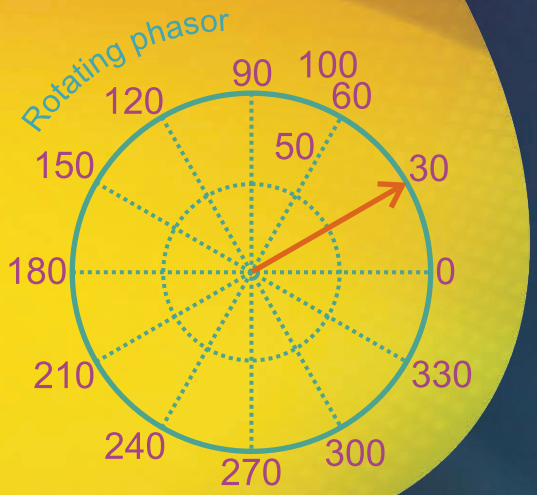
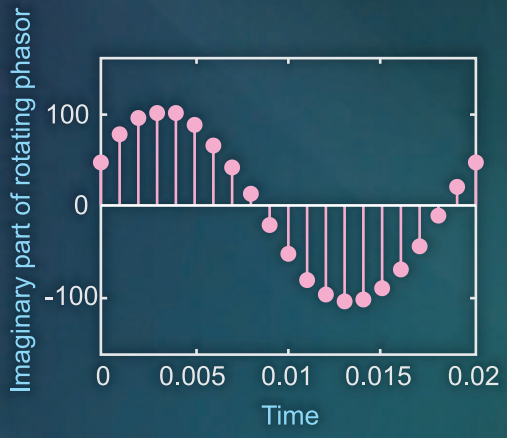
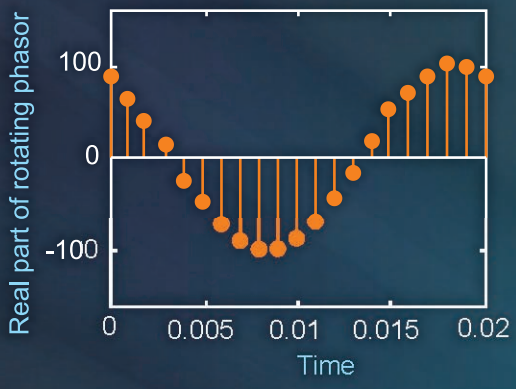






Eastern Economy Edition

DIGITAL POWER SYSTEM PROTECTION

S.R. Bhide

PHI

Digital Power System Protection

S.R. Bhide
Associate Professor of Electrical Engineering
Visvesvaraya National Institute of Technology
Nagpur

PHI Learning Private Limited
Delhi 110092
2014

DIGITAL POWER SYSTEM PROTECTION
S.R. Bhide

ISBN-978-81-203-4979-7

Published by Asoke K. Ghosh, PHI Learning Private Limited, Rimjhim House, 111, Patparganj Industrial Estate, Delhi-110092 and Printed by Star Print-O-Bind, F-31, Okhla Industrial Area Phase I, New Delhi-110020.

To
the memomy of **My Parents**
Pushpa and Ramkrishna Bhide

Contents

Preface

On the one hand the electronic technology is changing at a breath taking speed. The relentless march of Moor's law is making it not only possible to put computing power into every imaginable existing gadget but is also giving birth to entirely new gadgets and gizmos. Today's 'note-book' puts yesterday's supercomputer to shame. The internet and the digital computer have become ubiquitous. Interestingly, advances in the VLSI technology have also made the digital computer largely invisible by embedding it deeply into the innards of the system that it is controlling. You have digital computers embedded in every imaginable object.

Thus, the current trend is to replace the mechanical parts with electronic hardware and the electronic hardware, with software. It is no wonder, therefore, that the field of power system protection has also been caught up in the throes of the digital revolution. The result is that, the protective relay has been completely transformed into an 'intelligent electronic device (IED)'.

On the other hand, equally dramatic changes have taken place in the general power system scene. The electric utility business has been deregulated in large parts of the world. Environmental concerns are giving rise to the proliferation of non-conventional energy sources like wind, solar, bio-mass etc. There is more variety in the sources of power generation today than a decade back, leading to challenges and opportunities for the protection engineer.

All these factors make the present times very exciting and fascinating for the practicing protection engineers. However, for the new comer, to put it mildly, it becomes confusing and chaotic. This is because an appreciation of three major disciplines is required before one can make sense of all the digital brouhaha. Firstly, one must never lose sight of the underlying electrical principles. The various entities that are being protected; the generators, bus-bars, transmission lines and transformers have their own idiosyncrasies, i.e., unique set of responses and behaviours that set them apart from every other entity. These must first be thoroughly understood. Secondly, since the implementation of all the ideas is through the digital signal processing technology, one needs thorough appreciation of the art and science of DSP at the level of algorithms. Thirdly, we need a 'machine'; the 'DSP microprocessor' to practically

implement the algorithms. Thus, appreciation of this 'physical' layer of digital technology is equally important.

It is not possible to do justice to all the three areas mentioned above in a text like this. We have mainly attempted to introduce the reader to the 'middle-ware', i.e., various digital algorithms which are at the heart of the digital protective relays.

The text is aimed at the final year undergraduate and postgraduate students of Electrical Engineering who have already undergone courses on *Power System Protection*, *Signals and Systems* and *Microprocessors*. A course on *Numerical Methods* will be a definite advantage.

The practicing engineers, who routinely install, operate and maintain these systems, will also find the text useful in deepening their insight into the ideas that make the digital relays 'intelligent'.

The author feels indebted to all the researchers, inventors and pioneers who have enriched the state of the art by their contributions and made this field extremely fascinating.

The author wishes to thank his teacher and guide Prof. Y.G. Paithankar for introducing this fascinating field to him in the early 1990's. His love for the subject, enthusiasm for learning, his simplicity and pure humanism has been very inspiring. The author will always cherish the memories of his association with Dr. Paithankar.

The author would like to gratefully acknowledge the affection and encouragement received from all his colleagues at the Electrical Engineering Department, VNIT, Nagpur.

Last but not the least, the numerous students who have taken the courses on DSP Applications to Power Systems and Digital Protection, over the years, deserve special thanks for being catalysts in enriching authors' understanding of the subject and enhancing his patience.

S.R. BHIDE

1

Evolution of Power System Protection and the Emergence of Digital Relaying

1.1 Evolution of OC Relay from an Ammeter

The discipline of power system protection is, now, more than 100 years old. In the early days, circuit breakers were tripped, by directly making use of the current carried by them through a pickup coil. Thus, the concept of an entity called '*relay*', separate from the circuit breaker was yet to be evolved. The mechanism for tripping the circuit breaker was based on force developed, due to build-up of current to a large value, such as that encountered during fault. However, it was soon realised that it would help, if the setting of current above which the circuit breaker tripped is adjustable at will. Thus, what was needed was a trip decision based on precise measurement of current magnitude. Hence it was natural to evolve the protective over-current relay from an ammeter as shown in Figure 1.1.

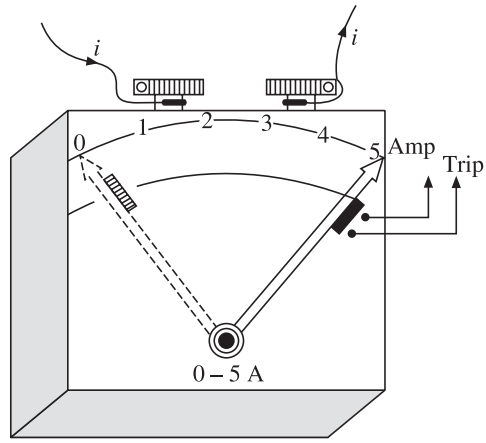


Figure 1.1 Evolution of an ammeter into an over-current relay.

The technology used for making ammeters, at that time was based on force experienced by a current carrying coil placed in a magnetic field. We call this the 'electromechanical' technology which made measurements through the balancing of deflecting torque against a restoring or restraining torque. The restraining torque could be developed by a spring, by gravity or by another current etc.

1.2 Evolution of a Directional Relay from a Wattmeter

We can trace the evolution of directional relay from wattmeter. Figure 1.2 shows that a wattmeter endowed with tripping contacts just beyond zero in the positive direction. If the wattmeter current coil and pressure coil are fed with relay current and voltage signals, it can be used as a directional relay with maximum torque angle (MTA) of zero degrees.

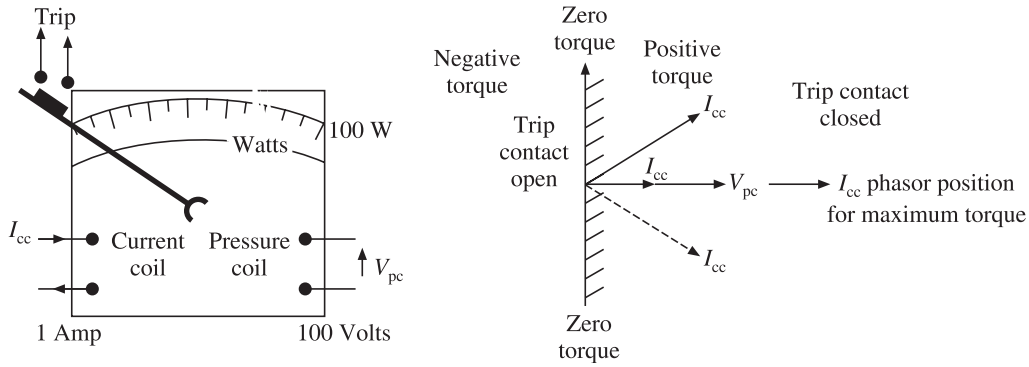


Figure 1.2 Evolution of directional relay.

It is easy to see that electromechanical measuring instruments evolved into their new avatar of dedicated and very sensitive and precision mechanisms called *protective relays*. Somewhere along the evolutionary path, the indicating needles were dropped. It is interesting to note that modern numerical trelays do store the instantaneous values of voltage and current and can display them on request of the user. Thus, the measuring instrument from which the relay has evolved is still there. This reinforces the idea that modern relays can be looked up on as measuring instruments endowed with numerical computational and logical decision making capabilities.

1.3 The Birth of Transistor

The transistor, invented in 1947 by **William Shockley**, **John Bardeen** and **Walter Brattain**, (for which all three were awarded the **Nobel Prize** for Physics in 1956) in the Bell telephone laboratories, brought phenomenal changes in electronics, signal processing, measurement and almost all of electrical engineering. Protective relaying field was no exception to this and within a year of the invention of the transistor, researchers started reporting relays built around the solid-state transistor. But it took long time for this technology to mature since the early

electronic devices were not as reliable as their present day counterparts. Further, user experience and confidence in the new technology was lacking. It was only during the early 1960's that solid-state analog technology was accepted by the industry.

Even though digital computers were around right from the early 1940's, they were very bulky, in fact monstrous by today's standards, consumed astounding amounts of power and lacked speed for real time applications demanded by a protective relay. Not only that, they were so expensive that only big corporations and government defence establishments could own them. In those days, it was unthinkable to use a digital computer in a relay. This is borne out by the fact that early research papers on use of digital computer for protective relaying go to great lengths to justify their use and almost sound apologetic about using digital computers for such a mundane task as protective relaying!

1.4 Start of a Revolution

The year 1972 saw the birth of the microprocessor which was CPU on a single chip of silicon. This opened the door to many unthinkable applications of computer to the relaying field. In fact, it was microprocessor which spawned the current information technology revolution which is still playing out at an unrestricted pace. One major change that happened with the introduction of the microprocessor technology, however, was that the element of 'software' was added to a product which was earlier purely an item of 'hardware'. In fact, it is the software which makes the modern protective relay so protean. One can now implement any protection concept that one can think of, provided one can write suitable software to do it. The hardware part of the digital relay has become much more standardised than the software. It is the software that distinguishes one vendor's protective relay from another. This change has also been quite slow and for many years even though microprocessors were used in the relays, they were doing peripheral mundane things while the time critical filtering and measurement functions were still handled by analog devices. But as the microprocessors evolved their architectures to cater the signal processing tasks, it became possible to handover the time critical signal processing tasks to DSP microprocessors bringing in an era of truly numerical relays. In the modern numerical relay, once the analog signals are sampled, and numbers which represent the instantaneous values of voltages and currents are generated by the ADC, every subsequent operation till issuing of trip signal is done through software numerically. Thus even though the use of microprocessor in relays is reported from the late 1970's, it was only during 1986 that the true numerical relay has emerged.

In addition to the ability of quickly disconnect a faulty element of the power system from the rest of the system using CT/PT in conjunction with the relays and circuit breakers, there are many routine things which need to be attended to. For example, consider a 100 MVA transformer in a substation. We need to do the following functions in addition to protection:

1. Display the status of the circuit breakers on either side of the transformer on a control panel
2. Metering of energy, active and reactive power, peak demand and power quality
3. Annunciation, logging and flagging
4. Archival of data

5. Keeping count of the number of circuit breaker tripping to facilitate CB maintenance
6. Keep a check on the health of the trip batteries by monitoring their voltage
7. Monitor and display the tap changer position
8. Communication with substation computer

Such other functions which are distinct from the protection per se, are called 'control and automation functions'. Much of the data used for control and automation is common with that used for protection. Further, the computational processing requirement of the 'control' functions is very small as compared to protection and can easily be accommodated in relay hardware and software. Thus, it was very natural to merge the 'control and automation' functionality with the 'protection' functionality giving rise to what is known as a 'numerical terminal' dedicated to the protection,control, automation and monitoring of a particular element of the power system. Presently vendors are offering such fully numerical terminals for protection,control, automation and monitoring of transformers, transmission lines bus-bars, motors, etc.

1.5 Architecture of the Modern Digital Relay

Hierarchy of protection can be loosely compared with the seven layers of communication defined in the OSI model (Open Systems Interconnection) adopted by the International Standards Organisation as shown in Table 1.1.

Table 1.1 Seven Layers of communication in the OSI model

Application layer
Presentation layer
Session layer
Transport layer
Network layer
Data link layer
Physical layer

On similar lines, we can conceptualise layers of protective relaying functions where different kinds of activities take place. At the lowest level is, of course,the interface between the high voltage, high current power system and the low voltage and low current measurement system made possible by the current transformers and the potential transformers. This is followed by layers of electronic hardware to convert the voltages and currents into numbers representing their instantaneous values. Once we have the numbers available in the memory, the software takes over and we are in the realm of algorithms. This conceptual scheme is shown in Table 1.2.

Table 1.2 Seven layers of protection

Algorithm for control, automation and monitoring functions
Algorithm for implementation of relay characteristics or protection scheme
Algorithm for extraction of information from raw samples
Analog to digital conversion
Sampling and holding
Filtering signal conditioning
CT and PT

In this book, we will mainly focus on the top 3 layers with special emphasis on algorithms for extraction of information useful for relaying purposes.

2

Digital Signal Processing Basics and Architecture of Numerical Relay

2.1 Introduction to Digital Signal Processing

We live in an analog world. Temperature, pressure, velocity, speed, mass, volume, voltage, current; are all analog quantities. Thus, it would be natural to process them in the analog domain. However, even though analog signal processing was extensively used in the 1950's, 1960's and early 1970's, it has been superseded by digital signal processing using digital methods. This has come about because digital computers have proved to be much more powerful, economical and flexible. The reason for this digital revolution is that the number of transistors that can be fabricated on a silicon chip is doubling every 18 months in accordance with Moore's law. It is no exaggeration to say that the computing power that was available only to big government defence establishments some 20 years back is now in the hands of a school boy in the form of a smart-phone!

The digital computers are controlled by software which is also continuously evolving and exhibiting exponential growth. All these factors rule out analog computers. Thus, it is inevitable that digital computers are finding their use in every imaginable application from dish-washer to DTH receiver. We are now in a situation where signal processing is entirely dominated by the digital method. Thus, on the one hand, all our activities are in the analog domain and on the other hand, all the computation, control and monitoring is to be done using the digital computer as depicted in Figure 2.1.

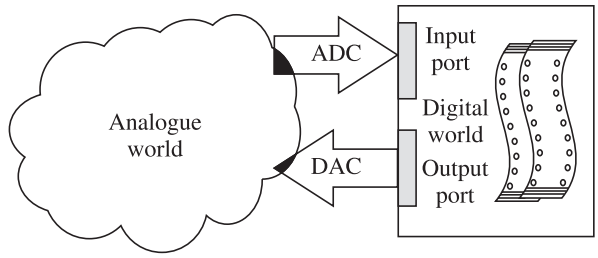


Figure 2.1 The two worlds.

2.2 The DSP Signal Processing Chain

The computer, with which we are going to process the power system signals, only understands digital signals. Hence, we must first convert the analog signal into digital signal using a sub-system known as analog to digital converter (ADC). These digital signals are stored as numbers representing the instantaneous values of various voltages and currents. The digital signal processing sub-system essentially works on these numbers using a mathematical procedure or algorithm and produces a digital output. In some applications like processing of audio signals we again convert these digital signals into analog form using a digital to analog converter (DAC), say, for feeding to a speaker or a headphone. However in the digital relaying field we are not interested in getting back the signal in analog form. Hence, the DAC is absent from the digital relaying hardware. It is shown in Figure 2.2 for the sake of showing the complete classical DSP signal chain. The digital relay issues a trip/restrain signal, which is essentially a binary or digital signal. This arrangement is shown in Figure 2.2.

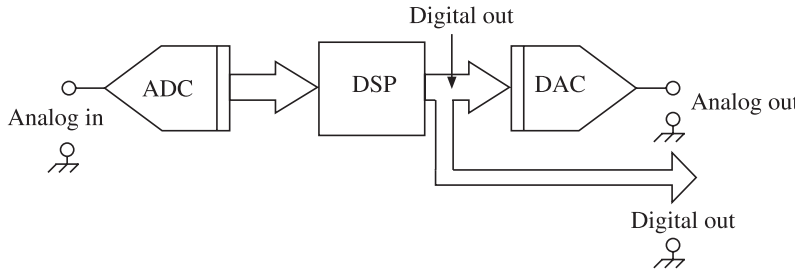


Figure 2.2 The classical DSP signal chain

2.3 Analog to Digital Converters

Thus, it can be seen that the ADC forms a very crucial link between the analog and the digital worlds. Therefore, the choice of the appropriate type of ADC will decide the overall performance of the digital relay. ADC's can be implemented using a variety of techniques. Each technique of A to D conversion has its pros and cons. However, it is possible to talk about the ADC at a functional level, without going into the details of its working. Figure 2.3(a) shows a bipolar ADC with '*N*' bits in the output. The ADC can accommodate signal between $-V_m$ and $+V_m$ volts and produces an '*N*' bit digital code corresponding to the input analog voltage.

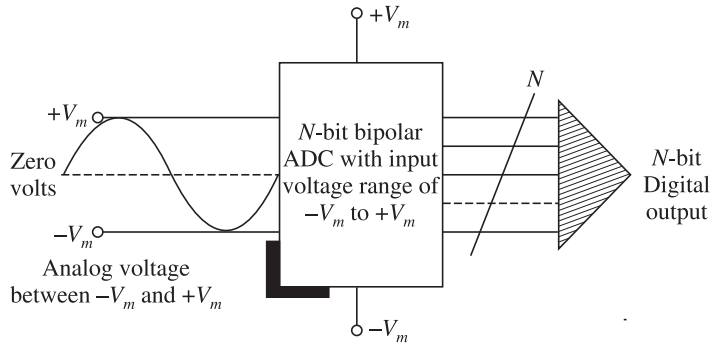


Figure 2.3(a) Input and output of a *N*-bit bipolar ADC.

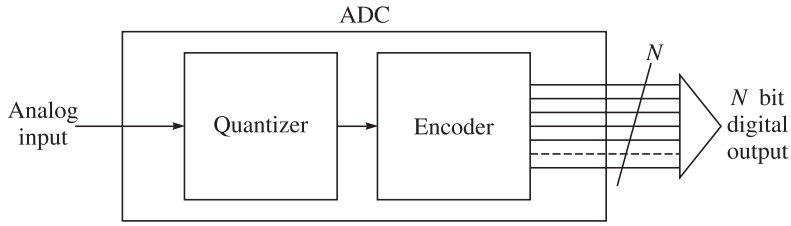
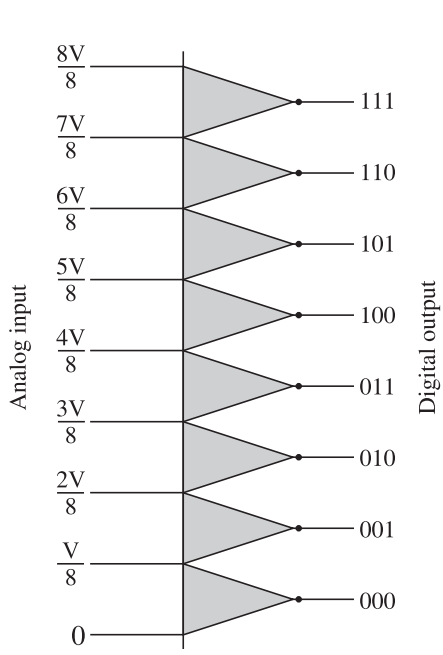


Figure 2.3(b) Functional diagram of an ADC.

2.3.1 Quantization Error

The ADC consists of a quantizer followed by an encoder (or code converter) as shown in the block diagram of Figure 2.3(b). The quantizer receives a sample of the analog signal and converts it into its digital version. The input analog signal can take infinite number of values while the output digital signal can take only 2^N number of values, for an N-bit ADC. This, process of *quantization*, which essentially maps an infinite space into a finite space, is shown in Figure 2.4. Quantization leads to a definite loss of information, leading to an unavoidable error, known as *quantization error*.



Analog input		Digital output code (3-bit)		
From	To	D_2	D_1	D_0
0	V/8	0	0	0
V/8	2V/8	0	0	1
2V/8	3V/8	0	1	0
3V/8	4V/8	0	1	1
4V/8	5V/8	1	0	0
5V/8	6V/8	1	0	1
6V/8	7V/8	1	1	0
7V/8	8V/8	1	1	1

Figure 2.4 Quantization in a unipolar 3-bit ADC.

It can be seen from Figure 2.4 that for a 3-bit ADC with an input voltage range of 0 to V volts, all voltages in the range 0 to (V/8) are assigned the same code 000. All voltages in the range from (V/8) to (2V/8) are assigned the same code 001, and so on. Thus, the resolution of the ADC is (V/8). As the input voltage changes from 0 to slightly more than (V/8) volts,

the code changes from 000 to 001. Thus, the LSB corresponds to a change of (V/8) volts. The quantum of (V/8) volts is therefore called 1 LSB, and the resolution of the ADC is said to be 1 LSB. Alternatively, ADC resolution is expressed simply as number of bits in the output.

Let us indicate the total excursion in input voltage that the ADC can handle by V_{span} where V_{span} can be written as:

$$V_{span} = V_{in,\ maxmim} - V_{in,\ minimum}$$

Magnitude of 1 LSB for an N-bit bipolar ADC with an input span of V_{span} is given by:

$$1\,\text{LSB}_{bipolar} = \frac{V_{span}}{2^N} = \frac{(V_m - (-V_m))}{2^N} = \frac{2V_m}{2^N}$$

Note that for a bipolar ADC, $V_{in\ ,\ minimum} = -V_m$ while for a unipolar ADC, $V_{in,\ minimum}$ = zero.

Let us assume $V_{in,\ maximum}$ for both ADCs as $+V_m$. Then, the span of a bipolar ADC will be $2V_m$ while that of unipolar ADC will be V_m.

If we apply a ramp to the input of the ADC, the output of the ADC will be a quantized digital output. If we feed this to a DAC, it will generate a staircase type waveform as shown in Figure 2.5. We define the quantization error of the ADC as:

Quantization error = output of ADC (as seen through DAC) – analog input

Note that even an ideal ADC having no other imperfections cannot escape from quantization error. We can only make the quantization error smaller and smaller by increasing the number of bits in the ADC, but we can never completely eliminate it.

We can plot the error signal waveform as shown in Figure 2.5. The error in the output

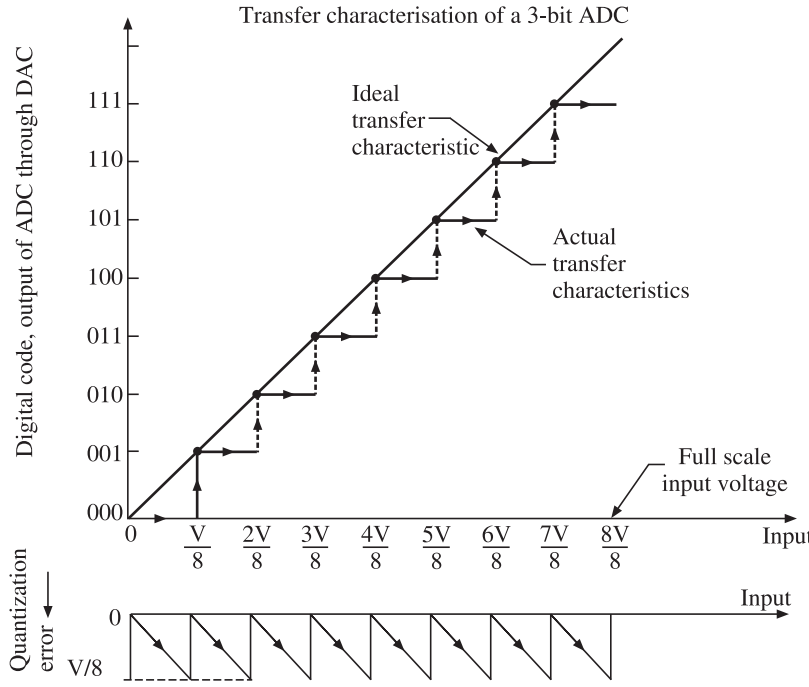


Figure 2.5 Transfer characteristics of a 3-bit ADC and quantization error.

of the ADC can be considered as a noise signal attributable to the quantization phenomenon. Hence, it is customary to quantify the error due to quantization in terms of signal to noise ratio defined as ratio of signal voltage to noise voltage usually expressed in decibels.

