Centro Universitário SENAI CIMATEC
Curso de Bacharelado em Engenharia Elétrica

Algoritmo para correção de tom por meio da análise em frequência

Cleber Couto Filho
Ícaro Nascimento Queiroz

Salvador-BA, 19 de outubro de 2018

Cleber Couto Filho
Ícaro Nascimento Queiroz

Algoritmo para correção de tom por meio da análise em frequência

Relatório apresentado como requisito parcial para obtenção de aprovação na disciplina Processamento Digital de Sinais, no centro universitário SENAI CIMATEC.

Docente: Leonardo Vasconsellos

Centro Universitário SENAI CIMATEC

Salvador-BA
19 de outubro de 2018

Lista de ilustrações

Introdução

Este relatório descreve o método utilizado para o projeto de um sistema de rádio enlace ponto a ponto, com base nos dois pontos fornecidos.

Segundo TUDE, o enlace de rádio pode ser definido como : "Uma aplicação da transmissão de informação por meio de ondas eletromagnéticas, se caracterizando como uma das aplicação que faz parte das Segundo Tude , “Um enlace rádio digital ponto a ponto é utilizado para o transporte de informação entre dois pontos fixos,tendo o espaço livre como meio de transmissão (wireless)”.[1]

Proposta

Projetar um sistema de rádio enlace ponto a ponto, entre os pontos mostrados nos mapas anexados.

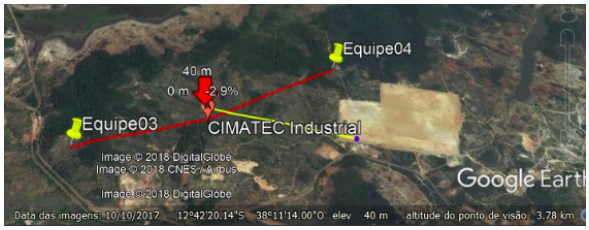

Figura 1 – Ponto A

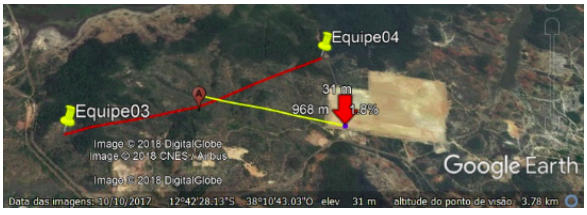

Figura 2 – Ponto B

Foi solicitado que sejam tomadas as seguintes decisões:

- Definir a frequência de operação;
- Calcular a Potência do Rádio Transmissor, Sensibilidade do Receptor e ganhos das Antenas, para o enlace fornecido;
- Definir solução a ser adotada com base em fornecedores comerciais reais (Ex: Intelbras, Ubiquiti, Tp Link, MikroTik, entre outros)

1 Medotologia

1.1 Escolha da frequência

A frequência foi escolhida com base no regulamento da Anatel. O mesmo estabelece que equipamentos de radiocomunicação com faixas restritas de: 902-907,5; 915-928; 2400-2483,5; 5725-5850 MHz, assim para os cálculos iniciais foi escolhida uma frequência de 920MHz.

1.2 Estudo do Obstáculo

O primeiro passado adotado foi a análise dos pontos, verificando se entre os dois existia algum obstáculo. Essa informação pode ser extraída dos mapas que foram fornecidos em anexo, como mostra a figura 3.

Figura 3 – Gráfico de obstáculo

Por meio da análise do gráfico, fica evidenciado que entre os dois pontos existe um grande obstáculo arredondado, assim sendo necessário calcular o seu raio de curvatura e a partir disso ver o quanto ele irá interferir na transmissão. É feita uma aproximação do topo do obstáculo, utilizando uma curvatura parabólica como mostrado na figura 4 ??.

