

3 Fourier transform

Knowledge of cyclic or oscillating activity in signals is very important. Fourier transform is the oldest method for analysing the cycles, i.e. obtaining frequency information. For example, analysing the frequency of variable stars is the oldest application of analysing cycles [1]. But then, what is frequency?

Frequency measures the periodicity (i.e. repetitiveness); it measures the number of cycles per second in Hz. Fundamental period (in seconds) is the inverse of the frequency. For example, see the saw-tooth waveform below.

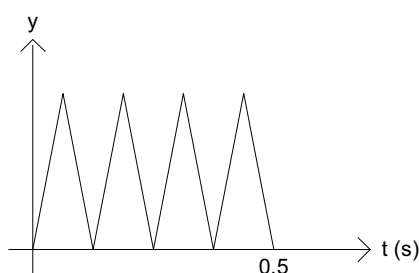


Figure 3.1: Saw-tooth waveform.

In the figure, there are 4 *fundamental* cycles in 0.5 s. Hence, each cycle is completed in 0.125 s and the freq is $1/0.125 = 8$ Hz.

Normally, the Fourier method is used to obtain the magnitude of frequency components – from Figure 3.1, we know that the frequency is 8 Hz but how strong is this frequency component? We can use Discrete Fourier Transform for this purpose.

Since we are dealing with discrete-time signals in this book, we will skip discussions on continuous Fourier transform and move into discrete Fourier transform (DFT). But we need to understand the concept of discrete frequency before discussing DFT.

3.1 Discrete frequency

Just like the discrete variable n for time domain, we have discrete frequency number m for frequency domain. Assume N is the number of sampled points, we have the discrete frequency variable (frequency number), m ranging from 0 to $N-1$. It should be noted that the actual frequency range will be from 0 to one point less than sampling frequency F_s with F_d intervals. Hence, the frequency spacing will be

$$F_d = \frac{1}{NT} \text{ or } F_d = \frac{F_s}{N}. \quad (3.1)$$

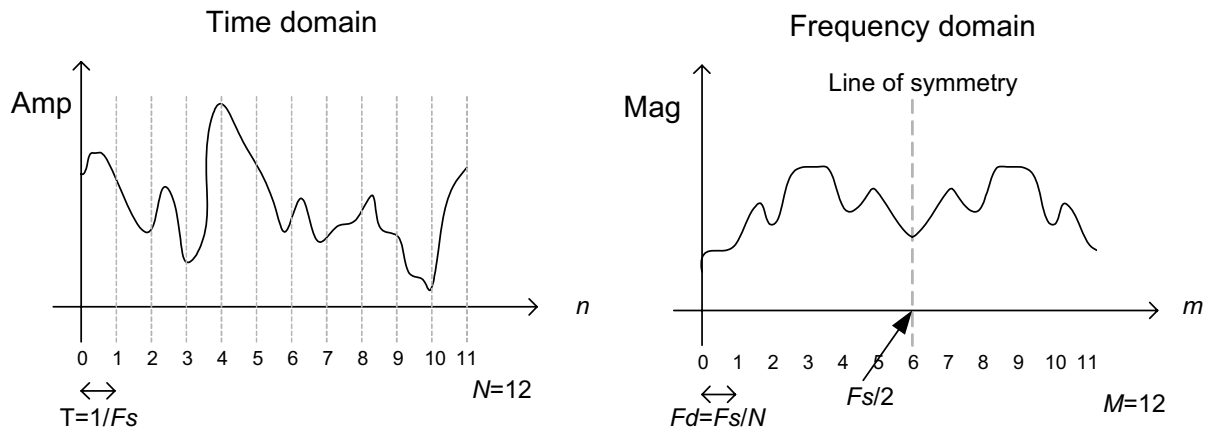


Figure 3.2: Example illustrating the connection between discrete-time and frequency number.

Figure 3.2 shows an example illustrating the discrete-time and frequency number. Here, the discrete-time signal is of length $N=12$ and as such, when transformed into the frequency domain, the frequency number m will range from 0 to $N-1$, i.e. 0,1,2,3,...,11. The actual frequency (which is $m \cdot F_d = m \cdot F_s / N$) ranges from 0 to one point less than F_s . It should be obvious from the figure that there is a line of symmetry and due to this symmetry, sometimes we plot spectral magnitude with the values from 0 to $(F_s/2)$ rather than up to F_s .

If $F_s = 6$ Hz for the discrete-time signal shown in Figure 3.2, the frequency abscissa will be from 0 to 3 Hz. Ignoring symmetrical parts, we will have Figure 3.3 where the frequency number $m=0,1,\dots,(N/2)$ (the rest of $N/2$ points are not plotted due to symmetry in frequency domain). Thus, $m = 0,1,2,3,4,5,6$ and actual frequency is $(m \cdot F_s) / N$: 0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0.

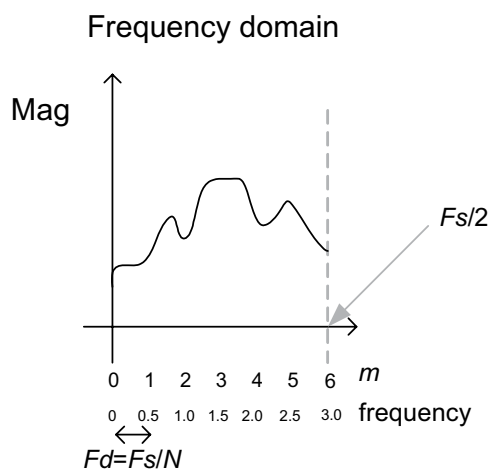


Figure 3.3: Frequency plot (from Figure 3.2) showing range up to $F_s/2$.

