Instituto Superior de Engenharia de Lisboa
Área Departamental de Engenharia Electrónica e
Telecomunicações e de Computadores

Comunicações*(LEIC)*

Relatório do 1º Trabalho

Eng.º David Coutinho

Semestre de Verão 2013-2014

Turma: LI31N – LEIC
Diogo Poeira, nº36238
Ana Sequeira, nº 35479

Introdução

Neste Trabalho foi pedido aos alunos que desenvolvessem certas funções em Matlab para resolver problemas e/ou que observassem gráficos para deduzir e calcular dados relativos a sinais e espectros.

Exercício 1

i)

O que se faz neste exercício é criar uma função que recebe por parâmetros a amplitude frequência fase e número de períodos a representar de um cosseno. A partir da frequência obtemos um período e a partir do período obtemos o intervalo de tempo correspondente ao número de períodos a representar. Depois disso é simplesmente multiplicar a função cosseno com os parâmetros apropriados, por A e desenhar x(t).

ii)

Neste exercício a logica para obter o cosseno é igual á do exercício anterior. Para obter o pulso retangular pelo qual se vai multiplicar o cosseno é necessário descobrir primeiro a dimensão do pulso, que é igual á duração do pulso dada por T a dividir pelo tempo de amostragem Ts dado pelo tempo de um período a dividir por 100 (nº de amostras). Depois de saber o tamanho do pulso é necessário descobrir a dimensão do deslocamento, a logica é a mesma com trocando o T pelo parâmetro desloc. A partir destas novas variáveis calculamos a quantidade de zeros á esquerda e á direita do pulso dentro do intervalo designado. Se o deslocamento for grande o suficiente é possível que tenha de se retirar amostras do pulso acrescentando zeros a um dos lados sendo que neste caso, só há zeros de um dos lados do pulso.

```
function x = pulsosSinusoidais(A,fo,phi,T,desloc)
to = 1/fo;
Ts = to/100;
t = -4*to+Ts:Ts:4*to;
dim = length(t);
dimPulso = round(T/Ts);
dimDesloc = round(desloc/Ts);
dimEsq=round(dim/2-dimPulso/2-dimDesloc);
dimDir=round(dim/2-dimPulso/2+dimDesloc);
if abs(dimDesloc)+dimPulso/2>dim/2;
   dimPulso = dimPulso - (abs(dimDesloc)+dimPulso/2-dim/2);
   if dimPulso<0
      dimPulso=0;
   end
   if dimDesloc<0
      dimDir = 0;
      dimEsq = floor(abs(dim-dimPulso));
   else
      dimDir = floor(abs(dim-dimPulso));
      dimEsq = 0;
   end
end
v = [zeros(1,dimEsq) ones(1,dimPulso) zeros(1,dimDir)];
x = A*cos(2*pi*fo*t+phi);
res = v.*x;
plot(t,res);
end
```

iii)

```
function musicalSequence(f, d, N) % f = frequencia, d = duração em segundos, N = número de notas
músicais

   n = 1 : 1 : f*d;
   Fs= 4000;
   for i=1 : N
      W = 2*pi*(f/Fs);
      Signal = cos( W * n );
      wavplay( Signal, Fs );
      f = f * power(2,1/12);
      wavplay( zeros(1, round((Fs*n)/10)), Fs );
   end
end
```

Para esta alínea foi reutilizado o código fornecido no guia de matlab, esta função recebe como parâmetros a frequência inicial da sequência, a duração de cada uma das notas e o número total de notas.

Aquilo que é feito nesta função é a criação de um sinal em função da frequência e da duração passados por parâmetro, para posteriormente ser efectuado o wavplay desse mesmo sinal e obter a nota musical.

iv)

```
function randomNoise(n, e)      % n = nº amostras, e = energia

x = (rand(1,n));
E = x* x';
x = x * sqrt(e / E );
wavplay(x);
plot(x);
end
```

Nesta função recebemos por parâmetro o número de amostras do sinal e a energia do mesmo.

É criado um vector com n colunas, sendo que as posições são preenchidas com valores random, usando a função rand, é depois calculada a energia desse vector. A variável x passa a ser o vector multiplicado pela divisão das duas energias. Por último realiza-se o wavplay e o plot do sinal.

