PROVA 1 - PROCESSAMENTO DE SINAIS BIOLÓGICOS

Davi de Alencar Mendes (dmendes@aluno.unb.br) - 16/0026415

Engenharia Eletrônica, UnB-FGA, Brasília, Brasil

Questão 1 - *QMF - Quadrature Mirror Filter Banks*

A - Objetivo da filtragem QMF

Inicialmente, a filtragem de decomposição tem como objetivo separa as componentes em frequência, discriminando entre aproximação (passa-baixas) e detalhes (passa-altas). Essa filtragem deve ser realizada de tal maneira que não haja perda de informação. Em termos da síntese, a filtragem tem o papel crucial em anular o termos de *alias*, tornando a transformada linear e invariante no tempo (LTI).

B - Critérios de Reconstrução Perfeita

Inicialmente tomamos como exemplo o QMF disposto a seguir (figura 1).

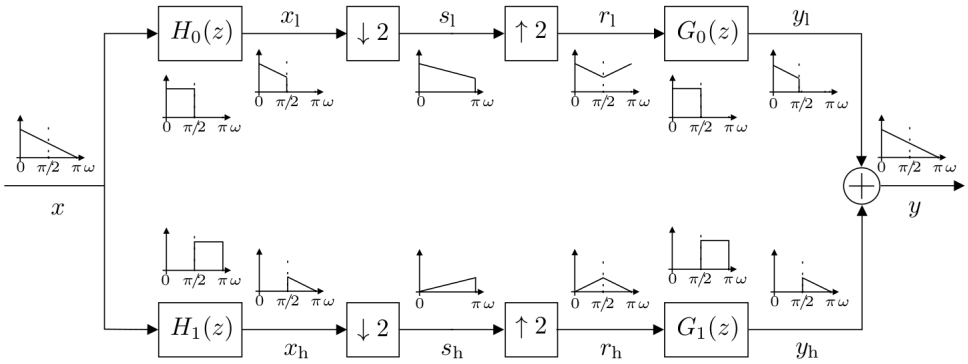


Fig. 1. QMF de 1 nível.

Usando as identidades nobres dos blocos sub-amostradores e sobre-amostradores é possível obter as equações no domínio z para os pontos do QMF.

$$\begin{aligned} x_1 &= H_0(z)X(z) \\ s_1 &= \frac{1}{2}[x_1(z^{1/2}) + x_1(-z^{1/2})] \\ s_1 &= \frac{1}{2}[H_0(z^{1/2})X(z^{1/2}) + H_0(-z^{1/2})X(-z^{1/2})] \\ r_1 &= s_1(z^2) \\ r_1 &= \frac{1}{2}[H_0(z)X(z) + H_0(-z)X(-z)] \\ y_1 &= \frac{1}{2}[G_0(z)H_0(z)X(z) + G_0(z)H_0(-z)X(-z)] \end{aligned} \tag{1}$$

De maneira análoga:

$$y_h = \frac{1}{2}[G_1(z)H_1(z)X(z) + G_1(z)H_1(-z)X(-z)] \tag{2}$$

Dividindo as equações em termos de elementos variantes no tempo e invariantes e aplicando as condições de reconstrução perfeita obtemos:

$$\begin{aligned} G_0(z)H_0(z) + G_1(z)H_1(z) &= 2 \cdot A \cdot z^{-d} \\ G_0(z)H_0(-z) + G_1(z)H_1(-z) &= 0 \end{aligned} \tag{3}$$

Sendo estas as condições de reconstrução perfeita para o QMF de 1 nível.

C - Inversão dos filtros de síntese e análise

Para a reconstrução perfeita foram obtidas um sistema de 2 equações e 4 filtros a serem obtidos. O problema original já apresenta infinitas soluções e ao adicionar mais uma equação para a inversão dos filtros de síntese e decomposição não altera o conjunto solução do problema.

D - Classificação do conjuntos de filtros

Iniciamos a análise computando os zeros para o filtro H_a sendo eles $1.02\angle\pm 2.74°$, $0.944\angle 0°$ e $0.3253\angle\pm 151.99°$. Pelo posicionamento dos zeros é possível constatar que se trata do filtro passa-altas de decomposição.

A inspeção do filtro H_b mostra que trata-se de uma versão de ordem reversa e cópia modulada (*reversed order & modulated copy*) de H_a, ou seja, $h_b(n) = (-1)^n h_a(6-n)$. Adicionalmente $G_a(z) = -H_a(-z)$ e $G_b(z) = H_b(-z)$. Finalmente, pode-se concluir que se trata de um conjunto de reconstrução perfeita já que as condições de inversão dos coeficientes para os filtros de síntese indicam que tratam de versões passa-baixa e passa-altas para os filtros de decomposição.

Logo, H_a é o filtro passa-alta de decomposição e G_b é o filtro passa-baixa de síntese. H_b é o filtro passa-baixa de decomposição de G_a é o filtros passa-altas de síntese. Apesar de não estar citado, é de conhecimento do autor que os respectivos filtros são da família Daubechies (db3).

E - Ortogonalidade do conjunto de filtros

Para provar a ortogonalidade usamos $\langle g_i(n)g_j(n)\rangle = \delta(i-j)$ para $i,j = \{0,1\}$. Iterando para i e j devemos encontrar:

$$\begin{aligned} \sum_n g_0^2(n) &= \sum_n g_1^2(n) = 1 \\ \sum_n g_0(n)g_1(n) &= 0 \end{aligned} \tag{4}$$

Aplicando para $g_0(n) = g_a(n)$ e $g_1(n) = g_b(n)$:

$$\begin{aligned} \sum_n g_0^2(n) &= 0.3327^2 + 0.8069^2 + 0.4599^2 - 0.1350^2 - 0.0854^2 + 0.0352^2 = 1 \\ \sum_n g_0(n)g_1(n) &= 0.0117 + 0.0689 - 0.0621 + 0.0621 - 0.0689 - 0.0117 = 0 \end{aligned} \tag{5}$$

Provamos que o conjunto é ortogonal e ortonormal. Por ser ortonormal também é bi-ortogonal. O resultado é análogo para os filtros de síntese.

F - Diagrama Decomposição QMF de 3 níveis

Utilizando as identidades nobres é possível reduzir o QMF de 3 níveis para a representação mostrada na figura 2.

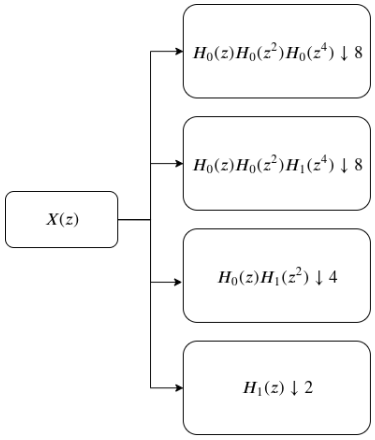


Fig. 2. Decomposição QMF - 3 níveis

G - Diagrama Síntese QMF de 3 níveis

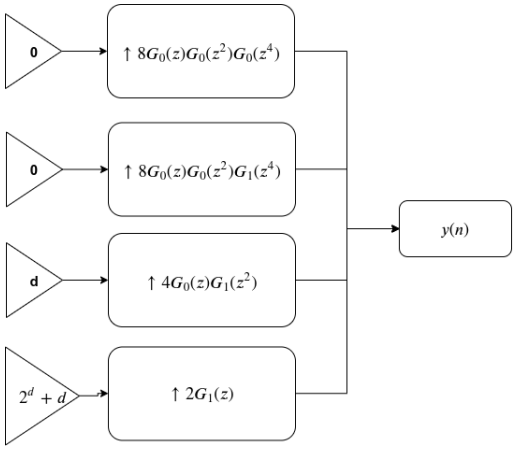


Fig. 3. Síntese QMF - 3 níveis

H - Resposta em Frequência dos Filtros de Decomposição

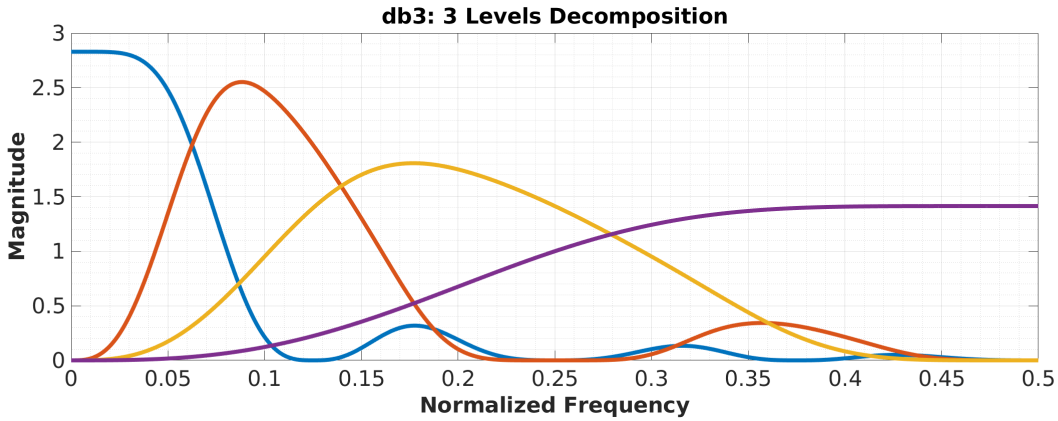


Fig. 4. Resposta em frequência - 3 níveis

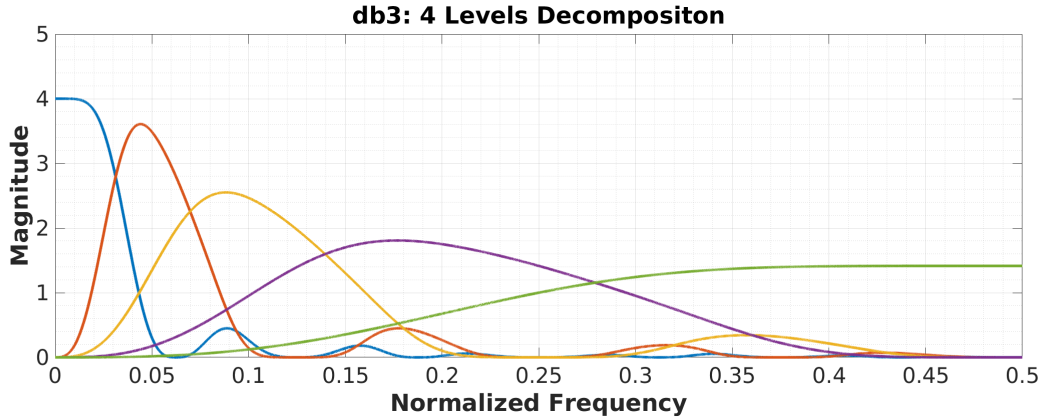


Fig. 5. Resposta em frequência - 4 níveis

I - Discussão acerca do item H

Para as diferentes bandas de frequência mostradas é possível perceber que as bandas de baixa frequência são mais estreitas, algo que indica um comportamento de maior resolução no tempo na medida que bandas de maior frequência são mais espalhadas e superpostas mostrando que há maior resolução no tempo. A representação multinível é interessante por ressaltar tanto as características em frequência como temporais já que cada nível de decomposição gera uma representação diferente.

