MATLAB CODE - FRANCISCO CASTILLO

Table of Contents

Preliminary Commands

```
clear all
close all
clc
linewidth=1.6;
labelfontsize=18;
legendfontsize=12;
```

Introduction

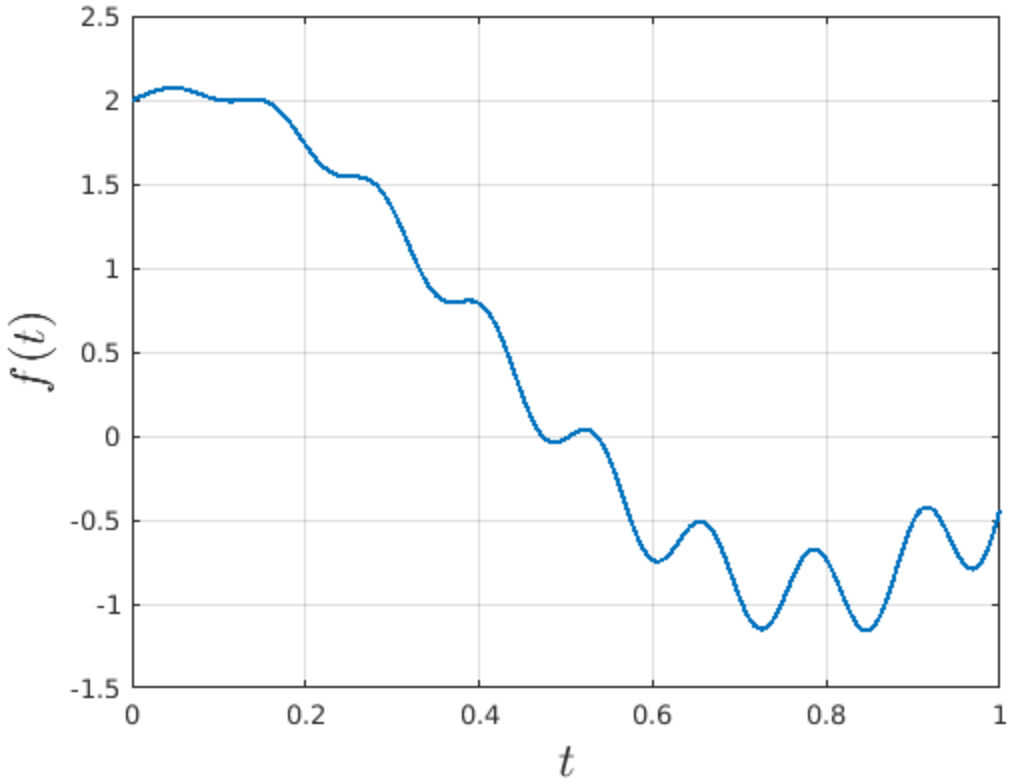
This code solves both problems 9 and 10 of the present homework assignment.

Define the function

```
t=linspace(0,1,256); % interval [0,1] partitioned into 2^8=256 pieces
y = exp(-t.^2/10).*(sin(2*t) + 2*cos(4*t) + .4*sin(t).*sin(50*t));
```

In the following figure we can see the function that we are going to work with in this two problems.

```
figure
plot(t,y,'linewidth',linewidth);
xlabel('$t$','interpreter','latex','fontsize',labelfontsize)
ylabel('$f(t)$','interpreter','latex','fontsize',labelfontsize)
grid on
axis([0 1 -1.5 2.5])
```