3.2 Discrete Fourier transform

DFT can be used to perform Fourier analysis of discrete-time signals. For discrete-time signal $x[n]$, it is defined as¹²

$$X_{DFT}[m] = \sum_{n=0}^{N-1} x[n] e^{-j\omega n}. \quad (3.2)$$

Since $\omega = 2\pi f / F_s$ and $f = mF_s / N$, we have

$$X[m] = \sum_{n=0}^{N-1} x[n] e^{-\frac{j2\pi mn}{N}}, \quad (3.3)$$

while inverse DFT (IDFT) is defined as

$$x[n] = \frac{1}{N} \sum_{m=0}^{N-1} X[m] e^{\frac{j2\pi mn}{N}}. \quad (3.4)$$

Using Euler's relation,

$$e^{jx} = \cos(x) + j \sin(x), \quad (3.5)$$

.....Alcatel-Lucent 

www.alcatel-lucent.com/careers

What if you could build your future and create the future?

One generation's transformation is the next's status quo. In the near future, people may soon think it's strange that devices ever had to be "plugged in." To obtain that status, there needs to be "The Shift".



DFT can now be expressed as

$$X[m] = \sum_{n=0}^{N-1} x[n] \cos\left(\frac{2\pi mn}{N}\right) - j \sum_{n=0}^{N-1} x[n] \sin\left(\frac{2\pi mn}{N}\right). \quad (3.6)$$

As can be seen, DFT now consist of a real part and an imaginary part for every frequency number m . The magnitude and phase spectra can be expressed as

$$\begin{aligned} |X[m]| &= \sqrt{\operatorname{Re}(X[m])^2 + \operatorname{Im}(X[m])^2}, \\ \theta(m) &= \arctan \frac{\operatorname{Im}(X[m])}{\operatorname{Re}(X[m])}, \end{aligned} \quad (3.7)$$

where

$$\begin{aligned} \operatorname{Re}[m] &= \sum_{n=0}^{N-1} x[n] \cos\left(\frac{2\pi mn}{N}\right), \\ \operatorname{Im}[m] &= -\sum_{n=0}^{N-1} x[n] \sin\left(\frac{2\pi mn}{N}\right). \end{aligned} \quad (3.8)$$

The following MATLAB program can be used to compute magnitude and phase spectra using DFT. Alternatively, these can also be computed efficiently using the built-in *fft* function. It should be noted that array indexing for MATLAB starts at 1 and not 0 as in other languages such as C.

```
%Computation of magnitude and phase spectra using DFT
%Input the values for N and x

for m=1:N,
    sumn=0;
    for n=1:N,
        sumn=sumn + x(n)*(cos (2*pi*(m-1)*(n-1)/N));
    end
    dftreal(m)=sumn;
end

for m=1:N,
    sumn=0;
    for n=1:N,
        sumn=sumn + x(n)*(sin (2*pi*(m-1)*(n-1)/N));
    end
    dftimag(m)=-sumn;
end

for m=1:N,
    dftmag(m)=sqrt(dftreal(m)*dftreal(m)+dftimag(m)*dftimag(m));
    dftphase(m)=atan(dftimag(m)/dftreal(m));
end
```

Consider the following example of central processing unit (CPU) usage of a computer in a typical working day. Table 3.1 shows the usage starting from midnight.

Time (hours)	0	2	4	6	8	10	12	14	16	18	20	22
Usage (%)	5	3	4	8	60	80	25	75	60	90	15	20

Table 3.1: CPU usage

Figure 3.4 shows the plot of the data in Table 3.1 obtained using MATLAB codes:

```
time=[0 2 4 6 8 10 12 14 16 18 20 22];
usage=[5 3 4 8 60 80 25 75 60 90 15 20];
plot(time,usage,'+-');
axis([0 24 0 100]);
title('CPU Usage Data');
xlabel('Time (h)');
ylabel('Usage (%)');
```

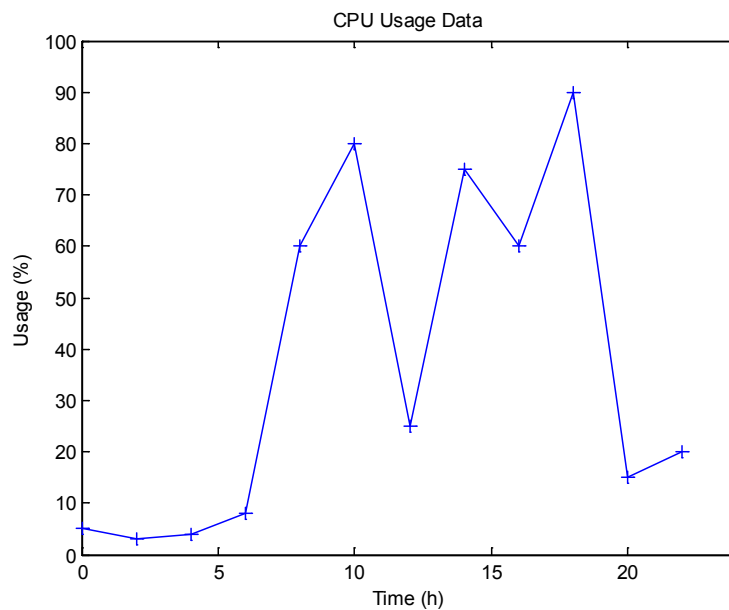


Figure 3.4: CPU usage data.