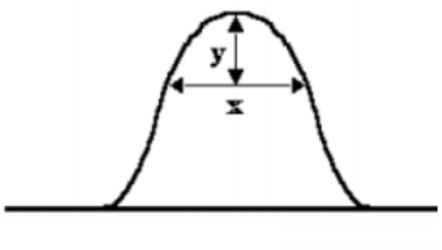

Figura 4 – Curvatura parabólica utilizada para aproximação

O raio r da parábola será calculado o α que possibilita encontrar quantos decibéis de interferência o obstáculo irá causar. O cálculo do raio r é feito utilizando a seguinte

formula:

$$r = \frac{x^2}{8y} * 10^{-3} \tag{1.1}$$

Onde x representa a distância em metros entre os dois pontos de igual nível, sendo um em cada lado do pico considerado e y representa a diferença de cota entre o pico do obstáculo e a curva de nível considerada para medida x.

1.3 Atenuação do Obstáculo

O cálculo do fator α relaciona a frequência f, o raio de curvatura da parábola r, a distância entre o vértice do obstáculo ao ponto de transmissão d_1 e a distância entre o vértice do obstáculo ao ponto de recepção d_2, ambas em *Km*.

$$\alpha = 0,0818 \frac{1}{\sqrt[6]{f}} \sqrt[3]{r} \sqrt{\frac{d_1 + d_2}{d_1 * d_2}} \tag{1.2}$$

A atenuação é encontrada a partir de α e a relação entre os fatores H_c e r_f por meio do gráfico 6. Onde H_c representa a diferença entre ponto máximo do obstáculo e o nivelamento das antenas, enquanto r_f é chamado de raio de *fresnel*, sendo calculado com as mesmas distâncias d_1 e d_2. A relação é encontrada como mostrado na figura 6.

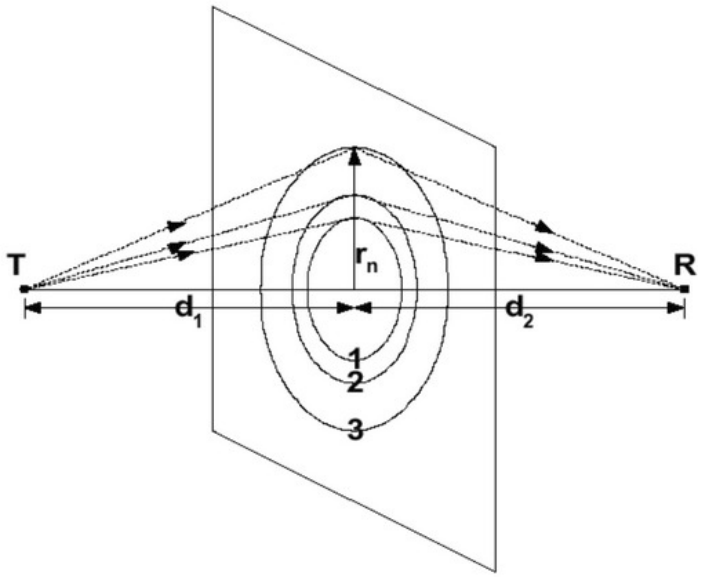

Figura 5 – Modelo da zona de *Fresnel*

O cálculo de r_f se dá por:

$$r_f = \sqrt{\frac{n\lambda d_1 d_2}{d_1 + d_2}} \tag{1.3}$$

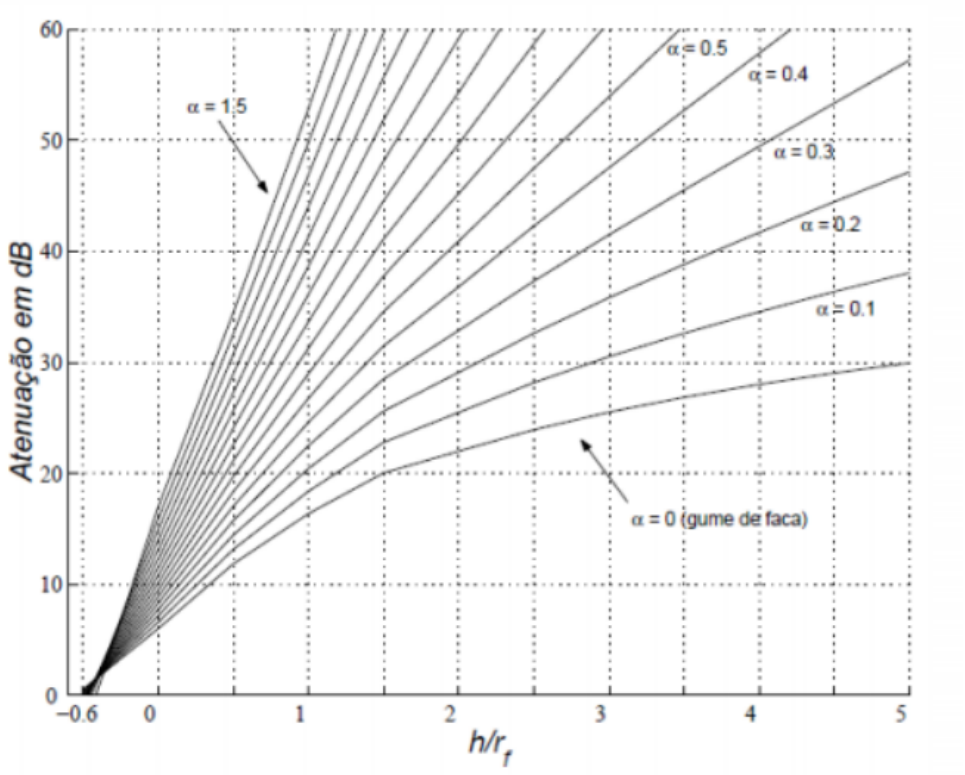

Figura 6 – Gráfico de atenuação

Rf	8,866
Hc	5
Hc/rf	0,563

1.4 Atenuação no espaço livre

O sinal também sofrerá atenuação ao ser transmitido no espaço livre, o cálculo da atenuação L se dá por meio da fórmula de *Fris*:

$$L = 32,45 + 20(\log_{10}(d_1 + d_2) + \log_{10} f) \tag{1.4}$$

1.5 Atenuação total

A atenuação total é a soma da atenuação no espaço livre com a atenuação do obstáculo, assim :

$$L_{tot} = L + L_{obstaculo} \tag{1.5}$$

2 Resultados

2.1 Dados de entrada

2.1.1 Dados Gerais

A altura escolhida para as torres foi de 0 e 9 para manter a linha de visada direta em paralelo com o chao e assim facilitando os cálculos e diminuindo os erros, os outros dados estão mostrados na tabela abaixo.

Distância Total (m)	0,96713
λ	0,001086957 x 10^{-6}
Altura das torres	0/9

2.1.2 Dados dos Obstáculos

X	325
Y	7
d_1	0,475
d_2	0,49213

2.2 Memorial de Cálculo

2.2.1 Frequência

Não foi necessário calcular a frequência, o valor utilizado foi de 920MHz e sua escolha está justificada em 1.1.

2.2.2 Raio da Parábola

Com os valores de X e Y do obstáculo o valor encontrado para o raio da parábola foi de:

$$r_{parabola} = 1,886160714 \tag{2.1}$$

2.2.3 Atenuação do obstáculo

Com os valores de d_1 e d_2 o α encontrado teve valor de :

$$\alpha = 0,6703932617 \tag{2.2}$$

O parâmetro H_c foi encontrado por meio dos dados fornecidos, como mostrado na figura a seguir:

O parâmetro r_f foi cálculado por meio da equação de $Fresnel$, com o valor dos parâmetros a seguir, foi encontrado no gráfico o valor da atenuação do obstáculo.

