Comunicações

1º Trabalho Práctico

Semestre de Inverno de 2009/2010

Autores:
30896 – Ricardo Canto
31401 – Nuno Cancelo
33595 – Nuno Sousa

Indície

Enunciado.....3
Exercício 1.....4
Alínea A.....4
Alínea B.....5
Alínea C.....6
Alínea D.....7
Exercício 2.....8
Alínea A.....8
Alínea B.....11
Alínea i.....11
Alínea ii.....11
Alínea C.....12
Exercício 3.....15
Alínea A.....15
Alínea i).....15
Alínea ii.....15
Alínea B.....17
Alínea c.....18
Alínea d.....20
Alterações à Versão anterior.....22

Enunciado

1. Escreva funções MATLAB para cumprir os seguintes objectivos:

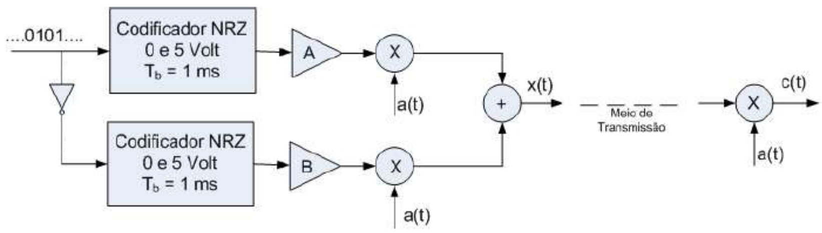
   a) Geração de sinusóides com amplitude, frequência e fase programável: $x_i(t) = A_i \cos(2\pi f_i t + \phi_i)$. Recorrendo a esta função elabore uma pequena aplicação que funcione como um sintetizador de notas musicais (piano).

   b) Efectue testes que permitam aferir a sensibilidade do sistema auditivo humano à variação na amplitude, frequência e fase da sinusóide. Escolha um valor adequado para a frequência de amostragem F_s. Apresente os resultados desses testes e comente-os.

   c) Identifique a funcionalidade da função `chirp` do MATLAB. Aplique a função `analysis` aos sinais produzidos pela função `chirp` e explique o formato dos gráficos obtidos.

   d) Seja o sistema definido por $y[n] = \frac{1}{3}x[n] + \frac{1}{3}x[n-1] + \frac{1}{3}x[n-2]$. Aplique este sistema a sinusóides e aos sinais de áudio disponibilizados com o enunciado. Recorrendo à função `analysis`, bem como a outros testes, identifique o tipo de filtragem realizado por este sistema.

2. Seja o sistema de comunicação digital apresentado na figura em que $a(t) = \cos(2\pi f_o t)$, com f_o =10000 Hz.

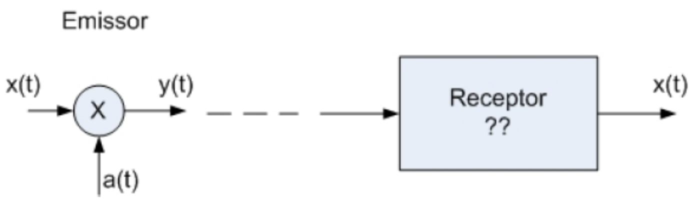


   a) Obtenha as expressões dos sinais $x(t)$ e $c(t)$ em função do valor do bit b que se apresenta à entrada do sistema. Qual o ritmo de transmissão obtido pelo sistema? Sendo $T_o = 1/f_o$, qual a relação entre T_b e T_o?

   b) Atribua valores aos parâmetros do sistema de forma em que $x(t)$ se observe o resultado de: i) modulação OOK; ii) modulação PSK.

   c) Realize o sistema em MATLAB de forma a produzir modulação PSK. Apresente os sinais $x(t)$ e $c(t)$ na codificação da sequência binária `010110010`.

3. Considere o sistema apresentado na figura com $a(t) = \cos(2\pi 10000 t)$.



   a) Esboce o espectro do sinal $y(t)$, considerando duas situações distintas: i) $x(t) = 1 + \cos(2\pi 2000 t)$; ii) $x(t) = \text{sinc}(t)$.

   b) Realize o emissor em MATLAB e confirme os resultados relativos a $y(t)$, recorrendo à função `analysis`.

   c) Projecte e realize o receptor de forma a que seja possível recuperar $x(t)$, na sua saída.

   d) Ilustre o funcionamento do conjunto emissor/receptor, com sinais áudio em formato wave.

Exercício 1

Alínea A

Para este exercício realizamos duas funções:

t1_nota.m

```
%Mais informação nos ficheiros .m .
function [nota,fs] = t1_nota(a,fo,sec,ph)
    if (nargin ~= 4)
        if (nargin == 3)
            ph=0;
        else
            fprintf('Numero de argumentos inválido.\n');
            help t1_notas;
            return;
        end
    end

    %Frequencia de Amostragem (em Hz), taxa equivalente do sinal de telefone
    if (fo>2048)
        fs=2.1*fo;
    else
        fs=4096;
    end

    n=1:1:fs*sec;
    W=2*pi*(fo/fs);
    %sinal a ser gerado
    if (fi == 0)
        nota=zeros(1, round((fs*sec)/100));
    else
        nota=a*cos(W * n  + ph);
    end
end
```

sintetizador.m

```
%Mais informação nos ficheiros .m .
function [s]= sintetizador(n)
    if (( mod(n,2)) ~= 0 )
       fprintf('O argumento tem que ser um conjunto par.\n');
       help sintetizador;
       return;
    end

    base_frq_Do=440;

    for i=1:2:length(n)
        exp=(n(i))/12;
        j=i+1;
        time=n(j);
        if (exp ~= 0)
            base_frq=base_frq_Do*power(2,exp);

        else
            base_frq=0;
        end

        [s,fs]=t1_nota(2,base_frq,time);
       sound(s,fs);
    end
end
```

Tendo em conta as alíneas seguintes tentamos tornar as nossas funções o mais genéricas possíveis e para tal efectuamos algumas alterações na nossa primeira função para poder contemplar a duração que um som vai tocar. Esta função somente é responsável por gerar o sinal que irá ser reproduzido, ou não, por quem solicitou a geração do sinal.

