DOING PHYSICS WITH MATLAB

FOURIER ANALYSIS

FOURIER TRANSFORMS

Ian Cooper
matlabvisualphysics@gmail.com

DOWNLOAD DIRECTORY FOR MATLAB SCRIPTS

GitHub

Google Drive

maths_ft_02.m

mscript used to calculate the Fourier transform, the power spectral density and the inverse Fourier transform functions by the direct integration of the Fourier integrals using Simpson's rule. A wide variety of functions, sound files and data files (eg ecg) can be investigated. All parameters can be changed within the mscript.

Wave and mat Files

wav_S1000_1008.wav
audio440.wav
audioGuitar1.wav
audioClarinet1.wav
audioVoice1.wav
Train.wav
ecg.mat

The Script calculates the Fourier transform H(f) for different signal functions h(t). The signal function is selected by the variable flagF (flagF = 1, 2, … , 17):

```
%  1   Gaussian function
%  2   Exponential function
%  3   Sinusoidal function
%  4   Superposition of sinusoidal functions
%  5   Square Wave
%  6   Sawtooth function
%  7   Single square pulse
%  8   Damped sinusoidal function
%  9   ECG
% 10   Beats
% 11   Beats: audio file
% 12   Audio file: 440 Hz signal
% 13   Audio file: Guitar 220 Hz
% 14   Audio file: Clarinet 220 Hz
% 15   Audio file: Voice 220 Hz
% 16   Audio file: train whistle
% 17  Digital Filtering
```

It is very easy to add other functions to the Script

Inputs for each function

Enter function or load data file h
Time domain: number of grid points for time (must be an odd number) nT; start time tMin = 0; end time tMax
Frequency domain: number of grid points for frequency (must be an odd number) nF; start time fMin = 0; end time fMax
Plots: XLIMS = [tMin tMax]; XTICKS = tMin:dt:tMax;
XLIMSF = [fMin fMax]; XTICKSF = fMin:df:fMax;

Audio files .wav

The signal for the audio recording is obtained using the load command

```
[signal, Fs] = audioread('audio440.wav');
```

The variable signal is the input function of time. Fs is the sampling frequency or frequency rate. The time interval dt between the signal data points is thus

```
dt = 1/Fs;
```

The input variable h is extracted from the signal function

```
h = signal (h1:h2);
```

where h1 and h2 are the start and end indices respectively. The number of time steps nT is given by the length of h

```
nT = length(h);
```

and the maximum time span tMax is

```
tMax = (nT-1)*dt;
```

The code to play the audio recording is

```
sound(signal, Fs);
```

Introduction

Fourier transform methods (or spectral methods) are used in a very large class of computational problems. Many physical processes can be described in the **time domain** by the values of a function $h(t)$ or else in the **frequency domain** $H(f)$. The function $H(f)$ is usually complex function specified by its amplitude and phase. It is useful to think of $h(t)$ and $H(f)$ as being two different representations of the same function. One goes back and forth between these two representations by means of the **Fourier transform equations**

$$(1) \qquad H(f) = \int_{-\infty}^{\infty} h(t) e^{i(2\pi f t)}\, dt$$

$$(2) \qquad h(t) = \int_{-\infty}^{\infty} H(f) e^{-i(2\pi f t)}\, df$$

Equation 1 is the **Fourier transform** and equation 2 gives the **inverse Fourier transform**. If t is measured in seconds, then the frequency f is measured in hertz. It is more straight forward to use the frequency f rather than the more commonly used angular frequency ω $(\omega \equiv 2\pi f)$.

The total power in a signal is the same whether we compute it in the time domain or in the frequency domain. This result is called **Parseval's theorem**

(3) $$Total\ Power = \int_{-\infty}^{\infty} |h(t)|^2\, dt = \int_{-\infty}^{\infty} |H(f)|^2\, df$$

If one wants to compute how much power there is the frequency interval between *f* and *df*, one does not usually distinguish between negative and positive values of *f*, but rather regards *f* as varying from zero frequency or DC $(f = 0)$ to $f \to \infty$. In such cases, one defines the **one-sided power spectral density** PSD of the function $h(t)$ as

(4) $$psd(f) = |H(f)|^2 + |H(-f)|^2$$

If $h(t)$ is a real function, then

(5) $$psd(f) = 2|H(f)|^2 \qquad 0 \le f < \infty$$

(6) $$Total\ Power = 2\int_0^{\infty} |H(f)|^2\, df$$

Usually, the Fourier transforms given by equations 1 and 2 are calculated by the **fast Fourier transform** method. There are Matlab functions **fft** and **ifft** that can be implemented to find the Fourier transforms. However, they are not easy to use as the sampling rate and frequency domain are **not** independent.

Historically, the fast Fourier transform is used because of the speed of the calculations is much faster than the direct

evaluation of the Fourier integrals. But, with the speed and memory of modern computers and using software such as Matlab, the computation of the Fourier integrals can be by the direct integration without any problems, thus, **Fourier Analyse can be made simple**.

Matlab

The time and frequency domains are specified by the variables for the time interval and number of samples:

time domain $tMin \;\; tMax \;\; nT$

frequency domain $-fMax \;\; fMax \; nF$

The number of samples nT and nF must be **odd** numbers for evaluating the integrals using Simpson's rule. Hence, the frequency domain includes both negative and positive frequencies in computing the inverse Fourier transform (equation 2) and the total power (equation 6).

The function $h(t)$ has only non-zero values in the interval time interval from $tMin$ to $tMax$

$$t \le tMin \quad h(t) = 0 \qquad t \ge tMax \quad h(t) = 0$$

The function $h(t)$ is specified by the variable $flagF$ in the switch/case statements. The defined function **simpson1d.m** uses Simpson's rule to evaluate each integral.

```
% FOURIER TRANSFORM CALCULATIONS ===============
   H = zeros(1,nF); hI = zeros(1,nT);HT = zeros(1,nF);

% Fourier Transform  H(f)
   for c = 1:nF
     g = h.* exp(1i*2*pi*f(c)*t);
     H(c) = simpson1d(g,tMin,tMax);
   end

% INVERSE Fourier Transform  hI(t)
for c = 1:nT
   g = H.* exp(-1i*2*pi*t(c)*f);
   hI(c) = simpson1d(g,fMin,fMax);
end
```

The code for the power calculations:

```
% One-sided power spectral density PSD  Ph(f)
   psd = 2.*conj(H).*H;

% Total power  PT (time domain) and PF (frequency domain)
   PT = simpson1d(h.^2,tMin,tMax);
   PF = simpson1d(Ph,fMin,fMax)./2;

   fprintf('PT = %4.4f  \n \n',PT);
   fprintf('PF = %4.4f  \n \n',PF);
```

EXAMPLES

1. Gaussian Function $h(t) = \exp\left(-\pi t^2\right)$

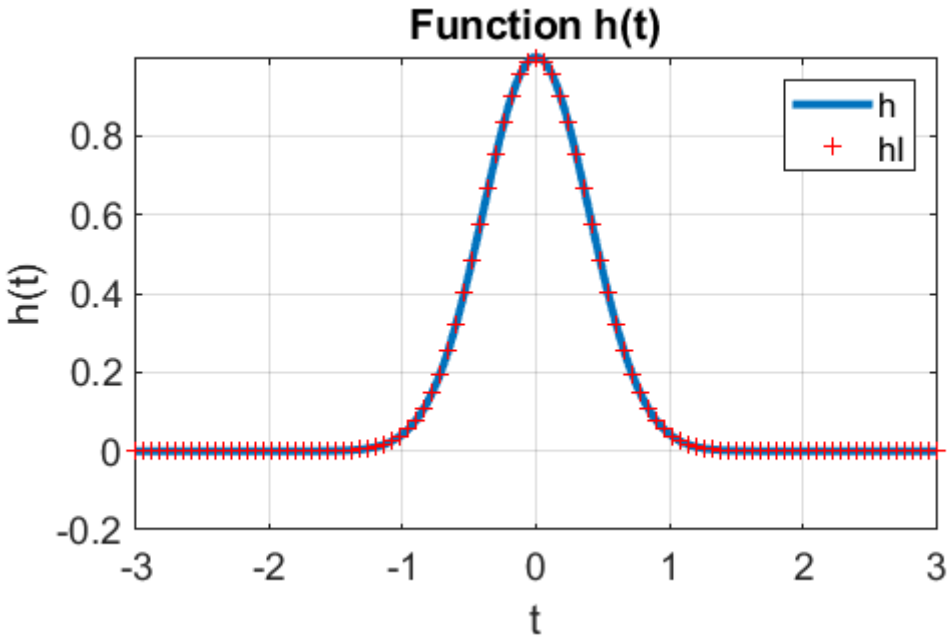


Fig. 1.1. A plot of a Gaussian function and its inverse Fourier transform.

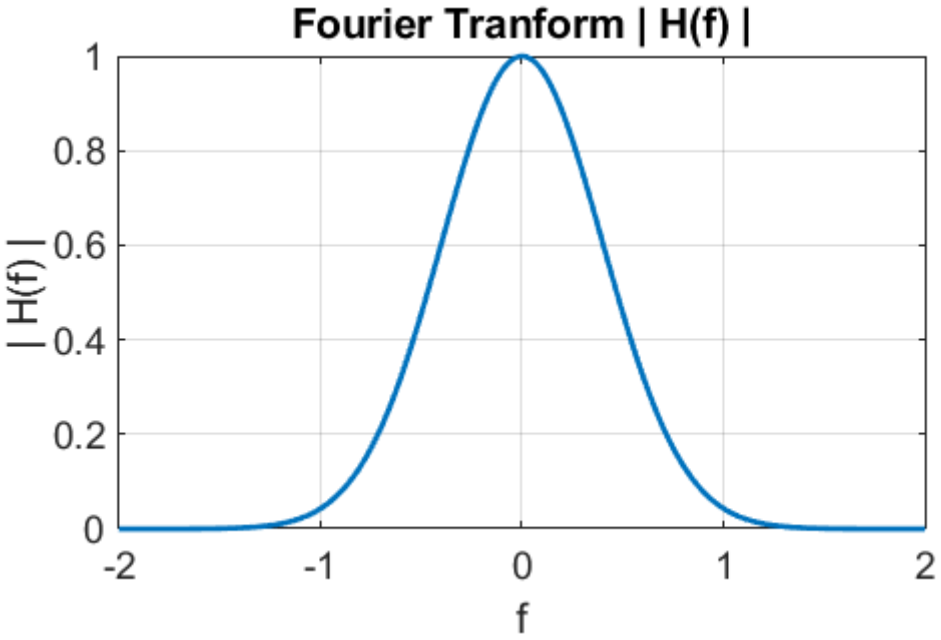


Fig. 1.2. The Fourier transform $|H(f)|$ of the function $h(t)$

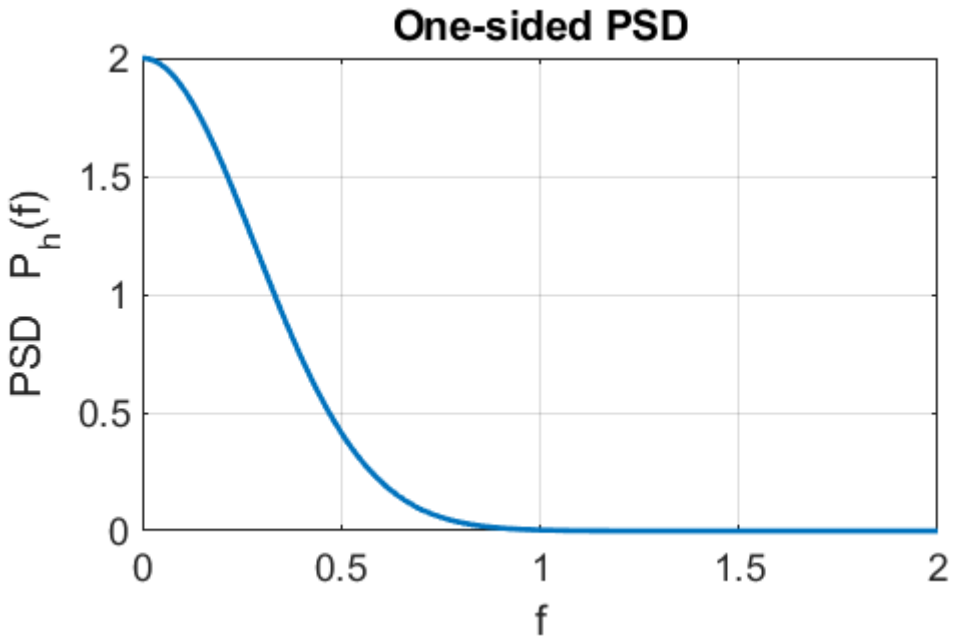


Fig. 1.3. One sided power spectral density PSD, $P_h(f)$. Only the positive frequency interval is displayed.

The total powers in the signal calculated from equation 3 are displayed in the Command Window

time domain PT = 0.7071

frequency domain PF = 0.7071

The execution time for the computation using the tic / toc commands is less than one second.

A narrow Gaussian signal has a wide spectrum whereas a wide Gaussian signal has a narrow spectrum such that $\Delta t \, \Delta f > K$ where Δ gives the widths and K is some positive constant.

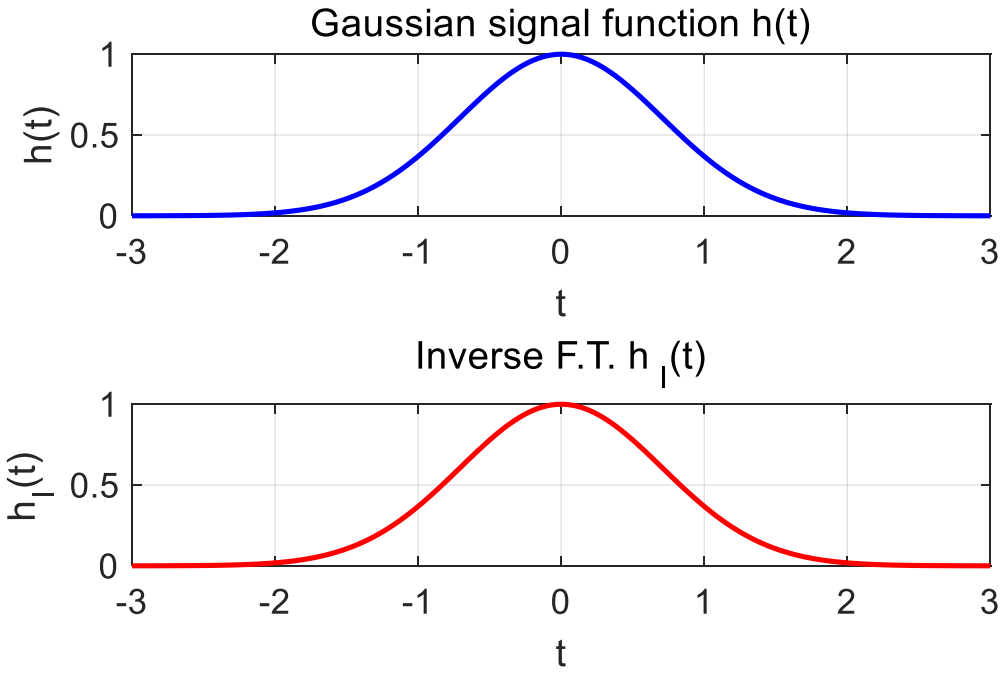


Fig. 1.4. Wide signal.

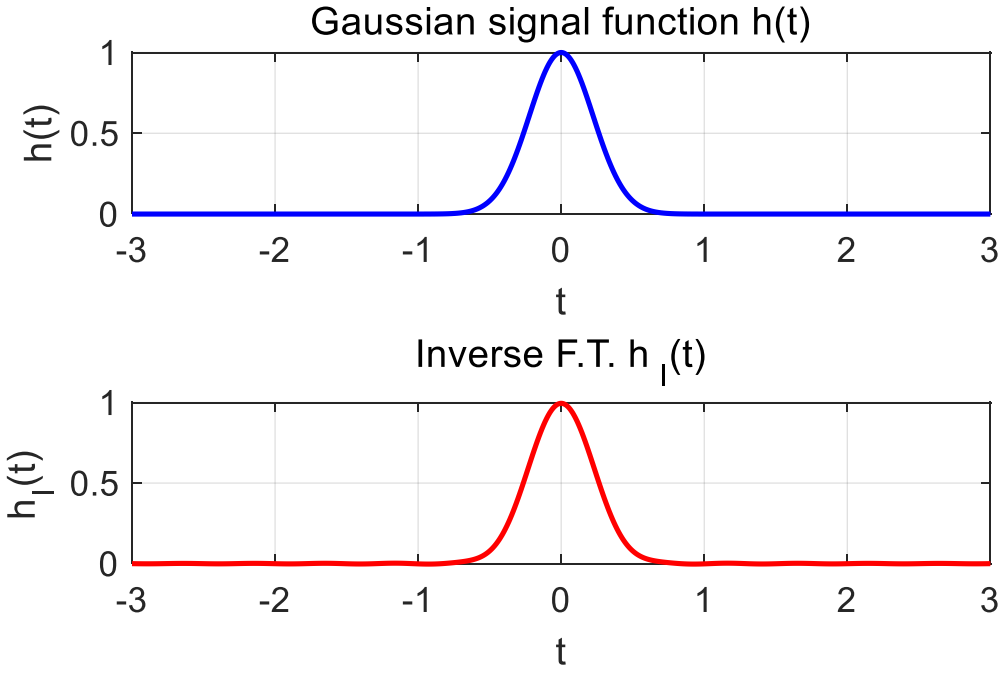


Fig. 1.5. Narrow signal.

