ICS - Trabalho III
Efeito Risset Contínuo

Juarez Aires Sampaio Filho 11/0032829

Universidade de Brasília

17 de Junho de 2014

Requisitos

Utilizar as classes do pacote **sintese** para implementar o *Paradoxo de Altura* e desenvolver uma interface gráfica conveniente e útil para a aplicação.

Introdução Teórica 1/5

- Implementou-se a versão contínua do efeito conhecida como *Escala Risset Contínua*.
- O efeito constitui-se de uma soma de senoides na forma:

$$f(t) = \sum_i A_i(t) sin(2\pi f_i(t)) \quad (1)$$

- A relação A(f) deve implementar uma gaussiana em função do logaritmo da frequência conforme a figura a seguir.

Introdução Teórica 2/5

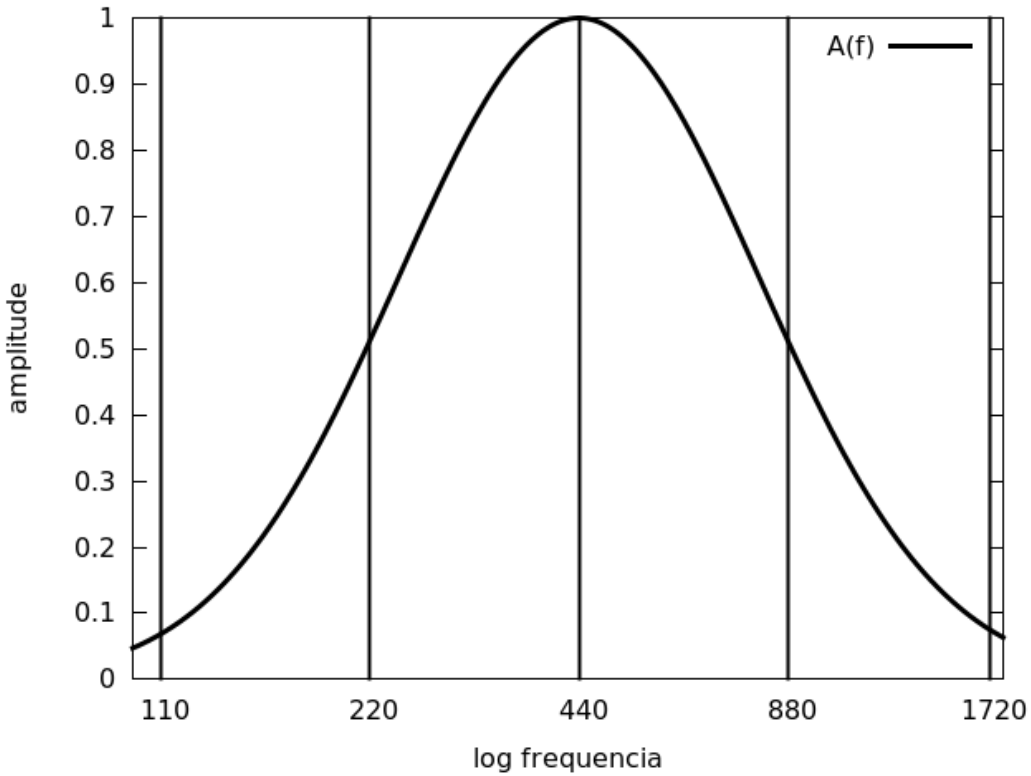

Figura: Relação Axf

Introdução Teórica 3/5

- Escolhemos a função $f_i(t)$ de tal forma que ela caia a metade no instante T_0.
- Escolhendo $f_{i_0} = f_0 2^i$, temos que $f_i(T_0) = f_{i-1}(0)$
- Finalmente:

$$f_i(t) = f_{i_0} \cdot 2^{-\frac{t}{T_0}} \tag{2}$$

Introdução Teórica 4/5

- a Gaussiana logaritma é dada por:

$$A_i(t) = exp(-\frac{(\log f_i(t) - \log f_c)^2}{(2\sigma^2)}) \quad (3)$$

- substituindo:

$$A_i(t) = exp(-\frac{(\log(f_{i_0} 2^{-\frac{t}{T_0}}) - \log f_c)^2}{(2\sigma^2)}) \quad (4)$$