Exercício 2

a)

Observando o resultado da função analisys para o sinal a com Fs = 50KHz deduz-se que:

N = 3 visto que existem 3 picos de amplitude em 3 frequências diferentes (maiores que zero).

Ak = {10 , 5 , 12 , 5} se observarmos os valores de cada pico de amplitude e multiplicarmos por 2 cada um dos picos que seja maior que zero, obtemos aproximadamente esses valores.

Fo = gcd(1950, gdc(2350,3550)) = 50Hz

LB = 3550-0 = 3550 Hz

Valor Médio de a = ma = -10

Valor máximo = 12

Valor mínimo = -27

b)

b.mat

ASK

Amplitude Máxima – 2 V

Tempo de Bit – 0.003s

Ritmo Binário – 333,333 Bit/s

Sequência - 011001

c.mat

PSK

Amplitude Máxima- 6 V

Tempo de Bit – 1s

Ritmo Binário – 1 Bit/s

Sequência - 000101

c)

Cada tecla é codificada em 0.2 segundos usando a função, definida por nós, numeroDeTelefone obtém-se o número de telefone 218317000. Esta função serve para descodificar as frequências correspondentes a cada tecla aproximando-as ás da tabela DTMF e associando esses valores a uma tecla.

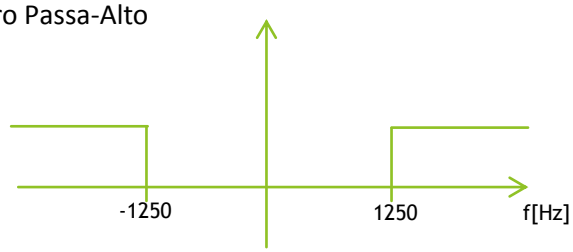
```
function x = numeroDeTelefone(sinal,Fs)
x = ['0' '0' '0' '0' '0' '0' '0' '0' '0'];
dimTecla = length(sinal)/9;
tecla = zeros(1,dimTecla);
freq = ((0 : 1 : dimTecla-1)*Fs / ( dimTecla )) - Fs/2;
for i=1:9
    for j=1:dimTecla
        tecla(j) = sinal((i-1)*3200 + j);
    end
    CK = fftshift( abs(fft(tecla)) );
    CK = CK/dimTecla;
    f = findMaxs(freq,CK);
    x(i) = descodificaTecla(f);
end
return
```

```
function t = descodificaTecla(f)
linha = [697 770 852 941];
coluna = [1209 1336 1477 1633];
l = 1;
c = 1;
aux = 1;
teclado = ['1' '2' '3' 'A' ; '4' '5' '6' 'B' ; '7' '8' '9' 'C' ; '*' '0' '#' 'D'];
for i = 1:4
   if i == 1
      aux = abs(linha(i)-f(1));
   end
   if aux > abs(linha(i)-f(1))
     aux = abs(linha(i)-f(1));
     l = i;
   end
end
for j = 1:4
   if j == 1
      aux = abs(coluna(j)-f(2));
   end
   if aux > abs(coluna(j)-f(2))
     aux = abs(coluna(j)-f(2));
     c = j;
   end
end
```

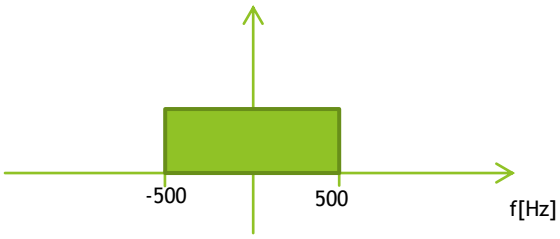
```
function x = findMaxs(freq,CK)
subir = 0;
x = zeros(1,2);
max1 = 1;
max2 = 1;
j=1;
for i = round((length(CK)/2))+1:1:length(CK)
   if(subir==0 && CK(i)>CK(i-1))
      subir=1;
   end
   if(subir==1 && CK(i)<CK(i-1))
     subir = 0;
     if freq(i-1)<=1000
        if CK(max1)<CK(i-1)
           max1 = i-1;
        end
     else
         if CK(max2)<CK(i-1)
           max2 = i-1;
        end
     end
     j = j+1;
   end
end
x(1) = freq(max1);
x(2) = freq(max2);
return
```

d)

e1.mat – Filtro Passa-Alto



e2.mat – Filtro Passa-Baixo



Exercício 3

a)

