ENE0332 - TÓPICOS EM ENGENHARIA · 2021-2

Integrantes

- Vinícius Almeida Barros - 18/0028758

1 Questão

Find an approximation to the function shown in Figure P1.4.7a using the first three **Walsh functions** shown in Figure P1.4.7b. Show also that the Walsh functions are orthonormal.

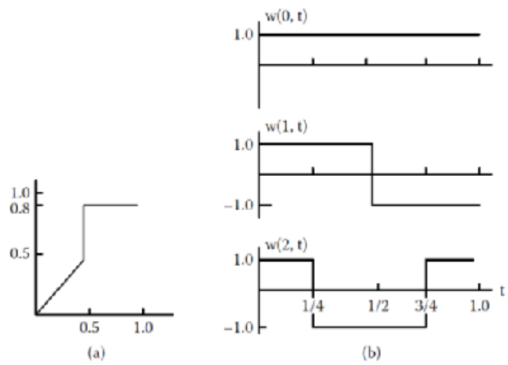

Figure P1.4.7

Figura 1:

Resolução da questão

Na resolução dessa questão, são gerados três vetores correspondentes às três primeiras funções de Walsh. Após isso é gerado um vetor tempo com taxa de amostragem igual a 1 kHz e também a função a ser aproximada. Então, são obtidos os coeficientes calculando o produto interno entre a função e as funções de base. Por último são somados os produtos dos coeficientes e respectivas funções de base para obter a aproximação da função.

A código da resolução da questão pode ser visto abaixo.

```
clear variables; close all; clc;
Fs = 1000; % sample frequency
Ts = 1/Fs; % sample time
Tt = 1; % total time
N = Fs*Tt; % number of samples
o = ones(1,N); % vector of ones
w0  = o; % Walsh function 0
teste = w0;
w1 = [1*o(1:N/2),-1*o(1:N/2)]; % Walsh function 1
w2 = [1*o(1:N/4),-1*o(1:N/4),-1*o(1:N/4),1*o(1:N/4)]; % Walsh function 2
x = (0:N-1)*Ts; % time frequency
fx = [x(1:N/2),0.8*o(1:N/2)]; % original function
ck1 = sum(fx.*w0)/(sum(w0.^2)); %coefficient 1
ck2 = sum(fx.*w1)/(sum(w1.^2)); %coefficient 2
ck3 = sum(fx.*w2)/(sum(w2.^2)); %coefficient 3
hold on
fx_w = ck1.*w0 + ck2.*w1 + ck3.*w2; % function approximation
plot(x,fx,'DisplayName','f(x) original');
plot(x,fx_w,'DisplayName','f(x) approximation');
xlabel('Time (s)');
ylabel('Amplitude');
title('Function approximation using 3 Walsh functions');
grid;
```

```
legend('Location','southeast');
sp1 = sum(w0.*w1)/N  %correlation between first and second Walsh functions
sp2 = sum(w1.*w2)/N %correlation between second and third Walsh functions
sp3 = sum(w2.*w0)/N  %correlation between first and third Walsh functions
len1 = sum(w0.*w0)/N  % Length of first Walsh function
len2 = sum(w1.*w1)/N  % Length of second Walsh function
len3 = sum(w2.*w2)/N  % Length of third Walsh function
```

A saída do código pode ser visto abaixo. Como pode ser visto, a correlação entre bases diferentes é igual a zero, e seus respectivos tamanhos é igual a 1, mostrando que as funções de Walsh são ortonormais.

```
Output:
corr1 = 0
corr2 = 0
corr3 = 0
len1 = 1
len2 = 1
len3 = 1
```

O gráfico gerado pelo código pode ser visto abaixo.