J - Escolha de conjunto de filtros

É interessante lembrar que a adoção do conjunto de filtros é uma representação transformada na qual adotam-se bases para o sinal. Nesse sentido, há bases que são mais interessantes por ressaltar características particulares do sinal. Em especial para as wavelets, que codificam diferenças, é crucial ter conjuntos variados que sejam adaptáveis para diferentes cenários de aplicações.

K - Comparativo entre filtros

A família de wavelets de Daubechies (db) possui propriedades de assimetria, ortogonalidade e biortogonalidade. A família de Coiflets apresenta quasi-simetria, ortogonalidade e biortogonalidade. A família de Symlets mantém as propriedades das

Coiflets. A família Biortogonal é simétrica e biortogonal.

Questão 2 - Análise de Sinais de ECG & EMG

A - Transformada de Fourier dos Sinais

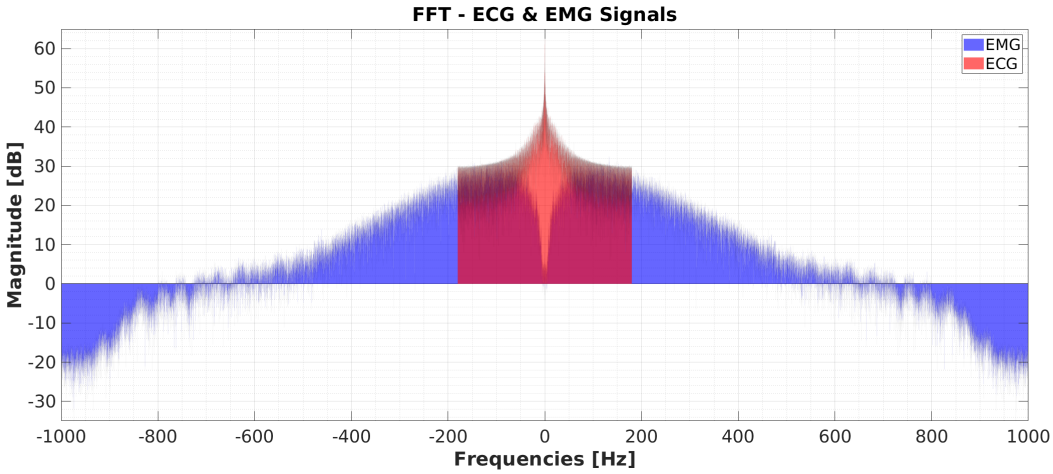


Fig. 6. Transformada de Fourier para ECG & EMG

B - Largura de Banda: ECG vs EMG

Tomando como ponto de partida a visualização gráfica de 6, pode-se perceber que o sinal de ECG está contido majoritariamente entre 0-50 Hz. Já para o sinal de EMG percebe-se que sua largura de banda estende-se até 700 Hz com magnitude considerável. Claramente o sinal de ECG apresenta variações mais lentas no tempo na medida que o sinal de EMG apresenta variações mais aceleradas.

C - Percentual de Energia por Banda

Foi computado um percentual de 88% da energia total para a banda de 2-150 Hz no sinal de EMG. Ademais, o sinal de ECG apresenta 74% da energia total para a banda de 2-40 Hz em sua versão com valor médio nulo. O procedimento de remoção do valor médio foi realizado já que os sinais de ECG e EMG apresentavam valores médios distintos, algo que poderia levar a uma análise errônea da energia total do sinal.

D - Discussão a respeito do item C

Por apresentar variações mais rápidas no domínio do tempo é esperado que ocorra uma maior largura de banda do sinal de EMG considerado.

E - Formação de Ruído

Durante a aquisição de sinais eletrofisiológicos é comum ocorrer contaminação com o sinal presente da rede elétrica (60Hz) já que, geralmente, o ambiente de coleta está sujeito a presença de outros equipamentos elétricos conectados em sua proximidade. Para o sinal considerado é possível notar que há um pico em 60 Hz com uma amplitude de +3 dB em relação ao nível da vizinhança.

A linha de base se dá em razão de sinais de baixa frequência proveniente do sistema respiratório (movimento muscular da respiração). O sistema de aquisição também pode gerar um offset de entrada na amostragem dos dados.

F - Filtragem com a DFT

Para remover a linha de base foi escolhido filtrar entre 0-5 Hz e 58-62 Hz para remoção da interferência da rede elétrica. Considerando a versão não deslocada da FFT, podemos calcular o índice correspondente a uma frequência usando $index = round(f_c * N/fs)$ no qual f_c denota a frequência desejada e N o comprimento correspondente da transformada do sinal. Para as frequências negativas utiliza-se $index = round((f_s - f_c) * N/fs)$ já que há a simetria na representação da magnitude. Para o sinal, foi calculada a FFT com 32768 pontos (próxima potência de 2 em relação ao comprimento do sinal de EMG) e foram eliminados os índices entre 1-456/32313-32768 para as frequências de 0-5 Hz e 5280-5644/27126-27409 para as frequências de 58-62 Hz.

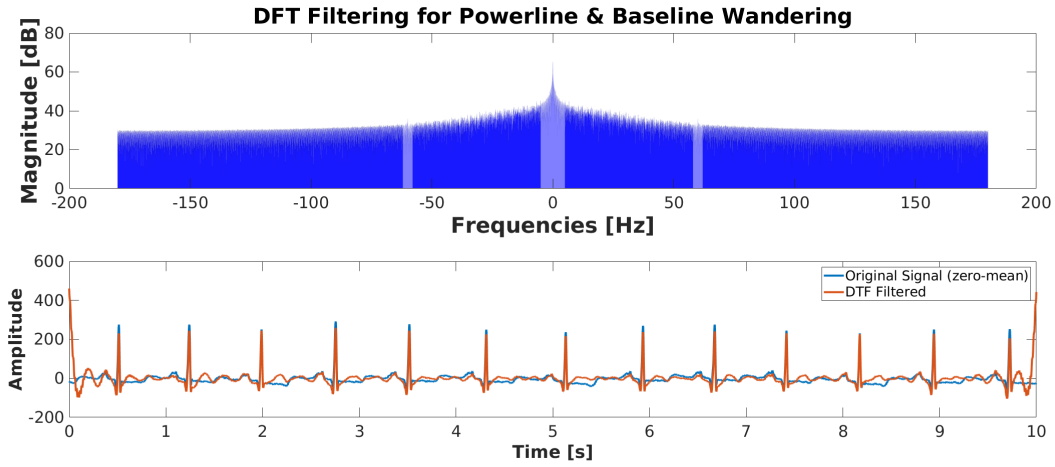


Fig. 7. DFT do Sinal e versão filtrada via DFT sobreposta a DFT original & Sinais no domínio do tempo filtrados

G - Filtragem FIR

Após algumas iterações de análise visual dos sinais filtrados, foi considerado que filtrar com um filtro de ordem 150, sendo passa altas para 8 Hz e rejeita faixa entre 56 e 64 Hz. Os resultados da filtragem são expostos na figura 7. A resposta em frequência pode ser vista na figura 8.

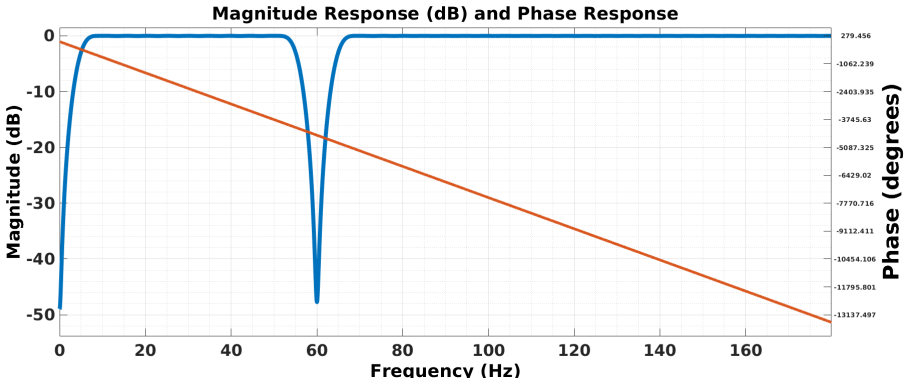


Fig. 8. Resposta em frequência e fase para o filtro FIR concebido

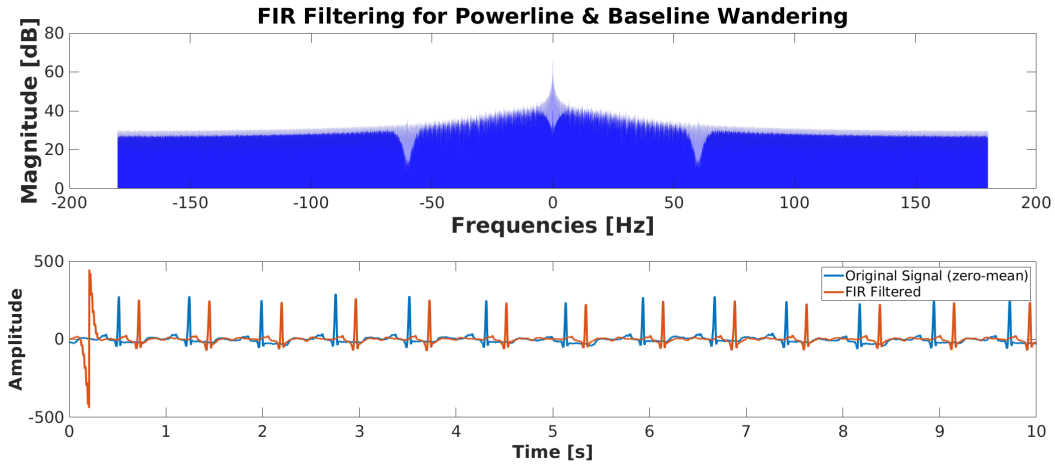


Fig. 9. DFT do Sinal e versão filtrada via FIR sobreposta a DFT original & Sinais no domínio do tempo filtrados