Neste exercício realizou-se um emissor de letras codificadas em Morse. O emissor foi dividido em duas partes a primeira parte é o codificador de fonte que transforma as letras em bits e a segunda parte é o codificador de canal que transforma esses bits num sinal sinusoidal ou em zero.

```
function x = Emissor(simbolos,Tb,A,F)
codigo = CodificadorFonte(simbolos);
x = CodificadorCanal(codigo,Tb,A,F);
return
```

```
function codigo = CodificadorFonte(simbolos)

chaves = 'abcdefghijklmnopqrstuvwxyz ';

valores = {[0 1],[1 0 0 0],[1 0 1 0],[1 0 0], 0 ,[0 0 1 0],[1 1 0],[0 0 0 0],[0 0],[0 1 1 1],[1 0
1],[0 1 0 0],[1 1],[1 0],[1 1 1],[0 1 1 0],[1 1 0 1],[0 1 0],[0 0 0],1,[0 0 1],[0 0 0 1],[0 1
1],[1 0 0 1],[1 0 1 1],[1 1 0 0],[1 0 0 1 0]};

codigo = [];

for i = 1 : length(simbolos)

    for j = 1 : length(chaves)

        if lower(simbolos(i)) == chaves(j);

            codigo = [codigo cell2mat(valores(j)) -1];

            break;

        end

    end

end

return
```

```
function codigo = CodificadorCanal(codigoSimbolos,Tb,A,F)
t = 0 : Tb/99 : Tb;
codigo = zeros(1,length(codigoSimbolos)*100);
i = 1;
cosineForOne = (A*cos(2*pi*F*t));
cosineForZero = (A*cos(2*pi*F*t+pi));
for j = 1 : length(codigoSimbolos)
   if codigoSimbolos(j) == 1
      for k = 1 : length(cosineForOne);
         codigo(i) = cosineForOne(k);
         i = i +1;
      end
   else
      if codigoSimbolos(j) == 0
         for k = 1 : length(cosineForZero);
            codigo(i) = cosineForZero(k);
             i = i +1;
         end
      else
            i = i +100;
      end

   end
end
return
```

b)

Nesta alínea do projecto é recebido por parâmetro um sinal x, e tendo como base os vectores utilizados para codificação e efectuada a descodificação para bits e posteriormente de bits para as letras correspondentes a cada conjunto de bits, conforme a tabela dada no enunciado.

```
function ret = receptor(x, Tb, A, f)

chaves = 'abcdefghijklmnopqrstuvwxyz ';
valores = {[0 1],[1 0 0 0],[1 0 1 0],[1 0 0], 0 ,[0 0 1 0],[1 1 0],[0 0
0 0],[0 0],[0 1 1 1],[1 0 1],[0 1 0 0],[1 1],[1 0],[1 1 1],[0 1 1 0],[1
1 0 1],[0 1 0],[0 0 0],1,[0 0 1],[0 0 0 1],[0 1 1],[1 0 0 1],[1 0 1
1],[1 1 0 0],[1 0 0 1 0]};
aux = [];
z = 0;
w = 0;
ret= [];
t = 0: Tb/99: Tb;
bit1 = (A * cos (2 * pi * f * t));
bit0 = (A * cos (2 * pi * f * t + pi));
nBits = length(x) / 100;
for i = 1 : nBits
    for j = 1 : 100
        z = z+ (x((i-1)*100+j) - bit0(j));
    end
    if z == 0
        aux= [aux 0];
    else
    for j = 1 : 100
        w = w+ (x((i-1)*100+j) - bit1(j));
    end
    if w == 0
        aux = [aux 1];
    else
        for c = 1 : length(valores)
            valoresl = length(valores(c));
            auxl = length(aux);
            if length(aux) == length(cell2mat(valores(c)))
                if aux == cell2mat(valores(c))
                    ret = [ret chaves(c)];
                    aux = [];
                    break
                end
            end
        end
    end
    end
    z = 0;
    w = 0;
end
return
```

Conclusão

Conclui-se que houve partes do trabalho que podiam ter ficado melhor estruturadas, como é o caso do 3.b que não foi feito por módulos como o 3.a, e como é o caso do 1.2 que podia ter ficado com períodos configuráveis.