r_f	8,866
H_c	5
$\frac{H_c}{r_f}$	0,563
$L_{obstaculo}$	$22,5dB$

2.2.4 Atenuação no Espaço livre

Com os valores de d_1, d_2 e f conhecidos o valor encontrado para L foi:

$$L = 91,43dB \tag{2.3}$$

2.2.5 Atenuação total

A atenuação total consiste nas soma das atenuação encontradas, logo:

$$L_{tot} = L + L_{obstaculo} = 91,43 + 22,5 = 113,9dB \tag{2.4}$$

2.3 Escolha do tranmissor e da antena

A escolha dos módulos e da antena se deu baseado na frequência escolhida para os cálculos e na versatilidade de cada dispositivo.

O modelo das antenas foi o *Yagi(AirMax Antenna 900Mhz)*,devido a sua faixa de trabalho e alta potência.

O modelo do transmissor foi o *Ubiquiti Networks(Rocket M9)* pot ser recomendado para trabalhar em conjunto com o modelo de antena *Yagi*.

2.4 Receptor

Escolha do receptor é encontrado à partir das potências das antenas, do módulo tranmissor e das perdas durante a transmissão. Sua potência foi calculada da seguinte forma:

$$P_{receptor} = P_{antena_tx} + P_{antena_rx} + P_{transmissor} - L_{tot} \tag{2.5}$$

A mesma antena utilizada para transmissão é utilizada para recepção, os dados da sua potência estão disponíveis nos seu *datasheet* onde $P_{antena_{tx}} = P_{antena_{rx}} = 19dBi$. A

potência do transmissõr também está disponível no datasheet, onde $P_{transmissor} = 28dBm$. O valor encontrado para potência do receptor foi de $P_{receptor} = -47.9dBi$

Com o valor de $-47.9dBi$, o módulo *Ubiquiti Networks(Rocket M9)* também poderá ser utilizado para recepção, tornando o sistema mais simplificado já que ambos receptores e transmissores estarão utilizando antenas recomendadas no datasheet.

Variáveis do projeto

Todos os valores utilizados e calculados estão registrados na figura a seguir:

Localização A			Escolha da antena	
Latitude	12° 42' 20.14" S			
Longitude	38° 11' 14" O		Transmissor	
Elevação (m)	40		Modelo	Yagi airMAX
Localização B			potencia dBi	19
Latitude	12° 42' 28.13" S			
Longitude	38° 10' 43.03" O		Receptor	
Elevação (m)	31		Modelo	Yagi airMAX
Distancia D (m)	0,96713		potencia dBi	19
Torre A (m)	0			
Torre B (m)	9		Modulo	
Frequencia	920			
			Transmissor	
X obstaculo (m)	325		Modelo	m900 rocket
Y obstaculo (m)	7		potencia dBm	28
R Obstáculo	1,886160714			
D1 (km)	0,475		RECEPTOR	
D2 (km)	0,49213		Modelo	m9 rocket
alfa	0,6703932617		potencia dBm	-47,93545365
Rf	8,866203803			
Altura visada direta (m	40			
Altura obstaculo (m)	45			
Hc (m)	5			
h/rf	0,5639392135			
Atuanação Obstaculo	22,5			
Atenuação espaço liv	91,43545365			
Atenuação total (db)	113,9354536			

Figura 7 – Variáveis do projeto

Referências

[1] TUDE, Eduardo. Enlace rádio digital ponto a ponto. 2004.