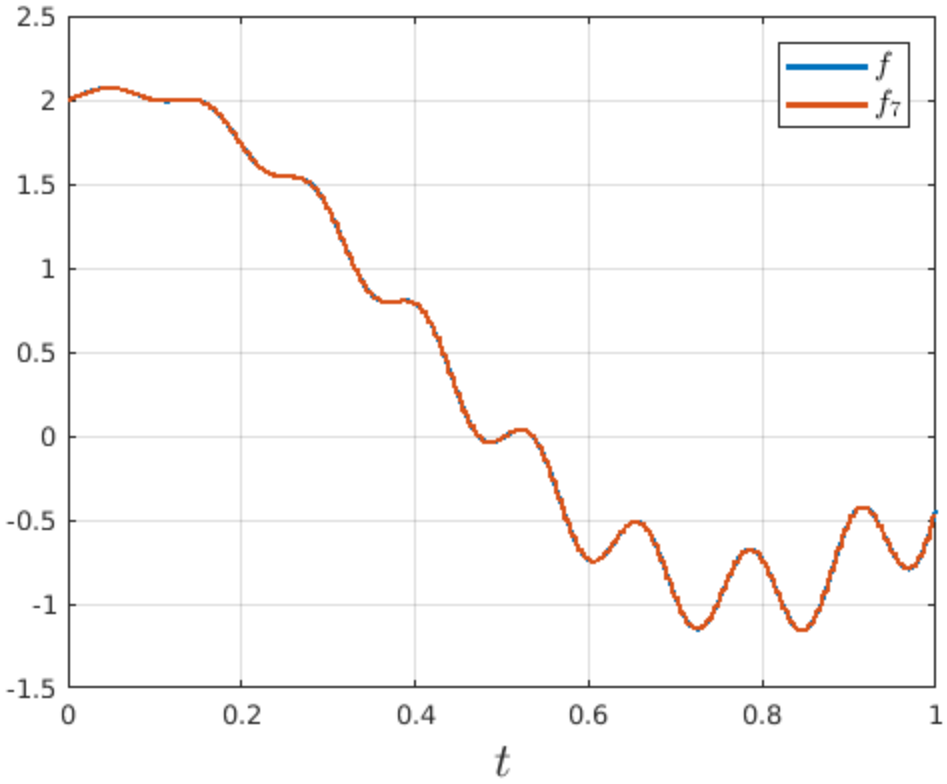


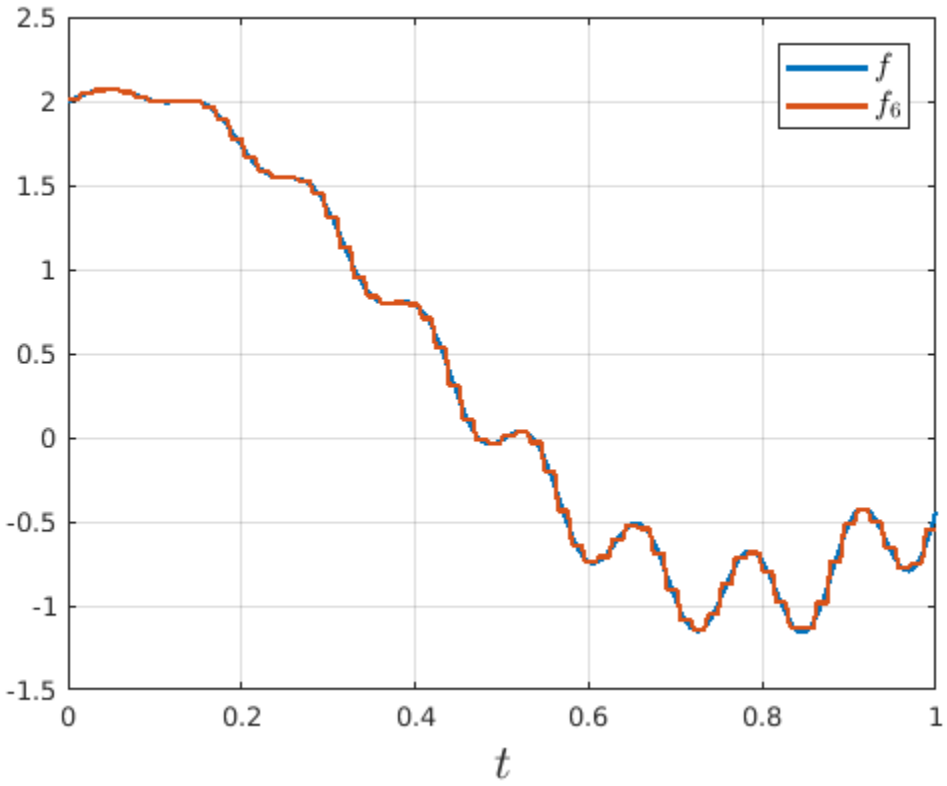
The goal in problem 9 will be decompose the signal using Haar wavelets and plot the resulting levels. In problem 10 the goal will be filter the high frequency noise and obtain a filtered signal, which we will compare to the initial.

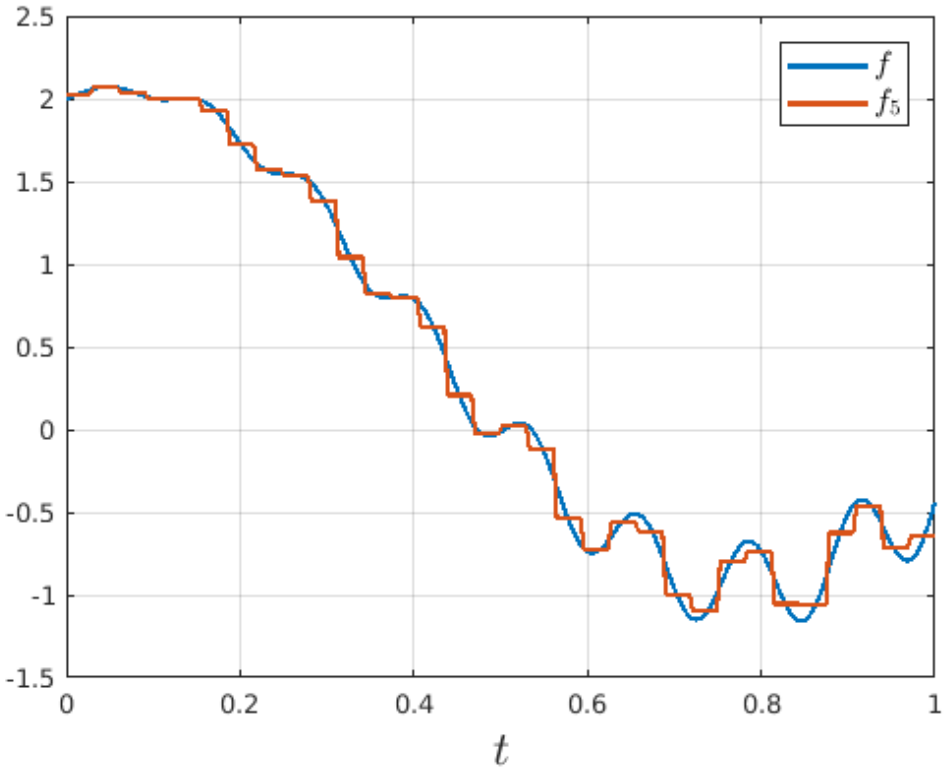
Decomposition (problem 9)

