Filter Design Results

Generated by: http://www-users.cs.york.ac.uk/~fisher/mkfilter

Summary

You specified the following parameters:

filtertype	= Butterworth
passtype	= Lowpass
ripple	=
order	= 6
samplerate	= 12000
corner1	= 3500
corner2	=
adzero	=
logmin	=

Results

```
Command line: /www/usr/fisher/helpers/mkfilter -Bu -Lp -o 6 -a 2.9166666667e-01 0.0000000000e+00
raw alpha1    =   0.2916666667
raw alpha2    =   0.2916666667
warped alpha1 =   0.4148295201
warped alpha2 =   0.4148295201
gain at dc    :   mag = 1.630918229e+01   phase =   0.0000000000 pi
gain at centre:   mag = 1.153233339e+01   phase =   0.5000000000 pi
gain at hf    :   mag = 0.000000000e+00

S-plane zeros:

S-plane poles:
         -0.6745990931 + j   2.5176380903
         -1.8430389971 + j   1.8430389971
         -2.5176380903 + j   0.6745990931
         -2.5176380903 + j  -0.6745990931
         -1.8430389971 + j  -1.8430389971
         -0.6745990931 + j  -2.5176380903

Z-plane zeros:
         -1.0000000000 + j   0.0000000000       6 times

Z-plane poles:
         -0.2070552361 + j   0.7464101615
         -0.1537831799 + j   0.4058274196
         -0.1338941254 + j   0.1293317937
         -0.1338941254 + j  -0.1293317937
         -0.1537831799 + j  -0.4058274196
         -0.2070552361 + j  -0.7464101615

Recurrence relation:
y[n] = (  1 * x[n- 6])
     + (  6 * x[n- 5])
     + ( 15 * x[n- 4])
     + ( 20 * x[n- 3])
     + ( 15 * x[n- 2])
     + (  6 * x[n- 1])
     + (  1 * x[n- 0])
```

```
    + ( -0.0039161874 * y[n- 6])
    + ( -0.0393599704 * y[n- 5])
    + ( -0.2150444144 * y[n- 4])
    + ( -0.5327615714 * y[n- 3])
    + ( -1.1436225376 * y[n- 2])
    + ( -0.9894650828 * y[n- 1])
```

