MATLAB CODE - FRANCISCO CASTILLO

Table of Contents

Preliminary Commands

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

Introduction

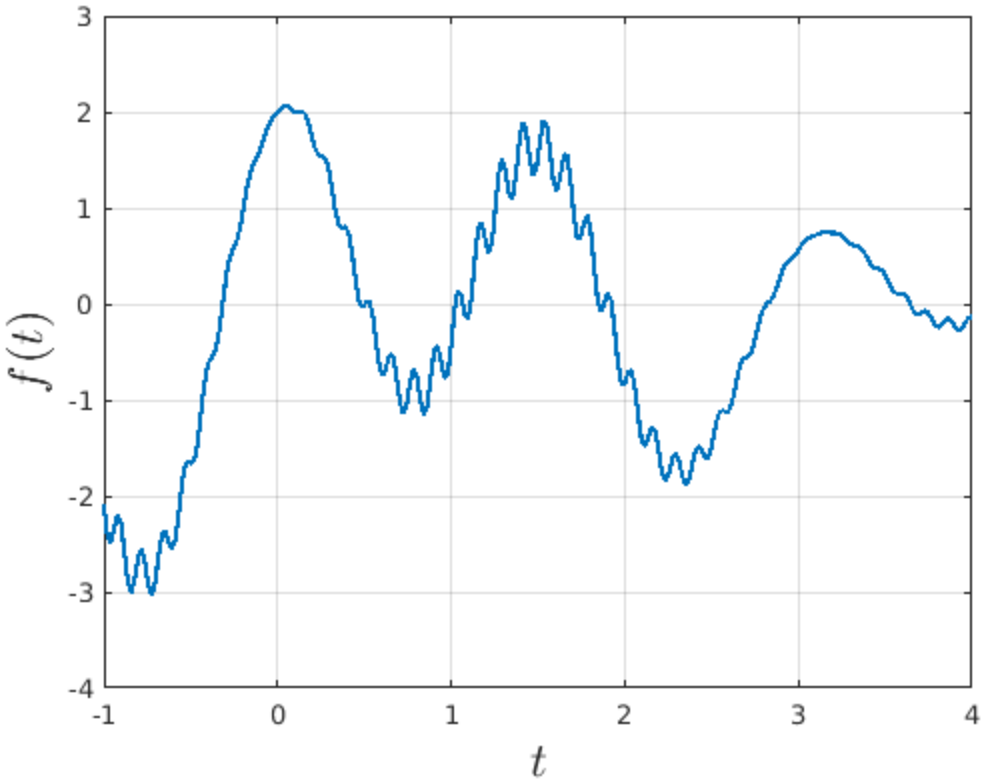
This code solves both problems 7 and 8 of the present homework assignment. As we can see, in the next section we define the function to filter (problem 7) and compress (problem 8).

Define the function

```
t=linspace(-1,4,256); % interval [-1,4] partitioned into 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
```

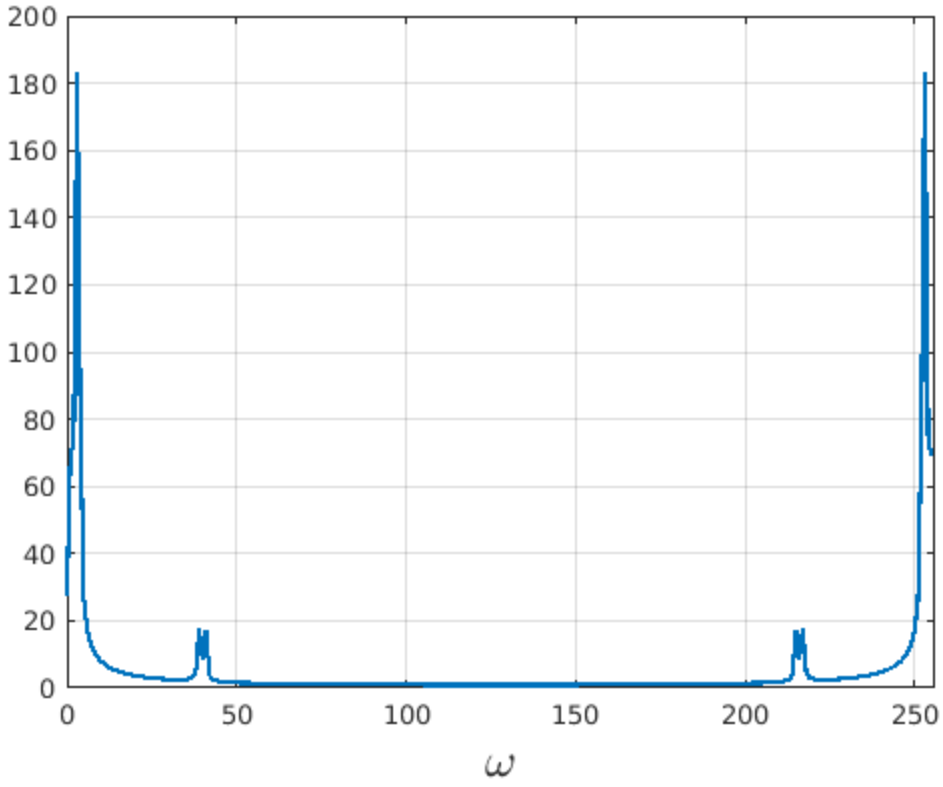


The goal in problem 7 will be filter the high frequency noise and obtain a filtered signal, which we will compare to the initial. In problem 8 we will take the filtered signal and compress by picking only the main frequencies. We will also compare it to the initial signal.

