University of Brasília

Electrical Engineering Department

Topics in Biomedical Engineering

Task 6

Authors:

Caio Luiz Candeias Flôres 190134283

Felipe Carneiro da Motta 200017616

João Pedro Daher Aranha 190109742

Brasília

September 16, 2022

Contents

1 Subject f2o02 - Elderly Subject

The elderly subject (**f2o02**) is the same analyzed in the Task 5. The processes of filtering (Notch Filter - 60 Hz in the ECG signal of this subject and Low-Pass Filter in the RRI signal), R peaks detection, SBP/DBP peaks detection, interpolation, resampling (Berger Algorithm) and spectral windowing (Hanning Window) were done according to the previous task.

The Low-Pass Filter applying in the RRI signal is important because it works as an anti-aliasing filter and must be applied before calculate the spectral analysis.

1.1 Signal Preprocessing

The steps of preprocessing were the same adopted in the Task 5. The time interval was restricted in five minutes (300 seconds), and, in this subject, the interval between 2100 and 2400 seconds (minute 35 and minute 40) has little noise and generates a good observation around electrocardiogram and blood pressure signals. Thus, the **interval 2100 to 2400** seconds was the observed interval.

1.2 Transfer Function (H(jw) - SBP → RRI)

In sequence, with the spectral analysis (PSD - Power Spectral Density) done on the CRSIDLab in the previous task, was determined the transfer function of this system. The system was defined considering SBP (Systolic Blood Pressure) as the input and the RRI (RR Interval) as the output.

The MATLAB's code to define the transfer function of the subject **f2o02** was:

```
%% Subject f2o02 - Elderly
clc; clear all; close all;
warning('off');

% obs.: only 60 Hz filter
%% TASK 6

load t5_f2o02.mat; % load of patient file - CRSIDLab
open('patient');