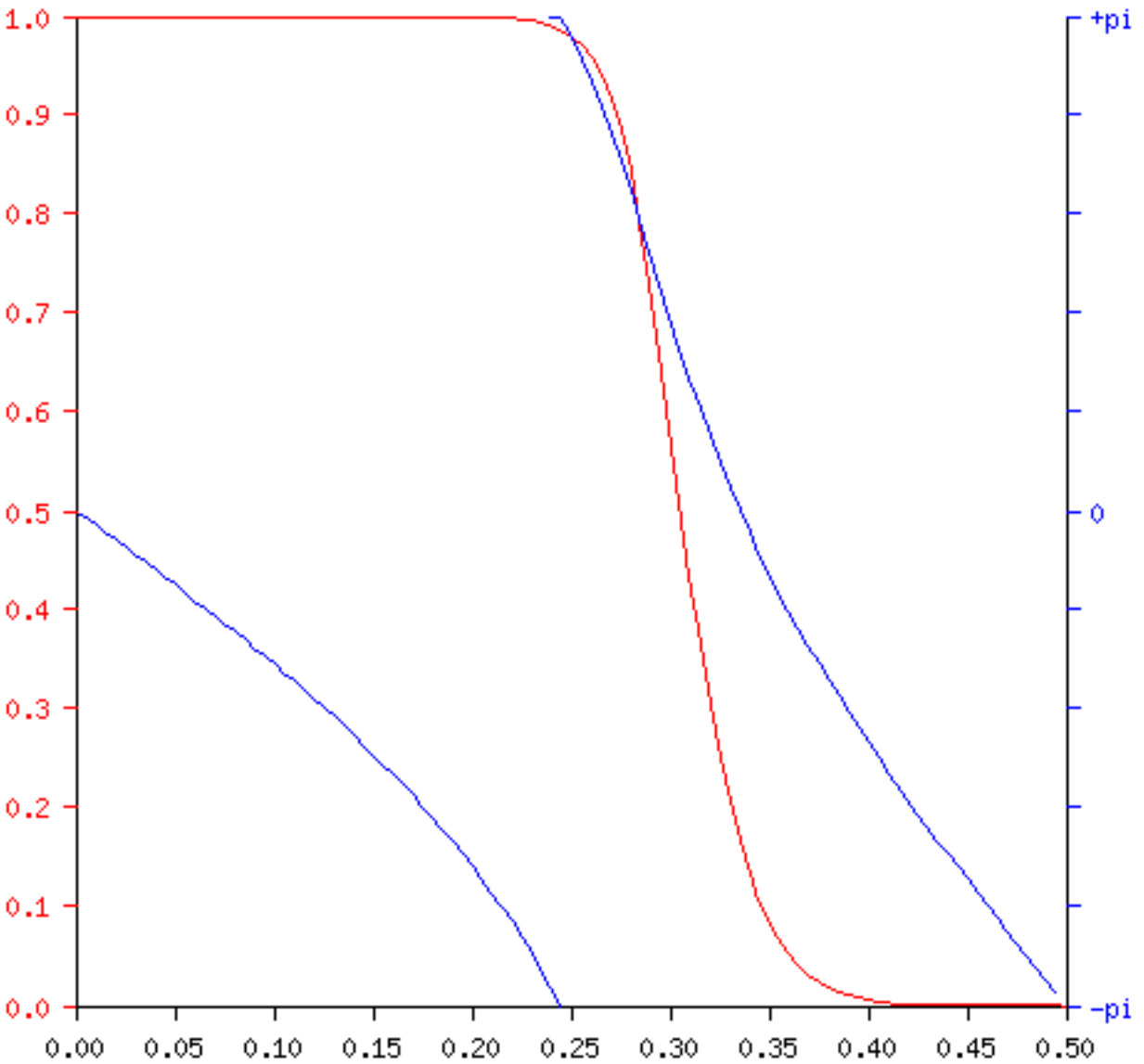
Ansi ``C'' Code

```
/* Digital filter designed by mkfilter/mkshape/gencode   A.J. Fisher
   Command line: /www/usr/fisher/helpers/mkfilter -Bu -Lp -o 6 -a 2.9166666667e-01 0.0000000000e+00 -l */

#define NZEROS 6
#define NPOLES 6
#define GAIN   1.630918228e+01

static float xv[NZEROS+1], yv[NPOLES+1];

static void filterloop()
  { for (;;)
      { xv[0] = xv[1]; xv[1] = xv[2]; xv[2] = xv[3]; xv[3] = xv[4]; xv[4] = xv[5]; xv[5] = xv[6];
        xv[6] = next input value / GAIN;
        yv[0] = yv[1]; yv[1] = yv[2]; yv[2] = yv[3]; yv[3] = yv[4]; yv[4] = yv[5]; yv[5] = yv[6];
        yv[6] =   (xv[0] + xv[6]) + 6 * (xv[1] + xv[5]) + 15 * (xv[2] + xv[4])
                     + 20 * xv[3]
                     + ( -0.0039161874 * yv[0]) + ( -0.0393599704 * yv[1])
                     + ( -0.2150444144 * yv[2]) + ( -0.5327615714 * yv[3])
                     + ( -1.1436225376 * yv[4]) + ( -0.9894650828 * yv[5]);
        next output value = yv[6];
      }
  }
```

Download code and/or coefficients: TERSE VERBOSE

Magnitude (red) and phase (blue) vs. frequency

- *x* axis: frequency, as a fraction of the sampling rate (i.e. 0.5 represents the Nyquist frequency, which is 6000 Hz)
- *y* axis (red): magnitude (linear, normalized)
- *y* axis (blue): phase