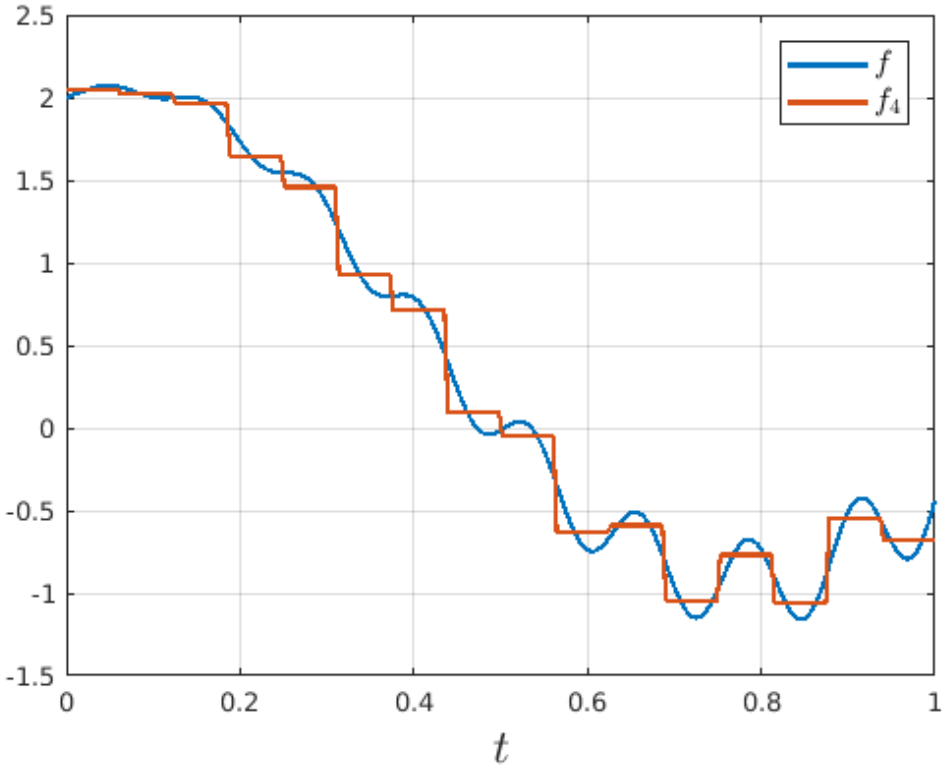
In this section we implement the decomposition algorithm described in Step 2 of Section 4.4.

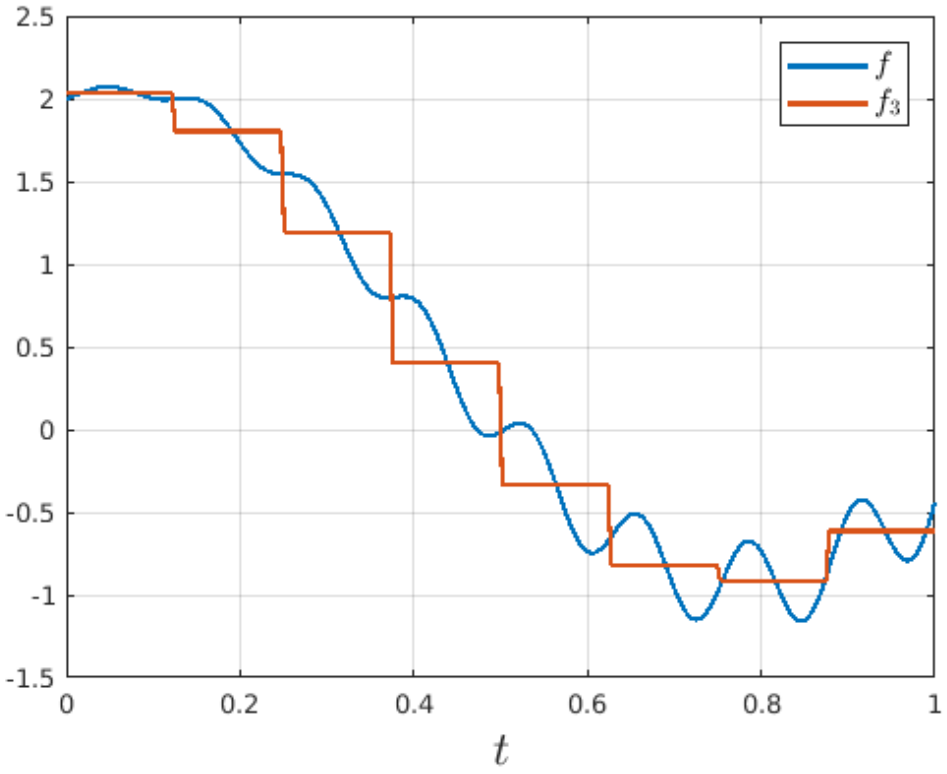
```
for i=1:8
    [C,L] = wavedec(y,i,'db1'); % i-th level decomposition
    Ai = upcoef('a',C(1:L(1)),'db1',i,length(y));
    figure
    plot(t,y,t,Ai,'linewidth',linewidth); % Plot original function and
 the i-th level decomposition
    grid on
    xlabel('$t$','interpreter','latex','fontsize',labelfontsize)
    axis([0 1 -1.5 2.5])
    h=legend('$f$',['$f_' num2str(8-i) '$']);
    set(h,'interpreter','latex','fontsize',legendfontsize);
    saveas(gcf,['f_' num2str(8-i) '_p9'],'png')
end
```