ANEXO

Função GravaAudio

```
function audio = gravaAudio(fs,duration)
%Cria um objeto do tipo audio recorder com uma frequencia de amostragem fs, 16 bit, canal mono
recorder = audiorecorder(fs,16,1);
 record(recorder);
 pause(duration);

 audio = getaudiodata(recorder);
end
```

Função istft

```
%%Downloaded from http://www.ee.columbia.edu/~dpwe/e4810/matlab/s24/istft.m

function x = istft(d, ftsize, w, h)
% X = istft(D, F, W, H)                    Inverse short-time Fourier transform.
%       Performs overlap-add resynthesis from the short-time Fourier transform
%       data in D.  Each column of D is taken as the result of an F-point
%       fft; each successive frame was offset by H points (default
%       W/2, or F/2 if W==0). Data is hann-windowed at W pts, or
%       W = 0 gives a rectangular window (default);
%       W as a vector uses that as window.
%       This version scales the output so the loop gain is 1.0 for
%       either hann-win an-syn with 25% overlap, or hann-win on
%       analysis and rect-win (W=0) on synthesis with 50% overlap.
% dpwe 1994may24.  Uses built-in 'ifft' etc.
% $Header: /home/empire6/dpwe/public_html/resources/matlab/pvoc/RCS/istft.m,v 1.5 2010/08/12 20:39:42 dpwe Exp $

if nargin < 2; ftsize = 2*(size(d,1)-1); end
```

```
18 if nargin < 3; w = 0; end
19 if nargin < 4; h = 0; end  % will become winlen/2 later
20
21 s = size(d);
22 if s(1) ~= (ftsize/2)+1
23   error('number of rows should be fftsize/2+1')
24 end
25 cols = s(2);
26
27 if length(w) == 1
28   if w == 0
29     % special case: rectangular window
30     win = ones(1,ftsize);
31   else
32     if rem(w, 2) == 0   % force window to be odd-len
33       w = w + 1;
34     end
35     halflen = (w-1)/2;
36     halff = ftsize/2;
37     halfwin = 0.5 * ( 1 + cos( pi * (0:halflen)/halflen));
38     win = zeros(1, ftsize);
39     acthalflen = min(halff, halflen);
40     win((halff+1):(halff+acthalflen)) = halfwin(1:acthalflen);
41     win((halff+1):-1:(halff-acthalflen+2)) = halfwin(1:
         acthalflen);
42     % 2009-01-06: Make stft-istft loop be identity for 25% hop
43     win = 2/3*win;
44   end
45 else
46   win = w;
47 end
48
49 w = length(win);
50 % now can set default hop
51 if h == 0
52   h = floor(w/2);
53 end
54
55 xlen = ftsize + (cols-1)*h;
```

```
x = zeros(1,xlen);

for b = 0:h:(h*(cols-1))
  ft = d(:,1+b/h)';
  ft = [ft, conj(ft([((ftsize/2)):-1:2]))];
  px = real(ifft(ft));
  x((b+1):(b+ftsize)) = x((b+1):(b+ftsize))+px.*win;
end;
end
```

Função mostraEspectrograma

```
function Y = mostraEspectrograma(S,F,T)
%%Mostra o spectrograma do sinal S, recebendo o vetor de
   frequencia F e o vetor de tempo T

X = abs(S);
imagesc(T,F,S);
colorbar;
axis xy
xlabel('Time (s)')
ylabel('Freq (Hz)')

end
```

Função tabelaDoMaior

```
function pitches = tabelaDoMaior()
%%Retorna todas as frequencias da escala de Do maior

pitches = [16.352 18.354 20.602 21.827 24.500 27.500 30.868
   32.703 36.708 41.203 43.654 48.999 55.000 61.735 65.406
   73.416 82.407 87.307 97.999 110.000 123.470 130.800 146.830
   164.810 174.610 196.000 220.000 246.940 261.630 293.660
   329.630 349.230 392.000 440.000 493.880 523.250 587.330
   659.260 698.460 783.990 880.000 987.770 1046.500 1174.700
   1318.500 1396.900 1568.000 1760.000 1975.500 2093.000
   2349.300 2637.000 2793.800 3136.000 3520.000 3951.100
   4186.000 4698.600 5274.000 5587.700 6271.900 7040.000
```

```
    7902.100 8372.000 9397.300 10548.100 11175.300 12543.900
    14080.000 15804.300 16744.000 18794.500 21096.200 22350.600];

end
```