Alínea B

```
t1b_testes.m
function [] = t1b_testes()
    %Amplitude:
    %variação de amplitude faz variar o volume do sinal audivel. maior
    %amplitude, som mais alto, menor amplitude som mais baixo.
    %Frequencia:
    %Quanto maior a frequencia mais rápido o som  se torna mais agudo, quanto menor, mais
    %lento, som mais grave.
    %Fase:
    %Não foi notada diferença
    A_test=2;
    frq_test=440;
    sec=0.5;
    ph=0;
    %Teste de Amplitude
    fprintf('Teste de Amplitude.\n');
    for A=1:5
        t1_nota(A,frq_test,sec,ph);
    end
    pause(2);

    %Teste de Frequencia
    fprintf ('##########################################\n');
    fprintf (' Teste ao nivel de frequencia mínima audivel.\n');
    fprintf ('##########################################\n');
    fprintf('Frequencia: ');
    for frq_test=1:30
        fprintf(' %i ',frq_test);
        [s,fs]=t1_nota(A_test,frq_test,sec,ph);
        sound(s,fs);
    end
    fprintf('\n');

    fprintf ('##########################################\n');
    fprintf (' Teste ao nivel de frequencia máxima audivel.\n');
    fprintf ('##########################################\n');
    fprintf('Frequencia: ');
    for frq_test=1:30
        x=19000 + 1000*frq_test;
        [s,fs]=t1_nota(A_test,x,sec,ph);
        fprintf('  [%i <-> %i] ',x,fs);
        sound(s,FS);
    end
     fprintf('\n');


    fprintf('Teste de Fase.\n');
    %Teste de Fase
    frq_test=263.61;
    for ph=0:pi/12:2*pi
      t1_nota(A_test,frq_test,sec,ph);
    end
```

Ao longo dos teste verificámos que a variação de amplitude faz variar o volume do som produzido. Quanto o maior o valor da amplitude mais "alto" está o som produzido, ou seja, a variação de amplitude faz um efeito atenuador/amplificador do som produzido.

Nos testes de frequência constatámos que a sua alteração modifica a velocidade com que o som é produzido. Esta velocidade é reconhecida pelo numero de ciclos que ocorrem dentro do mesmo período de tempo. Esta constatação é afere-se pelo facto de quando o som produzido está mais lento o som audível é mais grave e quando o som está mais rápido, o som produzido é mais agudo.

Nos nossos testes não verificámos diferença, quando alterámos a sua fase. Isto é explicado pelo facto de estarmos a analisar um único som produzido, e o facto de o som começar num ponto ou desfasado tornar-se um pouco indiferente ao ouvido humano.

No entanto a mudança de fase, usada em operações com outros sinais pode provocar atrasos na forma como o sinal é produzido. Esta alteração de fase levou-nos a ter em conta a frequência de amostragem utilizada para que o sinal utilizado pudesse ser reconstruído em operações ADC e DAC, tendo para tal a frequência que respeitar o ritmo de Nyquist, evitando desta forma o aliasing do sinal.

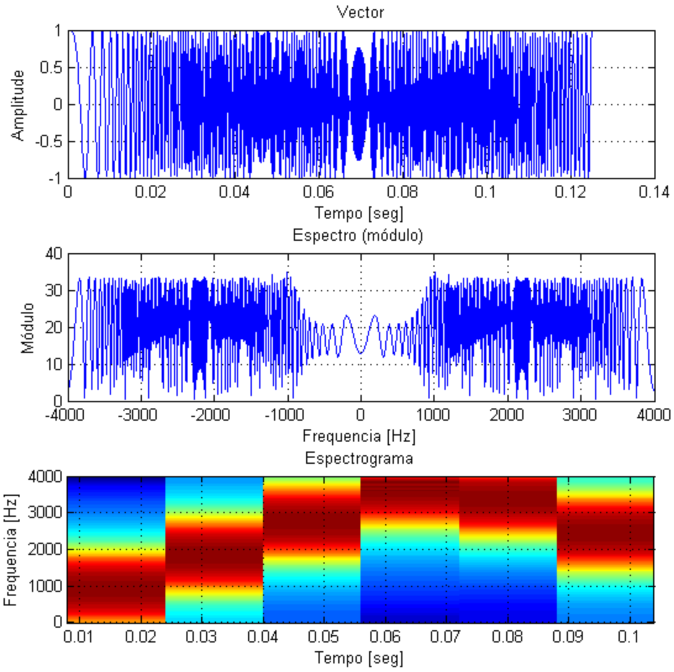
Alínea C

A função chirp realiza um varrimento na frequência aplicando um sinal coseno. Tem a vantagem de permitir estipular o intervalo de frequência a trabalhar, no entanto não deve ser utilizado em operações que utilizem o sinal do tipo "ruído branco".

Sendo o nosso ficheiro de teste o seguinte:

```
function t1c()
    close all;
    %Escollha de uma frequencia de amostragem
    fs=1000;
    %efectuado um varrimento ao sinal em 1 segundo
    t=0:1/fs:1;
    s_chirp=chirp(t,fs);
    %efectuado a analisys do sinal anterior.
    figure
    analysis(s_chirp,fs);

end
```

originou os seguintes gráficos:



No gráfico Vector, podemos analisar a variação de amplitude ao longo do tempo. Neste gráfico verificamos que o sinal vai ficando mais comprimido no tempo entre os valores 0.02 a 0.12 segundos. Sendo esta facto uma verdade, estamos à espera que o espectro seja expandido.

No gráfico Espectro, verifica-se a variação do sinal na frequência e como estávamos o sinal está expandido ao longo das frequências.