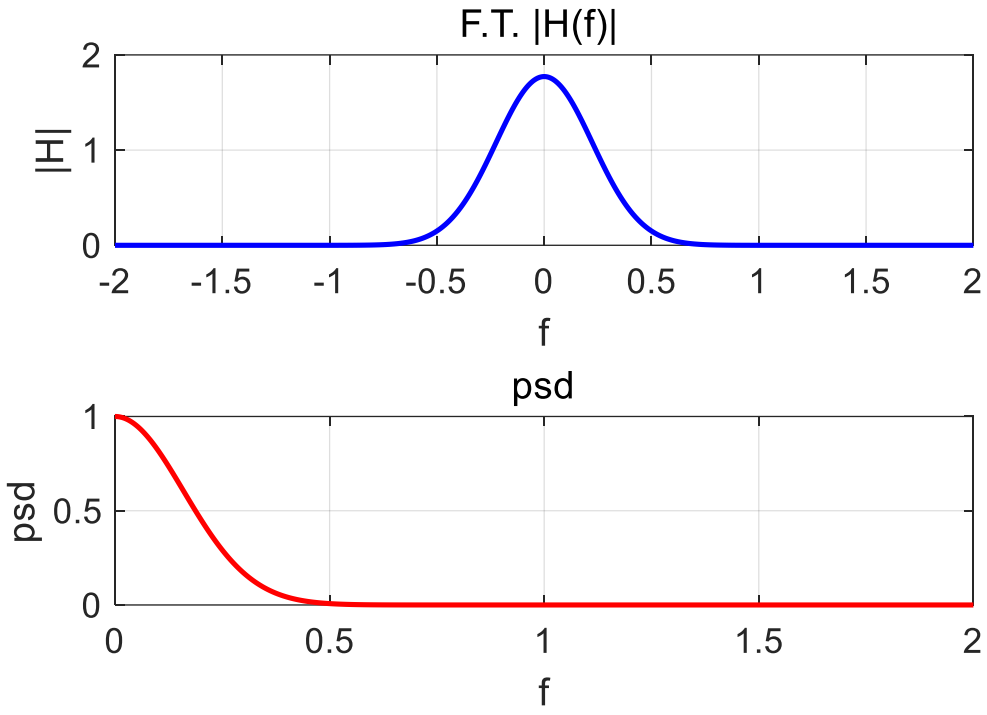


Fig. 1.6. Narrow spectrum for narrow pulse.

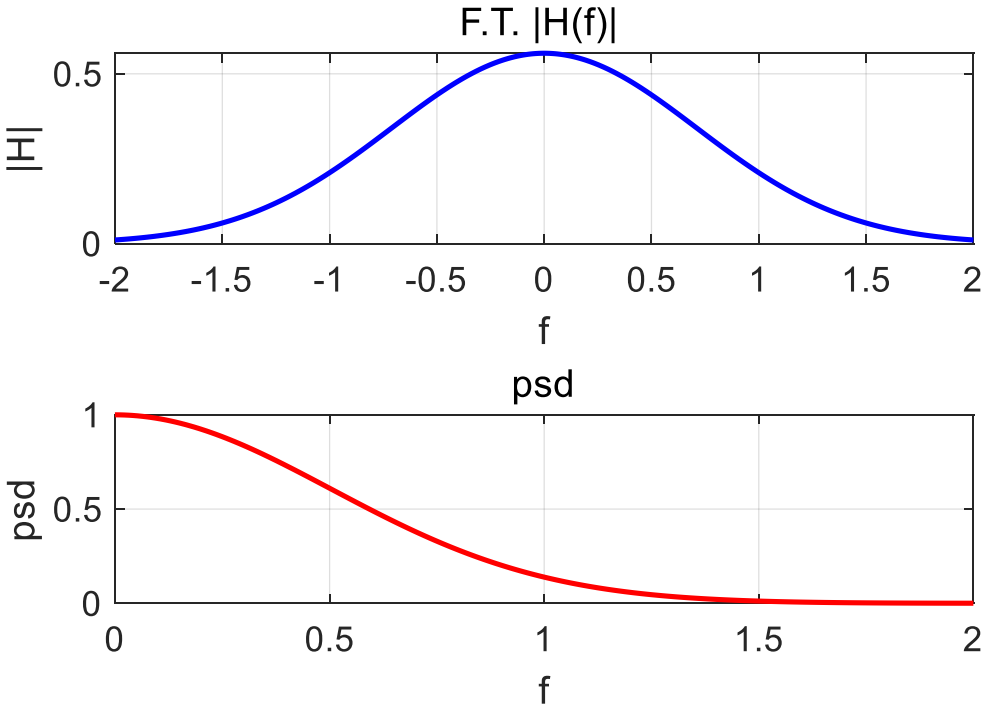


Fig. 1.7. Wide spectrum for narrow pulse.

2. EXPONENTIAL FUNCTION $h(t) = 2\exp(-3t)$

We can compute the continuous Fourier transform of an exponential function such as

$$h(t) = 2\, e^{-3t}$$

We can test our numerical estimate of the Fourier transform with the analytically estimate given by

$$H(f) = \frac{2}{3 + i(2\pi f)}$$

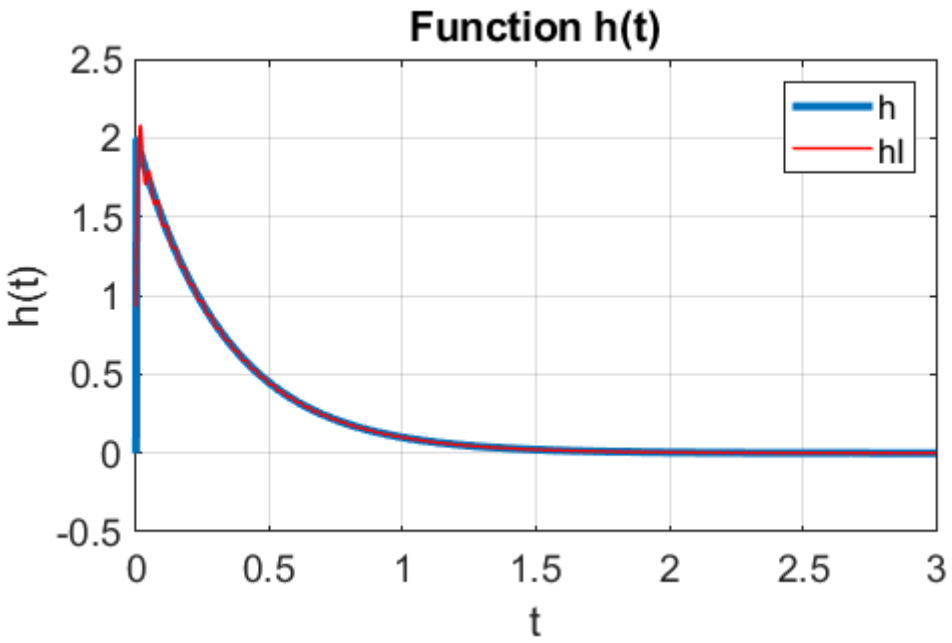


Fig. 2.1. The function $h(t)$ and the inverse Fourier transform $hI(t)$.

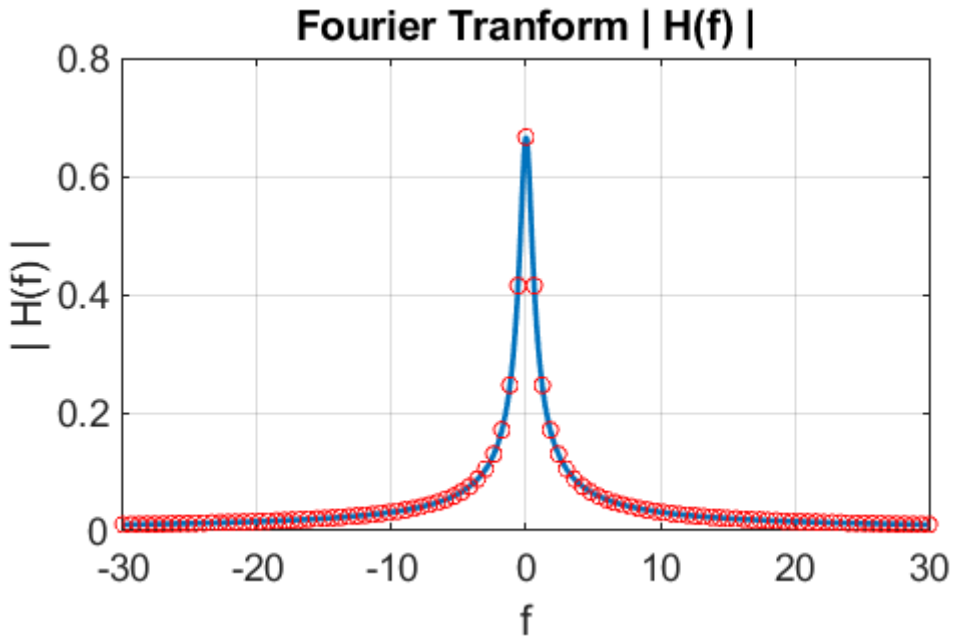


Fig. 2.2. The Fourier transform showing excellent agreement between the numerical results and the analytical prediction.

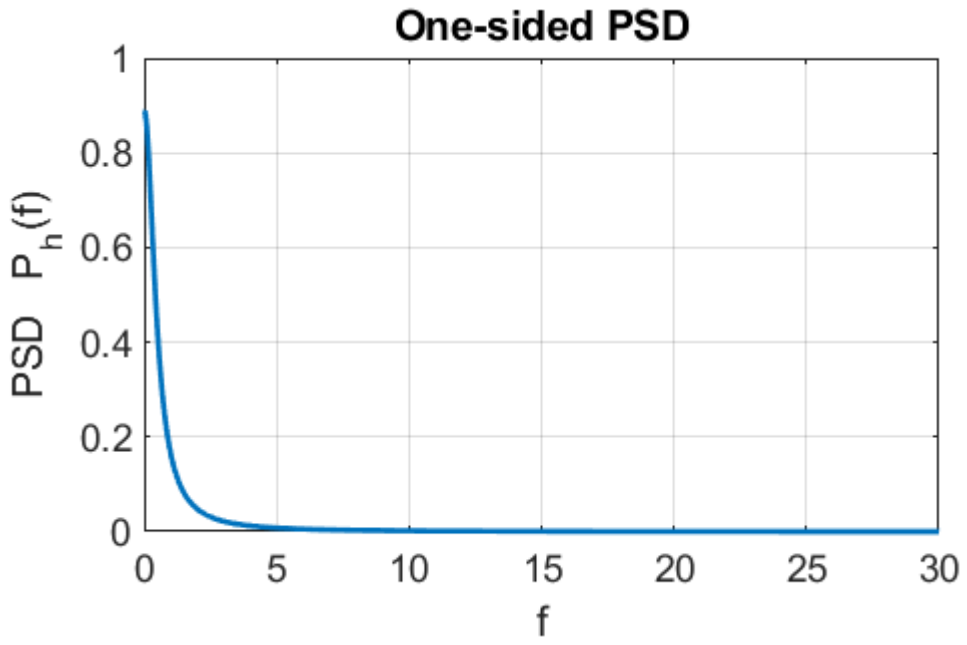


Fig. 2.3. One sided power spectral density PSD, $P_h(f)$. Only the positive frequency interval is displayed.

3. Sinusoidal functions $h(t) = A_0 \sin(2\pi\, f_0\, t + \phi_0)$

We can easily computer the Fourier transform of the sinusoidal function expressed in the form

$$h(t) = A_0 \sin(2\pi\, f_0\, t + \phi_0)$$

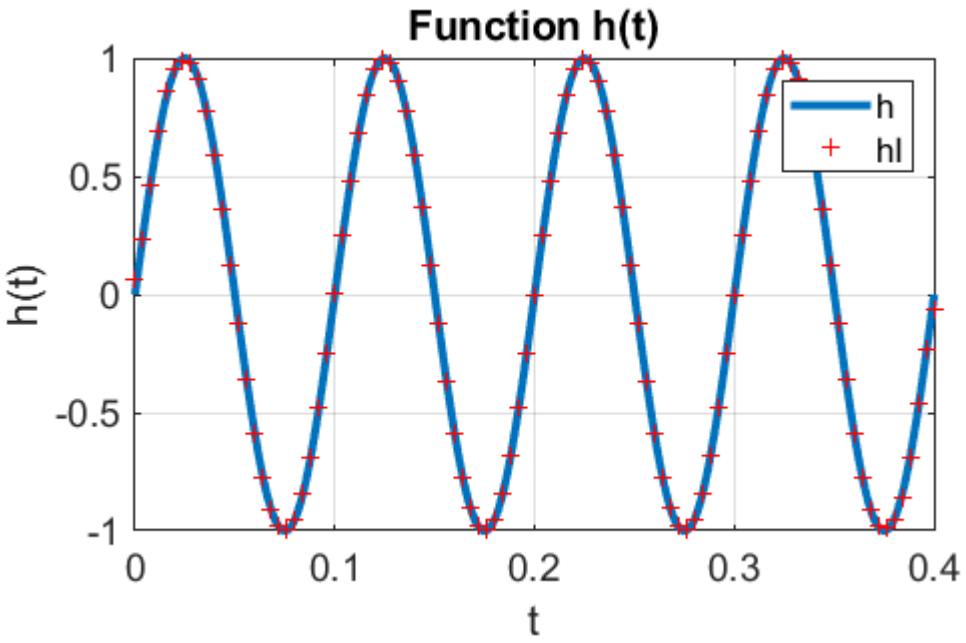


Fig. 3.1. The function $h(t) = A_0 \sin(2\pi\, f_0\, t + \phi_0)$ where $A_0 = 1$ $f_0 = 10$ Hz $\phi_0 = 0$ rad. A = 1. The units for time t are seconds.

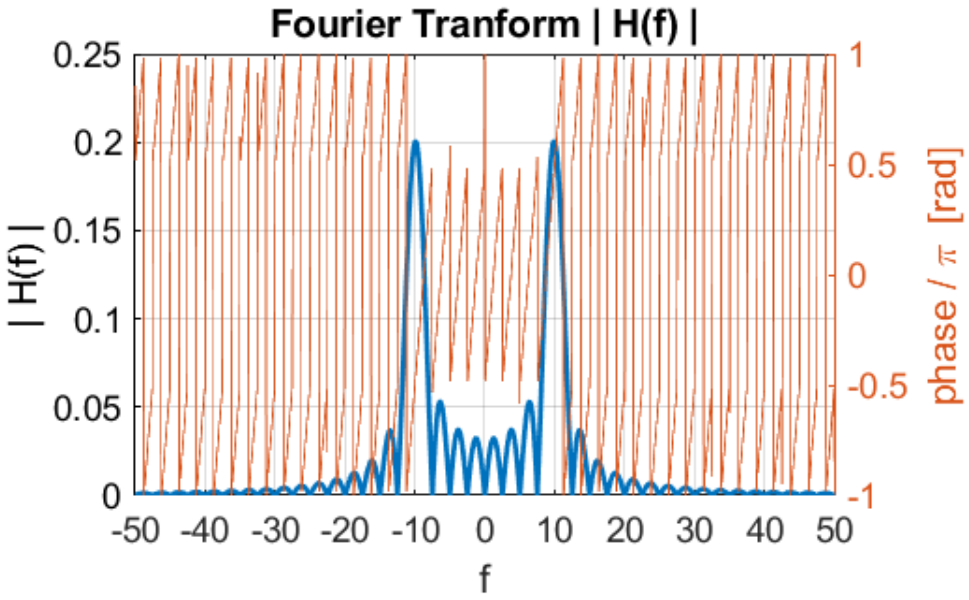


Fig. 3.2. Absolute value of the Fourier transform and its phase. To plot the phase, uncomment the code in the segment for figure 2.

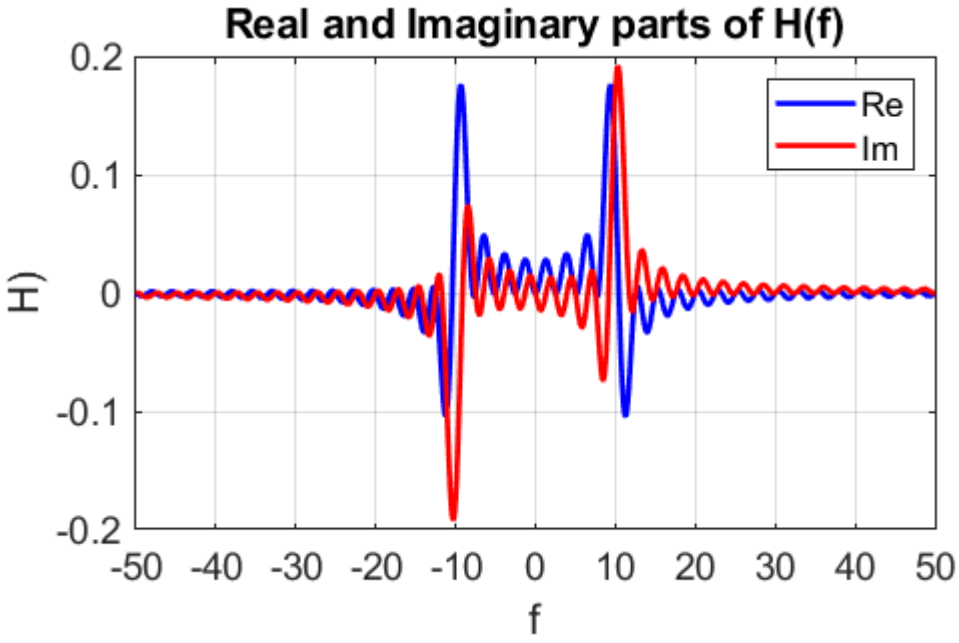


Fig. 3.3. Real and imaginary parts of the Fourier transform.

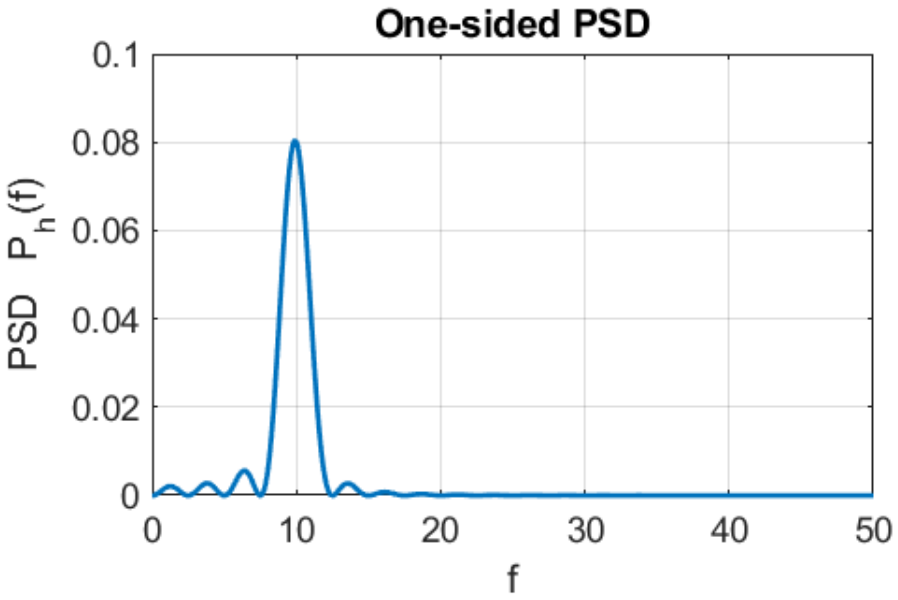


Fig. 3.4. One sided power spectral density PSD, $P_h(f)$. Only the positive frequency interval is displayed.