Questão 3 - Transformada de Fourier & Wavelets

A - Transformada de Fourier do Sinal de EMG

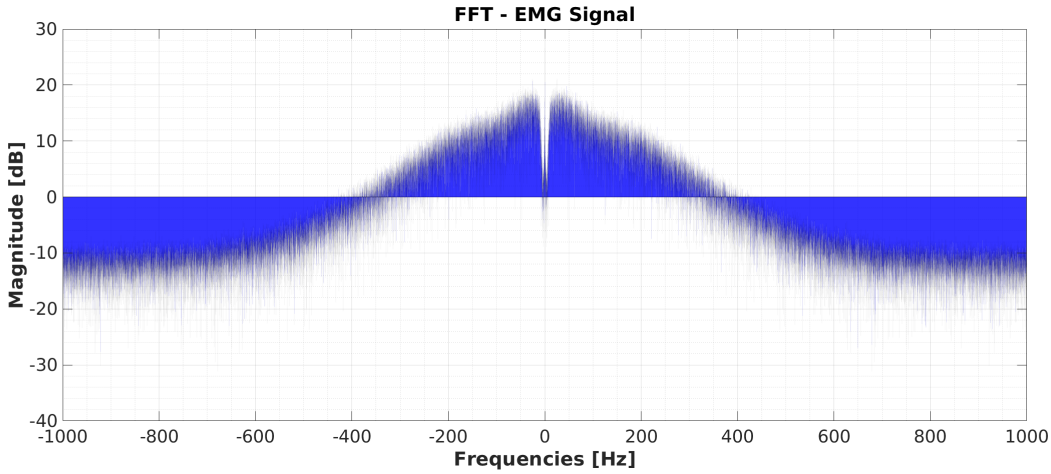


Fig. 10. Magnitude da Transformada de Fourier do Sinal de EMG

B - Adição de Eventos Senoidais de Alta Frequência

Para adicionar os eventos senoidais de alta frequência foi desenvolvida uma função que recebe o sinal de entrada, taxa de amostragem, ponto inicial do evento e duração em segundos além das frequências para as harmônicas e o percentual de energia máximo relativo a energia do sinal durante a duração do evento. **É importante ressaltar que os eventos (200 senoides de frequências linearmente espaçadas) foram adicionados em intervalos de 1 segundos nos segundos: (1,2,3) do sinal de EMG.**

```
1 %% aditiveHarmonicNoise: Aditive Harmonic Noise with len_s s. duration for freq
      frequencies with limited max energy
2 function [x] = aditiveHarmonicNoise(x, fs, startp, len_s, freq, p_energy)
```

```
        len = round(len_s*fs);
        endp = startp + len;

        % Obtain RMS value for the segment
        max_energy = p_energy*rms(x(startp:endp));
        % Obtaind Max Amplitude for each sinusoid
        max_amp = (sqrt(2)*max_energy) / (sqrt(length(freq)));

        % Generate Noise
        sn = zeros(1, endp-startp+1);
        tn = linspace(0, len_s, endp-startp+1);
        for n = 1:length(freq)
                ramp = -0 + (max_amp-0)*randn;
                sn = sn + ramp*sin(2*pi*freq(n)*tn);
        end
        % Add Noise to signal
        x(startp:endp) = x(startp:endp) + sn(:);
end
```

C - Transformada de Fourier para EMG com Eventos

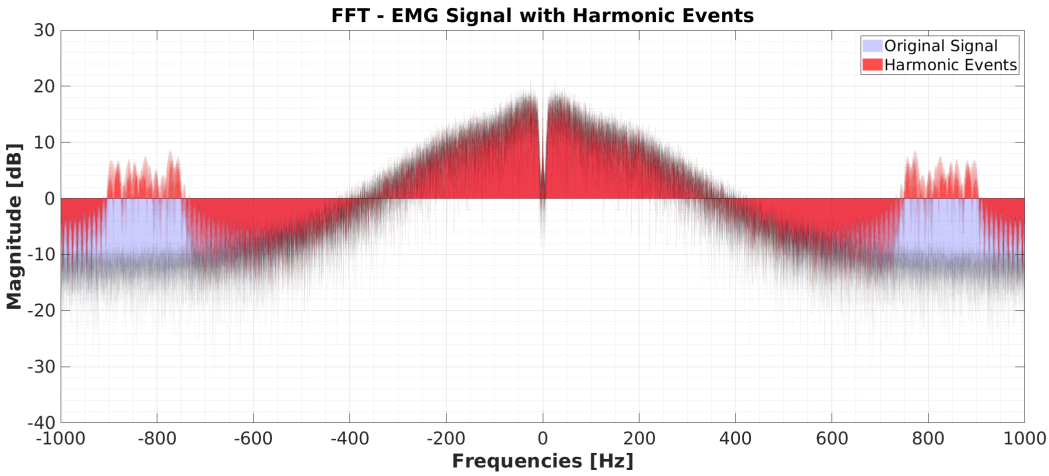


Fig. 11. Magnitude da Transformada de Fourier do Sinal de EMG & Adição de Eventos Senoidais de Alta Frequência

Comparando a figura 10 com 11 é possível notar que há conteúdos de alta frequência presentes no sinal mas é impossível distinguir o instante em que ocorrem os eventos no tempo. Embora a informação acerca dos instantes em que tais eventos ocorrem esteja embutida na fase da transformada este não é um dado de fácil interpretação.

D - Transformada Discreta de Wavelets: Daubechies 2 & Discrete Meyer com 3 níveis

Para observar os conteúdos de alta frequência do sinal deve-se observar o comportamento presente na menor escala para as figuras 12 e 13. Diferentemente da situação presente na FT, a decomposição multinível distingue claramente os eventos de alta frequência na menor escala nas posições 1000, 2000 e 3000. Nessas localizações encontram-se coeficientes com alto percentual de energia para toda a escala analisada.

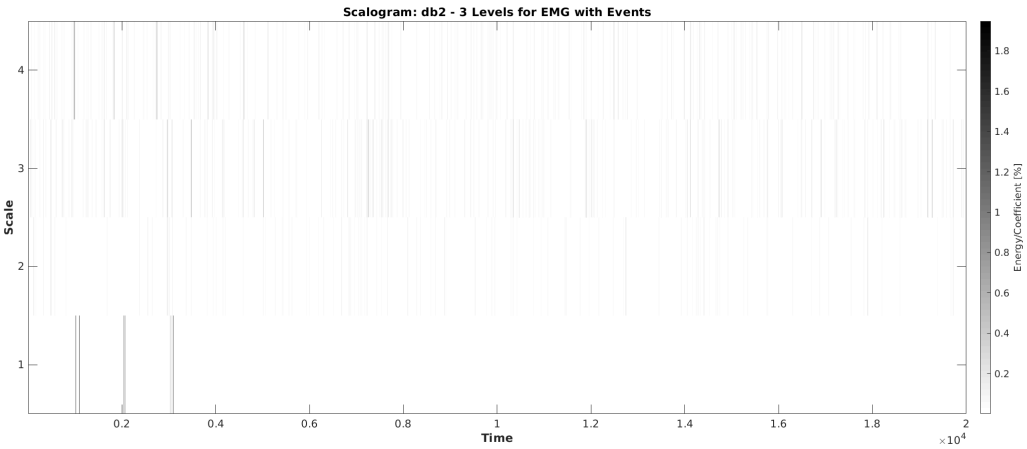


Fig. 12. Escalograma: db2 com 3 níveis

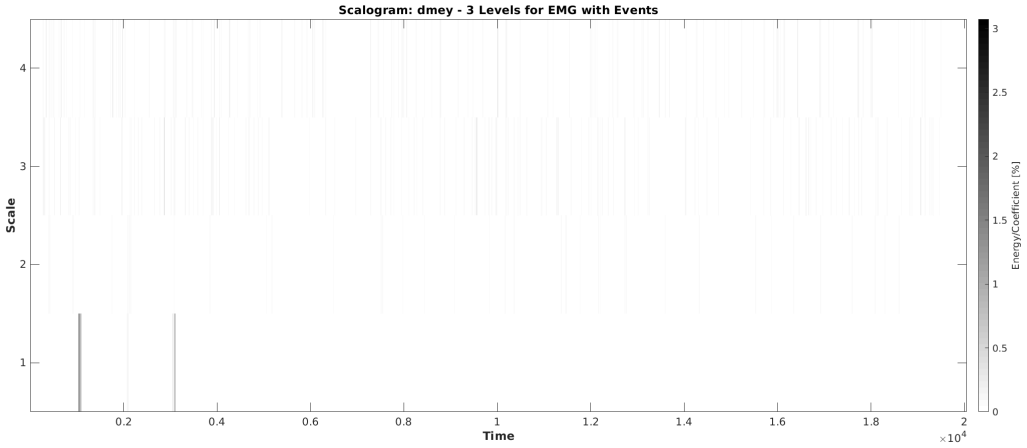


Fig. 13. Escalograma: dmey com 3 níveis

E - Comparativo: Espectrograma (STFT) vs. Escalograma (DWT)

A figura 14 mostra o espectrograma para o sinal com a utilização da janela de Hamming com 150 ms de duração e sobreposição de 50% entre janelas. Para o cômputo da FFT foram utilizados 512 pontos que é a próxima potência de 2 em relação ao comprimento escolhido para a janela. A figura 15 expõe os resultados da decomposição com a família Discrete Meyer para 4 níveis. É evidente que a representação multinível detalhe muito bem os conteúdos em frequência, incluindo os eventos adicionados na menor escala. Em geral, acredito que o espectrograma seja uma ferramenta de mais fácil interpretação visual porém é preciso configurar diversos parâmetros para obter um resultado coerente. Nesse sentido, a decomposição wavelets oferece maior versatilidade.

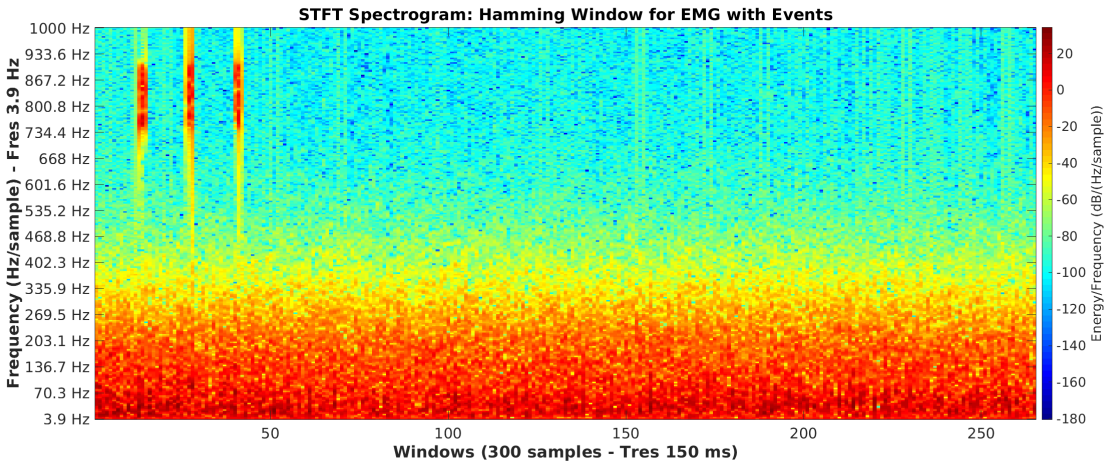


Fig. 14. Espectrograma: Janela de Hamming com 150 ms (sobreposição de 50%) e FFT com 512 Pontos