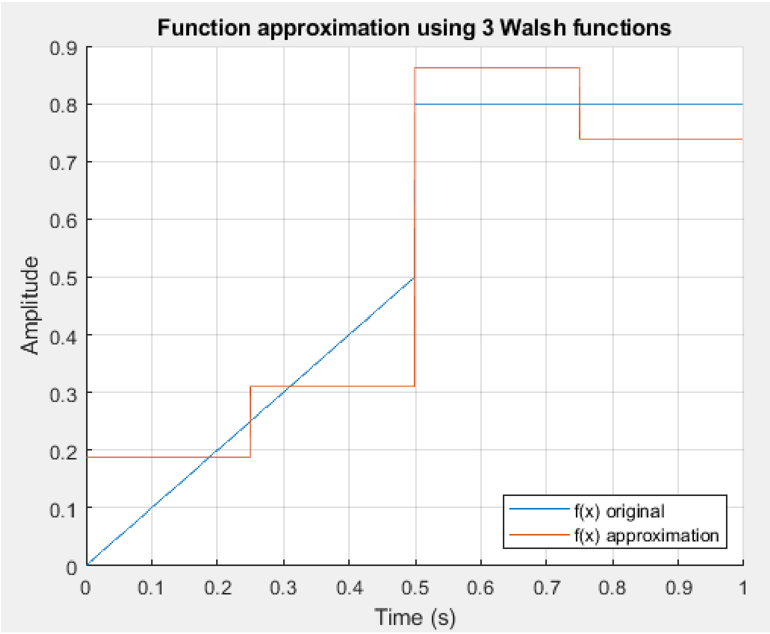

Figura 2:

A mesma questão foi feita agora utilizando um código que gera as funções de Walsh mais facilmente. Foi portanto, facilmente calculadas as aproximações da função utilizando 3, 16 e 32 funções de Walsh. O código para gerar as funções, retirado do site Mathworks e adaptado em uma função pode ser visto abaixo.

```
% This code is a   version of the one provided by Mathworks
% available on https://www.mathworks.com/help/signal/ug/discrete-walsh-hadamard-
    transform.html

function walshMatrix = walshmatrix(N) % N is the length of Walsh (Hadamard) functions
     and must be a power of 2
  hadamardMatrix = hadamard(N);
  HadIdx = 0:N-1;                          % Hadamard index
  M = log2(N)+1;
  binHadIdx = fliplr(dec2bin(HadIdx,M))-'0'; % Bit reversing of the binary index
  binSeqIdx = zeros(N,M-1);                  % Pre-allocate memory
  for k = M:-1:2
    % Binary sequency index
    binSeqIdx(:,k) = xor(binHadIdx(:,k),binHadIdx(:,k-1));
  end
  SeqIdx = binSeqIdx*pow2((M-1:-1:0)');     % Binary to integer sequency index
  walshMatrix = hadamardMatrix(SeqIdx+1,:); % 1-based indexing
```

O código, utilizando a função acima, além do gráfico gerado podem ser vistos abaixo.

```
clear variables; close all; clc;

Fs = 1024; % sample frequency
Ts = 1/Fs; % sample time
Tt = 1; % total time
N = Fs*Tt; % number of samples
w_nums = [3,16,32]; % Number os Walsh functions
num_iter = length(w_nums); % Number of iterations

for idx = (1:num_iter)
  w_num = w_nums(idx); % Number Walsh functions in the current iteration
  subplot(num_iter,1,idx);
  w_power2  = 2^nextpow2(w_num); % Find the next power of 2
  repeat = Fs/w_power2; % Calculate how many times each element of the Walsh matrix
    will be repeated
  aux_matrix = walshmatrix(w_power2);  % auxiliary matrix
  ws = repelem(aux_matrix(1:w_num,:),1,repeat); % Walsh matrix
  x = (0:N-1)*Ts; % time frequency
  fx = [x(1:N/2),0.8*ones(1,N/2)]; % original function
  cks = sum((fx.*ws)')./sum((ws.^2)'); % coefficients
  fx_w =  sum(cks' .* ws,1); % function approximation
  hold on
  plot(x,fx,'DisplayName','f(x) original');
  plot(x,fx_w,'DisplayName','f(x) approximation');
  xlabel('Time (s)');
  ylabel('Amplitude');
  title(sprintf('%d Walsh functions',w_num));
  grid;
  legend('Location','southeast');
  ylim([0,1]);
end
sgtitle('Function approximation using Walsh functions');
```