The total powers in the signal calculated from equation 3 are displayed in the Command Window

time domain PT = 0.2000

frequency domain PF = 0.2000

Doubling the amplitude: A = 2

time domain PT = 0.8000

frequency domain PF = 0.7999

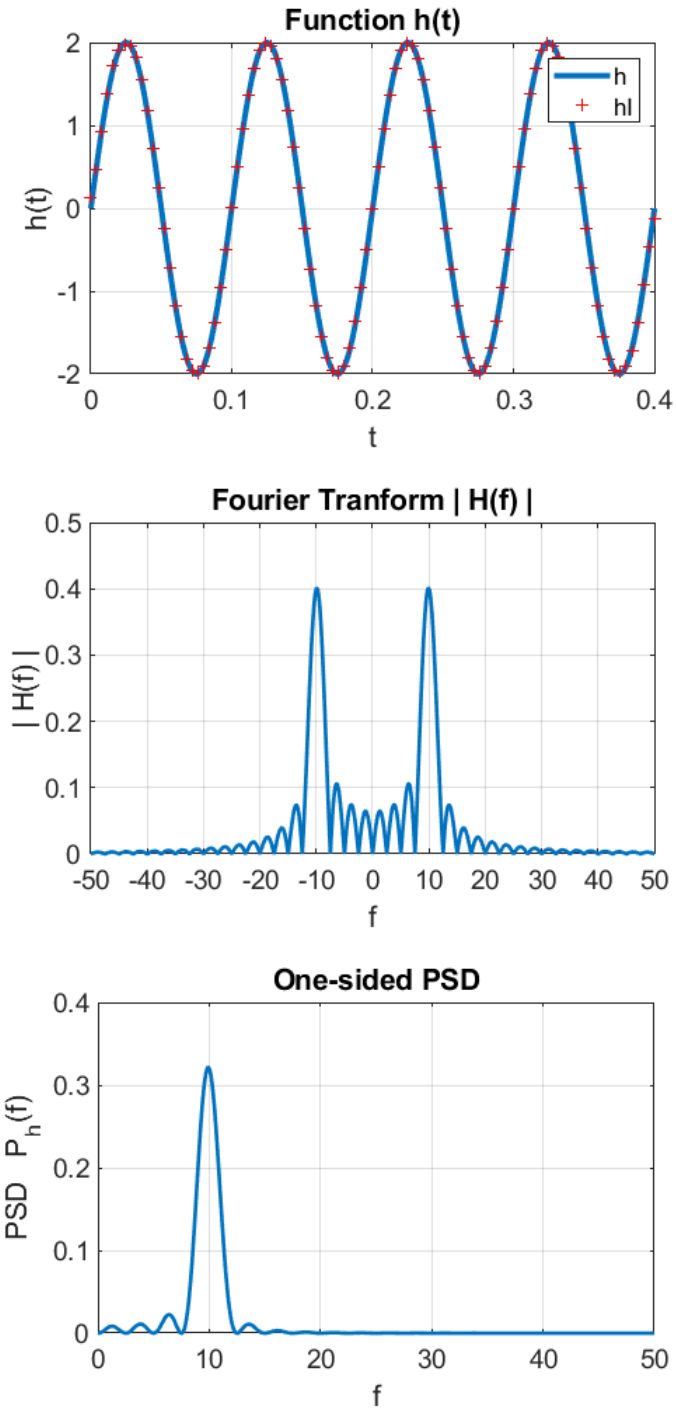


Fig. 3.5. When the amplitude is doubled, the energy supplied by the signal increases by a factor of 4 [$P \propto A^2$].

Doubling the frequency: f = 20 Hz

time domain PT = 0.2000

frequency domain PF = 0.1999

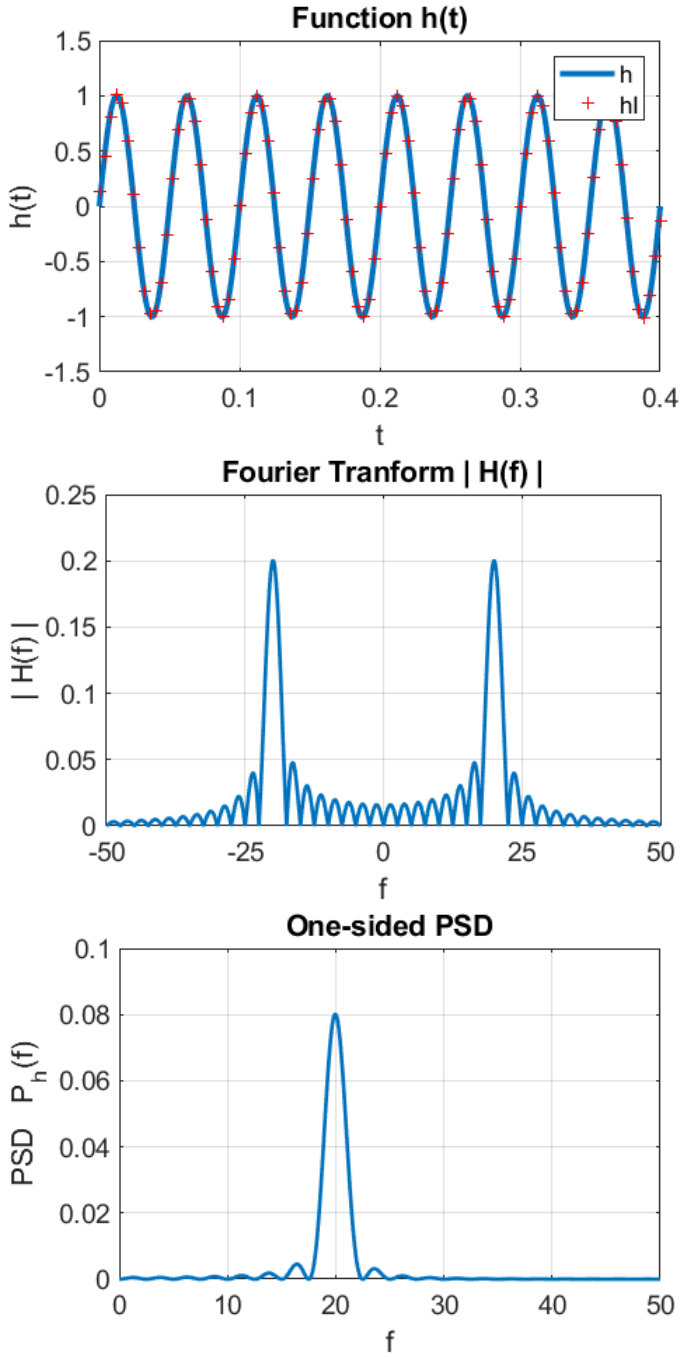


Fig. 3.6. When the frequency is doubled, the energy supplied by the signal does not change (not increase x4 ???).

Changing the phase: $\phi_0 = \pi / 6$

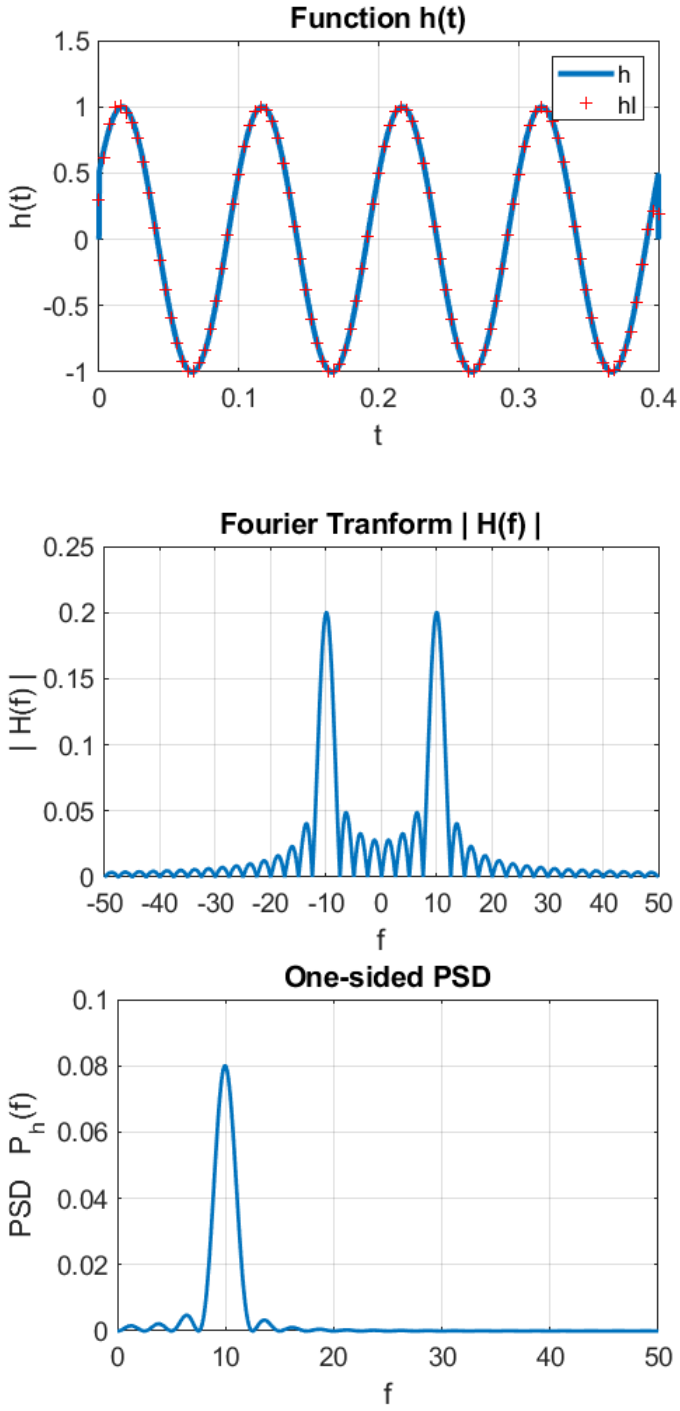


Fig. 3.7. Changing the phase of the sinusoidal function has negligible effect on the Fourier transform.

4. Superposition of sinusoidal functions

Consider the signal which is the superposition of four sinusoidal functions where Hz $f_0 = 20$.

$$h(t) = A_1 \sin\left(2\pi\, f_0 t\right) + A_2 \sin\left(2\pi (2 f_0) t\right) + A_3 \sin\left(2\pi (3 f_0) t\right) + A_4 \sin\left(2\pi (4 f_0) t\right)$$

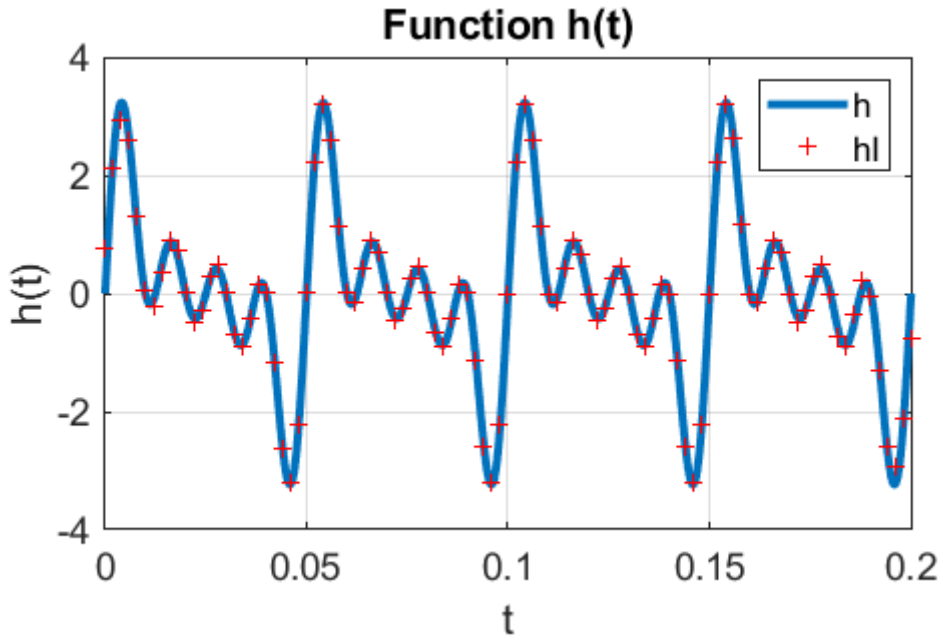


Fig. 4.1. The superposition of four sinusoidal function:
$f_0 = 20\ Hz \quad A_1 = A_2 = A_3 = A_4 = 1$

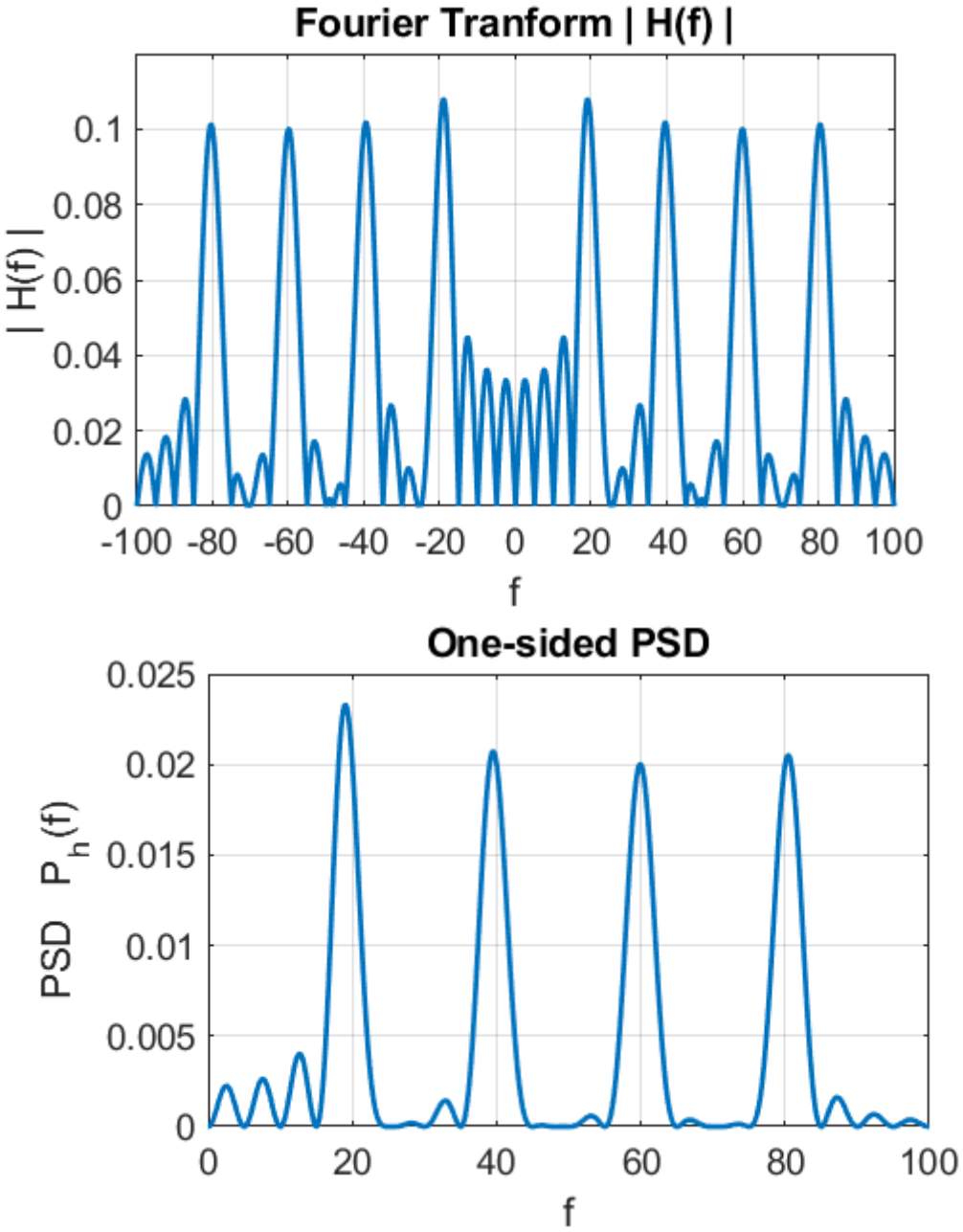


Fig. 4.2. The absolute value of the Fourier transform and the PSD function.