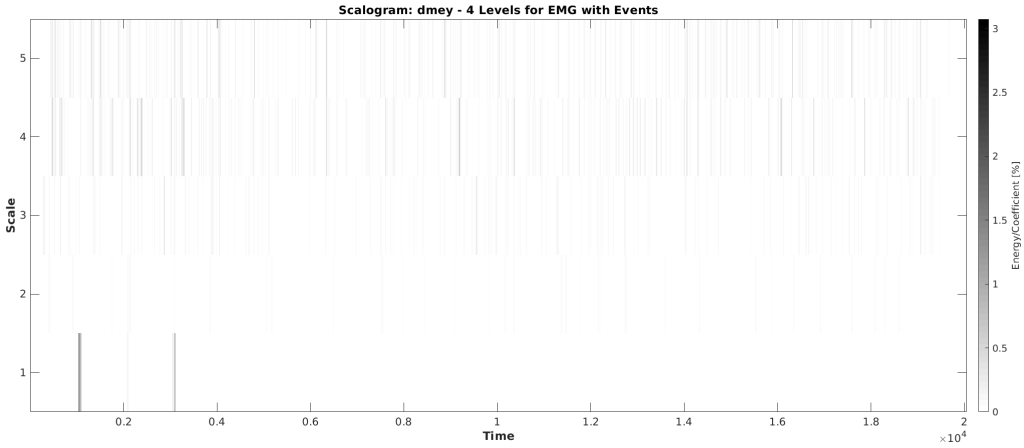


Fig. 15. Escalograma: dmey com 4 níveis

Questão 4 - Compressão de Sinais de ECG com DWT

0.1. A - Função para decomposição wavelet e quantização

```
%% dwtEcgQuant: DWT Compression scheme using a given percentage (NM%) of the
   transform coefficients
function [xq, len] = dwtEcgQuant(x, filter_type, levels, NM)
        xq = cell(1, length(NM));

        % Treat Inputs
        if class(filter_type) == 'char'
                [h0, h1, ~, ~] = wfilters(filter_type);
        elseif class(filter_type) == 'cell'
                h0 = filter_type{1};
```

```
                h1 = filter_type{2};
        else
                error('Error in filter_type! Provide {h0, h1} as a cell array or use
                    MATLAB std filters!');
        end

        % Perform Decomposition
        [xd, xdc, ~] = qmf_decomposition(x, h0, h1, levels);
        % Get length of each level
        len = cellfun(@length, xdc);
        % Sort
        [xd_sort, I] = sort(xd, 'descend', 'ComparisonMethod', 'abs');
        mpI = 1:1:length(xd);
        mpI = mpI(I); % map indexes according to sorted data
        % Quantize using Nm[%]
        NM = round(length(x) * (NM/100));
        for n = 1:length(NM)
                tmp = zeros(length(xd),1);
                tmp(mpI(1:NM(n))) = xd_sort(1:NM(n));
                xq{n} = tmp;
        end
end
```

B - Síntese dos sinais de ECG quantizados

```
%% dwtEcgRec: DWT Reconstruction for the dwtEcgQuant function
function [rx] = dwtEcgRec(xq, len, filter_type)
        rx = cell(1, length(xq));

        % Treat Inputs
        if class(filter_type) == 'char'
                [h0, h1, g0, g1] = wfilters(filter_type);
        elseif class(filter_type) == 'cell'
                h0 = filter_type{1};
                h1 = filter_type{2};
                g0 = filter_type{3};
                g1 = filter_type{4};
        else
                error('Error in filter_type! Provide {h0, h1} as a cell array or use
                    MATLAB std filters!');
        end

        % Split decomposition array
        levels = length(len) - 1;
        qxdc = cell(1, length(xq));
        for n = 1:length(xq) % Iterate over all signals
                qxdc{n} = cell(1,levels);
                for l = 1:length(len) % Split for each level
                        qxdc{n}{l} = xq{n}(1:len(l));
                        xq{n}(1:len(l)) = [];
                end

                % Perform QMF Reconstruction
```

```
28                  [rx{n}, ~, ~] =  qmf_reconstruction(qxdc{n}, h0, h1, g0, g1);
29          end
30  end
```

C - Visualização: SNR vs. NM %

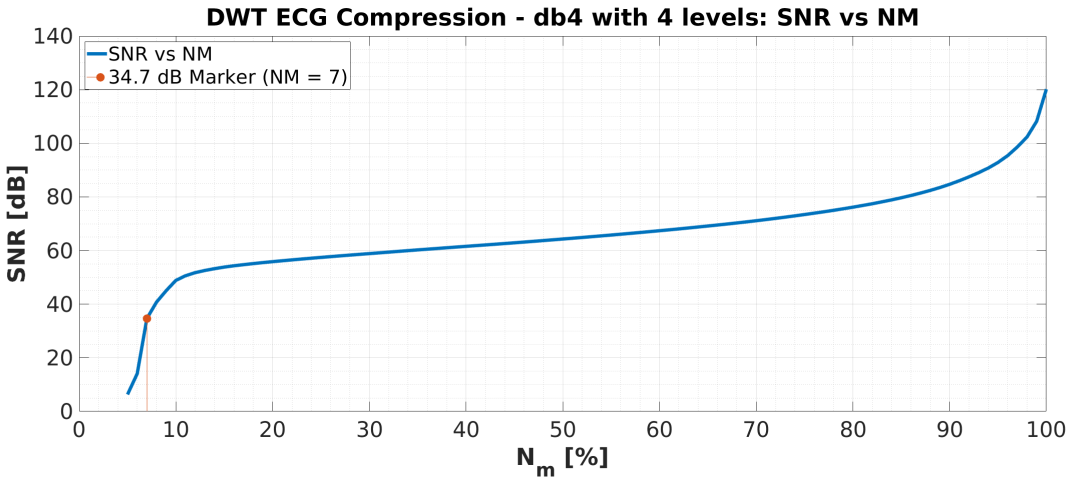


Fig. 16. SNR vs. NM para Daubechies 4 (db4) com 4 níveis

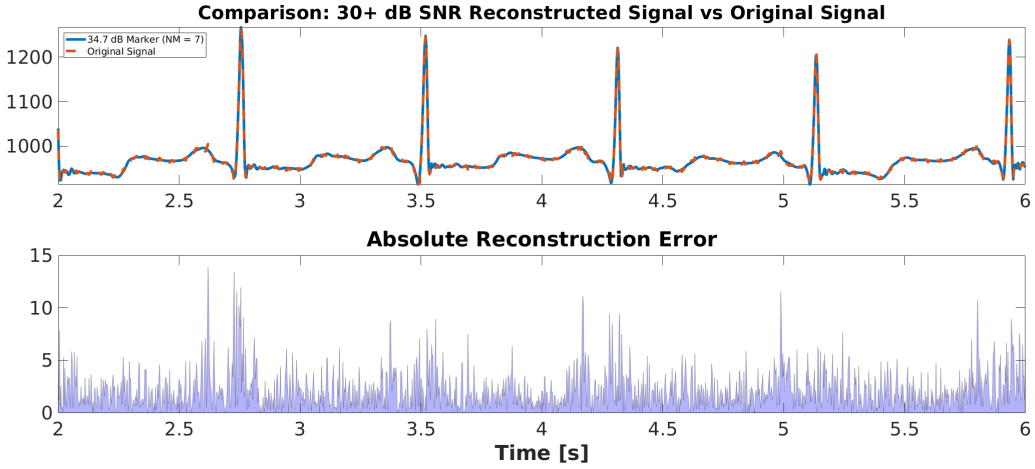


Fig. 17. Erro absoluto de reconstrução e comparação no domínio do tempo

D - Estratégia para compressão de sinais de ECG

Baseado na decomposição multinível utilizada sabemos que há uma elevada compactação de energia, gerando uma grande quantidade de elementos nulos em sequência para a versão quantizada da transformada. Nesse sentido, é possível utilizar a transformada quantizada aliada a um código de corrida de zeros (*zero run-length*) para codificar os elementos nulos. É preciso enviar os filtros utilizados na decomposição e síntese (h0, h1, g0 e g1) além do comprimento de cada nível de decomposição de wavelets.

Questão 5 - Extração de Características & Classificação SVM

A - Extração de Características - Janelas de 5 segundos para toda a banda do sinal

A função desenvolvida está nos anexos da prova! **Para os valores presentes nas figuras 18 e 19 foram utilizados os sinais 1 de cada conjunto de gravações.**

	RMS	RMV	STD
A	27.36	18.50	27.04
B	31.06	21.17	31.02
C	33.32	21.92	31.71
D	25.68	17.76	20.55
E	301.07	202.91	300.22

Fig. 18. Características Temporais extraídas - Valor Eficaz, Valor Médio Retificado e Desvio Padrão (Valores Absolutos)

	MEDFREQ	MEANFREQ	MODALFREQ
A	4.66	6.64	1.00
B	6.80	7.48	1.23
C	2.26	3.13	1.19
D	0.41	1.78	0.21
E	7.21	8.84	1.99

Fig. 19. Características Espectrais extraídas - Frequência Mediana, Média e Modal (Valores em Hz)

B - Extração de Características por Tipo de Onda ($\delta\,\theta\,\alpha\,\beta\,\gamma$)

Delta	Teta	Alfa	Beta	Gama
47.26	14.80	30.40	7.08	0.46
41.84	8.93	42.81	5.76	0.66
73.38	18.28	7.50	0.76	0.09
88.75	5.91	4.12	0.99	0.23
36.37	16.95	32.53	13.85	0.31

Fig. 20. Características Espectrais: Energia por tipo de ondas em valores percentuais em relação ao total de energia

C - Extração de Características com 10 Bandas

Para encontrar a frequência na qual há 95% da energia do sinal basta minimizar a equação da diferença entre o valor de 95% da energia total do sinal x e sua energia na banda $0 - f_m$ Hz.

```
fmfunc = @(fm) abs(0.95*bandpower(x, fs, [0 (length(x)-1)*fs/(2*length(x))]) -
    bandpower(x, fs, [0 fm]));
fm = fminsearch(fmfunc, 1); % Find fm starting at 1 Hz
```

Band1	Band2	Band3	Band4	Band5	Band6	Band7	Band8	Band9	Band10
34.66	12.89	12.51	6.79	6.67	17.02	4.54	1.79	1.58	1.56
26.42	13.35	5.54	4.93	4.70	15.03	20.58	7.78	1.27	0.41
24.44	16.07	17.21	14.50	6.36	8.19	5.35	3.29	2.26	2.33
66.50	10.03	8.22	4.65	3.41	2.17	1.94	1.35	0.99	0.73
1.40	25.82	16.05	6.55	5.65	6.90	10.49	11.66	13.53	1.96

Fig. 21. Características Espectrais: Energia por banda em valores percentuais em relação ao total de energia

D - Características Extraídas por Janela

RMS	RMV	STD
25.14	16.74	25.10
29.98	20.26	29.13
23.82	16.83	23.42
26.99	17.84	26.98
24.57	16.27	24.18
27.47	18.28	26.85
33.13	22.10	32.87
27.81	19.72	27.82

(a) A1

RMS	RMV	STD
29.62	20.68	29.58
31.83	21.62	31.74
28.55	19.03	28.55
33.70	23.83	33.54
39.54	26.55	39.49
31.17	20.97	31.13
26.15	17.74	26.17
27.94	18.90	27.96