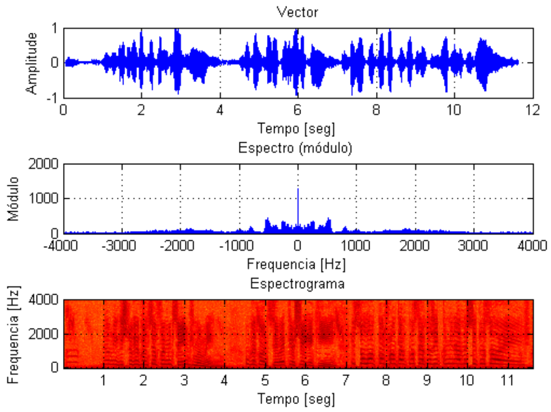
Por fim o espectrograma demonstra a utilização das frequências ao longo do tempo. Neste gráfico verificamos que uma largura de banda de cerca 2k[hz] diferente em cada 0.02 segundos.

Alínea D

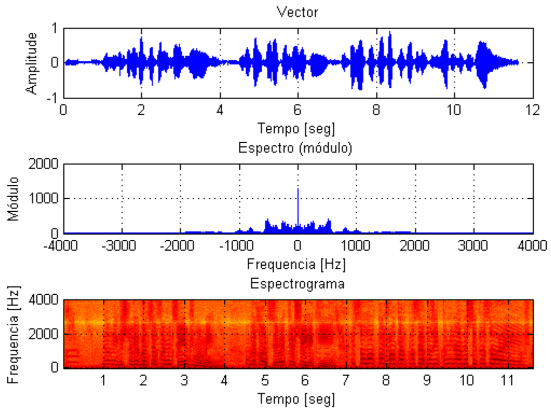
Ao implementar o sistema indicado obtemos a seguinte função:

```
function [] = t1d(signal)
    if(nargin ~= 1)
        fprintf('Necessita de colocar um sinal de entrada');
        return
    end
    close all;
    [x,fs]=wavread(signal);
    y=(1/3)*[x;0;0] + (1/3)*[0;x;0] + (1/3)*[0;0;x];

    figure;
    analysis(x,fs);

    figure;
    analysis(y,fs);
end
```

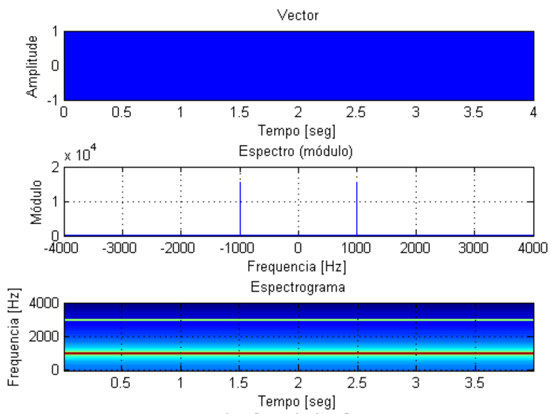
o resultado a função analisys é o seguinte:



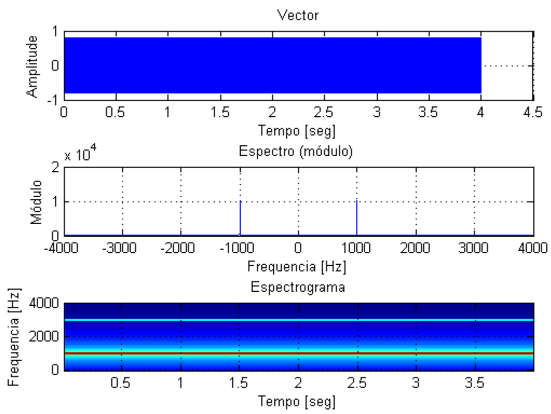
Sinal Original Fala



Sinal Filtrado Fala



Sinal Original Seno



Sinal Filtrado Seno

Nesta análise verifica-se que é aplicado um filtro passa-baixo, para permitir que as frequências de -2k[hz] a 2k[hz] fossem utilizadas, sendo todas a outras ignoradas. Esta atenuação teve uma consequência, a amplitude do sinal foi atenuada, uma vez que as amplitudes mais altas estavam fora da banda]-2k,2k[[hz].

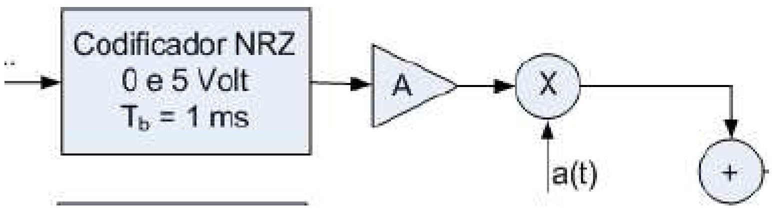
No espectrograma é evidente que há menos utilização das frequências em relação ao original.

Exercício 2

Alínea A

A análise do diagrama de blocos permitiu-nos realizar um esquema equivalente de forma a separa os ramos que representas o sinal de $x(t)$.

Assim temos um sinal $x_1(t)$ com o seguinte esquema:



Com base neste esquema podemos obter de forma genérica a expressão do $x_1(t)$.

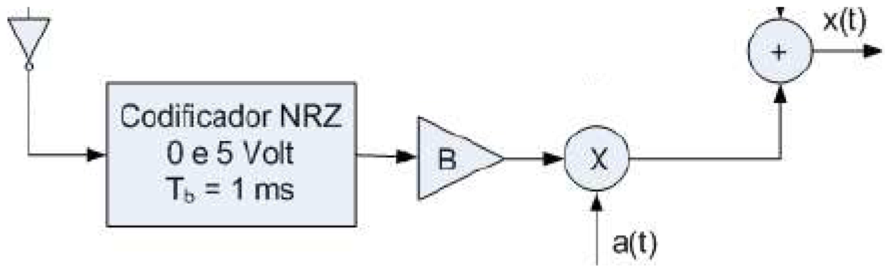
Para a obtenção desta expressão há que identificar os blocos que a representam.