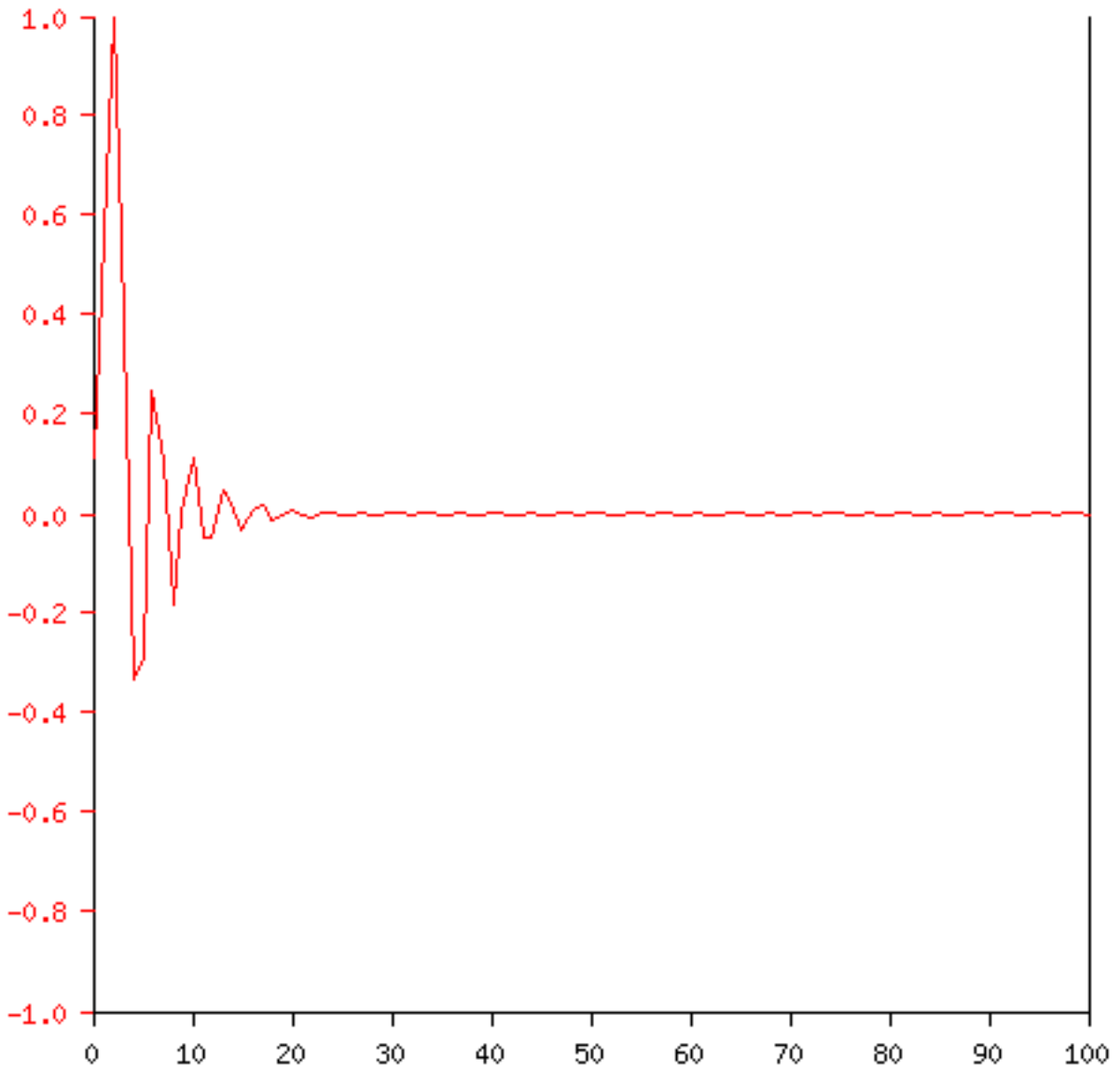


For an expanded view, enter frequency limits (as a fraction of the sampling rate) here:

Lower limit: Upper limit: zoom

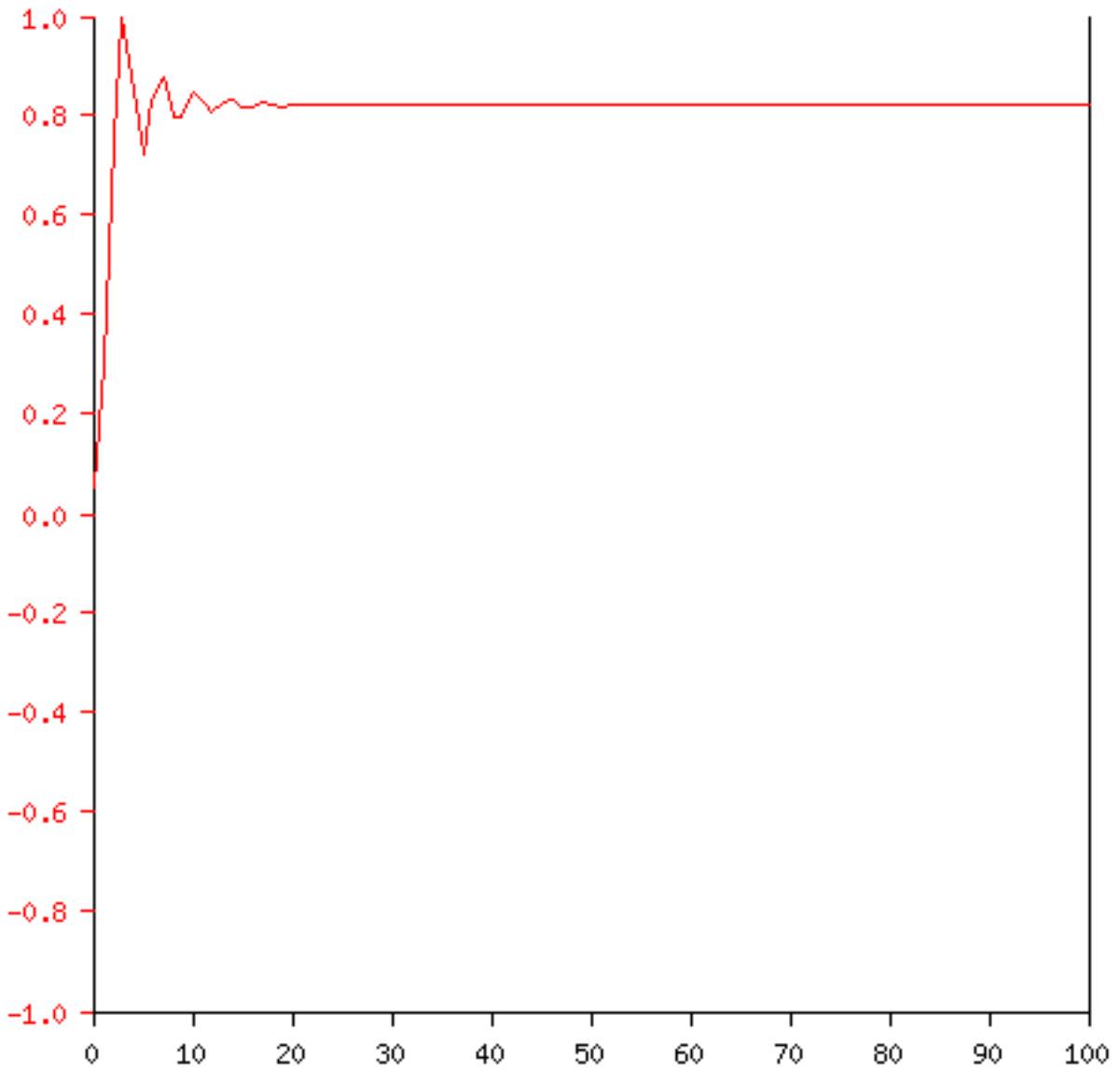
Impulse response

- *x* axis: time, in samples (i.e. 12000 represents 1 second)
- *y* axis (red): filter response (linear, normalized)



Step response

- *x* axis: time, in samples (i.e. 12000 represents 1 second)
- *y* axis (red): filter response (linear, normalized)



For a view on a different scale, enter upper time limit (integer number of samples) here:

Upper limit: zoom

Tony Fisher *fisher@minster.york.ac.uk*