The total powers in the signal calculated from equation 3 are displayed in the Command Window

time domain PT = 0.4000

frequency domain PF = 0.3972

Simulation with parameters:

$f_0 = 20\ Hz$

$A_1 = 1 \quad f_1 = 20 \quad A_2 = 2 \quad f_2 = 40$

$A_3 = 3 \quad f_3 = 60 \quad A_4 = 4 \quad f4 = 80$

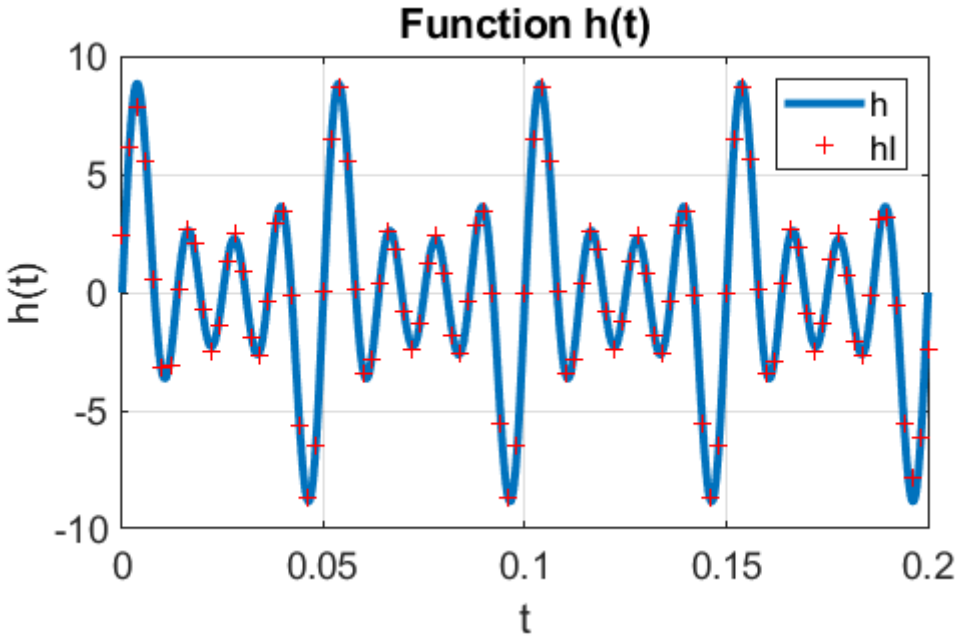


Fig. 4.3. The function and the inverse Fourier transform.

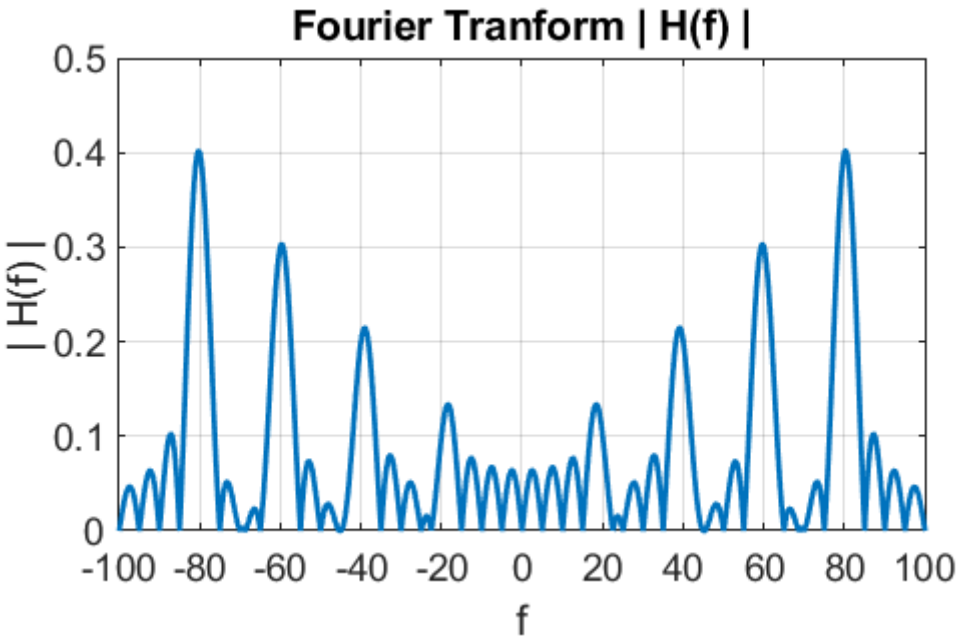


Fig. 4.4. The Fourier transform.

The major four peaks in figure 16 can be found in the Command Window using the statements

```
[a b]=findpeaks(abs(H),'MinPeakHeigh',0.13)
f(b)
a./a(4)
```

The frequencies of the peaks and their relative heights […] are:

18.4 [1.0] 39.1 [1.6] 59.6 [2.3] 80.3 [3.0]

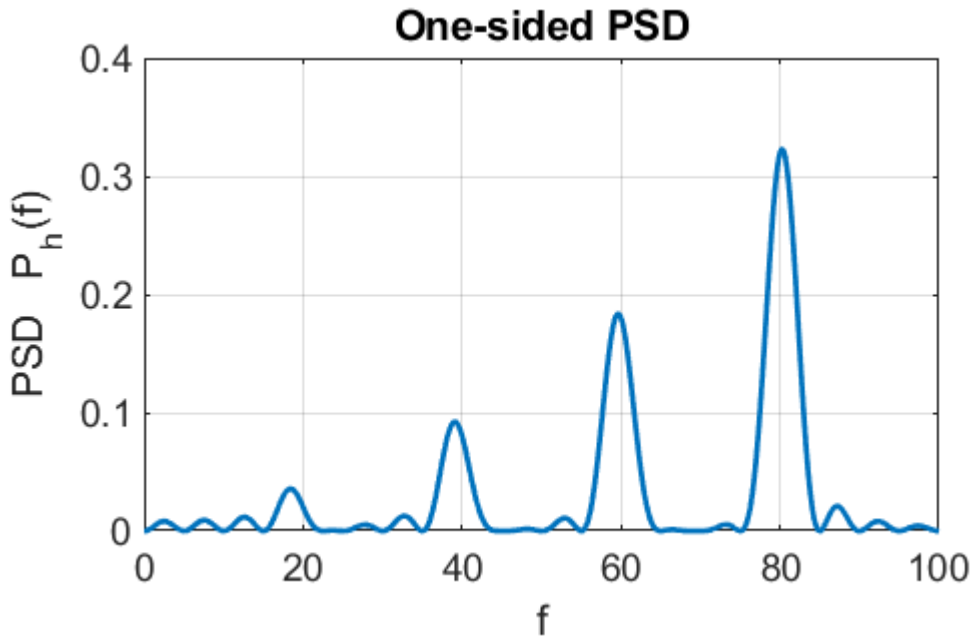


Fig. 4.5. The PSD function.

The major four peaks in figure 17 can be found in the Command Window using the statements

```
[a b] = findpeaks(Ph,'MinPeakHeigh',0.03)
f(b)
a./a(4)
```

The frequencies of the peaks and their relative heights […] are:

18.4 [1.0] 39.1 [2.6] 59.6 [5.1] 80.3 [9.1]

5. Square wave

A square wave can be approximated by a Fourier series of the form

$$h(t) = \sin\left(2\pi f_0 t\right) + \frac{\sin\left(2\pi (3 f_0) t\right)}{3} + \frac{\sin\left(2\pi (5 f_0) t\right)}{5} + \frac{\sin\left(2\pi (7 f_0) t\right)}{7} + \ldots$$

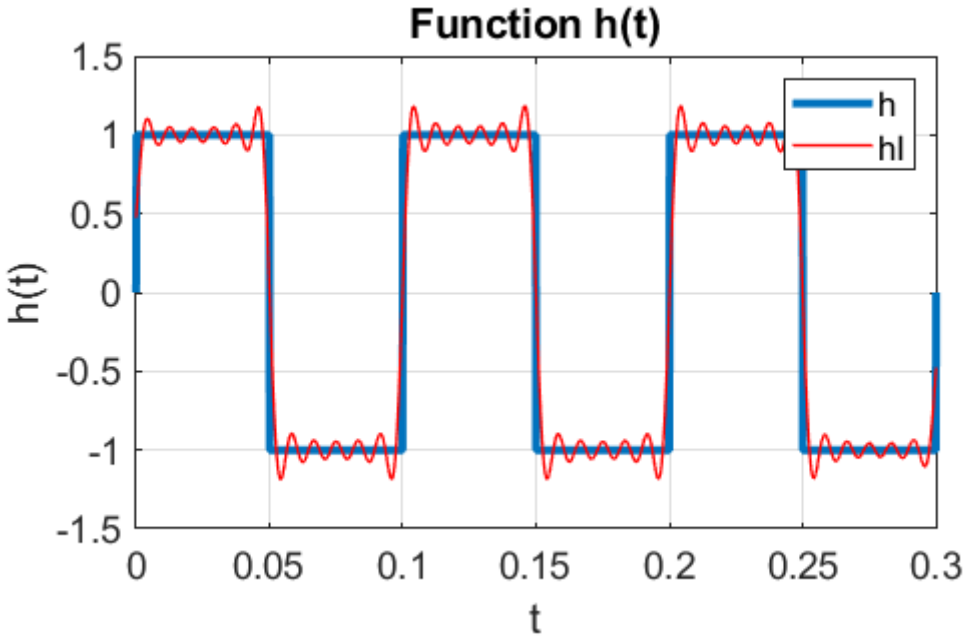


Fig. 5.1. The square wave signal $\left(f_0 = 10 \text{ Hz}\right)$ and its inverse Fourier transform.

The series exhibits a non-uniformity of convergence near a discontinuity. Note, the overshoot, which is called the **Gibb's phenomena**. The greater the number of terms included in the

series a better the approximation, however, the overshoot remains finite.

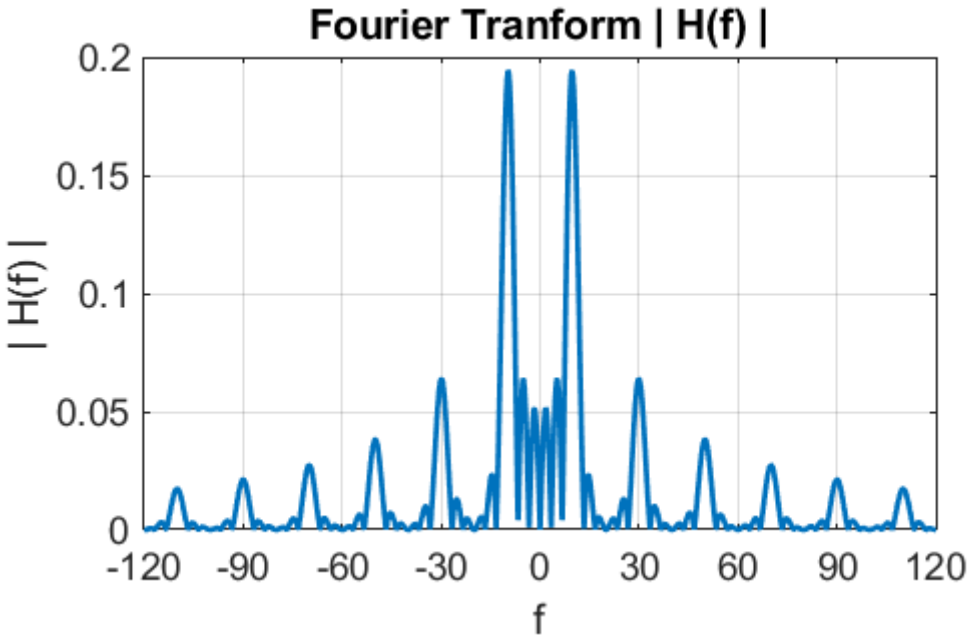


Fig. 5.2. The Fourier transform of the square wave signal.

The location and amplitude of the peaks can be estimated using the ginput function in the Command Window

```
>> xx = ginput
    xx =
          9.7581    0.1940
         29.9194    0.0638
         50.0806    0.0379
         69.9194    0.0268
         90.2419    0.0212
        110.0806    0.0176
```

The Fourier transform routine correctly calculated the peak frequencies of 10, 30, 50, 70, 90 and 110 Hz.

The relative heights of the peaks can be computed in the Command Window

>> xx(:,2)./xx(1,2)

1.0000	0.3287	0.1956	0.1384	0.1092	0.0905

The theoretical predictions for the relative amplitudes are

1/1	1/3	1/5	1/7	1/9	1/11
1.0000	0.3333	0.2.000	0.1429	0.1111	0.0909

Again, good agreement between the theoretical values and the computed values.

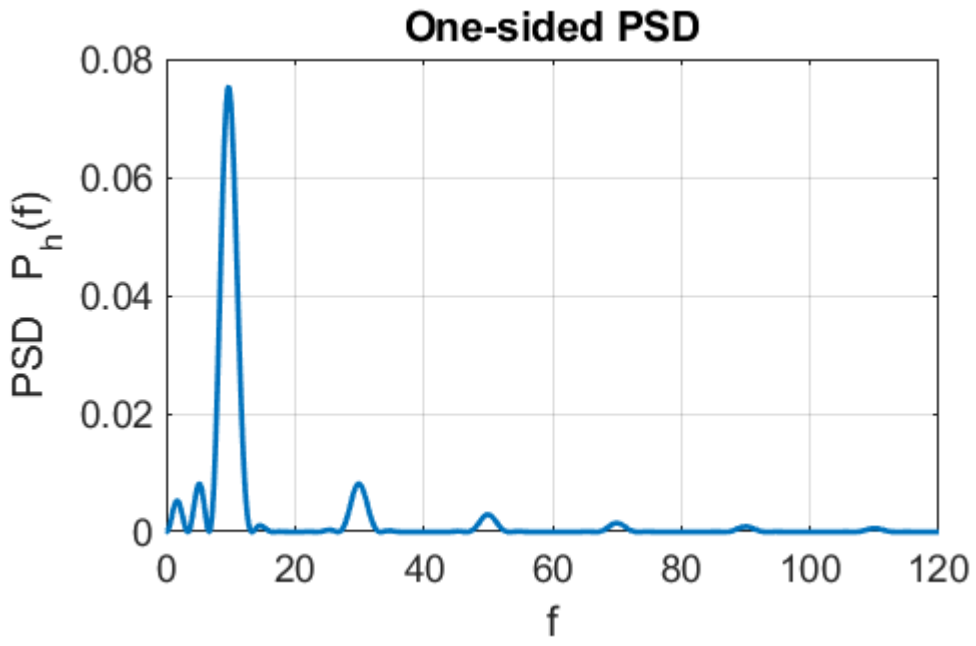


Fig. 5.3. The PSD function.

6. SAWTOOTH FUNCTION

A sawtooth function can be approximated by a Fourier series of the form

$$h(t) = A + B\sum_{n=1}^{\infty}(-1)^n \frac{\sin\left(2\pi n f_0 t\right)}{n}$$

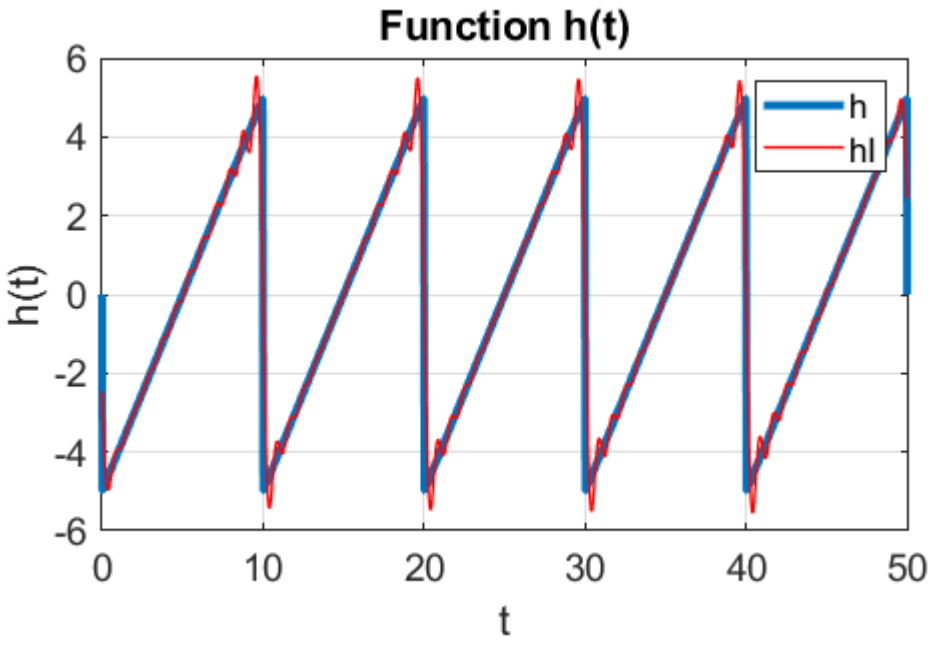


Fig. 6.1. The sawtooth function and the inverse Fourier transfer. The DC component of the function has been removed. The fundamental frequency is $\left(f_0 = 0.10\ \mathrm{Hz}\right)$.

As for the square wave, there are overshoots at the discontinuities.

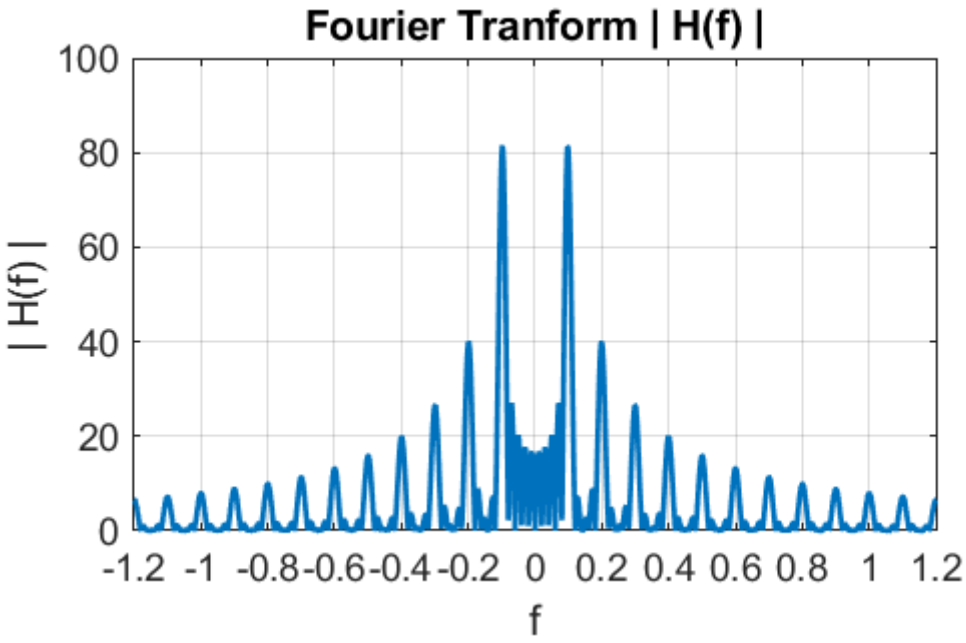


Fig. 6.2. The Fourier transform of the sawtooth function.