Temos um sinal de entrada, que identificaremos como b , que está em série com o codificador NRZ, temos um amplificador A e uma multiplicação com o sinal $a(t)$.

Com estes blocos identificados, a expressão é obtida rapidamente:

$$x_1(t) = b * NRZ * A * a(t)$$

Analogamente, a mesma análise foi realizada para um sinal $x_2(t)$.



Este diagrama é semelhante ao $x_1(t)$, diferindo no aspecto que o sinal de entrada, o b aparece "negado", significando que para este diagrama vem o complementar do sinal de entrada.

$$x_2(t) = \neg b * NRZ * B * a(t)$$

Tendo a expressão de ambos os sinais, é simples achar o valor de $x(t)$:

$$x(t) = x_1(t) + x_2(t) \Leftrightarrow x(t) = b * NRZ * A * a(t) + \neg b * NRZ * B * a(t) \Leftrightarrow$$
$$\Leftrightarrow x(t) = NRZ * a(t) * (b * A + \neg b * B)$$

Uma vez identificado a expressão de $x(t)$ facilmente obtemos a expressão de $c(t)$:

$$c(t)=x(t)*a(t)\Leftrightarrow c(t)=NRZ*a(t)*(b*A+\neg b*B)*a(t)\Leftrightarrow$$
$$\Leftrightarrow c(t)=NRZ*a(t)^2*(b*A+\neg b*B)$$

Estamos a admitir que o meio de transmissão é representado por sistema identidade, que como podemos verificar nas aulas, o sinal antes do meio de transmissão é igual ao sinal quando chega depois do meio de transmissão.

Apesar de termos esta expressão geral, que é possível aplicar para qualquer que sejam os sinais de entrada, a forma como o NRZ está implementado ou mesmo os parâmetros A e B, temos que o particularizar para o exercício. Para o podermos efectuar, temos que fazer um pequeno estudo somente do codificador NRZ, uma vez que o sinal $a(t)$ é fornecido pelo enunciado e todo este sistema pode ser aplicado para todo e qualquer sinal de b.

O codificador NRZ (Non-return-to-zero) representa os bits de entrada em valores 0v se o bit for 0 e 5v se o bit for 1 dado um tempo de bit ,que seguindo o diagrama de blocos verificamos que é 1ms.

Podemos concluir que este codificador pode ser representado como uma onda quadrada, em tempos de bit de 1ms, do qual se pode obter a seguinte expressão simplificada, visto que quando o bit de entrada for 0, o valor durante esse tb (tempo de bit) vai ser 0.

Expressão Matemática geral:

$$NRZ(tb)=\begin{cases}5, 0\leq\tau\leq T_b\, se\, bit=1\\ 0, 0\leq\tau\leq T_b\, se\, bit=0\end{cases}$$

Então a nossa expressão para o NRZ será:

$$NRZ(T_b)=5*\Pi\left(\frac{\tau-\frac{T_b}{2}}{T_b}\right)=5*\Pi\left(\frac{2*\tau-tb}{2\mathrm{T}_b}\right)$$

Sabendo todas as funções relevantes para obter a expressões específicas para o nosso enunciado, torna-se evidente a sua resolução.

Resumindo, sabemos :

- $a(t)=\cos(2*\pi*f_o*t)$ e que $f_o=10000\text{Hz}$
- $NRZ(T_b)=5*\Pi*\left(\frac{\tau-\frac{T_b}{2}}{T_b}\right)$ e que $tb=1\text{ms}$
- A e B parâmetros do sistema, que serão utilizados em alíneas mais adiante.

Neste momento estamos nas condições ideais para exprimir os valores de $x(t)$ e de $c(t)$:

$$x(t)=NRZ*a(t)*(b*A+\neg b*B)\Leftrightarrow$$

$$\Leftrightarrow x(t)=5*\Pi\left(\frac{\tau-\frac{T_b}{2}}{T_b}\right)*\cos(2*\pi*f_o*t)*(b*A+\neg b*B)\Leftrightarrow$$

$$\Leftrightarrow x(t)=5*\Pi\left(\frac{\tau-\frac{0,001}{2}}{0,001}\right)*\cos(2*\pi*10000*t)*(b*A+\neg b*B)$$

$$c(t)=NRZ*a(t)^2*(b*A+\neg b*B)\Leftrightarrow$$

$$\Leftrightarrow c(t)=5*\Pi\left(\frac{\tau-\frac{0,001}{2}}{0,001}\right)*\cos(2*\pi*10000*t)^2*(b*A+\neg b*B)$$

Para obtermos o ritmo de transmissão , partimos da relação geral do tempo de bit, $T_b=K*T_o$, sendo K o numero de vezes que ocorre que um sinal de “repete” no tempo, que corresponde a um tempo de bit.

Seguindo a análise:

$$T_b=K*T_o$$
$$T_b=10^{-3}$$
$$T_o=\frac{1}{f_o}=10^{-4}$$

$$T_b=K*T_o$$
$$10^{-3}=K*10^{-4}\Leftrightarrow\frac{10^{-3}}{10^{-4}}=K\Leftrightarrow$$
$$\frac{10^4}{10^3}=K\Leftrightarrow 10=K$$

Sabemos também que o ritmo de transmissão respeita a seguinte expressão:

$$R_b=\frac{f_o}{K}\Leftrightarrow R_b=\frac{10^4}{10^1}\Leftrightarrow R_b=10^3$$

Obtemos então que o ritmo de transmissão é de 1000[b/s] (bits por segundo).

Alínea B

Compreendendo como funciona a modelação OOK (On-off keying) que aplica um sinal nulo quando o valor é 0, um sinal diferente de 0 (suficientemente elevado para que possa ser identificado) torna-se evidente que valores podemos atribuir aos parâmetros do sistema A e B .