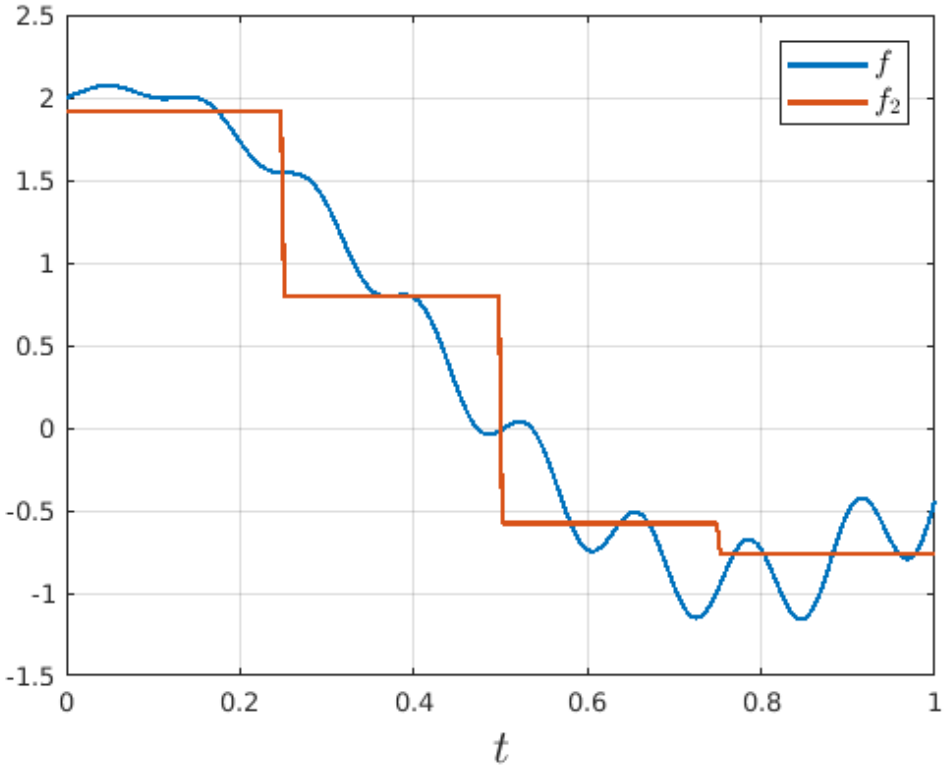

2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_7

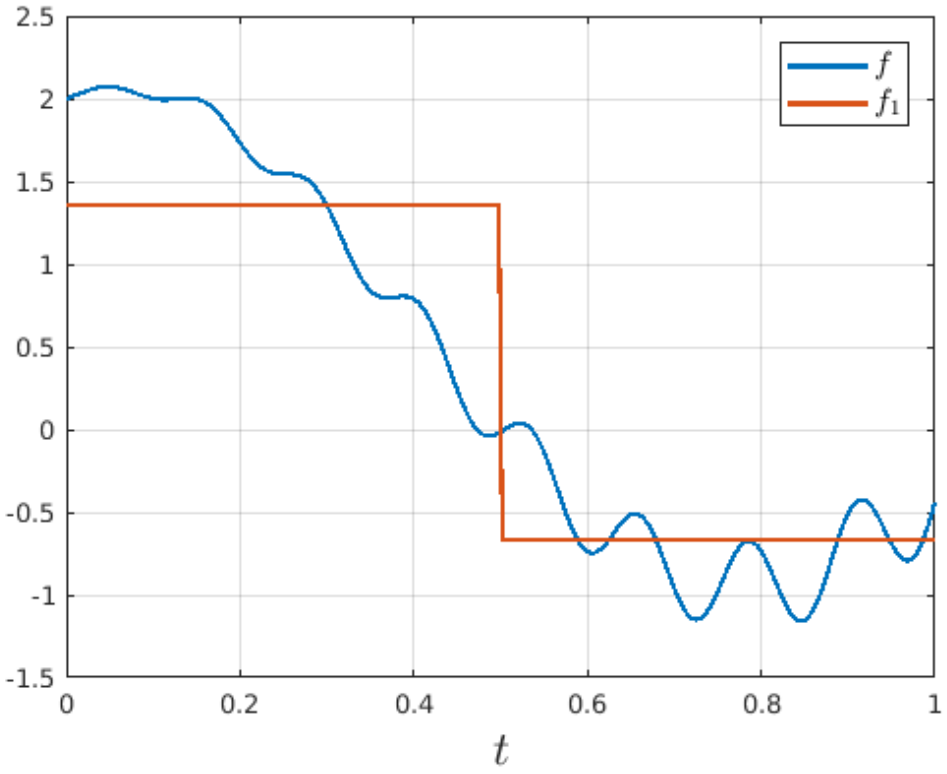

2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_6

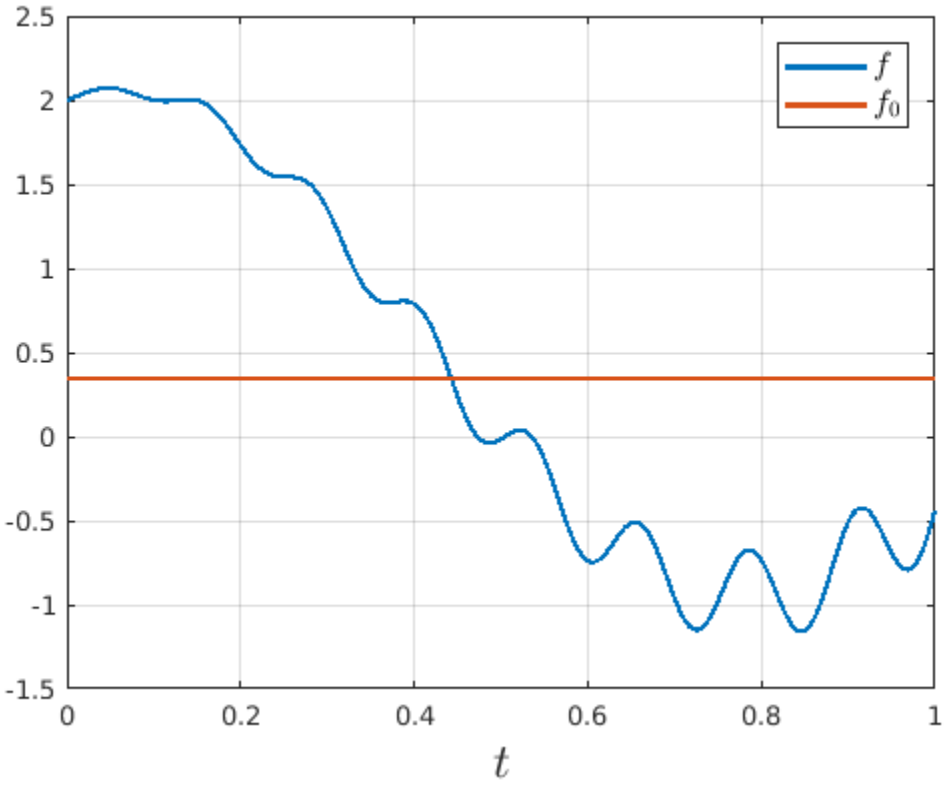

2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_5


2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_4


2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_3