(b) B1

RMS	RMV	STD
36.80	23.03	35.31
30.68	21.41	28.95
31.52	22.60	29.98
45.11	28.37	43.98
27.31	19.32	25.57
32.28	19.63	30.47
30.25	20.51	28.33
32.60	20.50	31.13

(c) C1

RMS	RMV	STD
27.46	18.99	23.02
25.99	17.28	20.78
25.31	17.84	19.43
22.07	16.63	16.75
25.83	17.70	20.17
24.09	16.27	19.01
27.34	18.43	23.30
27.32	18.93	21.90

(d) D1

RMS	RMV	STD
268.31	180.63	267.41
292.34	197.72	291.76
314.17	218.77	313.24
304.54	199.12	303.78
288.00	187.87	287.44
304.24	205.43	303.41
299.41	207.22	297.68
337.56	226.49	337.05

(e) E1

Fig. 22. Características Temporais para cada Gravação em cada janela - Valor Eficaz, Valor Médio Retificado e Desvio Padrão (Valores Absolutos)

E	MEDFREQ	MEANFREQ	MODALFREQ
548593.35	4.50	6.51	0.17
779976.18	2.15	5.17	0.17
492444.26	5.88	7.38	0.68
632118.14	5.79	7.08	0.17
523890.34	5.82	7.52	0.51
655169.10	4.18	6.73	0.17
952534.63	4.78	6.48	5.77
671541.69	4.14	6.28	0.34

(a) A1

E	MEDFREQ	MEANFREQ	MODALFREQ
761277.87	9.24	8.13	0.59
879455.92	4.15	6.43	0.59
707302.31	2.55	5.81	1.19
985713.48	6.80	7.24	0.25
1357358.35	9.42	7.72	0.17
843255.89	11.00	9.24	6.19
593719.68	5.74	7.47	0.68
677616.30	5.54	7.82	0.17

(b) B1

E	MEDFREQ	MEANFREQ	MODALFREQ
1175563.97	2.54	3.38	1.27
816996.82	3.43	3.67	2.38
862576.62	2.23	3.07	1.19
1766094.85	2.77	3.15	1.53
647266.87	1.74	3.31	0.17
904185.74	1.73	2.38	0.85
794478.35	1.50	2.94	0.68
922632.98	2.16	3.16	1.44

(c) C1

E	MEDFREQ	MEANFREQ	MODALFREQ
654701.95	0.38	1.77	0.17
586486.44	0.26	1.65	0.08
556113.13	0.21	1.64	0.08
422692.67	0.24	2.15	0.08
579302.51	0.21	1.38	0.08
503684.46	0.27	1.59	0.08
649001.89	1.49	2.25	1.02
647936.77	0.24	1.81	0.08

(d) D1

E	MEDFREQ	MEANFREQ	MODALFREQ
62486487.99	8.37	9.41	2.97
74180196.89	6.17	9.10	3.05
85675583.87	8.97	9.14	1.10
80500922.10	9.58	9.71	1.78
71993160.96	7.81	8.86	1.87
80342518.63	5.86	8.32	1.61
77813190.23	5.68	8.25	1.53
98907118.23	5.24	7.91	2.04

(e) E1

Fig. 23. Características Espectrais para cada Gravação em cada janela - Energia (Valor absoluto) & Frequência Mediana, Média e Modal (Valores em Hz)

E - Classificação com SVM

Para o classificador foram utilizadas as rotinas do MATLAB para 2 classes conforme o código a seguir usando reamostragem aleatória.

```
1 %% svm2ClassRandperm: Performs SVM Classification using random permutation for the
     features
2 function [svm, out_train, out_val, c_train, c_val, f_train, f_val] =
     svm2ClassRandperm(a, feat_p, feat_n)
3         rp = randperm(size(feat_p, 1));
4         n = round(a * size(feat_p, 1));
```

```
        m = size(feat_p, 1) - n;
        r_train = rp(1 : n);
        r_val = rp(n + 1 : end);
        f_train = [feat_p(r_train, :); feat_n(r_train, :)];
        c_train = [ones(n, 1); zeros(n, 1)];
        f_val = [feat_p(r_val, :); feat_n(r_val, :)];
        c_val = [ones(m, 1); zeros(m, 1)];
        svm = fitcsvm(f_train, c_train, 'KernelFunction', 'rbf', 'Standardize', true
            , 'ClassNames', [0,1]);
        out_train = predict(svm, f_train);
        out_val = predict(svm, f_val);
end
```

SVM em 1 rodada de treinamento e classificação: As figuras 24 a 26 mostram as matrizes de confusão em valores percentuais de acerto para cada classe. A primeira estratégia obteve melhores acertos para a segunda classe (D) porém a terceira estratégia obteve resultados mais uniformes para ambas as classes com mais de 95% de acerto em cada classe. **Para o conjunto de treinamento foram usados 80% dos dados disponíveis.**

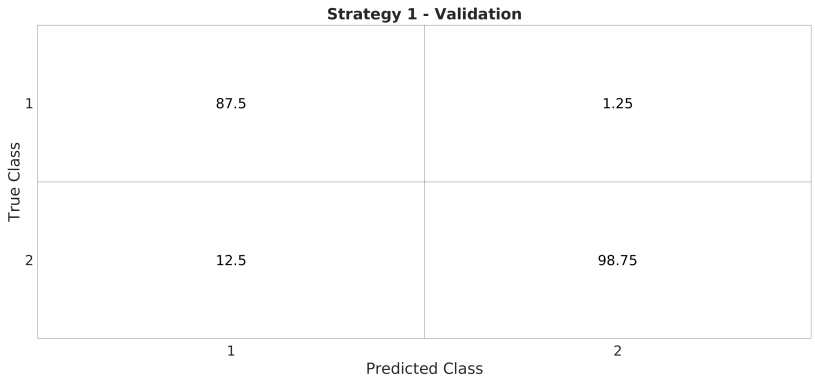


Fig. 24. *Confusion Matrix Chart:* Estratégia 1 - Energias de bandas ($\delta\,\theta\,\alpha\,\beta\,\gamma$)

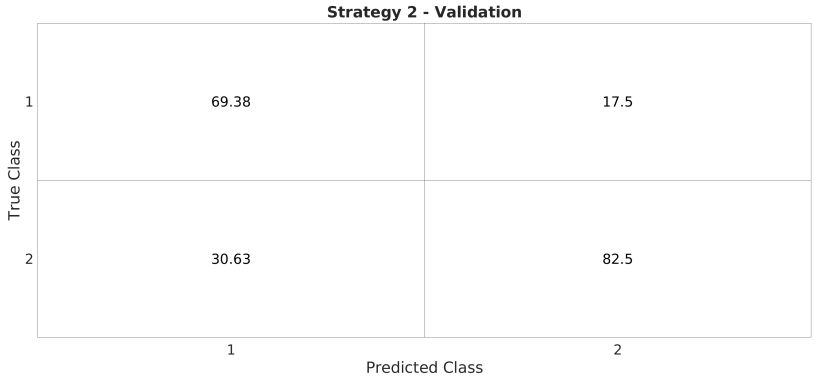


Fig. 25. *Confusion Matrix Chart:* Estratégia 2 - Energia em 10 bandas consecutivas

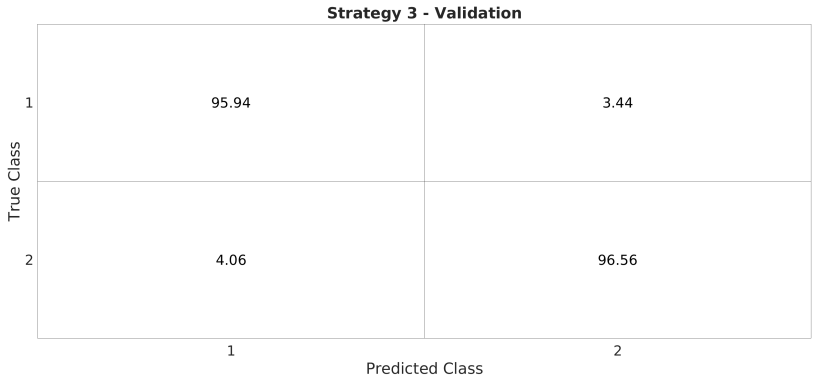


Fig. 26. *Confusion Matrix Chart:* Estratégia 3 - Valor Eficaz & Frequência Média, Mediana e Modal

F - SVM com Reamostragem Aleatória em 500 sessões

As figuras 27 e 28 dispõe os resultados obtidos para o classificador em 500 sessões de treinamento e validação com reamostragem aleatória. De maneira geral a estratégia 3 que utiliza a energia em bandas ($\delta\,\theta\,\alpha\,\beta\,\gamma$) resulta em melhores resultados para as métricas avaliadas. A estratégia 2 com energias em 10 bandas consecutivas, embora similar a estratégia 2, resulta em pouca convergência com índices bem menos expressivos. A estratégia 1 aproxima-se bastante dos melhores resultados por empregar frequências médias, modais e medianas além dos valores eficazes para as janelas consideradas.

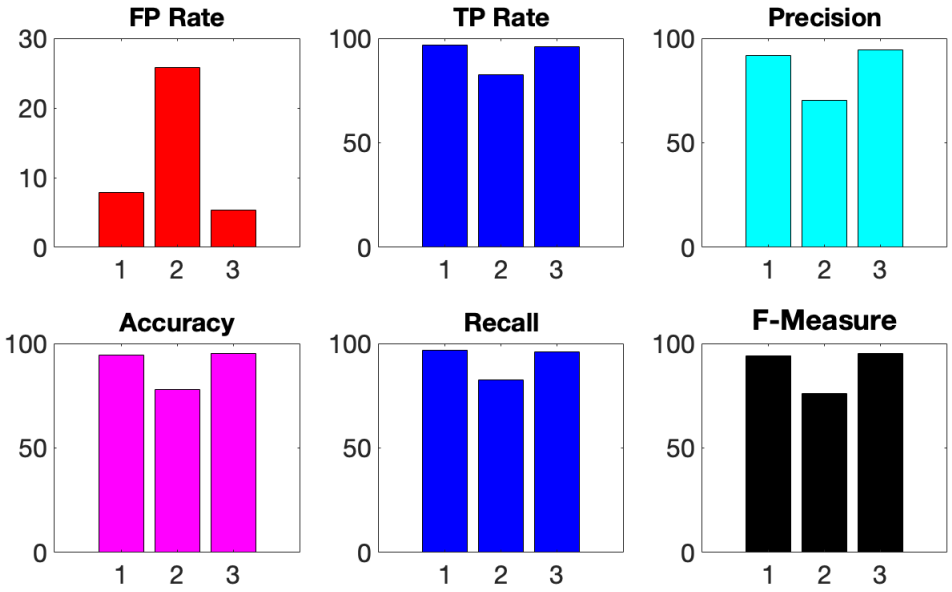
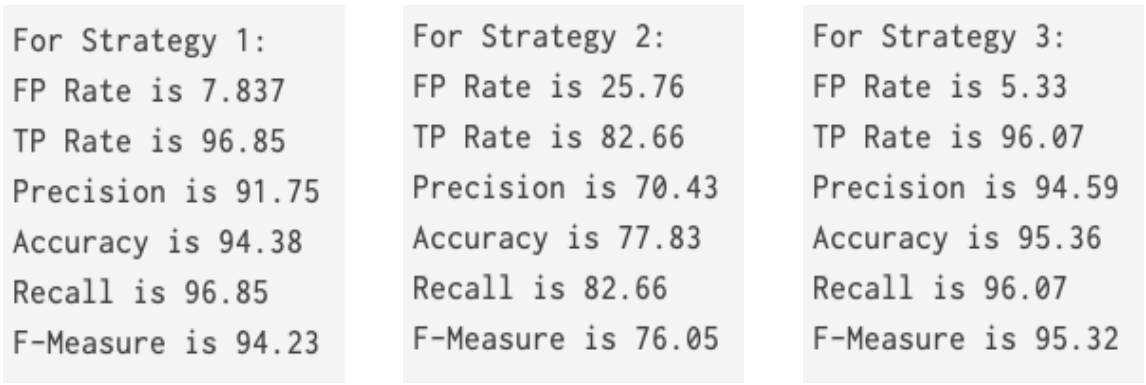