Analogamente podemos efectuar uma análise semelhante para a modulação PSK (Phase-shift keying) que aplica um desvio de fase em relação ao sinal de entrada, caracterizando de forma distinta o sinal quando o bit é 1 e quando o bit é 0.

Sintetizando ideias:

$$OOK=\{A\neq 0 \wedge B=0\}$$

$$PSK=\{A=-B\}$$

Alínea i

Por exemplo:

$$OOK=\begin{cases} A\neq 0\ B=0 \\ A=1, se\ bit=1 \\ A=0, se\ bit=0 \end{cases}$$

Alínea ii

Por exemplo:

$$PSK=\begin{cases} A=-B \\ A=1, se\ bit=1 \\ B=-1, se\ bit=0 \end{cases}$$

Alínea C

Realização do sistema levou-nos ao desenvolvimento das seguintes funções:

NRZ.m

```
function [FS,myX,mynX,n]=NRZ(signal,Amp,CarrierFreq)
    if(nargin == 0)
        fprintf('É necessário mais argumentos.\n');
        return;
    elseif(nargin >3)
        fprintf('Têm argumentos a mais.\n');
        return;
    elseif (nargin == 1)
        fprintf('Assumindo a Amplitude 5 e frequencia da portadora de 100Hz.\n');
        CarrierFreq=100;
        Amp=5;
    elseif (nargin == 2)
        fprintf('Assumindo a frequencia da portadora de 100Hz.\n');
        CarrierFreq=100;
    end


    %Tempo de Bit do Nosso NRZ
    nrzTs=0.001;

    %Numero de elementos do sinal de entrada
    nbrBits=length(signal);

    %Frequencia Fundamental de Saída, respeitando o Ritmo de Nyquist
    FS=2.2*(1/(nrzTs));
    if (FS > CarrierFreq)
         fprint('A frequencia da portadora é inferior à frequencia de amostragem do sinal amostrado. Não vai ser possível
reconstruir com exactidão o sinal.');
    end

    %Numero de elementos da nossa Base tempo.
    n=0:1/(nbrBits*FS-1):1;

    %Nosso conjunto que vai conter o sinal de saída
    myX= 1:FS;
    mynX= 1:FS;

    %Ciclo que vai criar a onda quadrada
    k=1;
    for i=1:nbrBits
        for  h=1:FS
            if( signal(i) == 1)
                myX(k)=Amp;
                mynX(k)=0;
                k=k+1;
            else
                myX(k)=0;
                mynX(k)=Amp;
                k=k+1;
            end
        end
    end
    FS=FS*nbrBits;
end
```

Esta função gera o sinal de entrada codificado pelo NRZ. Recebe o valor da frequência da portadora, como forma de controlar se o sinal gerado poderá ser reconstruído.

Codificador.m

```
function [FS,mySignal,t] = codificador(signal,A,B,fo)
    if(nargin == 0)
        printf('É necessário mais argumentos.\n');
        return;
    elseif(nargin >4)
        printf('Têm argumentos a mais.\n');
        return;
    end

    Amp=5;
    [FS,Xt,nXt,n]=NRZ(signal,Amp,fo);
    t=0:1/(FS-1):1;
    x1T = modula(Xt,A,fo,t);
    x2T = modula(nXt,B,fo,t);
    mySignal=x1T + x2T;
end
```

Esta função simula o sistema para gerar o sinal $x(t)$.

PSK.m

```
function [FS,mySignal,t]=PSK(signal,paramA,paramB,fo)
    if(nargin == 0)
        fprintf('É necessário mais argumentos.\n');
        return;
    elseif(nargin >4)
        fprintf('Têm argumentos a mais.\n');
        return;
    elseif (nargin == 1)
        fprintf('Assumindo o valor 1 do paramA e o valor -1 no paramB.\n');
        paramA=1;
        paramB=-1
    elseif (nargin == 2)
        fprintf('Assumindo o valor -paramA para o paramB.\n');
        paramB=-paramA;
    else
        if (paramB - paramA > 0)
            paramB = -paramB;
        end
    end
    [FS,mySignal,t]=codificador(signal,paramA,paramB,fo);

    figure;
    plot(t,mySignal);
    title('Sinal PSK');
    grid on;
end
```

Esta função simula a modulação PSK.

OOK.m

```
function [FS,mySignal,t]= OOK(signal,paramA,paramB,fo)
    if(nargin == 0)
        fprintf('É necessário mais argumentos.\n');
        return;
    elseif(nargin >4)
        fprintf('Têm argumentos a mais.\n');
        return;
    elseif (nargin == 1)
        fprintf('Assumindo o valor 1 do paramA e o valor 0 no paramB.\n');
        paramA=1;
    end
    paramB=0;

    [FS,mySignal,t]=codificador(signal,paramA,paramB,fo);
end
```

Esta função simula a modulação OOK.

sistema.m

```
function sistema(signal,fo)

    %Codificador NRZ

    [nrzFS,nrzmyX,nrzmynX,nrzn]=NRZ(signal,5,fo);

    %Modelação OOK

    [ookFS,ookSignal,ookn]=OOK(signal,1,0,fo);

    %Modelação PSK

    [pskFS,pskSignal,pskn]=PSK(signal,1,-1,fo);

    %Recuperação do x(t)

    cT = modula(pskSignal,1,fo,pskn);

    figure;

    subplot(4,1,1);

    plot(nrzn,nrzmyX);

    title('Sinal após condificação NRZ');

    subplot(4,1,2);

    plot(ookn,ookSignal);

    title('Sinal após modulação OOK');

    subplot(4,1,3);

    plot(pskn,pskSignal);

    title('Sinal após modulação PSK');

    subplot(4,1,4);

    plot(pskn, cT);

    title('Sinal c(t), ou o x(t) recuperado');

end
```

