

CompMus 2024 – Terceiro Exercício-Programa

Síntese de Karplus-Strong

Entrega da 1ª fase: 01/12/24 até 23:55

Entrega da 2ª fase: 08/12/24 até 23:55

O objetivo deste trabalho é implementar um instrumento para síntese de sons de cordas dedilhadas usando o algoritmo de Karplus-Strong. A entrega está dividida em duas fases: na primeira fase implementaremos a versão mais simples do algoritmo, com todos os parâmetros fixados e permitindo apenas a síntese de uma nota por vez; na segunda fase, introduziremos parâmetros para controlar melhor o timbre do ataque, a duração e a frequência fundamental. Após a entrega, você poderá incrementar o instrumento para permitir a síntese simultânea de várias notas (polifonia).

Algoritmo de Karplus-Strong (KS)

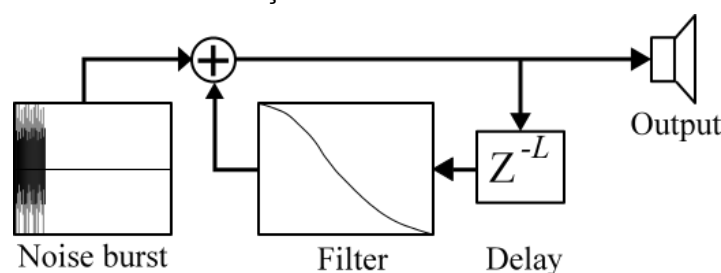
O algoritmo KS é um dos mecanismos de síntese mais conhecidos em processamento sonoro, sendo baseado na emulação de algumas características físicas dos cordofones beliscados, família de instrumentos que inclui alaúde, baixo, bandolim, cravo, guitarra, harpa e violão, entre outros. As características sonoras específicas emuladas pelo KS são:

- os sons possuem padrão harmônico (a partir de uma frequência fundamental definida);
- os ataques dos sons, embora curtos e rápidos, são bastante ruidosos/brilhantes;
- o decaimento sonoro possui padrão exponencial (do tipo α^t com $\alpha < 1$), ou seja, a cada intervalo de tempo Δt a amplitude é atenuada por um fator constante ($\alpha^{\Delta t}$);
- o decaimento sonoro depende da frequência, sendo mais rápido para as altas frequências (que são atenuadas mais rapidamente do que as baixas frequências).

As estratégias usadas para emular essas características são, respectivamente:

- a síntese se baseia na repetição aproximada de um padrão de L amostras (onde L corresponde ao período fundamental do som sintetizado);
- o padrão é inicializado com valores aleatórios (sinal aleatório \Rightarrow espectro rico);
- a cada repetição do padrão de L amostras um filtro é aplicado para atenuar o sinal;
- o filtro utilizado é do tipo passa-baixas (atenuação mais forte das altas frequências).

O diagrama a seguir ilustra essa construção:



A primeira propriedade está mapeada na repetição cíclica do padrão, realizada pelo atraso de L amostras e pela retroalimentação do sinal (o mesmo sinal que vai para a saída volta para o algoritmo). A segunda propriedade trata da inicialização: após introduzir L amostras aleatórias

no ciclo do diagrama, o algoritmo passa a funcionar por si só e a entrada externa deixa de existir. As duas últimas propriedades estão representadas na forma da resposta em frequência do filtro (que nesse exemplo se parece um pouco àquela do filtro da média).

Esse diagrama simplificado deixa várias questões em aberto para uma implementação razoável do KS. Em particular, nessa implementação o padrão de decaimento sonoro depende exclusivamente do filtro, e o sinal tem duração teórica infinita. Assim, ficam sem resposta questões como por exemplo "como controlar adequadamente a velocidade de decaimento e a duração do som?" ou "o que fazer se a frequência fundamental não corresponder a um período L inteiro?". Além destas, outras questões menos óbvias aparecem ao se investigar cuidadosamente a implementação. Essas questões serão tratadas nas seções a seguir, e também são discutidas muito mais detalhadamente no livro "Elements of Computer Music" de F. Richard Moore (seção 3.4.5, disponível no e-disciplinas).

Especificação da primeira fase

Na primeira fase iremos implementar o algoritmo KS em um arquivo **karplusstrong~.lua** usando a linguagem Lua¹. Sua implementação deve receber mensagens do tipo **|F D<**, onde F é uma frequência fundamental (em Hertz) e D é uma duração (em segundos). As mensagens serão recebidas por uma **function ofelia.list(lista)** que inicializará as estruturas necessárias para a síntese, que por sua vez será realizada pela **function ofelia.perform(bloco)**. Para simplificar a implementação, qualquer mensagem do tipo **|F D<** fará com que a síntese seja reiniciada, mesmo que um evento anterior ainda esteja sendo processado: para isso bastará reinicializar as estruturas de síntese com os valores correspondentes aos parâmetros novos.