Fig. 27. Métricas para o Classificador Obtido



(a) E1 (b) E2 (c) E3

Fig. 28. Métricas para cada uma das estratégias utilizadas

Considerações Finais

O desenvolvimento da prova foi bastante proveitoso para consolidar conceitos fundamentais e aplicar diferentes técnicas de processamento de sinais biológicos para filtragem, codificação & compressão em sinais de ECG, extração & escolha de características e classificação com SVM. Estimo que o tempo total de desenvolvimento da prova tenha tomado em torno de 13 horas para conclusão das questões incluindo o tempo necessário para redigir a parte textual da prova. Finalmente, espero que tenha gostado dos resultados obtidos.

Davi Mendes

Anexos

Q2 - Rotina computacional para visualização dos dados e filtragem FIR & DFT

```
clear all; close all; clc;

% Load EMG & ECG Data
emg = load('emg_1.mat');
ecg = load('ecg_1.mat');

% Calculate FFT
if(length(emg.x) > length(ecg.x)); np2 = 2^nextpow2(length(emg.x)); else np2 = 2^
    nextpow2(length(ecg.x)); end
emg.X = abs(fftshift(fft(emg.x, np2)));
ecg.X = abs(fftshift(fft(ecg.x, np2)));
figure('units', 'normalized', 'outerposition', [0 0 1 1]);
area(linspace(-0.5*emg.fs, 0.5*emg.fs, np2), 10*log10(emg.X), 'FaceColor','b','
    FaceAlpha',.6,'EdgeAlpha',.015,'EdgeColor','k');
hold on;
grid on;
grid minor;
area(linspace(-0.5*ecg.fs, 0.5*ecg.fs, np2), 10*log10(ecg.X), 'FaceColor','r','
    FaceAlpha',.6,'EdgeAlpha',.015,'EdgeColor','k');
ylim([-35 65]);
legend('EMG', 'ECG');
set(get(gca,'XLabel'),'String','Frequencies [Hz]');
set(get(gca,'YLabel'),'String','Magnitude [dB]');
title('FFT - ECG & EMG Signals');
set(findall(gcf,'type','text'), 'FontSize', 32, 'fontWeight', 'bold');
set(gca,'FontSize',26);

% Calculate Band Power for ECG & EMG
ecg.tp = bandpower(ecg.x-mean(ecg.x), ecg.fs, [0 ecg.fs/2]); % Using zero-mean
    signal for a fair comparison!
ecg.bp = bandpower(ecg.x-mean(ecg.x), ecg.fs, [2 40]);
disp(['ECG power % between 2-40 Hz = ' num2str(100*(ecg.bp/ecg.tp),2) '%.']);
emg.tp = bandpower(emg.x, emg.fs, [0 emg.fs/2]);
emg.bp = bandpower(emg.x, emg.fs, [2 150]);
disp(['EMG power % between 2-150 Hz = ' num2str(100*(emg.bp/emg.tp),2) '%.']);

% DFT Filtering
figure('units', 'normalized', 'outerposition', [0 0 1 1]);
subplot(2,1,1);
area(linspace(-0.5*ecg.fs, 0.5*ecg.fs, np2), 10*log10(ecg.X), 'FaceColor','b','
    FaceAlpha',.5,'EdgeAlpha',.015,'EdgeColor','k');
hold on;
ecg.Xf = fft(ecg.x, np2);
ecg.Xf = dftFiltering(ecg.Xf, ecg.fs, [58 62]);
ecg.Xf = dftFiltering(ecg.Xf, ecg.fs, [0 5]);
area(linspace(-0.5*ecg.fs, 0.5*ecg.fs, np2), 10*log10(abs(fftshift(ecg.Xf))), '
    FaceColor','b','FaceAlpha',.8,'EdgeAlpha',.015,'EdgeColor','k');
set(get(gca,'XLabel'),'String','Frequencies [Hz]');
set(get(gca,'YLabel'),'String','Magnitude [dB]');
title('DFT Filtering for Powerline & Baseline Wandering');
```

```
set(findall(gcf,'type','text'), 'FontSize', 32, 'fontWeight', 'bold');
set(gca,'FontSize',24);
subplot(2,1,2);
plot(linspace(0, length(ecg.x)/ecg.fs, length(ecg.x)), ecg.x-mean(ecg.x), 'LineWidth', 3);
hold on;
xr = real(ifft(ecg.Xf));
xr = xr(1:10*ecg.fs);
plot(linspace(0, length(xr)/ecg.fs, length(xr)), xr , 'LineWidth', 3);
set(get(gca,'XLabel'),'String','Time [s]');
set(get(gca,'YLabel'),'String','Amplitude');
set(findall(gcf,'type','text'), 'FontSize', 32, 'fontWeight', 'bold');
set(gca,'FontSize',24);
l = legend('Original Signal (zero-mean)', 'DTF Filtered');
l.FontSize = 20;

% FIR Filtering
forder = 150;
low = 8/ecg.fs;
band = [56*2/ecg.fs 64*2/ecg.fs];
h = fir1(forder, [low band]);
hnd = fvtool(h,1);
hnd.Fs = ecg.fs;
hnd.FrequencyRange='[0, Fs/2)';
ecg.xf = filter(h,1, ecg.x);

figure('units', 'normalized', 'outerposition', [0 0 1 1]);
subplot(2,1,1);
area(linspace(-0.5*ecg.fs, 0.5*ecg.fs, np2), 10*log10(abs(fftshift(fft(ecg.xf, np2)))), 'FaceColor','b','FaceAlpha',.8,'EdgeAlpha',.015,'EdgeColor','k');
hold on;
area(linspace(-0.5*ecg.fs, 0.5*ecg.fs, np2), 10*log10(ecg.X), 'FaceColor','b','FaceAlpha',.4,'EdgeAlpha',.015,'EdgeColor','k');
set(get(gca,'XLabel'),'String','Frequencies [Hz]');
set(get(gca,'YLabel'),'String','Magnitude [dB]');
title('FIR Filtering for Powerline & Baseline Wandering');
set(findall(gcf,'type','text'), 'FontSize', 32, 'fontWeight', 'bold');
set(gca,'FontSize',24);

subplot(2,1,2);
plot(linspace(0, length(ecg.x)/ecg.fs, length(ecg.x)), ecg.x-mean(ecg.x), 'LineWidth', 3);
hold on;
plot(linspace(0, length(ecg.xf)/ecg.fs, length(ecg.xf)), ecg.xf , 'LineWidth', 3);
set(get(gca,'XLabel'),'String','Time [s]');
set(get(gca,'YLabel'),'String','Amplitude');
set(findall(gcf,'type','text'), 'FontSize', 32, 'fontWeight', 'bold');
set(gca,'FontSize',24);
l = legend('Original Signal (zero-mean)', 'FIR Filtered');
l.FontSize = 20;


%% dftFiltering: function description
function [X] = dftFiltering(X, fs, fc)
```

```
        N = length(X);
        k1 = round(fc(1) * N / fs + 1); % Considering 0 <= fc <= fs/2
        k2 = round(fc(2) * N / fs + 1);
        k3 = round((fs-fc(2)) * N / fs + 1);
        k4 = round((fs-fc(1)) * N / fs + 1);
        if k4 > length(X)
                k4 = length(X);
                k3 = k3-1;
        end % Exception for low cut freq. = 0 Hz (k4 overflows to length(X)+1)
        X(k1:k2) = 0;
        X(k3:k4) = 0;
        disp(['For ' int2str(fc(1)) '-' int2str(fc(2)) 'Hz, using: ' int2str(k1) '
           to ' int2str(k2) ' and ' int2str(k3) ' to ' int2str(k4) '.']);
end
```

Q3 - Espectrograma (STFT)

```
%% stftSpectrogram: Spectrogram using Short-Time FT
function [varargout] = stftSpectrogram(x, fs, win, loverlap, nfft, zeropad)
        % Treat Inputs
        if ~exist('loverlap', 'var') || isempty(loverlap)
                loverlap = 0;
        end
        x = double(x(:));
        win = double(win(:));
        lwin = length(win);
        if ~exist('nfft', 'var') || isempty(nfft)
                nfft = lwin;
        end
        if ~exist('zeropad', 'var')
                zeropad = false;
        end

        % Reshape Signal without Zero Padding
        x = reshapeOverlap(x, lwin, loverlap, zeropad);

        % Split Apply Function
        spect_vals = @(col) 20*log10(abs(fftshift(fft(win.*col, nfft))).^2);
        x = splitapply(spect_vals, x, 1:size(x,2));
        % Remove the bottom half of the spectrogram
        x = x(1:floor(end/2),:);

        % Provide Outputs (Plot or Matrix)
        switch nargout
                case 0 % Plot Spectrogram
                        figure('units','normalized','outerposition',[0 0 1 1]);
                        imagesc(x);
                        colormap jet;
                        hcb = colorbar;
                        ylabel(hcb, 'Energy/Frequency (dB/(Hz/sample))');
                        set(get(gca,'YLabel'),'String', ['Frequency (Hz/sample) -
                           Fres ' num2str(fs/nfft, 2) ' Hz']);
                        ticks = linspace(1, size(x,1), 16);
```

```
                        yticks('manual');
                        yticks(ticks);
                        fv_label = arrayfun(@num2str, round(flip(ticks)*fs/nfft, 1),
                            'UniformOutput', false);
                        labelf = @(freq) [freq ' Hz'];
                        fv_label = cellfun(labelf, fv_label, 'UniformOutput', false)
                            ;
                        yticklabels('manual');
                        yticklabels(fv_label);
                        set(get(gca,'XLabel'),'String',['Windows (' int2str(lwin) '
                            samples - Tres ' num2str(1000*lwin/fs, 5) ' ms)']);
                        set(findall(gcf,'type','text'), 'FontSize', 22, 'fontWeight'
                            , 'bold');
                        set(gca,'FontSize',22);
                case 1 % Return Spectrogram as matrix
                        varargout{1} = x;
                otherwise
                        error('Error in varargout! Provide the right number of
                            outputs.');
        end
end
```