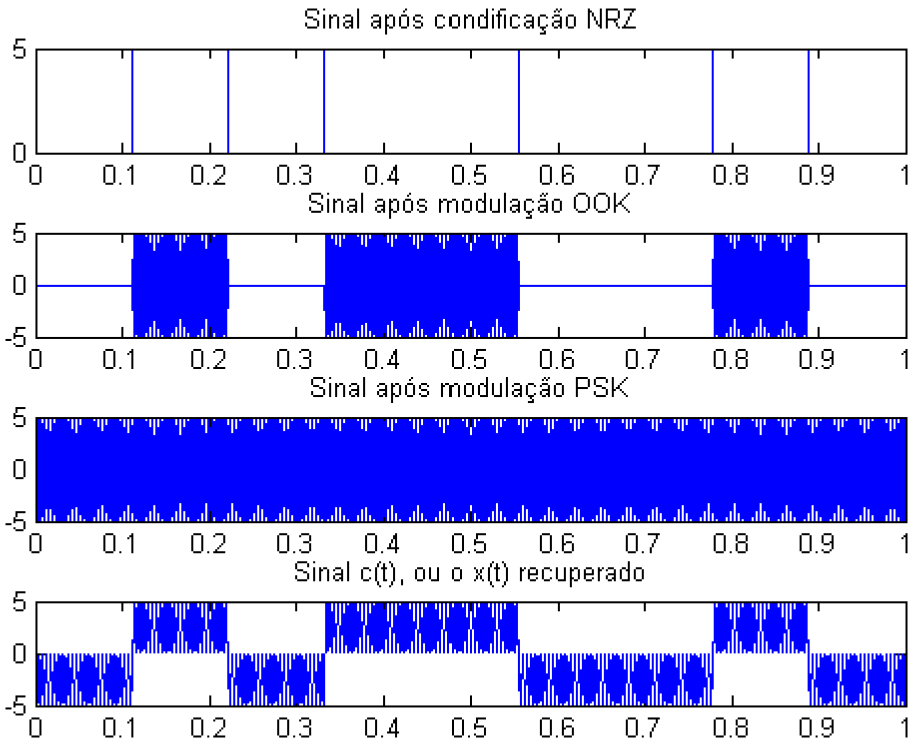
Esta função simula todo o nosso sistema para gerar o sinal $x(t)$

modula.m

```
function [mysignal]=modula(signal,amplierFactor,CarrierFreq,timebase)
    aT=cos(2*pi*CarrierFreq*timebase);
    mysignal= signal.*(amplierFactor*aT);
end
```

Esta função somente existe para reutilizar código. Devolve o sinal modulado por $a(t)$.

Na execução do comando (sistema([0;1;0;1;1;0;0;1;0],10000);), para cumprir com o solicitado apresentamos os resultados (esperados).



Exercício 3

Alínea A

$a(t)=\cos(2*\pi*10000*t);$

Alínea i)

$x(t)=1+\cos(2*\pi*2000*t);$

$y(t)=x(t)*a(t)\Leftrightarrow y(t)=\cos(2*\pi*10000*t)+\cos(2*\pi*2000*t)*\cos(2*\pi*10000*t)$

Aplicando a dedução matemática que nos indica:

$\cos(A+B)+\cos(A-B)=2\cos(A)\cos(B)$ podemos aferir que o nosso produto pode ser separado da seguinte forma:

$$\cos(A)\cos(B)=\frac{1}{2}\cos(A+B)+\frac{1}{2}\cos(A-B)$$

desta forma podemos continuar a dedução da nossa expressão.

$$y(t)=\cos(2*\pi*10000*t)+\frac{1}{2}*\cos(2*\pi*2000*t+2*\pi*10000*t)$$

$$+\frac{1}{2}*\cos(2*\pi*2000*t-2*\pi*10000*t)\Leftrightarrow$$

$$\Leftrightarrow y(t)=\cos(2*\pi*10000*t)+\frac{1}{2}*\cos(2*\pi*12000*t)+\frac{1}{2}*\cos(2*\pi*(-8000*t))\Leftrightarrow$$

$$\Leftrightarrow y(t)=\cos(2*\pi*10000*t)+\frac{1}{2}*\cos(2*\pi*12000*t)+\frac{1}{2}*\cos(2*\pi*8000*t)$$

Verificando a nossa expressão final podemos concluir que o nosso espectro (unilateral) terá representada as frequências 8000hz, 10000hz e 12000hz com amplitude $\frac{1}{2}$, 1 e $\frac{1}{2}$.

