MSAdd01

Manual Técnico

Outubro, 2021

1 Introdução

O MSAdd01 é um software de sintetizador musical do tipo aditivo no qual a síntese se dá pelo controle direto dos harmônicos de uma nota para definir seu timbre.

Toda lógica matemática por trás da síntese e controles disponíveis ao usuário foi programada no GNU-Octave. Toda interface gráfica e entradas do teclado foram programadas no Matlab na ferramenta App Designer.

2 Conceitos Musicais

2.1 Nota

Em uma linguagem, atribuímos *símbolos* a *ideias* ou *sons* para que os possamos registrar em algum meio *visível* (escrita).

Uma *nota* representa a unidade da emissão de som, ou seja, uma emissão que tem características mínimas, definidas ao longo de sua duração, enquanto que uma sequência de 2 ou mais notas simultâneas, são respectivamente chamadas de *intervalo* e *acorde*. As características normalmente atribuídas (além da duração) são a *altura* (frequência), *volume* (intensidade sonora) e *voz* (timbre). Uma *partitura* é uma forma *gráfica* de representar e armazenar todas essas informações.

Um teclado musical digital procura mimetizar um piano, logo as características de uma nota são normalmente definidas pelos seguintes procedimentos:

Duração: Tempo desde o pressionamento da tecla até a soltura desta.[1]

Frequência: A cada tecla é atribuída uma única frequência.

Intensidade: A intensidade é diretamente proporcional à força aplicada na tecla.

[1] Um instrumento só consegue "sustentar"uma nota se sua mecânica de produção de som for continuamente excitável, por exemplo, enquanto houver ar, pode-se manter uma nota em uma flauta.

Timbre: É determinado pela natureza acústica de um instrumento; no caso de um instrumento sintético, este parâmetro pode ser controlado atribuindo uma quantidade de controles equivalente ao grau de liberdade desejado.

Devido a limitações técnicas (ver secção 5), a duração e intensidade não podem ser simuladas por estes procedimentos, desta forma, são executadas por controles externos às teclas musicais.

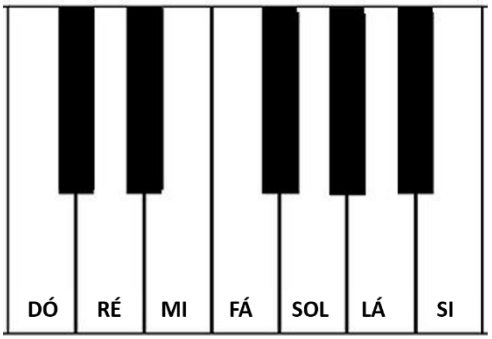

Figura 1: Teclas musicais

2.2 Altura

A atribuição entre frequências e teclas musicais tem o nome de *escala musical*. A escala utilizada neste trabalho é do tipo *maior* com *temperamento igual*, que possui as seguintes convenções e regras de construção:

- Uma *forma* possui 7 teclas brancas e 5 teclas pretas;
- A cada tecla branca é dado um *nome*, como pode ser visto na figura 3;
- Cada tecla preta pode ser nomeada de acordo com uma vizinha de referência, por exemplo, a tecla preta entre Dó e Ré, pode ser nomeada como Dó# (dó sustenido), ou Réb (ré bemol), dependendo do contexto;
- A *distância* de altura entre uma tecla preta e brancas adjacentes, ou duas teclas brancas adjacentes sem uma tecla preta no meio é denominada *semitom*;
- A *distância* de altura entre duas teclas pretas próximas ou duas teclas brancas adjacentes com um tecla preta no meio é denominada *tom* e equivale exatamente a 2 semitons;
- Um teclado musical é composto por uma repetição de formas; para diferenciar cada forma são atribuídos números inteiros de forma que uma forma a esquerda tem frequências mais graves e um número imediatamente inferior, enquanto que uma forma a direita tem frequências mais agudas e um número imediatamente superior;