```

```
tempo = patient.sig.ecg.rri.aligned.time;
rri_detrend = detrend(patient.sig.ecg.rri.aligned.data); % retira trend linear do rri
sbp_detrend = detrend(patient.sig.bp.sbp.aligned.data); % retira trend linear do sbp

% Janelamento (detrend) antes de se calcular a FFT:
N = length(rri_detrend);
u = sbp_detrend.*hanning(length(sbp_detrend)); % esta será a entrada X(s)
y = rri_detrend.*hanning(length(rri_detrend)); % esta será a saída Y(s)
fs = 4; % em Hz; frequencia de reamostragem

T = N/fs; % Tempo de observação, = N*dt, onde dt = 1/fs = 1/4 s
U = fft(u);
U = U(2:((N/2)+1)); % Usar apenas metade do vetor. U(1) representa o valor médio
Suu = 1/N*real(conj(U).*U); % PSD é real por definição(a parte imaginária deve ser muito pequena)

Y = fft(y); % FFT = Fast Fourier Transform
Y = Y(2:((N/2)+1));

Syy=1/N*real(conj(Y).*Y); % Densidade espectral de potencia (PSD) de y
Suy=1/N*conj(U).*Y; % Densidade espectral de potência cruzada (CPSD) de u e y
f=(1:(N/2)).'/T; % Vetor de frequências
Hsbp=Suy./Suu; % Estimativa para função de resposta em frequência (FRF)
Csbp=abs(Suy).^2./(Suu.*Syy); % Estimativa para a coerência

% Plotando a função resposta em frequência (FRF) e a coerência
% (seguindo o modelo implementado em "Lec3_SmoothSuu.m":
figure(1)
subplot(311)
```

```
loglog(f,abs(Hsbp), 'b', 'linewidth', 1.3); grid;
xlabel('Frequency');
ylabel('|H(w)|');
title('Module of Frequency Response');

subplot(312)
semilogx(f,angle(Hsbp)*180/pi, 'b', 'linewidth', 1.3);
    grid;
xlabel('Frequency');
ylabel('<H(w)');
title('Phase of Frequency Response');

subplot(313)
semilogx(f,Csbp, 'b', 'linewidth', 1.3); grid;
xlabel('Frequency');
ylabel('C(w)');
title('Coherence'); hold on;
saveas(gcf, 't6_old_fig1.png');
% Veja que o fato da coerência dar sempre 1 é um "
    problema" da FFT sem média alguma.
% Fazendo pelo método de Welch:

%% * MÉTODO DE WELCH *
% Resumo: Divide o sinal em vários trechos. Esses
    trechos são organizados de forma a
% ficarem com determinada percentagem de seu
    comprimento sobreposta ao trecho
% anterior. Assim, o método calcula o espectro
    aplicando a transformada de Fourier
% nesses trechos menores do sinal.
% Aqui, utiliza-se 50% de sobreposição e 8 trechos do
    sinal.

[SuuW,fw1] = cpsd(u,u,[],[],[],4);
[SyyW,fw2] = cpsd(y,y,[],[],[],4);
[SuyW,fw3] = cpsd(u,y,[],[],[],4);

%%% Função de transferência
HWsbp=SuyW./SuuW;
CWsbp=abs(SuyW).^2./(SuuW.*SyyW);
```

```
% Plotando o resultado (como em "Lec3_SmoothSuu.m"):
figure(2)
subplot(311)
loglog(fw1,abs(HWsbp), 'r', 'linewidth', 1.3); grid;
xlabel('Frequency');
ylabel('|H(w)|');
title('Module of the Frequency Response - Welch Method'
    );

subplot(312)
semilogx(fw1,angle(HWsbp)*180/pi, 'r', 'linewidth',
    1.3); grid;
xlabel('Frequency');
ylabel('<H(w)');
title('Phase of the Frequency Responde - Welch Method')
    ;

subplot(313)
semilogx(fw1,CWsbp,'r', 'linewidth', 1.3); grid %
    COMPARE COM AS FIGURAS DO FFT
xlabel('Frequency');
ylabel('C(w)');
title('Coherence - Welch Method');
saveas(gcf, 't6_old_fig2.png');
%% Cálculo das áreas de BF (baixa frequencia) e AF (
    alta frequencia) da FT:
% (continuação)

%close all; clear all; clc;

HWsbp_lf = zeros(size(HWsbp));
HWsbp_hf = zeros(size(HWsbp));
for i = 1:length(HWsbp)
    if (fw1(i)>= 0.04) && (fw1(i) <= 0.15)
    HWsbp_lf(i) = abs(HWsbp(i));
    else
    HWsbp_lf(i) = 0;
    end

    if (fw1(i)> 0.15) && (fw1(i) <= 0.4)
```

```
            HWsbp_hf(i) = abs(HWsbp(i));
      else
            HWsbp_hf(i) = 0;
      end
end

area_H_LF = trapz(HWsbp_lf);
area_H_HF = trapz(HWsbp_hf);

%% Cálculo das áreas de BF (baixa frequencia) e AF (alta frequencia) da FT:
% Considerando apenas os pontos com coerência acima de 0,5: parte onde a entrada
% e a saida possuem uma relacao linear independentemente do tipo de sys
% (onde faz sentido calcular a função de transferência (tf))

HWsbp_lf_c = zeros(size(HWsbp));
HWsbp_hf_c = zeros(size(HWsbp));
for i = 1:length(HWsbp)
      if (fw1(i)>= 0.04) && (fw1(i) <= 0.15) && (CWsbp(i) > 0.5)
            HWsbp_lf(i) = abs(HWsbp(i));
      else
            HWsbp_lf(i) = 0;
      end
      if (fw1(i)> 0.15) && (fw1(i) <= 0.4) && (CWsbp(i) > 0.5)
            HWsbp_hf(i) = abs(HWsbp(i));
      else
            HWsbp_hf(i) = 0;
      end
end

area_H_LF_c = trapz(HWsbp_lf);
area_H_HF_c = trapz(HWsbp_hf);

sprintf('Elderly Subject - f2o02: \n HLF_c = %d \n HLF_c = %d', area_H_LF_c, area_H_HF_c) % print areas
```