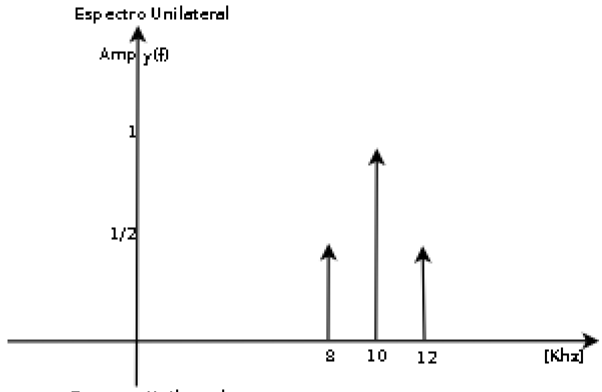
Alínea ii

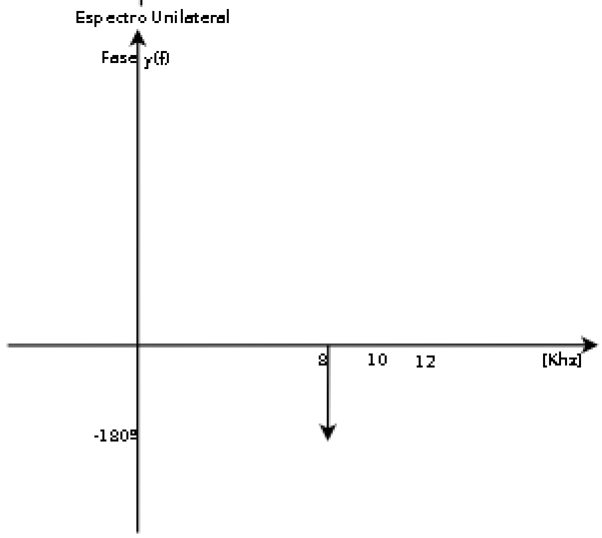
$x(t)=sinc(t)$

$y(t)=x(t)*a(t)\Leftrightarrow y(t)=sinc(t)*\cos(2*\pi*10000*t)$

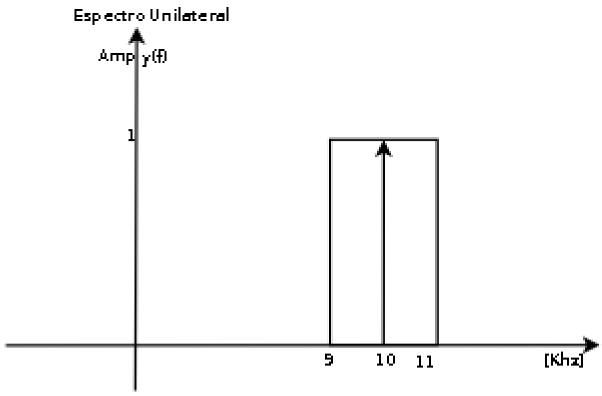
Podemos afirmar perante esta expressão que teremos um pulso rectangular centrado na frequência de 10000hz com largura 1 unidade.

Esboço i)





Esboço ii)



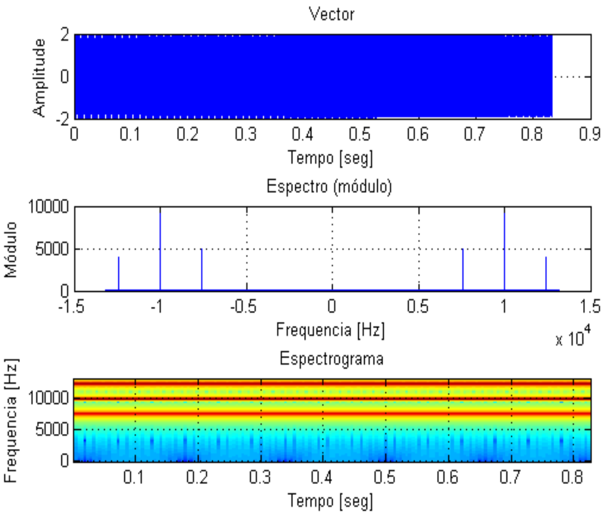
Alínea B

A nosso função emissor:

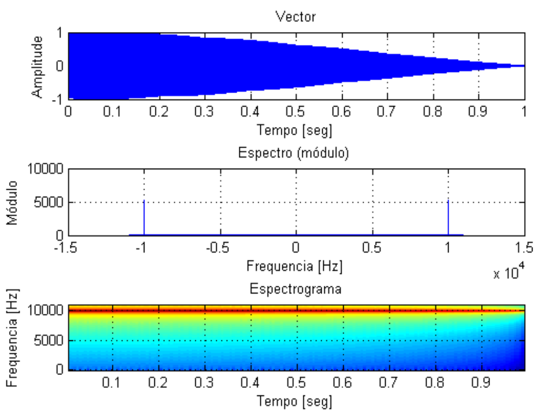
emissor.m

```
function [yF,FS]= emissor(signal,Fo)
    %wavplay(signal,Fo);
    carrierFS=10000;
    FS=2.2*(carrierFS+Fo);
    t=0:1:length(signal)-1;

   figure('name','Sinal Original');
   my_analysis(signal,Fo);


    %Sinal da Portadora
    aT=cos(((2*pi*carrierFS)/FS)*t)';
    yT=aT.*signal;
    %figure('name','Sinal yT à Saida do Emissor');
    %my_analysis(yT,FS);

    yF=fft(yT);
end
```

Da analise do sinal resulta:



O espectro do sinal vai de acordo com as nossas expectativas, estando o sinal centrado na frequência da portadora (10KHz)



No sinal da sinc, o espectro apresenta-se com a frequência de 10 k[hz].

Alínea c

De forma geral podemos simplificar o processo no seguinte diagrama.

Este processo tem vários passos que de forma sistemática significam o seguinte:

- Emissor:
  - **Filtro Anti-Aliasing:** Este filtro aplica-se para garantir, dentro do limiar do possível que o sinal de entrada respeita o Ritmo de Nyquist, para poder garantir a recuperação do sinal no fim do processo.
  - **Filtro de Amostragem**: Este filtro tem o objectivo de discretizar a amplitude do sinal no domínio do tempo.
  - **Filtro de Quantificação:** É aplicado para obter o numero de bits necessário para poder codificar o sinal de entrada. Este número depende do domínio do problema, mas podemos garantir que quanto maior ele for, menor será a margem de erro na reconstrução do sinal.
  - **Filtro de Codificação:** Constrói palavras binárias de acordo com a quantificação realizada anteriormente.
- Receptor:
- **Filtro Anti-Aliasing**: Este filtro pode-se utilizar no receptor porque não é mais do que um filtro passa-baixo, e uma vez que o sinal respeita (em princípio) o Ritmo de Nyquist, vai somente filtrar a banda de frequências correctas para poder reconstruir o sinal, não apanhando no processo as harmónicas fantasmas.
- **Filtro de Reconstrução**: Descodifica as frequências filtradas, para o sinal original.

De acordo com o enunciado, o nosso emissor terá em conta a frequência da portadora (10000hz) para poder garantir o Ritmo de Nyquist e desta forma garantir que o sinal poderá ser reconstruido com uma margem de erro mínima e utilizando os valores dos sinal poderá obter os valores necessários para a quantificação do sinal.