FFT Commands

In this section we simply compute the FFT of the signal f (shown in the next figure).

```
yhat=fft(y);
freq=0:255;
figure
plot(freq,abs(yhat),'linewidth',linewidth); % plot of the absolute
 value of frequencies
grid on
xlabel('$\omega$','interpreter','latex','fontsize',labelfontsize)
axis([0 256 0 200])
```



As we can see in the figure, the FFT does not give us reliable information of the high frequencies, they are simply a "copy" of the low frequencies. If we wanted to know how the FFT looks at frequency 256, we would need to sample up to N=528 (or more to be safe).

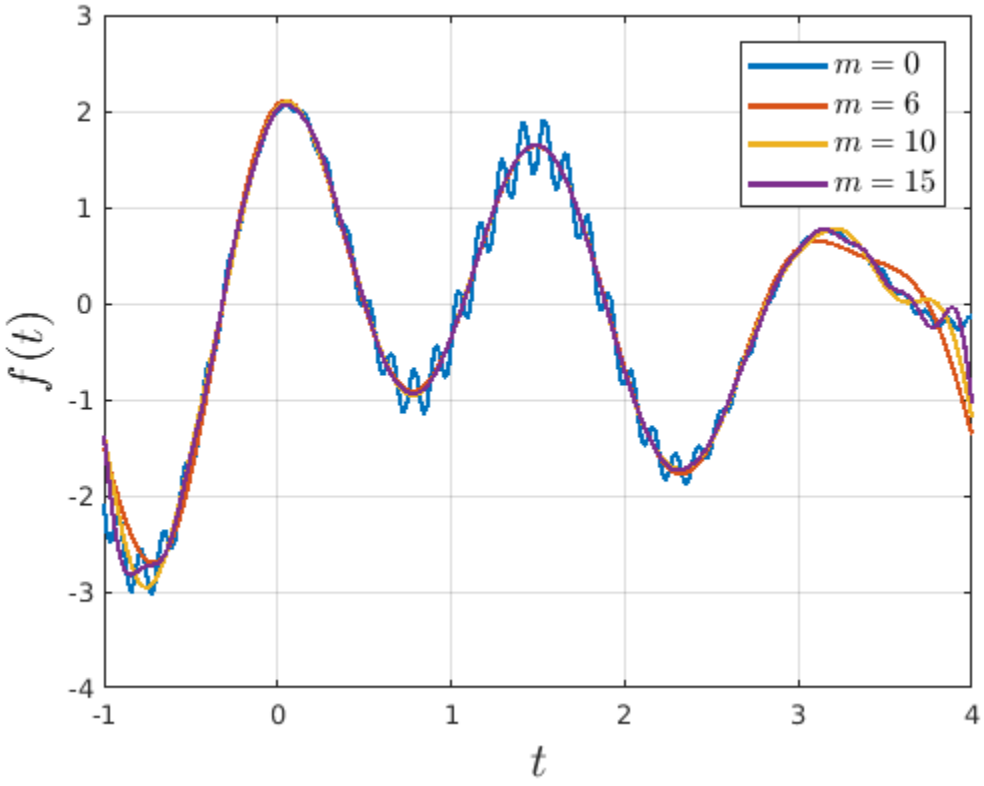
Filter out high frequencies (Problem 7)

In this section we will filter the m higher frequencies out. Once we have the filter Fourier transform of the signal, we use the inverse Fourier transform to obtain the filtered signal. In the figure, we can see the different filtered singals, depending on m, compared to the original signal $f(t)$ (corresponding to $m = 0$).

```
M=[6 10 20];
figure
plot(t,y,'linewidth',linewidth); % plot of original signal and
 filtered signal
grid on
hold on
for i=1:length(M)
    m=M(i);
    yhatf=[yhat(1:m), zeros(1,256-2*m), yhat(256-m+1:256)]; %keep
 lowes m frequencies
    yf=ifft(yhatf); % reconstruct filtered signal with ifft
    plot(t, yf,'linewidth',linewidth); % plot of original signal and
 filtered signal
end
xlabel('$t$','interpreter','latex','fontsize',labelfontsize)
```

```
ylabel('$f(t)$','interpreter','latex','fontsize',labelfontsize)
h1=legend('$m=0$','$m=6$','$m=10$','$m=15$');
set(h1,'interpreter','latex','fontsize',legendfontsize);
```

```
Warning: Imaginary parts of complex X and/or Y arguments ignored
Warning: Imaginary parts of complex X and/or Y arguments ignored
Warning: Imaginary parts of complex X and/or Y arguments ignored
```



As we can see, m only makes a difference on the extremes. This is due to the function not being periodic. Hence, the filtered signals match very well for average t values.