Next, using the DFT program given earlier, we obtain the real and imaginary parts of DFT for $m=0, \dots, 11$ and plots for these are given using the MATLAB codes below. Assume $F_s=12$ (since we have 12 readings for a day).

```
plot(0:11,dftreal,'+-');
hold on;
plot(0:11,dftimag,'r*-');
xlabel('Frequency number (m) ');
title('Real and imaginary DFT parts');
```

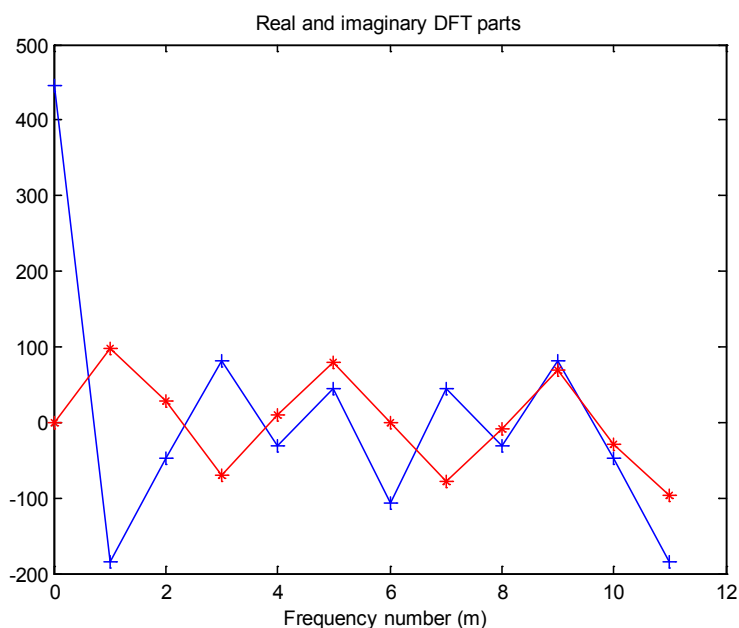


Figure 3.5: Real (in blue) and imaginary (in red) DFT values for CPU usage data.

The point of symmetry is at frequency number 5 for the real DFT values. It should be obvious that DFT imaginary values are conjugate symmetric at this point of symmetry.



Join the best at
the Maastricht University
School of Business and
Economics!

Top master's programmes

- 33rd place Financial Times worldwide ranking: MSc International Business
- 1st place: MSc International Business
- 1st place: MSc Financial Economics
- 2nd place: MSc Management of Learning
- 2nd place: MSc Economics
- 2nd place: MSc Econometrics and Operations Research
- 2nd place: MSc Global Supply Chain Management and Change

Sources: Keuzegids Master ranking 2013; Elsevier 'Beste Studies' ranking 2012; Financial Times Global Masters in Management ranking 2012

Maastricht University is the best specialist university in the Netherlands (Elsevier)

Visit us and find out why we are the best!
Master's Open Day: 22 February 2014

www.mastersopenday.nl



Figure 3.6 shows the magnitude and phase spectra obtained using MATLAB codes:

```
plot(0:6,dftmag(1:7),'+-');
xlabel('Scale is in m but note: frequency in cycles/day is also m');
ylabel('Magnitude');
title('CPU usage magnitude spectrum');
figure;
plot(0:6,dftphase(1:7),'r*-');
xlabel('Scale is in m but note: frequency in cycles/day is also m');
ylabel('Phase (rad)');
title('CPU usage phase spectrum');
```

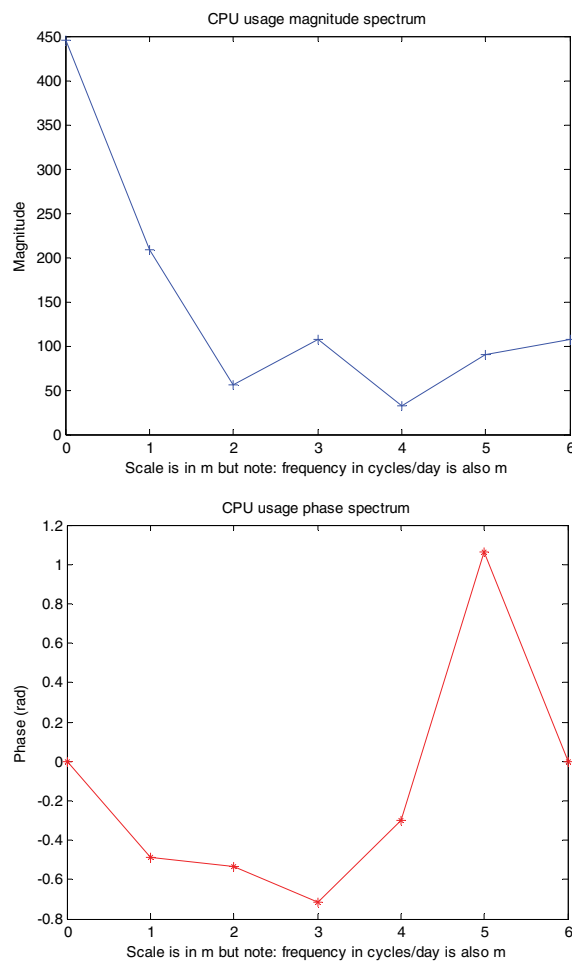


Figure 3.6: CPU usage spectra.

Frequently, the magnitude spectrum is the output that we want from DFT; it measures the *strength* of the signal at the specific frequencies. As explained earlier due to symmetry, magnitude spectrum is normally plotted from $f=0$ to $F_s/2$ with $N/2+1$ points, i.e. with F_s/N frequency spacing. In the example, $F_s=12$, $N=12$, so the actual abscissa (frequency scale) also ranges from 0,1,2,3,4,5,6 cycles/day.

3.3 DFT computation using matrix relation

There is another method to compute DFT using matrix relation [2]:

$$\mathbf{X} = D_N \mathbf{x}, \tag{3.9}$$

where \mathbf{X} is the vector composed of the N DFT samples and D_N is the N by N DFT matrix given by

$$\mathbf{D}_N = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & W_N^1 & W_N^2 & \dots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & \dots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \dots & W_N^{(N-1)(N-1)} \end{bmatrix}, \tag{3.10}$$

where

$$W_N = e^{-j2\pi/N} \text{ and } X_{DFT}[m] = \sum_{n=0}^{N-1} x[n]W_N^{mn}, \quad 0 \leq m \leq N-1. \tag{3.11}$$

For example, compute the DFT for $x=[9 \ 8 \ 3 \ 4]$ using matrix relation method. Using polar coordinates to get W_N :

$$W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}, \text{ hence we have } \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 9 \\ 8 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 24 \\ 6-j4 \\ 0 \\ 6+j4 \end{bmatrix}.$$

3.4 Picket fence effect

One of the problems with DFT is that it provides values of $X[m]$ only at a specific set of frequencies. What if an important value exists at the frequency, $f \neq m$? In our example, we have spectral values for frequencies 0,1,2,3,4,5,6 but what if there was an important cycle at frequency = 1.5? This problem is known as the picket fence effect and is due to discretising the frequency.