- A nota de referência para todas as formas, conhecida como *lá central*, é nomeada como: Lá4 ou A4 e tem uma frequência exata de 440Hz;
- Notas com o mesmo nome em uma forma adjacente a direita tem exatamente o dobro de frequência (oitava), e vice-versa, ou seja: Lá3 = 220Hz e Lá5 = 880Hz;
- A frequência de cada tecla de uma forma é definida de forma que a *oitava* (x2) seja igualmente divida entre as 12 teclas (7 brancas + 5 pretas), ou seja, aumentar um semitom (ir para a próxima tecla da direita), equivale a multiplicar a frequência da nota pela décima segunda raiz de 2, e vice-versa, ou seja: Láb4 $\approx 415,30$ e Lá#4 $\approx 466,16$. (ver seccção 3.2)

2.3 Timbre

A frequência atribuída a uma tecla é nomeada de *fundamental*; frequências múltiplas inteiras da fundamental são nomeadas de *harmônicos*.

Instrumentos podem ser classificados em *tonais* ou *atonais*. O primeiro grupo consiste em instrumentos com *altura* bem definida e conteúdo frequencial concentrado nos harmônicos, como o piano; enquanto o segundo grupo enquadra instrumentos cujo conteúdo frequencial é *espalhado* e normalmente não possuem alturas definidas, como um tambor.

Este sintetizador simula e cria apenas *vozes* (instrumentos) do tipo tonal. (Ver secção 3.3)

2.4 Dinâmica

Instrumentos podem permitir variação *dinâmica* durante a duração da nota, em muitos casos diretamente proporcional à força aplicada pelo interprete, seja exercendo mais/menos força com as mãos, como no piano, ou emitindo um fluxo mais/menos intenso de ar, como na flauta.

A constituição física de um instrumento também atribui naturalmente algumas características dinâmicas (ver secção) devido suas propriedades elásticas:

- Após o impacto inicial de uma tecla de piano, o sistema demora um tempo para atingir seu valor máximo de intensidade; a este tempo é dado o nome de *Ataque* (Attack);
- Após o ataque, a intensidade, em excesso, decai até um valor estável; a este tempo é dado o nome de *Decaimento* (Decay);
- Se o instrumento possuir uma característica de *Sustentação* do estímulo, a intensidade estável pode ser mantida por um tempo nomeado de forma análoga (Sustain);
- Após cessar qualquer estímulo à produção de som, o som irá decair até níveis inaudíveis; assim como o período de ataque, instrumentos naturais não podem ter transições instantâneas devido a elasticidade do material; este estágio final da nota é nomeado de *Repouso* (Release);

3 Núcleo

O código do núcleo se divide em três blocos de processamento:

- Controle da Frequência;
- Controle do Timbre;
- Controle da Dinâmica.

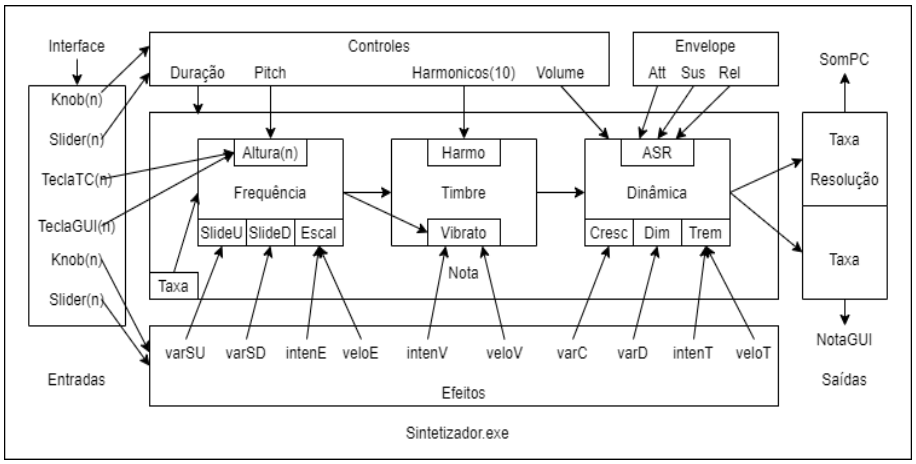

Figura 2: Diagrama de Blocos

Todos os cálculos relativos aos controles da síntese são executados a uma taxa fixada pelo programa equivalente a taxa de amostragem a ser enviada para a saída de áudio respeitando o critério de Nyquist para sons do espectro audível em uma resolução, também fixa, e ambos parâmetros não são acessíveis ao usuário:

- Taxa de Amostragem: 44100 amostras/segundo;
- Resolução do Áudio: 16 bits;

3.1 Duração da nota

A duração da nota é controlada pelo usuário e, assim como a taxa, é fundamental no cálculo de todos os módulos. Varia de 0,4 a 4 segundos.

3.2 Módulo de Frequência

A frequência fundamental de cada nota é atribuída através da escala *igualmente temperada*, explicada anteriormente, através do seguinte código:[2]

[2] A tecla de número 49 equivale ao A4, ou Lá4.

```
function Escalas(app)
    app.altura = @(tecla) app.A4.*(2).^((tecla-49)/12);
    app.escala = app.altura(1:88);
end
```

Além da frequência fundamental determinada pela tecla, é possível fazer alterações na frequência ao longo da duração da nota através do botão de *pitch* e de 4 efeitos disponibilizados ao usuário.

3.2.1 Slide-Up

Este efeito consiste em *aumentar* a altura da nota linearmente ao longo da duração. A posição do *slider* correspondente determina a *velocidade* em que ocorre a variação. O limite máximo de velocidade foi escolhido de forma que na duração máxima permitida (4s), uma nota possa passar da altura mais grave até a mais aguda permitida.

Se este efeito for aplicado de forma que durante a duração da nota ultrapasse a frequência limite (22050Hz), ocorre *aliasing* e o efeito parecerá ter sido invertido (diminuição da altura).

3.2.2 Slide-Down

Este efeito é o caso contrário do anterior e foi projetado de forma equivalente. Da mesma forma, também sofrerá *aliasing* se passar do limite inferior (0Hz).

```
function Slide(app)
    %SlideUp:
    app.slideUp = 1.+app.variacaoSU.*app.t;

    %SlideDown:
    app.slideDn = 1.-app.variacaoSD.*app.t;
end
```

3.2.3 Vibrato

Este efeito consiste em uma flutuação periódica (senóide) na altura da nota ao longo de sua duração. A *intensidade* determina o tamanho do deslocamento na altura por flutuação, e a *velocidade* determina a quantidade de flutuações por segundo.

```
function Vibrato(app)
    %Vibrato:
    app.vibrato =
    app.intensidadeV.*sin((2*pi.*app.velocidadeV).*app.t)
        ;
end
```

Se a velocidade for muito alta o efeito de "flutuação"deixa de ser percebido e a nota passa a ser distorcida devido a baixa definição temporal da altura.

Devido a forma como o *vibrato* é implementado no código, ele é efetivamente calculado apenas no *Módulo de Timbre* (ver seccção 3.3).

Os limites dos parâmetros foram escolhidos de forma que a máxima variação de altura na nota de referência (Lá4), não ultrapasse os limites inferior ou superior (evitar aliasing), e a região de máxima velocidade não possui diferenças marcantes entre valores próximos (saturação do efeito).

3.2.4 Escalator

Este efeito combina o anterior com *Slide-Up* para implementar uma *flutuação crescente*, porém apenas com parâmetros do mesmo tipo do efeito de vibrato, ou seja, a taxa de crescimento da flutuação não é um parâmetro diretamente controlável, e fica dependente dos outros dois (intensidade, velocidade).

```
function Escalator(app)
    app.escalator =
    1+app.intensidadeE*sin((2*pi*app.velocidadeE).*app.t)
        ;
end
```

Este efeito também apresenta distorção se a velocidade estiver alta, e diferente dos *slides*, o *aliasing* é desejado como uma característica integrante deste efeito.