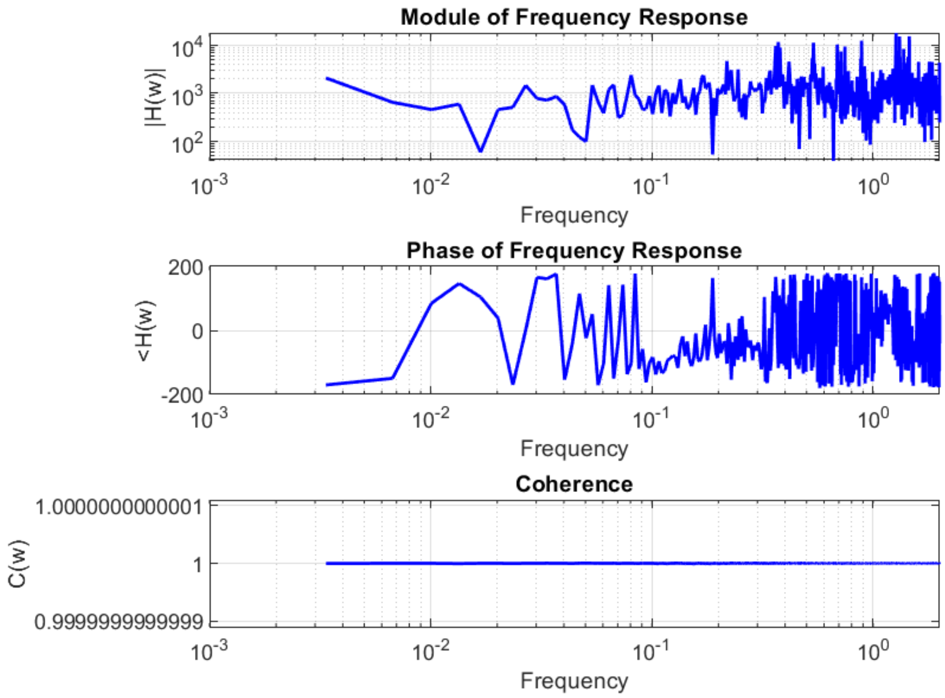

Figure 1

It is possible to visualize in the Figure 1 that the non-application of a spectral average model, the Welch method in this case, generates a unitary value of coherence.

In the Figure 2, the Welch method was applied in the input and output before calculating the transfer funcion. Visually, it provided significant values of coherence to be analyzed.

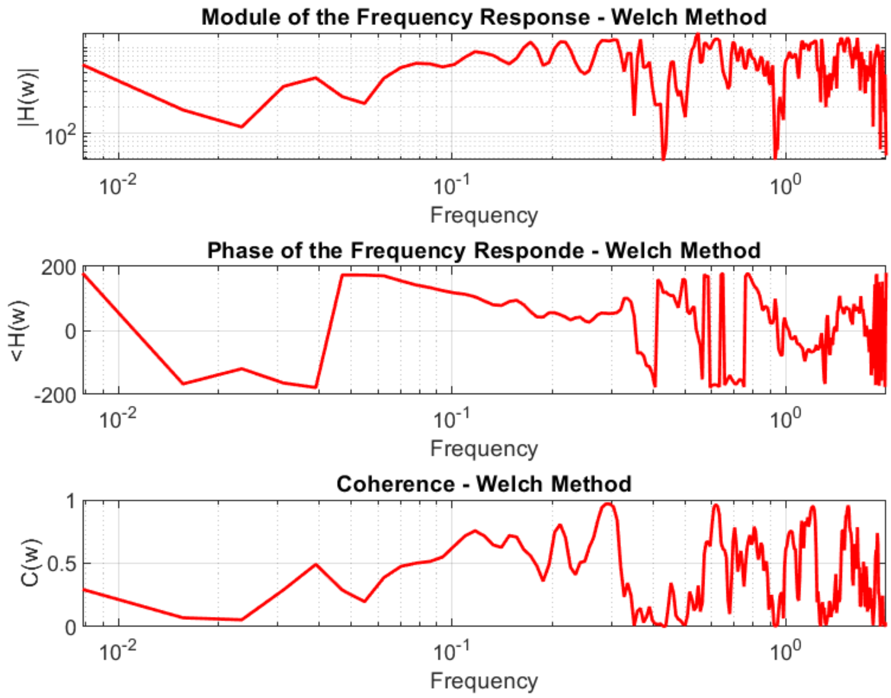

Figure 2

This code also calculates the absolute area in the low frequencies (0.04 to 0.15 Hz) and in the high frequencies (0.15 to 0.4 Hz). These calculated absolute areas, considering coherence bigger than 0.5, were:

```
'Elderly Subject - f2o02:
  HLF_c = 7.200607e+03
  HLF_c = 1.562884e+04'
```

Figure 3

The choice to calculate the absolute area of the signal spectrum only in snippets with coherence bigger than 0.5 was done in order to get more reasonable results. The coherence measure how linear the relationship between input and output is, and the transfer function is a relation between the ratio of the output and the input of a linear system. Therefore, this value of coherence (bigger than 0.5) is an acceptable marker in the biomedical signal analysis literature. It is stated on Saul, J. P., Berger et al. Transfer function analysis of the circulation: unique insights into cardiovascular regulation. American Journal of Physiology-Heart and Circulatory Physiology. 1991: "The coherence spectra are near or above 0.5 throughout at most indicating that the magnitude and phase estimates are relatively reliable at those frequencies."