The solution is to increase the signal size (length) by zero padding as this will increase N and hence reduce the frequency spacing (i.e. hypothetically increase the resolution of DFT). For example, assume we pad 12 zeros to the CPU usage data and compute the spectra as earlier:

$$\text{usage} = [\text{usage}, \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0].$$

The results are shown in Figure 3.7.

Now $N=24$ and frequency spacing $=Fs/N=12/24=0.5$. From the figure, it can be seen that the spectral values at 0,1,2,3,4,5,6 has not changed but there are values for the in-between frequencies. So, there are frequency values for 0,0.5,1.0,1.5,2.0,.....,6.0.

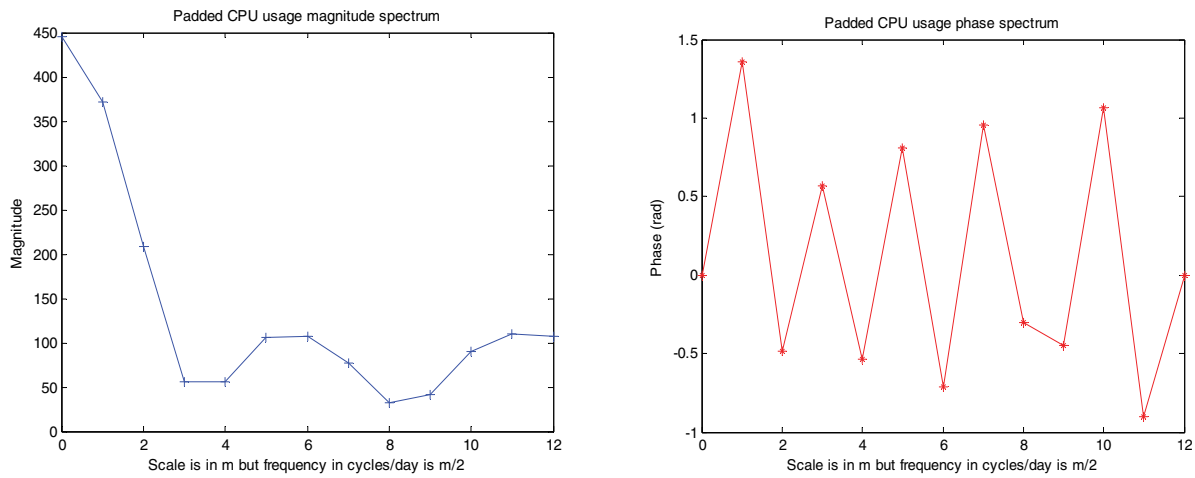



Figure 3.7: Padded CPU usage spectra.

It is important to note that zero padding isn't magic and does not actually improve the resolution of the DFT associated with the original signal, the only way to do that would be to provide a longer signal (i.e. more samples). In this example, as we were interested in an accurate value for frequency = 1.5, we had to double N .



BI

- Business
- Strategic Marketing Management
- International Business
- Leadership & Organisational Psychology
- Shipping Management
- Financial Economics

BI NORWEGIAN BUSINESS SCHOOL

EFMD **EQUIS ACCREDITED**

Empowering People. Improving Business.

BI Norwegian Business School is one of Europe's largest business schools welcoming more than 20,000 students. Our programmes provide a stimulating and multi-cultural learning environment with an international outlook ultimately providing students with professional skills to meet the increasing needs of businesses.

BI offers four different two-year, full-time Master of Science (MSc) programmes that are taught entirely in English and have been designed to provide professional skills to meet the increasing need of businesses. The MSc programmes provide a stimulating and multi-cultural learning environment to give you the best platform to launch into your career.

- MSc in Business
- MSc in Financial Economics
- MSc in Strategic Marketing Management
- MSc in Leadership and Organisational Psychology

www.bi.edu/master

 [Click on the ad to read more](#)

3.5 Effects of truncation

When we compute DFT, we assume that the signal is one complete cycle extracted from an infinite signal that is *periodic* and this is a basic assumption of DFT, i.e. we truncate the signal (though in actual fact, we do not). If the signal was indeed periodic, it would have the same starting and ending points after truncation. But for the CPU usage example, we would have discontinuity, so this creates a lot of noise, which is technically known as spectral leakage and can be solved using *windowing*.

In the CPU data example shown in Figure 3.8, the discontinuity causes spectral leakage and windows such as given in Equations¹³ 3.12 to 3.15 [3] can be used to *force* the signal values at the beginning and end to be the same. For most windows, this value is zero though not always.

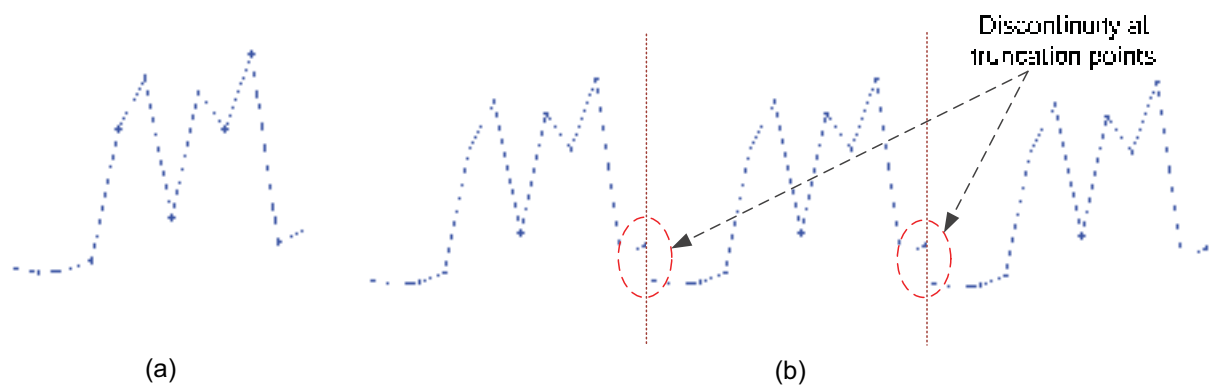


Figure 3.8: (a) Actual CPU usage signal (b) assumption that the signal is obtained from longer periodic signal through truncation.

```
%Blackman spectral window
N=101; %window size/order
plot(blackman(N), 'r-'); xlabel('n'); ylabel('w(n)');
title('Blackman window'); axis([0 105 0 1]);
```

