 here and 1 in the stopband will give ripples of the same size in both bands. You could experiment with allowing more ripple in the passband by making this number smaller.

```
Enter lower band edge for band 2:
      0.15
```

Lower edge of the stopband.

```
Enter upper band edge for band 2:
      0.5
```

Since this is the Nyquist frequency, it specifies the top of the stopband.

```
Enter desired value for band 2:
      0
```

Zero here defines this to be the stopband.

Chapter 6. Filter design programs

6.1 Introduction

Pending the inclusion of more sophisticated filter design software with Ptolemy, this distribution includes two Fortran programs for this purpose. These design FIR filters using the Parks-McClellan algorithm or the window method. The frequency sampling method can be done directly using Ptolemy stars, as shown in the demos included with the SDF domain.

These programs can be invoked standalone or through the filter command in pigi. If invoked through `pigi`, then an `xterm` window will be opened and the program started.

Filter specifications can be entered by hand or loaded from a file. The first prompt from the program is the name of the input command file. Simply typing return will result in manual entry. To enter data from a file, the data should appear in the file in exactly the order that it would appear if it were being entered by hand, with one question answered per line. Unfortunately, these command files are rather difficult to read, and not too easy to create correctly. It is recommended to first do manual entry, then imitate the entries in a file. Data can be entered in any reasonable numeric format.

Caveats

These are public domain Fortran programs that have been converted from C, provided for convenience; they are not an integral part of Ptolemy. Incorrect formatting of data line can lead to ungraceful exits (often with a core dump) or incorrect results. Either program can be invoked through the Ptolemy graphical interface, in which case it is started in its own window, in the background. Note that in this case the program always starts in the current working directory of `pigi`, probably your home directory.

6.2 optfir — equiripple FIR filter design

`Optfir` is a Fortran program performing classical FIR filter design using the Parks-McClellan algorithm. There is nothing unusual about this program, so almost any DSP book will give an adequate explanation of the algorithm as well as the meaning of its arguments. Briefly, the program permits the design of bandpass filters, differentiators, hilbert transformers, and half-band filters.

Bandpass filters include lowpass, highpass, bandstop, and multiband, where each band can have a different gain (the "desired value"). A desired value of 0 specifies a stopband. The weight associated with each band determines the ratio of ripple in each band; a higher weight means less ripple.

The following example should serve to illustrate use of the program. Invoke the program through `pigi` by calling up “equiripple FIR” in the “Filter” menu. Alternatively, you can start the program directly from any terminal window by typing the command `optfir`. The questions posed by the program are indicated below. The text in **`Courier-Bold`** are your response.