2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_2


2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_1

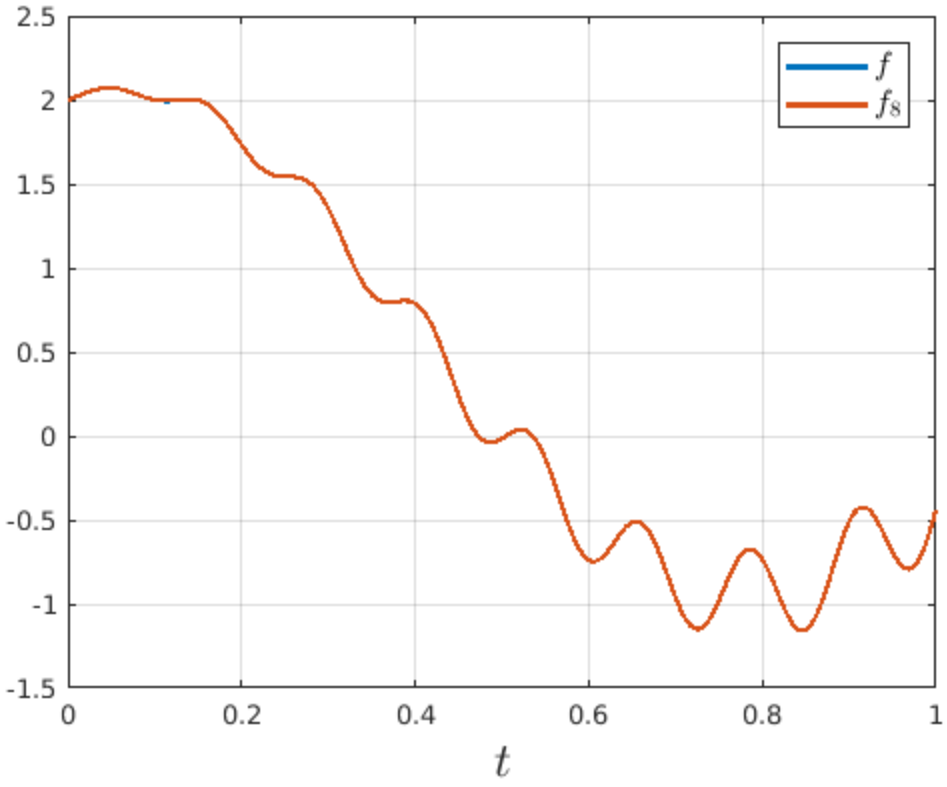

2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_0

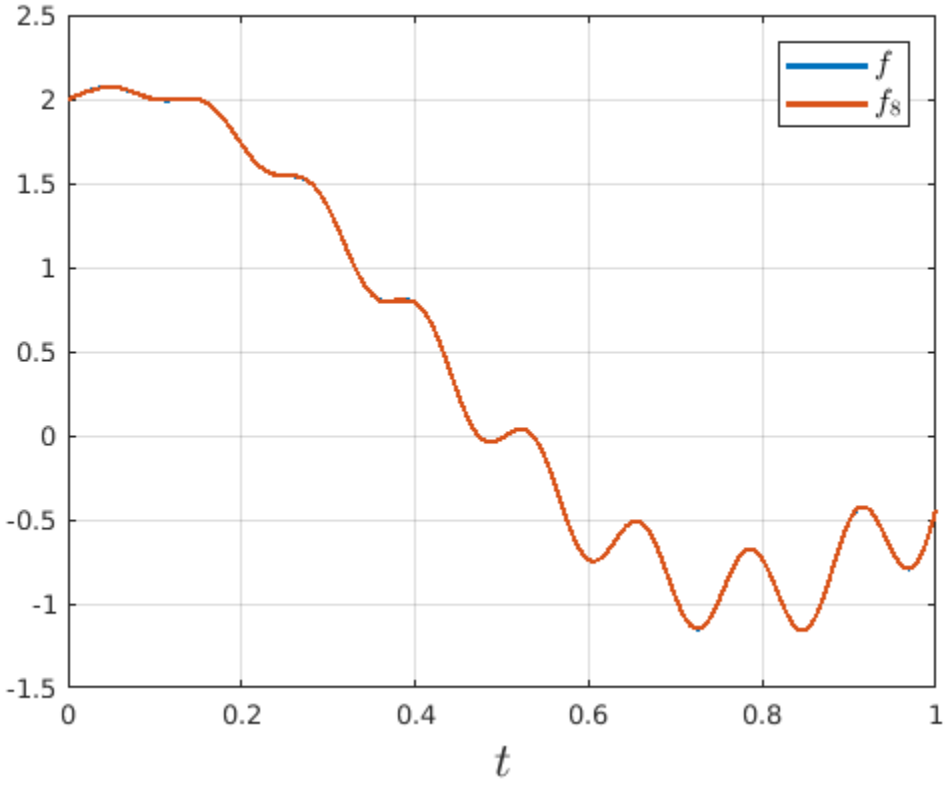
As we can see in the figures, the further we take the decomposition, the more inacurate the approximation. This is expected because for lower js the decomposed signal does not catch the higher frequencies.