- juntando tudo, o efeito tem a forma:

$$f(t) = \sum_i e^{-\frac{(\log(f_0 2^i 2^{-\frac{t}{T_0}}) - \log f_c)^2}{(2\sigma^2)}} \cdot \sin(2\pi f_0 2^i \cdot 2^{-\frac{t}{T_0}}) \quad (5)$$

Introdução Teórica 5/5

- Se quisermos o efeito crescente, muda-se apenas o sinal do expoente:

$$f(t) = \sum_i e^{-\frac{(\log(f_0 2^i 2^{\frac{t}{T_0}}) - \log f_c)^2}{(2\sigma^2)}} \cdot \sin(2\pi f_0 2^i \cdot 2^{\frac{t}{T_0}}) \qquad (6)$$

Classes Desenvolvidas

Com base na fórmula geral do efeito, desenvolveu-se:

- EnvoltoriaAmplitudeRisset: implementa um termo $A_i(t)$
- EnvoltoriaFrequenciaRisset: implementa um termo $f_i(t)$
- OsciladorRisset: implementa uma parcela da soma $A_i(t)sin(2\pi f(t))$
- EfeitoRisset: implementa a soma $\sum_i A_i(t)sin(2\pi f(t))$
- GUI: interface gráfica para comandar os parâmetros do EfeitoRisset

EnvoltoriaAmplitudeRisset

- extends Envoltoria
- implementa uma parcela cíclica do efeito Risset definida pelos parâmetros Fi, T0, Fc, Var(σ)
- Reescrevemos os métodos *clock()* e *getSaida()* para fazermos a envoltória cíclica em função do período T_0 e não dá duração.
- Ou seja, digamos que a duração seja de 10s e $T_0 = 2$, teremos então 5 ciclos.
- Na verdade, gostaríamos que essa classe fosse um oscilador, mas como não temos acesso à tabela da classe oscilador, essa foi a forma mais rápida de contornar o problema