Figure 2.5 shows that the actual transfer curve of the ADC, always hovers on the lower side of the ideal transfer curve. Statistically it will be more appropriate if the actual transfer curve symmetrically encompasses the ideal transfer curve. This can be easily achieved by adding a ½ LSB offset to all the analog levels at which the code makes transition as shown in Figure 2.6. This results in the error curve also being symmetric about the *x*-axis.

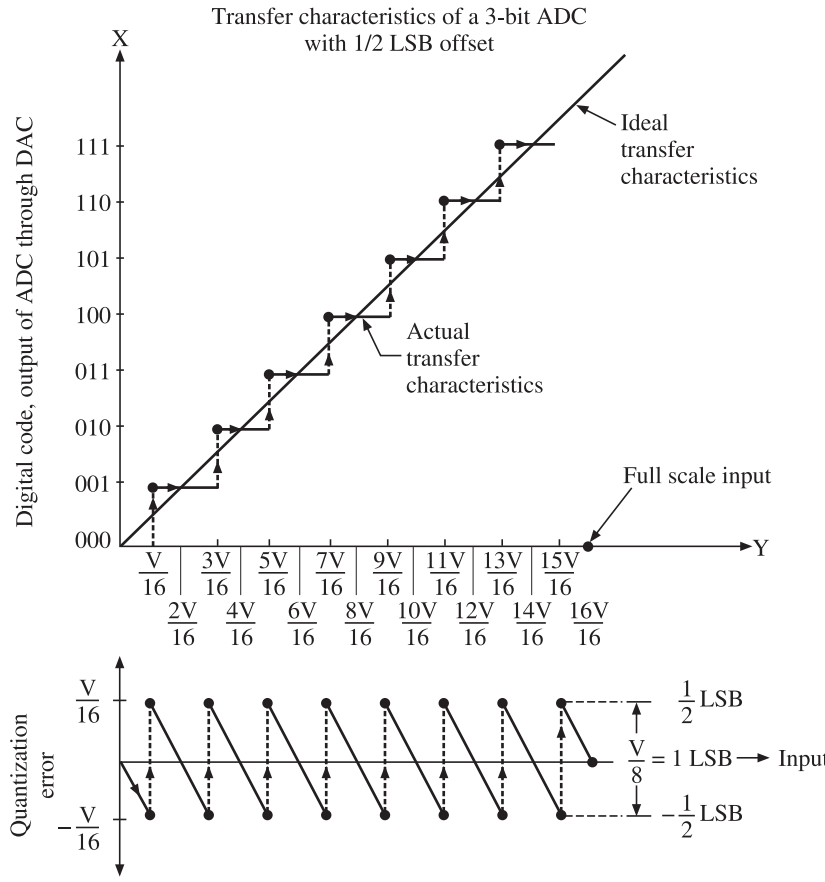


Figure 2.6 Transfer characteristics of a 3-bit ADC with half LSB offset and quantization error.

We are now in a position to express the quantization error in terms of signal to noise ratio. Let us assume that a sinusoidal wave with a peak value of V_m is applied to the ADC. The RMS value of the signal is thus $V_m/\sqrt{2}$. Note that the signal excursion is from $-V_m$ to $+V_m$, which is a change of $2V_m$ requiring an ADC with a span of $2V_m$. Thus, the ADC is representing a voltage range of $2V_m$ using *N*-bits.

The RMS value of the saw-tooth wave, generated because of quantization phenomenon, which is considered as noise can be written as:

$$V_{\text{noise}} = \sqrt{\frac{1}{T}\int_{-T/2}^{T/2} v_e^2 dt}$$

where v_e, the error signal, having the saw-tooth waveform, is given by:
$V_e = (-V_{\text{HLSB}}/(T/2))t$, where T is the time period of the saw-tooth error waveform and V_{HLSB} is the magnitude of voltage corresponding to ½ LSB given by:

$$V_{\text{HLSB}} = \frac{1}{2}\text{ LSB} = \frac{1}{2}\frac{2V_m}{2^N} = \frac{V_m}{2^N}$$

As noted earlier, the value of $2V_m$ in the numerator of above equation is due to the fact that a sine wave with a peak of $|V_m|$ needs a bipolar ADC which can accommodate input voltage excursion from $-V_m$ to $+V_m$ which is actually an excursion of $2V_m$.
Hence, we can write:

$$V_{\text{noise}} = \sqrt{\frac{1}{T}\int_{-T/2}^{T/2}\left(\frac{-V_{\text{HSLB}}}{(T/2)}t\right)^2 dt}$$

$$V_{\text{noise}} = \frac{V_{\text{HSLB}}}{T/2}\sqrt{\frac{1}{T}\int_{-T/2}^{T/2}(t)^2 dt}$$

$$V_{\text{noise}} = \frac{V_{\text{HSLB}}}{T/2}\sqrt{\frac{1}{T}\left[\frac{t^3}{3}\right]_{-T/2}^{T/2}} = 2\frac{V_{\text{HLSB}}}{\sqrt{12}}$$

Hence,

$$V_{\text{noise, RMS}} = 2\frac{V_m}{2^N\sqrt{12}} = \frac{V_m}{2^{(N-1)}\sqrt{12}}$$

We can therefore write signal to noise ratio (SNR) as:

$$\text{SNR} = \frac{V_{\text{signal,RMS}}}{V_{\text{noise,RMS}}}$$

We have, $V_{\text{signal,RMS}} = \frac{V_m}{\sqrt{2}}$ and $V_{\text{noise,RMS}} = \frac{V_m}{2^{(N-1)}\sqrt{12}}$

Hence, SNR is obtained as:

$$\text{SNR} = \frac{V_m}{\sqrt{2}}\frac{2^{(N-1)}\sqrt{12}}{V_m}$$

$$\text{SNR} = 2^{N-1}\sqrt{6}$$

Let us express it in decibels:

$$\text{SNR}_{\text{dB}} = 20\log_{10}\left(\frac{V_{\text{signal,RMS}}}{V_{\text{noise,RMS}}}\right)$$

$$\mathrm{SNR_{dB}} = 20\log_{10}\left(2^{N-1}\sqrt{6}\right)$$

$$\mathrm{SNR_{dB}} = 20(N-1)\log_{10} 2 + 20\log_{10}\left(\sqrt{6}\right)$$

$$\mathrm{SNR_{dB}} = 20\ N\log_{10} 2 - 20\log_{10} 2 + 20\log_{10}\left(\sqrt{6}\right)$$

$$\mathrm{SNR_{dB}} = 6.02N - 6.02 + 7.78$$

$$\mathrm{SNR_{dB}} = \{6.02\,N + 1.76\}\mathrm{dB}$$

However, when sampling is done above the Nyquist sampling rate then noise gets spread over the frequency range from 0 to ($f_{\mathrm{sampling}}/2$) and a part of the noise gets pushed to frequencies above $f_{\mathrm{sig,\ max}}$. This decreases the 'in band' noise, improving the signal to noise ratio to:

$$\mathrm{SNR_{dB}} = 6.02N + 1.76 + 10\log_{10}\left(\frac{(f_{\mathrm{sampling}}/2)}{f_{\mathrm{signal,max}}}\right)\mathrm{dB}$$

Noting that $f_{\mathrm{sig,max}}$ is the bandwidth of the signal, and representing it by BW, we can write:

$$\mathrm{SNR_{dB}} = 6.02N + 1.76 + 10\log_{10}\left(\frac{(f_{\mathrm{sampling}}/2)}{\mathrm{BW}}\right)\mathrm{dB}$$

2.3.2 ADC Types

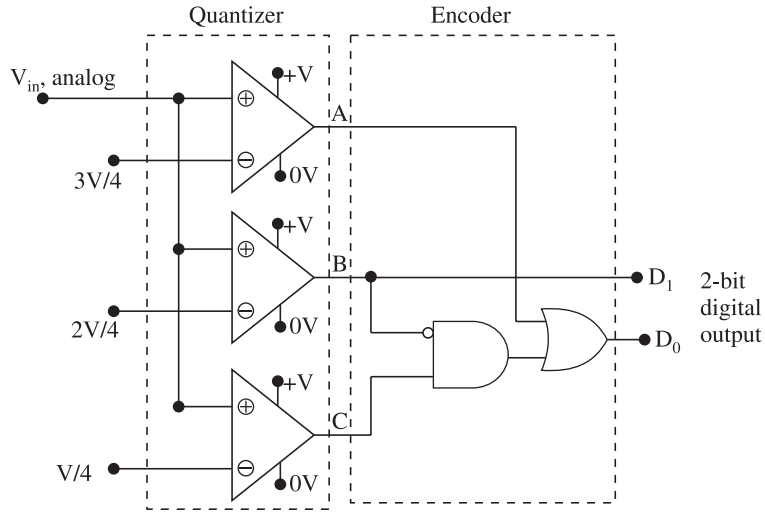
There are several types of ADCs. Each type of ADC has its advantages and disadvantages. Table 2.1 lists various ADC types with some pertinent remarks.

Table 2.1 Analog to digital converter types

	Type of ADC	Conversion speed	Remarks
1	Flash or parallel comparator	Fastest	N-bit ADC needs (2^N–1) number of comparators. Hence, becomes impractical when N is large. Suitable for real time applications like numerical relays.
2	Successive approximation	Intermediate	Moderate complexity. Can be built for high value of N. Used in numerical relays.
3	Dual slope	Slowest	Highly accurate but slow. Not suitable for real time applications. Used in bench top high precision voltmeters.
4	Sigma-delta	High	Trades sampling rate for number of bits. Uses noise shaping. Due to high sampling frequency, anti-aliasing filter design is simplified. Down sampling can be performed to reduce sampling rate at the output. Digital filtering can also be done to enhance signal. Used in audio applications.

The flash type ADC

A 2-bit 'flash' or 'parallel-comparator' type ADC is shown in Figure 2.7. It can be seen that the quantizer is based on 3 numbers of comparators. In general, it can be shown that a '*N*' bit flash converter requires $(2^N - 1)$ number of comparators. Hence, an 8-bit flash ADC would require $2^8 - 1 = 256 - 1 = 255$ comparators while a 10-bit flash comparator would require $(2^{10} - 1) = 1024 - 1 = 1023$ comparators. Such large number of comparators becomes unmanageable when fabricated on a chip because of imperfections of the analog components. Hence, flash converters become impractical for higher number of bits.



Input voltage	Quantizer output code			Desired digital code	
	A	B	C	D_1	D_0
0 to V/4	0	0	0	0	0
V/4 to 2V/4	0	0	1	0	1
2V/4 to 3V/4	0	1	1	1	0
3V/4 to 4V/4	1	1	1	1	1

K map for D_1 output

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$	0	X	X	X
C	0	1	1	X

$D_1 = B$

K map for D_0 output

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$	0	X	X	X
C	1	0	1	X

$D_0 = \bar{B}C + A$

Figure 2.7 A flash type 2-bit A to D converter.

The reason for calling it a 'flash' converter is that the conversion from analog to digital value in this topology is very fast since it involves only the switching time of the comparators which is very small. In fact all other methods of analog to digital conversion are slower than the 'flash' method. Since all the comparators work in parallel, it is also known as '*parallel comparator*' type ADC.

It can be seen from Figure 2.7 that the comparators have thresholds of $V/2^2$, $2V/2^2$, and $3V/2^2$, i.e., V/4, 2V/4 and 3V/4. Further, the output digital code generated by the quantizer is not a natural binary code. Thus, the ADC needs a 'code converter' or the so-called 'encoder'. The encoder is a simple combinatorial circuit. The encoder can be easily designed as shown in Figure 2.7 using K-map method.

2.4 Sampling

An ADC cannot handle a continuously changing analog signal. We must first sample the analog signal and then feed one sampled analog value to the ADC at a time. Figure 2.8(a) shows the sampling of an analog signal at 16 samples per cycle.

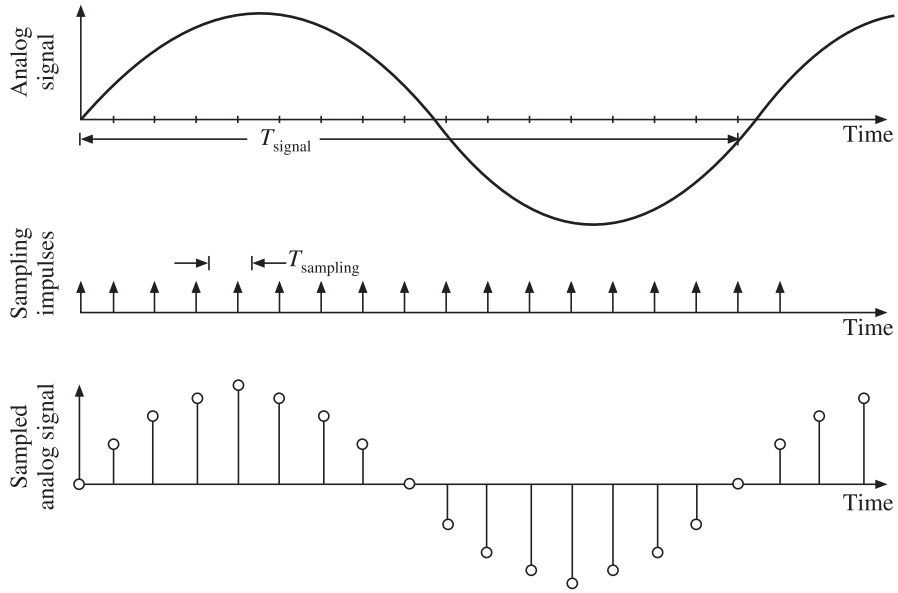


Figure 2.8(a) Analogue signal and its sampled version.

In the above discussion we had assumed an ideal ADC. However, a real-life ADC has many imperfections. One such important imperfection is that all ADCs require a finite amount of time to convert the input analog voltage into its digital value.

2.4.1 Need for Sample and Hold Circuit

Let us see what happens when we feed the analog signal directly to the ADC. In order to avoid error, we need to hold the analog voltage constant during the conversion time. Thus, we not only need to sample the input but also hold it constant. If the signal changes by more than +/- ½ LSB during the time it is being converted, it will cause erroneous output. Hence, we should only feed one unique analog voltage to the ADC by sampling the analog voltage at a sharply defined instant of time. We would tend to think that if we have a fast enough ADC,

this will solve the problem of finite conversion time of ADC. To illustrate how things get more complicated, consider Figure 2.8(b) where a unipolar ADC is assumed and three analog voltage waveforms with different frequencies but same peak value are shown. Using the rule that signal should not change by more than ½ LSB during conversion we see that for a given number of ADC bits and peak value of input signal, the frequency of the signal that can be handled without error goes down as we employ a *better ADC*. A better ADC is the one with a smaller conversion time and greater number of bits. We must therefore hold the input to the ADC constant for a time equal to conversion time of ADC.

Figure 2.8(b) shows three voltages of different frequencies LF, MF and HF having the same peak value. It can be seen the LF < MF < HF. In a time duration equal to conversion time T_c, each voltage changes by a different quantum. The quantum being highest for highest frequency signal since all signals are assumed to have same amplitude. The figure also shows the voltage corresponding to ½ LSB. For error free operation of ADC, the input voltage should not change by more than this value of ½ LSB during conversion time T_c. It can be seen that the condition that input voltage should not change by more than ½ LSB is satisfied only by the lowest frequency signal LF.

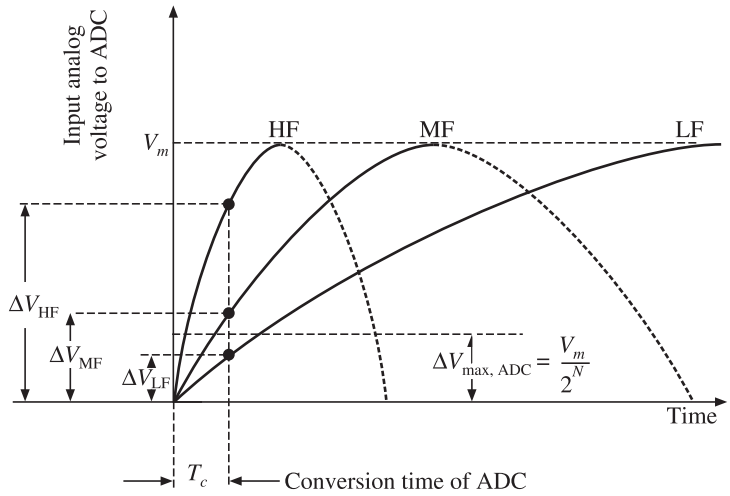


Figure 2.8(b) Need for sample and hold circuit.

Now if we employ a better ADC, i.e., an ADC with higher number of bits N and smaller conversion time T_c, the maximum frequency that can be handled goes down still further than LF. This can be seen from Figure 2.8(c).

Thus, we cannot improve upon the maximum frequency that can be handled by using ADC's with faster conversion time and more number of bits. This can be easily appreciated by finding out the maximum frequency that can be handled by an ADC with excellent specifications. Consider a 16-bit ADC with a conversion time of 10 μs. Let us find the maximum frequency that can be handled by an ADC with so good specifications.

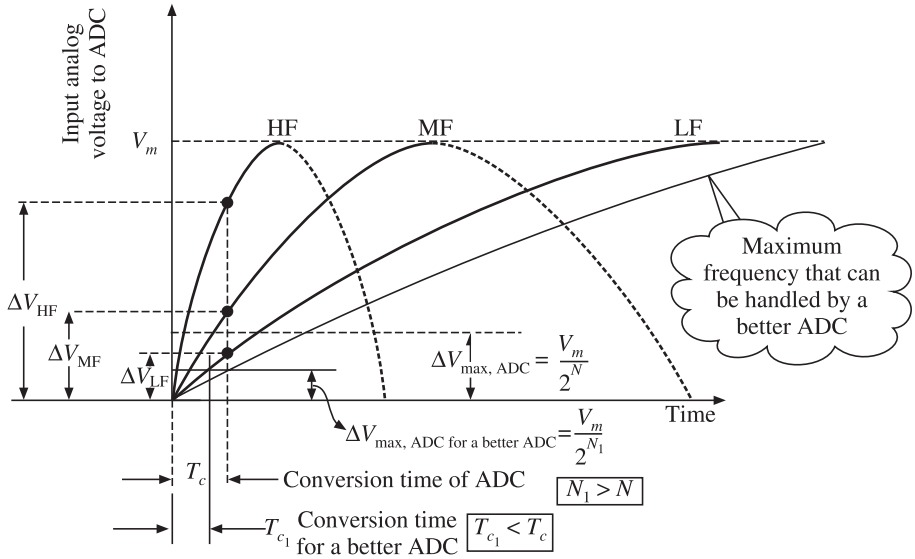


Figure 2.8(c) Maximum frequency goes down for a better ADC.

Let the signal be represented by

$$v = V_m \sin(\omega t)$$

$$v = V_m \sin(2\pi ft)$$

$$\frac{dv}{dt} = 2\pi f\, V_m \cos \omega t$$

$$\left(\frac{dv}{dt}\right)_{\text{signal,max}} = 2\pi f_{\max} V_m$$

$(dv/dt)_{\text{ADC,max}}$ i.e., the maximum rate of change of voltage that the ADC can accommodate is given by $(dv/dt)_{\text{ADC,max}} = (\Delta v/\Delta t)_{\text{ADC, max}}$where and $\Delta v = (V_m/2^N)$ and $\Delta t = T_c$

$$\left(\frac{dv}{dt}\right)_{\text{ADC,max}} = \frac{V_m}{2^N T_c}$$

Equating $(dv/dt)_{\text{signal,max}}$ and $(dv/dt)_{\text{ADC,max}}$ we get:

$$2\pi f_{\max} V_m = \frac{V_m}{2^N T_c}$$

Hence,

$$f_{\max} = \frac{1}{2^{(N+1)} \cdot \pi T_c}$$

For the ADC with given specifications of $N = 16$ and $T_c = 10\,\mu s$, we get

$$f_{max} = \frac{1}{2^{(16+1)}\pi(10)(10^{-6})}$$

$$f_{max} = 0.2428 \text{ Hz}$$

This is too low a frequency to be of any use in power system protection work! And the frequency will become still lower if we use a better ADC! Thus we have to think of a device in between the analog input signal and the ADC which will be able to cope up with the high (dv/dt) rates involved. Fortunately, a simple solution to this problem exists. If we use a 'sampler' followed by a 'hold' circuit, i.e., a 'sample and hold circuit (S/H)' then we can go for sampling much higher frequencies. This becomes possible because now it is the aperture time of the hold circuit, T_a, that has to meet the (dv/dt) requirement rather than the ADC. Fortunately, the aperture times of S/H circuits are many orders of magnitude smaller than the ADC conversion times. Thus, f_{max} with sample and hold can be obtained by substituting T_a in place of T_c in the expression for f_{max} derived earlier.

$$f_{max} = \frac{1}{2^{(N+1)}\pi T_c} \quad \ldots \textit{without hold circuit}$$

$$f_{max} = \frac{1}{2^{(N+1)}\pi T_a} \quad \ldots \textit{with hold circuit with aperture time } T_a$$

Let us rework the maximum frequency by assuming a realistic aperture time of 250 ps.

$$f_{max} = \frac{1}{2^{(16+1)}\pi(250)(10^{-12})} \quad \ldots \textit{with hold circuit}$$

$$f_{max} = 9.714 \text{ kHz}$$

Thus, we get improvement in maximum frequency from 0.2428 Hz to 9.714 kHz because of simple expedient of using a sample and hold circuit ahead of the ADC. We make suitable modification in the DSP signal chain as shown in Figure 2.9(a). Figure 2.9(b) shows an analog signal , the sampling impulses, the sampled signal and sampled-and-held versions of an analog signal. Note that and $f_{signal} = (1/T_{signal})$ and $f_{sampling} = (1/T_{sampling})$

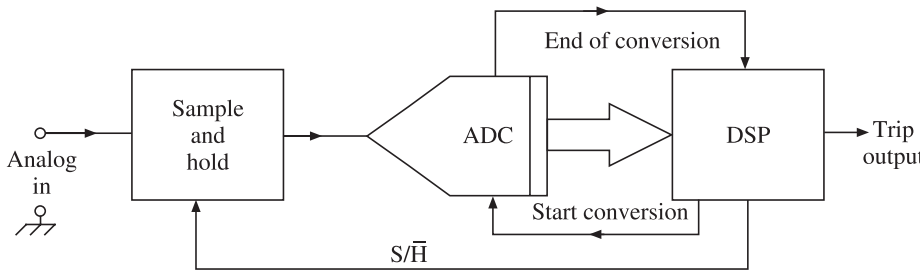


Figure 2.9(a) Modified DSP signal chain with sample and hold circuit included.

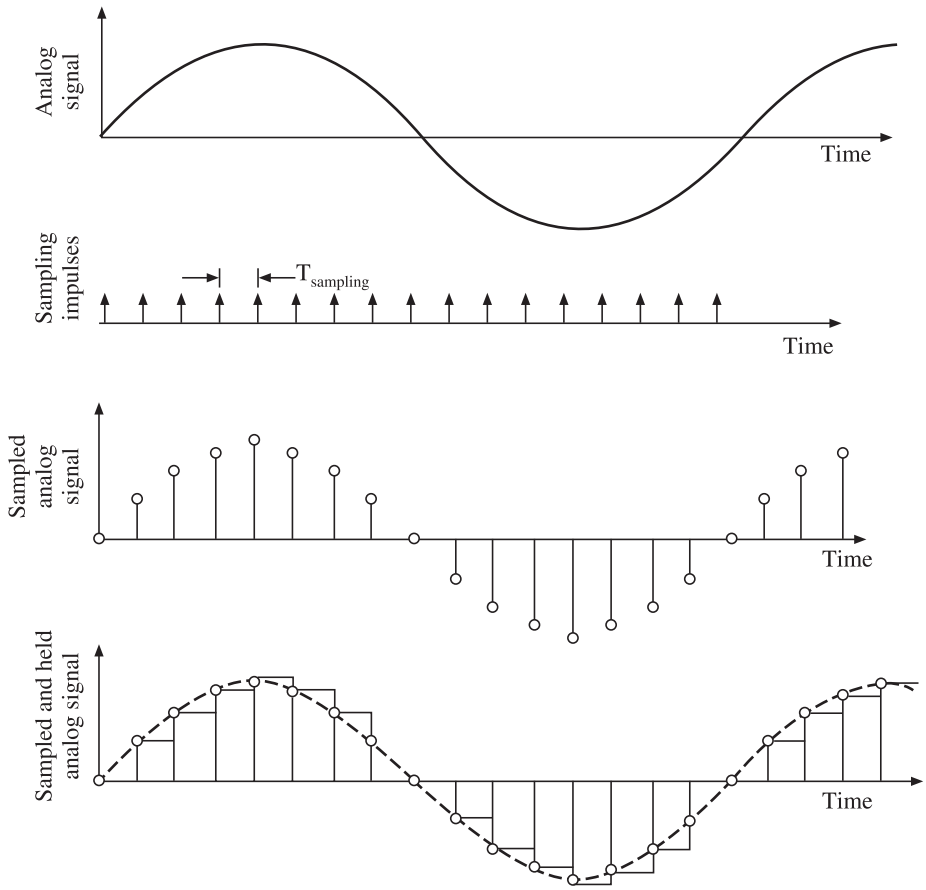


Figure 2.9(b) Sampled analog signal.