The location and amplitude of the peaks can be estimated using the ginput function in the Command Window

```
>> xx = ginput

xx =
   0.0992   81.1835
   0.1992   40.1076
   0.3008   26.6535
   0.3992   20.0283
   0.5008   15.9513
   0.6008   13.3012
   0.7008   11.4666
   0.8008    9.8358
   0.9008    8.9185
   0.9992    8.1031
   1.1008    7.1857
```

The Fourier transform routine correctly calculated the peak frequencies of 0.10 0.20 0.30 0.40 0.50 0.60 0.70 0.80 0.90 1.00 and 1.10 Hz.

The relative heights of the peaks can be computed in the Command Window (xx(:,2)/xx(1,2)) and again there is good agreement between the theoretical values and the computed values.

n	1/n	Theory	Computed
1	1/1	1.000	1.000
2	1/2	0.500	0.494
3	1/3	0.333	0.328
4	1/4	0.250	0.247
5	1/5	0.200	0.197
6	1/6	0.167	0.164
7	1/6	0.143	0.141
8	1/8	0.125	0.121
9	1/9	0.111	0.110
10	1/10	0.100	0.100
11	1/11	0.091	0.089

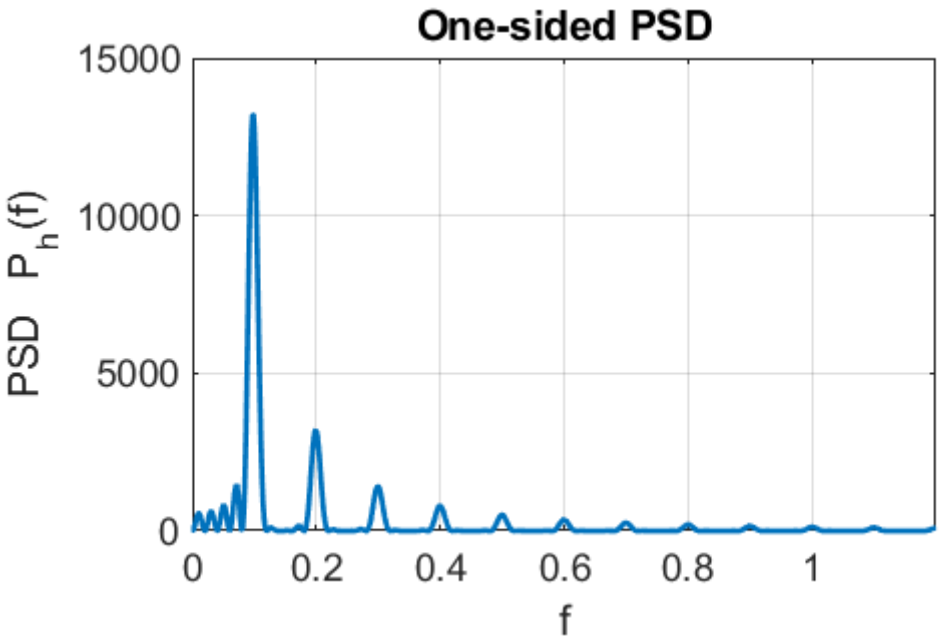


Fig. 6.3. The PSD function.

7. Single Square Pulse

The Fourier and inverse of a single square pulse can easily be computed.

Square pulse generated by the code

```
h = zeros(1,nT);
h(round(nT/3) : round(2*nT/3)) = 1;
% h = h - 0.5;
```

The DC component of the square pulse can be removed by un-commenting the statement % h = h – 0.5.

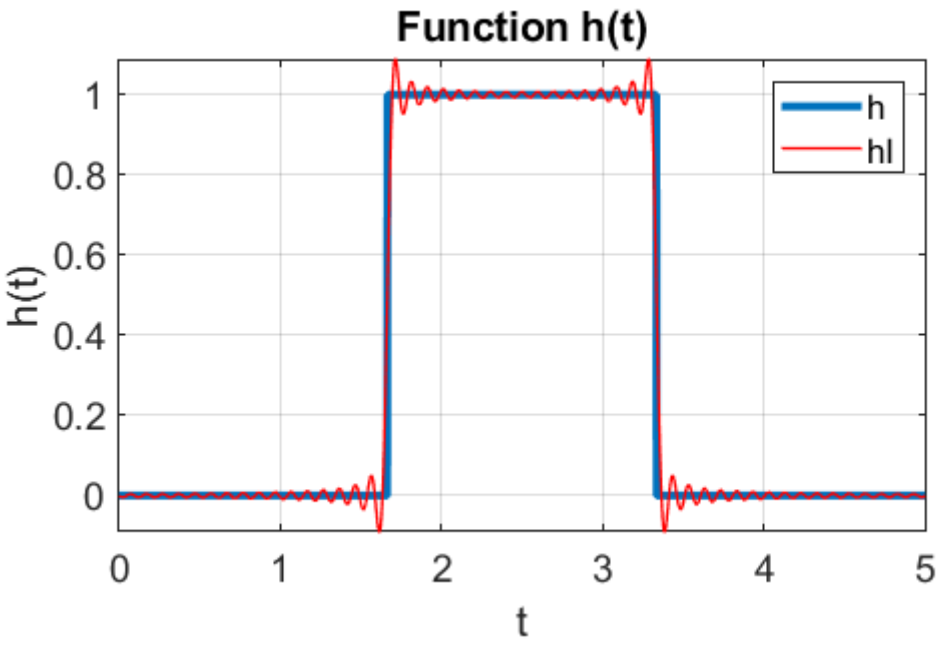


Fig. 7.1. The square pulse and its inverse Fourier transform.

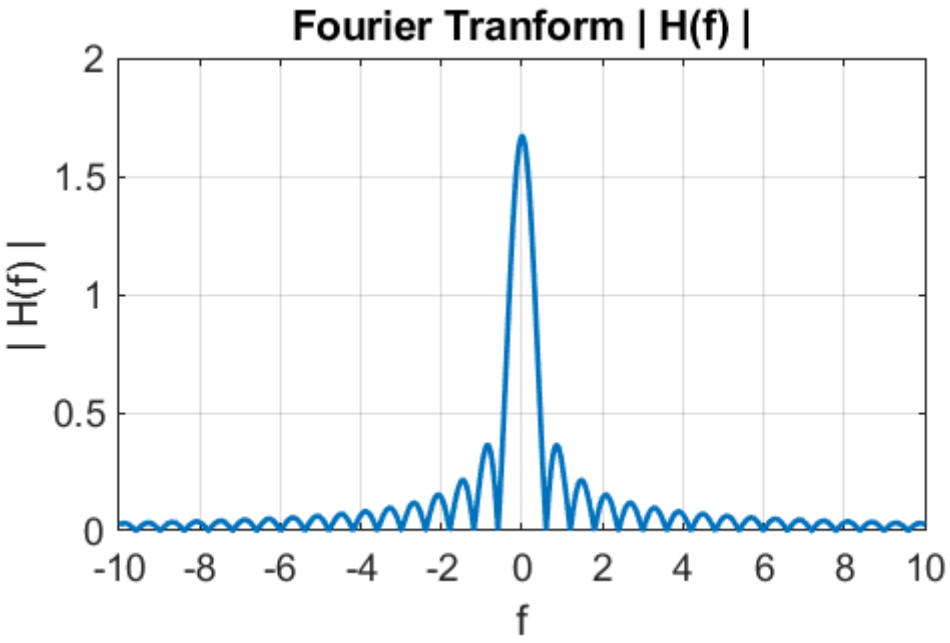


Fig. 7.2. The Fourier transform of the square pulse with a large DC component.

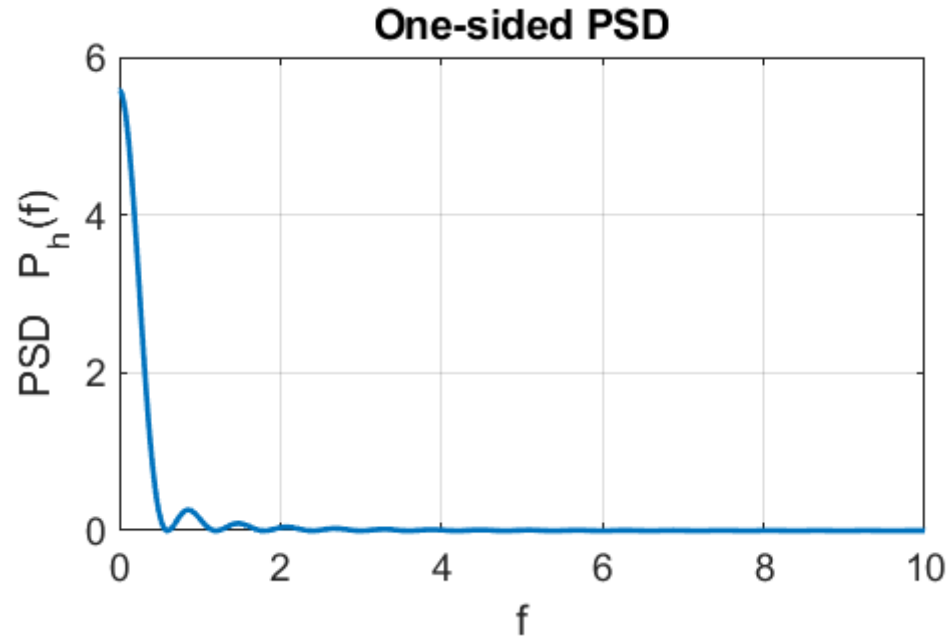


Fig. 7.3. The PSD function for the square pulse.

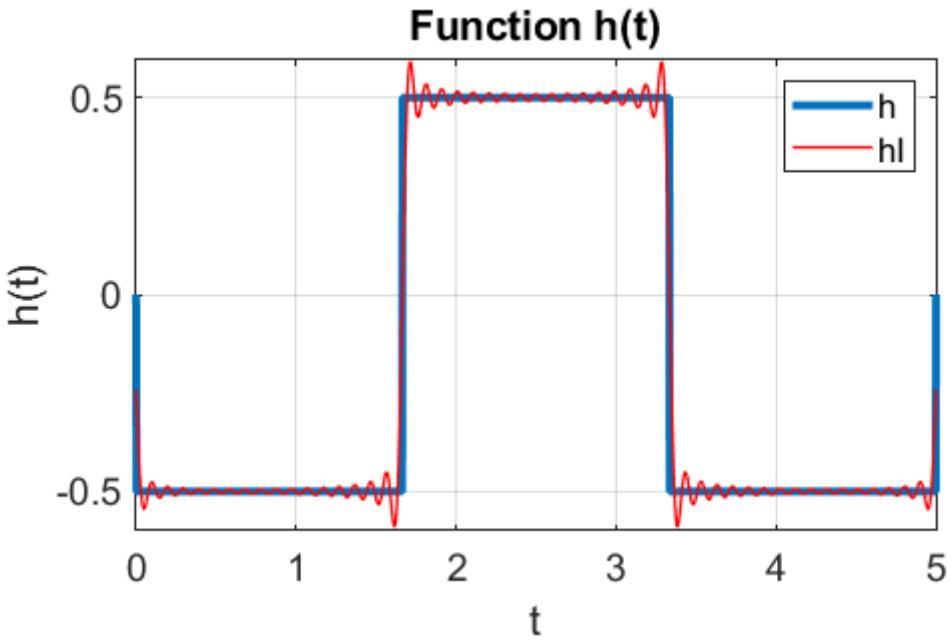


Fig. 7.4. The square pulse and its inverse Fourier transform with the DC component removed.

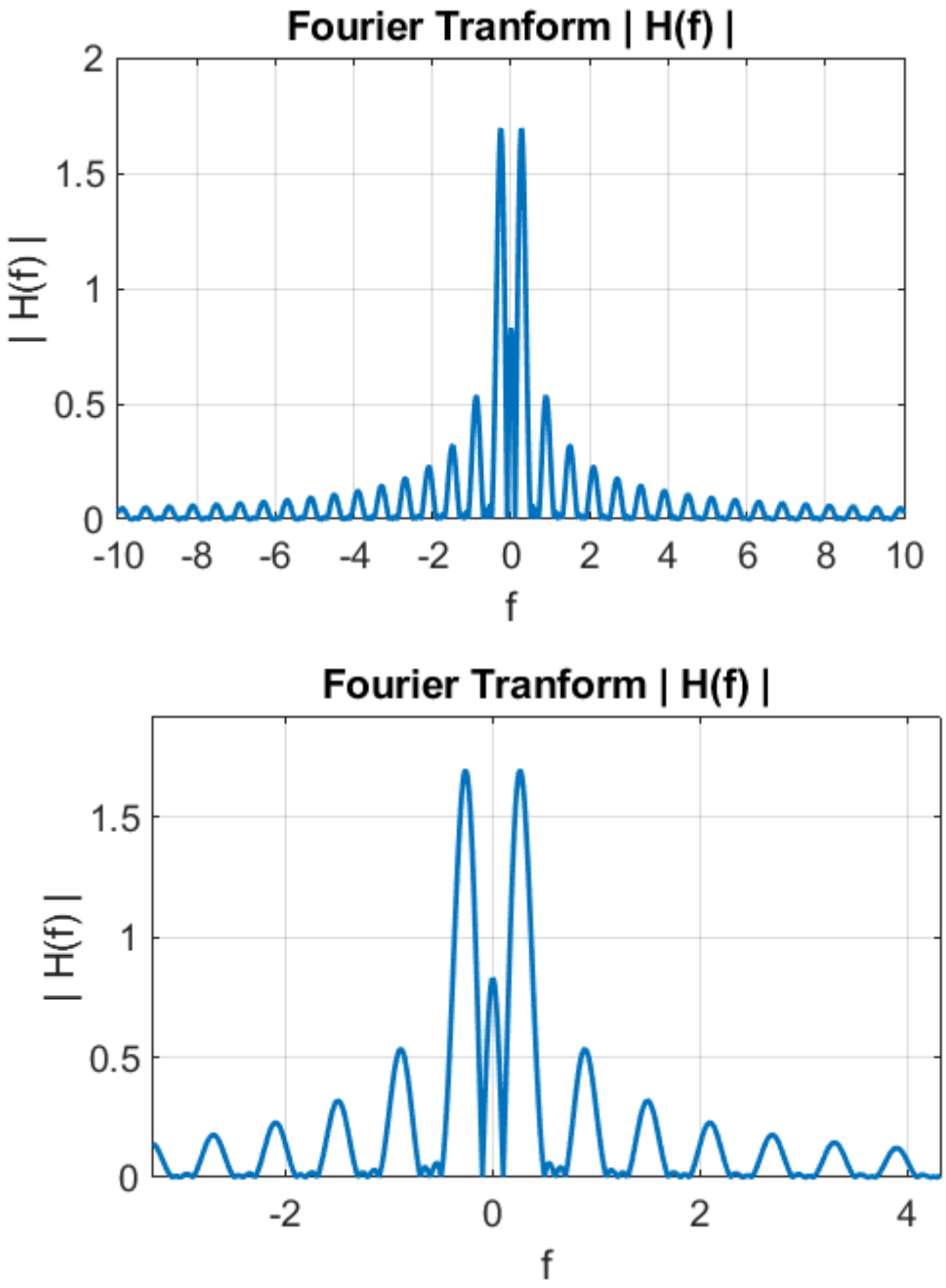


Fig. 7.5. The absolute value of the Fourier transform of the square pulse with the DC component removed. The spacing between the peaks for $f > 0$ or $f < 0$ is constant $(\Delta f = 0.60\ \mathrm{Hz})$.

We can explore the concept that a narrow pulse has a much broader frequency spectrum.

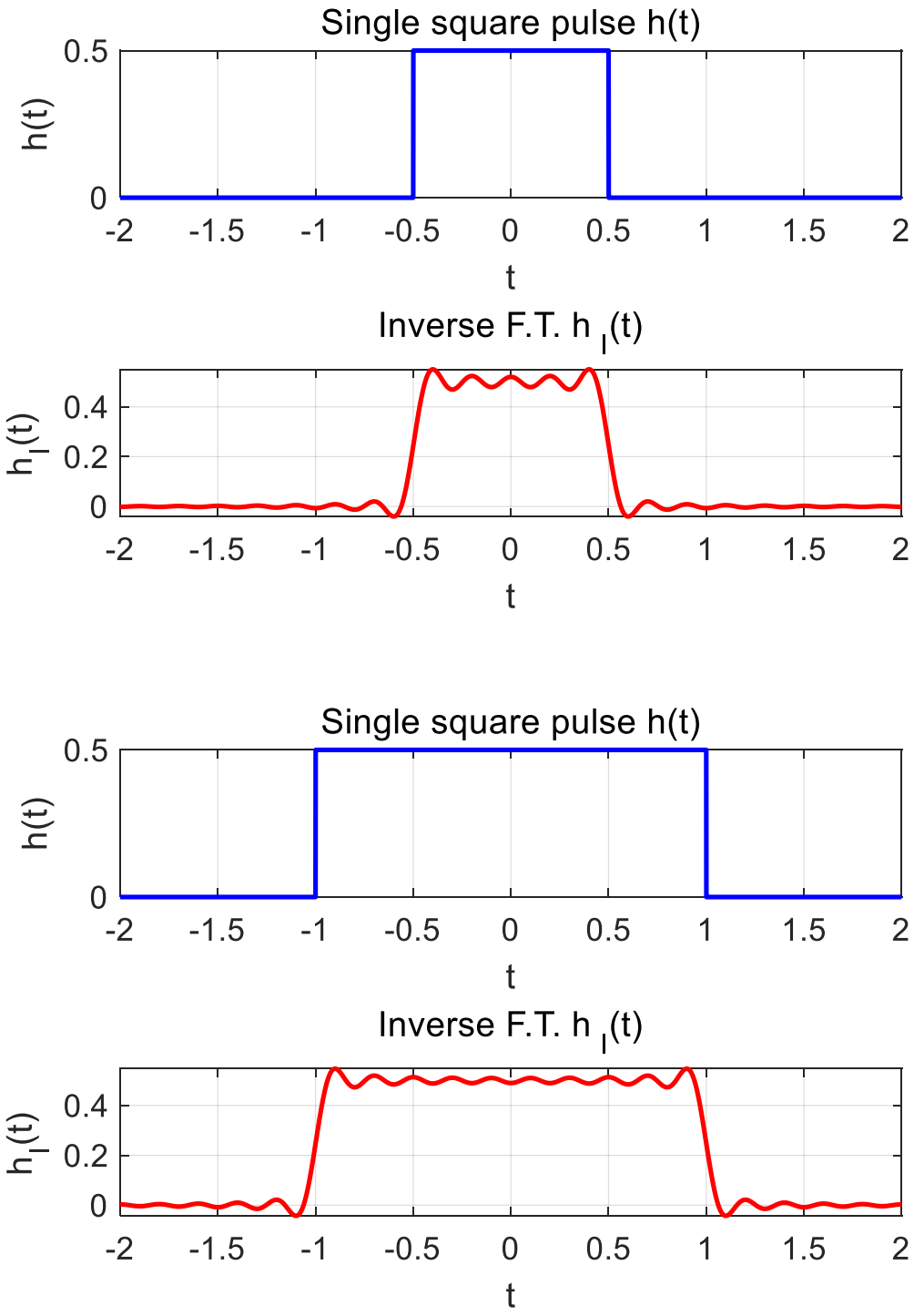


Fig. 7.6 Square wave pulses.