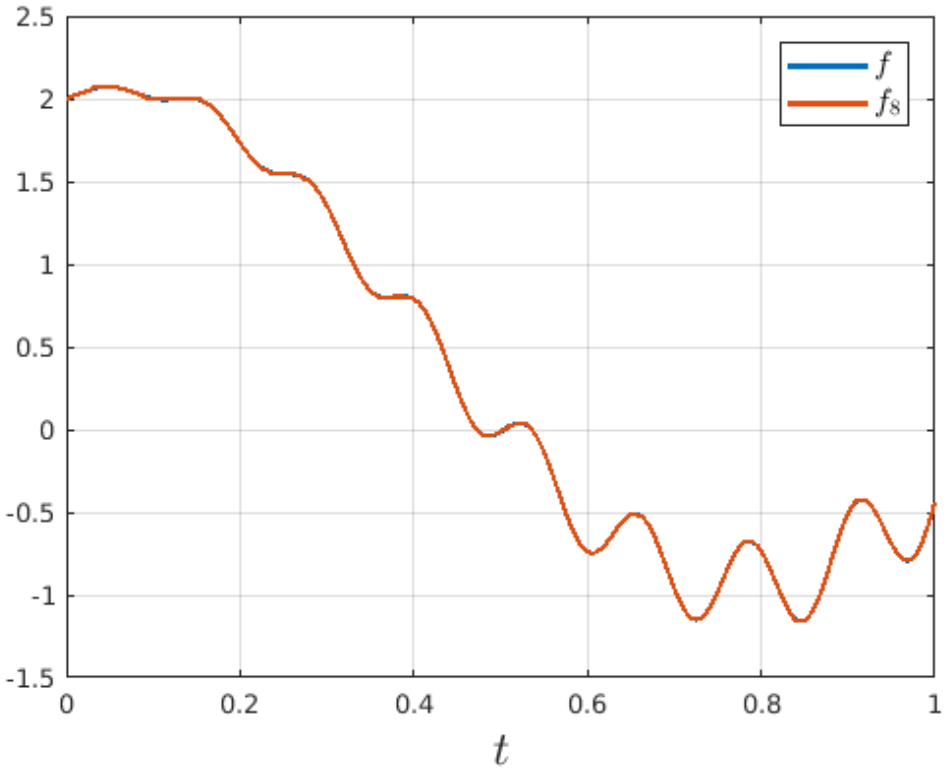
Filter out high frequencies (Problem 10)

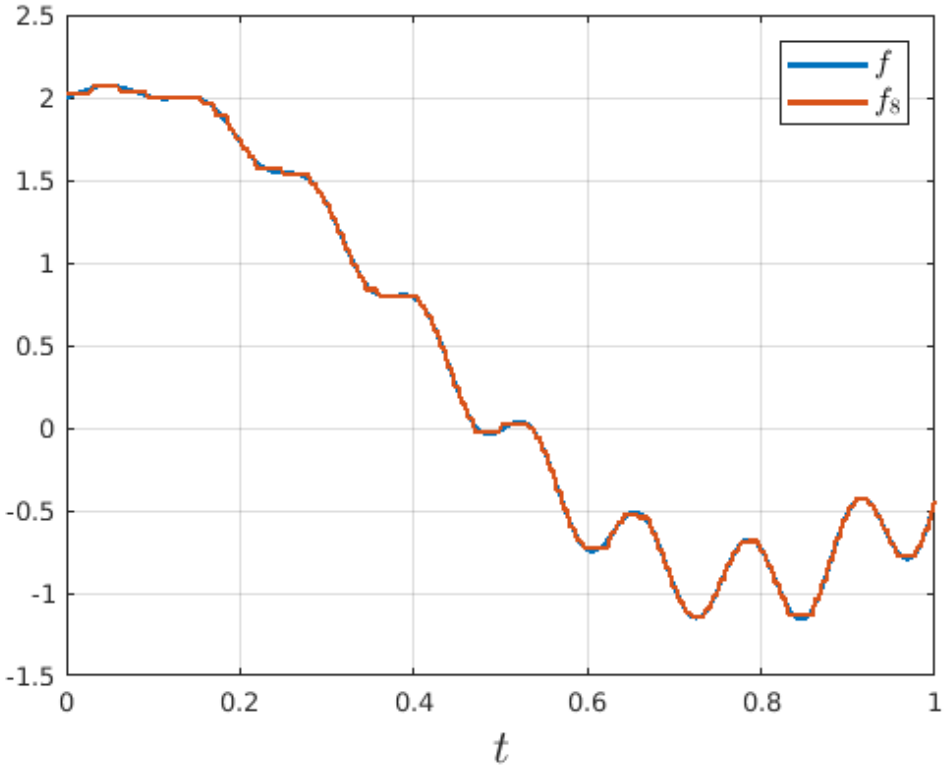
In this section we will filter out those coefficients smaller than a preset tolerance by setting them to zero. Once that is done, we will reconstruct the signal and compare it to the original one.