2.4.2 Shannon's Sampling Theorem and Aliasing

Note that we have to sample the analog signal before feeding it to the ADC. Thus, now, there are two frequencies in picture; the *signal frequency* and the *sampling frequency*. In real life, we may never encounter a signal with a very well defined single frequency. Real life signals tend to contain a band of frequencies from the lowest frequency of zero Hz, i.e., DC to some maximum frequency $f_{signal,max.}$ Hence, we talk about the signal bandwidth denoted by 'BW'. Thus, the question that naturally arises is: What should be the relationship between the signal bandwidth (or maximum frequency contained in the signal) and the sampling frequency? Here, the famous theorem known as Shannon's sampling theorem comes into play. According to the sampling theorem the minimum sampling frequency must be at least twice the maximum signal frequency, so that there is no loss of information. The sampling theorem can be stated as:

$\boxed{f_{sampling,minimum} = 2\, f_{signal}}$ if signal is a undistorted single frequency sinusoid

$f_{\text{sampling, minimum}} = 2\, f_{\text{signal, max}}$ if the signal happens to be broad-band

$f_{\text{sampling, minimum}} = 2\ \text{BW}$ where BW is the bandwidth = $f_{\text{signal, maximum}}$

The minimum sampling frequency stipulated by Shannon's sampling theorem is known as the Nyquist sampling rate.

We can therefore in general write that:

$$f_{\text{samping}} \geq 2\ \text{BW}$$

The sampling theorem is pictorially shown in Figure 2.10. It can be seen from Figure 2.10 that as dictated by the sampling theorem, there is a band of frequencies from zero to $2f_{\text{signal, max}}$ which are forbidden for the sampling frequency.

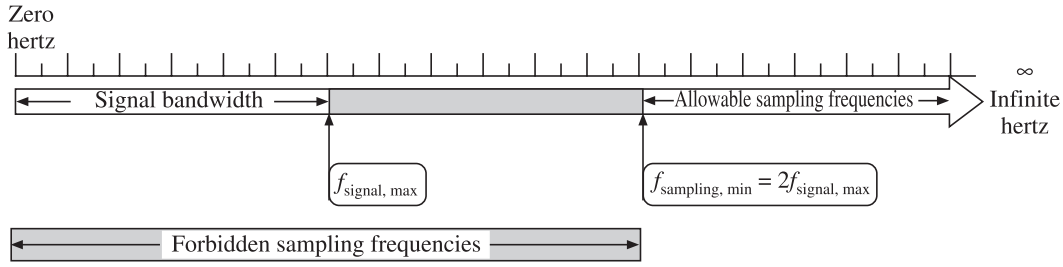


Figure 2.10 Shannon's sampling theorem.

This naturally leads to the next question: What happens if we violate the sampling theorem and sample at a forbidden frequency, i.e., at a frequency less than that stipulated by Shannon?

We encounter the phenomenon of aliasing in such an eventuality, wherein a higher frequency signal appears as a lower frequency signal. When aliasing takes place, we cannot recover correct information from the signal. Thus, aliasing is to be avoided at all costs.

Figure 2.11(a) shows the effect of violating the sampling theorem. It can be seen that the sampling is at rate less than twice the signal frequency since there are no samples during alternate halves of the input sine wave. The wave that will be recreated from the samples will be a low frequency *aliased* wave different from the input.

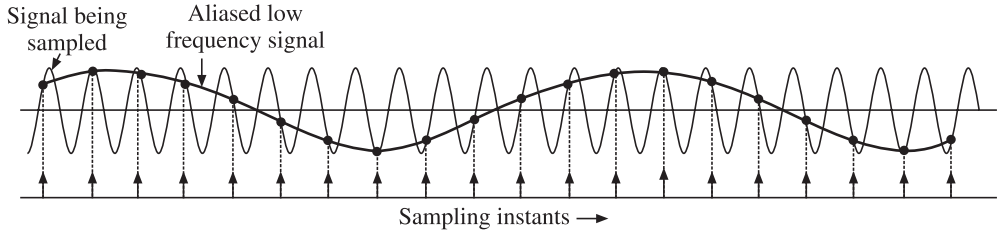


Figure 2.11(a) Aliasing.

Figure 2.11(b) shows a more specific example of a 50 Hz signal sampled at frequencies of 49 Hz ,51 Hz, 100 Hz and 500 Hz. This figure was generated using a small MATLAB script.

As expected, there is aliasing at sampling frequencies of 49 Hz, 51 Hz and even at 100 Hz. Sampling at 49 Hz and 51 Hz give an aliased signal of 1 Hz with phase angles of zero and 180° respectively. At 100 Hz, since the sampling is synchronised with the zero crossing of the waveform, we get a DC with zero magnitude as the aliased wave. However, at sampling frequency of 500 Hz we are able to correctly recreate a 50 Hz wave, since there are 50 complete oscillations in 1 second.

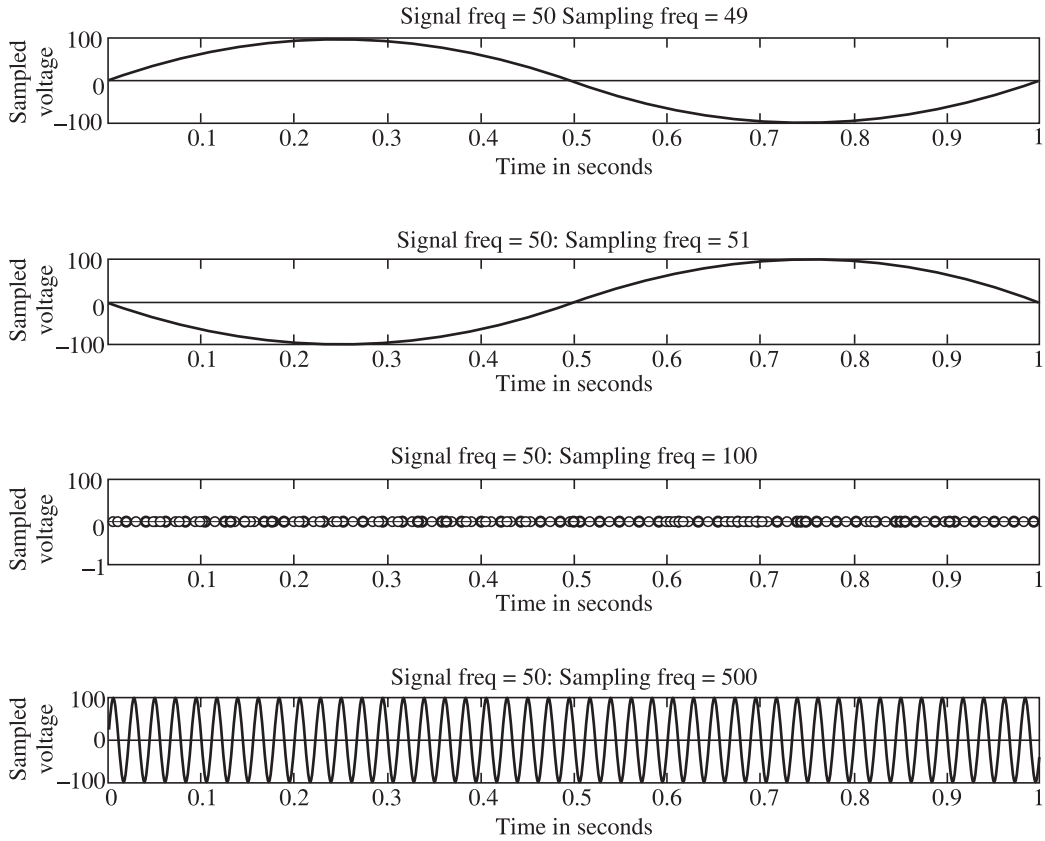


Fig. 2.11(b) A 50 Hz wave sampled at various frequencies.

In DSP based systems, sampling plays very important role as what we 'perceive' through the sampling process strongly depends upon the relation between the signal bandwidth and the sampling frequency. Figure 2.11(c) shows 'perceived' frequency Vs the actual input frequency for a fixed sampling frequency. It can be seen that only in the region where Shannon's sampling theorem is not violated, do we correctly 'perceive' the input frequency. Outside this range we actually either have no perception or are working in the illusory realm because of aliasing.

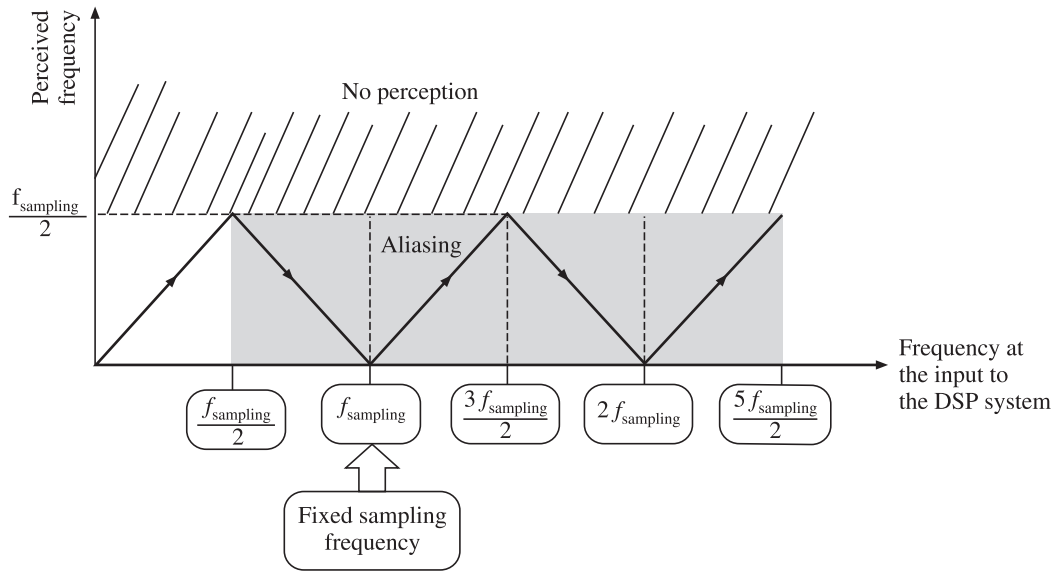


Fig. 2.11(c) Perceived frequency Vs input frequency for a fixed sampling frequency.

2.5 Anti-aliasing Filter

It would appear that in order to avoid aliasing, all that we need to do is to choose a sampling frequency which is greater than twice the maximum signal frequency. However in practise, it is found that a real life power system signal contains a number of unwanted high frequency components right into the tens or hundreds of megahertz range. Thus, we need to remove these high frequency components or at least attenuate them below ½ LSB magnitude before we sample the analog signal. In the absence of such a filter, the high frequency components will get aliased back below ½ $f_{sampling}$. Hence, the low-pass filter used for removing all frequency components above ½ $f_{sampling}$ is called '*anti-aliasing*' filter. The anti-aliasing filter has to be of analog type. Thus, the DSP signal chain gets further modified as shown in Figure 2.12(a).

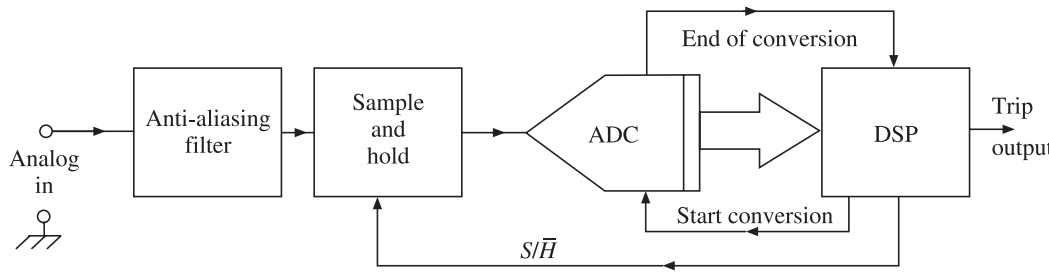


Figure 2.12(a) Addition of anti-aliasing filter to the DSP signal chain.

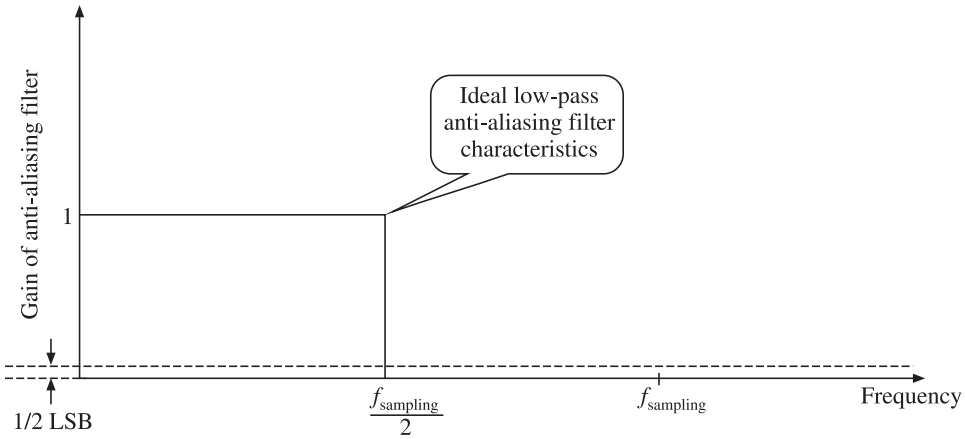


Figure 2.12(b) Characteristics of ideal low-pass filter characteristics.

Figure 2.12(b) shows the characteristics of an ideal anti-aliasing filter that will entirely filter out all frequency components above $f_{sampling}/2$. It can be seen that such an ideal anti-aliasing filter will have to have its cut-off frequency at $f_{sampling}/2$ and exhibit an infinite roll-off. However, all practical filters have finite roll-off. Hence in case of a practical anti-aliasing filter we will have to push-back the cut-off frequency so that the magnitude of the signal falls below ½ LSB in the stop band as shown in Figure 2.12(c).

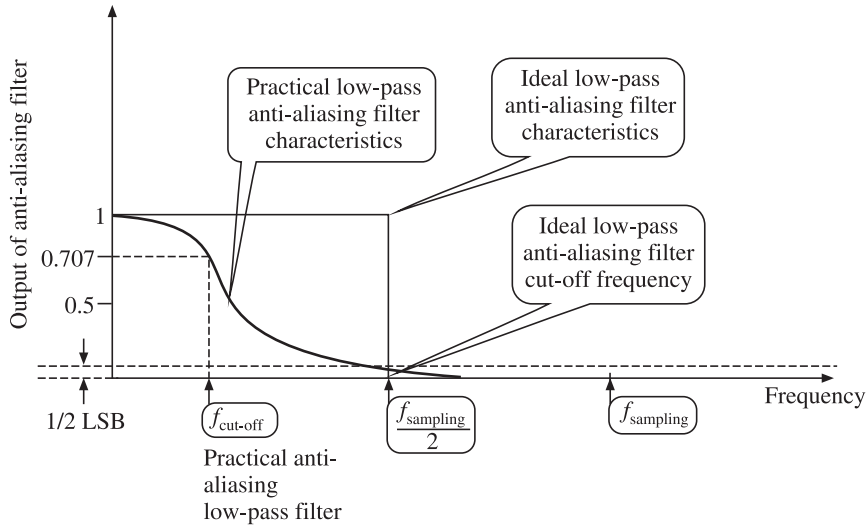


Figure 2.12(c) Practical low-pass anti-aliasing filter characteristics.

2.5.1 Design of Anti-aliasing Filter

Let us design a Butterworth type low-pass anti-aliasing filter, by noting that the gain of the n^{th} order low-pass Butterworth filter at any frequency, f_{actual}, is given by:

$$A(f_{\text{actual}}) = \frac{1}{\sqrt{1+\left(\frac{f_{\text{actual}}}{f_{\text{cut-off}}}\right)^{2n}}}$$

so that when $f_{\text{actual}} = f_{\text{cutoff}}$, the gain of the filter is $= 1/\sqrt{2} = 0.707$
For an N-bit, bipolar ADC, the magnitude corresponding to ½ LSB is $(1/2)(2V_m/2^N)$

To satisfy the requirement that signal magnitude should fall below ½ LSB beyond $½f_{\text{sampling}}$, the following inequality applies:

$$\frac{V_m}{\sqrt{1+\left(\frac{\frac{f_{\text{sampling}}}{2}}{f_{\text{cut-off}}}\right)^{2n}}} \le \frac{1}{2}\frac{2V_m}{2^N}$$

If we represent the ratio $[(f_{\text{sampling}}/2)/f_{\text{cut-off}}]$ by α, then we can write the above equation as:

$$\frac{V_m}{\sqrt{1+(\alpha)^{2n}}} \le \frac{1}{2}\frac{2V_m}{2^N}$$

Solving the above equation gives order of the low-pass filter as:

$$n \ge \frac{\log_e(4^N - 1)}{2\log_e(\alpha)}$$

2.5.2 Numerical Example

Let us explain the above concept with a numerical example.

EXAMPLE 2.1 Decide the order of a low-pass Butterworth type anti-aliasing filter to be used in a DSP system with the following specifications:

(i) Number of bits in the ADC = 12
(ii) Sampling frequency = 3000 Hz
(iii) Cut-off frequency of the anti-aliasing filter = 150 Hz

Solution

Given: $N = 12$

$$\alpha = \frac{\frac{f_{\text{sampling}}}{2}}{f_{\text{cut-off}}} = \frac{\frac{3000}{2}}{150} = \frac{1500}{150} = 10$$

$$n \ge \frac{\log_e(4^{12} - 1)}{2\log_e(10)} \ge (3.61 \text{ rounded to } 4)$$

Hence, 4th order low-pass anti-aliasing Butterworth filter should be used for the given system.

2.6 Functional Block Diagram of Numerical Relay

Typical number of input signals to a numerical relay in a three-phase power system is 6 viz., the three line to neutral voltages V_{an}, V_{bn}, V_{cn} and the three line currents I_a, I_b, I_c. Ideally therefore we should use six number of analog to digital converters to acquire all the data in parallel. However for reasons of economy we may sample all the six channels in parallel and hold the values before they are converted to digital. We can use a 6-into-1 analog multiplexer which is digitally addressable to achieve this. Further we need extensive signal conditioning before the signals from CTs and PTs reach the sensitive electronic circuits. Protective devices to bypass any high voltage surges to ground are required. Similarly, clamping circuits, to limit signal excursions to safe values, so that they do not exceed the input range of analog to digital converter are required as shown in Figure 2.13.

Numerical relays are continuously evolving further. Since the late 1980's, numerical relays are being endowed with synchrophasor measurement capabilities. This has become possible because of widespread and cheap availability of GPS (Geographical Positioning Satellite) technology. A numerical relay can now act as phasor measurement unit (PMU). Through a high speed optical fibre network data from various PMU's can be collected and collated by a "phasor data concentrator" in a hierarchical manner to implement "wide area protection" and "control schemes". Many of the routine monitoring and control functionalities in a substation can be easily integrated into the numerical protective relay. Thus, the protective relay is morphing in to an *intelligent electronic device* (IED). Some vendors are calling such devices as *protection, control and monitoring systems* rather than mere relays. Figure 2.14 illustrates a modern numerical relay with a variety of connectivities, inputs and outputs.

We conclude, by observing, that regardless of the apparent complexity of the modern numerical relay, once the signals are acquired and converted to digital domain, we are left with *numbers* which represent the instantaneous values of the signals. Beyond this point, the protective relaying software takes over. These *numbers* will now be processed as per the relaying algorithm which finally may come up with either a trip output or a no-trip (restrain) output. Note that this is a continuous and un-ending process. A relaying algorithm therefore works in an infinite loop which is typical of all embedded system software. In the sections that follow, we will go into the details of various relaying algorithms.

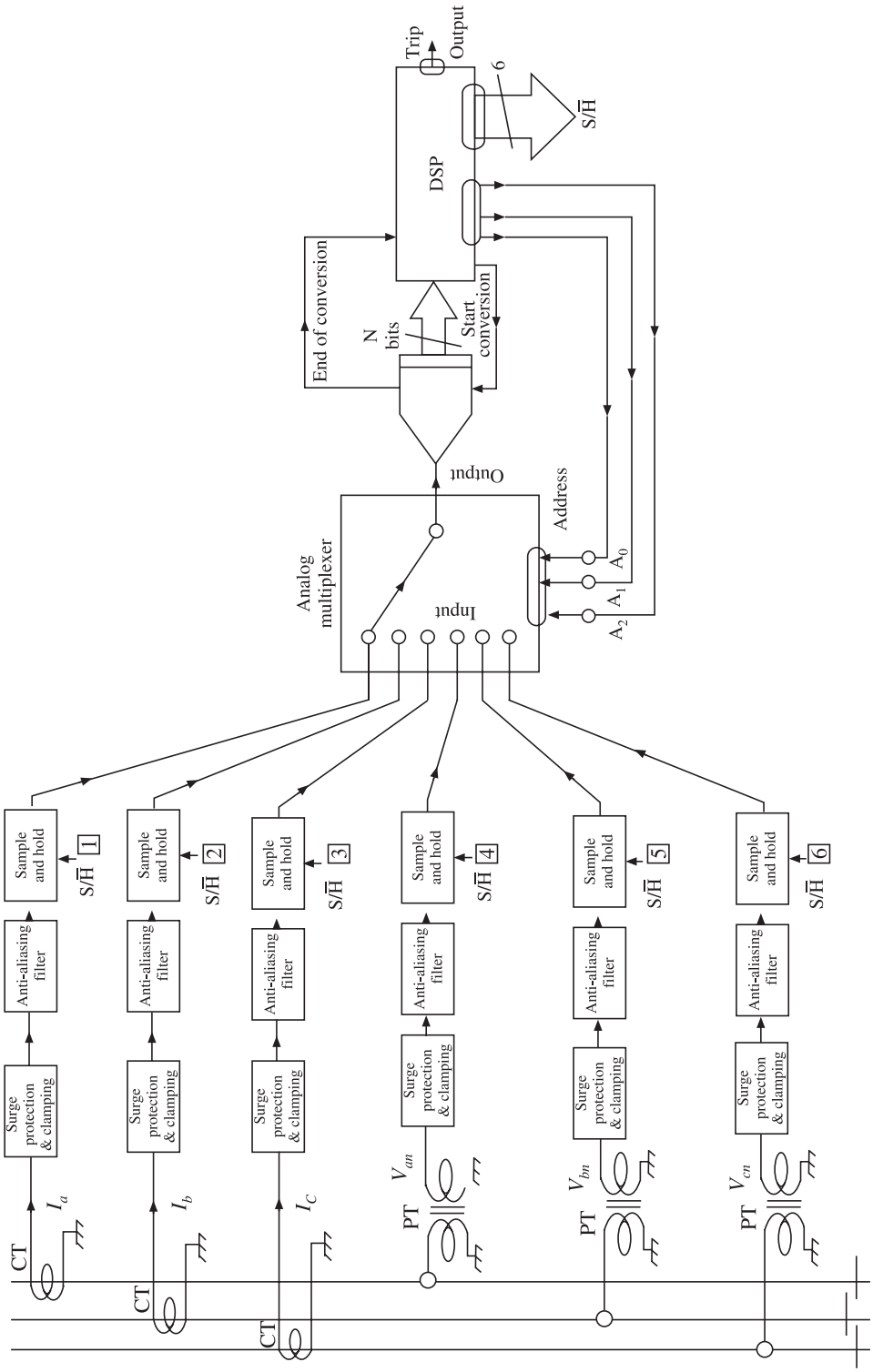


Figure 2.13 Block diagram of a numerical relay.