Função compareToPitches

```
%%A funcao recebe um tom e a tabela contadndo a escola de tons,
    retornando o tom mais proximo da escala
function exactpitch = compareToPitches(pitch, pitchtable)

%%Organiza a tabela em ordem crescente
sortedpitchtable = sort(pitchtable);

%%Cria vetor de zeros
midpitchtable = zeros(1,length(sortedpitchtable)-1);

%%Variavel que indica se o tom mais proximo foi encontrado
found = 0;

%%Gera uma tabela de tons na media dos tons da tabela de tons
    para tornar a comparacao
%%mais facil.
%%log2(0.5) e utilizado 0.5 afim de contabilizar para percepcao
    do som
for i = 1:(length(sortedpitchtable)-1)
    midpitchtable(i) = (sortedpitchtable(i+1) - sortedpitchtable
        (i))*log2(0.5) + sortedpitchtable(i);
end

%%Caso a frequencia seja menor que o menor tom da tabela, se
    retorna o menor tom
if (pitch <= midpitchtable(1))
    exactpitch = sortedpitchtable(1);
    found = 1;
end

%%Se o tom exato nao for encontrado ainda considera o maior tom
    da tabela
```

```
27 if (found==0)
28     if (pitch > midpitchtable(length(midpitchtable)))
29         exactpitch = sortedpitchtable(length(sortedpitchtable));
30         found = 1;
31     end
32 end
33
34 %%Se o tom exato nao foi encontrado ainda, compare o a todos os
       valores
35 %da midpitchtable e a saida corresponde ao tom na tabela de tom
36 i = 1;
37 while(found==0)
38     if(pitch > midpitchtable(i) && pitch <= midpitchtable(i+1))
39         exactpitch = sortedpitchtable(i+1);
40         found=1;
41     end
42     i = i+1;
43 end
44
45 end
```

Função pitchCorrector

```
1 %%A funcao recebe um tom e a tabela contadndo a escola de tons,
      retornando o tom mais proximo da escala
2 function exactpitch = compareToPitches(pitch, pitchtable)
3
4 %%Organiza a tabela em ordem crescente
5 sortedpitchtable = sort(pitchtable);
6
7 %%Cria vetor de zeros
8 midpitchtable = zeros(1,length(sortedpitchtable)-1);
9
10 %%Variavel que indica se o tom mais proximo foi encontrado
11 found = 0;
12
13 %%Gera uma tabela de tons na media dos tons da tabela de tons
      para tornar a comparacao
14 %%mais facil.
```

```
15 %%log2(0.5) e utilizado 0.5 afim de contabilizar para percepcao
       do som
16 for i = 1:(length(sortedpitchtable)-1)
17     midpitchtable(i) = (sortedpitchtable(i+1) - sortedpitchtable
           (i))*log2(0.5) + sortedpitchtable(i);
18 end
19
20 %%Caso a frequencia seja menor que o menor tom da tabela, se
       retorna o menor tom
21 if (pitch <= midpitchtable(1))
22     exactpitch = sortedpitchtable(1);
23     found = 1;
24 end
25
26 %%Se o tom exato nao for encontrado ainda considera o maior tom
       da tabela
27 if (found==0)
28     if (pitch > midpitchtable(length(midpitchtable)))
29         exactpitch = sortedpitchtable(length(sortedpitchtable));
30         found = 1;
31     end
32 end
33
34 %%Se o tom exato nao foi encontrado ainda, compare o a todos os
       valores
35 %da midpitchtable e a saida corresponde ao tom na tabela de tom
36 i = 1;
37 while(found==0)
38     if(pitch > midpitchtable(i) && pitch <= midpitchtable(i+1))
39         exactpitch = sortedpitchtable(i+1);
40         found=1;
41     end
42     i = i+1;
43 end
44
45 end
```