2 Subject f2y02 - Young Subject

The young subject (**f2o02**) is also the same analyzed in the Task 5. The processes of filtering (No filters in the ECG signal of this subject and Low-Pass Filter in the RRI signal), R peaks detection, SBP/DBP peaks detection, interpolation, resampling (Berger Algorithm) and spectral windowing (Hanning Window) were done according to the previous task.

The Low-Pass Filter applying in the RRI signal is important because it works as an anti-aliasing filter and must be applied before calculate the spectral analysis.

2.1 Signal Preprocessing

In the same way of the elderly subject, the steps of preprocessing were the same adopted in the Task 5. The time interval was restricted in five minutes (300 seconds), and, in this subject, the interval between 2100 and 2400 seconds (minute 35 and minute 40) has also little noise and provides a good observation around electrocardiogram and blood pressure signals. Thus, the **interval 2100 to 2400** seconds was the observed interval.

2.2 Transfer Function (H(jw) - SBP $\rightarrow$ RRI)

In this subject, was also determined the transfer function of this system. The system was also defined considering SBP (Systolic Blood Pressure) as the input and the RRI (RR Interval) as the output.

The MATLAB's code to define the transfer function of the subject **f2y02** was:

```matlab
%% Subject f2y02 - Young
clc; clear all; close all;
warning('off');

% obs.: not necessary to apply filters
%% TASK 6

load t5_f2y02.mat; % load of patient file - CRSIDLab
open('patient');

tempo = patient.sig.ecg.rri.aligned.time;
rri_detrend = detrend(patient.sig.ecg.rri.aligned.data); % retira trend linear do rri
sbp_detrend = detrend(patient.sig.bp.sbp.aligned.data); % retira trend linear do sbp

% Janelamento (detrend) antes de se calcular a FFT:
N = length(rri_detrend);
u = sbp_detrend.*hanning(length(sbp_detrend)); % esta será a entrada X(s)
y = rri_detrend.*hanning(length(rri_detrend)); % esta será a saída Y(s)
fs = 4; % em Hz; frequencia de reamostragem

T = N/fs; % Tempo de observação, = N*dt, onde dt = 1/fs = 1/4 s
U = fft(u);
U = U(2:((N/2)+1)); % Usar apenas metade do vetor. U(1) representa o valor médio
Suu = 1/N*real(conj(U).*U); % PSD é real por definição(a parte imaginária deve ser muito pequena)

Y = fft(y); % FFT = Fast Fourier Transform
Y = Y(2:((N/2)+1));

Syy=1/N*real(conj(Y).*Y); % Densidade espectral de potencia (PSD) de y
Suy=1/N*conj(U).*Y; % Densidade espectral de potência cruzada (CPSD) de u e y
f=(1:(N/2)).'/T; % Vetor de frequências
```

```
Hsbp=Suy./Suu; % Estimativa para função de resposta em frequência (FRF)
Csbp=abs(Suy).^2./(Suu.*Syy); % Estimativa para a coerência

% Plotando a função resposta em frequência (FRF) e a coerência
% (seguindo o modelo implementado em "Lec3_SmoothSuu.m":
figure(1)
subplot(311)
loglog(f,abs(Hsbp), 'b', 'linewidth', 1.3); grid;
xlabel('Frequency');
ylabel('|H(w)|');
title('Module of Frequency Response');

subplot(312)
semilogx(f,angle(Hsbp)*180/pi, 'b', 'linewidth', 1.3); grid;
xlabel('Frequency');
ylabel('∢H(w)');
title('Phase of Frequency Response');

subplot(313)
semilogx(f,Csbp, 'b', 'linewidth', 1.3); grid;
xlabel('Frequency');
ylabel('C(w)');
title('Coherence'); hold on
saveas(gcf, 't6_young_fig1.png');
% Veja que o fato da coerência dar sempre 1 é um "problema" do método de coerência sem média alguma.
% Fazendo pelo método de Welch:

%% * MÉTODO DE WELCH *
% Resumo: Divide o sinal em vários trechos. Esses trechos são organizados de forma a
% ficarem com determinada percentagem de seu comprimento sobreposta ao trecho
% anterior. Assim, o método calcula o espectro aplicando a transformada de Fourier
```

```
% nesses trechos menores do sinal.
% Aqui, utiliza-se 50% de sobreposição e 8 trechos do sinal.

[SuuW,fw1] = cpsd(u,u,[],[],[],4);
[SyyW,fw2] = cpsd(y,y,[],[],[],4);
[SuyW,fw3] = cpsd(u,y,[],[],[],4);

%% Função de transferência
HWsbp=SuyW./SuuW;
CWsbp=abs(SuyW).^2./(SuuW.*SyyW);

% Plotando o resultado (como em "Lec3_SmoothSuu.m"):
figure(2)
subplot(311)
loglog(fw1,abs(HWsbp), 'r', 'linewidth', 1.3); grid;
xlabel('Frequency');
ylabel('|H(w)|');
title('Module of the Frequency Response - Welch Method');

subplot(312)
semilogx(fw1,angle(HWsbp)*180/pi, 'r', 'linewidth', 1.3); grid;
xlabel('Frequency');
ylabel('<H(w)');
title('Phase of the Frequency Responde - Welch Method');

subplot(313)
semilogx(fw1,CWsbp,'r', 'linewidth', 1.3); grid % COMPARE COM AS FIGURAS DO FFT
xlabel('Frequency');
ylabel('C(w)');
title('Coherence - Welch Method');
saveas(gcf, 't6_young_fig2.png');

%% Cálculo das áreas de BF (baixa frequencia) e AF (alta frequencia) da FT:
% (continuação)