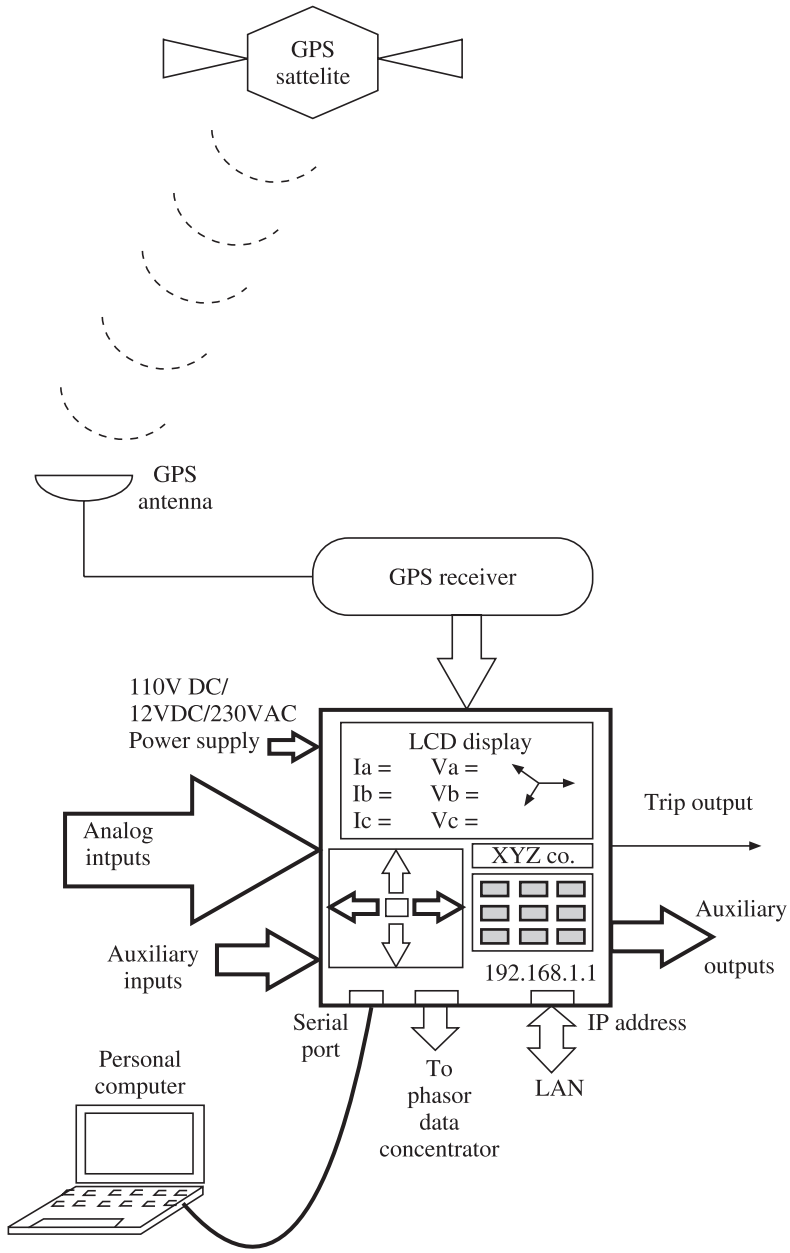


Figure 2.14 Modern numerical relay.

REVIEW QUESTIONS

1. What do you mean by resolution of an ADC?

2. What do you mean by 1 LSB, ½ LSB w.r.t. ADCs?

3. In a 10-bit ADC with input range of –10 V to 10 V, what is the value of 1 LSB and ½ LSB?

4. In an ADC with input range of +/–5 V, a resolution (1 LSB) of 10 mV is required. What is the minimum number of bits it must have?

5. Find the signal to noise ratio (in dB) due to quantization in an DSP system which has ADC of 16-bits and sampling is done at the Nyquist rate. What will be the error if sampling is done at 10 times the Nyquist rate?

6. Find the SNR (in dB) due to quantization in a DSP system with the following specifications:
Number of bits in the ADC = 12, signal bandwidth = 1000 Hz and sampling frequency = 5000 Hz.

7. How does one decide the minimum sampling frequency in a DSP system?

8. A power system signal is given by:

$$v(t) = 100\sin(2\pi 50t + 45°) + 50\sin(2\pi 100t + 25°) + \sin(2\pi 150t + 15°)$$

Suggest minimum sampling frequency, in hertz to avoid aliasing errors.

9. What, if any, are the disadvantages of sampling at very high frequencies , much above the Nyquist sampling rate?

10. Give at least two examples where aliasing is encountered in daily life.
(**Hint:** Movies, Table fan)

11. The frequency spectra of signal is as shown in Figure Q.11. If the signal is ideally sampled at intervals of 1 ms, then state with justification whether this will lead to aliasing. Sketch the frequency spectra of the sampled signal.

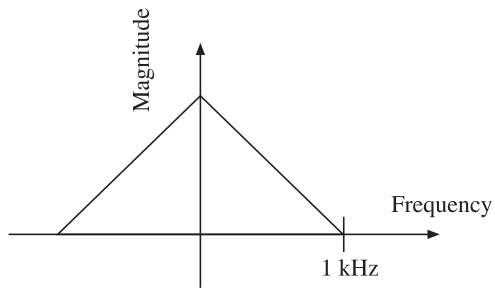


Figure Q.11

12. Find the maximum frequency that can be sampled without using hold circuit for a DSP system with the following specifications:

Conversion time of ADC = 5 μs and number of bits in the ADC = 16

13. Find the order of low-pass anti-aliasing filter of Butterworth type for a DSP system with the following specifications:-

ADC resolution = 12-bits , sampling frequency = 6000 Hz and cut-off frequency = 300 Hz

14. Provide a frequency domain explanation of the aliasing phenomenon.

3

Algorithms Based on Undistorted Single Frequency Sine Wave

3.1 Mann and Morrison Algorithm [26, 27]

In this section, we will discuss the algorithms for digital protection of power systems. One may feel awe and wonder at the variety of approaches that the researchers have taken. It seems that the search for algorithmic solutions to problems in the area of power system protection is an un-ending one. Inventions and discoveries made in other fields like communication, statistics, artificial neural networks, fuzzy logic, genetic algorithms and expert systems have fuelled research in this area.

3.1.1 Historical Perspective

Even though this algorithm was not the first among algorithms for digital protection, it was the first to get wide publicity. The algorithm proposed by M. Ramamoorthy, which was based on Fourier analysis, in the CIGRE Journal can be considered as the first algorithm to be proposed for digital protection of power system.

It should be kept in mind that in the early 70's when these algorithms were first proposed, the microprocessor had not burst on the scene and the digital revolution which was to be fuelled by the availability of cheap microprocessors was just round the corner. Computers were a very costly affair. The confidence levels of practising engineers about the reliability of the new technology were very low to put it mildly. In those days researchers had to justify their idea of using computer for protection purposes by citing additional and possibly more important duties that the computer could do. However, as is well-known, the digital revolution in the late 70's totally changed the rules of the game. The microprocessor just became a commodity and another component of a bigger computing system. As hardware prices plummeted, it was not long before engineers realised that software has become a key part of the whole and as the cliché goes, rest is history.

3.1.2 Derivation of the Sample and Derivative (Mann & Morrison) Algorithm

In this algorithm, we model the signal in a very simplistic manner. It is assumed that the signal can be represented as a pure sinusoid whose amplitude as well as frequency is constant during the period under consideration as shown in Figure 3.1.

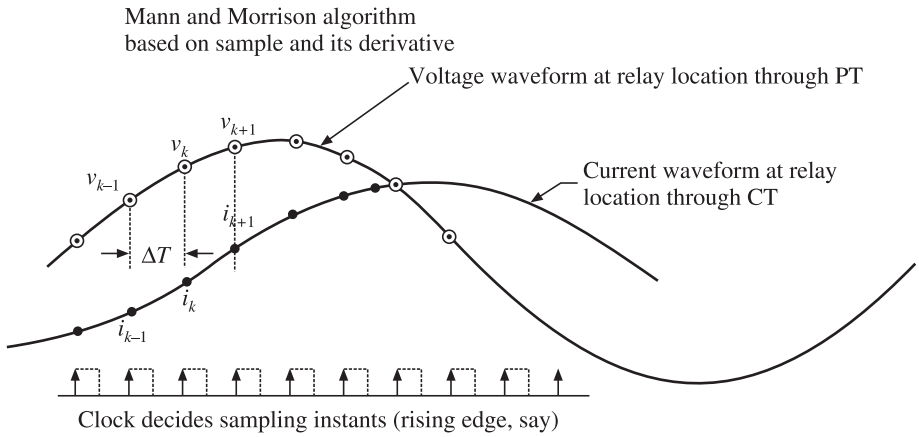


Figure 3.1 Mann and morrison algorithm.

Thus, voltage samples collected by the relay are assumed to belong to a signals of the type:

$$v(t_k) = V_m \sin(\omega t_k + \theta_v) \tag{3.1}$$

where we assume V_m, ω and θ_v to be constant. Further we assume that the frequency 'ω' is known. Thus in the above equations we have two unknowns, namely the peak value of the signal and its phase angle. Hence, we need one more equation to solve for the unknowns. Let us generate another equation by taking derivative of the first equation giving:

$$v'(t_k) = \omega V_m \cos(\omega t_k + \theta_v) \tag{3.2}$$

from which we get:

$$\frac{v'(t_k)}{\omega} = V_m \cos(\omega t_k + \theta_v) \tag{3.3}$$

Combining Eqs. (3.1) and (3.3), we get:

$$V_m^2 = (v(t_k))^2 + \left(\frac{v'(t_k)}{\omega}\right)^2 \tag{3.4}$$

Hence, we can find the amplitude of voltage signal as:

$$V_m = \sqrt{\left((v(t_k))^2 + \left(\frac{v'(t_k)}{\omega}\right)^2\right)} \tag{3.5}$$

Similarly, we can write for the current signal as:

$$i(t_k) = I_m \sin(\omega t_k + \theta_i) \tag{3.6}$$

Differentiating Eq. (3.6) at instant t_k we get:

$$i'(t_k) = \omega I_m \cos(\omega t_k + \theta_i) \tag{3.7}$$

Combining Eqs. (3.6) and (3.7) we get:

$$I_m^2 = (i(t_k))^2 + \left(\frac{i'(t_k)}{\omega}\right)^2 \tag{3.8}$$

Hence, we can find the amplitude of current signal as:

$$I_m = \sqrt{\left((i(t_k))^2 + \left(\frac{i'(t_k)}{\omega}\right)^2\right)}$$

Dividing Eq. (3.1) by Eq. (3.3) we get:

$$\tan(\omega t_k + \theta_v) = \frac{v(t_k)}{v'(t_k)/\omega} \tag{3.9}$$

$$\omega t_k + \theta_v = \tan^{-1}\left(\frac{v(t_k)}{v'(t_k)/\omega}\right) \tag{3.10}$$

$$\theta_v = \tan^{-1}\left(\frac{v(t_k)}{\frac{v'(t_k)}{\omega}}\right) - \omega t_k \tag{3.11}$$

Similarly, we can get:

$$\theta_i = \tan^{-1}\left(\frac{i(t_k)}{\frac{i'(t_k)}{\omega}}\right) - \omega t_k \tag{3.12}$$

The derivatives can be computed numerically using the central difference formula as:

$$v_k' = \frac{(v_{k+1} - v_{k-1})}{2\Delta t} \tag{3.13}$$

and

$$i_k' = \frac{(i_{k+1} - i_{k-1})}{2t} \tag{3.14}$$

Thus, computation of one derivative requires three samples. Now, we cannot base the trip decision on a single estimate (which involves only three samples), we have to continuously keep estimating the peak and the phase angle and hence the phasor, in order to be sure about what way the phasor is really going. We should, thus, base our trip decision on a sufficiently large number of estimates rather than a single estimate. This will make the decision more robust and provide some measure of noise immunity as discussed in the following section.

Let us summarise the results of the Mann and Morrison algorithm as shown in Table 3.1.

Table 3.1 Mann and Morrison algorithm: Summary of results

Model of voltage signal	$v(t_k) = V_m \sin(\omega t_k + \theta_v)$
Model of current	$i(t_k) = I_m \sin(\omega t_k + \theta_i)$
$V_m =$	$\sqrt{\left((v(t_k))^2 + \left(\frac{v'(t_k)}{\omega}\right)^2\right)}$ where $v'_k = \frac{(v_{k+1} - v_{k-1})}{2\Delta t}$
$\theta_v =$	$\tan^{-1}\left(\frac{v(t_k)}{v'(t_k)/\omega}\right) - \omega t_k$ radians
$I_m =$	$\sqrt{\left((i(t_k))^2 + \left(\frac{i'(t_k)}{\omega}\right)^2\right)}$ where $i'_k = \frac{(i_{k+1} - i_{k-1})}{2\Delta t}$
$\theta_i =$	$\tan^{-1}\left(\frac{i(t_k)}{i'(t_k)/\omega}\right) - \omega t_k$ radians

3.1.3 Instantaneous OC Relay Based on Mann and Morrison Algorithm

We can implement a variety of relays based on the Mann and Morrison algorithm viz., an over-current relay, directional over current relay or a distance relay. Every relay requires first to compute from the instantaneous values of the input quantities, the phasor for the input quantity. Since the Mann and Morrison algorithm estimates the phase angle and the peak value, the RMS value being given by $\text{Peak Value}/\sqrt{2}$, we are essentially estimating the phasor. Note that this is possible only because of the underlying assumption of pure sine waveform.

Let us consider the example of implementing an instantaneous over-current relay based on this algorithm. Obviously we will have to wait for three samples to make one estimate of the phasor. Afterwards, however, on the occurrence of every new incoming sample we can modify our estimate by sifting-in the newest sample and shifting-out the oldest sample.

Thus we have a calculation window of three samples, which keeps sliding over the signal. In practise we will not issue a trip signal as soon as the RMS value is more than the pickup value because, the increase in RMS value could, in all probability, be due to a noise spike.

In order to avoid, maloperation, we set up a 'fault counter' and increment it every time the RMS value is more than the threshold and decrement it if the RMS value is less than the threshold. In case the counter crosses a predetermined threshold, we can be sure that there indeed is a fault on the system and issue trip command to the circuit breaker.

The threshold value to be used for the counter will be based on severity of noise at a particular location. The general logic of such an implementation is shown in Figure 3.2.

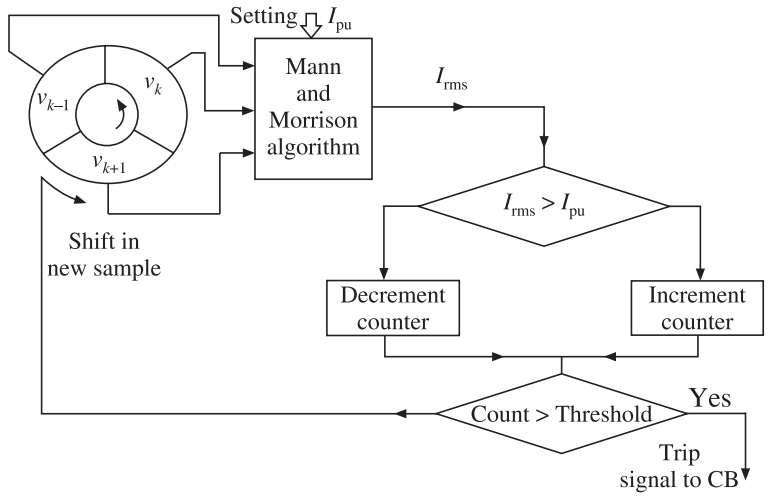


Figure 3.2 Instantaneous OC relay based on Mann and Morrison algorithm.

3.1.4 Simulation of the Mann and Morrison Algorithm in Spreadsheet

Spreadsheet software provides a very convenient environment for simulation of numerical relaying algorithms. We can quickly visualise and plot the relevant information. Spreadsheets are famous for their capability of performing 'what if' analysis.

For the sake of demonstrating the working of the algorithm, we take a single phase system as shown in Figure 3.3. Spreadsheet shown in Figure 3.4 shows the samples of voltage and

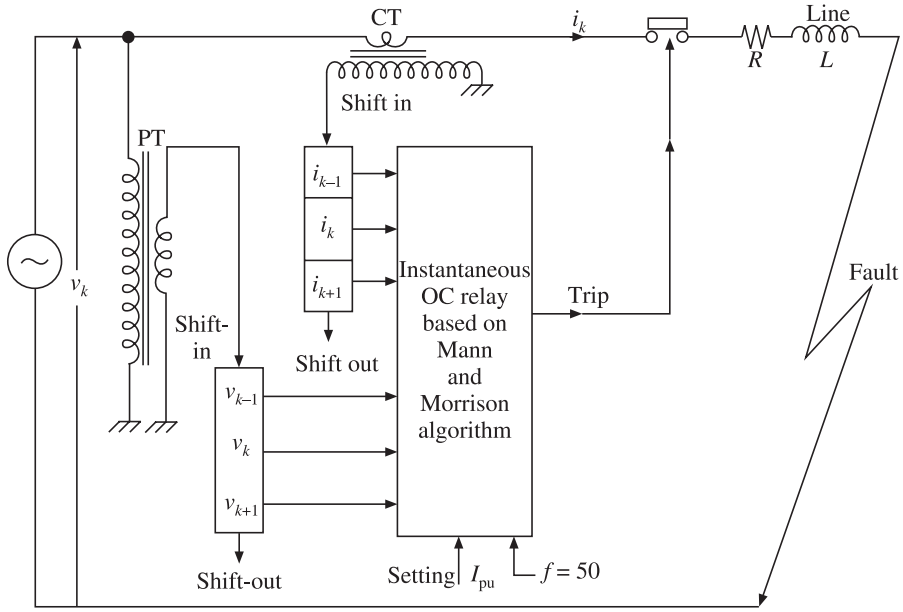


Figure 3.3 Simulation of SIR based on Mann and Morrison algorithm.

current which are generated for a cycle. A window of three samples of voltage and three samples of current, slides through these samples and computes the apparent resistance and reactance as seen from the relay location.

Spreadsheet for Mann and Morrison Algorithm

					/=====		Settings		=====/
Sampling Freq	Set fs==>	50000	Set Vm===>	230	Theta(deg)=	10		f=	50
Sample No. ========>	1	2	3	4					
Time ============>	t0	t1	t2	t3					
(seconds) ========>	0	0.00002	0.00004	0.00006					
v(t) ===>	39.9391	41.3615	42.7822	44.2013					
v'		71078.1821	70995.1359						
v^2		1710.7705	1830.3173						
v'/w		226.2484	225.9840						
(v'/w)^2		51188.3165	51068.7718						
v^2+(v'/w)^2		52899.0870	52899.0891						
Vm=sqrt(v^2+(v'/w)^2)		229.9980	229.9980						
v/(v'/w)		0.1828	0.1893						
Phi_V=atan(v/(v'/w))-wt		10.0001	10.0001						
				/==============	Settings	==============/			
			Set Im===>	40	Theta(deg)=	-75		f=	50
i(t) ===>	-38.6370	-38.5712	-38.5039	-38.4350					
i'		3328.5959	3404.6657						
i^2		1487.7392	1482.5495						
i'/w		10.5952	10.8374						
(i'/w)^2		112.2588	117.4484						
i^2+(i'/w)^2		1599.9980	1599.9979						
Im=sqrt(i^2+(i'/w)^2)		40.0000	40.0000						
i/(i'/w)		-3.6404	-3.5529						
Phi_i=atan(i/(i'/w))-wt		-75.0000	-75.0000						
	| Z|	5.7500	5.7500						
	Ang_Z_deg	85.0000	85.0001						
	R=========>	0.5011	0.5011						
	X=========>	5.7281	5.7281						
	L ========>	0.0182	0.0182						

Figure 3.4 Spread sheet for simulation of Mann and Morrison algorithm.

3.1.5 Simulation of Mann and Morrison Algorithm in MATLAB

The system shown in Figure 3.3 is simulated with the help of the program written in MATLAB is shown below:

```
% Mann and Morrison Algorithm
clear;clc

%=================== Settings =================

% System frequency and Sampling freuency
f=50; Over_samp_factor =100; % OSF = (fsamp/2fsig)

% Voltage Amplitude and Phase angle
Vm = 110; Phi_v=30;% deg

% Line R-L Parameters
R_perKm =.01; XbyR =10;Line_Length = 100 ; %km

%================================================
% =============== End Of settings=============
R = R_perKm * Line_Length ;
X = R * XbyR ;
w = 2 * pi * f;
L = X / w ;
Z = sqrt(R^2 + X^2);
Im = Vm / Z ;
Phi_i = (atan2(XbyR,1))*(180/pi);
% v = Vm sin(2 * pi * f * t + Phi_v)
% i = Im sin(2 * pi * f * t + Phi_v - Phi_i)
%================================================
R ; L ; X ; Z ; %<== For error calculation
Vm; Phi_v ;   %<== For error calculation
Im; Phi_i;    %<== For error calculation
%================================================
fsamp = 2* f * Over_samp_factor;
dT   = 1/ fsamp ;
T    = 1 / f ;
k=0;
for t = 0 :dT:T
 k=k+1;
 v(k)= Vm * sin(2 * pi * f * t + (Phi_v)*(pi/180) );
 i(k)= Im * sin(2 * pi * f * t + (Phi_v - Phi_i)*(pi/180));
end
%plot(v):holdon
%plot(i)
no_of_samples=numel(v);
% for n=1:no_of_samples
```

```
%   fprintf(' v = %f i =%f \n ',v(n),i(n))
% end
for k = 2 : no_of_samples-1
  v_dash(k)=(v(k+1)-v(k-1))/(2*dT);
  i_dash(k)=(i(k+1)-i(k-1))/(2*dT);

  Vm_est(k)=sqrt(v(k)^2 + (v_dash(k)/w)^2);
  Im_est(k)=sqrt(i(k)^2 + (i_dash(k)/w)^2);
  Phi_V_est_rad(k) = (atan2(((v(k))/((v_dash(k))/w)),1))-(w*k*dT);
  Phi_V_est_deg(k)= (Phi_V_est_rad(k))*(180/pi);

  Phi_I_est_rad(k) = (atan2(((i(k))/((i_dash(k))/w)),1))-(w*k*dT);
  Phi_I_est_deg(k)= (Phi_I_est_rad(k))*(180/pi);

  Zmag_est(k) = Vm_est /Im_est;
  Zang_est_rad(k)= Phi_V_est_rad(k)- Phi_I_est_rad(k);
  Zang_est_deg(k)= Phi_V_est_deg(k)- Phi_I_est_deg(k);

  R_est(k) = Zmag_est(k)*cos(Zang_est_rad(k));
  X_est(k) = Zmag_est(k)*sin(Zang_est_rad(k));
  L_est(k) = X_est(k)/w;
end
fprintf('============================================================
==========================================================\n')
fprintf('Samp   v     i   v_dash    Vm_est    i_dash    Im_est   Phi_V_est
Phi_I_est  R_est  L_est\n')
fprintf('True Value =                  %5.2f          %5.4f   %5.2f   %5.4f
%5.2f% 5.4f\n',Vm,Im,Phi_v,Phi_v-Phi_i,R,L )
fprintf('============================================================
==========================================================\n')
for n = 1:no_of_samples-1
  if(n==1)
    fprintf('%2i |  %f %f\n',n,v(n),i(n))
    continue
  end

fprintf('%2i |   %f  %f| %f| %f| %f| %f| %f| %f| %f| %f\n',n,v(n),i(n),v_
dash(n),Vm_est(n),i_dash(n),Im_est(n),Phi_V_est_deg(n),Phi_I_est_
deg(n),R_est(k),L_est(k) )
end
% Vm_est
% Im_est
%
% Phi_V_est_rad
% Phi_V_est_deg
% Phi_I_est_deg
```

Samp	v	i	v_dash	Vm_est	i_dash	Im_est	Phi_V_est	Phi_I_est	R_est	I_est
True Value =				110.00		10.9454	30.00	-54.2894	1.00	0.0318
1	55.000000	-8.887405								
2	57.965138	-8.682345	29365.352084	109.986931	2093.448010	10.944742	28.204221	-56.093959	0.998554	0.0318
3	60.873070	-8.468716	28778.957265	109.987448	2178.078140	10.944687	28.204344	-56.094027	0.998554	0.0318
4	63.720929	-8.246729	28164.161090	109.987978	2260.558771	10.944631	28.204451	-56.094076	0.998554	0.0318
5	66.505903	-8.016604	27521.570289	109.988521	2340.808503	10.944575	28.204539	-56.094107	0.998554	0.0318
6	69.225243	-7.778567	26851.819020	109.989073	2418.748139	10.944518	28.204610	-56.094119	0.998554	0.0318
7	71.876266	-7.532854	26155.568248	109.989632	2494.300763	10.944462	28.204662	-56.094113	0.998554	0.0318
8	74.456357	-7.279707	25433.505088	109.990197	2567.391814	10.944405	28.204696	-56.094088	0.998554	0.0318
9	76.962867	-7.019376	24686.342129	109.990764	2637.949158	10.944349	28.204712	-56.094045	0.998554	0.0318
10	79.393625	-6.752117	23914.816730	109.991333	2705.903166	10.944294	28.204708	-56.093983	0.998554	0.0318
11	81.745931	-6.478195	23119.690295	109.991899	2771.186773	10.944239	28.204687	-56.093903	0.998554	0.0318
12	84.017563	-6.197880	22301.747317	109.992462	2833.735554	10.944186	28.204646	-56.093806	0.998554	0.0318
13	86.206280	-5.911448	21461.795606	109.993019	2893.487780	10.944134	28.204588	-56.093691	0.998554	0.0318
14	88.309922	-5.619183	20600.663494	109.993568	2950.384483	10.944083	28.204511	-56.093559	0.998554	0.0318
15	90.326413	-5.321371	19719.201015	109.994107	3004.369512	10.944034	28.204417	-56.093411	0.998554	0.0318
16	92.253762	-5.018309	18818.278065	109.994633	3055.389592	10.943987	28.204305	-56.093247	0.998554	0.0318
17	94.090069	-4.710293	17898.783746	109.995145	3103.394371	10.943942	28.204176	-56.093068	0.998554	0.0318
18	95.833519	-4.397630	16961.625489	109.995640	3148.336474	10.943899	28.204031	-56.092875	0.998554	0.0318
19	97.482394	-4.080626	16007.728155	109.996117	3190.171549	10.943859	28.203870	-56.092668	0.998554	0.0318
20	99.035065	-3.759595	15038.033126	109.996573	3228.858311	10.943821	28.203693	-56.092447	0.998554	0.0318
21	100.490000	-3.434855	14053.497374	109.997007	3264.358578	10.943786	28.203502	-56.092215	0.998554	0.0318
22	101.845764	-3.106724	13055.092517	109.997417	3296.637318	10.943754	28.203297	-56.091972	0.998554	0.0318
23	103.101019	-2.775327	12043.803860	109.997802	3325.662675	10.943725	28.203079	-56.091719	0.998554	0.0318
24	104.254525	-2.441991	11020.629422	109.998159	3351.406005	10.943698	28.202849	-56.091456	0.998554	0.0318
25	105.305145	-2.105246	9986.578954	109.998489	3373.841901	10.943675	28.202608	-56.091186	0.998554	0.0318
26	106.251841	-1.766823	8942.672938	109.998788	3392.948223	10.943656	28.202356	-56.090908	0.998554	0.0318
27	107.093679	-1.426656	7889.941583	109.999057	3408.706114	10.943639	28.202095	-56.090625	0.998554	0.0318
28	107.829829	-1.085082	6829.423809	109.999293	3421.100025	10.943626	28.201826	-56.090337	0.998554	0.0318
29	108.459564	-0.742436	5762.166217	109.999497	3430.117722	10.943617	28.201550	-56.090045	0.998554	0.0318
30	108.982262	-0.399058	4689.222064	109.999667	3435.750307	10.943611	28.201267	-56.089750	0.998554	0.0318

Figure 3.5 Output of MATLAB simulation.