A frequência F determinará o comprimento $L = \text{math.floor}(\text{ofGetSampleRate()}/F)$ de uma tabela que será processada ciclicamente². A tabela deve ser um atributo/variável do módulo (criada com **local tabelaKS = ofTable()** fora das funções/métodos) e inicializada com L valores aleatórios usando a expressão **2*math.random()-1**; em seguida, a tabelaKS deve ser corrigida subtraindo-se de cada posição a média μ dos L valores, *a fim de eliminar a componente dc do sinal gerado* (isso é importante porque a componente de 0 Hz nunca seria eliminada pelo filtro passa-baixas).

Apesar do diagrama que representa o KS sugerir um ciclo de L amostras, nosso algoritmo será executado dentro do ambiente padrão do Pd com seu tamanho de bloco de $N=64$ amostras. Assim, você deve considerar que o ciclo do KS e o ciclo do Pd em geral não coincidem, mas isso não fará muita diferença se você realizar a leitura/escrita cíclica da tabela do KS controlando um índice i de acesso à tabela que é atualizado com **$i = (i\%L)+1$** (lembre-se que os índices em Lua começam em 1). Nessa implementação do KS, convém pensar que o processamento do sinal, ilustrado no diagrama da página anterior, ocorre uma amostra de cada vez, dentro de um laço **for j=1, 64** na função **ofelia.perform**. Cada amostra x que entra no ciclo (no ponto indicado por \oplus) será lida da posição i da tabelaKS e será copiada no sinal de saída **bloco[j]** antes de passar pelos filtros de atraso de L amostras e pelo filtro passa-baixas. Mais ainda, como trataremos a informação do sinal sempre processando a mesma tabela de forma cíclica, o filtro de atraso *não precisa ser implementado diretamente*: basta deixar a amostra lá na posição i da tabelaKS que ela naturalmente será processada de novo

1 Para usar o código no Pd, basta criar um objeto [ofelia d -s11] e mandar a mensagem |read -c karplusstrong~.lua<. Usar -s11 permite usar uma tabela **a** como entrada da função perform para usá-la como saída do algoritmo. Você deve encapsular essa implementação em uma abstração karplusstrong~.pd incluindo um [inlet] e um [outlet~]. Exemplos nas notas de aula de implementações usando arquivos .lua podem ser encontrados nos patches **icsm15.pd**, **icsm22.pd** e **icsm51.pd**.

2 O tamanho da tabela equivale ao arredondamento da expressão $R/F-0.5$, que seria o período exato da onda sintetizada (mais detalhes sobre isso aparecerão a seguir, e também na Fase 2).

após L amostras (já que L é o tamanho da tabelaKS, que é lida circularmente).

O filtro passa-baixas utilizado na primeira fase será o filtro da média, cuja equação é $y(n) = \frac{1}{2}[x(n) + x(n-1)]$. **Atenção:** pela circularidade da tabelaKS, o termo $x(n-1)$ deve ser escrito como $tabelaKS[(n-2)\%L+1]$. A partir de sua equação, vemos que esse filtro possui resposta em frequência $H(z=e^{i\omega}) = \frac{1}{2}z^0 + \frac{1}{2}z^{-1} = z^{-\frac{1}{2}}\cos(\frac{\omega}{2})$, o que permite concluir que, além da maior atenuação das altas frequências, *o filtro passa-baixas atrasa meia amostra do sinal*, fato esse que impacta na duração efetiva/percebida do ciclo, que corresponde a um atraso de $L+\frac{1}{2}$ amostras, justificando a expressão utilizada para o L. Além disso, a saída do filtro deve não apenas alimentar o sinal de saída (variável **bloco**) como também voltar para a tabelaKS para ser processado no próximo ciclo do algoritmo de Karplus-Strong. **Atenção:** você deve ter o cuidado de não sobrescrever informações importantes ao processar a tabelaKS com o filtro passa-baixas, já que o valor de $x(n)$ será importante tanto para o cálculo de $y(n)$ quanto de $y(n+1)$; ou seja, apenas após computar o valor de $y(n+1)$ você poderá armazenar o valor de $y(n)$ em cima da posição anteriormente ocupada por $x(n)$ na tabelaKS; isso demanda variáveis auxiliares.