```

```
%close all; clear all; clc;

HWsbp_lf = zeros(size(HWsbp));
HWsbp_hf = zeros(size(HWsbp));
for i = 1:length(HWsbp)
    if (fw1(i)>= 0.04) && (fw1(i) <= 0.15)
    HWsbp_lf(i) = abs(HWsbp(i));
    else
    HWsbp_lf(i) = 0;
    end

    if (fw1(i)> 0.15) && (fw1(i) <= 0.4)
        HWsbp_hf(i) = abs(HWsbp(i));
    else
        HWsbp_hf(i) = 0;
    end
end

area_H_LF = trapz(HWsbp_lf);
area_H_HF = trapz(HWsbp_hf);

%% Cálculo das áreas de BF (baixa frequencia) e AF (alta frequencia) da FT:
% (continuação)
% Considerando apenas os pontos com coerência acima de 0,5: parte onde a entrada
% e a saida possuem uma relacao linear independentemente do tipo de sys
% (onde faz sentido calcular a função de transferência (tf))

HWsbp_lf_c = zeros(size(HWsbp));
HWsbp_hf_c = zeros(size(HWsbp));
for i = 1:length(HWsbp)
    if (fw1(i)>= 0.04) && (fw1(i) <= 0.15) && (CWsbp(i) > 0.5)
        HWsbp_lf(i) = abs(HWsbp(i));
    else
        HWsbp_lf(i) = 0;
    end
    if (fw1(i)> 0.15) && (fw1(i) <= 0.4) && (CWsbp(i) >
```

```
            0.5)
            HWsbp_hf(i) = abs(HWsbp(i));
      else
            HWsbp_hf(i) = 0;
      end
end

area_H_LF_c = trapz(HWsbp_lf);
area_H_HF_c = trapz(HWsbp_hf);

sprintf('Young Subject - f2y02: \n HLF_c = %d \n HLF_c
    = %d', area_H_LF_c, area_H_HF_c) % print areas
```