3.2 Three-Sample Technique [20, 65]

Three-sample algorithm, kind of, extends Mann and Morrison algorithm. It removes the condition in Mann and Morrison algorithm that frequency should be known. Recall that a single frequency sinusoid given by $v = V_m \sin(\omega t + \theta)$ or $v = V_m \cos(\omega t + \theta)$ has three unknowns associated with it viz., amplitude V_m, frequency ω and phase angle θ. This algorithm enables us to estimate all the three unknowns using only three samples. It is possible, because, to solve for three unknowns we need only three equations. Three samples that we gather from the signal are the known quantities on the LHS of the three equations in three unknowns. The signal and its sampling are shown in Figure 3.6.

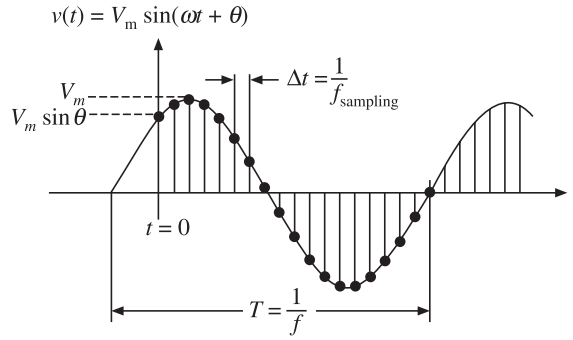


Figure 3.6 Sampled signal for three-sample algorithm.

The model of the signal assumed for this algorithm is:

$$v = V_m \sin(\omega t + \theta) \tag{3.15}$$

$$v = V_m \sin(2\pi f t + \theta) \tag{3.16}$$

Let the three signal samples, starting from an arbitrary index 'k' be:

$$v_k = A \sin(2\pi f(k\Delta t) + \theta) \tag{3.17}$$

$$v_{k+1} = A \sin(2\pi f(k\Delta t + \Delta t) + \theta) \tag{3.18}$$

$$v_{k+2} = A \sin(2\pi f(k\Delta t + 2\Delta t) + \theta) \tag{3.19}$$

3.2.1 Determination of Frequency of the Signal

Now, consider $v_k + v_{k+2}$

$$v_k + v_{k+2} = A[\sin(2\pi f(k\Delta t) + \theta) + \sin(2\pi f(k\Delta t + 2\Delta t) + \theta)] \tag{3.20}$$

If we consider,

$$X = 2\pi f(k\Delta t + \Delta t) + \theta \text{ and } Y = 2\pi f \Delta t$$

then

$$X + Y = 2\pi f(k\Delta t + \Delta t) + \theta + 2\pi f \Delta t = 2\pi f(k\Delta t + 2\Delta t) + \theta \tag{3.21}$$

$$X - Y = 2\pi f(k\Delta t + \Delta t) - 2\pi f\Delta t = 2\pi f(k\Delta t) + \theta \tag{3.22}$$

Hence $v_k + v_{k+2}$ can be written as:

$$v_k + v_{k+2} = A\,[\sin(X - Y) + \sin(X + Y)] \tag{3.23}$$

We can use the following trigonometric identity:

$$[\sin(X - Y) + \sin(X + Y)] = 2\,\sin(X)\cos(Y) \tag{3.24}$$

and we can write:

$$v_k + v_{k+2} = A\,[2\,\sin(X)\cos(Y)] \tag{3.25}$$

$$v_k + v_{k+2} = A\,[2\,\sin(2\pi f(k\Delta t + \Delta t) + \theta)\cos(2\pi f\Delta t)] \tag{3.26}$$

However,

$$\text{A}\,\sin(2\pi f(k\,\Delta t + \Delta t) + \theta) = v_{k+1} \tag{3.27}$$

Hence, we can write

$$v_k + v_{k+2} = [2\,v_{k+1}\cos(2\pi f\,\Delta t)] \tag{3.28}$$

$$\cos(2\pi f\Delta t) = \frac{v_k + v_{k+2}}{2\,v_{k+1}} \tag{3.29}$$

$$2\pi f\Delta t = \cos^{-1}\frac{v_k + v_{k+2}}{2\,v_{k+1}} \tag{3.30}$$

$$f = \frac{1}{2\pi\Delta t}\cos^{-1}\left(\frac{v_k + v_{k+2}}{2\,v_{k+1}}\right) \tag{3.31}$$

Thus, since Δt, the sampling time period which is $1/f_{\text{sampling}}$ is known, the unknown frequency can be estimated with the help of three consecutive samples.

3.2.2 Determination of Amplitude of the Signal

Now, in order to estimate the peak value 'A' of the signal, consider $(v_{k+1})^2 - (v_k)(v_{k+2})$

or, $$(A\sin(2\pi f(k\Delta t + \Delta t) + \theta))^2 - (A\,\sin(2\pi f(k\,\Delta t) + \theta))(A\sin(2\pi f(k\Delta t + 2\Delta t) + \theta)) \tag{3.32}$$

$$(A^2)((\sin(2\pi f(k\Delta t + \Delta t) + \theta))^2 - (\sin(2\pi f(k\Delta t) + \theta))(\sin(2\pi f(k\Delta t + 2\Delta t) + \theta)) \tag{3.33}$$

Using the trigonometric identity $(\sin\theta)^2 = \dfrac{1 - \cos 2\theta}{2}$
we can write

$$(\sin(2\pi f(k\,\Delta t + \Delta t) + \theta)^2 = \frac{1 - \cos(2(2\pi f(k\Delta t + \Delta t\,) + \theta))}{2} \tag{3.34}$$

further, if we write

$$(2\pi f(k\Delta t + 2\Delta t) + \theta) = X \tag{3.35}$$

and
$$(2\pi f(k\Delta t)+\theta)=Y \tag{3.36}$$
then we can write
$$(\sin(2\pi f(k\Delta t)+\theta))(\sin(2\pi f(k\Delta t+2\Delta t)+\theta))=\sin X\sin Y \tag{3.37}$$
Using trigonometric identity
$$\sin X\sin Y=\frac{\cos(X-Y)-\cos(X+Y)}{2} \tag{3.38}$$
$$X-Y=(2\pi f(k\Delta t+2\Delta t)+\theta-(2\pi f(k\,\Delta t)-\theta))=2\pi f(2\Delta t) \tag{3.39}$$
$$X+Y=(2\pi f(k\Delta t+2\Delta t)+\theta+(2\pi f(k\Delta t)+\theta))=2\pi f(2k\Delta t+2\Delta t)+2\theta \tag{3.40}$$
Hence, we can write
$$(\sin(2\pi f(k\Delta t)+\theta))(\sin(2\pi f(k\Delta t+2\Delta t)+\theta))=\frac{\cos(2\pi f(2\Delta t))-\cos(2\pi f(2k\Delta t+2\Delta t)+2\theta)}{2} \tag{3.41}$$
Hence, we can now write
$$(v_{k+1})^2-(v_k)\,(v_{k+2})=A^2\left(\left(\frac{1-\cos(2(2\pi f(k\Delta t+\Delta t)+\theta))}{2}\right)\right.$$
$$\left.-\frac{\cos(2\pi f(2\Delta t))-\cos(2\pi f(2k\Delta t+2\Delta t)+2\theta)}{2}\right) \tag{3.42}$$
or, $(v_{k+1})^2-(v_k)\,(v_{k+2})$
$$=A^2\left(\frac{1-\cos(2(2\pi f(k\Delta t+\Delta t)+\theta))-\cos(2\pi f(2\Delta t))+\cos(2(2\pi f(k\Delta t+\Delta t)+\theta))}{2}\right) \tag{3.43}$$
$$=A^2\left(\frac{1-\cos 2\pi f(2\Delta t)}{2}\right) \tag{3.44}$$
Using the trigonometric identity $\cos(2X)=2(\cos(X))^2-1$
$$=A^2\left(\frac{1-(2(\cos(2\pi f(\Delta t)))^2-1)}{2}\right) \tag{3.45}$$
$$=A^2\left(\frac{2-(2(\cos(2\pi f(\Delta t)))^2)}{2}\right) \tag{3.46}$$
$$=A^2\left(2\frac{(1-(\cos(2\pi f(\Delta t))^2)}{2}\right) \tag{3.47}$$
$$(v_{k+1})^2-(v_k)\,(v_{k+2})=A^2(\sin(2\pi f(\Delta t)))^2 \tag{3.48}$$
$$\sqrt{(v_{k+1})^2-(v_k)\,(v_{k+2})}=A\sin(2\pi f\Delta t) \tag{3.49}$$

$$A = \frac{\sqrt{(v_{k+1})^2 - (v_k)\,(v_{k+2})}}{\sin(2\pi f \Delta t)} \tag{3.50}$$

Since frequency 'f' has already been evaluated, the amplitude 'A' can be estimated.

3.2.3 Determination of Phase of the Signal

To estimate the phase angle we can use the equation for the first sample as:

$$v_k = A \sin(2\pi f(k\Delta t) + \theta) \tag{3.51}$$

which can be manipulated as:

$$\frac{v_k}{A} = \sin(2\pi f(k\Delta t) + \theta) \tag{3.52}$$

$$2\pi f(k\Delta t) + \theta = \sin^{-1}\frac{v_k}{A} \tag{3.53}$$

$$\theta = \sin^{-1}\left(\frac{v_k}{A}\right) - 2\pi f(k\Delta t) \tag{3.54}$$

In the above expression v_k, A, k and Δt all are known. Hence, phase angle θ can be estimated.

3.2.4 Summary of Three Sample Algorithm

Measurement of amplitude, phase and frequency from three-sample is summarised in Table 3.2.

Table 3.2 Summary of three-sample algorithm

Signal		$v = V_m \sin(\omega t + \theta)$
Samples		$v_k = A \sin(2\pi f(k\Delta t) + \theta)$ $v_{k+1} = A \sin(2\pi f(k\Delta t + \Delta t) + \theta)$ $v_{k+2} = A \sin(2\pi f(k\Delta t + 2\Delta t) + \theta)$
Frequency	f	$f(k) = \frac{1}{2\pi\Delta t}\cos^{-1}\left(\frac{v_k + v_{k+2}}{2v_{k+1}}\right)$
Amplitude	A	$A(k) = \frac{\sqrt{(v_{k+1})^2 - (v_k)\,(v_{k+2})}}{\sin(2\pi f \Delta t)}$
Phase	θ	$\theta(k) = \sin^{-1}\left(\frac{v_k}{A}\right) - 2\pi f(k\Delta t)$

3.2.5 Excel Spreadsheet for Simulation of Three-Sample Algorithm

Spreadsheet 3 Sample Algorithm for Measurement of Amplitude,Phase and Frequency														Signal Assumed==>v=Acos(wt+Q)
	f	49.5												
	sampling factor	10		cos(wT)=(v(o)+v(2))/v(1)										
	fsamp	495		A=sqrt(v(1)^2-v(0)v(2))/sinwT)										
	Vm =	234		Q=acos(v(o)/A)										
	w	311.0177		Note:- A=Vm										
Radian=>	Ph. AngQ	0.523599	30	<== Deg										
	v=Vm cos(wt+Q)													
Sample No.		0	1	2	3	4	5	6	7	8	9	10		
Time	t	0	0.00202	0.00404	0.006061	0.008081	0.010101	0.012121	0.014141	0.016162	0.018182	0.020202		
Instt. Volt	v(t)	202.6499	95.17637	-48.6513	-173.896	-232.718	-202.65	-95.1764	48.65134	173.8959	232.7181	202.6499		
	cos(wT)		0.809017	0.809017	0.809017	0.809017	0.809017	0.809017	0.809017	0.809017	0.809017	0.809017		
	wT		0.628319	0.628319	0.628319	0.628319	0.628319	0.628319	0.628319	0.628319	0.628319	0.628319		
2 *PI*f	w		311.0177	311.0177	311.0177	311.0177	311.0177	311.0177	311.0177	311.0177	311.0177	311.0177		
	A		234	234	234	234	234	234	234	234	234	234		
Phase angl	Q(Rad)		0.523599	0.523599	0.523599	0.523599	0.523599	0.523599	0.523599	0.523599	0.523599	0.523599		
	Q(deg)		30	30	30	30	30	30	30	30	30	30		
	f (Hz)		49.5	49.5	49.5	49.5	49.5	49.5	49.5	49.5	49.5	49.5		

Figure 3.7 Spreadsheet for simulation of Three-sample algorithm.

3.2.6 MATLAB Program for Simulating Three-Sample Algorithm

```
%  ------------------------------------------------------------------------
-------
%3 Sample's Algorithm for measurement of amlitude, phase and frequency
% Signal is modelled as:-
% x = A cos( wt +theta) ; dT=1/fsamp  ;
% cos(w dT)=[x(0)+x(2)] / [ (2 x(1))]        ;
% w = acos(x(0)+x(2)/ (2 x(1)))/w dT   ;
% A =sqrt[(x(1)^2 -x(0)x(2))]/sin(wdT) ;
% theta =  acos[ x(0)/A]                  ;
%-------------------------------------------------------------
clear,clc
%===== Settings =================
A = 100+round(rand*10) ; fsig = 50+(0.5-rand) ; theta= 30+round(rand*10);
Over_sampling_factor = 10;
%================================
w =2*pi*fsig;
fsamp =2*fsig*Over_sampling_factor;
dT = 1/ fsamp ; T=1/fsig;
k=0;
% Let us generate samples of signal
for t=0:dT:T
  k=k+1;
  x(k)=A * cos((2*pi*fsig*t) + (theta)*(pi/180));
end
plot(x)
for n=2:(numel(x))-1
w_est(n)=(acos( (x(n-1)+x(n+1) )/(2*x(n))))/(dT);
  f_est(n)=w_est(n)/(2*pi);
  A_est(n)=(sqrt(x(n)^2-x(n-1)*x(n+1)))/(sin(w_est(n)*dT));
  t = (n-1)*dT;
  theta_est(n)=acos((x(n))/(A_est(n)))- ((w_est(n))*t);
  % Why (n-1)?
  % Because when n=1; t=0;
  theta_est_deg(n)=(theta_est(n)*(180/pi));
  err_A(n)=((A_est(n)-A)/A)*(100);
  err_w(n)  =(( w_est(n)-w  )/(w))*(100);
  err_theta(n)=((theta_est_deg(n)-theta)/(theta))*(100);
end
% myQ=acos(x(1)/A_est(2))
% myQQ=myQ*(180/pi)
fprintf('=====================3  Sample  Technique
==========================\n')
fprintf('The true values are:-   \n')
fprintf(' A=%f  fsig=%f  theta=%f \n',A,fsig,theta )
fprintf('=========================================================\n')
```

```
fprintf('======The Estimated values and Errors are ================\n')
fprintf('n Omega    freq  Ampltd  PhAng  Err_A%% Err_w%% Err_Ph_Ang%% \n')
fprintf('=====================================================\n')


for n=2:numel(x)-1
 fprintf('%d %8.3f  %8.3f  %8.3f %8.3f %8.3f %8.3f  %8.3f\n',n,w_est(n),f_
est(n),A_est(n), theta_est_deg(n),err_A(n),err_w(n),err_theta(n) )
end
fprintf('\n\n')
for p=1:3
fprintf('Problem No. %d \n',p)
 fprintf('The samples of a single frequency signal are as shown below.\n')
fprintf('Make at least three estimates of the amplitude, phase and frequency\n')
 fprintf('  0      1      2     3       4       5    6    \n')
for n=p+1:p+7
fprintf(' %f ',x(n) )
end
fprintf('\n')
fprintf('=================================================== \n')
fprintf('\n\n')
end
```

```
=============================================
The true values are:-
A=104.000000 fsig=50.195001 theta=39.000000
=============================================
======The estimated values and errors are ===================================================
```

n	Omega	freq	Ampltd	PhAng	Err_A%	Err_w%	Err_Ph_Ang%
2	315.384	50.195	104.000	39.000	–0.000	0.000	–0.000
3	315.384	50.195	104.000	39.000	0.000	–0.000	0.000
4	315.384	50.195	104.000	39.000	0.000	–0.000	0.000
5	315.384	50.195	104.000	39.000	–0.000	0.000	–0.000
6	315.384	50.195	104.000	39.000	–0.000	0.000	–0.000
7	315.384	50.195	104.000	39.000	–0.000	0.000	–0.000
8	315.384	50.195	104.000	39.000	0.000	–0.000	0.000

3.3 First and Second Derivative Algorithm [11, 48]

Samples of voltage collected by the relay are assumed to belong to a signals of the type:

$$v(t_k) = V_m \sin(\omega t_k + \theta_v) \tag{3.55}$$

where we assume V_m, ω and θ_v to be constant. Further we assume that the frequency ω is known. Thus in the above equations, we have two unknowns, namely the peak value of the signal and its phase angle. Hence, we need one more equation to solve for the unknowns. Let us generate another equation by taking derivative of the first equation giving:

$$v'(t_k) = \omega V_m \cos(\omega t_k + \theta_v) \tag{3.56}$$

$$v''(t_k) = -\omega^2 V_m \sin(\omega t_k + \theta_v) \tag{3.57}$$

Hence, we can find the amplitude of the voltage signal as:

$$V_m = \sqrt{\left(\left(\frac{v'(t_k)}{\omega}\right)^2 + \left(\frac{v''(t_k)}{\omega^2}\right)^2\right)} \tag{3.58}$$

Similarly, we can write for the current signal as:

$$I_m = \sqrt{\left(\left(\frac{i'(t_k)}{\omega}\right)^2 + \left(\frac{i''(t_k)}{\omega^2}\right)^2\right)} \tag{3.59}$$

Further, on the lines of Mann and Morrison algorithm, it can be easily shown that:

$$\theta_v = \tan^{-1}\left(-\frac{v''(t_k)/\omega^2}{v'(t_k)/\omega}\right) - \omega t_k \tag{3.60}$$

Similarly, we can write for the current signal:

$$\theta_i = \tan^{-1}\left(-\frac{i''(t_k)/\omega^2}{i'(t_k)/\omega}\right) - \omega t_k \tag{3.61}$$

The derivatives can be computed numerically as shown below:

$$v'_k = \frac{(v_{k+1} - v_{k-1})}{2\Delta t} \text{ and } i'_k = \frac{(i_{k+1} - i_{k-1})}{2\Delta t} \tag{3.62}$$

$$v''(k) \cong \frac{v(k+1) - 2v(k) + v(k-1)}{\Delta t^2} \tag{3.63}$$

$$i''(k) \cong \frac{i(k+1) - 2i(k) + i(k-1)}{\Delta t^2} \tag{3.64}$$

The above results are summarised in Table 3.3.

Table 3.3 First and second derivative algorithm: Summary of results

Model of voltage signal	$v(t_k) = V_m \sin(\omega t_k + \theta_v)$
Model of current	$i(t_k) = I_m \sin(\omega t_k + \theta_i)$
V_m	$V_m = \sqrt{\left(\left(\frac{v'(t_k)}{\omega}\right)^2 + \left(\frac{v''(t_k)}{\omega^2}\right)^2\right)}$ where, $v'_k = \frac{(v_{k+1} - v_{k-1})}{2\Delta t}$ $v''(k) \cong \frac{v(k+1) - 2v(k) + v(k-1)}{\Delta t^2}$
θ_v	$\theta_v = \tan^{-1}\left(-\frac{v''(t_k)/\omega^2}{v'(t_k)/\omega}\right) - \omega t_k$ radians
I_m	$I_m = \sqrt{\left(\left(\frac{i'(t_k)}{\omega}\right)^2 + \left(\frac{i''(t_k)}{\omega^2}\right)^2\right)}$ where, $i'_k = \frac{(i_{k+1} - i_{k-1})}{2\Delta t}$ $i''(k) \cong \frac{i(k+1) - 2i(k) + i(k-1)}{\Delta t^2}$
θ_i	$\theta_i = \tan^{-1}\left(-\frac{i''(t_k)/\omega^2}{i'(t_k)/\omega}\right) - \omega t_k$ radians

3.3.1 Excel Spreadsheet for Simulation of First and Second Derivative Algorithm

Spreadsheet for First and Second Derivative Algorithm

Set	Set	Set	Set	Set	Im		Set						
Vm	f_system	ϕv (deg)	Line Length km	R per km		L per km	Over-sampling factor	f_samp	delta_t	Z	omega	Line Angle	Phi_i
100	50	30	100	0.036	2.763992	0.001146	100	10000	0.0001	36.17955	314.1593	84.28941	-54.2894
		Total R =	3.6	Tot L =	0.114592	XbyR	10	Tot R =	3.6	Tot L=	0.114592		

Estimation of Voltage Phasor Using First and Second Derivative Algorithm

Estimation of R , L from V and I

Sample number	time	Instt Volt	v'	v'/w	v''/w^2	Vm_est= sqrt((v'/w)^2+(v''/w^2)^2)	err_Vm%	ϕv _est(deg)	err_Phi_V %	R_estim	X_estimate	L_estim	Error in R	Error in L
0	0	0												
0.031416	1	0.0001	52.69558	26695.77	84.97529	-52.6912	99.9858	-0.01416	30.00211	0.007035	3.59714	35.99923	0.114589	-0.07945	-0.00213
0.062832	2	0.0002	55.33915	26162.69	83.27842	-55.3346	99.9861	-0.01393	30.00217	0.007241	3.597097	35.99942	0.11459	-0.08063	-0.00162
0.094248	3	0.0003	57.92812	25603.78	81.49937	-57.9234	99.9863	-0.01369	30.00223	0.007418	3.597066	35.9996	0.11459	-0.08149	-0.00111
0.125664	4	0.0004	60.45991	25019.61	79.63989	-60.4549	99.9866	-0.01344	30.00227	0.007565	3.597047	35.99978	0.114591	-0.08203	-0.0006
0.15708	5	0.0005	62.93204	24410.74	77.70181	-62.9269	99.9868	-0.01319	30.0023	0.007683	3.597039	35.99997	0.114591	-0.08225	-8.3E-05
0.188496	6	0.0006	65.34206	23777.79	75.68705	-65.3367	99.9871	-0.01294	30.00233	0.00777	3.597043	36.00016	0.114592	-0.08214	0.000433
0.219911	7	0.0007	67.6876	23121.37	73.5976	-67.682	99.9873	-0.01268	30.00235	0.007827	3.597059	36.00034	0.114593	-0.0817	0.000948
0.251327	8	0.0008	69.96633	22442.13	71.43552	-69.9606	99.9876	-0.01242	30.00236	0.007853	3.597086	36.00053	0.114593	-0.08095	0.001459
0.282743	9	0.0009	72.17602											

Estimation of Current Phasor Using First and Second Derivative Algorithm

		Instt Current	i'	i'/w	i''/w^2	Im_est=sqrt((i'/w)^2+(i''/w^2)^2)	err_Im%	ϕi _est(deg)	err_Phi_i %
0	0	-2.24429							
1	0.0001	-2.19251	528.6485	1.68274	2.192331	2.76368	-0.01127	-54.2917	0.004193
2	0.0002	-2.13856	550.0197	1.750767	2.138389	2.76367	-0.01153	-54.2917	0.004255
3	0.0003	-2.08251	570.8482	1.817066	2.082336	2.76367	-0.01178	-54.2917	0.0043
4	0.0004	-2.02439	591.1133	1.881572	2.024228	2.76366	-0.01204	-54.2918	0.004329
5	0.0005	-1.96428	610.795	1.944221	1.964123	2.76365	-0.01229	-54.2918	0.00434
6	0.0006	-1.90224	629.8739	2.004951	1.90208	2.76365	-0.01255	-54.2918	0.004334
7	0.0007	-1.83831	648.3313	2.063703	1.838159	2.76364	-0.01281	-54.2917	0.004311
8	0.0008	-1.77257	666.1488	2.120417	1.772424	2.76363	-0.01307	-54.2917	0.004271
9	0.0009	-1.70508							

Figure 3.8 Spreadsheet for implementation of first and second derivative algorithm.