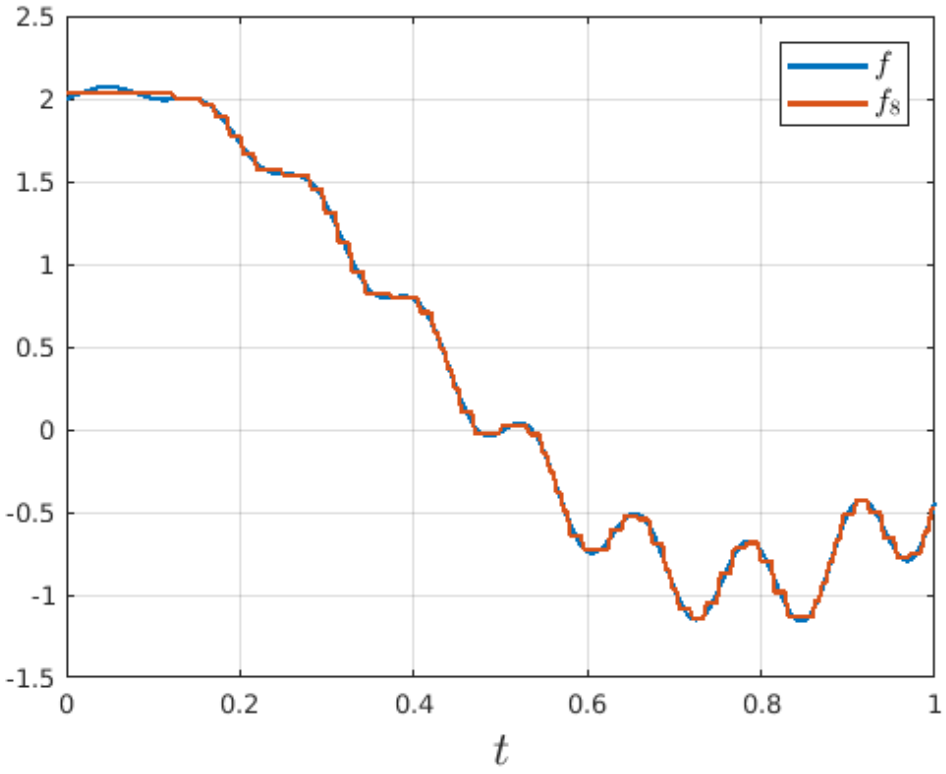
```
N=2^8;
tol=[1e-3 5e-3 1e-2 5e-2 1e-1 5e-1 1 5];
for i=1:length(tol)
    count(i)=0;
    [C,L]=wavedec(y,8,'db1');   %level 8 decomposition; analogous to
 full fft
    for j=1:N
        if (abs(C(j))<tol(i))
            C(j)=0;
            count(i)=count(i)+1;
        end
    end
    yc=waverec(C,L,'db1');   %reconstruct - analogous to ifft
    figure
    plot(t,y,t,yc,'linewidth',linewidth); % Plot original function and
 the i-th level decomposition
    grid on
    xlabel('$t$','interpreter','latex','fontsize',labelfontsize)
    axis([0 1 -1.5 2.5])
    h=legend('$f$','$f_8$');
    set(h,'interpreter','latex','fontsize',legendfontsize);
    saveas(gcf,['f_tol' num2str(i) '_p10'],'png')
    e(i)=norm(y-yc)/norm(y);
end
```