EnvoltoriaAmplitudeRisset 1/4

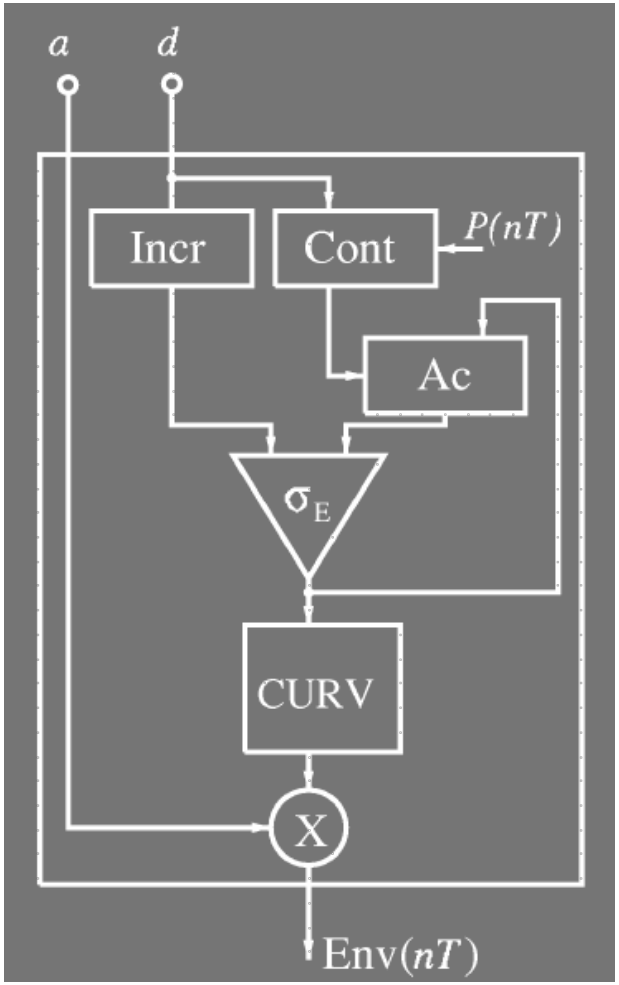

Figura: Gerador de Envoltória

EnvoltoriaAmplitudeRisset 2/4

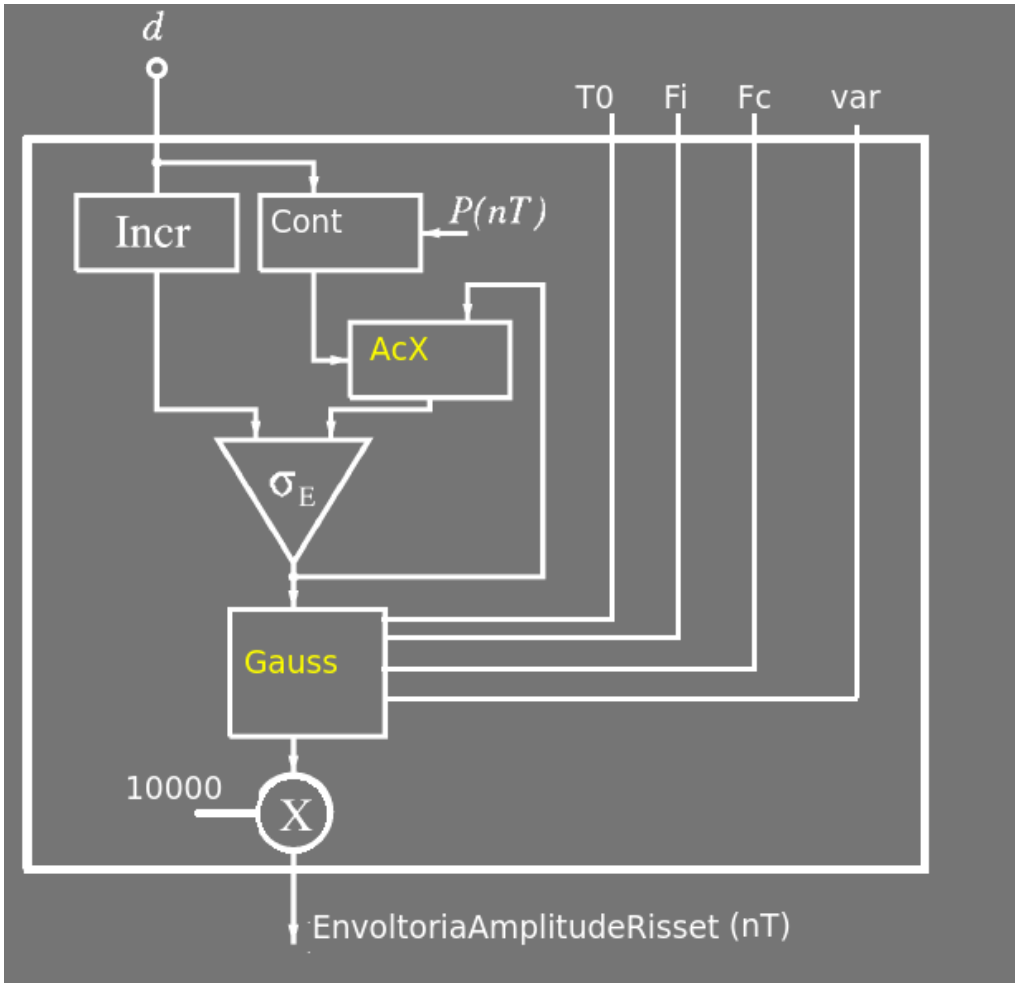

Figura: Esquemático da Construção

EnvoltoriaAmplitudeRisset 3/4

```
private void set(float Fi, float T0, float Fc, float var, boolean crescente){
    Curva curva = new Curva(720);
    for(int i = 0; i<= 720; i++){
        double t = T0*i/720.0;
        double a;
        if(crescente == false)
            a = 10000*Math.exp(-1.0*Math.pow(Math.log10(Fi*Math.pow(2, -1*t/T0))  - Math.log10(Fc), 2)/(2*var*var));
        else
            a = 10000*Math.exp(-1.0*Math.pow(Math.log10(Fi*Math.pow(2, +1*t/T0))  - Math.log10(Fc), 2)/(2*var*var));

        curva.addPonto(i, a);
    }

    setCURVA(curva);
    setSR(44100);

    fContador = 0;
    fFi = Fi;
    fT0 = T0;
    fFc = Fc;
    fVar = var;

    if(crescente == false)
        bCrescente = false;
    else
        bCrescente = true;

}
```

Figura: Construção de EnvoltoriaAmplitudeRisset

EnvoltoriaAmplitudeRisset 4/4

```
/**
 * Reescrevemos o método para controlar a saída e permitir uma envoltória cíclica.
 */
public void clock(){
    super.clock();

    fContador += 721.0/(fT0*44100.0);
}

/**
 * Reescrevemos o método para permitir uma envoltória de periodo definido por T0.
 */
public float getSaida(){
    float saida = getCURVA().getValorNoIndice(fContador);
    if(fContador >= 720.0)
        fContador = 0;

    return saida;
}
```

Figura: Reescrevemos os métodos de clock e getSaida

EnvoltoriaFrequenciaRisset 1/3

- extends Envoltoria
- implementa uma parcela cíclica do efeito Risset definida pelos parâmetros Fi, T0
- Novamente, reescrevemos os métodos *clock()* e *getSaida()*