3.3.2 MATLAB Simulation of First and Second Derivative Algorithm

```
% ------------------------------------------------------------------
% First and Second Derivative Algorithm
% ------------------------------------------------------------------

clear ; clc ; clf
%================================================
mag_z_set      = 30   ; % Relay Reach Setting
theta_n       = pi/2  ; % Relay MTA setting
%================================================
%--------------Numerical Problem Setting ---------------------------------
f          = 50   ; % System frequency
OverSamplingFactor = 20  ; % fsamp = 2 * OSF * fsig
km          = 100  ; % kilometers
R          = 0.037 ; % ohms per km for 345kV line
XbyR_ratio      = 9.9189 ; % for 345 kV line
%================================================

f_samp    = 2*OverSamplingFactor*f      ;
X      = R*XbyR_ratio           ;
L      = X/(2*pi*f)            ;
Z      = sqrt( R*R + X* X)         ;
line_ang_deg = (180/pi)*atan2(X,R)        ;
w      = 2* pi* f              ;
T      = 1/f                 ;
dT      = 1/f_samp              ;
Vm      = 100                ;
Im      = Vm / Z              ;
Qv_deg    = 30                ;
Qv      = Qv_deg*(pi/180)          ;
Qi      = Qv - atan2(X,R)          ;
Qvdeg     = Qv * (180/pi)           ;
Qi_deg     = Qi*(180/pi)            ;
k = 0 ;
sss=50;
%---------------------- Generation of Samples ---------------------------
 for tim   = 0:dT: T
   k    = k + 1                      ;
   t(k)  = tim                        ;
   v(k)  = Vm * sin( 2* pi* f * t(k) + Qv)          ;
   i(k)  = Im * sin( 2* pi* f * t(k) + Qi)          ;

 end
% ------------------_------------------------------------------------------
samples    = k                          ;

for n = 2:samples-1
```

```
%---------- Estimation of voltage phasor ------------------------------
  v_d(n) = ( v(n+1)-v(n-1)) / (2*dT) ;% First derivative
  v_dd(n) = (v(n+1)-2*v(n)+ v(n-1))/(dT*dT) ;% 2nd Derivative
  Vm_est(n)  = sqrt( (v_d(n)/w)^2 +  ( v_dd(n)/(w*w) )^2  )  ;
  num   = (v_dd(n))/((w*w))                  ;
  den   =  v_d(n)/w                    ;
  y    = num/den;
  theta_v_est(n) = (atan2(- y,1))-((w*(n-1)*dT));
  theta_v_deg_est(n)=theta_v_est(n)*(180/pi);
  err_Vpc(n)=(((Vm_est(n))-(Vm))/(Vm))*100;
  err_theta_v_deg(n)=(((theta_v_deg_est(n))-(Qv_deg))/(Qv_deg))*100;
%-------------------------------------------------------------------------
%---------- Estimation of Current phasor ------------------------------
  i_d(n) = ( i(n+1)-i(n-1)) / (2*dT)   ;% First derivative
  i_dd(n) = (i(n+1)-2*i(n)+ i(n-1))/(dT*dT) ;% 2nd Derivative
  Im_est(n)  = sqrt( (i_d(n)/w)^2 +  ( i_dd(n)/(w*w) )^2  )  ;
  numi   = (i_dd(n))/((w*w))                  ;
  deni   =  i_d(n)/w                    ;
  y    = numi/deni;
  theta_i_est(n) = (atan2(- y,1))-((w*(n-1)*dT));
  theta_i_deg_est(n)=theta_i_est(n)*(180/pi);
  err_ipc(n)=(((Im_est(n))-(Im))/(Im))*100;
  err_theta_i_deg(n)=(((theta_i_deg_est(n))-(Qi_deg))/(Qi_deg))*100;
%-------------------------------------------------------------------------
R_est(n)=(Vm_est(n)/Im_est(n))*cos(theta_v_est(n)-theta_i_est(n));
X_est(n)=(Vm_est(n)/Im_est(n))*sin(theta_v_est(n)-theta_i_est(n));
err_R_pc(n)=(((R_est(n))-(R))/(R))*100;
err_X_pc(n)=(((X_est(n))-(X))/(X))*100;
end
% ----------------------------------------------------------------
fprintf('------------------First and Second Derivative Algo.-------------
----------\n')
fprintf('-------------------------------------------------------------------
-----------\n')
fprintf('Vm     Vm_est(n) Error in Vm  Ph Ang Estimate_of_ph  Error in Ph\n')
fprintf('-------------------------------------------------------------------
-----------\n')
for n=2:numel(Vm_est)
 fprintf('%f  %f  %f  %f  %f   %f\n',Vm,Vm_est(n),err_Vpc(n),Qvdeg, theta_v_
deg_est(n),err_theta_v_deg(n))
end
%-------------------------------------------------------------------------
fprintf('-------------------------------------------------------------------
-----------\n')
fprintf('Im     Im_est(n) Error in Im  Ph Ang Estimate_of_ph  Error in Ph\n')
fprintf('-------------------------------------------------------------------
-----------\n')
for n=2:numel(Im_est)
```

```
fprintf('%f %f %f %f %f  %f\n',Im,Im_est(n),err_ipc(n),Qi_deg, theta_i_
deg_est(n),err_theta_i_deg(n))
end
fprintf('-----------------------------------------------------------------
-----------\n')
fprintf('R      R_est   err_R_pc   X     X_est   err_X_pc   \n')
fprintf('-----------------------------------------------------------------
-----------\n')
for n=2:numel(R_est)
fprintf('%f  %f  %f  %f  %f   %f\n',R ,R_est(n), err_R_pc(n),  X, X_est(n),
err_X_pc(n)  )
end
```

%——————————————————First and Second Derivative Algo.——————————————————

Vm	Vm_est(n)	Error in Vm	Ph Ang	Estimate_of_ph	Error in Ph
100.000000	99.670624	–0.329376	30.000000	30.057713	0.192377
100.000000	99.702694	–0.297306	30.000000	30.058660	0.195534
100.000000	99.733704	–0.266296	30.000000	30.053867	0.179558
100.000000	99.760621	–0.239379	30.000000	30.043808	0.146026
100.000000	99.780814	–0.219186	30.000000	30.029469	0.098229
100.000000	99.792312	–0.207688	30.000000	30.012252	0.040841

Im	Im_est(n)	Error in Im	Ph Ang	Estimate_of_ph	Error in Ph
271.105781	270.273043	–0.307164	–54.243028	–54.302015	0.108746
271.105781	270.186930	–0.338927	–54.243028	–54.299300	0.103742
271.105781	270.108992	–0.367675	–54.243028	–54.291073	0.088573
271.105781	270.046867	–0.390591	–54.243028	–54.278134	0.064720
271.105781	270.006650	–0.405425	–54.243028	–54.261750	0.034516
271.105781	269.992288	–0.410723	–54.243028	–54.243529	0.000924

R	R_est	err_R_pc	X	X_est	err_X_pc
0.037000	0.036244	–2.042321	0.366999	0.366992	–0.001959
0.037000	0.036279	–1.948958	0.366999	0.367226	0.061793
0.037000	0.036384	–1.664476	0.366999	0.367438	0.119546
0.037000	0.036550	–1.216498	0.366999	0.367607	0.165617
0.037000	0.036760	–0.648868	0.366999	0.367717	0.195458
0.037000	0.036994	–0.017362	0.366999	0.367756	0.206118

3.4 Two-Sample Technique [25]

Let the signal be

$$v(k) = V_m \sin(\omega t_k) \tag{3.65}$$

then we can write

$$v(k+1) = V_m \sin(\omega(t_k + \Delta t)) \tag{3.66}$$

$$v(k+1) = V_m \sin(\omega t_k + \omega \Delta t) \tag{3.67}$$

$$v(k+1) = V_m \sin(\omega t_k)\cos(\omega \Delta t) + V_m \cos(\omega t_k)\sin(\omega \Delta t) \tag{3.68}$$

Using Eq. (3.65), we can substitute the value of *v(k)*

$$v(k+1) = v(k)\cos(\omega \Delta t) + V_m \cos(\omega t_k)\,\sin(\omega \Delta t) \tag{3.69}$$

The above can be simplified to:

$$\frac{v(k+1) - v(k)\cos(\omega \Delta t)}{\sin(\omega \Delta t)} = V_m \cos(\omega t_k) \tag{3.70}$$

Combining Eqs. (3.65) and (3.70), we get:

$$(v(k))^2 + \frac{(v(k+1))^2 - 2v(k)v(k+1)\cos\omega\Delta t + (v(k))^2\cos^2(\omega \Delta t)}{\sin^2(\omega \Delta t)} = V_m^2 \tag{3.71}$$

Thus, we can write V_m as:

$$V_m^2 = \frac{(v(k))^2\sin^2(\omega \Delta t) + (v(k))^2\cos^2(\omega \Delta t) + (v(k+1))^2 - 2v(k)v(k+1)\cos(\omega \Delta t)}{\sin^2(\omega \Delta t)} \tag{3.72}$$

Simplifying further, we get:

$$V_m^2 = \frac{(v(k))^2 + (v(k+1))^2 - 2v(k)v(k+1)\cos(\omega \Delta t)}{\sin^2(\omega \Delta t)} \tag{3.73}$$

Thus, V_m can finally be written as:

$$V_m = \sqrt{\frac{(v(k))^2 + (v(k+1))^2 - 2v(k)v(k+1)\cos(\omega \Delta t)}{\sin^2(\omega \Delta t)}} \tag{3.74}$$

Note that all the terms on the RHS of Eq. (3.74) are known.
Similarly,

$$I_m = \sqrt{\frac{(i(k))^2 + (i(k+1))^2 - 2i(k)i(k+1)\cos(\omega \Delta t)}{\sin^2(\omega \Delta t)}} \tag{3.75}$$

Next, we will investigate, how to find the phase angle between voltage and current.

Let
$$i(k) = I_m \sin(\omega t_k + \theta) \tag{3.76}$$
$$i(k) = I_m \sin(\omega t_k)\cos(\theta) + I_m \cos(\omega t_k)\sin(\theta) \tag{3.77}$$
$$i(k+1) = I_m \sin(\omega(t_k + \Delta t))\cos(\theta) + I_m \cos(\omega(t_k + \Delta t))\sin(\theta) \tag{3.78}$$
$$\begin{aligned} i(k+1) = {} & I_m[\sin((\omega t_k)\cos(\omega\Delta t) + \cos(\omega t_k)\sin(\omega\Delta t)]\cos(\theta) \\ & + I_m[\cos(\omega t_k(\cos(\omega\Delta t) - \sin(\omega t_k)\sin(\omega\Delta t)]\sin(\theta) \end{aligned} \tag{3.79}$$

We have $v(k) = V_m \sin(\omega t_k)$, hence,
$$\sin(\omega t_k) = \frac{v(k)}{V_m} \tag{3.80}$$

Similarly, we have shown above that:
$$\frac{v(k+1) - v(k)\cos(\omega\Delta t)}{\sin(\omega\Delta t)} = V_m \cos(\omega t_k) \tag{3.81}$$
$$\cos(\omega t_k) = \frac{v(k+1) - v(k)\cos(\omega\Delta t)}{V_m \sin(\omega\Delta t)} \tag{3.82}$$

By manipulating the above equations, we can show that:
$$\theta = \cos^{-1}\left(\frac{i(k)v(k) + i(k+1)v(k+1) - (i(k)v(k+1) + i(k+1)v(k)\cos(\omega\Delta t)}{I_m V_m \sin^2(\omega\Delta t)}\right) \tag{3.83}$$

Summary of two-sample algorithm is shown in Table 3.4.

Table 3.4 Summary of 'two-sample algorithm'

Signal model of voltage	$v(k) = V_m \sin(\omega t_k)$
Signal model of current	$i(k) = I_m \sin(\omega t_k + \theta)$
V_m	$\sqrt{\frac{(v(k))^2 + (v(k+1))^2 - 2v(k)v(k+1)\cos(\omega\Delta t)}{\sin^2(\omega\Delta t)}}$
I_m	$\sqrt{\frac{(i(k))^2 + (i(k+1))^2 - 2i(k)i(k+1)\cos(\omega\Delta t)}{\sin^2(\omega\Delta t)}}$
θ	$cos^{-1}\left(\frac{i(k)v(k) + i(k+1)v(k+1) - (i(k)v(k+1) + i(k+1))v(k)\cos(\omega\Delta t)}{I_m V_m \sin^2(\omega\Delta t)}\right)$

3.4.1 Spreadsheet for Two-Sample Technique

Note :- In this simulation phase angle of voltage is assumed to be zero as per the original paper

Set Vm	Set f_system	Set ϕv (deg)	Set Line Length km	Set R per km	Im	L per km		Set Over-sampling factor	f_samp	delta_t	Z	omega	Line Angle	Phi_I
100	50	0	100	0.036	2.763992	0.001145916		5	500	0.002	36.17955224	314.159265	84.28940686	-84.289407
		Total R =	3.6	Tot L =		0.114591559	XbyR	10		Tot R =	3.6	Tot L=	0.11459156	

Estimation of Voltage Phasor Using Two Sample Algorithm

Estimation of R and L from estimated phasors for V and I

Sample number	time	Instt Volt	v'	V'/w	Vm_est=sqrt (v^2+(v'/w)^2) BY M&M	Vm_est by Two Samp Tech	Phi_i_est by 2 Samp Tech	err_Vm%	err_Phi_i	R_estim	X_estimate	L_estim	Error in R	Error in L
0	0	0			0									
0.628319	1	0.002	58.77853	23776.41	75.68267	95.8268334	100	84.289407	1.42109E-14	0	3.6	36	0.11459156	-9.9E-14	0
1.256637	2	0.004	95.10565	9081.782	28.90821	99.40205984	100	84.289407	-1.42109E-14	-1.68596E-14	3.6	36	0.11459156	8.64E-14	-3.6E-14
1.884956	3	0.006	95.10565	-9081.78	-28.9082	99.40205984	100	84.289407	-1.42109E-14	-1.68596E-14	3.6	36	0.11459156	1.23E-13	0
2.513274	4	0.008	58.77853	-23776.4	-75.6827	95.8268334	100	84.289407	-1.42109E-14	1.68596E-14	3.6	36	0.11459156	-5.4E-13	0
3.141593	5	0.01	1.23E-14	-29389.3	-93.5489	93.54892838	100	84.289407	5.68434E-14	-3.37192E-14	3.6	36	0.11459156	4.32E-13	8.48E-14
3.769911	6	0.012	-58.7785	-23776.4	-75.6827	95.8268334	100	84.289407	-4.26326E-14	3.37192E-14	3.6	36	0.11459156	-7.4E-13	4.84E-14
4.39823	7	0.014	-95.1057	-9081.78	-28.9082	99.40205984	100	84.289407	-1.42109E-14	-1.68596E-14	3.6	36	0.11459156	1.23E-13	0
5.026548	8	0.016	-95.1057	9081.782	28.90821	99.40205984	100	84.289407	4.26326E-14	1.01158E-13	3.6	36	0.11459156	-1.7E-12	-1.2E-14
5.654867	9	0.018	-58.7785	23776.41	75.68267	95.8268334	100	84.289407	-1.27898E-13	-6.74384E-14	3.6	36	0.11459156	6.54E-13	-1.3E-13
6.283185	10	0.02	-2.5E-14	29389.26	93.54893	93.54892838	100	84.289407	0	-1.68596E-14	3.6	36	0.11459156	1.23E-13	0
6.911504	11	0.022	58.77853	23776.41	75.68267	95.8268334	100	84.289407	4.26326E-14	-1.18017E-13	3.6	36	0.11459156	1.44E-12	-1.2E-14
7.539822	12	0.024	95.10565	9081.782	28.90821	99.40205984	100	84.289407	-4.26326E-14	0	3.6	36	0.11459156	-2.7E-13	-1.7E-13
8.168141	13	0.026	95.10565	-9081.78	-28.9082	99.40205984	100	84.289407	-8.52651E-14	-2.19175E-13	3.6	36	0.11459156	3.1E-12	6.06E-14
8.796459	14	0.028	58.77853	-23776.4	-75.6827	95.8268334	100	84.289407	-1.42109E-14	0	3.6	36	0.11459156	-6.2E-14	4.84E-14
9.424778	15	0.03	3.68E-14	-29389.3	-93.5489	93.54892838	100	84.289407	-1.42109E-14	-1.01158E-13	3.6	36	0.11459156	1.25E-12	0
10.0531	16	0.032	-58.7785	-23776.4	-75.6827	95.8268334	100	84.289407	4.26326E-14	-1.01158E-13	3.6	36	0.11459156	1.31E-12	6.06E-14
10.68142	17	0.034	-95.1057	-9081.78	-28.9082	99.40205984	100	84.289407	-1.42109E-14	3.37192E-14	3.6	36	0.11459156	-7.6E-13	2.42E-14
11.30973	18	0.036	-95.1057	9081.782	28.90821	99.40205984	100	84.289407	-5.68434E-14	-1.18017E-13	3.6				
11.93805	19	0.038	-58.7785	23776.41	75.68267	95.8268334	100	84.289407	0	-1.01158E-13					
12.56637	20	0.04	-4.9E-14		4.90059E-14										

Estimation of Current Phasor Using Two Sample Algorithm

	Instt Current	I'	I'/w	Im_est=sqrt (i^2+(i'/w)^2) BY M&M	Im_est by 2 Samp Tech	err_Im%
0	-2.75028					
1	-2.06336	540.49	1.720433	2.686513231	2.7639922	1.60669E-14
2	-0.58832	793.7026	2.526434	2.594028358	2.7639922	3.21339E-14
3	1.111448	743.7478	2.367423	2.615341055	2.7639922	-1.60669E-14
4	2.386676	409.7067	1.304137	2.719742031	2.7639922	-1.60669E-14
5	2.750275	-80.8286	-0.25729	2.762283194	2.7639922	-1.60669E-14
6	2.063362	-540.49	-1.72043	2.686513231	2.7639922	-6.42678E-14
7	0.588315	-793.703	-2.52643	2.594028358	2.7639922	0
8	-1.11145	-743.748	-2.36742	2.615341055	2.7639922	8.03347E-14
9	-2.38668	-409.707	-1.30414	2.719742031	2.7639922	1.60669E-14
10	-2.75028	80.82856	0.257285	2.762283194	2.7639922	1.60669E-14
11	-2.06336	540.49	1.720433	2.686513231	2.7639922	6.42678E-14
12	-0.58832	793.7026	2.526434	2.594028358	2.7639922	1.44603E-13
13	1.111448	743.7478	2.367423	2.615341055	2.7639922	-1.60669E-13
14	2.386676	409.7067	1.304137	2.719742031	2.7639922	-4.82008E-14
15	2.750275	-80.8286	-0.25729	2.762283194	2.7639922	-1.60669E-14
16	2.063362	-540.49	-1.72043	2.686513231	2.7639922	-1.60669E-14
17	0.588315	-793.703	-2.52643	2.594028358	2.7639922	0
18	-1.11145	-743.748	-2.36742	2.615341055	2.7639922	1.60669E-14
19	-2.38668	-409.707	-1.30414	2.719742031	2.7639922	-1.60669E-14
20	-2.75028					

Figure 3.9 Spreadsheet for two-sample algorithm.

3.4.2 MATLAB Simulation of Two-Sample Technique

```
% ------------------------------------------------------------
% Two-Sample Algorithm
% ------------------------------------------------------------
clear ; clc ; clf

%================================================
mag_z_set      = 30   ; % Relay Reach Setting
theta_n       = pi/2  ; % Relay MTA setting
%================================================
%============ Numerical Problem Setting ==================
f             = 50   ; % System frequency
OverSamplingFactor = 20   ; % fsamp = 2 * OSF * fsig
km            = 100  ; % kilometers
R             = 0.037 ; % ohms per km for 345kV line
XbyR_ratio    = 9.9189 ; % for 345 kV line
%================================================
f_samp        = 2*OverSamplingFactor*f      ;
X             = R*XbyR_ratio           ;
L             = X/(2*pi*f)            ;
Z             = sqrt( R*R + X* X)         ;
line_ang_deg  = (180/pi)*atan2(X,R)        ;
w             = 2* pi* f              ;
T             = 1/f                  ;
dT            = 1/f_samp               ;
Vm            = 100                  ;
Im            = Vm / Z                ;
Qv_deg        = 0                   ;% Theta_V = 0
Qv            = Qv_deg*(pi/180)          ;
Qi            = Qv - atan2(X,R)          ;
Qi            = -1 * Qi                      ;  %In the algorithm ph. angle of
                                                    current is taken positive
Qvdeg         = Qv * (180/pi)           ;
Qi_deg        = Qi*(180/pi)             ;
k             = 0 ;
sss           =50;
%========== Generation of Samples ================
 for tim   = 0:dT: T

   k       = k + 1                          ;
   t(k)    = tim                            ;
   v(k)    = Vm * sin( 2* pi* f * t(k) )              ;

   i(k)    = Im * sin( (2* pi* f * t(k) ) + Qi)          ;
% both +Qi and -Qi give +Qi as angle
% it is not possible to distinguish bet. lead or lag

 end
```

```
% ================================================================
samples    = k                              ;
for n = 1:samples-1
%---------- Estimation of voltage phasor ------------------------------
Vm_est(n)=sqrt((((v(n))^2)+((v(n+1))^2)-(2*v(n)*v(n+1)*cos(w*dT)))/
((sin(w*dT))^2));
Im_est(n)=sqrt((((i(n))^2)+((i(n+1))^2)-(2*i(n)*i(n+1)*cos(w*dT)))/
((sin(w*dT))^2));
%-----------------------------------------------------------------
%---------- Estimation of Current phasor ------------------------------
num=(i(n)*v(n))+(i(n+1)*v(n+1))-(((i(n)*v(n+1))+(i(n+1)*v(n)))*cos(w*
dT));
den=(Im_est(n)*Vm_est(n)*((sin(w*dT))^2));
Ph_i_est(n)=acos(num/den);
Ph_i_est_deg(n)=(Ph_i_est(n))*(180/pi);
%-----------------------------------------------------------------
%-------- Error in estimation of voltage and current phasors
err_Vm(n)=(((Vm_est(n))-(Vm))/(Vm))*100;
err_Im(n)=(((Im_est(n))-(Im))/(Im))*100;
err_Ph_i_deg(n)=(((Ph_i_est_deg(n))-(Qi_deg))/(Qi_deg))*100;
 R_est(n)=(Vm_est(n)/Im_est(n))*cos(Ph_i_est(n));
 X_est(n)=(Vm_est(n)/Im_est(n))*sin(Ph_i_est(n));
 err_R_pc(n)=(((R_est(n))-(R))/(R))*100;
 err_X_pc(n)=(((X_est(n))-(X))/(X))*100;
end
% ----------------------------------------------------------------
fprintf('------------------Two Sample Technique  -----------------------
\n')
fprintf('-----------------------------------------------------------------
------------\n')
fprintf('Vm      Vm_est(n) Error in Vm  \n')
fprintf('-----------------------------------------------------------------
------------\n')
for n=1:numel(Vm_est)
 fprintf('%f  %f  %f  \n',Vm,Vm_est(n),err_Vm(n))
end
%----------------------------------------------------------------
fprintf('-----------------------------------------------------------------
------------\n')
fprintf('Im     Im_est(n) Error in Im  Ph Ang Estimate_of_ph  Error in Ph\n')
fprintf('-----------------------------------------------------------------
------------\n')
for n=2:numel(Im_est)
 fprintf('%f  %f  %f  %f  %f   %f\n',Im,Im_est(n),err_Im(n),Qi_deg,Ph_i_est_
deg(n),err_Ph_i_deg(n))
end
fprintf('-----------------------------------------------------------------
------------\n')
```

```
fprintf('R      R_est   err_R_pc   X     X_est   err_X_pc   \n')
fprintf('------------------------------------------------------------------
-----------\n')
for n=2:numel(R_est)
fprintf('%f  %f  %f  %f  %f   %f\n',R ,R_est(n), err_R_pc(n),  X, X_est(n),
err_X_pc(n)  )
end
```

Two Sample Technique.

Vm	Vm_est(n)	Error in Vm
100.000000	100.000000	0.000000
100.000000	100.000000	–0.000000
100.000000	100.000000	0.000000
100.000000	100.000000	0.000000
100.000000	100.000000	0.000000
100.000000	100.000000	–0.000000
100.000000	100.000000	–0.000000
100.000000	100.000000	0.000000
100.000000	100.000000	0.000000
100.000000	100.000000	–0.000000

Im	Im_est(n)	Error in Im	Ph Ang	Estimate _of_ph	Error in Ph
271.105781	271.105781	–0.000000	84.243028	84.243028	0.00
271.105781	271.105781	0.000000	84.243028	84.243028	0.00
271.105781	271.105781	–0.000000	84.243028	84.243028	0.00
271.105781	271.105781	0.000000	84.243028	84.243028	0.00
271.105781	271.105781	0.000000	84.243028	84.243028	0.00
271.105781	271.105781	–0.000000	84.243028	84.243028	0.00
271.105781	271.105781	–0.000000	84.243028	84.243028	0.00
271.105781	271.105781	0.000000	84.243028	84.243028	0.00
271.105781	271.105781	–0.000000	84.243028	84.243028	0.00
271.105781	271.105781	–0.000000	84.243028	84.243028	0.00

R	R_est	err_R_pc	X	X_est	err_X_pc
0.037000	0.037000	–0.000000	0.366999	0.366999	0.000000
0.037000	0.037000	0.000000	0.366999	0.366999	–0.000000
0.037000	0.037000	0.000000	0.366999	0.366999	0.000000
0.037000	0.037000	–0.000000	0.366999	0.366999	–0.000000
0.037000	0.037000	0.000000	0.366999	0.366999	–0.000000
0.037000	0.037000	0.000000	0.366999	0.366999	–0.000000
0.037000	0.037000	0.000000	0.366999	0.366999	0.000000
0.037000	0.037000	0.000000	0.366999	0.366999	0.000000
0.037000	0.037000	–0.000000	0.366999	0.366999	–0.000000
0.037000	0.037000	0.000000	0.366999	0.366999	–0.000000

REVIEW QUESTIONS

1. The voltage and current samples at a relay location are shown below. The system frequency is known to be 50 Hz and the sampling frequency is 10 kHz. Using Mann and Morrison algorithm find the apparent resistance and reactance as seen from the relay location.

