DOING PHYSICS WITH MATLAB

FOURIER ANALYSIS

FOURIER TRANSFORMS

Ian Cooper

matlabvisualphysics@gmail.com

DOWNLOAD DIRECTORY FOR MATLAB SCRIPTS

https://github.com/D-Arora/Doing-Physics-With-Matlab/tree/master/mpScripts

https://drive.google.com/drive/u/3/folders/1j09aAhfrVYpiMavajrgSvUMc89ksF9Jb

maths_ft_01.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.

simpson1d.m

mscript for computing the value of a definite integral using Simpson's rule. An odd number of grid points must be used to accurately calculate the integral.

wav_S1000_1008.wav

Data sound file to find the Fourier transform of a beat signal.

Train.wav

Data sound file for a train whistle.

ecg.mat

Raw data for the recording an ecg.

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**

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

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

$$(4) \quad PSD \quad P_h(f) = |H(f)|^2 + |H(-f)|^2$$

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

$$(5) \quad PSD \quad P_h(f) = 2|H(f)|^2 \quad 0 \le f < \infty$$

$$(6) \quad Total\ Power = 2\int_{0}^{\infty}|H(f)|^2\ df$$

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 $\qquad\qquad$ *tMin tMax nT*

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

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 $tMax$

$$t \leq tMin \quad h(t) = 0 \qquad t \geq tMax \quad h(t) = 0$$

The function $h(t)$ is specified by the variable $flagF$ in the switch/case statements.

The function **simpson1d.m** uses Simpson's rule to evaluate each integral.

```matlab
% 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:

```matlab
% One-sided power spectral density PSD  Ph(f)
   Ph = 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 $\quad h(t) = \exp\left(-\pi t^2\right)$
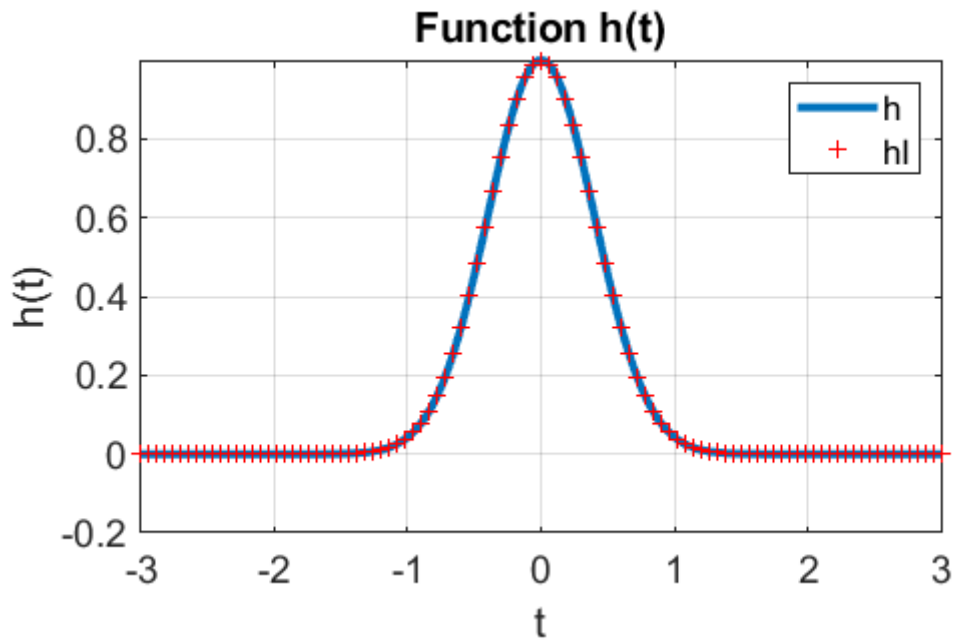
Function h(t)

Fig. 1.   A plot of a Gaussian function and its inverse Fourier transform.

Fourier Tranform | H(f) |
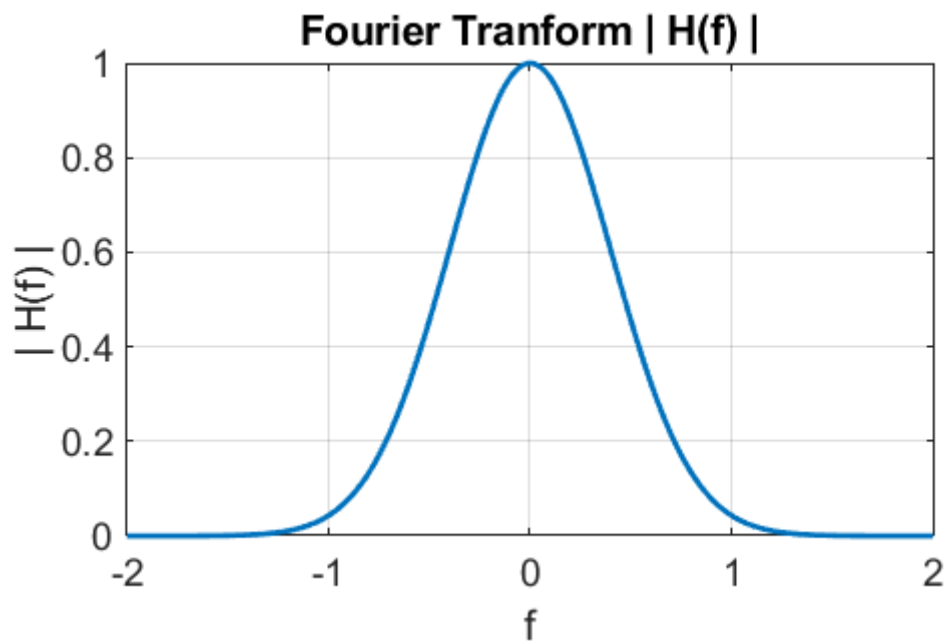
Fig. 2.   The Fourier transform $\left|H(f)\right|$ of the function $h(t)$.
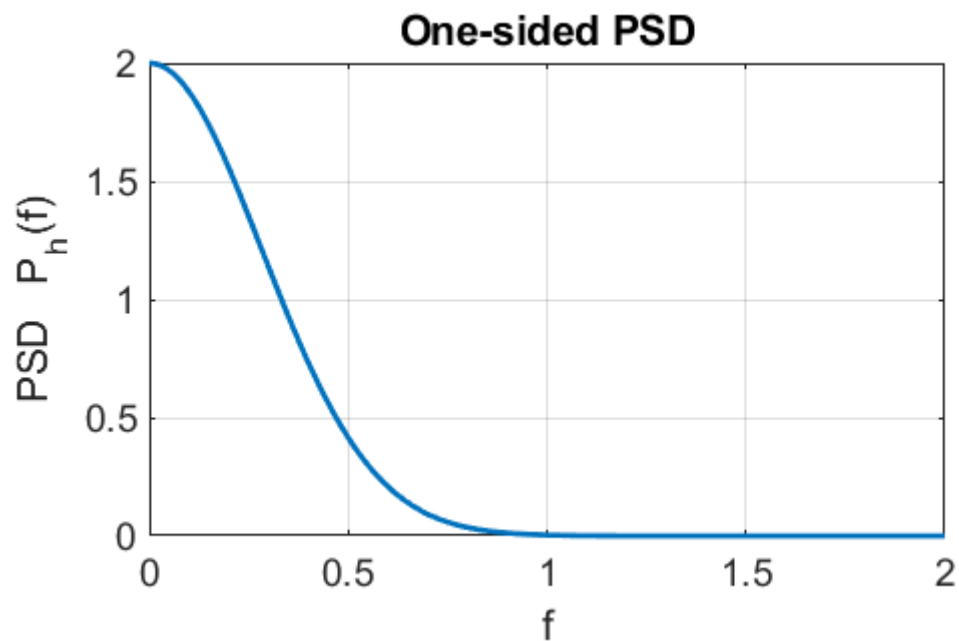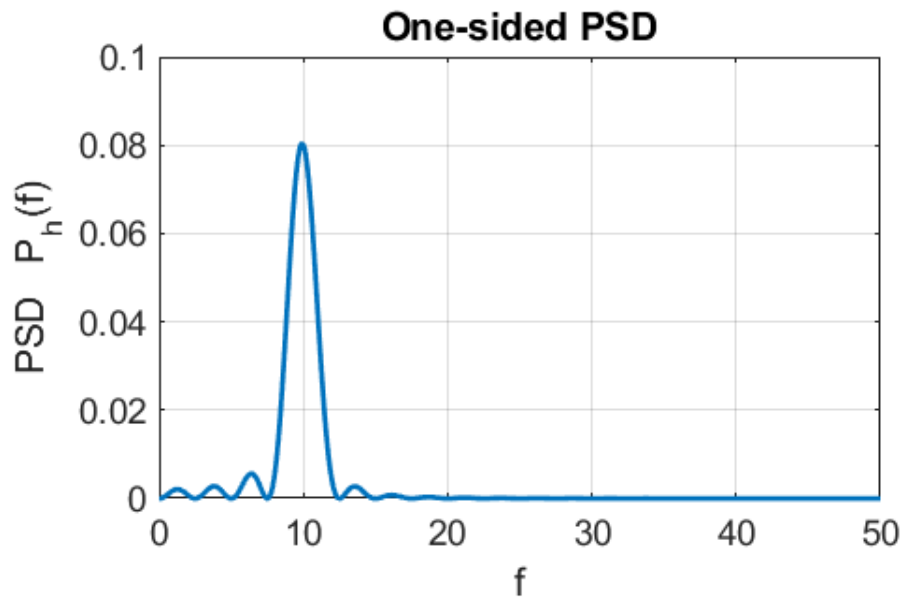
Fig. 3. One sided power spectral density PSD, $P_h(f)$.
Only the positive frequency interval is displayed.

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.

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

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

$$H(f) = \frac{2}{3 + i(2\pi f)}$$
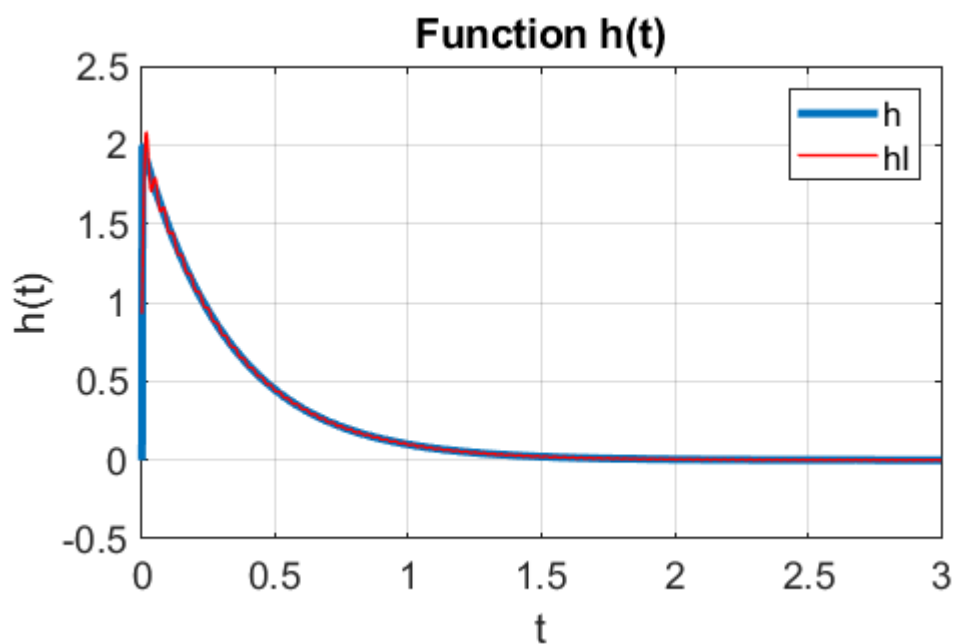
Fig. 4.   The function $h(t)$ and the inverse Fourier transform $hI(t)$.
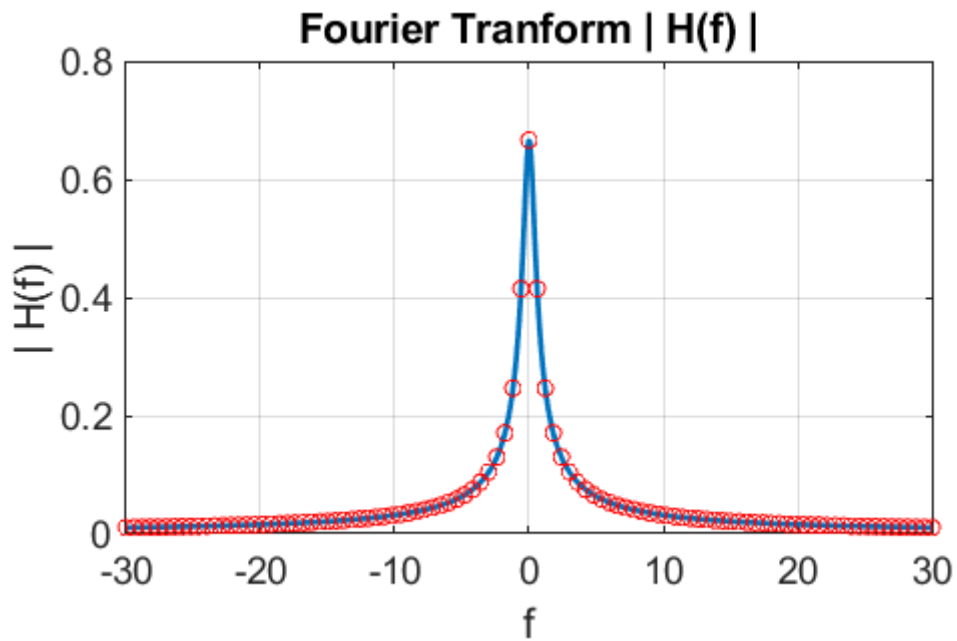
Fig. 5. The Fourier transform showing excellent agreement between the numerical results and the analytical prediction.
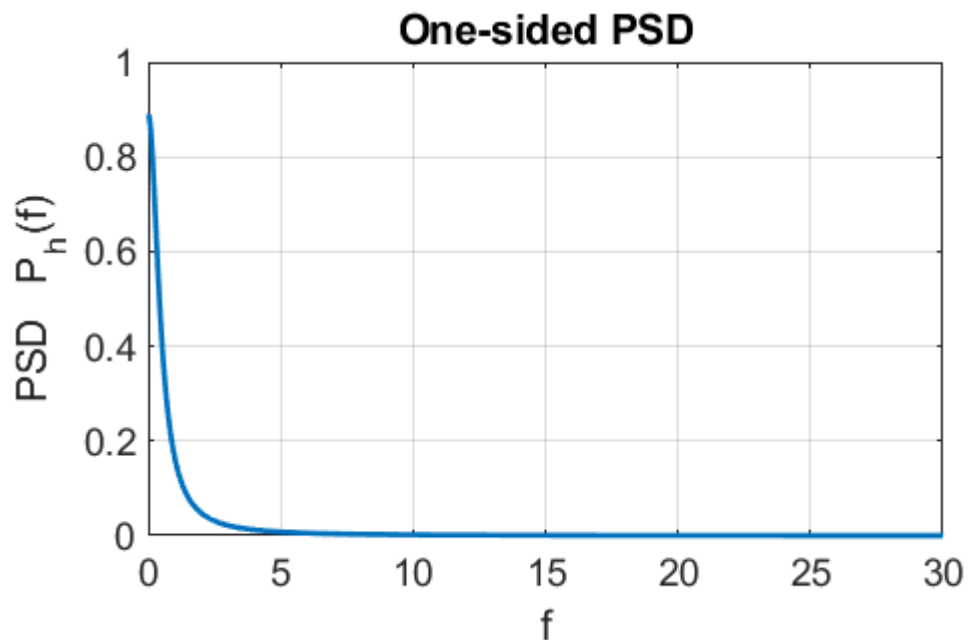
Fig. 6.   One sided power spectral density PSD, $P_h(f)$. Only the positive frequency interval is displayed.

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

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

$$h(t) = A_0 \sin\left(2\pi f_0 t + \phi_0\right)$$
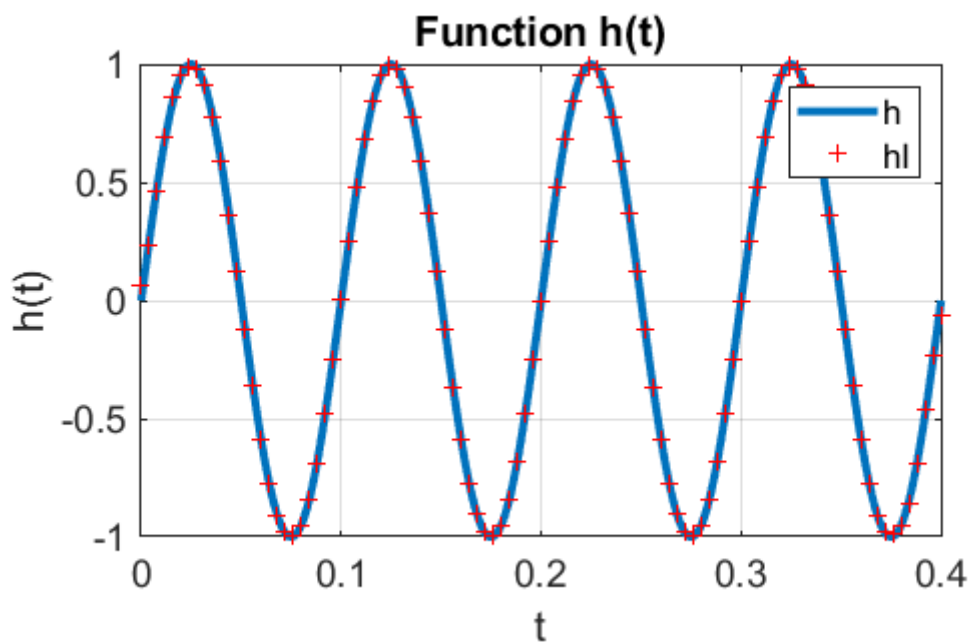
Fig. 7. The function $h(t) = A_0 \sin\left(2\pi f_0 t + \phi_0\right)$ where

$A_0 = 1$ $\quad f_0 = 10$ Hz $\quad \phi_0 = 0$ rad. $A = 1$. The units for time

t are seconds.
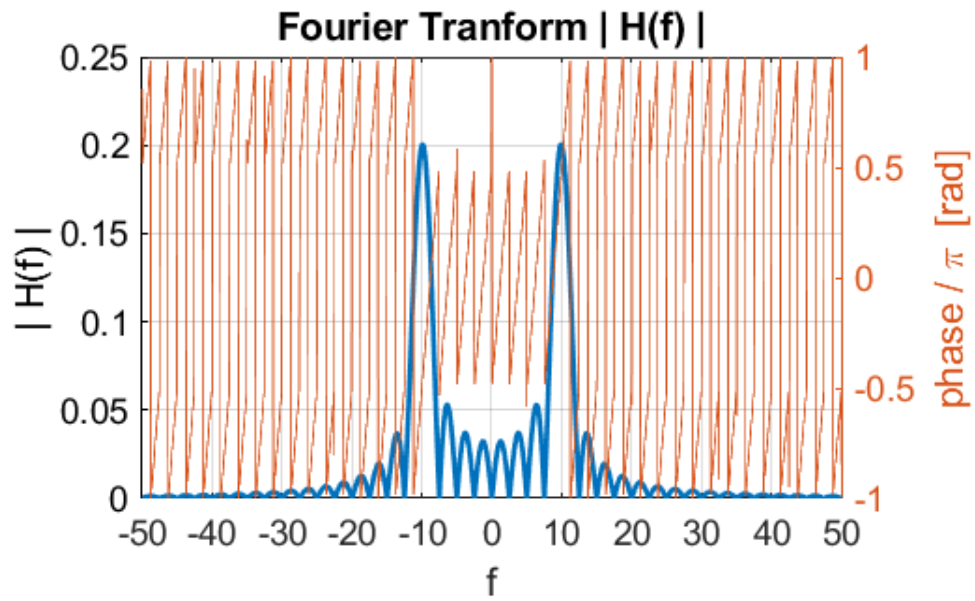
Fig. 8a.   Absolute value of the Fourier transform and its phase. To plot the phase, uncomment the code in the segment for figure 2.
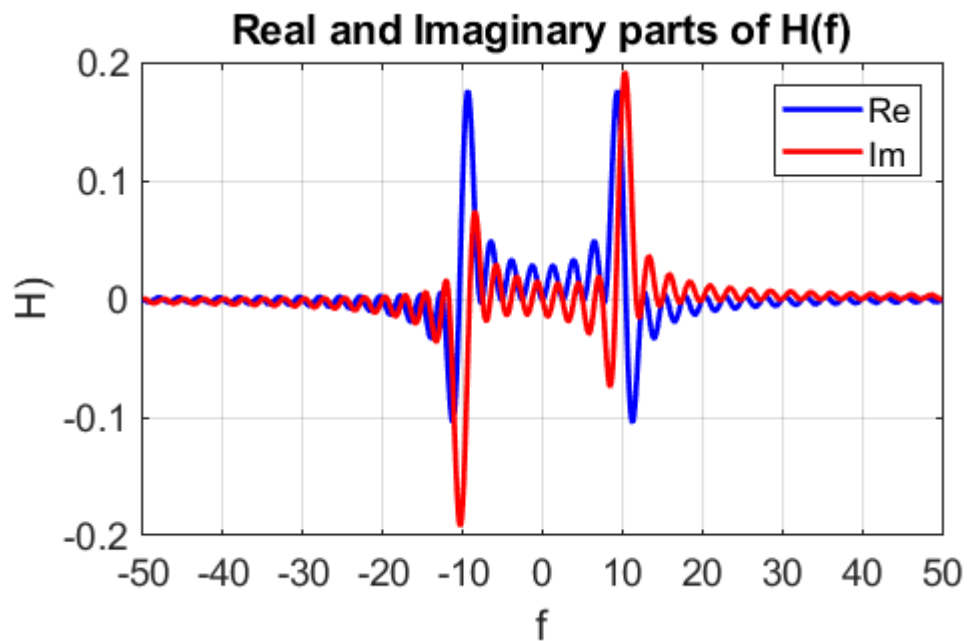
Fig. 8b.   Real and imaginary parts of the Fourier transform.

Fig. 9. 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 $\qquad$ PT = 0.2000

frequency domain $\quad PF$ = 0.2000

Doubling the amplitude: $A = 2$

time domain $\quad\quad\quad PT = 0.8000$

frequency domain $\quad PF = 0.7999$
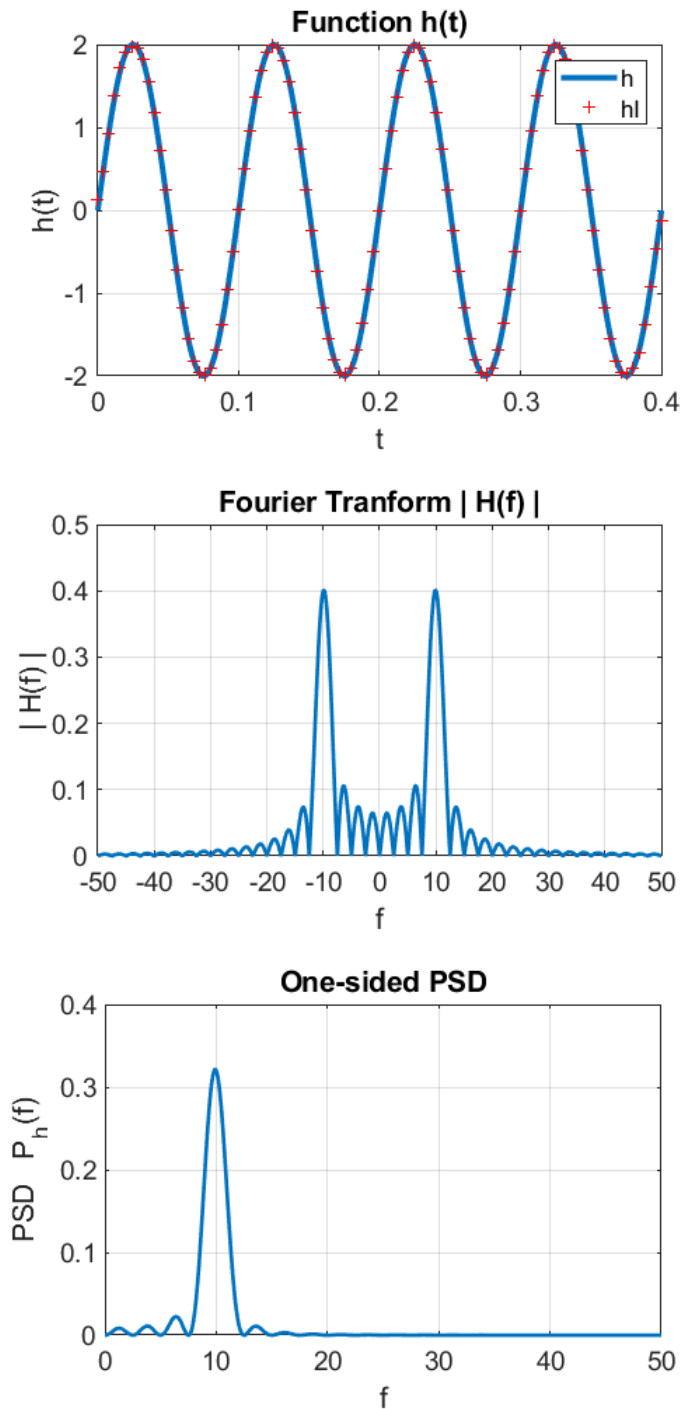
Fig. 10. When the amplitude is doubled, the energy supplied by the signal increases by a factor of 4 [$P \propto A^2$].

Doubling the frequency: f = 20 Hz

time domain PT = 0.2000

frequency domain PF = 0.1999
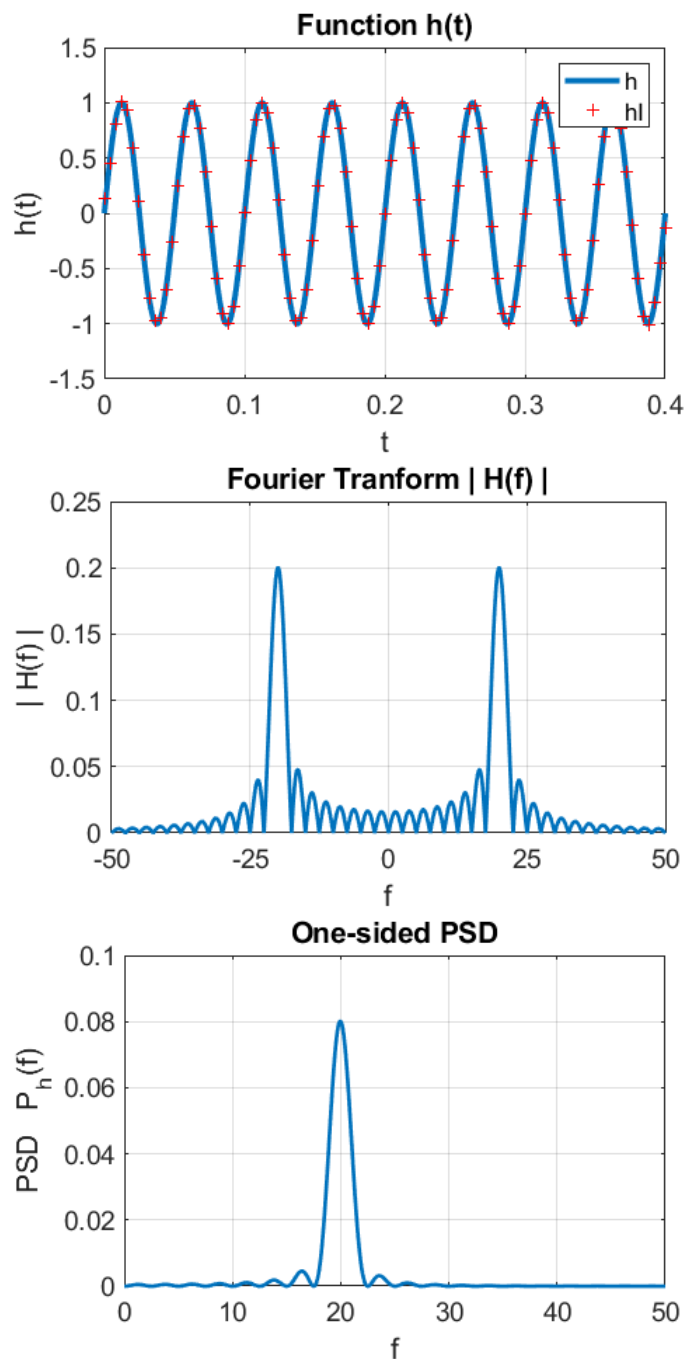
Fig. 11.   When the frequency is doubled, the energy supplied by the signal does not change (not increase x4 ???).

Changing the phase: $\phi_0 = \pi / 6$
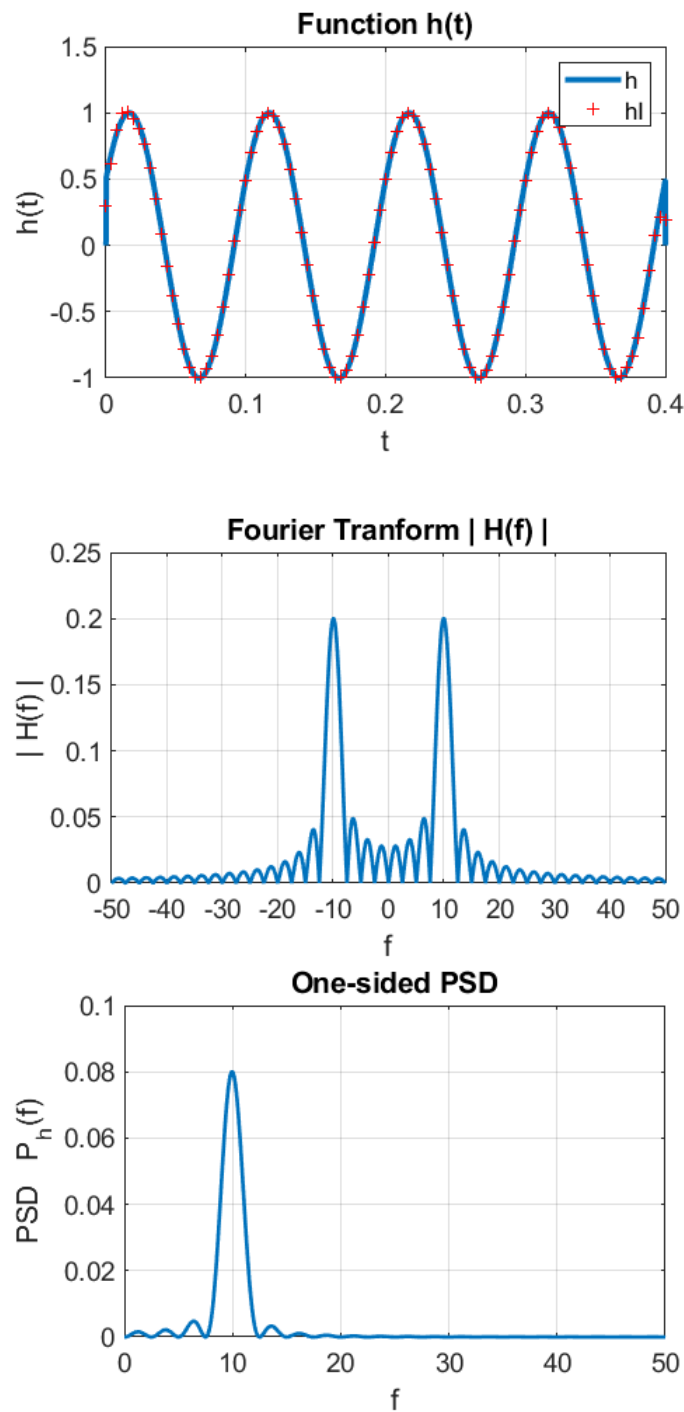
Fig. 12. Changing the phase of the sinusoidal function has negligible effect on the Fourier transform.

4. Superposition of sinusoidal functions

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

$$h(t) = A_1 \sin\left(2\pi f_0 t\right) + A_2 \sin\left(2\pi(2f_0)t\right) +$$
$$A_3 \sin\left(2\pi(3f_0)t\right) + A_4 \sin\left(2\pi(4f_0)t\right)$$
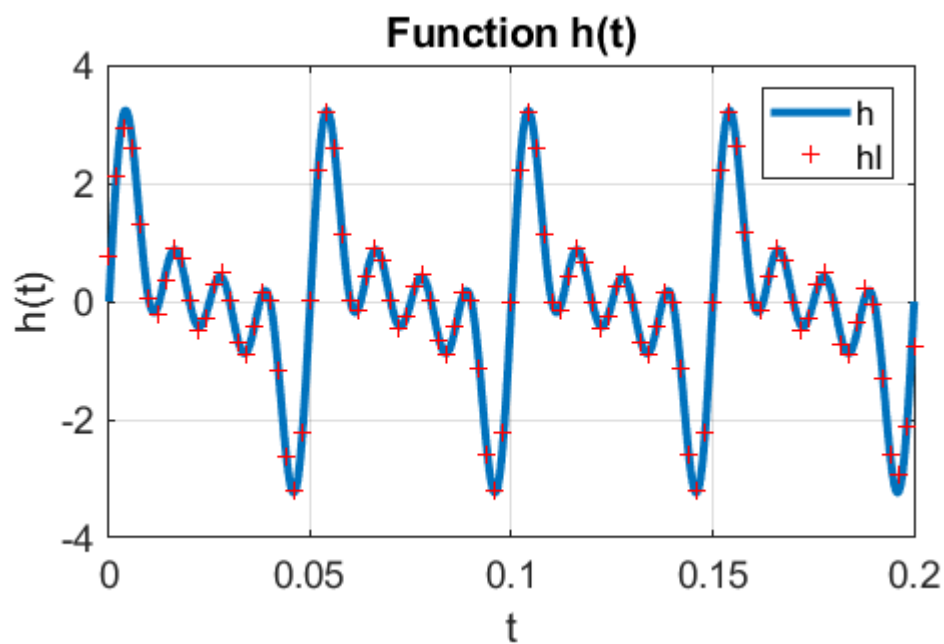
Fig. 13. The superposition of four sinusoidal function:
$f_0 = 20\ Hz \quad A_1 = A_2 = A_3 = A_4 = 1$
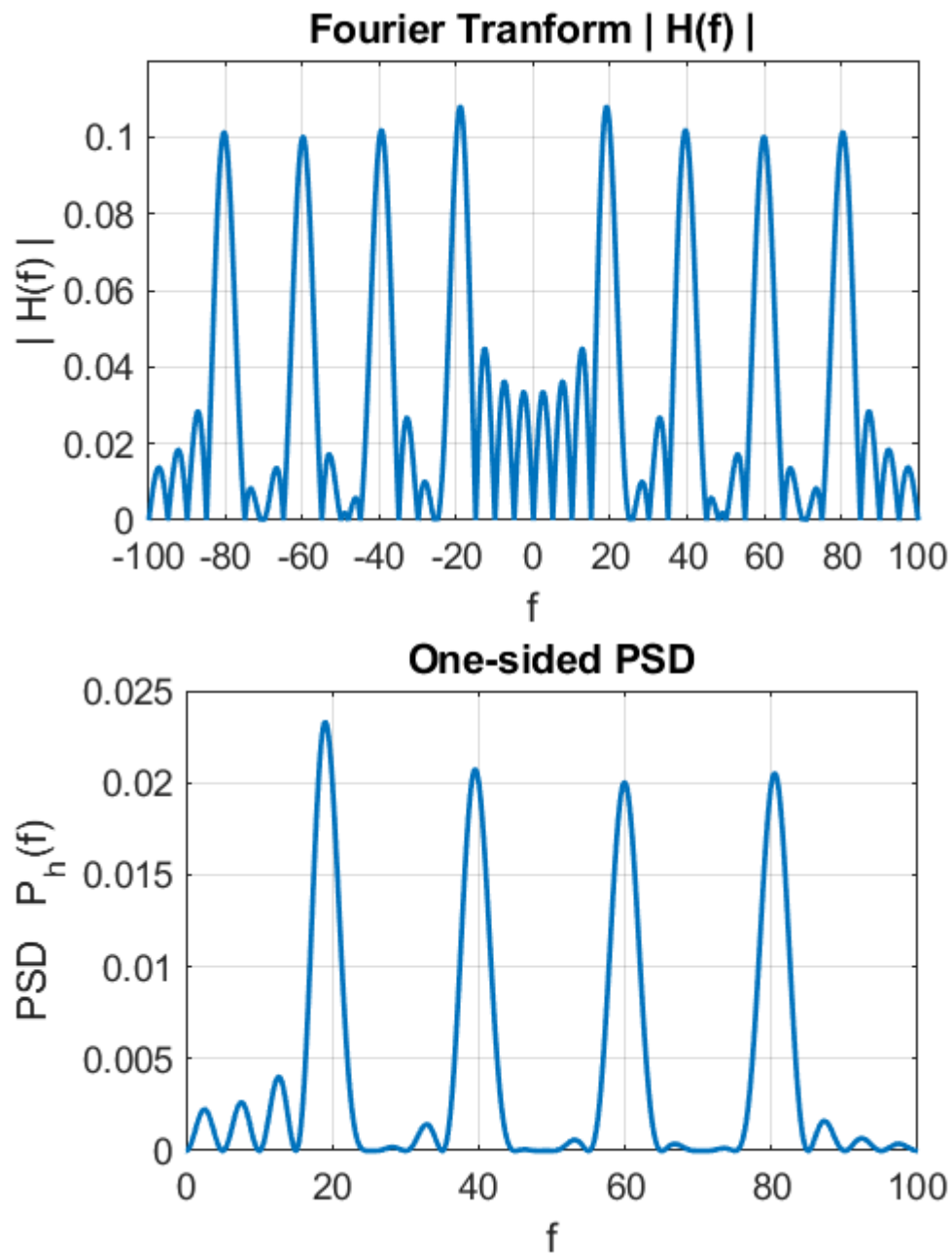
Fig. 14. The absolute value of the Fourier transform and the PSD function.

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:

$$f_0 = 20 \; Hz$$

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

$$A_3 = 3 \quad f_3 = 60 \quad A_4 = 4 \quad f4 = 80$$
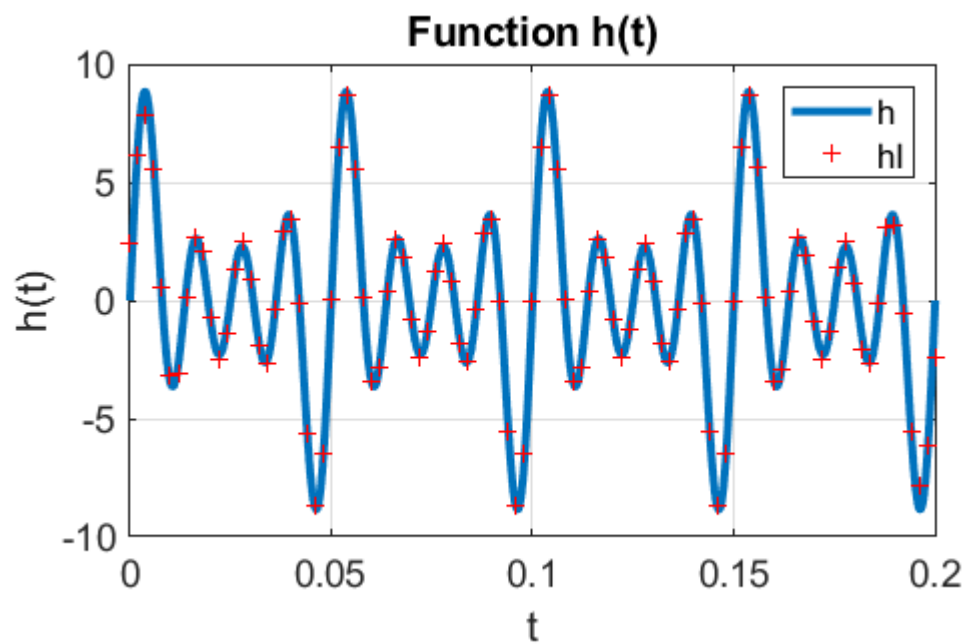
Fig. 15.   The function and the inverse Fourier transform.

Fourier Tranform | H(f) |
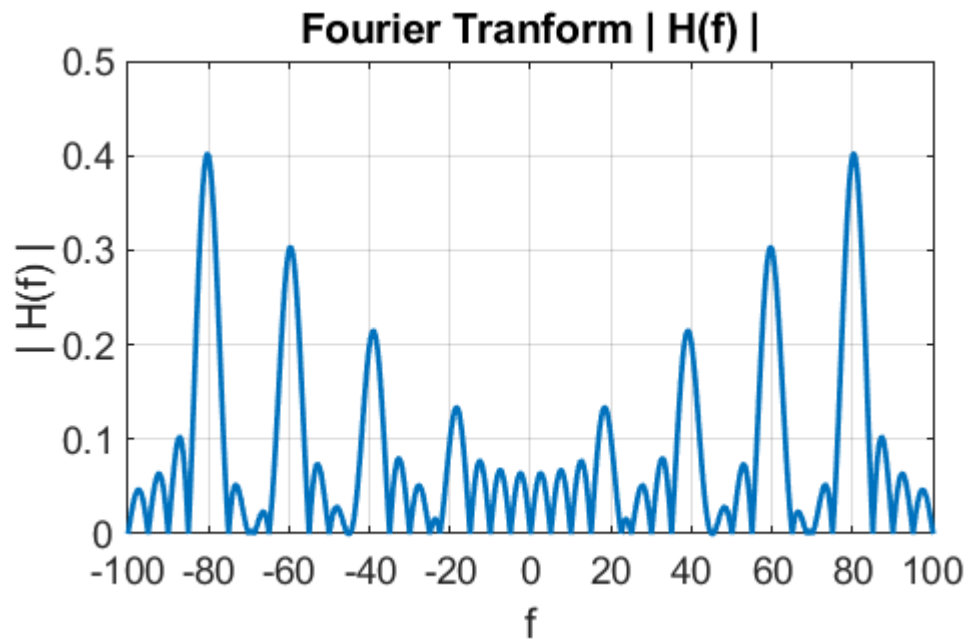
Fig. 16.   The Fourier transform.

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]
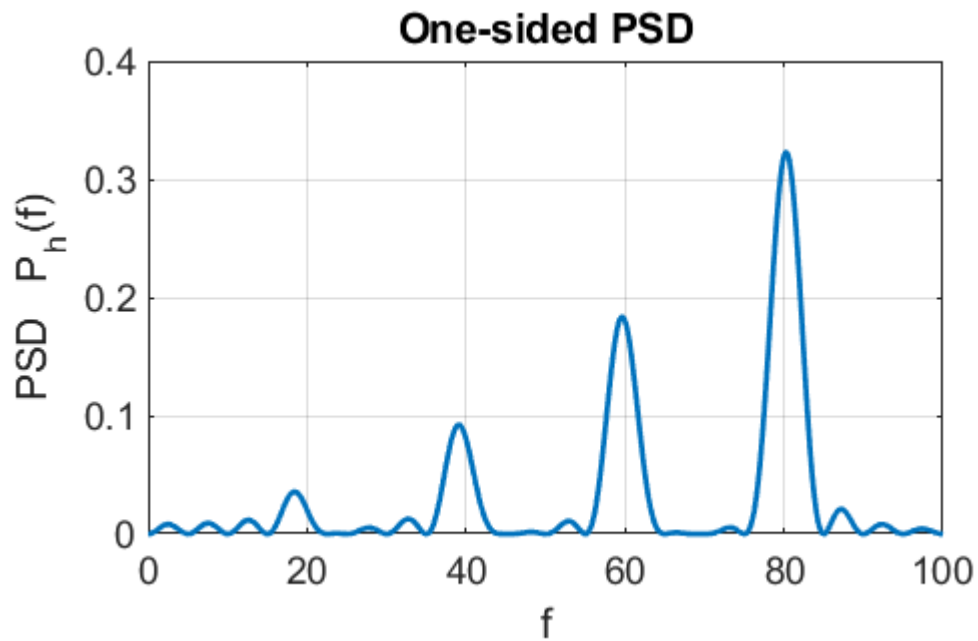
Fig. 17. The PSD function.

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

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

$$\frac{\sin\left(2\pi\left(5f_0\right)t\right)}{5} + \frac{\sin\left(2\pi\left(7f_0\right)t\right)}{7} + \ldots$$
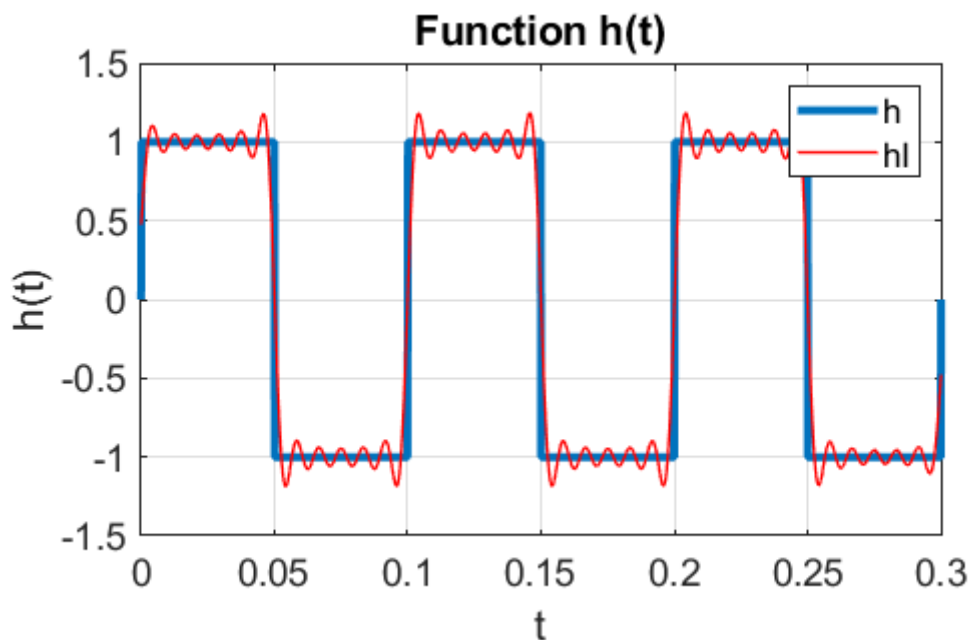
Fig. 18.   The square wave signal $\left(f_0 = 10 \text{ Hz}\right)$ and its inverse Fourier transform.

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.
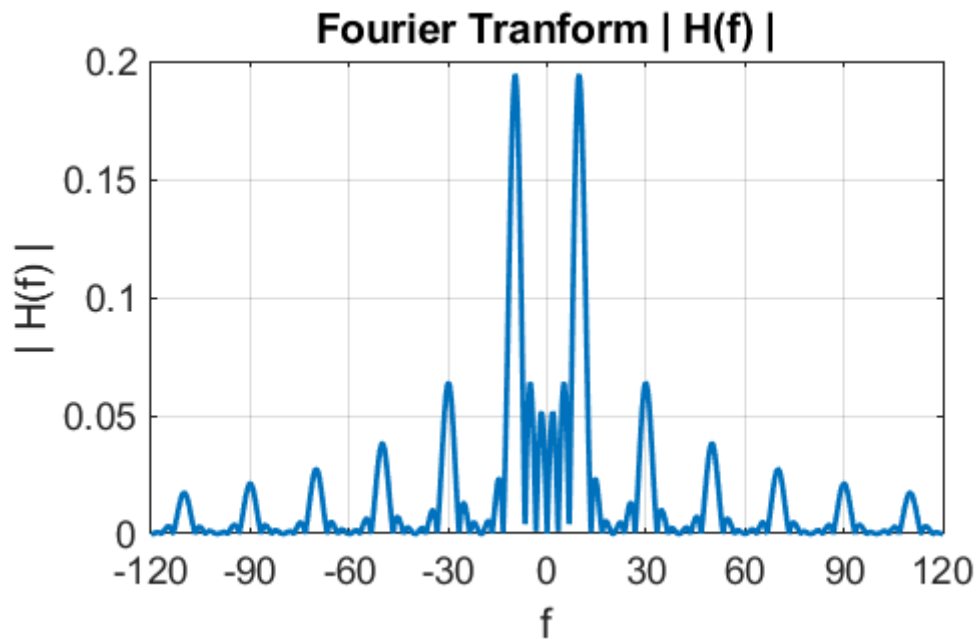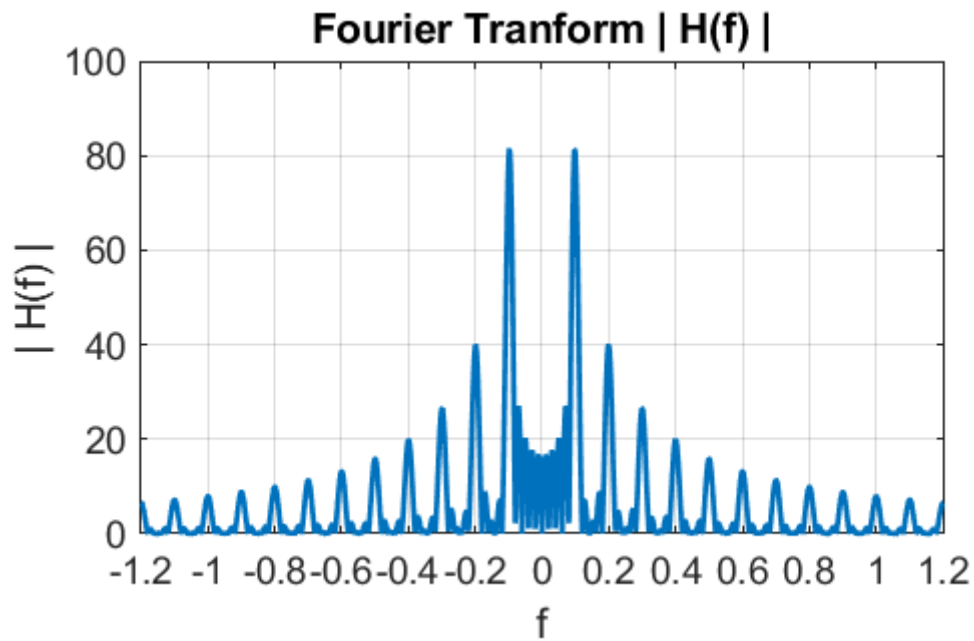
Fig. 19. The Fourier transform of the square wave signal.

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.
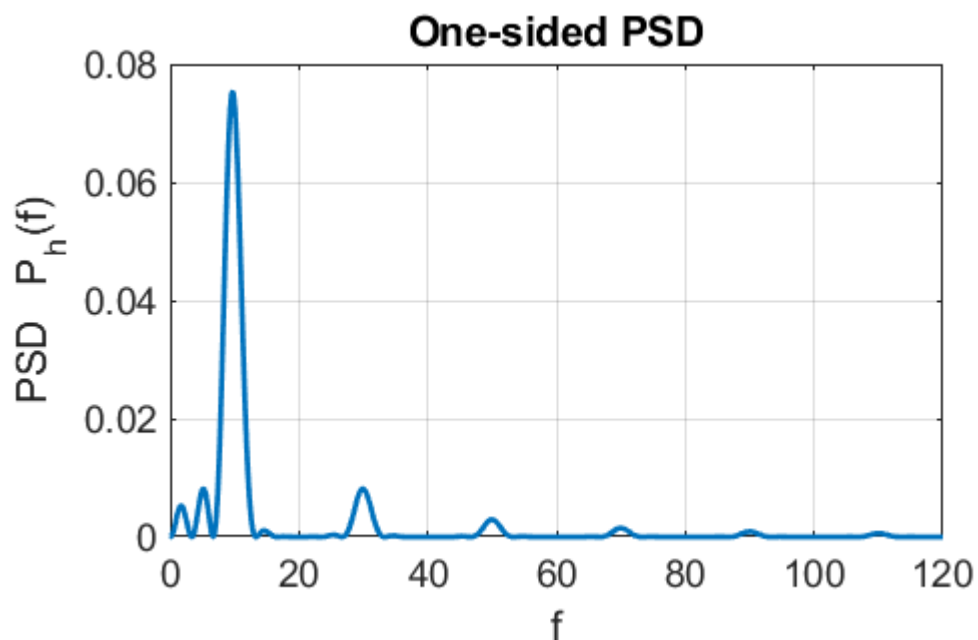
Fig. 20.   The PSD function.

6.   SAWTOOTH FUNCTION

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

$$h(t) = A + B \sum_{n=1}^{\infty} (-1)^n \frac{\sin(2\pi n f_0 t)}{n}$$
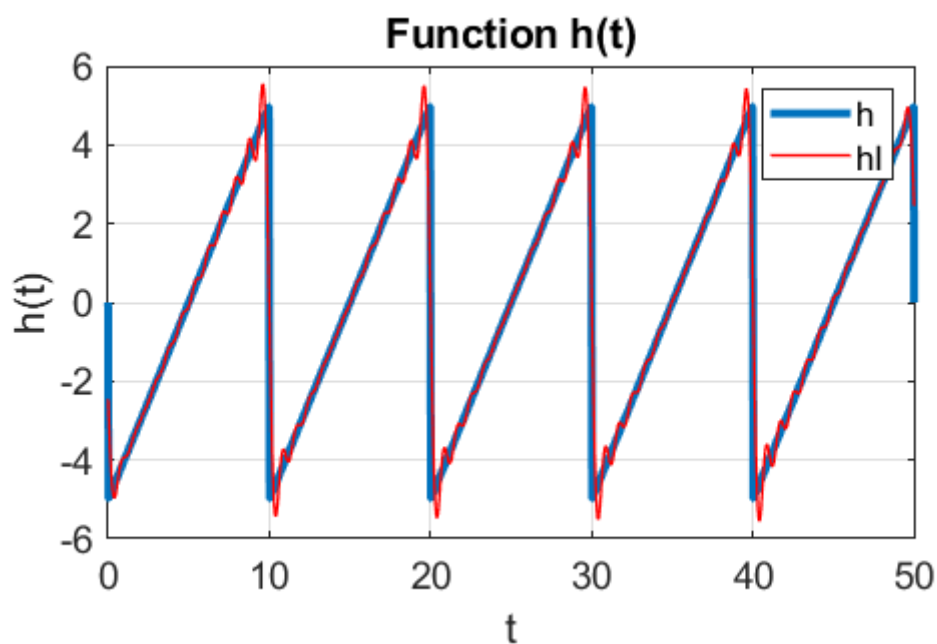
Fig. 21.   The sawtooth function and the inverse Fourier transfer. The DC component of the function has been removed. The fundamental frequency is $(f_0 = 0.10 \text{ Hz})$.

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

Fig. 22.   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
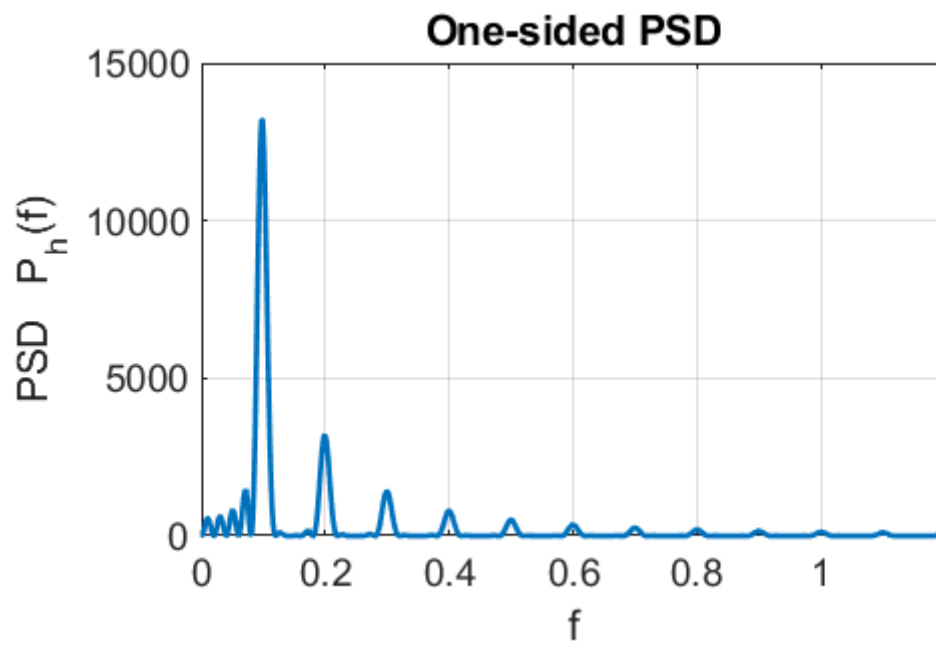
Fig. 23. The PSD function.

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.
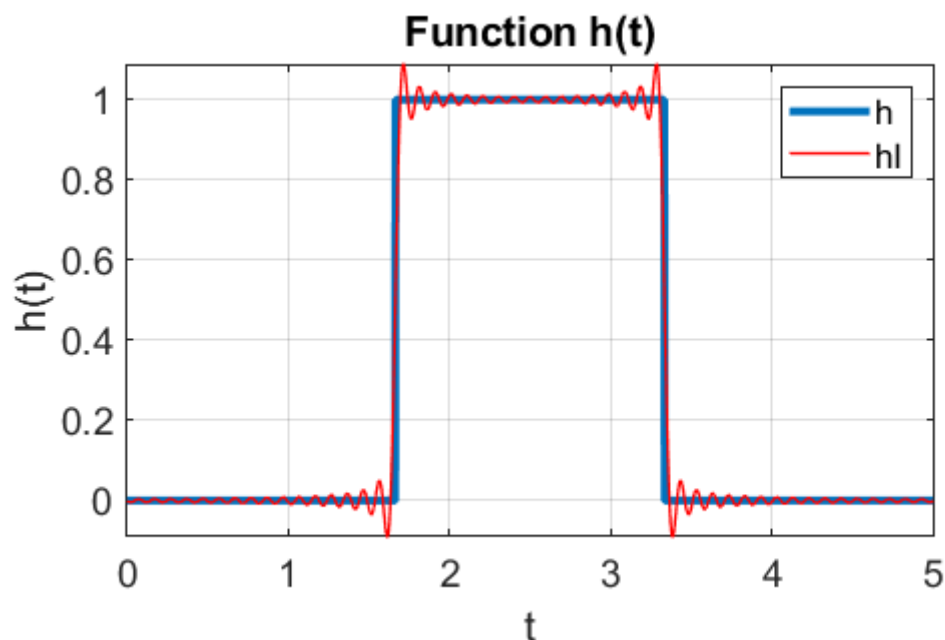
Fig.24.   The square pulse and its inverse Fourier transform.
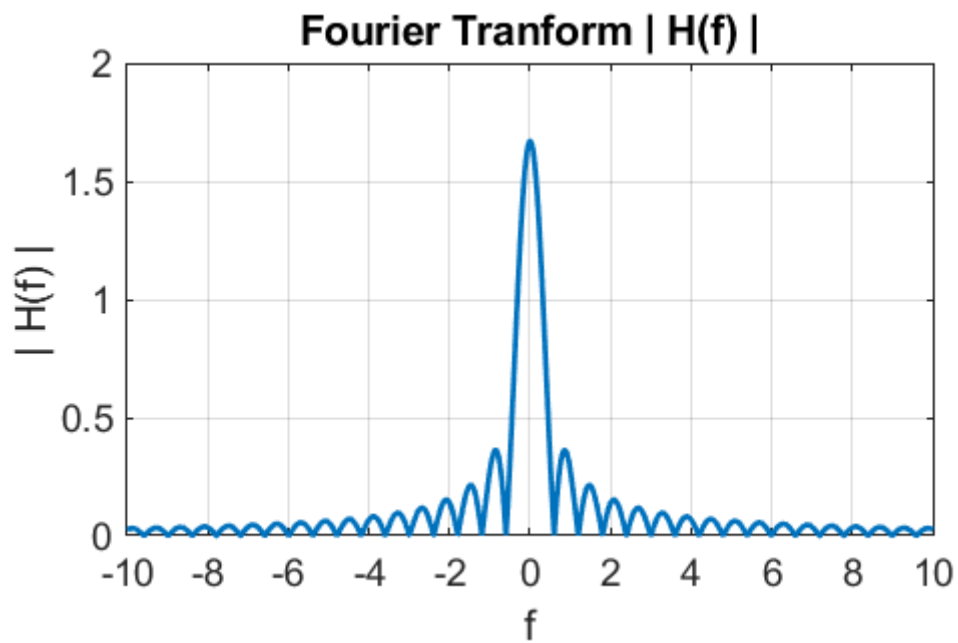
Fig. 25. The Fourier transform of the square pulse with a large DC component.
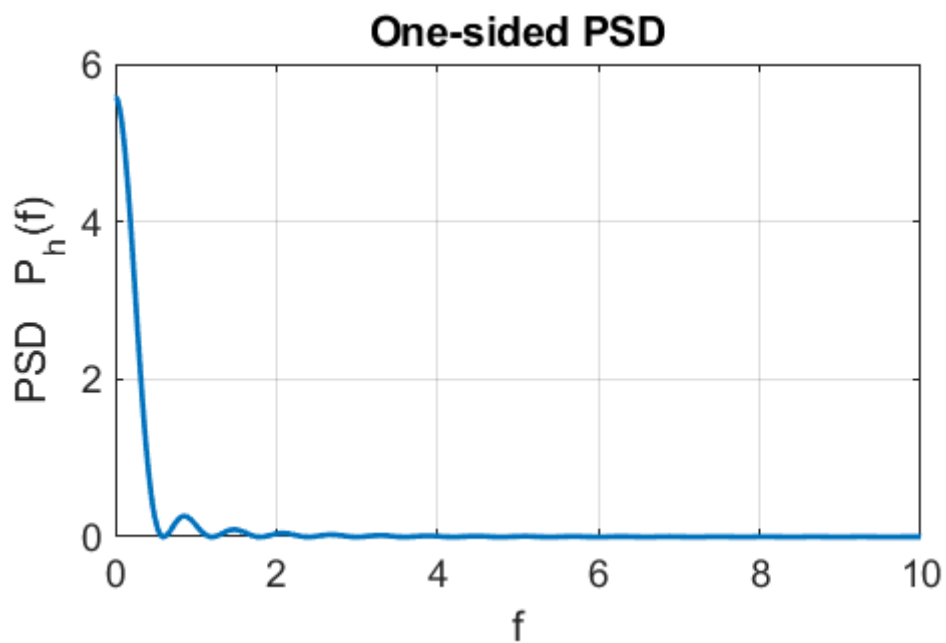
Fig. 26. The PSD function for the square pulse.
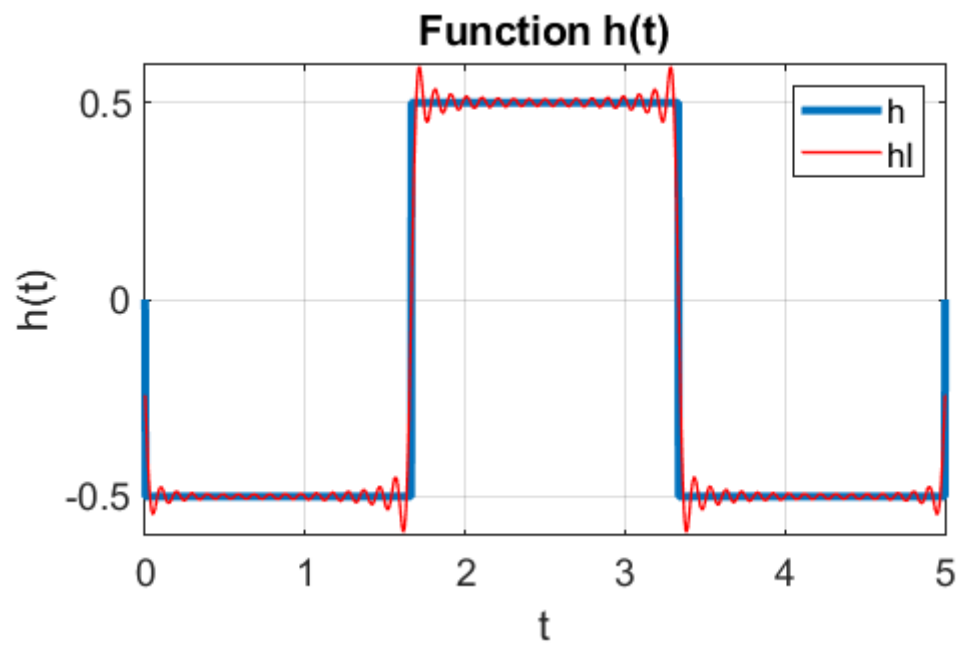
Fig.27. The square pulse and its inverse Fourier transform with the DC component removed.
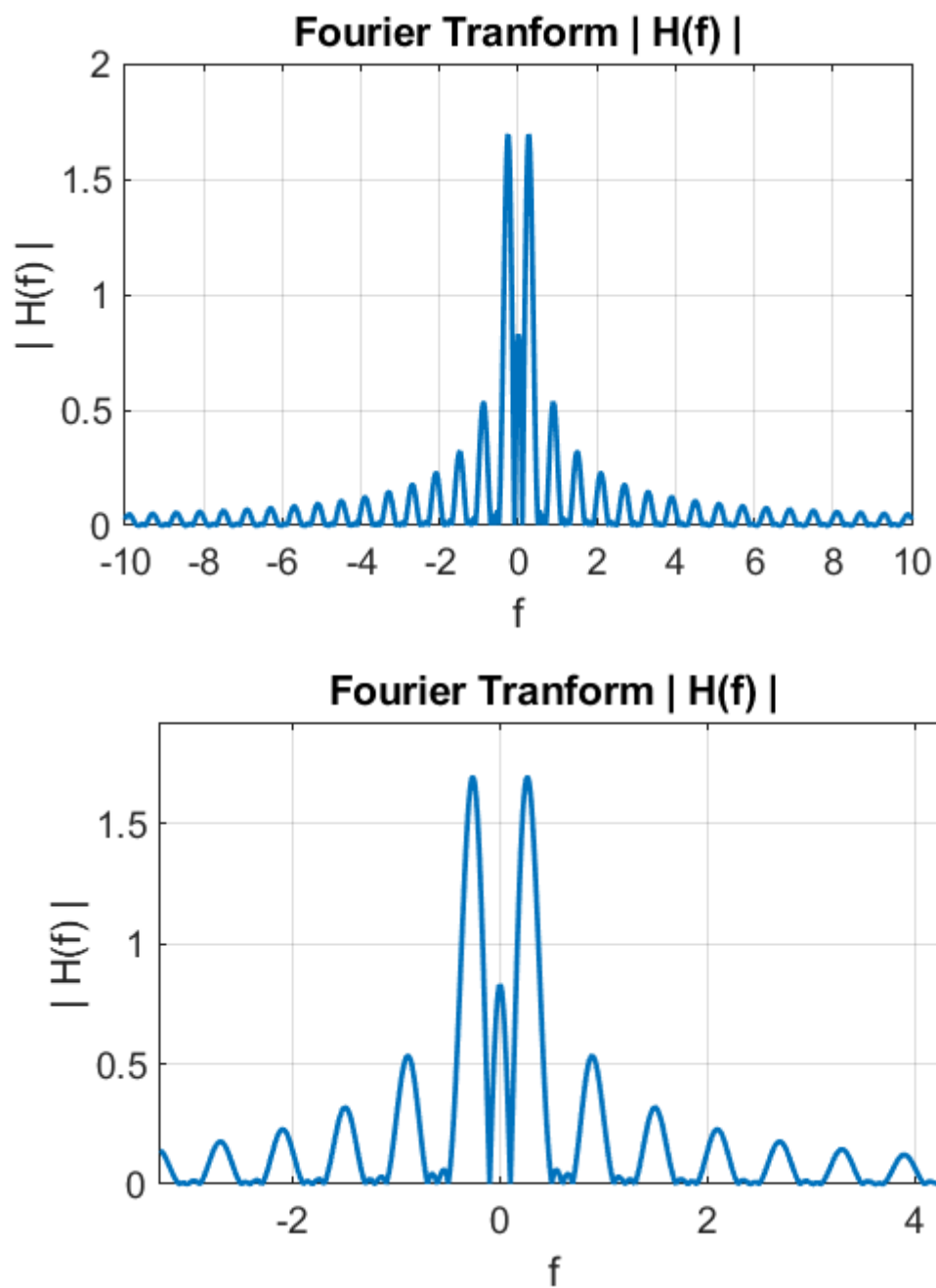
Fig. 28. The absolute value of the Fourier transform of the square pulse with the DC component removed. The spacing between the peaks for $f > 0$ or $f < 0$ is constant $(\Delta f = 0.60 \text{ Hz})$.

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.
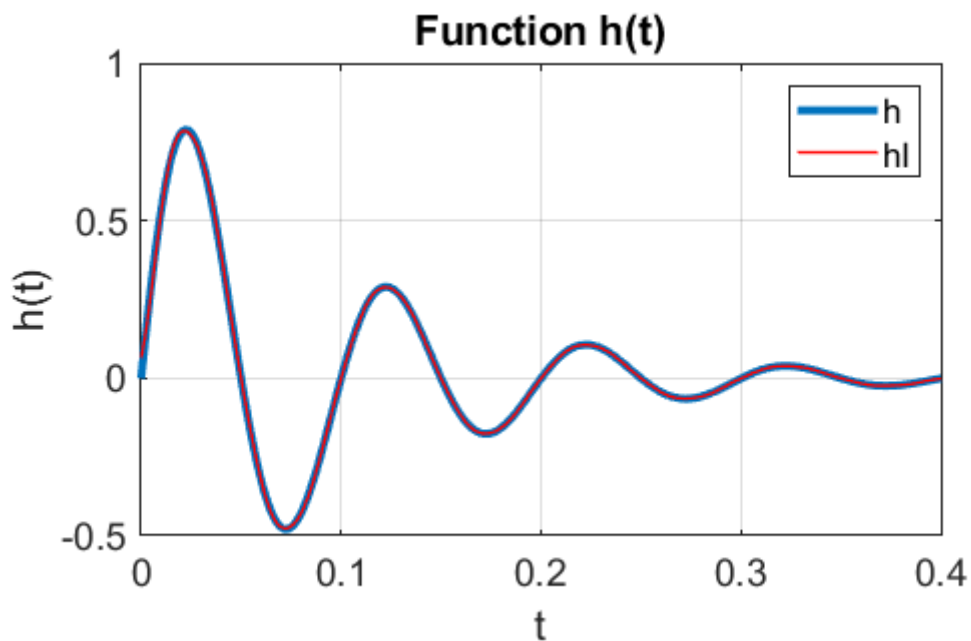
Fig. 29. A damped sine wave with

$$A = 1 \quad f_0 = 10 \text{ Hz} \quad \tau = 0.1 \text{ s}.$$
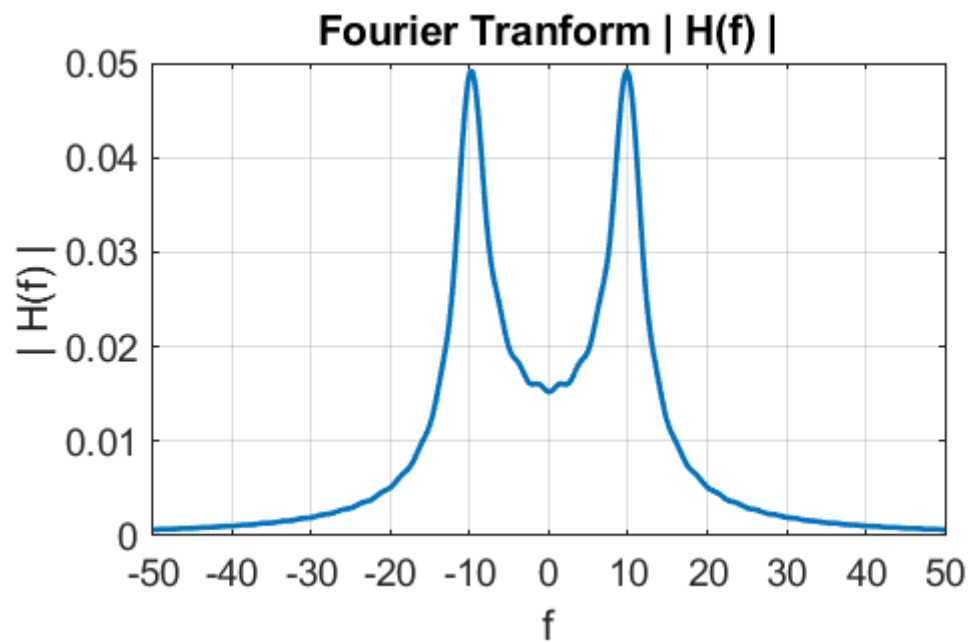
Fig. 30. The Fourier transform of the damped sine.
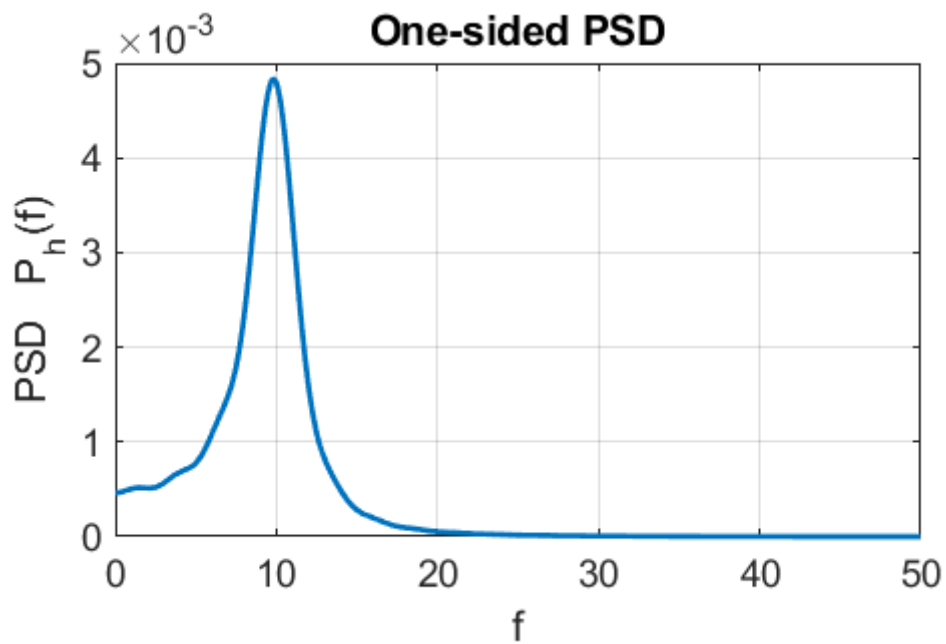
Fig. 31. Energy spectrum for the damped sine wave.

The width of the PSD function at half maximum power is approximately equal to $\Delta f \approx \dfrac{1}{\pi \tau} = 3.18 \ \text{Hz}$. 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.
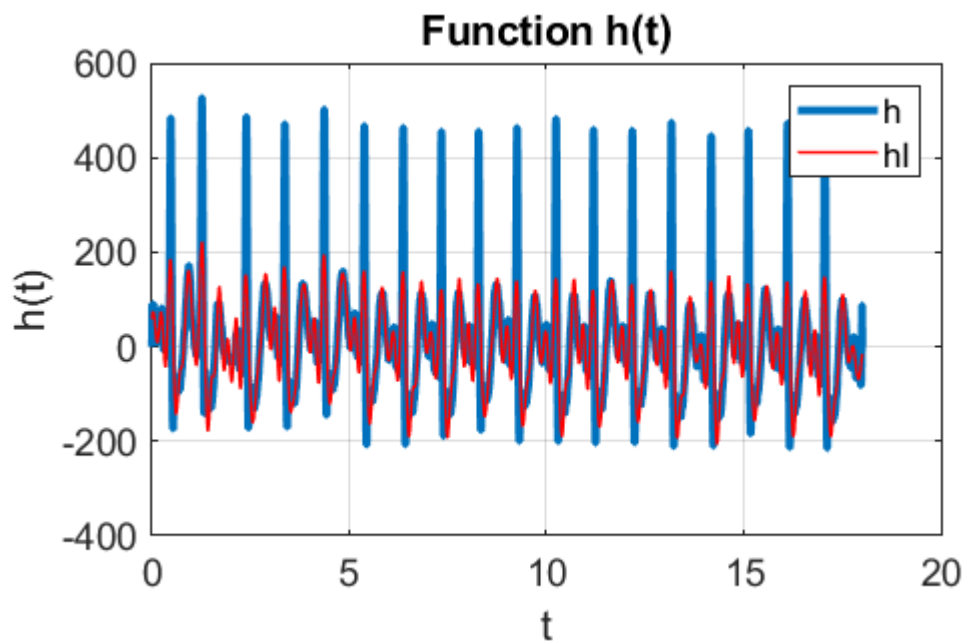
Fig. 31.   The ECG recording.
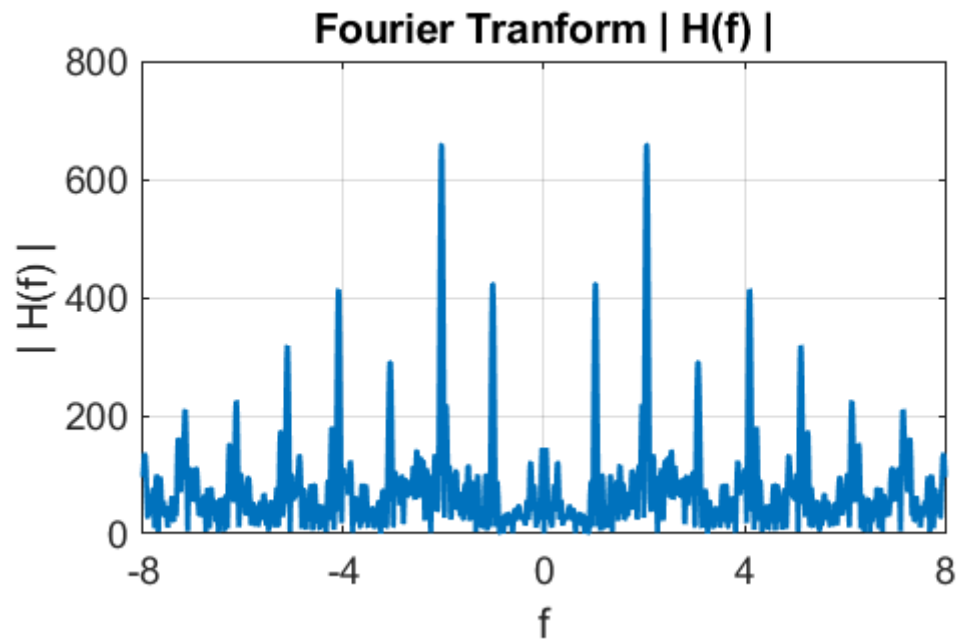
Fig. 32.   The Fourier transform of the ECG recording.
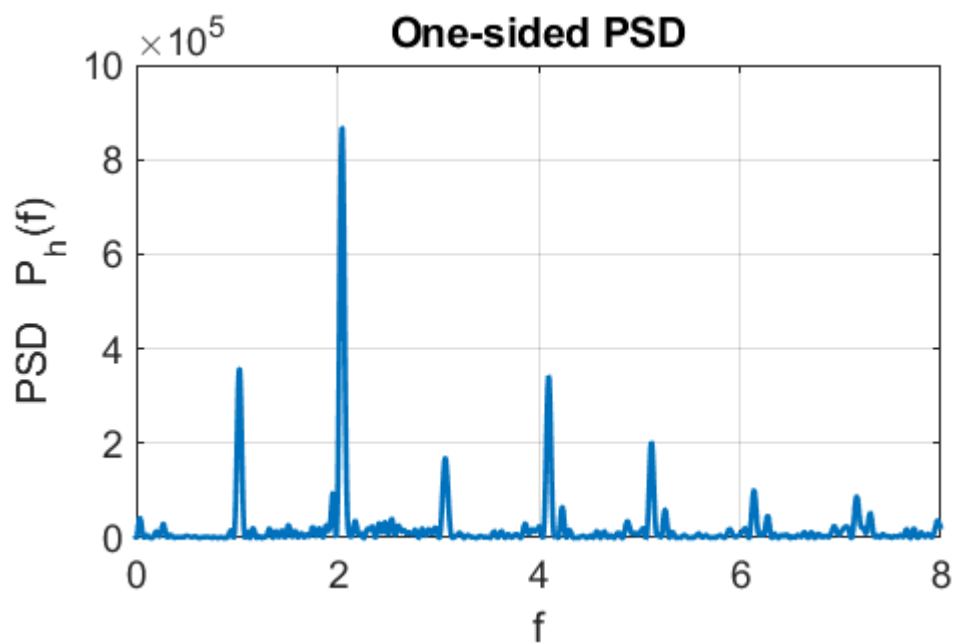
Fig. 33.   PSD function for the ECG recording.

10 BEATS (sound files)

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**.
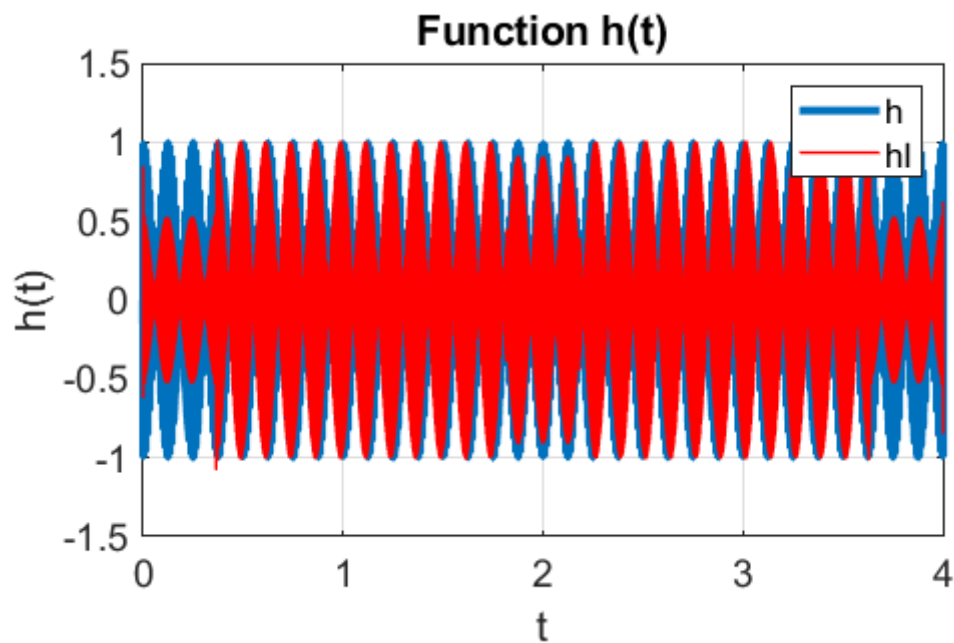
Fig. 34. Beat pattern of two pure tones 1000 Hz and 1008 Hz.
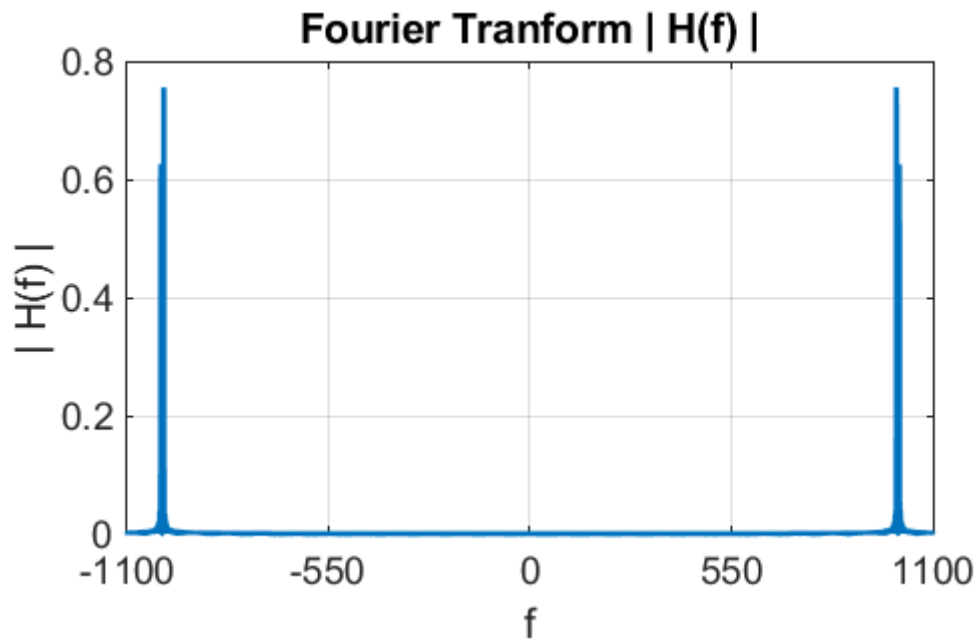
Fig. 35. The Fourier transform of the beat signal. It took about 43 seconds to calculate the Fourier transform.
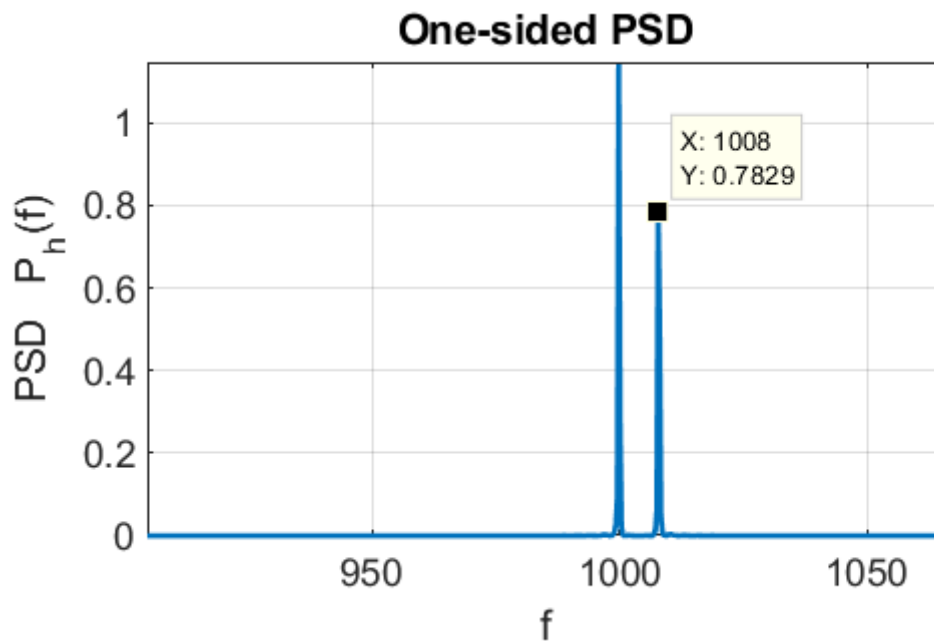

Fig. 36. Zoom view of the PSD function showing the two peaks at frequencies 1000 Hz and 1008 Hz.

11.   Train Whistle

The recording of a train whistle is loaded from the wav file
train.wav.
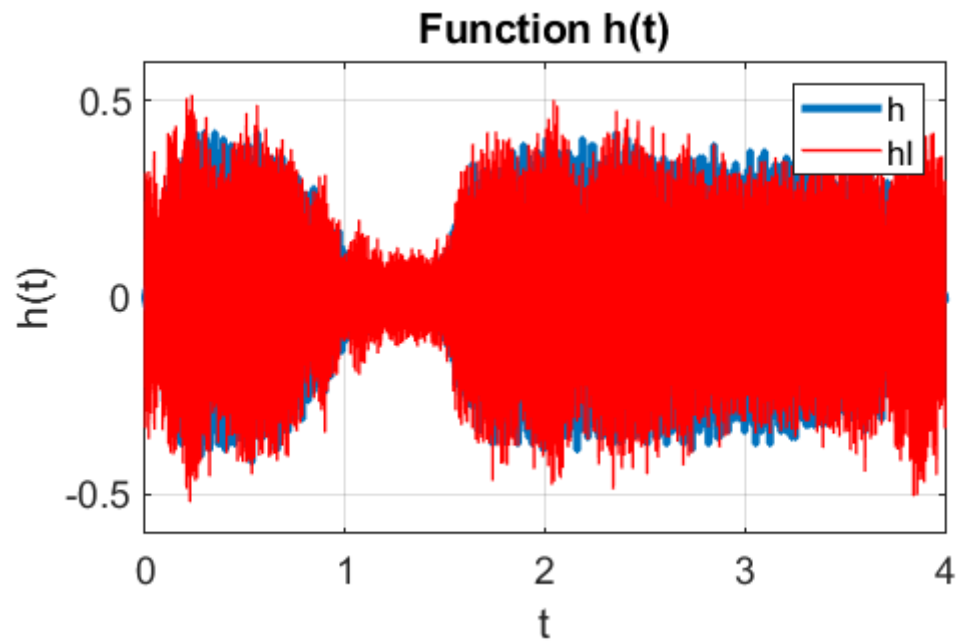
Function h(t)

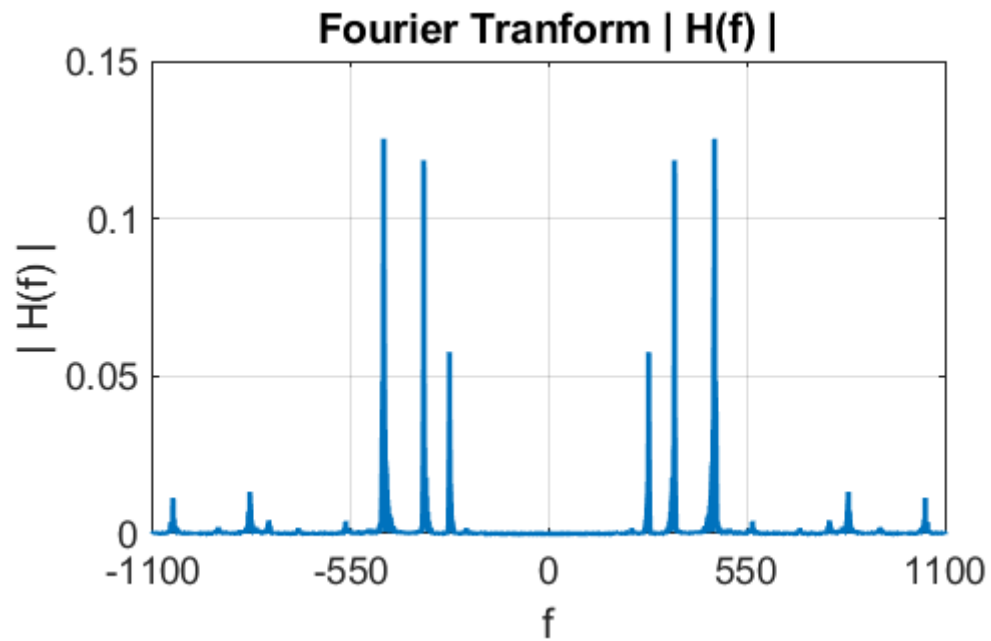

Fig. 37.   The sound of a train whistle.
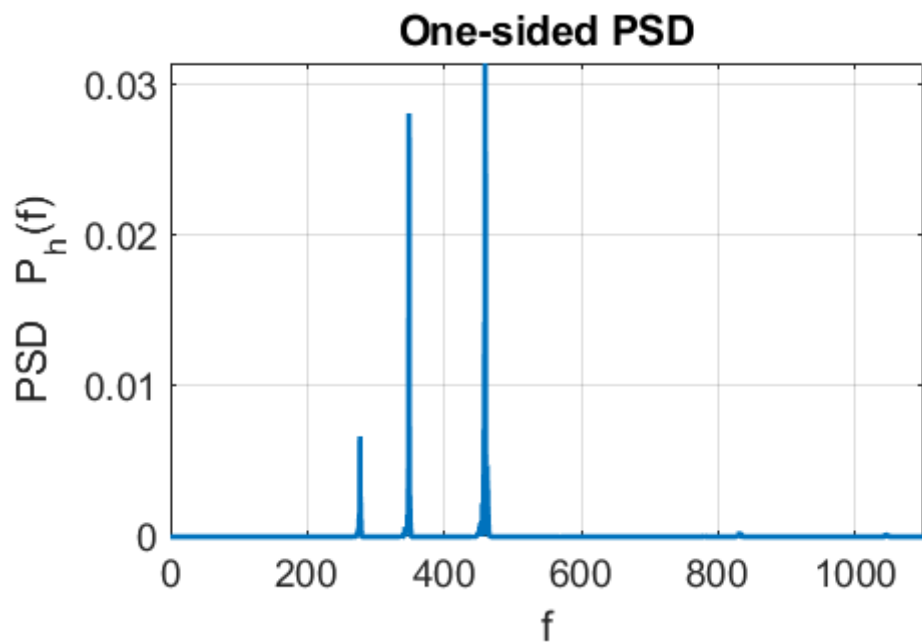
Fig. 38.   The Fourier transform of a train whistle.

Fig. 39.   The PSD function for a train whistle.

The peak frequencies at 277 Hz, 348 Hz and 459 Hz. *(Not sure of the time scaling so frequencies maybe incorrect)*.

12. 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);
```
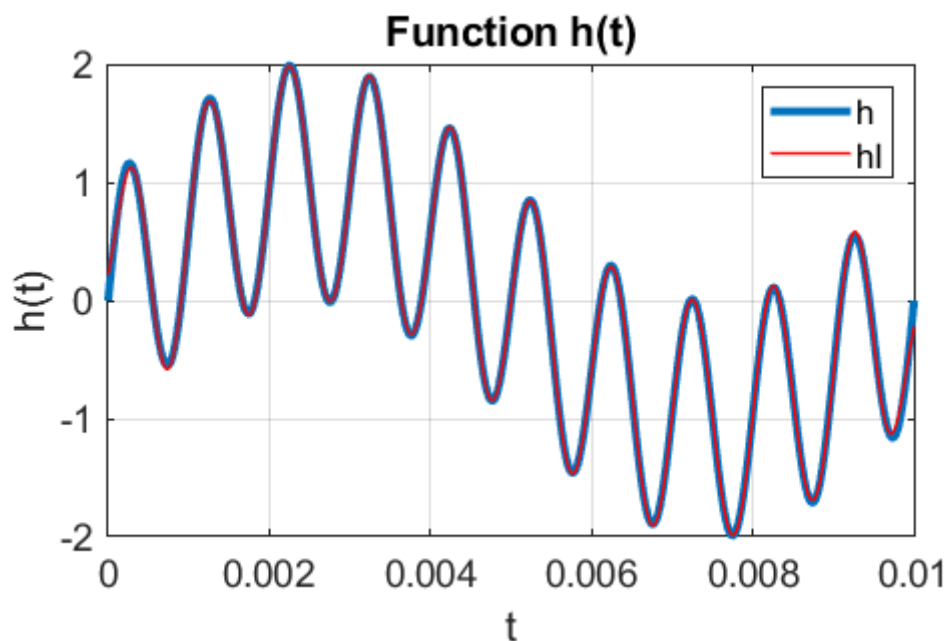

Fig. 40. Unfiltered signal with frequency components 1000 Hz and 100 Hz.
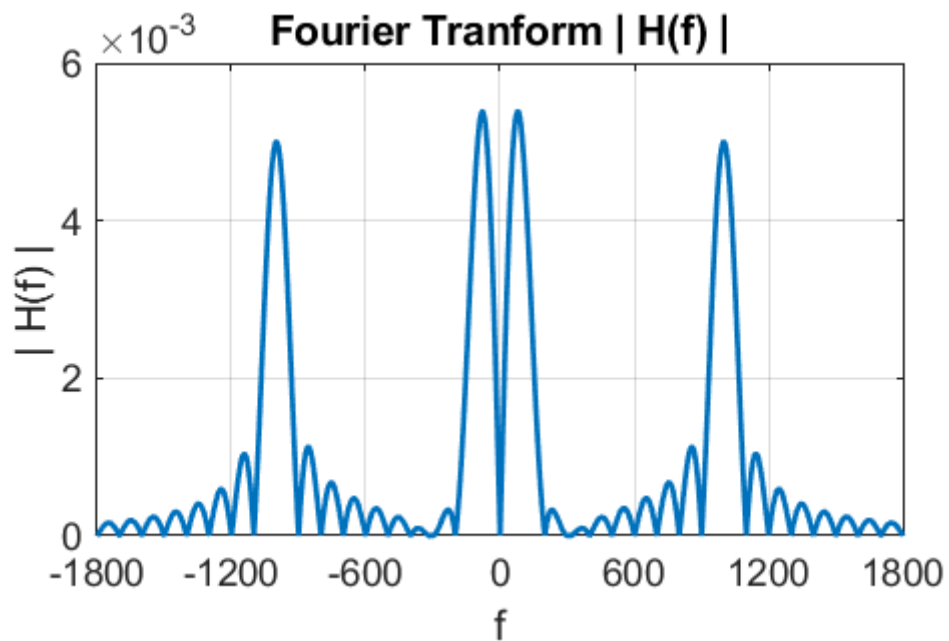
Fig. 41.   Fourier transform of the unfiltered signal with frequency components 1000 Hz and 100 Hz.
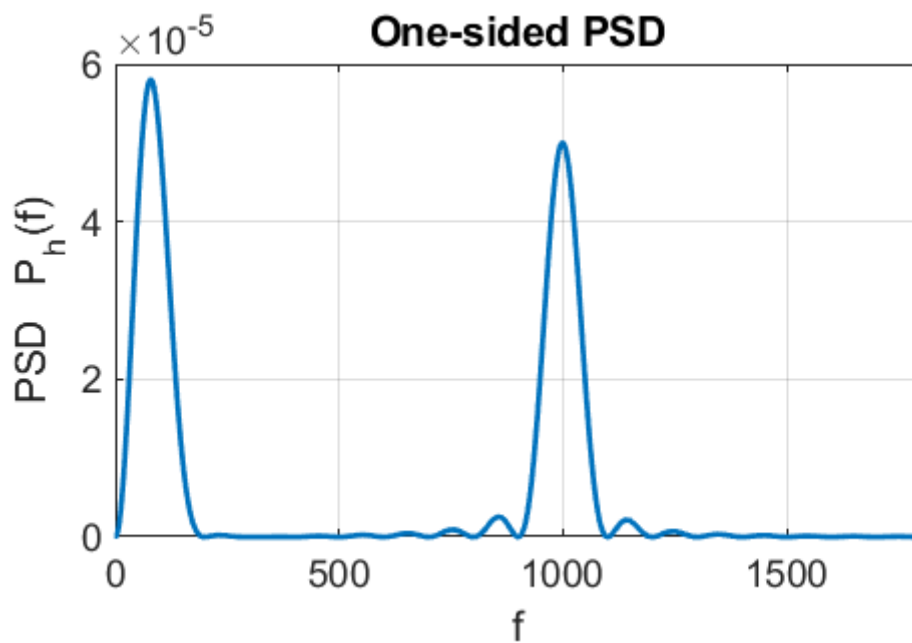

Fig. 40.   PSD function for the unfiltered signal with frequency components 1000 Hz and 100 Hz.

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;
```
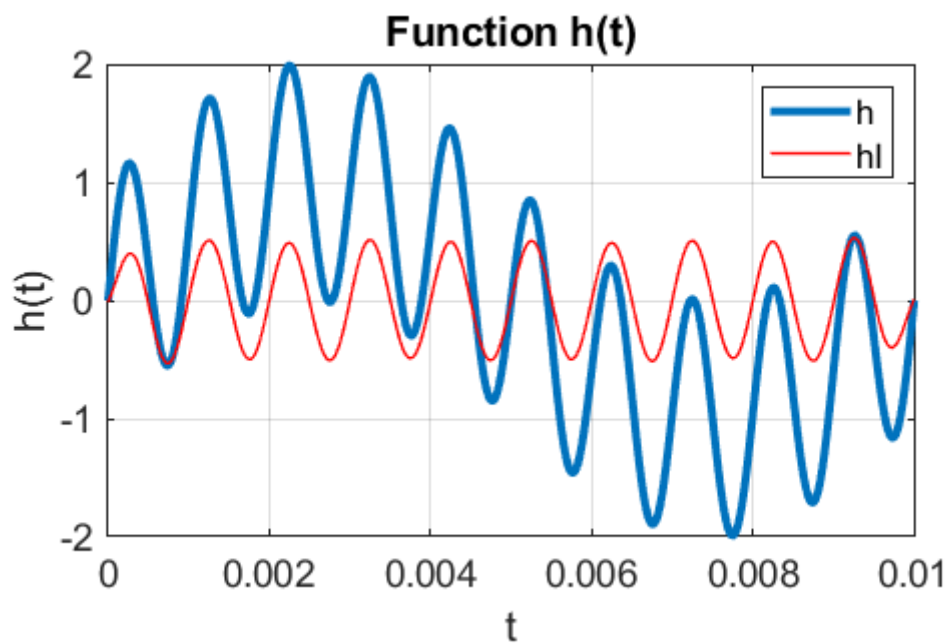

Fig. 41. The 100 Hz frequency component is removed and we are left with the pure 1000 Hz signal.
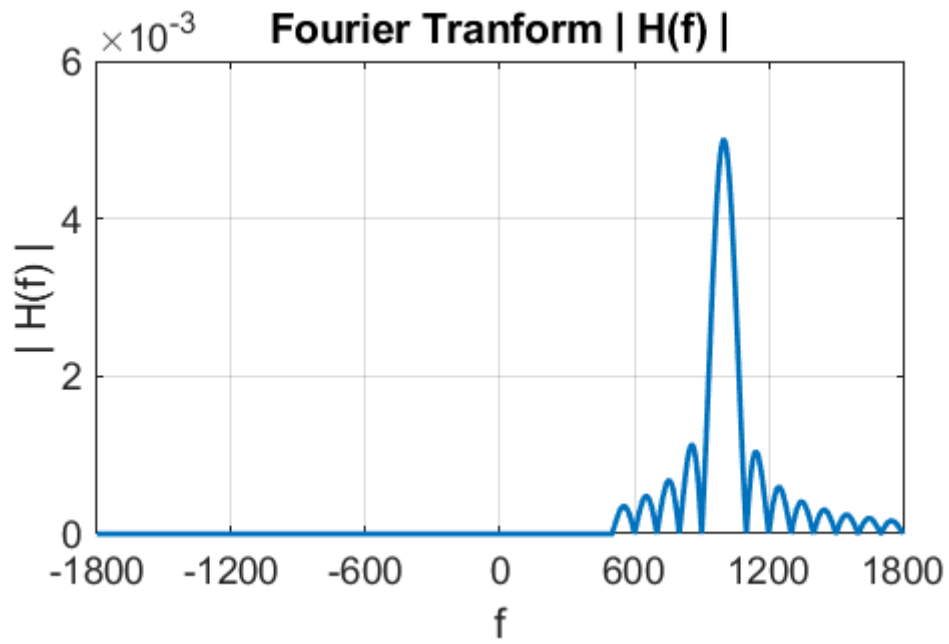
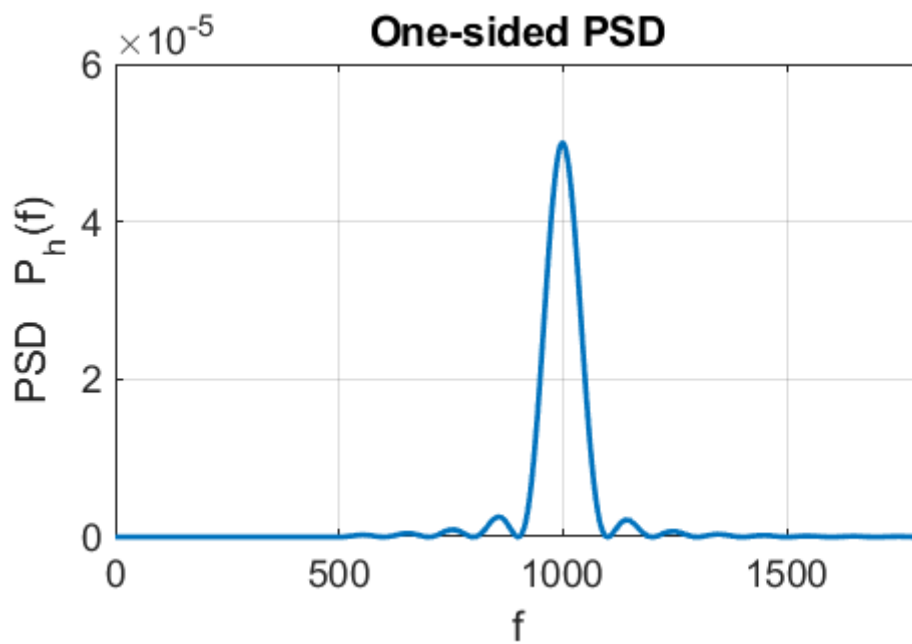Fig. 42. The filtered Fourier transform with the low frequency 100 Hz component removed.

Fig. 43. The filtered PSD function with the low frequency 100 Hz component removed.