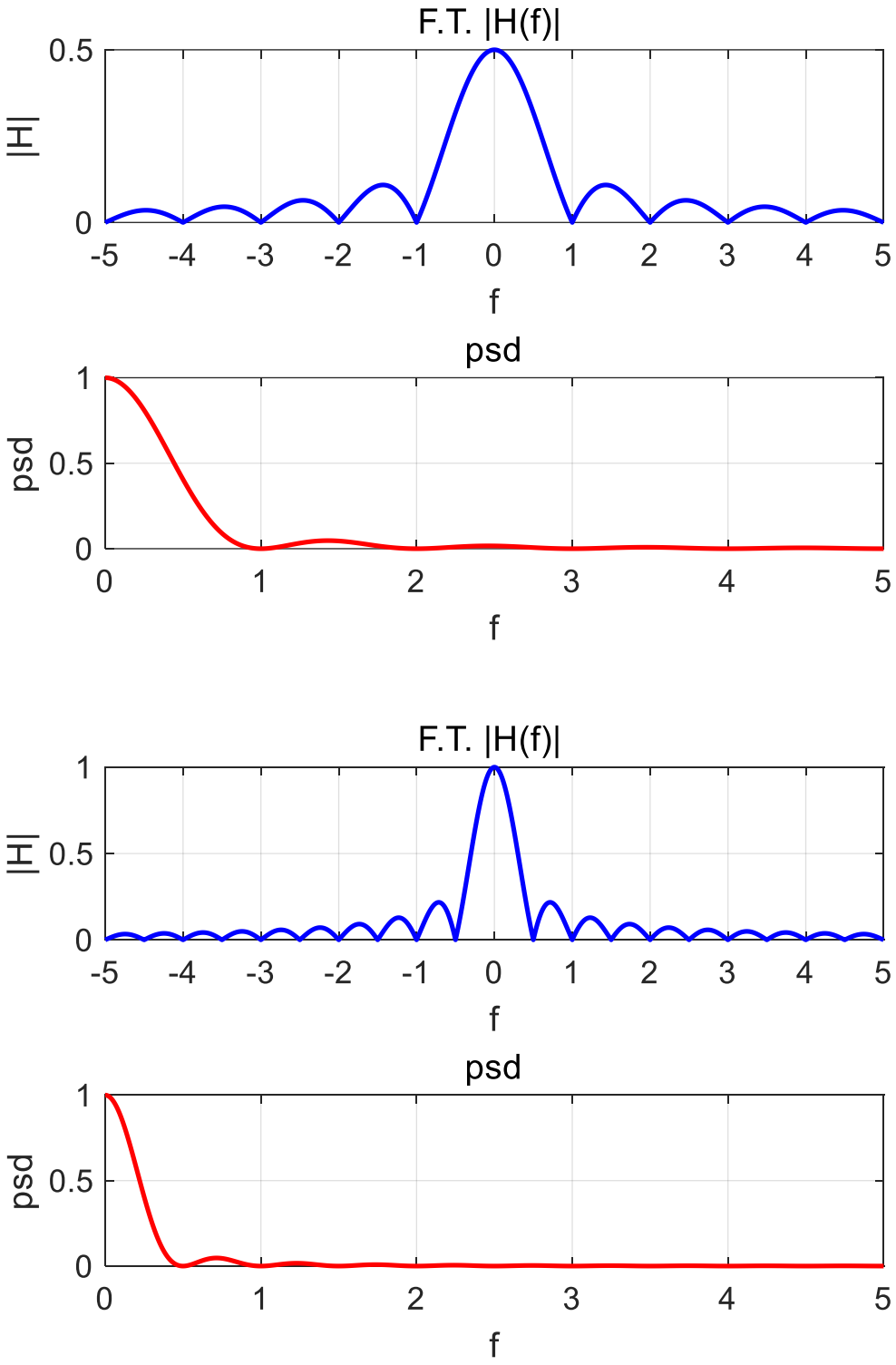


Fig. 7.7. The narrow pulse (top graph) does have a broader spectrum where higher frequency components make a contribution to the Fourier Transform.

8. Damped sine wave

The function for a damped sine wave can be expressed as

$$h(t) = A\exp(-t/\tau)\,\sin(2\pi\, f_0\, t)$$

This function might represent the displacement of a damped harmonic oscillator or the electric field in a radiated wave, or the current in an antenna.

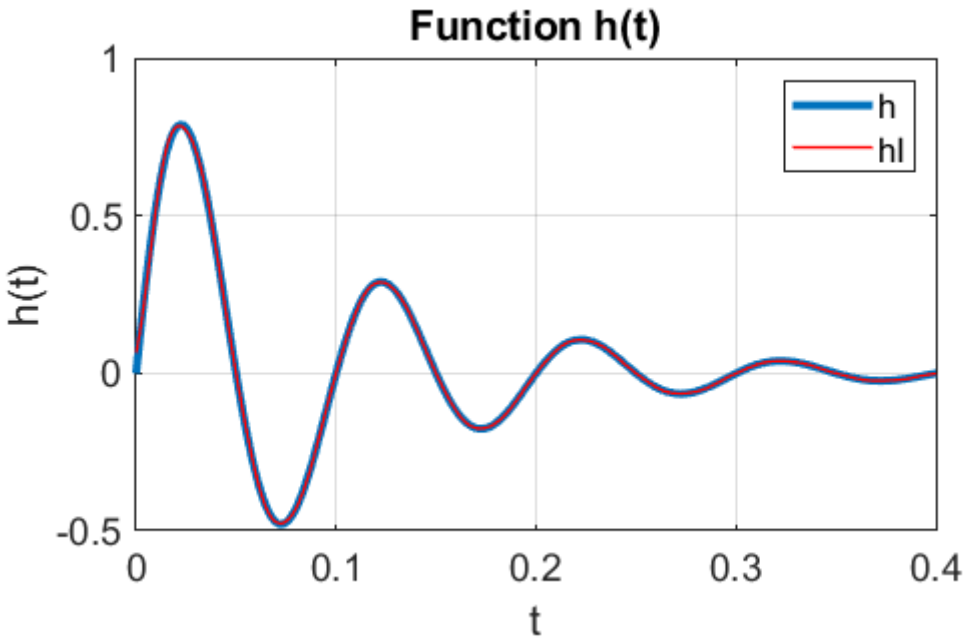


Fig. 8.1. A damped sine wave with $A = 1$ $f_0 = 10$ Hz $\tau = 0.1$ s

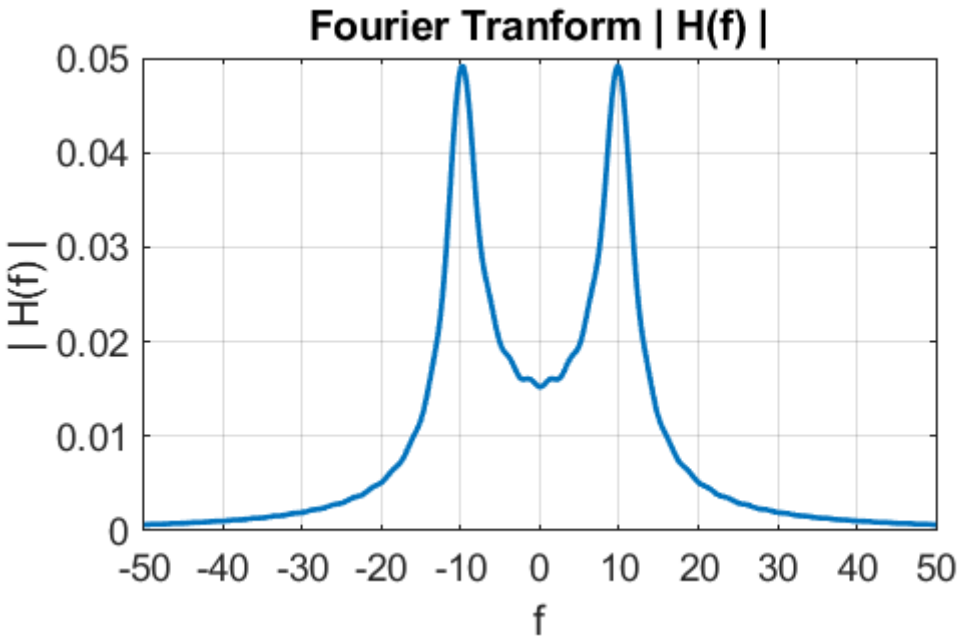


Fig. 8.2. The Fourier transform of the damped sine.

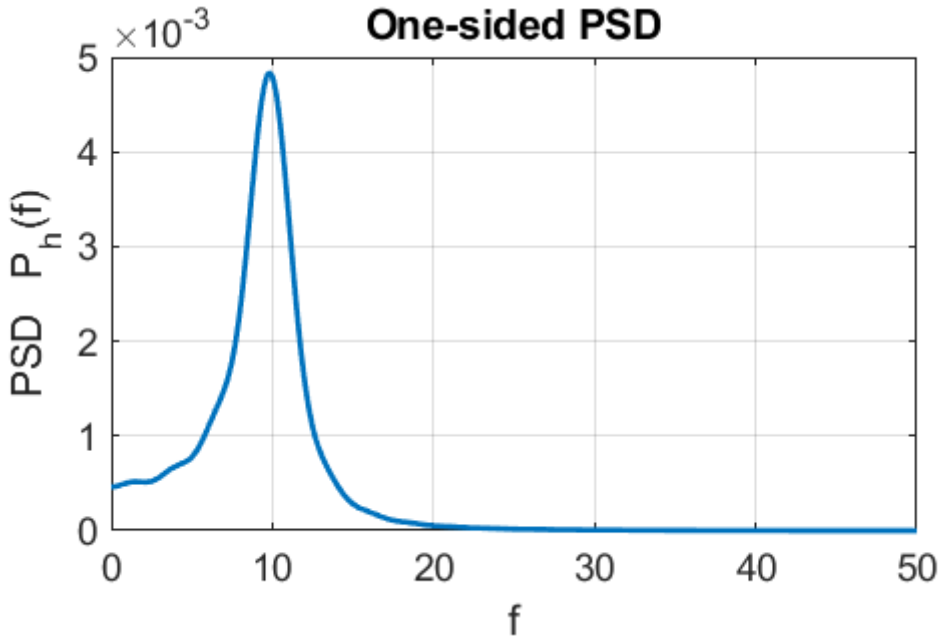


Fig. 8.3. Energy spectrum for the damped sine wave.

The width of the PSD function at half maximum power is approximately equal to $\Delta f \approx \frac{1}{\pi \tau} = 3.18 \text{ Hz}$. The width at half maximum power using the data Cursor of figure 31 is 3.40 Hz.

9. ECG recording

The raw data for an ECG recoding is stored in the data file **ecg.mat**. An estimate of the time scale is used and the DC component of the signal is removed.

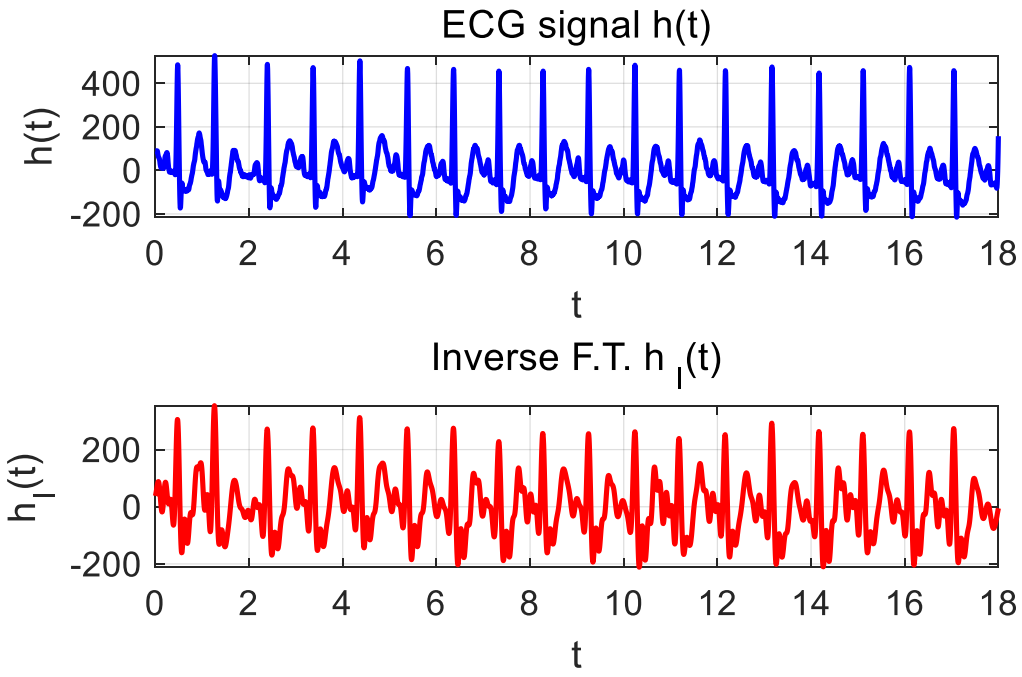


Fig. 9.1. The ECG recording.

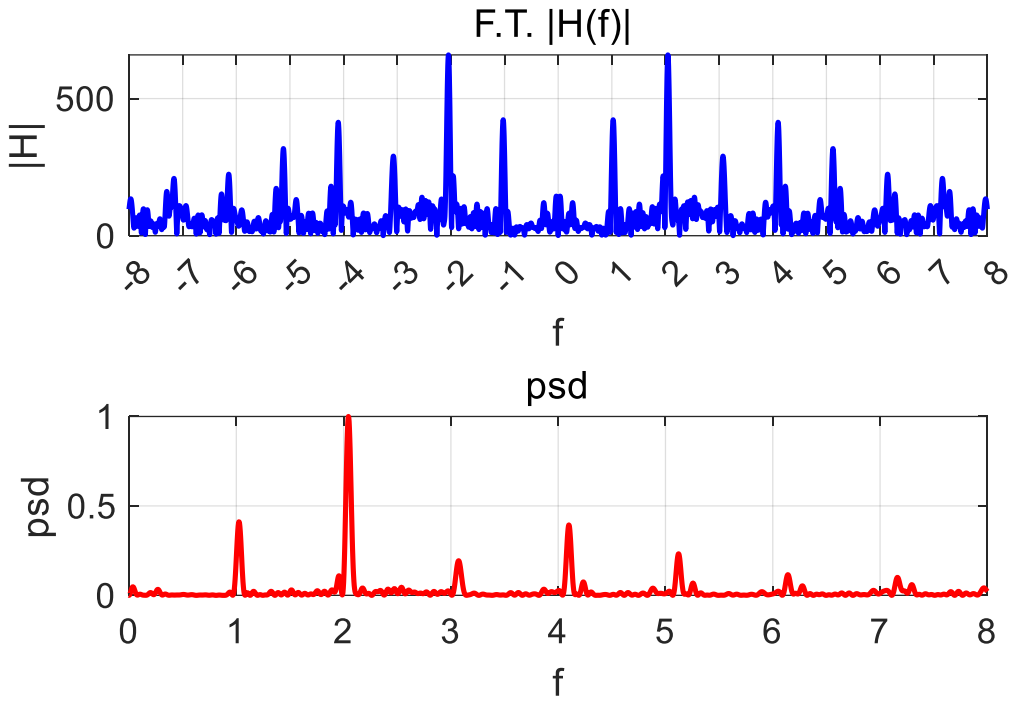


Fig. 9.2 The Fourier transform of the ECG recording and PSD function for the ECG recording.

10 BEATS (sound files)

A beat signal is computed by the addition of two sinusoidal functions

```
A1 = 1; A2 = 1;
f1 = 1000; f2 = 1008;
phi1 = 0; phi2 = 0;
h = A1.*sin(2*pi*f1*t + phi1) + A2.* sin(2*pi*f2*t+ phi2);
```

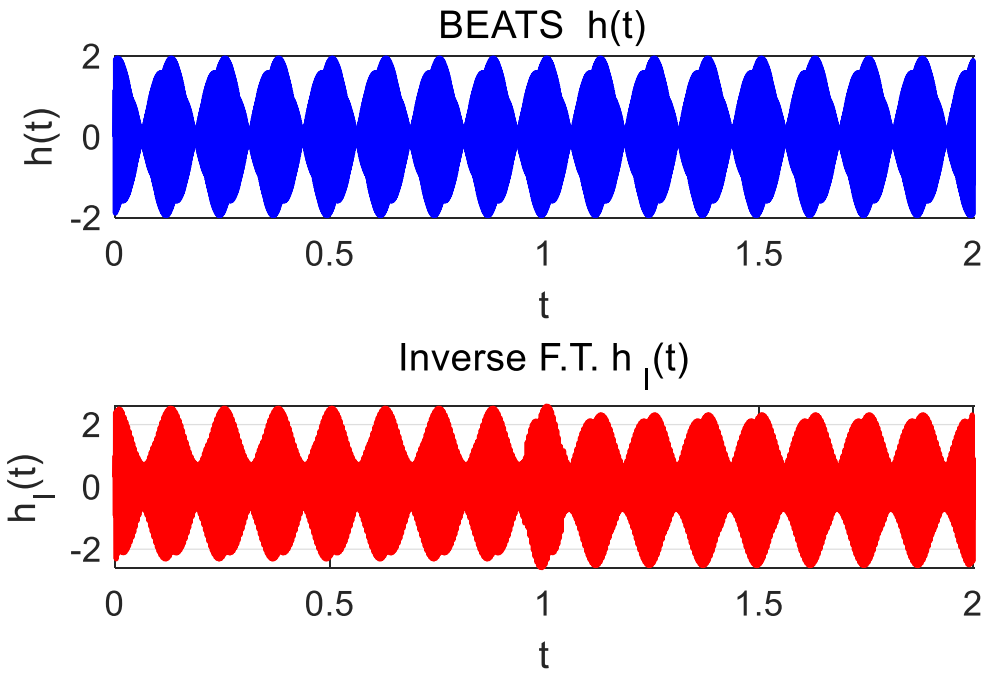


Fig. 10.1. Beat pattern of two pure tones 1000 Hz and 1008 Hz.