Sample set no. 1

	v	*i*
(i)	55.000000	–8.887405
(ii)	57.965138	–8.682345
(iii)	60.873070	–8.468716
(iv)	63.720929	–8.246729

Sample set no. 2

(i)	66.505903	–8.016604
(ii)	69.225243	–7.778567
(iii)	71.876266	–7.532854
(iv)	74.456357	–7.279707

Sample set no. 3

(i)	76.962967	–7.019376
(ii)	79.393625	–6.752117
(iii)	81.745931	–6.478195
(iv)	84.017563	–6.197880

Sample set no. 4

(i)	86.206280	–5.911448
(ii)	88.309922	–5.619183
(iii)	90.326413	–5.321371
(iv)	92.253762	–5.018309

Sample set no. 5

(i)	94.090069	–4.710293
(ii)	95.833519	–4.397630
(iii)	97.482394	–4.080626
(iv)	99.035065	–3.759595

Sample set no. 6

(i)	100.490000	–3.434855
(ii)	101.845764	–3.106724
(iii)	103.101019	–2.775527
(iv)	104.254525	–2.441591

2. What are the assumptions in the Mann and Morrison algorithm?

3. Derive the Mann and Morrison algorithm.

4. What are the drawbacks of using numerical derivative?

5. Assuming that amplitude of signal is known, can you find its frequency by using an algorithm similar to Mann and Morrison algorithm.

6. An ac signal $v = V_m \sin(2\ \pi ft + \phi)$ has three unknown V_m, f and ϕ. Thus, we need three equations to solve for the three unknown. Using this idea can you derive an algorithm?

7. Three-sample algorithm assumes a sine waveform. Derive three-sample algorithm when cosine waveform is taken as reference.

8. What could be the applications of three-sample algorithm in general and in the field of power system protective relaying?

9. Discuss the importance of measurement of power system frequency.

10. What are the assumptions underlying three-sample algorithm?

11. What is the minimum number of samples required for making one estimate each of amplitude, phase and frequency, assuming the signal to be undistorted sine wave?

12. What happens to the estimation using three-sample algorithm if one of the samples happens to be zero or very low in magnitude?

13. Compare and contrast three-sample algorithm with Mann and Morrison algorithm.

14. Can three-sample algorithm be applied under non-stationary conditions?

15. Numerical problem on three-sample technique:

(i) The samples of a single frequency signal are as shown below:
Make at least three estimates of the amplitude,phase and frequency

0	1	2	3	4	5	6
56.642460	26.917181	–5.442939	–37.270267	–65.449321	–87.221739	–100.456286

==========

The sampling frequency is 1003.90002 Hz.

(ii) The samples of a single frequency signal are as shown below:
Make at least three estimates of the amplitude,phase and frequency

0	1	2	3	4	5	6
–5.442939	-37.270267	–65.449321	–87.221739	–100.456286	–103.857472	–97.092364

==========

The sampling frequency is 1003.90002 Hz.

16. Derive the 'first derivative and second derivative algorithm'.

17. What are the assumptions underlying the first and second derivative algorithm?

18. What could be the motivation for such algorithms involving higher derivatives?

19. Derive the 'two-sample algorithm'. What are the advantages of using this algorithm?

20. What are the assumptions underlying the 'two-sample algorithm'?

21. Develop a spreadsheet for simulating the first and second derivative and two-sample algorithm.

22. Write MATLAB programs for simulating the first and second derivative algorithm and two-sample algorithm.

4

Algorithms Based on Solution of Differential Equation

4.1 Differential Equation Algorithm [20, 47, 56]

In this algorithm, we model the transmission line as a series lumped R–L circuit as shown in Figure 4.1 for a single phase line with zero fault resistance (bolted fault).

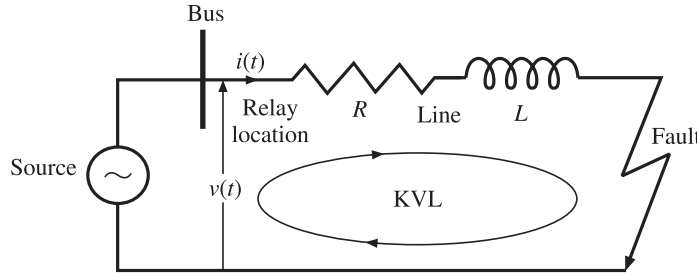


Figure 4.1 Series lumped *R-L* model of faulted transmission line.

It may be noted that we have ignored the shunt capacitance of the line and it is assumed that fault resistance is zero. Note that $v(t)$ and $i(t)$ are the voltage and current available at the relay location through CT and PT which are not shown in Figure 4.1. Hence, both the CT and the PT ratio may be considered to be equal to 1. Figure 4.2 shows the sampling instants and sampled values of voltage and current.

If we apply KVL around the loop formed because of fault, at instant t_k , we can write:-

$$v(t_k) = R\,i(t_k) + L\frac{di(t_k)}{dt}$$

It can be easily seen that in the above differential equation, the knowns are $v(t_k)$ and $i(t_k)$, and the unknowns are R and L, i.e., the resistance and the inductance of the transmission line from relay location upto the fault point. The rate of change of current term, even though

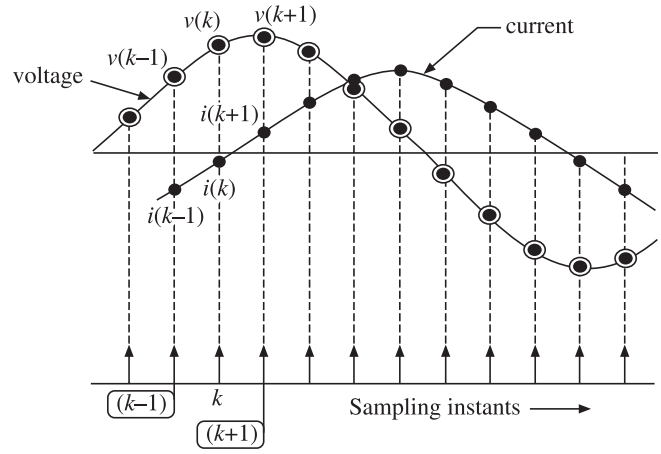


Figure 4.2 Voltage and current samples.

not measured directly, can be computed from the current measurements. In order to solve for the two unknowns, we need two equations. The second equation can be generated by applying KVL at another instant of time say $(k + 1)^{th}$ instant as shown below:

$$v(t_k) = Ri(t_k) + L\frac{di(t_k)}{dt} \tag{4.1}$$

$$v(t_{k+1}) = Ri(t_{k+1}) + L\frac{di(t_{k+1})}{dt} \tag{4.2}$$

Let us represent $di(t)/dt$ by $i'(t)$ and rewrite Eqs. (4.1) and (4.2) as:

$$v(k) = Ri(k) + Li'(k) \tag{4.3}$$

$$v(k+1) = Ri(k+1) + Li'(k+1) \tag{4.4}$$

Note that when the numerical values of the derivative of the current are inserted in the above equations, they are no longer differential equations but become simultaneous algebraic equations with constant coefficients.

We can write the above two equations in matrix form as:

$$\begin{bmatrix} v(k) \\ v(k+1) \end{bmatrix} = \begin{bmatrix} i(k) & i'(k) \\ i(k+1) & i'(k+1) \end{bmatrix} \begin{bmatrix} R \\ L \end{bmatrix} \tag{4.5}$$

Which gives us:

$$\begin{bmatrix} R \\ L \end{bmatrix} = \begin{bmatrix} i(k) & i'(k) \\ i(k+1) & i'(k+1) \end{bmatrix}^{-1} \begin{bmatrix} v(k) \\ v(k+1) \end{bmatrix} \tag{4.6}$$

$$\begin{bmatrix} R \\ L \end{bmatrix} = \frac{1}{i(k)i'(k+1) - i'(k)i(k+1)} \begin{bmatrix} i'(k+1) & -i'(k) \\ -i(k+1) & i(k) \end{bmatrix} \begin{bmatrix} v(k) \\ v(k+1) \end{bmatrix} \tag{4.7}$$

Finally we can express R and L as a function of various samples of voltage and current and computed derivatives of current

$$R = \frac{i'(k+1)v(k) - i'(k)v(k+1)}{i(k)i'(k+1) - i'(k)i(k+1)} \tag{4.8}$$

$$L = \frac{i(k)v(k+1) - i(k+1)v(k)}{i(k)i'(k+1) - i'(k)i(k+1)} \tag{4.9}$$

Note that the numerical derivatives can be calculated using central difference formula as:

$$v'(k) = \frac{v(k+1) - v(k-1)}{2\Delta t} \tag{4.10}$$

$$i'(k) = \frac{i(k+1) - i(k-1)}{2\Delta t} \tag{4.11}$$

where Δt is the sampling time period given by $1/f_{\text{sampling}}$.

The block-diagram showing the input to the algorithm and the block required for implementation of specific distance relay characteristics is shown in Figure 4.3.

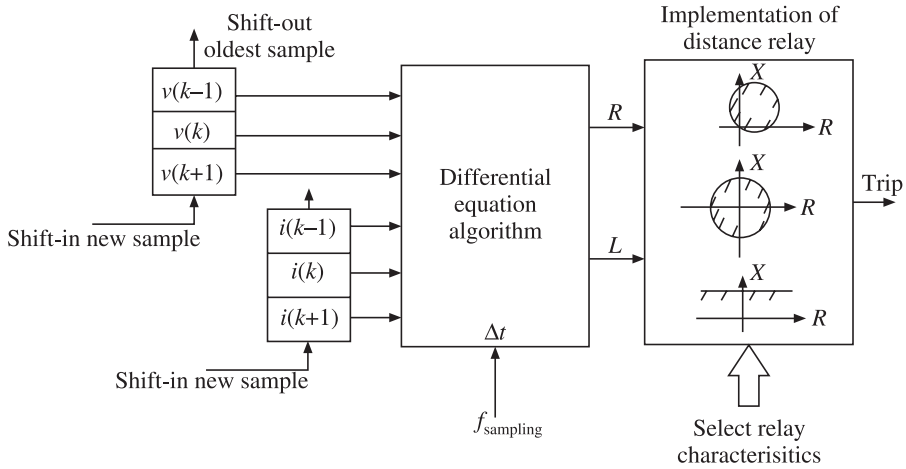


Figure 4.3 Block-diagram showing inputs to differential equation algorithm and block for implementation of specific distance relay.

It can be seen from Figure 4.3 that after estimating the apparent *R* and *L* as seen from the relay location, we will have to check whether the apparent impedance lies within the trip region of the relay characteristics being implemented. Thus, in addition to the differential equation algorithm, we will have to write another algorithm for implementing specific distance relay characteristics.

4.2 Justification for Lumped Series R-L Model

Before we proceed further, let us find out how far we are justified in representing the transmission line as a lumped series *R-L* circuit. Let us calculate the shunt and series parameters of a length

of 100 km, 345 kV line. We will use the data from Table 4.1 which happens to be for a 60 Hz power system.

Table 4.1 Data of a 60 Hz power system

	230 kV	345 kV	500 kV	765 kV	1100 kV
R Ω/km	0.050	0.037	0.028	0.012	0.005
X_L	0.488	0.367	0.326	0.329	0.292
X_L/R	9.76	9.9189	11.6428	27.4166	58.4
$\tan^{-1}(x_L/\text{R})$ deg	84.15	84.2430	85.09	87.91	89.019
S μs/km	3.371	4.518	5.20	4.978	5.544
Z_c	380	285	250	257	230

Succeptance, $S = 2\pi f\text{C} = (2)(\pi)(60)(\text{C}) = 4.518 \times 10^{-6}$ S/km

Hence, $$C = \frac{4.518 \times 10^{-6}}{(2)(\pi)(60)} = 0.01198436721 \text{ micro-farad } (\mu\text{F})$$

Consider 100 km length of line, with total capacitance lumped at one end.

$$C_{\text{total}} = (100)\ (0.01198436721)\ \mu\text{F} = 1.2\ \mu\text{F}$$

$$Z_{\text{shunt}} = X_C = \frac{1}{j\omega C} = \frac{1}{j((2)(\pi)60(1.2 \times 10^{-6}))} = -j2210.486\ \Omega$$

For π model, ½ of shunt capacitance is connected at each end. Hence, X_C becomes double, i.e., $-j$ 4420.97 Ω.

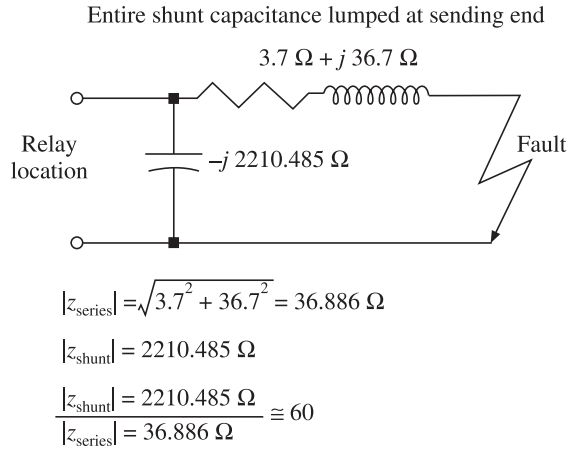


Figure 4.4 Entire shunt capacitance modelled at sending end with fault on.

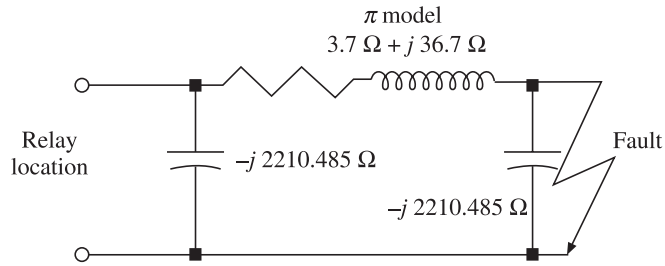


Figure 4.5 π model of the transmission line with fault on.

Table 4.2

Consider of the 100 km length of the 345 kV line modelled by:		
Series lumped model	Total capacitance lumped at sending end	π model. with ½ capacitance lumped at each end
$X_L = (100)\,(0.367) = j\,36.7\ \Omega$ $R = (100)\,(0.037) = 3.7\ \Omega$ $Z_{seen} = (3.7 + j\,36.7)\ \Omega$	$Z_{seen} = (3.7 + j36.7)\|\|(-j\,2210.486)$ $= \dfrac{(3.7 + j\,36.7)\times(-j\,2210.486)}{(3.7 + j\,36.7)(-j\,2210.486)}$ $= (3.825 + j\,37.313)\ \Omega$	$Z_{seen} = (3.7 + j36.7)\|\|(-j\,4420.97)$ $= \dfrac{(3.7 + j36.7)\times(-j\,4420.97)}{(3.7 + j36.7)(-j\,4420.97)}$ $= (3.7622 + j37.00403)\ \Omega$
$\|Z_{seen}\|$=36.886 ∠84.243°	Z_{seen} = 37.5087 ∠84.1455°	Z_{seen} = 37.194∠84.19468°
	% Error in lumped *R-L* model w.r.t this model $\left[\dfrac{(36.886 - 37.5087)}{(37.5087)}\right]100 = -1.66\ \%$	% Error in lumped *R-l* model w.r.t. this model $\left[\dfrac{(36.886 - 37.194)}{(37.194)}\right]100$ $= -0.8280\ \%$

Thus, we see that the error introduced because of ignoring shunt capacitance is less than 1%, which is not at all serious. Hence, our representation of the transmission line as a series *R-L* circuit is justified.

4.3 Excel Spreadsheet Implementation of the Differential Equation Algorithm

Spreadsheet for Differential Equation Algorithm

							Setting	Setting								
v(t) = Vm*sin(w* t + Phi_V) = (Vm / Z)*sin(w *t + Phi_V - LineAngle)						i(t)	Vm	R	X	L	w=2 pi f L	Z	LineAngle (rad)	LineAngle (deg)	Im
							400	7	70	0.2228	314.15927	70.349	1.47112767	84.289407	5.685927
Sampling Freq		fs= 5000								Set==>	X/R ==>	10	Phi-V(rad)	Phi_V(deg)	
Sample No. ==========>	1	2	3	4	5	6	7	8					0.31415927	18	
Time ==========>	t0	t1	t2	t3	t4	t5	t6	t7							
(seconds) ==========>	0	0.0002	0.0004	0.0006	0.0008	0.001	0.0012	0.0014							
v(t) = Vm*sin(w* t + Phi_V)	123.6068	147.25	170.31	192.7	214.33	235.11	254.97	273.82							
i(t)=Im*sin(wt+phi_v-line_ang)	-5.205968	-5.0521	-4.8784	-4.6853	-4.4738	-4.2446	-3.9987	-3.737							
i'(k)=i(k+1)-i(k-1)/2.dT		819.03	917.01	1011.4	1101.7	1187.8	1269.1								
Computed L	L ===>	0.223	0.223	0.223	0.223	0.223									
Computed X	X===>	70.046	70.046	70.046	70.046	70.046									
Computed R	R===>	7	7	7	7	7									
	Num1=	I'(k+1) v(k)-I'(k) v(k+1)			Num2=	i(k)v(k+1)-i(k+1)v(k)									
		Den =	I(k) I'(k+1)-I(k+1)I'(k)												
	R=Num1/Den			I=Num2/Den											

Figure 4.6 Excel spreadsheet implementation of differential equation algorithm.

4.4 MATLAB Implementation of Differential Equation Algorithm

```
% ----------------------------------------------------------------
% Numerical Problem on Differential Equation Algorithm
% ----------------------------------------------------------------

clear ; clc ; clf
%================================================================
mag_z_set     = 30   ; % Relay Reach Setting
theta_n       = pi/2  ; % Relay MTA setting
%================================================================
%============= Numerical Problem Setting ========================
f             = 50    ; % System frequency
OverSamplingFactor = 2000  ; % fsamp = 2 * OSF * fsig
km            = 100   ; % kilometers
R             = 0.037  ; % ohms per km for 345kV line
XbyR_ratio    = 9.9189 ; % for 345 kV line
%================================================================

f_samp        = 2*OverSamplingFactor*f     ;
X             = R*XbyR_ratio               ;
L             = X/(2*pi*f)                  ;
Z             = sqrt( R*R + X* X)          ;
line_ang_deg  = (180/pi)*atan2(X,R)          ;
w             = 2* pi* f                    ;
T             = 1/f                            ;
dT            = 1/f_samp                    ;
Vm            = 400*sqrt(2)                 ;
Im            = Vm / Z                     ;
Qv_deg        = -15                         ;
Qv            = Qv_deg*(pi/180)             ;
Qi            = Qv - atan2(X,R)            ;

Qvdeg         = Qv * (180/pi)               ;
Qideg         = Qi*(180/pi)                 ;

k             = 0                          ;
sss           = 50                          ;
 for tim      =  0:dT: T
   k          =  k + 1                     ;
   t(k)       = tim                         ;
   v(k)       = Vm * sin( 2* pi* f * t(k) + Qv)  ;
   i(k)        = Im * sin( 2* pi* f * t(k) + Qi)   ;
 end

samples       = k                           ;
m             = 0                           ;
for n         =  2:samples-1
  m           =  m+1                                ;
```

```
  i_dash(m)   =  ( i(n+1)-i(n-1)) / (2*dT)                ;
end

fprintf('================================================ \n')
disp('Numerical Problem')
fprintf(' A Mho Relay with following settings is to be implemented \n') ;
fprintf(' Z Setting = %f Angle Z setting = %f\n', mag_z_set, (180/pi)* theta_n)
fprintf(' Use differential algorithm to make two estimates apparent R and L
\n')
fprintf(' Given that the system freq is    %f \n',f )
fprintf(' and given that the sampling freq is %f \n',f_samp )
%fprintf('\n Sampling Interval =%f  Sampling Freq = %f\n ', dT , 1/dT)
%-------------------------------------------------------------------------
--
fprintf('The Voltage and Current Samples are :- \n');
fprintf('Voltage Samples ')
fprintf('\n=================================================\n');
nn     =8                        ;
for   kk   = 1 :  nn
  fprintf( '%f | ',v(kk))             ;
end
fprintf('\n=================================================\n');
fprintf('\nCurrent Samples')
fprintf('\n=================================================\n');
for kk = 1 :nn
  fprintf('%f | ',i(kk))                ;
end
fprintf('\n=================================================\n');

% ===== The DEA Algorithm
% den           = i(k)i'(k+1)-i(k+1)i'(k)
% R(k)          = i'(k+1)v(k)- i'(k)v(k+1)/den
% L(k)          = i(k)v(k+1) - i(k+1)v(k) /den
%(numel(v))-4
num_calc       = 8                                          ;

for k=2:num_calc
 den(k)        = (i(k)*i_dash(k+1) )- ( i(k+1)*i_dash(k) )        ;
 RR(k)         = ((i_dash(k+1)*v(k))-(i_dash(k)*v(k+1)))/(den(k)) ;
 LL(k)         = ((i(k)*v(k+1))-(i(k+1)*v(k)))/(den(k))           ;
 err_R_pc(k)   = (((RR(k))-(R))/(R))*100                    ;
  err_L_pc(k)  = (((LL(k))-(L))/(L))*100                     ;
end

 R,RR,err_R_pc
 L,LL,err_L_pc

%---------      Printout the true values      -------------------
fprintf('\n R       = %f  X = %f  Z = % f ',  R,X,Z     )  ;
```

```
fprintf(' Line Angle = %f          \n', line_ang_deg    )   ;
fprintf(' X/R ratio  = %5.2f        \n ', X/R          )   ;
%-------------------------------------------------------------------_
```

```
=============================================================================
Numerical Problem
 A mho relay with following settings is to be implemented:
 Z setting = 30.000000 and Angle Z setting = 90.000000
 Use differential algorithm to make two estimates apparent R and L.
 Given that the system frequency = 50.000000 and the sampling frequency = 200000.000000
The Voltage and Current Samples are :–
Voltage Samples
=============================================================================
–146.410162 | –145.551682 | –144.692844 | –143.833648 | –142.974098 | –142.114195 | –141.253941 | –140.393338 |
=============================================================================
Current samples
=============================================================================
–1513.693429 | –1514.078498 | –1514.459831 | –1514.837427 | –1515.211286 | –1515.581406 | –1515.947787 |
–1516.310427 |
=============================================================================
R =  0.0370

RR =      0     0.0376     0.0376     0.0376     0.0376     0.0376     0.0376     0.0376

err_R_pc = 0     1.5581     1.5581     1.5581     1.5581     1.5581     1.5581     1.5581

L =  0.0012
LL = 0     0.0012     0.0012     0.0012     0.0012     0.0012     0.0012     0.0012
err_L_pc = 1.0e–003 *

     0     0.1645     0.1645     0.1645     0.1645     0.1645     0.1645     0.1645

 R   = 0.037000,    X = 0.366999,    Z = 0.368860,    Line angle = 84.243028
 X/R ratio = 9.92
```

4.5 Solution of Differential Equation Algorithm Using Numerical Integration

We use the same model of the faulted transmission line, as before, as shown in Figure 4.7. We have already seen that the differential equation relating the voltage and current at the relay location $v(t) = Ri(t) + L\,di(t)/dt$ gets converted into an algebraic equation when we substitute the numerical value of the derivative term. However, a derivative has the problem that it tends to amplify noise because of its high-pass nature. This is very easy to see. Let the signal be

$$v = \boldsymbol{V_{m,\,signal}}\ \sin(\omega_{\text{signal}}t) + V_{m,\text{noise}}\ \sin(\omega_{\text{noise}}t) \tag{4.12}$$

Note that $\omega_{\text{noise}} >> \omega_{\text{signal}}$ and $V_{m,\,\text{noise}} << \boldsymbol{V_{m,\,signal}}$

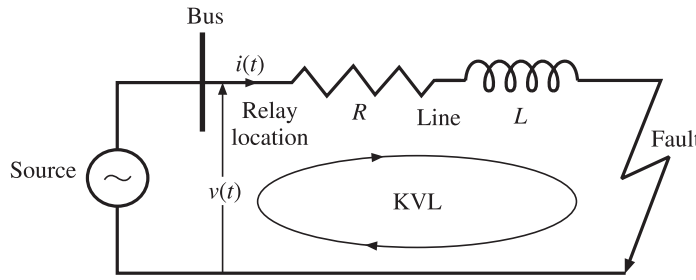


Figure 4.7 Model of faulted transmission line.