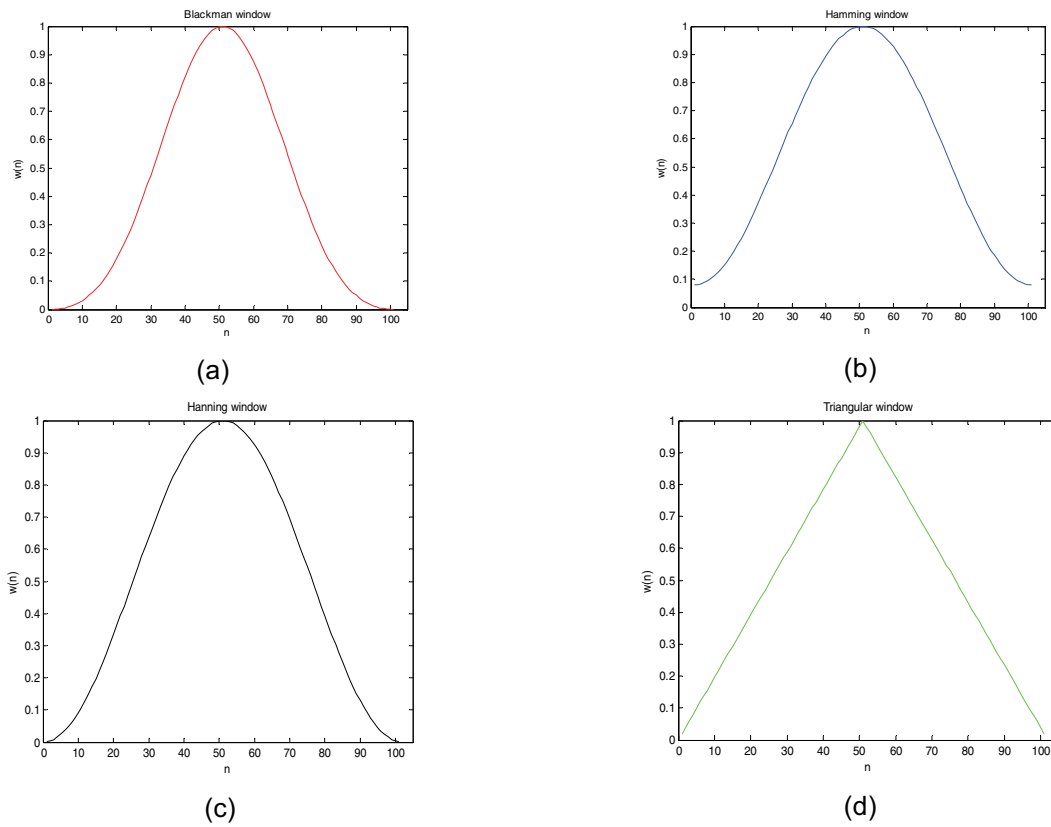


Figure 3.9: Examples of common spectral windows: (a) Blackman (b) Hamming (c) Hanning (d)Triangular.

$$w[n]_{\text{triangular}} = 1 - \frac{2 \left| n - \frac{N-1}{2} \right|}{N-1}, \quad 0 \leq n \leq N-1; \quad (3.12)$$

$$w[n]_{\text{blackman}} = 0.42 - 0.5 \cos\left(2\pi \frac{n}{N-1}\right) + 0.08 \cos\left(4\pi \frac{n}{N-1}\right), \quad 0 \leq n \leq N-1; \quad (3.13)$$

$$w[n]_{\text{hanning}} = 0.5 \left(1 - \cos\left(2\pi \frac{n}{N-1}\right) \right), \quad 0 \leq n \leq N-1; \quad (3.14)$$

$$w[n]_{\text{hamming}} = 0.54 - 0.46 \cos\left(2\pi \frac{n}{N-1}\right), \quad 0 \leq n \leq N-1. \quad (3.15)$$

The new windowed signal for every point n is $y[n]=x[n].w[n]$. For example, let us apply the Hamming window to the ECG signal discussed in Chapter 1 and compute the magnitude spectrum (shown in Figure 3.10). It can be seen that the use of windowing gives a smoother spectrum (i.e. with less noise). The choice of windowing is outside the scope of this book as it involves ratio analysis of spectral magnitudes of main lobes and side lobes of the window and hence, we'll skip them.

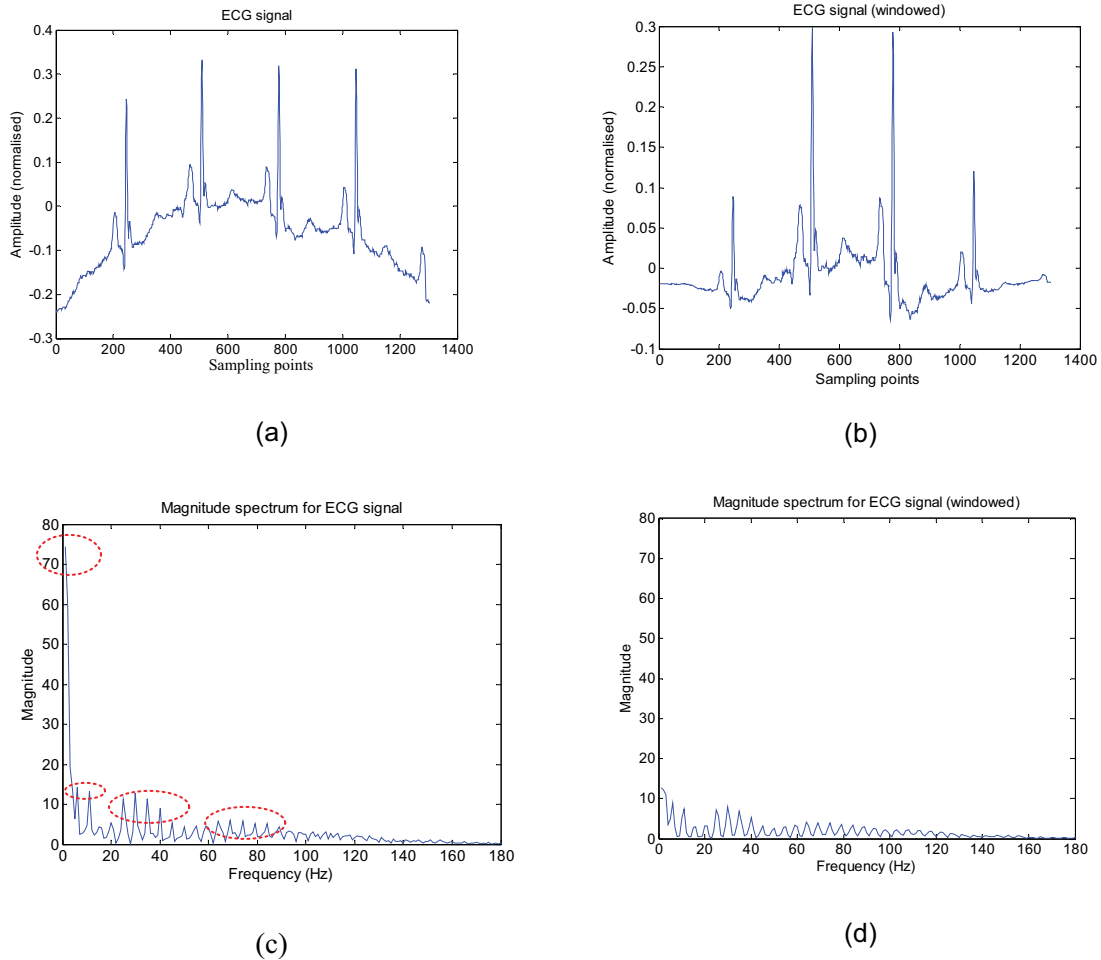


Figure 3.10: Example of spectral leakage: (a) Original ECG signal (b) windowed ECG signal (c) magnitude spectrum for the signal in (a) with noise from spectral leakage denoted with red ellipses (d) magnitude spectrum for the signal in (b).

3.6 Examples of using DFT to compute magnitude spectrum

Example 1

Consider a sinusoidal wave with frequency of 10 Hz generated with $F_s=200$ Hz using the MATLAB codes:

```
N=200; Fs=200;
x(1:N)=sin(2*pi*(0:N-1)*10/Fs); %Note, MATLAB index starts from 1
```

The magnitude spectrum is computed for this signal and is shown in Figure 3.11.

```
plot(0:N-1,x(1:N)); %Note, MATLAB index starts from 1
xlabel('Sampling points')
ylabel('Amplitude')
title('10 Hz sine wave (N=200)')
figure;
plot(0:N/2,dftmag(1:N/2+1)); %Note, MATLAB index starts from 1
xlabel('Frequency number (m)')
ylabel('Magnitude')
title('Magnitude spectrum of a 10 Hz sine wave (N=200)')
```

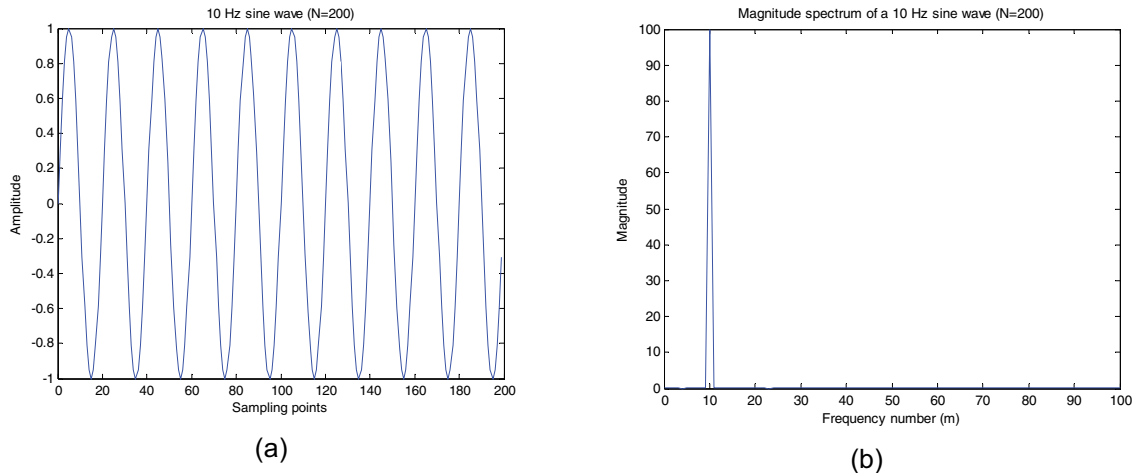


Figure 3.11: A 10 Hz sine wave (a) for length $N=200$ (b) spectral magnitude.

Now let us assume that the signal is available only up to $N=20$. We obtain the magnitude spectrum as shown in Figure 3.12. What has happened? The peak is at frequency number, $m=1$ which still corresponds correctly to 10 Hz as the actual frequency is given by mFs/N .

Need help with your dissertation?

Get in-depth feedback & advice from experts in your topic area. Find out what you can do to improve the quality of your dissertation!

[Get Help Now](#)



Go to www.helpmyassignment.co.uk for more info

 **Helpmyassignment**

 [Click on the ad to read more](#)

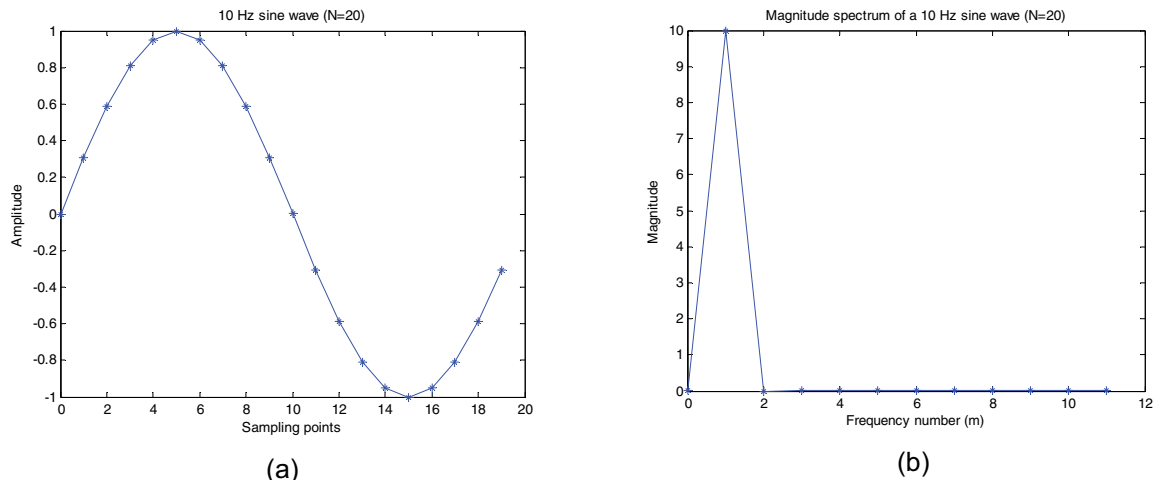


Figure 3.12: A 10 Hz sine wave (a) for length $N=20$ (b) spectral magnitude.

Example 2

Consider a combination of sinusoidal waves with frequencies of 5 Hz and 20 Hz with $F_s=200$ Hz:

```
N=20; fs=200; f1=5; f2=20; A=1; B=1;
x(1:N)=A*sin(2*pi*(0:N-1)*f1/fs)+B*sin(2*pi*(0:N-1)*f2/fs);
plot(0:N-1,x(1:N));
xlabel('Sampling points')
ylabel('Amplitude')
title('Combination of two sine waves (N=20)')
```

The magnitude spectrum is shown in Figure 3.13. As the frequency spacing is 10 Hz¹⁴, to avoid the picket fence effect, we pad the signal with 20 zeros and compute the spectrum. The magnitude spectrum shows a peak at 20 Hz and there is a value at 5 Hz but with a magnitude different to 20 Hz, so there is a problem as both the generated signals are of the same amplitude (i.e. $A=B=1$).

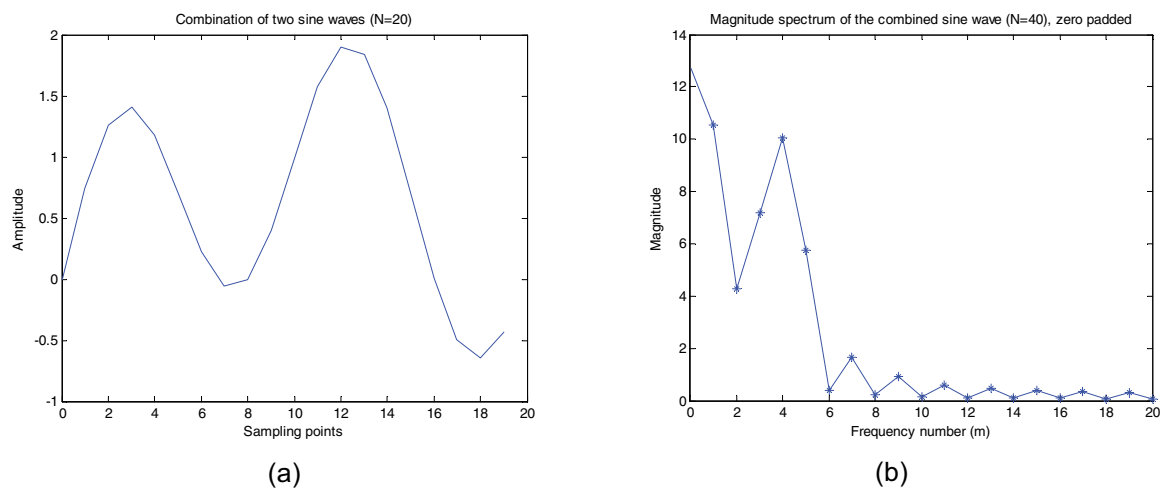


Figure 3.13: Combined sine wave (a) for length $N=20$ (b) spectral magnitude with 20 padded zeros.

DFT generated spectrum will work properly only if there is at least one complete cycle of the signal. Hence, we need to use at least 40 points (since 5 Hz signal completes a cycle in 40 points with $F_s=200$ Hz). See Figure 3.14 that has been generated with $N=40$ which shows both the peaks correctly at frequency numbers 1 and 4. But, in reality, we need to include multiple cycles in order to obtain reliable DFT information.