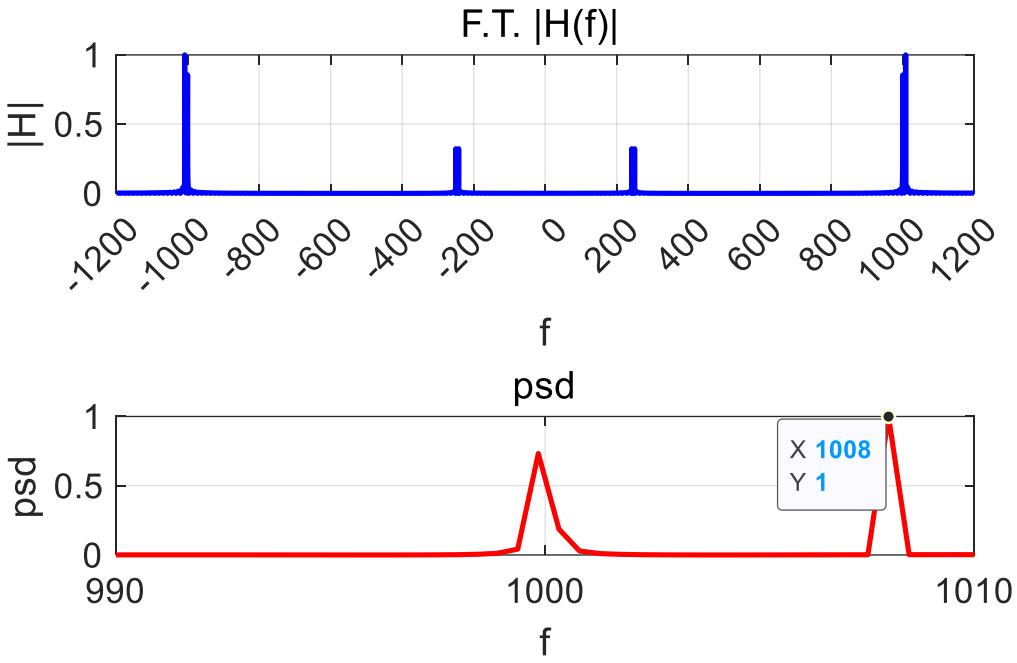


Fig. 10.2. The Fourier transform of the beat signal. It took about 43 seconds to calculate the Fourier transform. Zoom view of the PSD function showing the two peaks at frequencies 1000 Hz and 1008 Hz.

11. BEATS (audio file)

The recording of a beat pattern produced by two pure tones of 1000 Hz and 1008 Hz is loaded from the wav file **wav_S1000_1008.wav**.

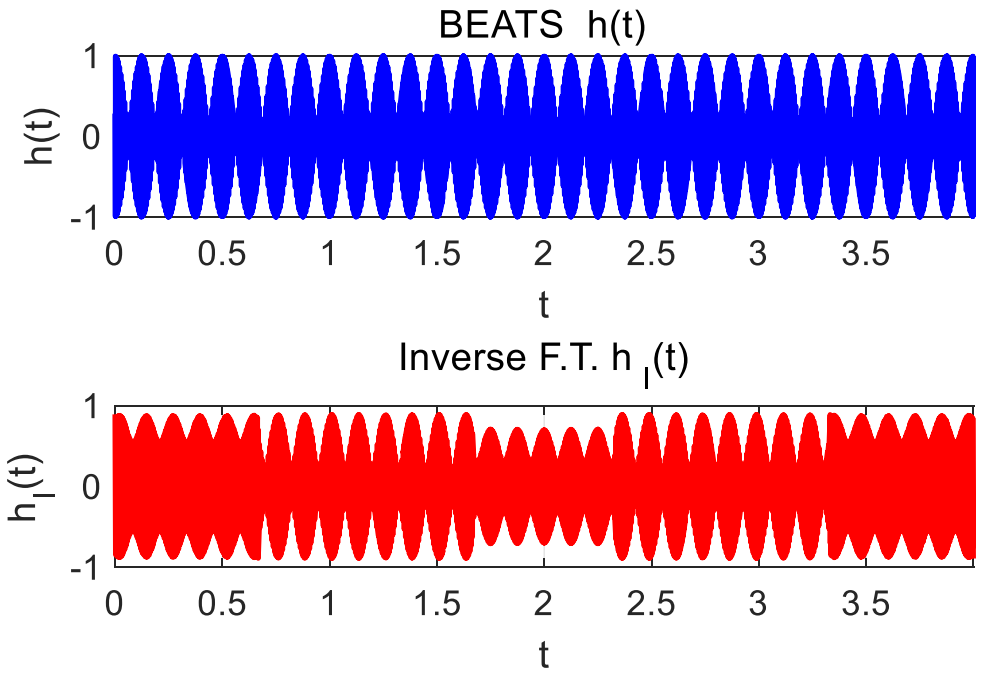


Fig. 11.1. Beat pattern.

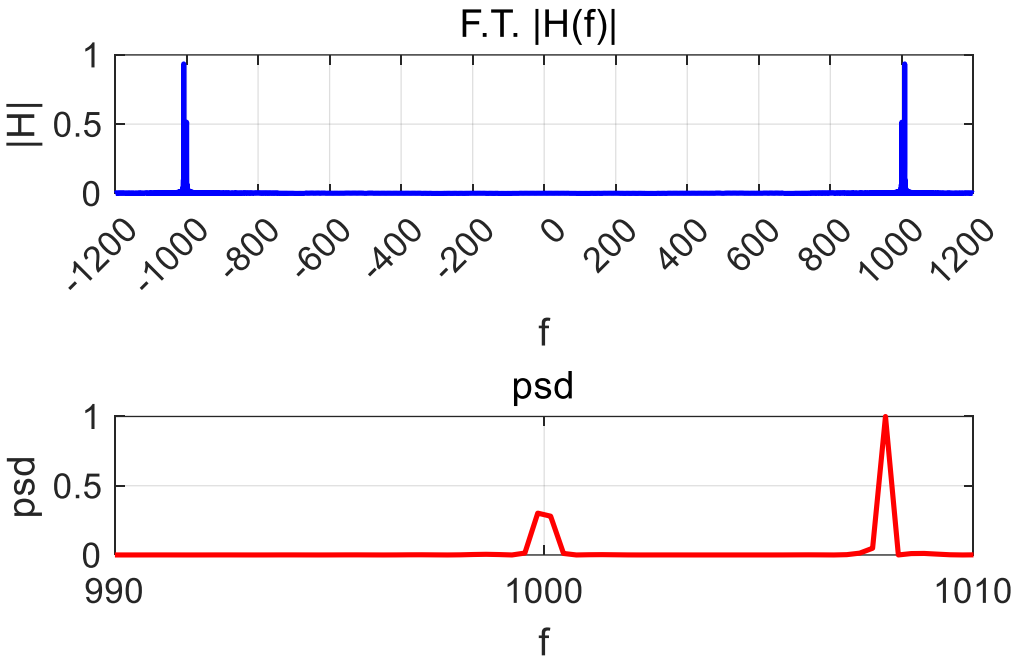


Fig. 11.2. The Fourier transform of the beat signal. It took about 43 seconds to calculate the Fourier transform. Zoom view of the PSD function showing the two peaks at frequencies 1000 Hz and 1008 Hz.

12. Sound recording at 440 Hz

```
[signal, Fs] = audioread('audio440.wav');
```

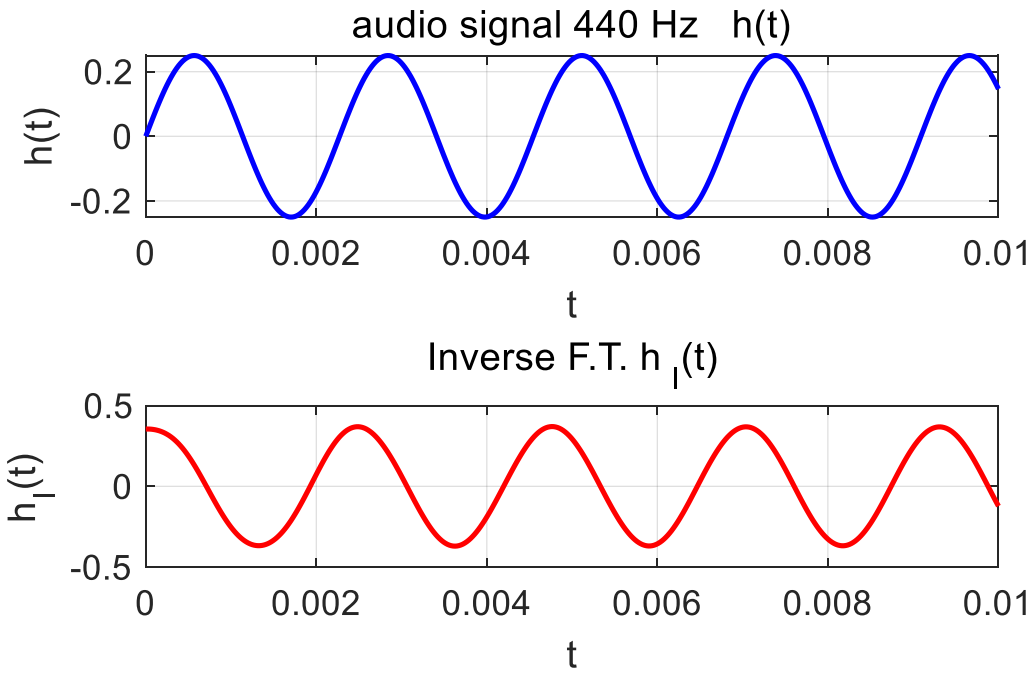


Fig. 12.1. 440 Hz Signal

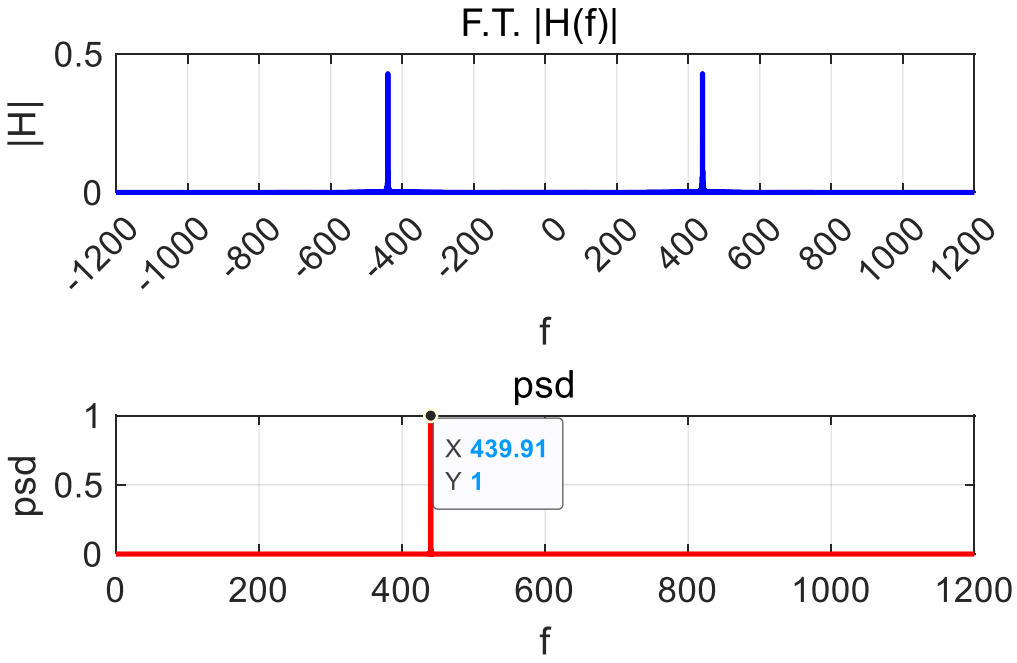
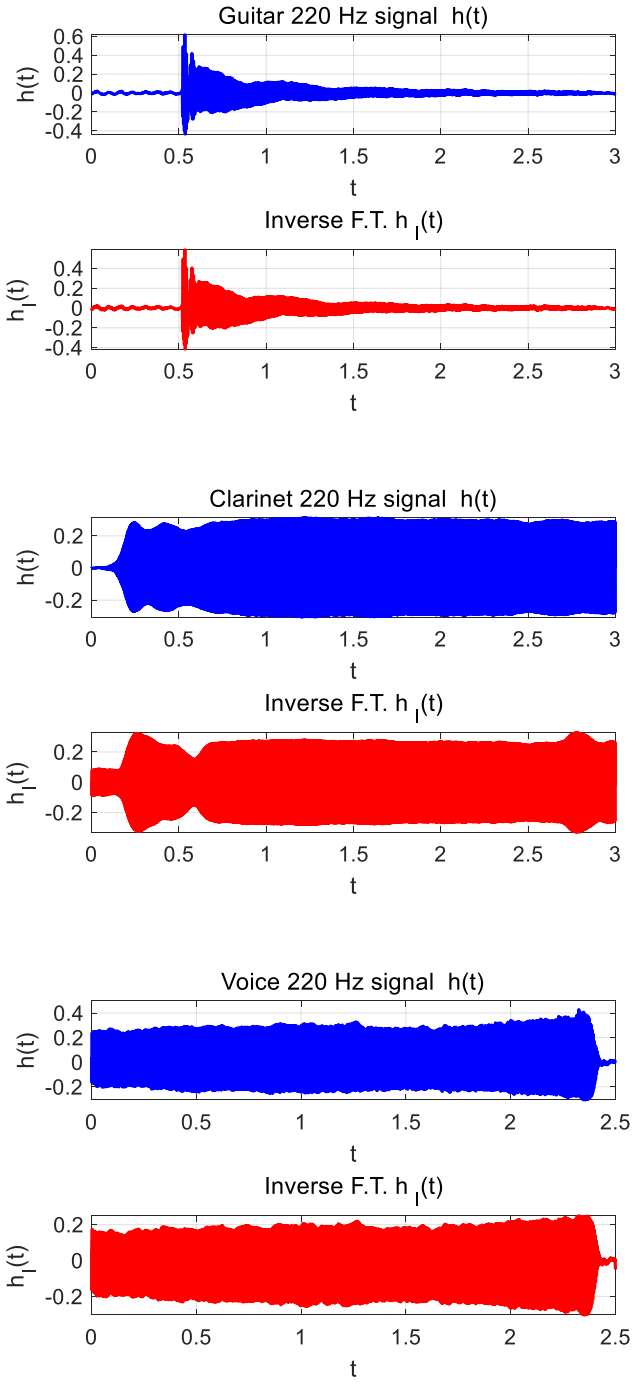


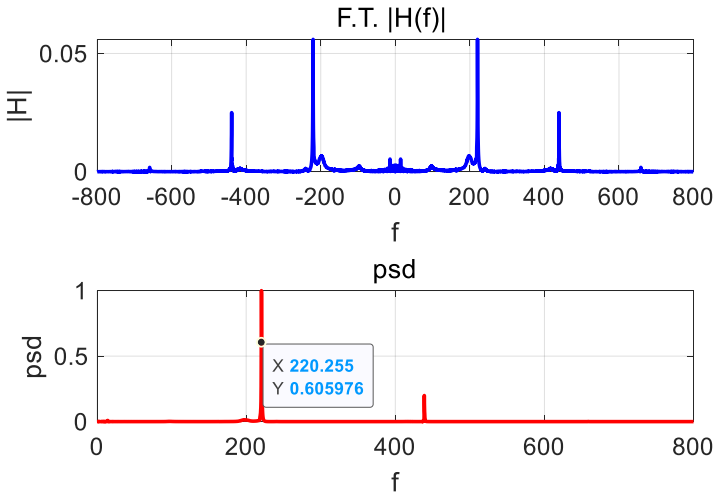
Fig. 12.2. The Fourier transform accurately predicts the signal frequency at 440 Hz.

13. 220 Hz Guitar / 14. 220 Hz Clarient / 15. 220 Hz Voice

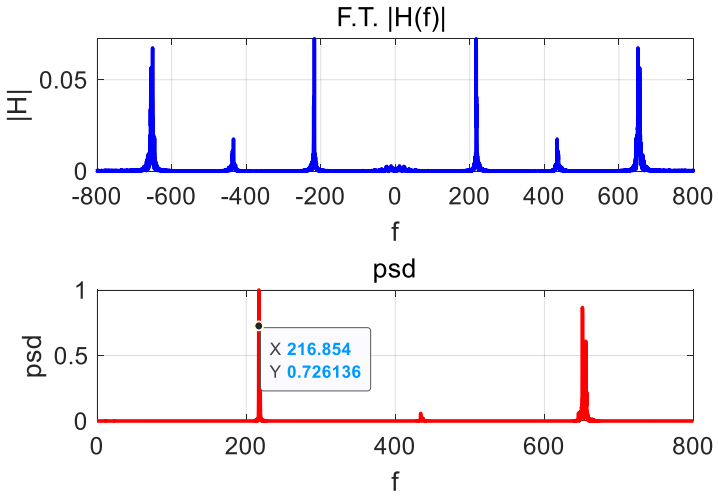
We can compare three recording of a note at 220 Hz produced by a guitar, clarinet and by a voice.



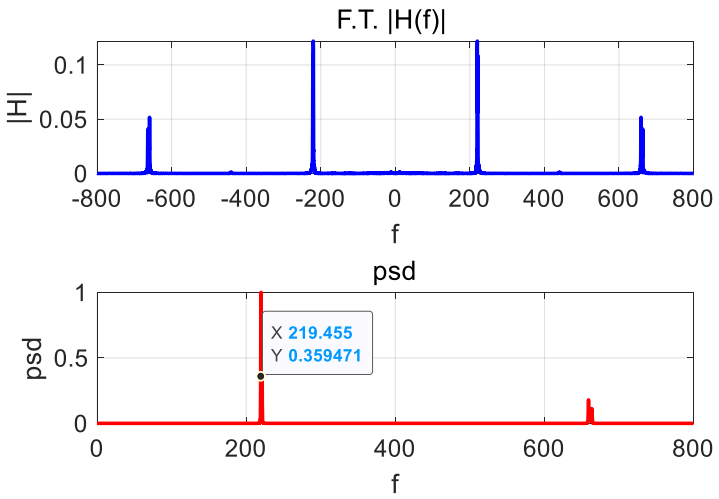
Guitar



Clarinet



Voice



The predominant frequencies for the guitar are its fundamental (220 Hz) and the 1st harmonic (440 Hz). The clarinet fundamental is at 220 Hz, the 1st harmonics at 440 Hz and a strong 3rd harmonic at 660 Hz. The voice has the fundamental at 220 Hz and the 3rd harmonic at 660 Hz and the 2nd harmonic is absent.

16. Train Whistle

The recording of a train whistle is loaded from the wav file

Train.wav.

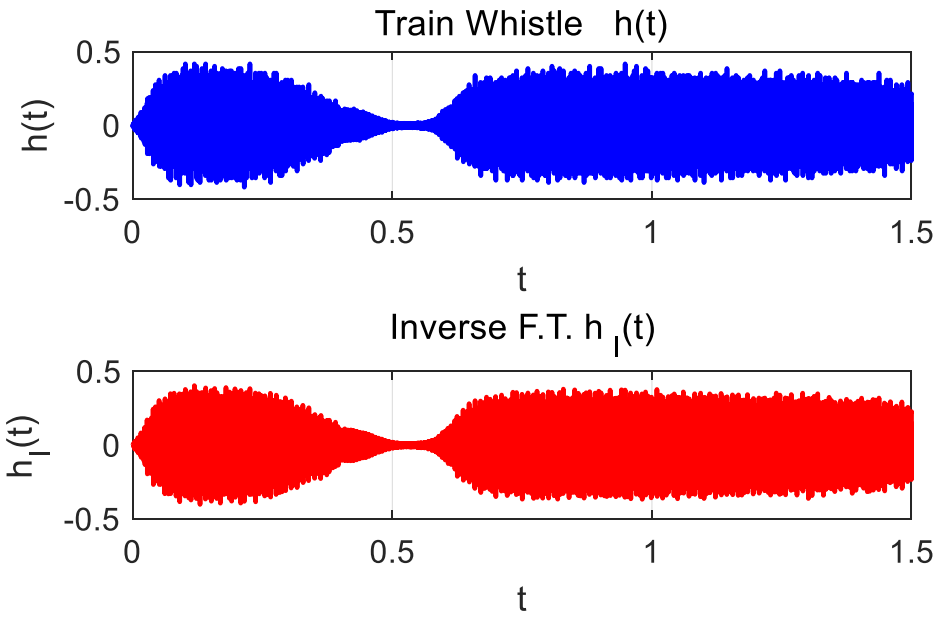


Fig. 16.1. The sound of a train whistle.

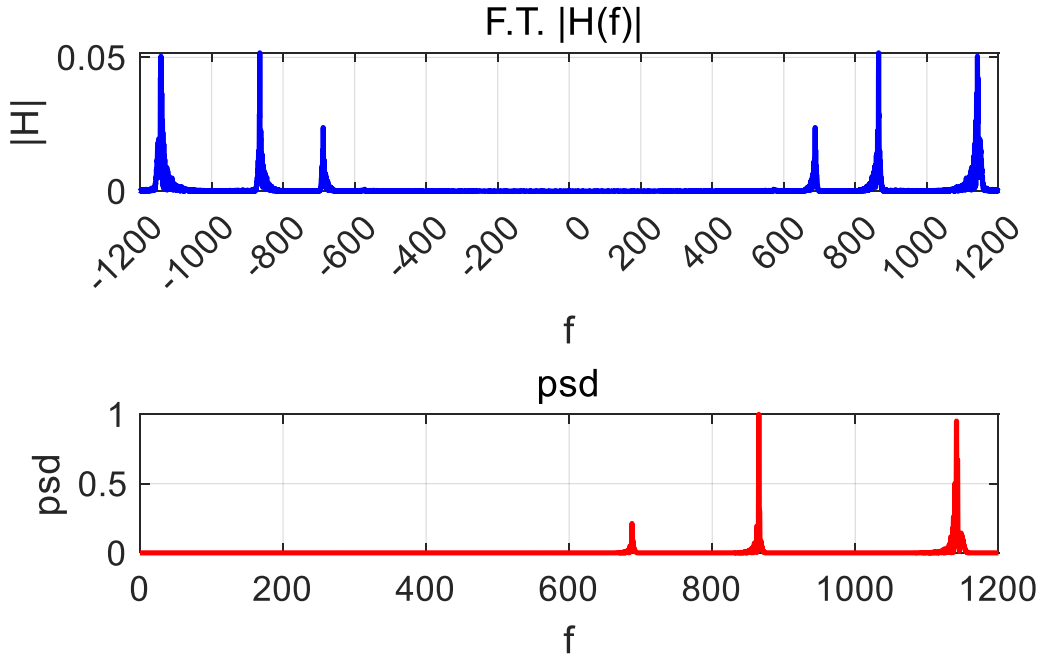


Fig. 16.2. The Fourier transform of a train whistle.

17. Digital Filtering

We can digitally filter a signal by setting the Fourier transfer function to zero for the desired frequency range. For example, consider the superposition of two sine functions with frequencies 1000 Hz and 100 Hz.

```
h = sin(2*pi*1000*t) + sin(2*pi*100*t);
```

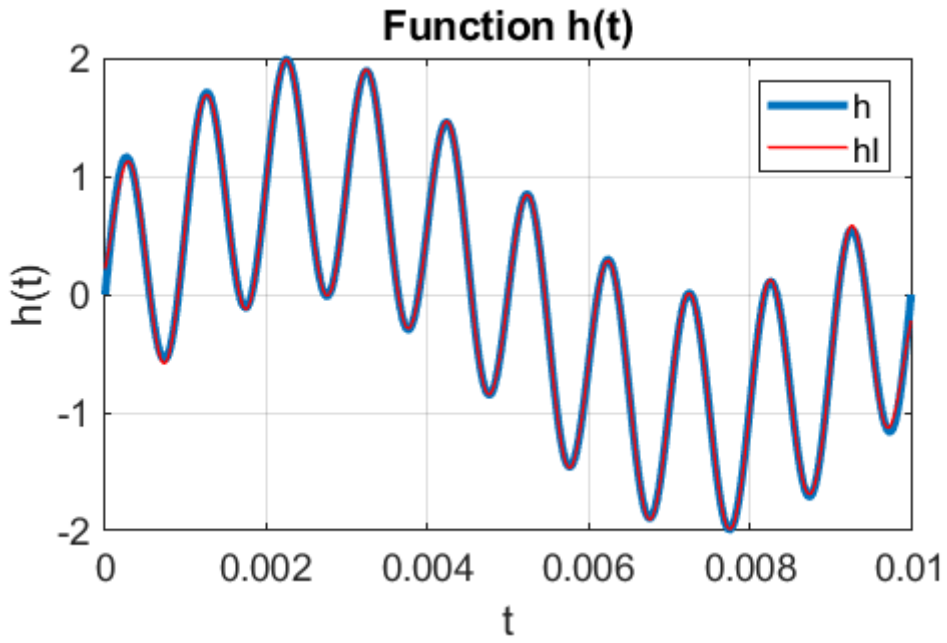


Fig. 17.1. Unfiltered signal with frequency components 1000 Hz and 100 Hz.

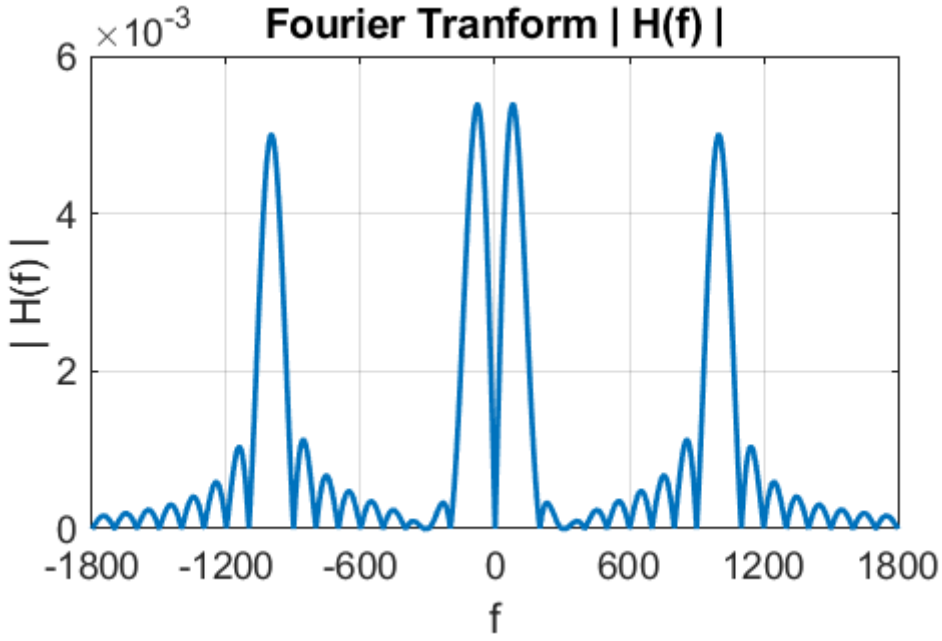


Fig. 17.2. Fourier transform of the unfiltered signal with frequency components 1000 Hz and 100 Hz.

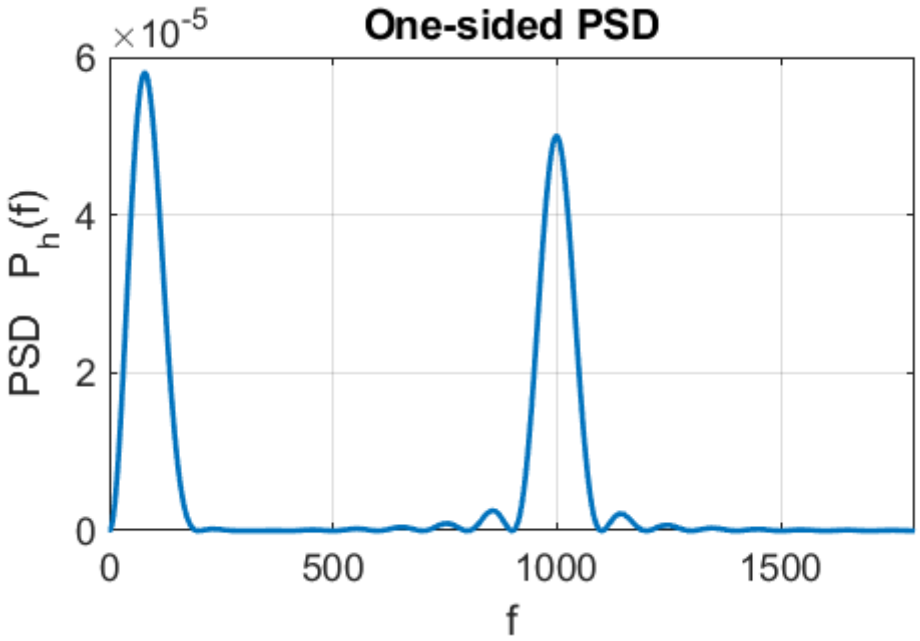


Fig. 17.3. PSD function for the unfiltered signal with frequency components 1000 Hz and 100 Hz.

The original signal can be low pass filtered by removing all frequency components with values less than 500 Hz by uncommenting the lines of code

```
% Filtering  (uncomment for filtering effects)
% H(f<500) = 0;

  H(f<500) = 0;
```

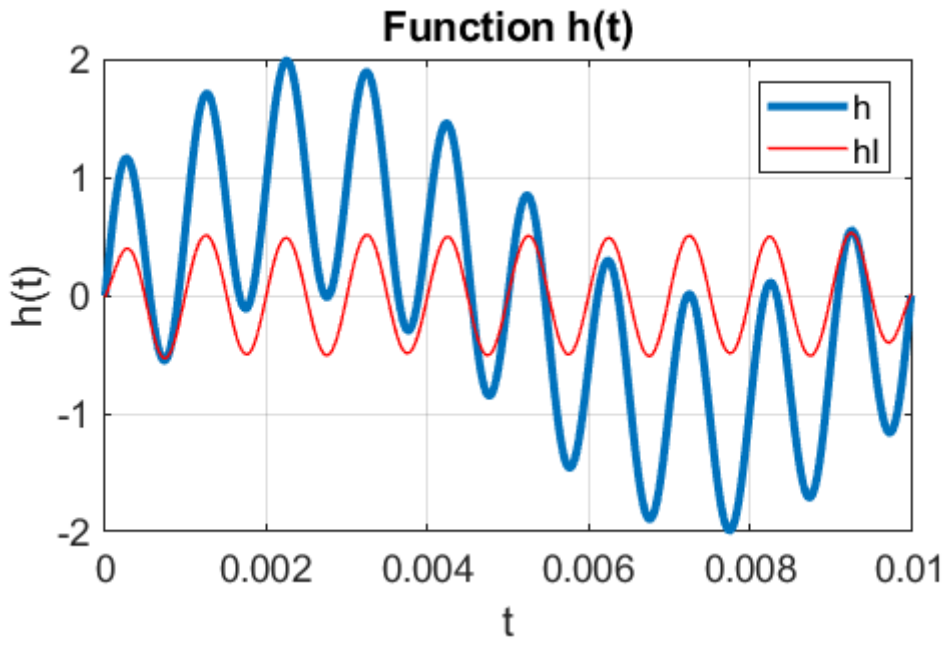


Fig. 17.4. The 100 Hz frequency component is removed and we are left with the pure 1000 Hz signal.

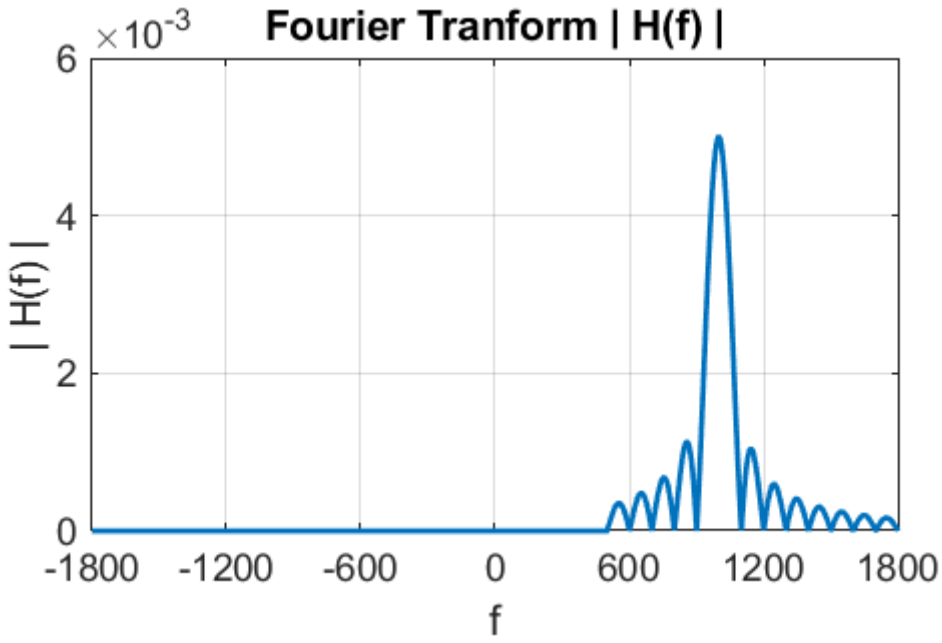


Fig.17.5. The filtered Fourier transform with the low frequency 100 Hz component removed.

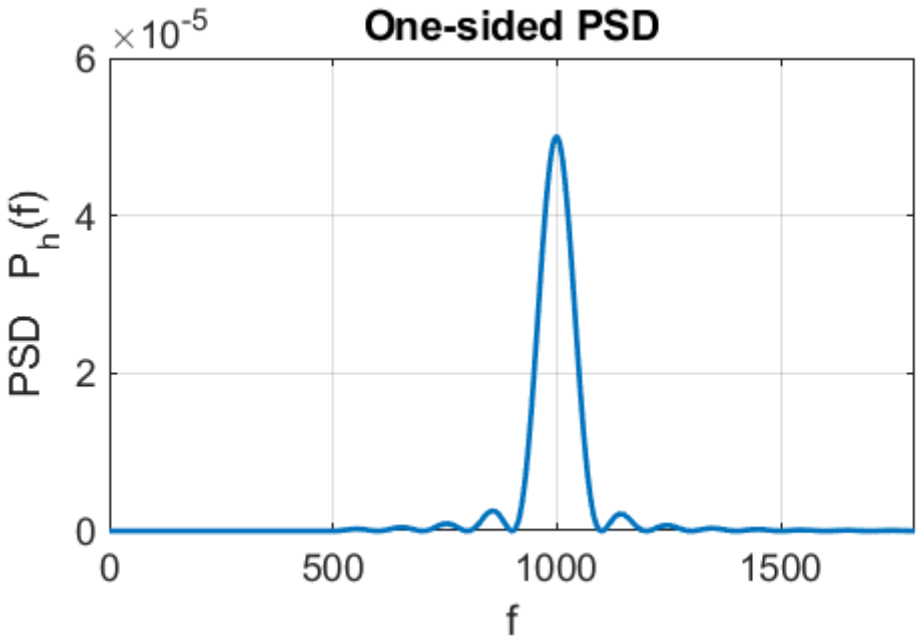


Fig. 17.6. The filtered PSD function with the low frequency 100 Hz component removed.