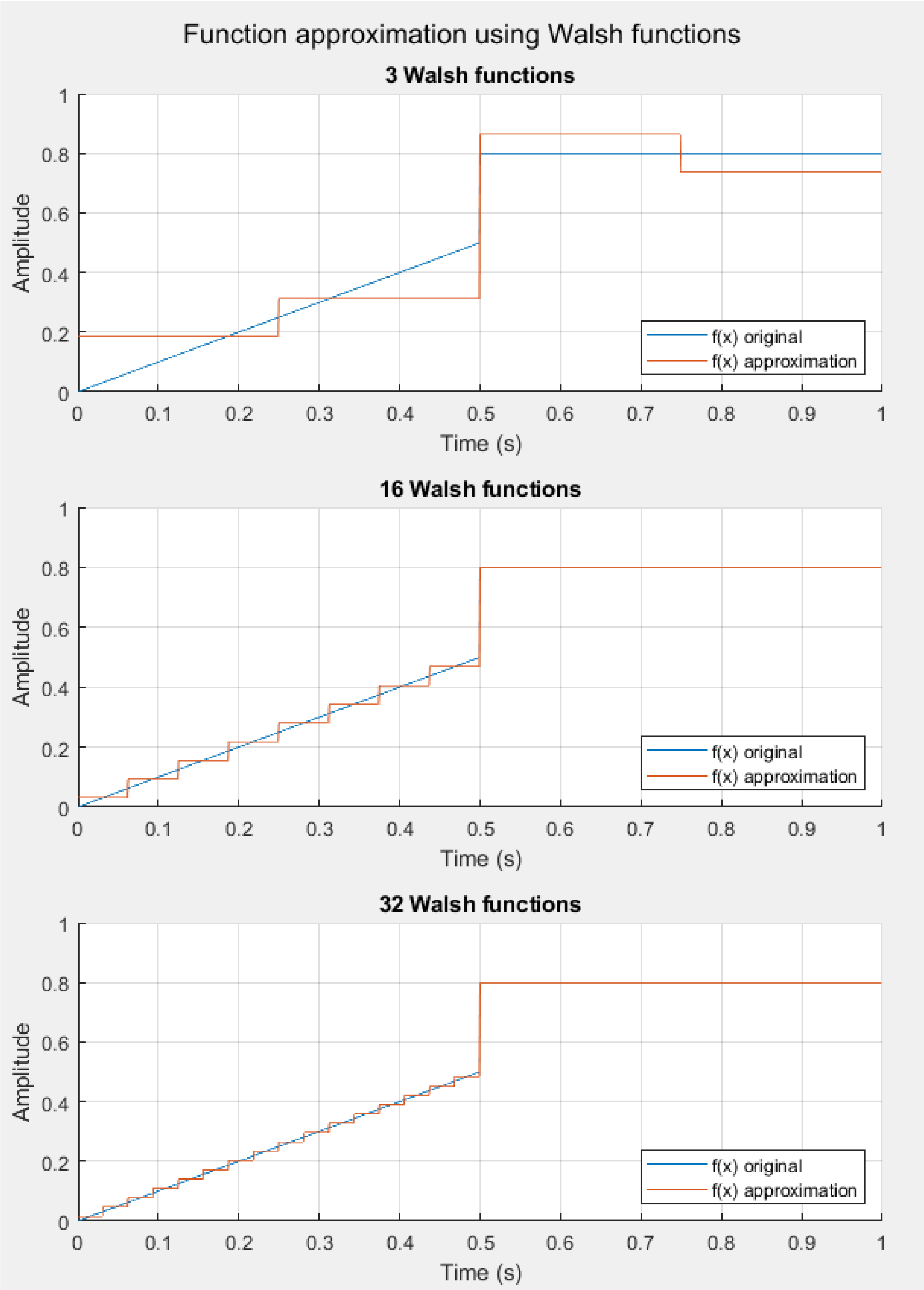
Function approximation using Walsh functions
3 Walsh functions
16 Walsh functions
32 Walsh functions
Amplitude
Time (s)
f(x) original
f(x) approximation
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Figura 3: