Lab 4 – filter design

Experiments

1. Prepare 2-nd order filter with no significant zeros and with poles at $0.9e^{\pm j0.2\pi}$.(hint: `help poly`) Plot its impulse response, frequency response (mag. and phase, use 64 points for freqz), group delay (approximate dy/dx with $\Delta y/\Delta x$ using `diff`). Save coefficients for future use (e.g. on paper, just in case of reboot). Test the filter with sine waves of different frequencies (check if the measured gain is consistent with the plotted frequency response).

2. Put poles at $0.9e^{\pm j0.8\pi}$. Compare impulse/frequency plots..

3. Design an easy (wide transition band) lowpass FIR filter using `remez`. Use order of 10, increase if necessary. Then put narrower transition constraint with the same order – check the change in frequency characteristics. Allow greater order.
   Hint: stopband gain is better viewed in log (decibel) scale. But decibels may run to $-\infty$ for zero input, so use `max(-80,yourvalue-in-decibels)` for nice plot.

4. Design FIR filters for the same specifications, but by window method (Standard: use `fir2`. *Extra: do it by hand, using your knowledge from lecture.*)

   (a) with boxcar window
   (b) with hamming window

   Identify the window effects. Compare results against the Parks-McClellan filter.

5. Design an IIR LP filter with passband till 0.2π, stopband from 0.3π, passband ripple 0.5 dB, stopband 80 dB down (not all the specifications are used with different filter types – see help). Compare following types:

   (a) Butterworth
   (b) Chebyshev type I and II
   (c) Cauer (elliptic)

   Plot zeros and poles, and frequency characteristics for each filter.

6. Record (`y=getdata(Nsamples_in_row, [Krepeats, [Tsampling, [leave_bias]]])`) 1024 samples of:

   (a) a sinusoid being in passband of the designed filters
   (b) a sinusoid being in stopband
   (c) a square wave with base frequency in passband and harmonics in stopband
   (d) a noise

   Use your saved 2-nd order filter to obtain coloured noise as a fifth signal.

   Mix the five signals (by digital summation in matlab) to taste, then design filters to separate components. Plot spectral and time-domain plots before and after filtering.

   Use `sound(x, 1/tsampling);` to listen to the results (plug headphones into the green jack of the PC sound card).

7. *Calculate `roots(poly(1:20))` and `roots(poly(1:22))`. Compare the biggest and smallest coefficient of a polynomial. Try to guess how the experiment corresponds to the lecture on filter design.*

MATLAB resources for filter design

Note: in the parameters of MATLAB filter design functions, the frequency value is a multiplier to π (e.g. 1.0 means $f = f_s/2$ or equivalently $\theta = 1.0 \cdot \pi$).

filter implements a digital filter

```
Y = filter(B, A, X);
```

$$y(n) = \sum_{k=1}^{length(B)} B_k \cdot x(n-(k-1)) - \sum_{l=2}^{length(A)} A_l \cdot y(n-(l-1))$$

freqz computes digital filter frequency response
`[H,W] = freqz(B,A,N);` $H(e^{jW})$, B,A - coefficients, N - num. points
Put A=[1] for FIR filters. In Matlab $\geq$ 4.0 freqz with nargout==0 gives nice plot of magnitude and phase of H.

abs calculates the magnitude of a complex number

angle calculates the phase angle of a complex number (in radians)

unwrap `unwrap(angle(H))` unwraps jumps from $-\pi$ or π (the phase is then continuous, but it may turn out greater than $\pm\pi$)

remez (in Matlab > 6.2 renamed as **firpm**) Parks-McClellan optimal equirriple FIR filter design.
`B = remez(N,F,M);` N - order, F - frequencies, M - magnitudes
example: F=[0 0.1 0.2 1], M=[1 1 0 0] specifies LP filter with passband of 0.1π, stopband from 0.2π to $1 \cdot \pi$ (`plot(F,M);` plots the specified frequency response)

fir2 FIR filter design using the window method
`B = fir2(N,F,M,win);` as in remez; win – chosen window (e.g `bartlett(N+1)`)

windows: `blackman boxcar butter chebwin hamming hanning kaiser`

******ord** compute the minimum order of a digital filter with specified constraints (use help). Substitute *butt, cheb1, cheb2, ellip* for ****.

********* design a digital filter using a bilinear transformation. Substitute *butter, cheby1, cheby2, ellip* for ****.

poly *poly(r)* gives the vector of coefficients of a polynomial whose roots are specified in vector r.

roots *roots(p)* gives roots of a polynomial whose coefficients are in vector p

Elliptical filters are also known as Cauer filters.

File: lab4 LaTeXed on December 4, 2009