Ao receber o gatilho de uma tecla (digital ou teclado do computador), o número correspondente aquela tecla será associado a escala e o Módulo de Frequência irá chamar os efeitos empregados (com exceção do *vibrato*):

```
function DefinirFrequencia(app, valueNota)
    app.frequencia =
    app.escala(valueNota).*app.pitch.*app.slideUp
    .*app.slideDn.*app.escalator;
end
```

3.3 Módulo de Timbre

Este módulo é responsável por selecionar uma voz dentre exemplos pré-selecionados, ou criar uma voz através do controle direto do nível relativo de intensidade entre os harmônicos. Desta forma, cada botão do tipo [harmônico-n] varia de 0 a 1 e correspondem respectivamente a fundamental (1), e seus 9 harmônicos consecutivos.

$$\sum_{n=1}^{10} \text{harmonico}(n) * sen(2\pi * n * \text{fundamental} * (t + \text{vibrato})) \qquad (1)$$

Diferente do *escalator*, o *vibrato* não é aplicado diretamente ao cálculo da frequência, mas ao cálculo temporal de cada senoide (timbre).

3.4 Módulo de Dinâmica

Este módulo determina a variação de intensidade da nota ao longo de sua duração, para isso, foram implementados um botão de *volume* geral, um submódulo de *envelope ASR*, e 3 efeitos.

3.4.1 Envelope ASR

O envelope implementado neste sintetizador não possui a porção de *Decay*, pois além de ser menos expressivo do que as outras porções, também seria necessário especificar o excesso de energia do pico, e para isso mais uma normalização seria necessária para manter todas as amostras de som dentro dos valores aceitáveis para a interface de som, assim evitando saturação. Os períodos de *attack* e *release* são implementados de forma *exponencial* para representar as propriedades elásticas das vozes.

Os botões de *attack* e *release* variam de 0 a 1 e determinam diretamente a porcentagem de tempo da duração em que a nota se encontra nestes estados. Estes botões estão diretamente relacionados, de forma que a soma máxima permitida sempre 1, ou seja, ao colocar um botão em 0,2, o outro será limitado até 0,8.

```
app.fA = 1.-exp(1).^((-5/app.att) .*app.A); %5 constantes
    de tempo
app.fS = app.S./app.S; %vetor de uns;
app.fR = exp(1).^((-5/app.rel) .*app.R); %5 constantes de
    tempo

app.envelope = [app.fA app.fS app.fR];
```

3.4.2 Crescendo

Assim como *slide-up*, o *crescendo* constitui em um aumento linear do volume da nota ao longo da duração. O respectivo botão regula a velocidade desse aumento.

```
function EfeitoCrescendo(app)
    app.cres = 1.+app.variacaoC.*app.t;
end
```

Se este efeito for empregado de forma que o volume total ultrapasse o limite da interface de som ([-1,1]), haverá *saturação* da nota, prejudicando a qualidade do timbre.

3.5 Diminuendo

De forma análoga ao *slide-down* este efeito também é contrário ao *crescendo* e seu botão também regula a velocidade da variação, porém, neste caso, o efeito

é implementado de forma exponencial assim como *Release*, pois se fosse implementado de forma linear, ao atingir o limite inferior (0), o efeito passaria a se comportar como *crescendo*.

```
function EfeitoDiminuendo(app)
    app.dim = exp(-0.5756*app.variacaoD*app.t);
end
```

3.5.1 Tremolo

Análogo ao *vibrato*, este efeito consiste em uma variação periódica do volume, e possui dois botões para regular a *intensidade* e *velocidade* dessa variação.