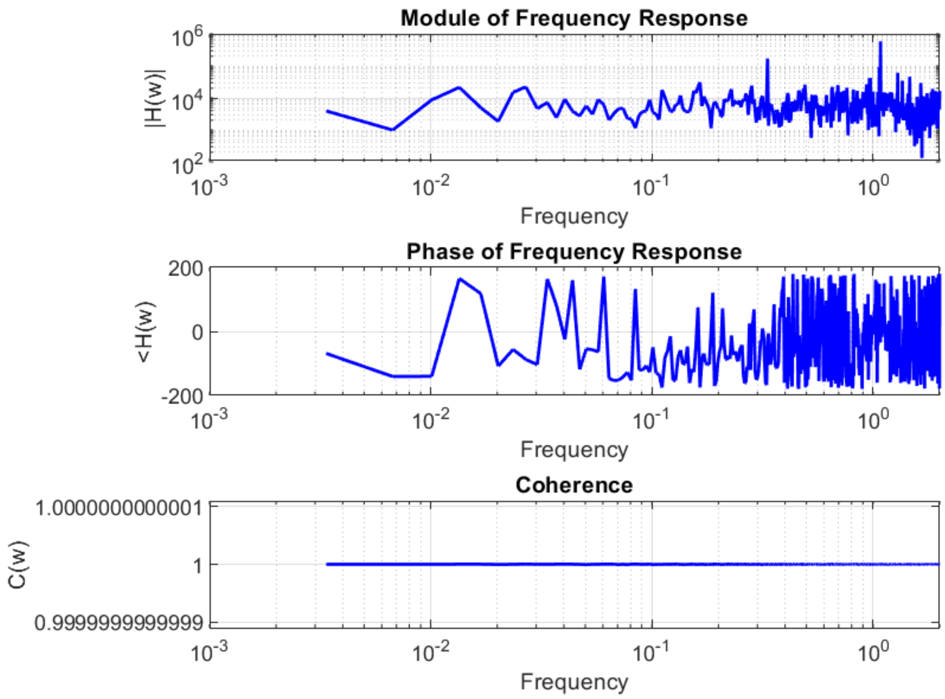

Figure 4

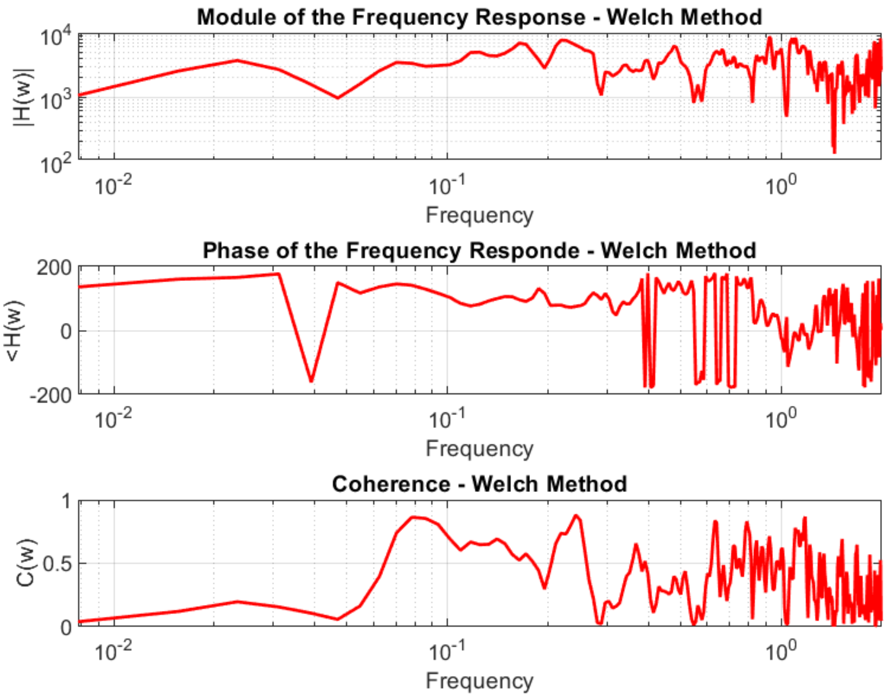

Figure 5

This code also calculates the absolute area in the low frequencies (0.04 to 0.15 Hz) and in the high frequencies (0.15 to 0.4 Hz). These calculated absolute areas, considering coherence bigger than 0.5, were:

```
'Young Subject - f2y02:
  HLF_c = 4.329858e+04
  HLF_c = 8.190362e+04'
```

Figure 6

References

[1] "PhysioBank ATM." https://archive.physionet.org/cgi-bin/atm/ATM?database=fantasia&tool=plot_waveforms.