EnvoltoriaFrequenciaRisset 2/3

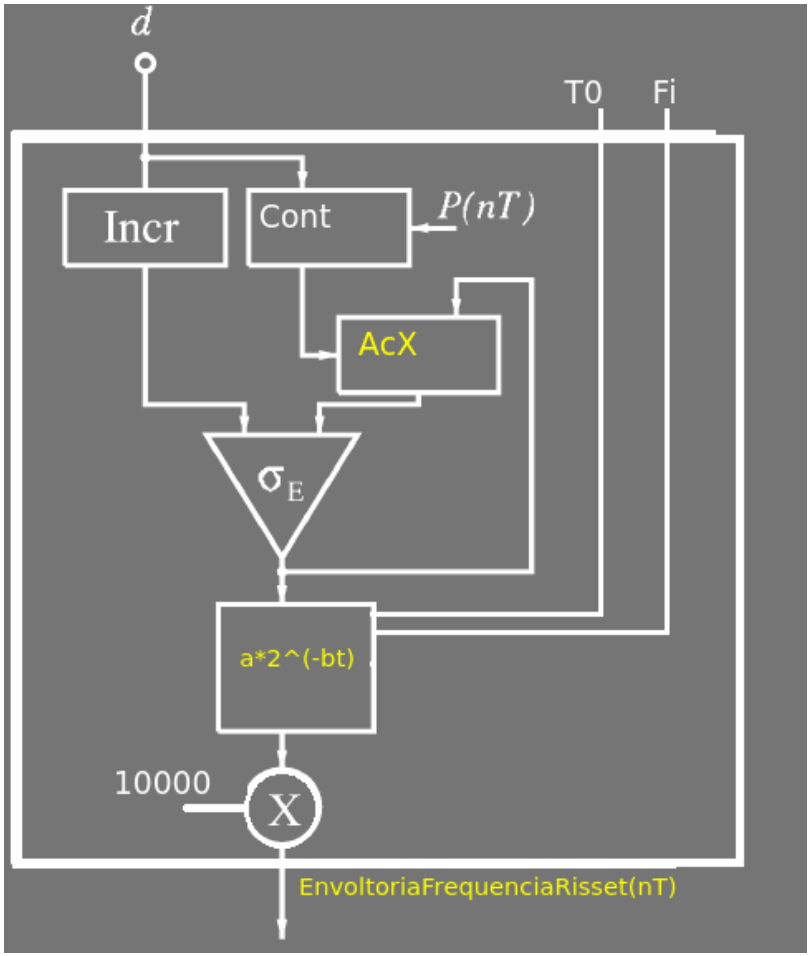

Figura: Esquemático da Construção

EnvoltoriaFrequenciaRisset 3/3

```
private void set(float Fi, float T0, boolean crescente){
    Curva curva = new Curva(720);
    for(int i = 0; i<= 720; i++){
        double t = T0*i/720.0;
        //double f = Fi*T0/Math.log10(2)*(1.0 - Math.pow(2, -1.0*t/T0));
        double f;
        if(crescente == false)
            f = Fi*Math.pow(2, -1.0*t/T0);
        else
            f = Fi*Math.pow(2, +1.0*t/T0);

        curva.addPonto(i, f);
    }

    setCURVA(curva);
    setSR(44100);

    fFi = Fi;
    fT0 = T0;
    fContador = 0;

    if(crescente == false)
        bCrescente = false;
    else
        bCrescente = true;
}
```

Figura: Construção de EnvoltoriaFrequenciaRisset

OsciladorRissetContinuo 1/4

- extends Oscilador
- faz a conexão das últimas duas classes para formar um termo cíclico do efeito Risset