Compression (Problem 8)

In this part we will compress the signal f. This means we will take only those frequencies whose power is above the threshold tol. If they are not, they will be set to zero.

```
TOL=[1 2 4 8];
figure
plot(t,y,'linewidth',linewidth);
xlabel('$t$','interpreter','latex','fontsize',labelfontsize)
ylabel('$f(t)$','interpreter','latex','fontsize',labelfontsize)
grid on
hold on
for i=1:length(TOL)
    tol=TOL(i);
```

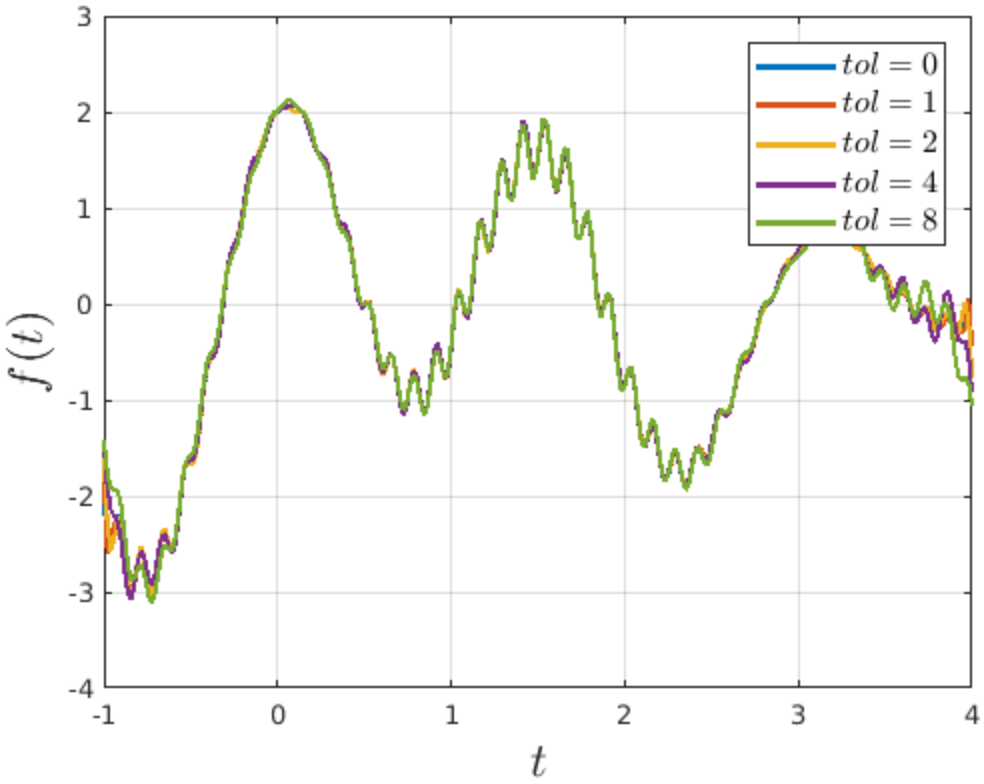
```
    count(i)=0; % for compression - keep frequencies which are larger
 than tol
    for j=1:256
        if (abs(yhat(j))<tol)
            yhatc(j)=0;
            count(i)=count(i)+1;
        else
            yhatc(j)=yhat(j);
        end
    end
    yc=ifft(yhatc);   %reconstruct compressed signal with ifft
    if tol==2
        yctol2=yc;
    end
    plot(t,yc,'linewidth',linewidth);    % plot of original and
 compressed signals
end
h1=legend('$tol=0$','$tol=1$','$tol=2$','$tol=4$','$tol=8$');
set(h1,'interpreter','latex','fontsize',legendfontsize);
TOL
count             %count = number of zeroed-out terms


TOL =

     1     2     4     8


count =

    87   167   211   231
```



We see how, except on the extremes, all compressions match the original signal very well, even having $tol = 8$ (which results in cancelling 236 out of 256 frequencies). Howevery, If we want to be more precies on the extremes we would have to take one of the first two values. The difference between the values of tol 1 and 2 is minimum. However, we cancel double as many terms using $tol = 2$. For this reason, having rather the same behaviour while having less data to store, I would pick a $tol = 2$.

To finish, we compute the relative l^2 error of the compressed signal, when using tol=2, as compared with the original signal

```
e = norm(y-yctol2,2)/norm(y,2)
```

```
e =

    0.0442
```

Published with MATLAB® R2017a