Diferente do vibrato, este efeito possui mais um controle chamado de *offset* que controla o nível mínimo do decaimento da intensidade, ou seja, uma variação de 0 a 0,5 pode receber um offset de 0.5 e variar de 0,5 a 1.

```
function Tremolo(app)
    %Tremolo:
    app.tremo = (1-app.offseT/(2*app.volume))
    +(app.intensidadeT/(2*app.volume))*sin((2*pi*app.
        velocidadeT).*app.t);
end
```

Assim como o efeito de *crescendo*, aplicar este efeito a volumes muito altos pode fazer com que a nota entre na região de *saturação* periodicamente.

O Módulo de Dinâmica é capaz de aplicar todos os efeitos ao mesmo tempo:

```
app.dinamica = app.volume*app.envelope.*app.tremo.*app.
    cres.*app.dim;
```

3.6 Ruído Branco

O sintetizador possuí um controle de ruído branco aditivo implementado direto à saída de som com intensidade na mesma faixa do *volume*.

```
function DefinirRuido(app)
    app.ruido = app.levelN.*randn(1,length(app.t));
end
```

4 Interface

A interface do sintetizador digital foi construída através de um recurso nativo no *Matlab 2019a* chamado *App Designer*. O mesmo possibilita a criação de aplicativos ou os conhecidos como *toolbox* de interface gráfica de forma interativa e visual. Este recurso possui uma série de componentes nativos que facilitam a criação de uma interface de usuário, entre eles: *Axes*, *button*, *slider*, *dropdown*, entre outros.

4.1 Teclas

As teclas da interface são componentes *button* presentes nativamente no recurso *App Designer*.

Figura 3: Interface das *teclas*

Quando um desses botões é apertado, ele irá acionar um *callback* do *Matlab*, que consiste em uma função assíncrona.

O botão A por exemplo, irá chamar a seguinte função:

```
function AButtonPushed(app, event)
    PressKey(app,'a');
end
```

Então essa função *PressKey()* irá chamar a função que irá determinar qual nota tocar, a cor do botão após clicar e por quanto tempo ele ficará com essa cor:

```
function PressKey(app, key)
    center = 49;
    timerStyle = 0.2;
    color = [0.8,0.8,0.8];
    switch(key)
        case 'a'
            PlaySound(app,center-10)
            app.AButton.BackgroundColor = color;
            pause(timerStyle);
            app.AButton.BackgroundColor =
                [0.96,0.96,0.96];
        end
        ...
    end
end
```

Então ele determina a nota e chama a função que irá prepará-la e tocá-la:

```
function PlaySound(app,valueNota)
    Nota(app, valueNota);
    sound(app.nota,app.taxa,app.resolucao)
end
```

Já a nota é determinada pela função a seguir que define a frequência da nota e o timbre a ser tocado, para então armazenar a nota que será tocada pelo *PlaySound()*:

```
function Nota(app, valueNota)
    % Define a frequencia da nota
    DefinirFrequencia(app, valueNota);

    % Define o timbre a partir dos harmonicos
    Timbre(app, app.frequencia);

    % Define a nota que sera tocada
    app.nota = app.dinamica.*app.timbre + app.ruido;
end
```

4.2 Knobs

Os *knobs* também são componentes nativos do *App Designer* que foram usados para diversas funções dentro da nossa interface, tais como regular volume, tempo, efeitos e harmônicos.

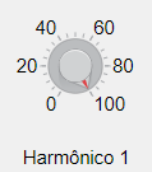

Figura 4: Exemplo de *knob*

Muito similar as teclas, o knob irá chamar um *callback* com um evento do tipo *ValueChanged*, que permite chamar uma função assíncrona quando o valor do *knob* é alterado, essas funções deverão então armazenar os valores e atualizar as notas e timbres pela função *ParameterUpdate()* que serão tocados.

```
function ParameterUpdate(app, typeWave)
    app.duracaoInit = 0;

    % Atualiza o tempo da nota
    DefinirTempo(app)

    % Define as escalas das notas
    Escalas(app)

    % Aplica os efeitos
    Efeitos(app)
```

```
        % Cria a sintese dos harmonicos
        SinteseAditiva(app,typeWave)
        SinteseSubtrativa(app)

        % Adiciona ruido ao sinal
        DefinirRuido(app);

        % Define os multiplicadores das notas
        app.dinamica = app.volume*app.envelope.*app.tremo
            .*app.cres.*app.dim;

        % Define a nota para plotar
        Nota(app,49)

        % Plota o sinal da nota no componente axes
        plot(app.UIAxes,app.t,app.nota);
    end
```

4.3 Sliders

Os *sliders* tem o mesmo princípio dos *knobs*, sendo componentes nativos do *App Designer* que foram usados para alterar os valores do *Pitch* e dos efeitos *Slide Up* e *Slide Down*.

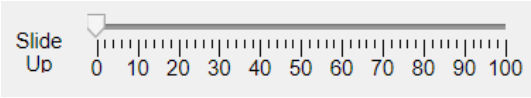

Figura 5: Exemplo de *slider*

O *callback* dos *sliders* são iguais ao dos *knobs*, onde os mesmos irão chamar uma função quando o evento de mudar valor do *slider* for ativado, então os valores para o *Pitch*, *Slide Up* e *Slide Down* serão armazenados, em seguida as notas e timbres serão atualizados pela função *ParameterUpdate()*.

4.4 Gráfico da Nota

O componente que mostra o sinal da nota que deve ser tocada ao apertar uma tecla, é o componente *Axes* que também é um nativo do *App Designer*. Através dele é possível usar o comando de *plot* e similares do *Matlab*, assim como as suas configurações.

Não é necessário usar *callback* para o componente, tendo em vista que o mesmo pode ser atualizado sempre pelo *callback* dos outros componentes presentes na interface.

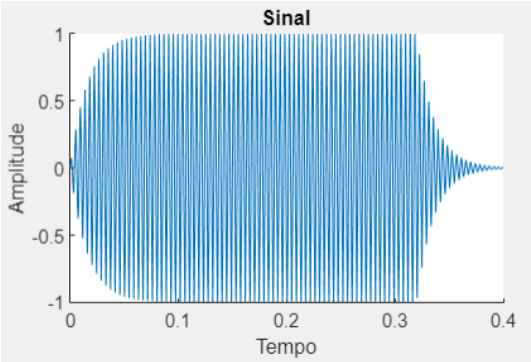

Figura 6: Exemplo de *Axes*

4.5 Menu de Timbres

Para o menu de timbres, foi utilizado o componente *Dropdown*, nativo do *App Designer*. Componente que permite você selecionar apenas um dos itens de uma lista por vez.

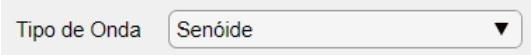

Figura 7: Exemplo de *dropdown*

Assim como em outros componentes, um *callback* é ativado com o evento de mudar o valor do *dropdown*, com isso o mesmo irá chamar uma função que deverá armazenar a escolha de timbre e atualizar o novo timbre pela função *ParameterUpdate()*.

5 Limitações

- Num geral, teclados para PC não possuem *teclas sensitivas* para detectar diferenças na força aplicada, detectando apenas a ativação da tecla.
- Uma tecla pressionada gera gatilhos na CPU até que seja solta, porém ao pressionar mais de uma tecla ao mesmo tempo, apenas uma tecla continua enviando gatilhos.
- As funções *sound* e *soundsc*, responsáveis por enviar um vetor número para a saída de som, ocupam o processamento do programa até o final da instrução, desta forma, não é possível executar outras instruções enquanto o som não for produzido por completo, e por tanto, não é possível antecipar a chamada de outra função *sound*, desta forma, há uma interrupção entre instruções, pois há um delay entre o processamento de uma instrução até o começo da reprodução do som.

- O item anterior implica na impossibilidade de produzir uma nota sustentada por pressionamento de tecla, uma vez que necessitaria de um looping na instrução até que a tecla seja solta, porém a interrupção entre cada loop implica em cliques no som.
- Pelo mesmo motivo, não foi possível fazer alterações no timbre e efeitos *durante* a emissão de uma nota.