OsciladorRissetContinuo 2/4

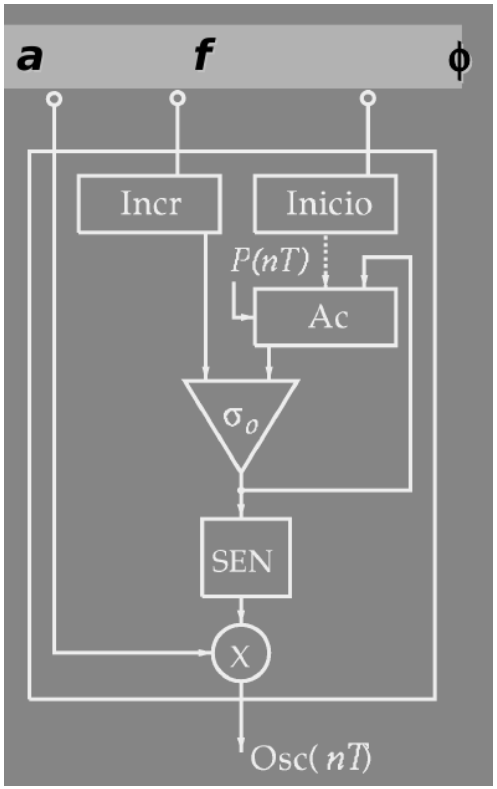

Figura: Esquemático do oscilador padrão

OsciladorRissetContinuo 3/4

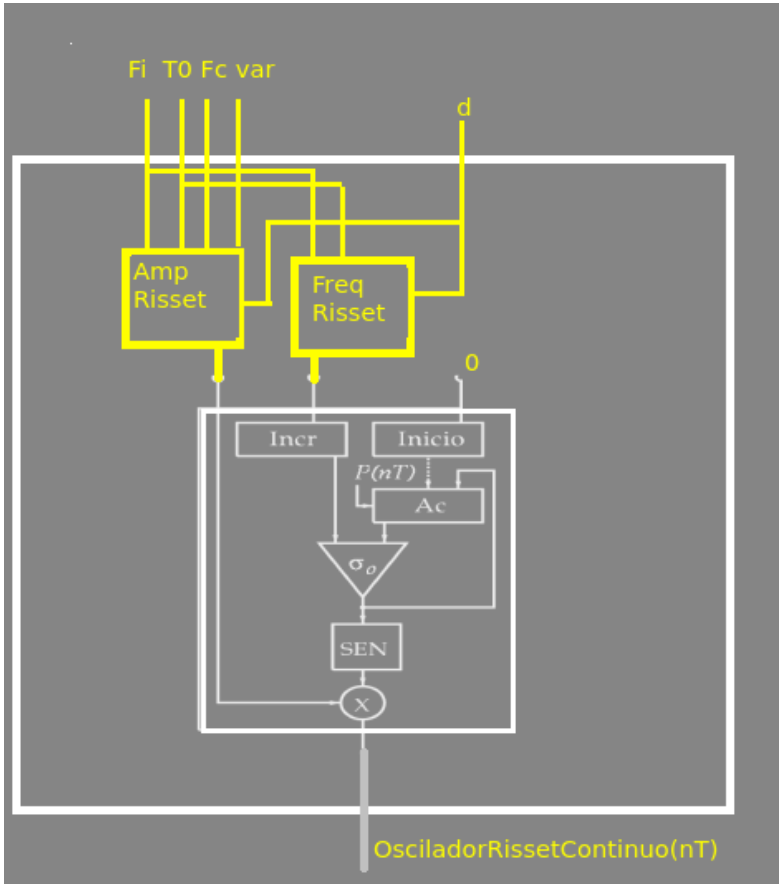

Figura: Esquemático do osciladot Risset

OsciladorRissetContinuo 4/4

```
private void set(float Fi, float T0, float Fc, float var){
    amp_env = new EnvoltoriaAmplitudeRisset(Fi, T0, Fc, var);
    freq_env = new EnvoltoriaFrequenciaRisset(Fi, T0);

    setDispositivoAmplitude(amp_env);
    setDispositivoFrequencia(freq_env);

    fFi = Fi;
    fT0 = T0;
    fFc = Fc;
    fVar = var;
}
```

Figura: Construção de OsciladorRissetContinuo

EfeitoRissetContinuo 1/2

- Conecta vários OsciladoresRissetContinuo definindo as frequências como potências de dois de uma frequência base.
- possui métodos para configurar todos os parâmetros do efeito e um método para obter o som resultante.
- na implementação utilizamos 9 harmônicos, começando com Fc/16, até Fc e então até 16Fc

EfeitoRissetContinuo 2/2

```
private void set(float T0, float Fc, float var, float duracao, boolean crescente){
    fVar = var;
    fFc = Fc;
    fT0 = T0;
    fF0 = fFc/16.0f;
    fDuracao = duracao;
    bCrescente = crescente;

    for(int i = 0; i < 9; i++){
        float Fi = fF0*((float )Math.pow(2.0, i));
        osci[i] = new OsciladorRissetContinuo(Fi, fT0, fFc, fVar);
    }

    for(int i = 0; i < 9; i++){
        osci[i].setCrescente(bCrescente);
    }


    sum[0] = new Somador(osci[0], osci[1]);
    for(int i = 1; i < 8; i++){
        sum[i] = new Somador(sum[i-1], osci[i+1]);
    }

    Curva curva = new Curva(720);
```

Interface Gráfica

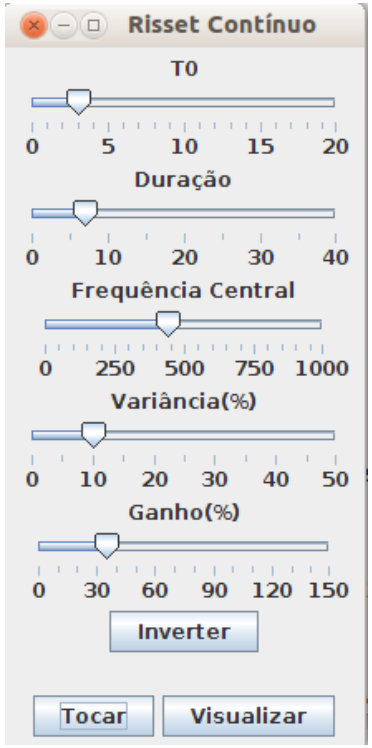

Figura: Interface Desenvolvida

Documentação e Referências

- Documentação produzida com javadoc
- Documentação da API sintese
- CIRCULARITY IN PITCH JUDGEMENT, MIT, Introduction to Computational Neuroscience