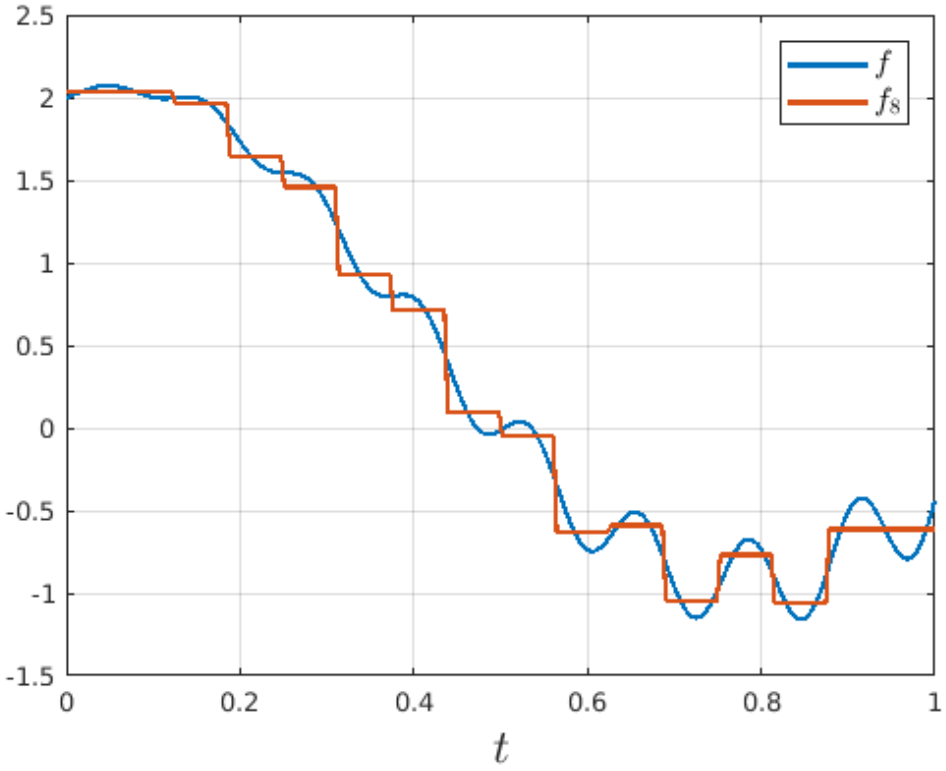

2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_8

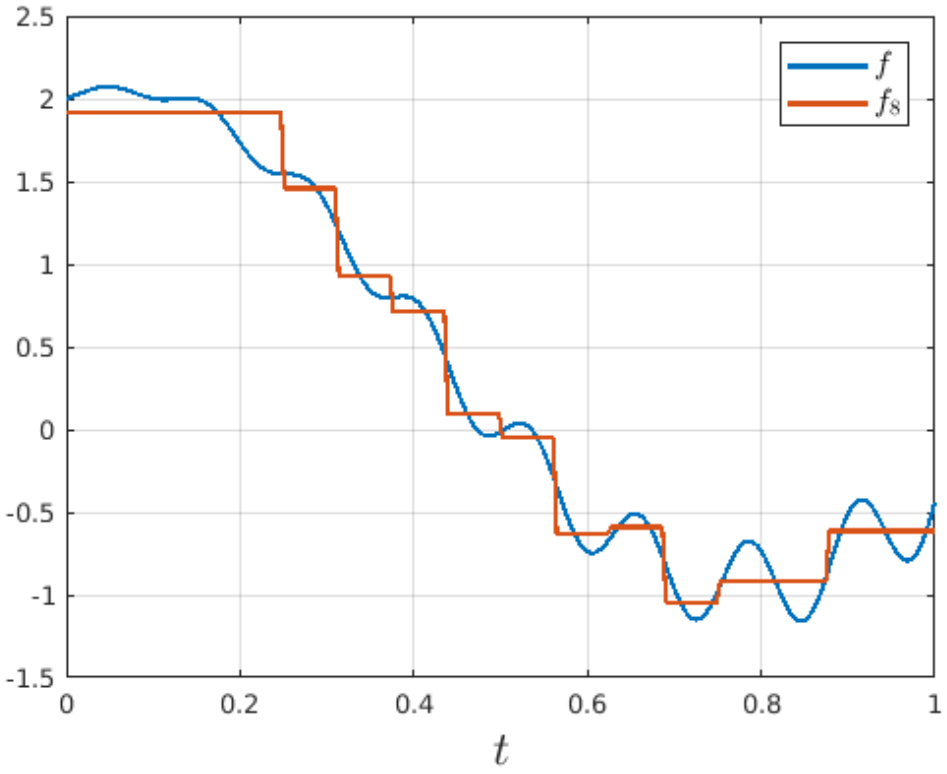

2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_8

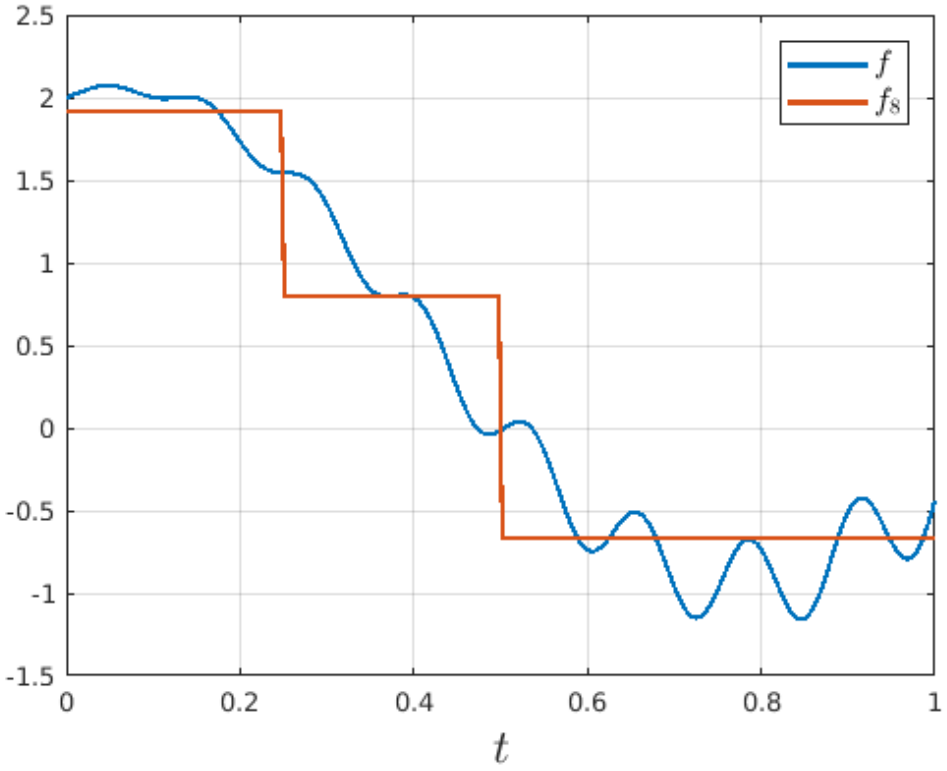

2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_8


2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_8


2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_8


2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_8


2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_8


2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.2
0.4
0.6
0.8
1
t
f
f_8

In the figures above it can be seen how we increase the tolerance and the reconstructed signal gets more and more inaccurate compared to the original signal. I have used the following tolerances:

```
tol
```

```
tol =

  Columns 1 through 7

    0.0010    0.0050    0.0100    0.0500    0.1000    0.5000    1.0000

  Column 8

    5.0000
```

Which have produced the removal of the following number of coefficients, respectively:

```
count
```

```
count =

     9    40    65   173   200   242   245   253
```

This represent, respectively, the following percentages of the total number of data points

```
percentage=count*100/N
```

```
percentage =

  Columns 1 through 7

    3.5156   15.6250   25.3906   67.5781   78.1250   94.5312   95.7031

  Column 8

   98.8281
```

To finish, the relative error obtained is detail below:

```
e
```

```
e =

  Columns 1 through 7

    0.0001    0.0009    0.0021    0.0158    0.0258    0.0843    0.1161

  Column 8
```

```
    0.2908
```

Published with MATLAB® R2017a