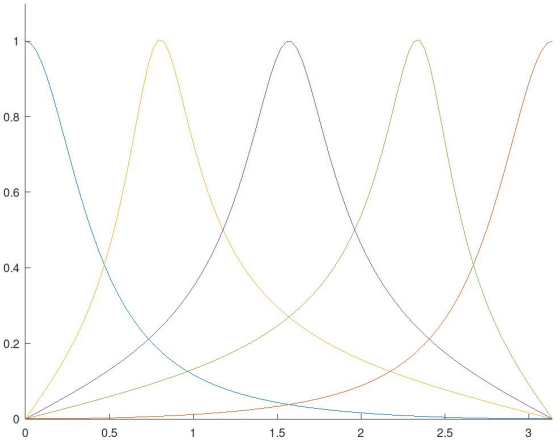
CompMus 2020 — Segundo Trabalho Maior
Equalizador Programável
Entrega da 1ª fase: 14/11/19 até 23:55
Entrega da 2ª fase: 28/11/19 até 23:55

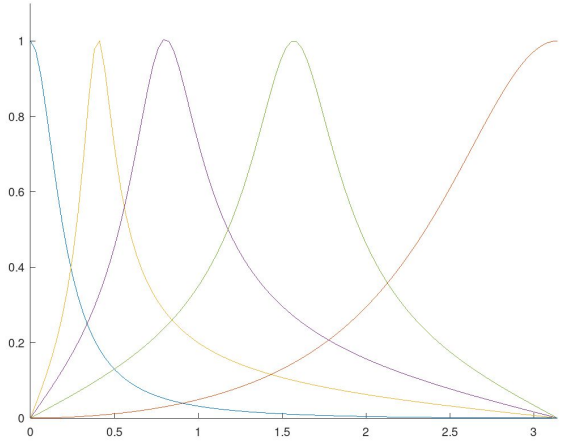
O objetivo deste trabalho é implementar um equalizador programável para sinais de áudio estéreo. A entrega está dividida em duas fases: na primeira fase faremos um equalizador com um número fixo de 5 faixas de frequência, e na segunda fase generalizaremos a construção para um número arbitrário de faixas definidas pelo usuário através dos parâmetros do objeto.

Especificação da primeira fase

Na primeira fase iremos partir de uma implementação existente `equalizador~.pd` de um equalizador de N=5 faixas, que pode ser baixado do e-disciplinas. O equalizador é implementado como um banco de N filtros passa-faixas em paralelo, sendo que o ganho de cada faixa é controlado por um slider vertical. Os filtros passa-faixas possuem 2 polos e 2 zeros, sendo que cada par de polos (complexo-conjugados) é "afinado" em uma das frequências centrais, e tem sua magnitude determinada por uma condição de sobreposição das respostas em frequência do banco de filtros. A figura ao lado ilustra o banco de filtros do `equalizador~.pd` original.



As principais diferenças entre aquele equalizador e o que deveremos entregar na primeira fase são: (1) o equalizador fornecido possui 5 faixas fixas, distribuídas linearmente (frequências centrais de 0, $\frac{\pi}{4}$, $\frac{\pi}{2}$, $\frac{3\pi}{4}$, π radianos/amostra[1], ou equivalentemente 0, 5512.5, 11025, 16537.5 e 22050 Hz quando R=44100), ao passo que nosso equalizador possui 5 faixas programáveis; a figura ao lado ilustra uma possível aplicação, onde as faixas estão divididas em intervalos de oitava; (2) as posições dos polos nos filtros do `equalizador~.pd` original também são fixas, enquanto as nossas serão calculadas em função das frequências centrais; e (3) o equalizador fornecido possui apenas um canal, enquanto o nosso deve ser estéreo, ou

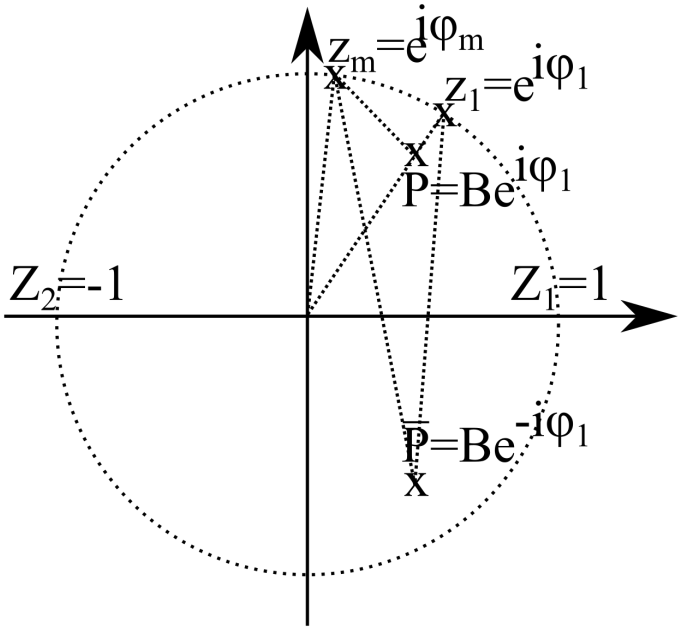


[1] Não custa lembrar que cada frequência *F* em Hz corresponde ao ângulo 2πF/R, onde *R* é a taxa de amostragem.

seja, ter 2 **[inlet~]** e 2 **[outlet~]**. São poucas coisas para mudar em relação à implementação dada, por isso essa fase terá um prazo de entrega mais curto. O objetivo aqui é mergulharmos rapidamente no contexto do EP2.

Sua implementação deve computar as fórmulas que inicializam os parâmetros dos polos a partir das 5 frequências (em Hz) informadas na criação do objeto. O exemplo da última figura corresponde ao **[equalizador~ 0 2756.25 5512.5 11025 22050]**, que corresponde às frequências angulares de $0, \frac{\pi}{8}, \frac{\pi}{4}, \frac{\pi}{2}, \pi$ (rad/amostra). A lista de parâmetros pode ser acessada com um **[loadbang]→|args<→[pdcontrol]→** e a partir dessa saída é possível processar a lista de frequências (em Lua ou em Pd puro) para inicializar os filtros individuais.

Para cada frequência central F_k Hz, devemos calcular o ângulo $\varphi_k = 2\pi F_k/R$, para então determinar um dos polos $P_k = B_k e^{i\varphi_k}$ do filtro de 2 polos e 2 zeros. A magnitude B e o ângulo φ são usados para inicializar cada abstração **[2polos2zeros~ B phi]**, também fornecida com o enunciado, e que cria automaticamente um *filtro normalizado* com polos $Be^{i\varphi}$ e $Be^{-i\varphi}$, além de zeros em DC e Nyquist; a própria abstração se encarrega de jogar os 2 zeros em Nyquist se os polos estão em DC, e de jogar os 2 zeros em DC se os polos estão em Nyquist, além de normalizar o filtro resultante, assim não teremos que nos preocupar com nada além da definição de um dos polos. A magnitude B de cada polo deve ser calculada de tal forma a compatibilizar a largura de banda com um dos filtros vizinhos: se um filtro de frequência (angular) φ_1 (não necessariamente o primeiro da lista) possui um filtro vizinho (à esquerda ou à direita) de frequência angular φ_2, então na frequência intermediária $\varphi_m = \frac{\varphi_1+\varphi_2}{2}$ gostaríamos que os dois filtros tivessem ganho de $\frac{1}{2}$ (veja as duas figuras anteriores para ganhar intuição). Uma condição para determinar a magnitude do polo do filtro de frequência φ_1 em relação à frequência φ_m pode ser deduzida da figura ao lado. Se $H(z) = a_0 \frac{(z-1)(z-(-1))}{(z-P)(z-\overline{P})}$ é a função de transferência do filtro com frequência φ_1, queremos garantir que $|H(z_m)| = \frac{1}{2}H(z_1)$, ou seja, que



$$a_0 \frac{|z_m-1||z_m+1|}{|z_m-P||z_m-\overline{P}|} = \frac{a_0}{2} \frac{|z_1-1||z_1+1|}{|z_1-P||z_1-\overline{P}|},$$

o que ainda pode ser simplificado eliminando-se a_0, elevando-se ao quadrado (para eliminar as raízes quadradas dos módulos) e fazendo-se o produto em cruz para eliminar as divisões. Um valor aproximado da solução $B = |P|$ pode ser obtido por busca binária no intervalo [0...0.999], usando-se o código `Botimo.pd` disponível no e-disciplinas.

Note que a estratégia acima é apenas uma heurística, pois calcular B a partir de φ_1 e φ_2 não garante que o filtro em φ_2 também terá ganho de $\frac{1}{2}$ na frequência intermediária $\frac{\varphi_1+\varphi_2}{2}$, a menos que a magnitude do filtro em φ_2 também fosse definida em função de φ_1, mas aí o problema

apenas mudaria de lugar. Para simplificar o desenho dos filtros, consideraremos que a magnitude de cada polo será calculada em função do filtro vizinho *que estiver na direção do centro do espaço de frequências*, ou seja, para $\varphi_1 < \pi$ tomaremos $\varphi_2 > \varphi_1$ e para $\varphi_1 \geq \pi$ escolheremos como vizinho $\varphi_2 < \varphi_1$.

O tratamento de sinais de áudio estéreo será feito duplicando-se o banco de filtros: o canal esquerdo deve ser tratado por uma das cópias do banco de filtros, e o canal direito pela outra. As frequências centrais e larguras de banda são rigorosamente as mesmas, os sliders são compartilhados, apenas o processamento de cada canal deve ser feito de forma independente.

Especificação da segunda fase

Na segunda fase seu equalizador deve ser adaptável a uma quantidade N arbitrária de faixas de frequência, definidas pelo usuário a partir da criação do objeto de acordo com o número de argumentos. Assim **[equalizador~ 0 5512.5 11025 16537.5 22050]** deve gerar um equalizador idêntico ao disponibilizado originalmente, ao passo que **[equalizador~ 0 3150 6300 9450 12600 15750 18900 22050]** deve produzir um equalizador com as 8 faixas de frequência indicadas.

Essa generalização dependerá do uso de técnicas de *metaprogramação*, também conhecida como *dynamic patching* em Pd, que veremos na aula de 13/11/2020 (não perca!) e na tarefa prática de 20/11/2020. Como veremos, a maneira mais simples de controlar a geração dinâmica de um patch é começar a partir de um patch contendo um único subpatch **[pd meta]**, e acrescentar instruções para a criação de objetos copiando todos os seus parâmetros de inicialização *a partir de versões dos mesmos objetos geradas manualmente*. Lembre-se sempre também de apagar os objetos gerados automaticamente antes de salvar seu patch.

Algumas dicas e sugestões complementares a esse enunciado serão postadas no fórum durante o período de desenvolvimento. Participe compartilhando suas dúvidas por ali.

Finalmente...

- Esse trabalho é individual: conversar com os colegas para tirar dúvidas da linguagem é absolutamente normal, mas compartilhar soluções específicas e códigos não...
- Leia o enunciado mais de uma vez. É comum surgirem dúvidas que já estão respondidas no enunciado, mas que não demos atenção na primeira leitura.
- Use o fórum para tirar dúvidas!
- Entregue quantas versões parciais você quiser, antes do prazo, por precaução.
- Divirta-se programando!