Q3 - Escalograma (DWT)

```
%% dwtScalogram: function description
function [varargout] = dwtScalogram(x, fs, filter_type, levels)
        % Treat Inputs
        if class(filter_type) == 'char'
                [h0, h1, ~, ~] = wfilters(filter_type);
        elseif class(filter_type) == 'cell'
                h0 = filter_type{1};
                h1 = filter_type{2};
        else
                error('Error in filter_type! Provide {h0, h1} as a cell array or use
                    MATLAB std filters!');
        end

        % Perform Decomposition
        [~, xdc, ~] = qmf_decomposition(x, h0, h1, levels);

        % Interpolate frame
        len_xdc = cellfun(@length, xdc);
        max_len = max(len_xdc(:));
        frame = zeros(levels+1,max_len);
        for l = 1:levels+1
                xi = linspace(0, length(x)*fs, length(xdc{l}));
                xq = linspace(0, length(x)*fs, max_len);
                y = xdc{l}.^2;
                y = 100 * y/sum(y);
                frame(l,:) = interp1(xi, y, xq, 'nearest');
        end

        figure('units','normalized','outerposition',[0 0 1 1]);
```

```
        imagesc(frame);
        title('Scalogram');
        colormap parula;
        hcb = colorbar;
        ylabel(hcb, 'Energy/Coefficient [%]');
        set(get(gca,'XLabel'),'String','Time');
        set(gca,'YTick', 1:levels+1);
        fv_label = arrayfun(@num2str, levels+1:-1:1, 'UniformOutput', false);
        yticklabels('manual');
        yticklabels(fv_label);
        set(get(gca,'YLabel'),'String','Scale');
        set(findall(gcf,'type','text'), 'FontSize', 22, 'fontWeight', 'bold');
        set(gca,'FontSize',16);

        % Provide Outputs
        if nargout == 1
                varargout{1} = frame;
        elseif nargout > 1
                error('Error with varargout outputs!');
        end
end
```

Q3 - Rotina computacional para visualização dos gráficos da Questão 3

```
clear all; close all; clc;
rng('default'); % initialize the random number generator to make the results in this
    example repeatable.

% Load Signal
emg = load('emg1.mat', '-mat');

% Calculate FFT
np2 = 2^nextpow2(length(emg.x));
emg.X = abs(fftshift(fft(emg.x, np2)));
figure('units', 'normalized', 'outerposition', [0 0 1 1]);
area(linspace(-0.5*emg.fs, 0.5*emg.fs, np2), 10*log10(emg.X), 'FaceColor','b','
   FaceAlpha',.8,'EdgeAlpha',.02,'EdgeColor','k');
grid on;
grid minor;
set(get(gca,'XLabel'),'String','Frequencies [Hz]');
set(get(gca,'YLabel'),'String','Magnitude [dB]');
title('FFT - EMG Signal');
set(findall(gcf,'type','text'), 'FontSize', 32, 'fontWeight', 'bold');
set(gca,'FontSize',26);

% Add Harmonic Events in EMG Signal
emg.xn = emg.x;
freq = linspace(0.75*emg.fs/2, 0.9*emg.fs/2, 200);
% First Event in 1s
emg.xn = aditiveHarmonicNoise(emg.xn, emg.fs, round(1*emg.fs), 100e-3, freq, 0.7);
% First Event in 2s
emg.xn = aditiveHarmonicNoise(emg.xn, emg.fs, round(2*emg.fs), 100e-3, freq, 0.7);
% First Event in 3s
```

```
emg.xn = aditiveHarmonicNoise(emg.xn, emg.fs, round(3*emg.fs), 100e-3, freq, 0.7);

% Calculate FFT
emg.Xn = abs(fftshift(fft(emg.xn, np2)));
figure('units', 'normalized', 'outerposition', [0 0 1 1]);
area(linspace(-0.5*emg.fs, 0.5*emg.fs, np2), 10*log10(emg.X), 'FaceColor','b','
    FaceAlpha',.2,'EdgeAlpha',.02,'EdgeColor','k');
hold on;
area(linspace(-0.5*emg.fs, 0.5*emg.fs, np2), 10*log10(emg.Xn), 'FaceColor','r','
    FaceAlpha',.7,'EdgeAlpha',.02,'EdgeColor','k');
grid on;
grid minor;
legend('Original Signal', 'Harmonic Events');
set(get(gca,'XLabel'),'String','Frequencies [Hz]');
set(get(gca,'YLabel'),'String','Magnitude [dB]');
title('FFT - EMG Signal with Harmonic Events');
set(findall(gcf,'type','text'), 'FontSize', 32, 'fontWeight', 'bold');
set(gca,'FontSize',26);

% Scalogram Functions to PATH
addpath('../utils_P1/');
addpath('../filterbanks/');
rehash path;
% Set Colormap
gray_ = flipud(gray);
% Plot Scalogram
dwtScalogram(emg.xn, emg.fs, 'db2', 3);
colormap(gray_);
title('Scalogram: db2 - 3 Levels for EMG with Events');
dwtScalogram(emg.x, emg.fs, 'db2', 3);
a = dwtScalogram(emg.x(0.8*emg.fs:1.2*emg.fs), emg.fs, 'db2', 3);
title('Scalogram: db2 - 3 Levels for EMG');
colormap(gray_);
% Scalogram
dwtScalogram(emg.xn, emg.fs, 'dmey', 3);
colormap(gray_);
title('Scalogram: dmey - 3 Levels for EMG with Events');

% Spectrogram
win = hamming(round(150e-3*emg.fs));
stftSpectrogram(emg.xn, emg.fs, win, length(win)/2, 2^nextpow2(length(win)));
title('STFT Spectrogram: Hamming Window for EMG with Events');

dwtScalogram(emg.xn, emg.fs, 'dmey', 4);
colormap(gray_);
title('Scalogram: dmey - 4 Levels for EMG with Events');
```

Q4 - Código para cômputo das métricas e gráficos

```
clear all; close all; clc;

% Add QMF Functions to PATH
addpath('../filterbanks/');
```

```
rehash path;

DATA = load('ecg_1.mat', '-mat');
x = DATA.x(:); fs = DATA.fs;

% Process Data
NM = 5:100;
[xq, len] = dwtEcgQuant(x, 'db4', 4, NM);
[rx] = dwtEcgRec(xq, len, 'db4');

% Distortion Evaluation Metrics
SNR = @(ref_data, n_data) mean( n_data.^ 2 ) / mean( (n_data - ref_data).^2 );
PRD = @(ref_data, n_data) sqrt( (sum((ref_data - n_data).^ 2)) / (sum((ref_data).^2)
    ) )*100;
% Compression Evaluation Metric (CR - Compression Ratio = OriginalSize/
    CompressedSize)
cr = length(x)./round(length(x) * (NM/100));

% Evaluate Distortion
snr = zeros(1, length(NM));
prd = zeros(1, length(NM));
for n = 1:length(rx)
        prd(n) = PRD(x, rx{n});
        snr(n) = SNR(x, rx{n});
end

% Plot Results
figure('units', 'normalized', 'outerposition', [0 0 1 1]);
plot(NM, 10*log10(snr), 'LineWidth', 5);
hold on;
grid on;
grid minor;
title("DWT ECG Compression - db4 with 4 levels: SNR vs NM");
set(get(gca,'YLabel'),'String','SNR [dB]');
set(get(gca,'XLabel'),'String','N_m [%]');
set(findall(gcf,'type','text'), 'FontSize', 32, 'fontWeight', 'bold');
set(gca,'FontSize',32);
% Annotate 30+ dB Point
y = ceil(interp1(10*log10(snr), NM, [30], 'linear'));
db30 = find(NM == y);
stem(y, 10*log10(snr(db30)), 'filled', 'MarkerSize', 12);
legend('SNR vs NM', [num2str(10*log10(snr(db30)),3) ' dB Marker (NM = ' int2str(y) '
    )'], 'Location', 'northwest');

% PRD vs CR
figure('units', 'normalized', 'outerposition', [0 0 1 1]);
plot(cr, prd, 'LineWidth', 5);
legend('PRD vs CR');
grid on;
grid minor;
xlim([0 13]);
title("DWT ECG Compression - db4 with 4 levels: PRD vs CR");
set(get(gca,'YLabel'),'String','PRD [%]');
set(get(gca,'XLabel'),'String','CR');
```

```
set(findall(gcf,'type','text'), 'FontSize', 32, 'fontWeight', 'bold');
set(gca,'FontSize',32);

% Plot Signal in time domain and reconstruction error
figure('units', 'normalized', 'outerposition', [0 0 1 1]);
subplot(2,1,1);
start_time = 2; end_time = 6;
x_seg = x(start_time*fs:end_time*fs);
plot(linspace(start_time, end_time, length(x_seg)), rx{y}(start_time*fs : end_time*
    fs), 'LineWidth', 4);
hold on;
plot(linspace(start_time, end_time, length(x_seg)), x_seg, '--', 'LineWidth', 4);
title('Comparison: 30+ dB SNR Reconstructed Signal vs Original Signal');
l = legend([num2str(10*log10(snr(db30)),3) ' dB Marker (NM = ' int2str(y) ')'], '
    Original Signal', 'Location', 'northwest');
set(findall(gcf,'type','text'), 'FontSize', 28, 'fontWeight', 'bold');
set(gca,'FontSize',28);
l.FontSize = 14;
hold off;
subplot(2,1,2);
area(linspace(start_time, end_time, length(x_seg)), abs(x_seg-rx{y}(start_time*fs :
    end_time*fs)), 'FaceColor', 'b', 'FaceAlpha', .3, 'EdgeAlpha', .3);
set(get(gca,'XLabel'),'String','Time [s]');
title('Absolute Reconstruction Error');
set(findall(gcf,'type','text'), 'FontSize', 28, 'fontWeight', 'bold');
set(gca,'FontSize',28);
```