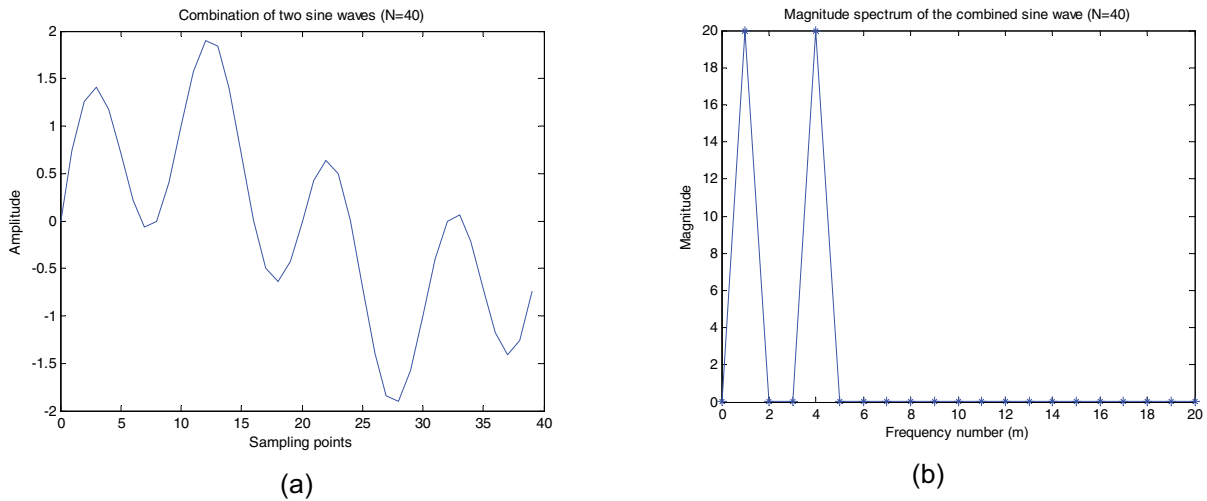


Figure 3.14: Combined sine wave (a) for length $N=40$; (b) spectral magnitude.

3.7 Periodogram

Periodogram is a measure of power spectrum and is similar to the square of magnitude spectra- actually, the only difference is the scale factor. If we compute magnitude spectra using DFT, square, scale and then convert to decibels (dB); we will have periodogram.

The following code can be used to obtain the periodogram of a sinusoidal wave:

```
N=200; Fs=100; f1=5;
x=sin(2*pi*(1:N)*f1/Fs);
plot(x);
y=fft(x,N);
py = power(abs(y),2);
pdgrm=10*log10(py/N); %normalise by N
plot(pdgrm(1:N/2));
```

Windowing is not necessary for this signal¹⁵ but when a window is used, the normalisation is no longer N but CN where C is given by

$$C = \frac{1}{N} \sum_{n=0}^{N-1} |w[n]|^2 \quad (3.16)$$

and $w[n]$ is the used window. This is to avoid any bias in the estimate due to the use of the window. The obtained spectrum is known as modified periodogram [2].

3.7.1 Welch method

Welch method is an improved method of computing the periodogram. The first step in this method is to divide the data into K overlapping segments each containing M sample points. The non-overlap length is D , i.e. the segments overlap with $M-D$ samples. Thus $K=(N-M)/D + 1$ and the percentage overlap is $100*(M-D)/M$.

Next, we apply the conventional method to compute periodogram (or modified periodogram) for each segment and finally average all the periodograms. In MATLAB, *pwelch* function can be used where the percentage overlap is 50% with $K=8$ and Hamming window is used by default.

As an example, let us revisit the ECG signal used earlier. The first plot is obtained by using the periodogram code given earlier, while the second is obtained by using *pwelch* function. From the plots, we can see that the Welch method gives a smoother power spectrum than conventional periodogram. A note about the *abscissa* scale when using *pwelch* function in MATLAB: the frequency scale is normalised from 0 to 1 representing the range 0 to $F_s/2$.

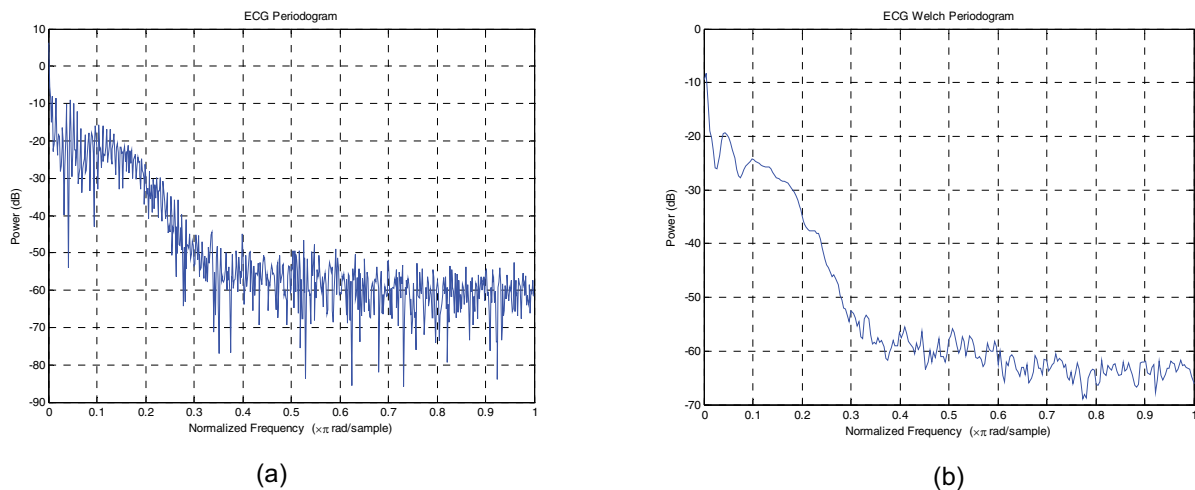


Figure 3.15: Power spectrum for ECG signal (a) using periodogram; (b) using Welch periodogram.

3.8 References

- [1] R. Shiavi, *Introduction to Applied Statistical Signal Analysis (2nd edition)*, Academic Press, 1999.
- [2] S.K. Mitra, *Digital Signal Processing: A Practical Approach (3rd edition)*, McGraw-Hill, 2006.
- [3] J.G. Proakis and D.K. Manolakis, *Digital Signal Processing (4th edition)*, Prentice Hall, 2006.