O receptor terá somente que fazer a operação inversa com os mesmos valores de parâmetros para proceder à reconstrução.

receptor.m

```
function [xT,TS]= receptor(signal,FS)
    carrierFS=10000;
    Fo=FS/2.2 - carrierFS;
    %Filtro PassaBaixo/PassaBanda
    TS=1/carrierFS;
    yT=sinc(TS);
    yF=fft(yT);

    wF=signal.*yF;

    xT=ifft(wF);
    %analise do sinal
    figure('name','Sinal à Saída do Receptor.');
    my_analysis(xT,Fo);
    %wavplay(xT,Fo);
end
```

exercicio3.m

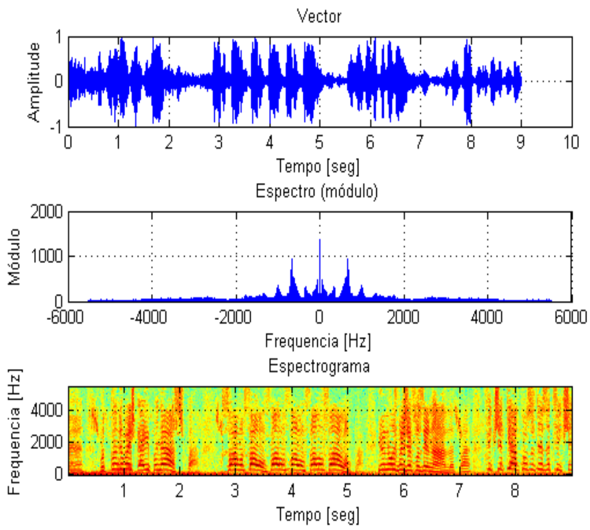
```
function [newXT,TS]= exercicio3(nbr)

    fo=10000;

    fs=2.2*fo;
    t=0:1/(fs-1):1;

    switch nbr
        case 1
            fo2=2000;
            xT=(1+cos(2*pi*fo2*t))';
        case 2
            fo2=2000;
            xT=(sinc(t))';
        case 3
            [xT,fo2]=wavread('fala4.wav');
        otherwise
            return;
    end
    %
    %Fase de Emissor
    %

    [yF,FS]=emissor(xT,fo2);

    %
    %Fase de Receptor
    %
    [newXT,TS]= receptor(yF,FS);
end
```

Alínea d

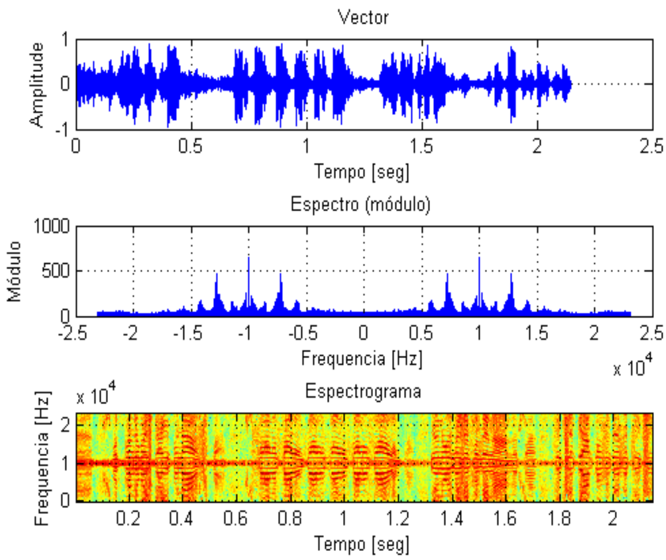
Neste exercício, encontrámos alguma dificuldade (que não foi ultrapassada) na reconstrução do sinal original, uma vez que não estávamos a conseguir limpar o sinal emitido pelo emissor. Uma vez que a entrega do trabalho está bastante atrasada, enviaremos os nossos resultados (incorrectos) e posteriormente entregaremos a sua correcção.

Assim ao executar o comando ***exercio3(3);*** o nosso script irá correr o emissor e de seguida o receptor "automáticamente".

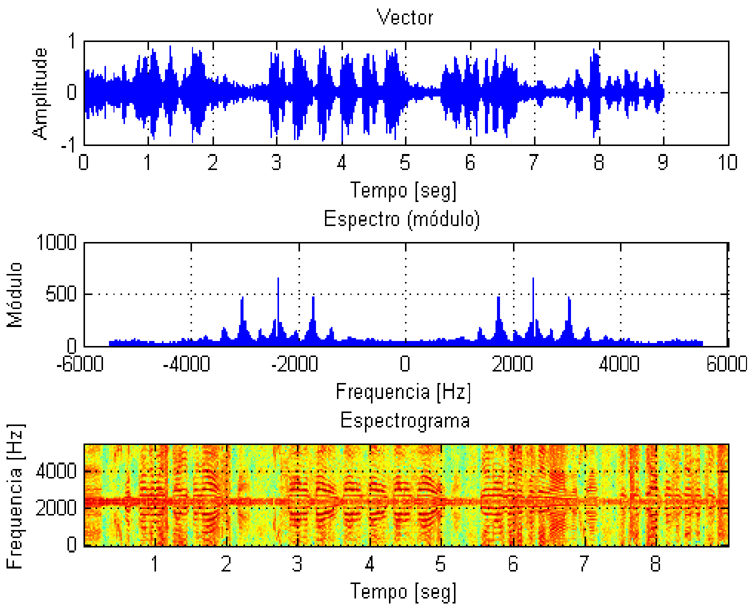
Estes são os gráficos obtidos:




```
figure('name','Sinal Original');
```




```
figure('name','Sinal yT à Saida do Emissor');
```




```
figure('name','Sinal à Saída do Receptor.');
```

Como se pode verificar, sinal final tem algumas semelhanças em relação ao inicial, mas não é o mesmo. Esperamos no futuro compreender o porquê de estar menos correcta a implementação.

Alterações à Versão anterior

- Nesta versão do relatório, foram alterados os scripts de execução do exercício 3.
- Correcção do gráfico do exercício 3b e respectiva conclusão.
- Conclusão do exercício 3d, embora não esteja correcta a implementação.