Q5 - Extrator de características (Temporais & Espectrais)

```
%% featureExtractor: function description
function [temporal_features, frequency_features] = featureExtractor(data, fs, win,
    loverlap, zeropad, iirfilter, bands, nbands, boverlap, maxfreq, fmaxorder, ftol,
    fnum, fden)

        if ~exist('iirfilter', 'var') || isempty(iirfilter)
                iirfilter = @butter;
        end
        if ~exist('fmaxorder', 'var') || isempty(fmaxorder)
                fmaxorder = 15;
        end
        if ~exist('boverlap', 'var') || isempty(boverlap)
                boverlap = 0.25;
        end
        if ~exist('ftol', 'var') || isempty(ftol)
                ftol = 1e-5;
        end
        if ~exist('maxfreq', 'var') || isempty(maxfreq)
                maxfreq = fs/2;
        end
        if ~exist('zeropad', 'var') || isempty(zeropad)
                zeropad = false;
        end
        lwin = length(win);
```

```
23          % Reshape Signal
24          data = reshapeOverlap(data, lwin, loverlap, zeropad);
25          s_data = size(data);
26          % Perform Window Multiplication
27          win_matrix = repelem(win, 1, s_data(2));
28          data = data.*win_matrix;
29
30          % Extract temporal features
31          rMeanVal = @(w) mean(abs(w)); % Rectified Mean Value
32          RMS = splitapply(@rms, data, 1:s_data(2));
33          STD = splitapply(@std, data, 1:s_data(2));
34          RMV = splitapply(rMeanVal, data, 1:s_data(2));
35
36          %% Extract frequency features
37          % Evaluate Filtering bands
38          if ~exist('bands', 'var') || isempty(bands)
39                  maxnfreq = maxfreq / fs;
40                  bands = [ (0 : maxnfreq / nbands : maxnfreq * (1 - 1/nbands)).' (
                        maxnfreq / nbands : maxnfreq / nbands : maxnfreq).'];
41                  bands(2:end,1) = bands(2:end,1) - boverlap / 2 * maxnfreq / nbands;
42                  bands(1:end-1,2) = bands(1:end-1,2) + boverlap / 2 * maxnfreq /
                        nbands;
43          end
44          % Fix bands if nbands = 1 (The cutoff frequencies must be within the
               interval of (0,1))
45          if isequal(nbands,1)
46                  bands(2) = bands(2) - ftol;
47          end
48          % Design Filters
49          if (~exist('fnum', 'var') || isempty(fnum)) && (~exist('fnum', 'var') ||
               isempty(fnum))
50                  fnum = cell(1, nbands);
51                  fden = cell(1, nbands);
52                  for b = 1 : nbands
53                          [fnum{b}, fden{b}] = iir_design(iirfilter, fmaxorder, ftol,
                               bands(b, :));
54                  end
55          end
56          % Perform Filtering
57          nfft = 2^(nextpow2(lwin));
58          freq = linspace(0,fs/2, nfft);
59          getEnergy = @(w) norm(w)^2;
60          magfft = @(w) abs(fft(w-mean(w), nfft)); % Use zero-mean version!!
61          fdata = zeros(s_data(1), s_data(2), nbands);
62          fftdata = zeros(nfft, s_data(2), nbands);
63          E = zeros(s_data(2), nbands);
64          MEDFREQ = zeros(s_data(2), nbands);
65          MEANFREQ = zeros(s_data(2), nbands);
66          MODALFREQ = zeros(s_data(2), nbands);
67          for b = 1:nbands
68                  fdata(:,:,b) = filter(fnum{b}, fden{b}, data, [], 1); % Filtering
                        along the cols
69                  E(:,b) = splitapply(getEnergy, fdata(:,:,b), 1:s_data(2)); % Obtain
                        Energy
```

```
                MEANFREQ(:,b) = meanfreq(fdata(:,:,b), fs); % Obtain Mean Frequency
                MEDFREQ(:,b) = medfreq(fdata(:,:,b), fs); % Obtain Median Frequency
                fftdata(:,:,b) = splitapply(magfft, fdata(:,:,b), 1:s_data(2)); %
                    Obtain Frequency Response
                [~,I] = max(fftdata(:,:,b), [], 1);
                MODALFREQ(:,b) = freq(I);
        end
        table_names = {'RMS', 'RMV', 'STD', 'E', 'MEDFREQ', 'MEANFREQ', 'MODALFREQ'
            };
        temporal_features = table(RMS(:), RMV(:), STD(:), 'VariableNames',
            table_names(1:3));
        frequency_features = table(E, MEDFREQ, MEANFREQ, MODALFREQ, 'VariableNames',
             table_names(4:end));
end

% iir_design: function description
function [num, den] = iir_design(iir_type, maximum_order, filter_tol, bands)
        not_finished = 1;
        order = maximum_order;
        bands_ = bands;
        mode = 'bandpass';
        if bands(1) == 0
                bands_ = bands(2);
                mode = 'low';
        end
        if bands(2) == 0.5
                bands_ = bands(1);
                mode = 'high';
        end
        while not_finished
                [num, den] = iir_type(order, bands_ * 2, mode);
                not_finished = any(abs(roots(den)) > 1 - filter_tol);
                order = order - 1;
        end
end
```

Q5 - Rotina para Características de acordo com Estratégia

```
clear all; close all; clc;

% Add Functions to PATH
addpath("../utils_P1/");
rehash path;

% Load A & D Sets
sets_PATH = '~/Dropbox/processamento_sinais_biologicos_02_2019/
    programas_sinais_exemplos/sinais_eeg_epilepsia_e_grupo_controle/sinais/';
rec = 1:1:100;
A = loadSetRecords('A', rec, sets_PATH);
D = loadSetRecords('D', rec, sets_PATH);

% Feature Extractor Basic Params
fs = 173.61;
```

```
win = hamming(round(5*fs)); % 5 sec. win
loverlap = length(win)/2; % 50% overlap (2.5s)
zeropad = false; % no zero padding while reshaping windowed signal
iirfilter = @butter;
boverlap = 0;
nbands = {};
bands = {};

% Strategy 1 Params
nbands{1} = 5;
bands{1} = [0 4; 4 7; 7 15; 16 31; 32 fs/2]./fs;
% Strategy 2 Params
nbands{2} = 10;
bands{2} = [];
% Strategy 3 Params
nbands{3} = 1;
bands{3} = [];

A_ff = cell(length(rec), 3);
D_ff = cell(length(rec), 3);
A_tf = cell(length(rec), 3); % Temporal Feature - RMS Value
D_tf = cell(length(rec), 3); % Temporal Feature - RMS Value
try
        load('A-D_features.mat');
catch ME
        if ~isequal(ME.identifier, 'MATLAB:load:couldNotReadFile')
                rethrow(ME);
        end
        % Apply featureExtractor to all the records
        for n = 1:length(rec)
                % First Strategy
                [A_tf{n,1}, A_ff{n,1}] = featureExtractor(A(:,n), fs, win, loverlap,
                    zeropad, iirfilter, bands{1}, nbands{1}, boverlap);
                [A_tf{n,1}, D_ff{n,1}] = featureExtractor(D(:,n), fs, win, loverlap,
                    zeropad, iirfilter, bands{1}, nbands{1}, boverlap);
                % Second Strategy
                x = A(:,n);
                fmfunc = @(fm) abs(0.95*bandpower(x, fs, [0 (length(x)-1)*fs/(2*
                   length(x))]) - bandpower(x, fs, [0 fm]));
                fm = fminsearch(fmfunc, 1); % Find fm starting in 1 Hz
                [A_tf{n,2}, A_ff{n,2}] = featureExtractor(A(:,n), fs, win, loverlap,
                    zeropad, iirfilter, bands{2}, nbands{2}, boverlap, fm);
                x = D(:,n);
                fmfunc = @(fm) abs(0.95*bandpower(x, fs, [0 (length(x)-1)*fs/(2*
                   length(x))]) - bandpower(x, fs, [0 fm]));
                fm = fminsearch(fmfunc, 1); % Find fm starting in 1 Hz
                [D_tf{n,2}, D_ff{n,2}] = featureExtractor(D(:,n), fs, win, loverlap,
                    zeropad, iirfilter, bands{2}, nbands{2}, boverlap, fm);
                % Third Strategy
                [A_tf{n,3}, A_ff{n,3}] = featureExtractor(A(:,n), fs, win, loverlap,
                    zeropad, iirfilter, bands{3}, nbands{3}, boverlap);
                [D_tf{n,3}, D_ff{n,3}] = featureExtractor(D(:,n), fs, win, loverlap,
                    zeropad, iirfilter, bands{3}, nbands{3}, boverlap);
        end
```

```
end

A_all_features = cell(1, 3);
D_all_features = cell(1, 3);
for s = [1,2]
        for n = 1:size(A_ff,1)
                A_all_features{s} = [A_all_features{s}; A_ff{n,s}{:,1}];
                D_all_features{s} = [D_all_features{s}; D_ff{n,s}{:,1}];
                A_all_features{3} = [A_all_features{3}; A_tf{n,3}{:,1} A_ff{n
                    ,3}{:,2:end}];
                D_all_features{3} = [D_all_features{3}; D_tf{n,3}{:,1} D_ff{n
                    ,3}{:,2:end}];
        end
end
save('A-D_selectedFeatures.mat', 'A_all_features', 'D_all_features');
```

Q5 - Rotina para classificação em 500 sessões de treinamento

```
clear all; close all; clc;

% Load Features
load('A-D_selectedFeatures.mat', 'A_all_features', 'D_all_features');
sessions = 500;

% Perform SVM Classification
confusion_matrix = zeros(2,2,sessions,3);
for s = 1:sessions
        disp(s);
        for n = 1:3
                [svm, out_train, out_val, c_train, c_val, f_train, f_val] =
                    svm2ClassRandperm(0.8, A_all_features{n}, D_all_features{n});
                confusion_matrix(:,:,n,s) = confusionmat(out_val, c_val);
        end
end

% Plot Confusion Matrix Chart
for n = 1:3
        chart_title =  ['Strategy ' int2str(n) ' - Validation in ' int2str(sessions)
            ' sessions.'];
        confusionChart(mean(confusion_matrix(:,:,n,:),4), chart_title);
end

% ROC Metrics
plotSessionResults(confusion_matrix);

%% plotSessionResults: Obtain ROC Metrics
function plotSessionResults(confusion_matrix)
        % Metrics
        fpRate = @(cm) cm(1,2)/sum(cm(:,2));
        tpRate = @(cm) cm(1,1)/sum(cm(:,1));
        precision = @(cm) cm(1,1)/sum(cm(1,:));
        accuracy = @(cm) sum(diag(cm))/sum(cm(:));
        recall = @(cm) cm(1,1)/sum(cm(:,1));
```

```
        fMeasure = @(cm) 2/( 1/precision(cm) + 1/recall(cm) );

        % Display in console
        disp(['FP Rate is ' num2str(fp_val(confusion_matrix),4)]);
        disp(['TP Rate is ' num2str(tp_val(confusion_matrix),4)]);
        disp(['Precision is ' num2str(precision_val(confusion_matrix),4)]);
        disp(['Accuracy is ' num2str(accuracy_val(confusion_matrix),4)]);
        disp(['Recall is ' num2str(recall_val(confusion_matrix),4)]);
        disp(['F-Measure is ' num2str(fMeasure_val(confusion_matrix),4)]);
        disp(' ');
end

%% confusionChart: Plot Confusion Matrix Chart
function [varargout] = confusionChart(C, chart_title)
        C = round(100*C/(sum(C(:))/2),2);
        figure('units','normalized','outerposition',[0 0 1 1]);
        h = heatmap(C);
        h.ColorbarVisible = 'off';
        colormap white;
        title(chart_title);
        h.YLabel = 'True Class';
        h.XLabel = 'Predicted Class';
        set(findall(gcf,'type','text'), 'FontSize', 32, 'fontWeight', 'bold');
        set(gca,'FontSize',26);
        if nargout == 1
                varargout{1} = h;
        end
end
```