A duração D será usada para encerrar a síntese, que deve durar exatamente **math.floor(D*R+0.5)** amostras (valor arredondado de $D*R$). Como o final da síntese pode cair em qualquer posição em relação ao bloco do Pd, convém acompanhar o progresso do algoritmo KS contando o número de amostras sintetizadas desde a inicialização com a mensagem **|F D<**, a fim de garantir a terminação no momento correto. Para evitar sons desagradáveis (clicks) no final da síntese, *use as últimas 10 amostras da síntese para introduzir uma rampa de fade-out*, multiplicando a saída do algoritmo original por 0.9, 0.8, ..., 0.1 e 0.0, respectivamente.

Especificação da segunda fase

Na segunda fase nosso instrumento incluirá mecanismos mais acurados para o controle da frequência fundamental, da duração percebida do som e do timbre no ataque. Esses mecanismos serão discutidos nas seções a seguir. A implementação destes controles é bastante simples após a compreensão do enunciado (que não é tão simples). Após a entrega da Fase 2, você poderá utilizar as informações da seção opcional para incluir em seu instrumento a habilidade de processar várias notas simultâneas (polifonia), tratando a síntese correspondente a mensagens novas sem interromper o processamento de eventos ainda não concluídos.

Controle do timbre no ataque

A principal diferença entre uma corda dedilhada com força ou suavemente, ou ainda entre o uso da unha (ou palheta) e o uso da parte macia dos dedos, é o timbre inicial, que é espectralmente mais rico/ruidoso no primeiro caso e mais concentrado nas frequências graves no segundo. Uma maneira simples de obter esse efeito no contexto do algoritmo KS é deixar o processamento a partir da tabela aleatória inicial completar alguns ciclos através do filtro passa-baixas *antes de se começar a produzir a saída*. Isso pode ser feito na própria função de inicialização. Inclua nas mensagens do instrumento um terceiro parâmetro A (=ataque) que corresponde ao número de ciclos completos na tabela que devem ser processados na inicialização (A=0 é o algoritmo original, A=2 aplica o filtro passa-baixas duas vezes na tabelaKS inteira antes de começar a produzir o sinal de saída, etc). Aqui também é necessário o cuidado para

não sobrescrever informações importantes: o valor $x(n)$ continuará sendo necessário para computar $y(n+1)$, o que demanda o mesmo mecanismo usando variáveis auxiliares.

Controle da duração percebida

O uso do filtro passa-baixas por si só já introduz fatores de decaimento em função da frequência que determinam indiretamente uma duração percebida do som (até que se torne inaudível devido ao decaimento acumulado). Essa duração percebida pode diferir consideravelmente da duração D especificada na mensagem de inicialização. Mais ainda, um som de corda que terminasse naturalmente em 2 segundos (do ponto de vista da duração percebida) não se extingue com a mesma distribuição espectral de um outro som que naturalmente durasse 10 segundos mas fosse interrompido abruptamente após 2 segundos com uma rampa de *fade-out*. A maneira mais adequada de controlar a duração do som é alterando a forma do filtro passa-baixa, e isso pode ser feito através de parâmetros de encurtamento e prolongamento da duração perceptual, a fim de aproximá-la da duração desejada D .

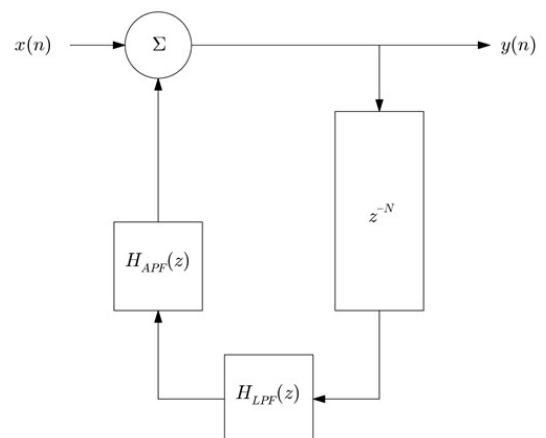
A equação do filtro passa-baixas modificado será $y(n) = \rho[(1-S)x(n) + Sx(n-1)]$ (equação 3-99 do Moore), onde $\rho \in [0,1]$ é um parâmetro de encurtamento e $S \in [0, \frac{1}{2}]$ é um parâmetro de prolongamento (observe que $\rho=1$ e $S=\frac{1}{2}$ correspondem ao filtro da média). A análise detalhada da resposta em frequência desse filtro pode ser encontrada no livro do Moore (pg. 284), mas como regra prática vamos adotar o seguinte procedimento:

- 1) calcule os ganhos $G_0 = \cos(\pi F/R)$ do filtro original da média na frequência F , e $G_1 = 10^{-3/(FD)}$ o ganho correspondente do filtro modificado considerando um decaimento desejado de 60 dB na frequência F após D segundos;
- 2) se $G_0 \geq G_1$, defina $\rho = \frac{G_1}{G_0}$ e $S = \frac{1}{2}$ (isso garante o ganho de G_1 na frequência F);
- 3) caso contrário, defina $\rho=1$ e calcule $S = \frac{1}{2} - \frac{1}{2} \sqrt{1 - \frac{4(1-G_1^2)}{2-2\cos(2\pi F/R)}}$ (fazendo $G(F) = G_1$)³.

O filtro resultante possui um atraso de S amostras (ao invés do atraso fixo de 0.5 amostra do filtro da média). Esse atraso deve ser contabilizado no cálculo de L : ao invés de $L = \text{math.floor}(R/F)$ como na primeira fase, definiremos⁴ $L = \text{math.floor}(R/(F-S))$ na segunda fase.

Controle da frequência fundamental

Após o ajuste do filtro passa-baixas, devemos considerar que a frequência **efetiva** do sinal produzido será definida pelo período de $L+S$ amostras, ou seja, será de $R/(L+S)$ Hz, o que ainda pode ser diferente da frequência F desejada. Para “afinar” o instrumento KS, será necessário introduzir um outro filtro no ciclo do diagrama, cuja única função será atrasar o



³ Essa equação corresponde à menor das soluções da equação quadrática 3-106 do Moore.


⁴ Para quem estiver muito atento vai parecer que falta um +0.5 nessa expressão, mas ela está correta, por causa do filtro adicional da próxima seção. L realmente tem que ser truncado para um valor inteiro menor, para que o filtro adicional complete o que falta para o atraso ideal correspondente à frequência fundamental F pretendida.

sinal o tanto necessário para se atingir exatamente a frequência F desejada. Isso será feito com um **filtro passa-tudo**, que é um filtro que não altera as relações de amplitude das componentes senoidais do sinal da entrada, porém introduz um deslocamento de fase que equivale à correção temporal desejada.

A equação do filtro passa-tudo usado para afinar a frequência fundamental do KS é $z(n) = Cy(n) + y(n-1) - Cz(n-1)$ (equação 3-102 do Moore, usando y para representar a entrada e z para a saída), onde C é um parâmetro a ser calculado em função do atraso $\Delta \in [0,1]$ necessário para se atingir o período fundamental. Como esse período é de R/F amostras, temos que $\Delta = R/F - (L+S)$ e assim o parâmetro C será calculado como $C = \frac{1-\Delta}{1+\Delta}$ (equação 3-107 do Moore). Note que esse filtro será aplicado sobre a saída $y(n)$ do filtro passa-baixas, e seu resultado deve ser armazenado na mesma tabela para ser reutilizado no ciclo KS (observe ainda que os dois filtros requerem variáveis auxiliares; na página 287 do Moore existe uma implementação em C que pode ajudar).

Processamento polifônico (seção opcional, para depois da entrega da Fase 2)

Para permitir que uma nova nota seja processada enquanto uma nota antiga não acabou, você precisará controlar várias tabelas em paralelo, onde cada nota possui seus parâmetros específicos (F, D, A, L, p, S, C). A maneira natural de gerenciar todas essas estruturas é usar uma tabela de eventos, a fim de permitir o controle independente tanto das tabelas de amostras usadas em cada síntese KS como também os parâmetros respectivos. Observe que Lua permite criar tabelas dentro de tabelas ($t[i] = \text{ofTable}()$; $t[i][1] = 2$) e também tabelas com campos nomeados, como structs em C ($t[i] = \text{ofTable}()$; $t[i].L = 5$). Para facilitar o gerenciamento automático da tabela de eventos, use as funções `table.insert` e `table.remove` da linguagem Lua, como no exemplo ao lado. Assim você poderá facilmente percorrer a lista de eventos para processar cada amostra, como também poderá remover facilmente os eventos que terminaram.



```
ofelia f;
t=ofTable();
table.insert(t, 2);
table.insert(t, "dois");
for i=1, #t do print(t[i]) end;
table.remove(t, 1);
for i=1, #t do print(t[i]) end;
```

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! Podemos postar códigos de teste (sem o instrumento KS propriamente dito), bem como arquivos wav com a síntese correspondente a algum exemplo em discussão.
- Entregue quantas versões parciais você quiser, antes do prazo, por precaução.
- Divirta-se programando!