Thus, the noise amplitude may be smaller than the signal amplitude to start with, but since the noise frequency is many orders of magnitude higher than the signal frequency, after taking derivative it will dominate over the signal as can be seen from the expression for derivative,

$$v' = (\omega \boldsymbol{V_m}) \cos(\omega_{\text{signal}} t) + (\omega_{\textbf{noise}} \boldsymbol{V_{m,noise}}) \cos(\omega_{\text{noise}} t) \tag{4.13}$$

Look at the amplitude of the noise. It is amplified by 'ω_{noise}'. This is problematic because real life signals are always replete with noise. Thus, in the process of differentiation the noise will get amplified. This may happen so much so that many times the noise will mask out the signal of interest. Thus, wherever possible we should try to avoid taking derivative. This leads to implementation of the differential equation algorithm using numerical integration. We start with the basic equation relating the relay voltage and current:

$$v = Ri + L\frac{di}{dt} \tag{4.14}$$

Integrating both sides between two instants of time t_1 and t_2 and then between t_3 and t_4, we get:

$$\int_{t_1}^{t_2} v(t)dt = R\int_{t_1}^{t_2} i(t)dt + L\int_{t_1}^{t_2} \frac{di(t)}{dt}dt \tag{4.15}$$

$$\int_{t_3}^{t_4} v(t)dt = R\int_{t_3}^{t_4} i(t)dt + L\int_{t_3}^{t_4} \frac{di(t)}{dt}dt \tag{4.16}$$

Which can be written as:

$$\int_{t_1}^{t_2} v(t)dt = R\int_{t_1}^{t_2} i(t)dt + L\int_{t_1}^{t_2} di(t)dt \tag{4.17}$$

$$\int_{t_3}^{t_4} v(t)dt = R\int_{t_3}^{t_4} i(t)dt + L\int_{t_3}^{t_4} di(t)dt \tag{4.18}$$

Note that the last term in each equation does not involve finding the numerical integration since it reduces simply to:

$$\int_{t_1}^{t_2} di(t)dt = i(t)_{t_1}^{t_2} = i(t_2) - i(t_1) \tag{4.19}$$

$$\int_{t_3}^{t_4} di(t)dt = i(t)_{t_3}^{t_4} = i(t_4) - i(t_3) \tag{4.20}$$

Note, further, that the other integrals can be evaluated numerically. Thus, when we substitute the numerical values of the integrals, we get algebraic equations. To simplify the derivation further, let us use the following representation for the numeric values of the integrals:

$$\int_{t_1}^{t_2} v(t)dt = E \qquad \int_{t_1}^{t_2} i(t)dt = A \qquad \int_{t_1}^{t_2} di(t)dt = B \tag{4.21}$$

$$\int_{t_3}^{t_4} v(t)dt = F \qquad \int_{t_3}^{t_4} i(t)dt = C \qquad \int_{t_3}^{t_4} di(t)dt = D \tag{4.22}$$

Thus, we can write the two differential equations as the following two simultaneous algebraic equations with constant coefficients:

$$E = R\,A + L\;B \tag{4.23}$$

$$F = R\,C + L\,D \tag{4.24}$$

Let us represent the system of equations in matrix form as shown below:

$$\begin{bmatrix} E \\ F \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} R \\ L \end{bmatrix} \tag{4.25}$$

Thus, we can find the unknowns R and L as follows:

$$\begin{bmatrix} R \\ L \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} \begin{bmatrix} E \\ F \end{bmatrix} \tag{4.26}$$

$$\begin{bmatrix} R \\ L \end{bmatrix} = \frac{1}{(AD - BC)} \begin{bmatrix} D & -B \\ -C & A \end{bmatrix} \begin{bmatrix} E \\ F \end{bmatrix} \tag{4.27}$$

Solving the above, we get:

$$R = \frac{(DE - BF)}{(AD - BC)} \tag{4.28}$$

and

$$L = \frac{(AF - CE)}{(AD - BC)} \tag{4.29}$$

Note that A, C, E and F are the values returned by the numerical integrals defined above.

4.6 MATLAB Implementation of Solution of Differential Equation Algorithm Using Numerical Integration

We can implement the numerical integration using a number of numerical techniques. However we restrict ourselves to the two very well-known techniques viz., trapezoidal rule and the famous Simpson's rule of integration. We have to appreciate that numerical integration , like any other numerical method, is not perfect. It involves errors. Depending upon how far the waveform being integrated conforms to the underlying assumption on which the algorithm is based, the errors will be more or less. Next we take up the derivation of the two methods of numerical integration.

4.6.1 Trapezoidal Rule for Numerical Integration

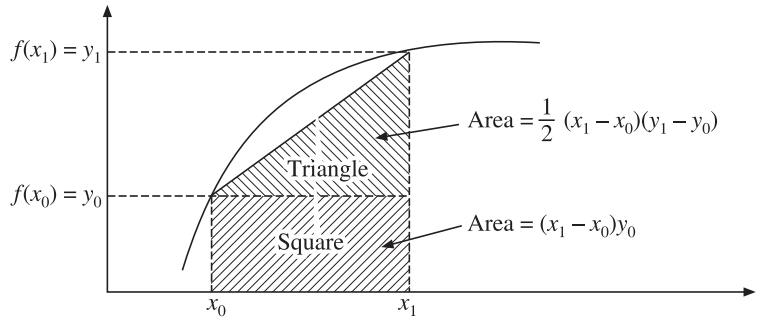


Figure 4.8 Trapezoidal rule of integration.

Referring to Figure 4.8, the area under the curve from x_0 to x_1 is

$$\int_{x_0}^{x_1} f(x)dx \cong \text{Area of square} + \text{Area of traiangle} \tag{4.30}$$

$$= (x_1 - x_0)y_0 + \frac{1}{2}(x_1 - x_0)(y_1 - y_0) \tag{4.31}$$

$$= (x_1 - x_0)y_0 - \frac{1}{2}(x_1 - x_0)y_0 + \frac{1}{2}(x_1 - x_0)y_1 \tag{4.32}$$

$$= \frac{1}{2}(x_1 - x_0)y_0 + \frac{1}{2}(x_1 - x_0)y_1 \tag{4.33}$$

$$= \frac{1}{2}((x_1 - x_0)(y_0 + y_1)) \tag{4.34}$$

substituting $x_1 - x_0 = h$ we get:

$$= \frac{h}{2}(y_0 + y_1) \tag{4.35}$$

Similarly, the next adjacent segment has area:

$$= \frac{h}{2}(y_1 + y_2) \tag{4.36}$$

and so on till the last segment having area

$$= \frac{h}{2}(y_{n-1} + y_n) \tag{4.37}$$

Hence we can write:

$$\int_{x_0}^{x_n} f(x)\,dx = \frac{h}{2}\left[y_0 + y_1 + y_1 + y_2 + y_2 + \cdots + y_{n-1} + y_{n-1} + y_n\right] \tag{4.38}$$

$$\int_{x_0}^{x_n} f(x)\,dx = \frac{h}{2}\left[y_0 + 2[y_1 + y_2 + \ldots + y_{n-1}] + y_n\right] \tag{4.39}$$

We can put it in words as "Trapezoidal rule of integration", i.e.,

$$\int_{x_0}^{x_n} f(x)dx = \frac{h}{2}[\text{first sample + last sample + 2(sum of remaining samples)}] \tag{4.40}$$

4.6.2 Simpson's Rule for Numerical Integration

The interpolating curve in case of trapezoidal rule was a straight line. Instead, if we use other interpolating curves, we get other numerical integration formulas. Simpson's 1/3 rule uses parabola as the interpolating curve and results in the following formula for integration as follows:

$$\int_{x_0}^{x_n} f(x)dx = \frac{h}{3}[y_0 + y_1 + y_3 + \cdots + y_{n-1} + 2y_2 + y_4 + \cdots + y_{n-2}) + y_n] \tag{4.41}$$

Note that $x_n = x_0 + nh$;
where n is the number of sub-intervals into which the interval of integration is divided. For Simpson's rule 'n' should be an even number as shown in Figure 4.9. We can put the Simpson's rule for numerical integration into words , for ease in recall as "Simpson's 1/3 Rule of Integration", i.e.,

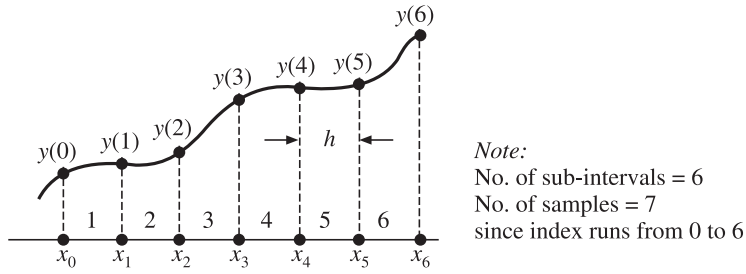


Figure 4.9 Simpson's rule for numerical integration.

$$\int_{x_0}^{x_n} f(x)dx = \frac{h}{3}[\text{first sample} + 4(\text{sum of odd indexed samples}) + 2(\text{sum of even indexed samples})] \tag{4.42}$$

4.6.3 MATLAB Script of Implementation of Solution of Differential Equation Algorithm Using Numerical Integration

```
% DEA Implementation by Numerical Integration
% ------------------------------------------------------
% Faulted line is modelled as a series R-L circuit.
% v(k)  = R i(k)  + L di(k) /dt  ...1
% v(k+1) = R i(k+1) + L di(k+1)/dt  ...2
% In (1) and (2) the only unknowns are R and L.
% Integrating between t1,t2 and t3,t4 to get two equations
% ------------------------------------------------------------
% I_t1_t2(v) = R I_t1_t2(i) + L [i(t2)- i(t1)]
% I_t3_t4(v) = R I_t3_t4(i) + L [i(t4)- i(t3)]
% A     = R B     + L C
% D     = R E     + L F
% Where A , B , D , E are numerical integrals
% C = i(t2) - i(t1) and F = i(t4) - i(t3)

% [A ]    =  [ B     C ] [R ]
% [D ]    =  [ E     F ] [L ]

% [R ]    =  [ F/d    -C/d ] [A ]
% [L ]    =  [ -E/d     B/d] [D ]
% d     =  1/(BF-EC)
% R     =  A F - C D  / B F - C E
% L     =  B D - A E /  B F - C E
% ------------------------------------------------------------
clf                         ;
clear                       ,
clc

% ------------------------------------------------------------
f                     = 50           ;
OverSamplingFactor     = 500          ;
Vm                   = 110          ;
R                     = 1            ;
XbyR                  = 20           ;
% ------------------------------------------------------------
fsampling     = 2 * f * OverSamplingFactor ;
delta_t        = 1/fsampling         ;
L            = XbyR /(2*pi*f)      ;
X            = 2*pi*f*L           ;
phi           = atan2(X,R)         ;
```

```
phi_deg       = phi*180/pi          ;
theta          = phi                ; % Switching angle
tau           = L/R                ;
X            = 2*pi*f*L            ;
Z            = R + j * X           ;
Zmag          = abs(Z)             ;
Im           = Vm/Zmag             ;
phi           = atan2(X,R)         ;

% v = Vm sin(2*pi*f*t)
% i = Imsin(2*pi*f*t + theta  - phi) - Im exp(-t / tau) sin(theta -phi);
k               = 0                ;
%delta_t        = 1e-5              ;
T               = 1/50             ;
h            = delta_t             ; % for Simpson's Rule
% ------- Generation of samples of v and i ----------------------
% ---------------------------------------------------------------

for t=0:delta_t:T
  k        = k +1                                          ;
  v(k)      = Vm * sin(2 * pi * f * t )                      ;
  i_ss(k)    = Im*sin((2*pi*f*t)-phi)                          ;
  i_tran(k) = Im*(exp(-t/tau))*(sin(theta-phi))                    ;
  %i(k)    = Im*sin((2*pi*f*t)-phi)- Im*(exp(-t/tau))*(sin(theta-phi));
  i(k)      = i_ss(k) - i_tran(k)       ;

end % end of for loop
% ---------------------------------------------------------------
subplot(3,1,1)
plot(v)                          ;
grid on                         ;
xlabel('Sample No. ')
ylabel('v ')
axis([1 length(v) -Vm Vm ])
title('Solution of Differential Equation Algorithm ')
subplot(3,1,2)
plot(i);grid on
axis([1 length(v) -2*Im 2*Im ])
xlabel('Sample No. ')
ylabel('i ')

subplot(3,1,3)
plot(i_tran)
axis([1 length(v) -Im Im])
xlabel('Sample no.')
ylabel('DC offset in current')
```

```
% ------------------------------------------------------------------------
% ------- Creation of two arrays of v() and i() for integration -----
% ------------------------------------------------------------------------
samp = 64              ;
for n = 1:samp
  v1(n) = v(n)          ;
  v2(n) = v(n+samp)     ;
  i1(n) = i(n)           ;
  i2(n) = i(n+samp)      ;
end
v1_intg     = my_simpson(v1,h)         ;
v2_intg     = my_simpson(v2,h)         ;
i1_intg     = my_simpson(i1,h)          ;
i2_intg     = my_simpson(i2,h)          ;
A = v1_intg; B  = i1_intg;C = i1(samp)-i1(1) ;
D = v2_intg; E  = i2_intg; F =i2(samp)-i2(1) ;
% ---------------------------------------------------------------
den       = B*F - C*E             ;
RR        = ( A*F - C*D) / (den )    ; % Computed value of L
LL        = (B*D - A*E) / (den )     ; % Computed value of R
XX        = 2 * pi * f * LL          ;
Error_R   = ((RR-R)/(R))*100       ;
Error_L   = ((LL - L)/(L))*100      ;
% ---------------------------------------------------------------------
fprintf('Sampling frequency = %f \n', fsampling);
sprintf('Samples per integration interval %d',samp)
sprintf('Setting R = %f   Setting L = %f Setting  X=%f',R,L,X)
sprintf('Computed R = %f  Computed L = %f Computed  X=%f',RR,LL,XX)
sprintf('Error in R = %f %% Error in L = %f %%',Error_R,Error_L)

% input: samp and delta_theta || output : value of integral
% my_simpson(samp,delta_theta)
% samp is array of samples , delta_theta is sampling interval in radians
% ----- code for the function [integral]=my_simpson(samples,h)
% % % function [integral]= my_simpson(samples ,h)
% % % % start
% % %  L = length(samples);
% % %  first=samples(1);
% % %  last = samples(L);
% % %  even= 0 ; odd = 0;
% % %  for k = 2:2:L-2
% % %   even = even + samples(k);
% % %   odd  = odd  + samples(k+1);
% % %  end
```

```
%%%  integral= (h/3)* ( first + last + 4*odd + 2 *even );
%%% % end of function my_simpson
%%%-----------------------------------------------------------------------
```

Sampling frequency = 50000.000000
Ans = Samples per integration interval 64
Ans = Setting R = 1,000000 Setting L = 0.063662 Setting X = 20.000000
Ans = Computed R = 0.979276 Computed, L = 0.063329, Computed X = 19.895521
Ans = Error in R = –2.072414%, Error in L = –0.522396 %

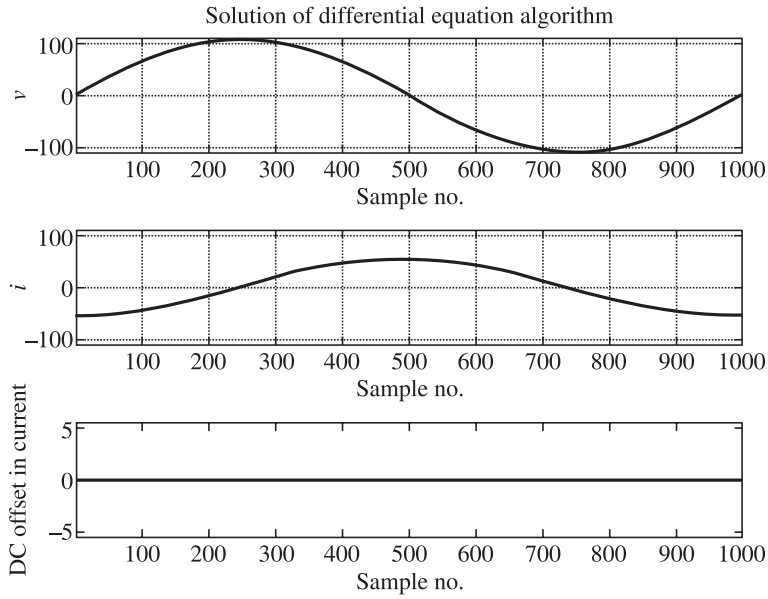


Figure 4.10 MATLAB output.

4.7 Application of Differential Equation Algorithm to Three-Phase Line

Differential equation algorithm when applied to Three-phase line needs careful consideration. This is because there are a total of 11 shunt faults that can take place on the three-phase line. Further, whatever happens in one-phase of a three-phase line depends upon what is happening in other two-phase because of the inductive and capacitive coupling between the phases.

The protection of a three-phase line is organised in the form of two distinct protections viz., ground-fault protection and phase-fault protection. Appropriate voltage and current signals need to be selected from the three voltage signals (v_a, v_b, v_c) and three current signals (i_a, i_b, i_c) available at the relay location to implement these two protections. Thus, six numbers of measuring units (M.U.) are required for implementing protection against all 11 shunt faults that can occur on a three-phase line as shown in Figure 4.11.

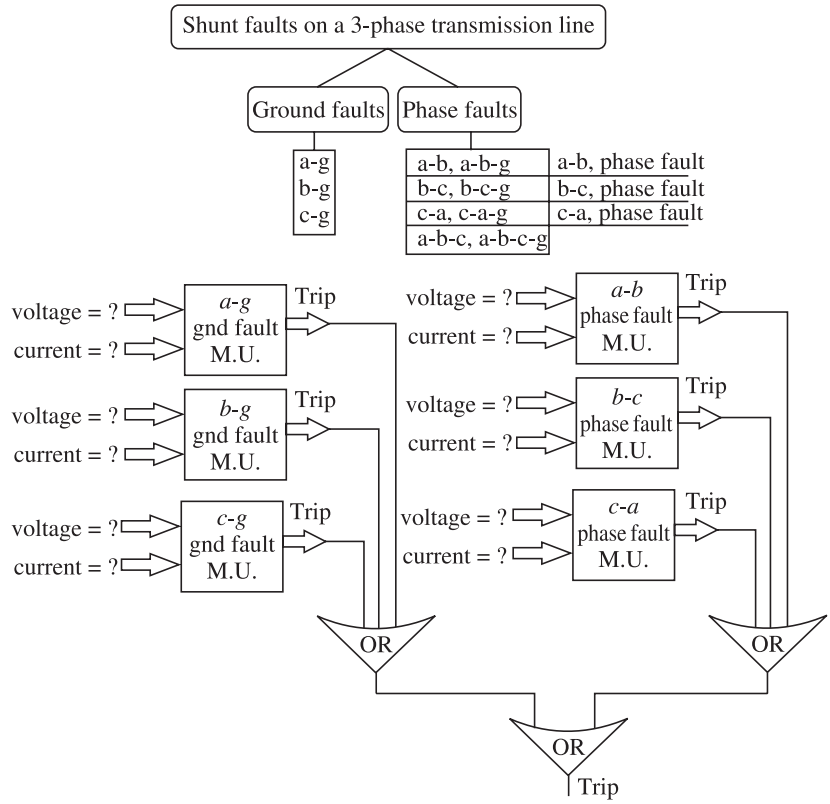


Figure 4.11 Organisation of distance protection of a three-phase line.

4.7.1 Ground Fault Protection of Three-Phase Line Using Phase Quantities

We will first take up ground fault protection. Figure 4.12 shows phase '*a*' to ground fault at a distance '*x*' from the sending end (relaying end). We are interested in finding out the voltage/current pair whose ratio correctly measures the impedance of the faulted line.

We will adopt the model shown in the Figure 4.12 for the sake of developing differential equation algorithm for three-phase line. It can be seen that we are again neglecting the shunt capacitance. Initially, we develop equations based on the series distributed parameter model and then lump the parameters.

Voltage drop across a small length *dx* of phase '*a*' of the line, denoted by $dV_{aa'}$ can be written as:

$$dV_{aa'} = (R_a dx)i_a + \left(L_a dx\right)\left(\frac{di_a}{dt}\right) + (L_{ab} dx)\left(\frac{di_b}{dt}\right) + \left(L_{ac} dx\right)\left(\frac{di_c}{dt}\right) \tag{4.43}$$

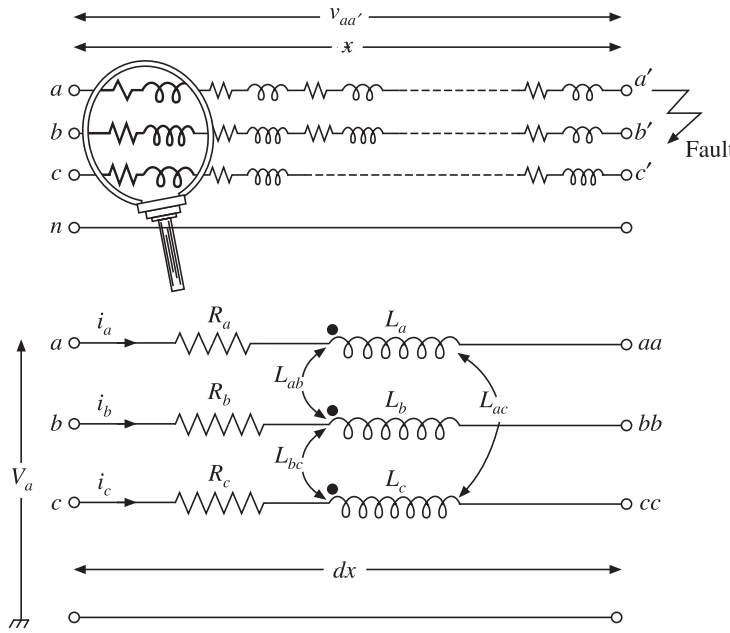


Figure 4.12 Model of three-phase line for developing DEA for three-phase lines.

Assuming that there is a single line to ground bolted fault at length '*x*' from the sending end, the voltage across length '*x*' can then be written as:

$$V_{aa'} = (R_a x)i_a + (L_a x)\left(\frac{di_a}{dt}\right) + (L_{ab} x)\left(\frac{di_b}{dt}\right) + (L_{ac} x)\left(\frac{di_c}{dt}\right) \tag{4.44}$$

$$V_{aa'} = x(R_a)i_a + x(L_a)\left(\frac{di_a}{dt}\right) + x\left(L_{ab}\right)\left(\frac{di_b}{dt}\right) + x(L_{ac})\left(\frac{di_c}{dt}\right) \tag{4.45}$$

$$V_{aa'} = x(R_a)i_a + x(L_a)\frac{d}{dt}\left(i_a + \left(\frac{L_{ab}}{L_a}\right)i_b + \left(\frac{L_{ac}}{L_a}\right)i_c\right) \tag{4.46}$$

Thus, voltage of phase '*a*' at the relay location not only depends upon the current in phase '*a*' but also upon currents in the other phases as well as the self-inductance of phase '*a*' and mutual inductances between the phases '*a*' and '*b*', '*a*' and '*c*'.

Note that $v_{aa'} = v_a$ for line to ground fault on phase '*a*'. Hence, we can write:

$$v_a = x(R_a)i_p + x(L_a)\frac{di_q}{dt} \quad \text{where } i_p = i_a \text{ and } i_q = i_a + \left(\frac{L_{ab}}{L_a}\right)i_b + \left(\frac{L_{ac}}{L_a}\right)i_c \tag{4.47}$$

Similarly, for *b-g* fault and *c-g* fault we can write:

$$v_b = x(R_b)i_p + x(L_b)\frac{di_q}{dt} \quad \text{where } i_p = i_b \text{ and } i_q = \left(\frac{L_{ab}}{L_b}\right)i_a + i_b + \left(\frac{L_{bc}}{L_b}\right)i_c \tag{4.48}$$

$$v_c = x(R_c)i_p + x(L_c)\frac{di_q}{dt} \text{ where } i_p = i_c \text{ and } i_q = \left(\frac{L_{ca}}{L_c}\right)i_a + \left(\frac{L_{cb}}{L_c}\right)i_b + i_c \tag{4.49}$$

In each of these equations, there are two unknowns (xR_a) and (xL_a) and thus we again have equation of the form $v = Ri(t) + L\frac{di(t)}{dt}$ albeit the current terms $i(t)$ and $di(t)/dt$ are different. Thus, we can follow the same methodologies that we followed for the single phase case. Figure 4.13 shows the input and output of the phase 'a' ground fault measuring unit. Similar measuring units in block-diagram form for '*b*' to '*g*' and '*c*' to '*g*' faults will have to be set up to monitor and provide protection against *b-g* and *c-g* faults.

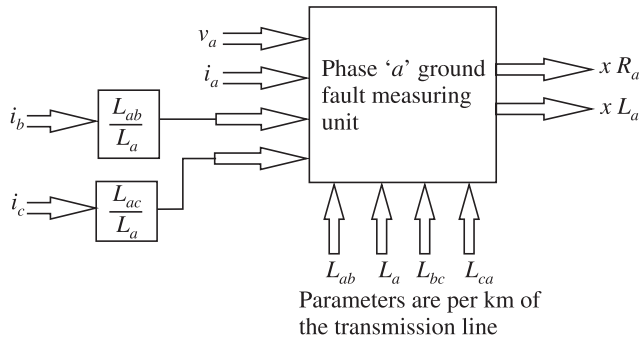


Figure 4.13 Phase 'a' ground fault measurement unit of a three-phase line.

Note that we have exclusively used all the phase quantities in the above measurement. We can instead make use of sequence (symmetrical component) quantities. We will take this up in a later section.

4.7.2 Ground Fault Protection of Three-Phase Line Using Sequence Quantities

We can write the voltage V_a at the relay location for a bolted line to ground fault at distance x as:

$$v_a = (R_a x)i_a + (L_a x)\left(\frac{di_a}{dt}\right) + (L_{ab}x)\left(\frac{di_b}{dt}\right) + (L_{ac}x)\left(\frac{di_c}{dt}\right) \tag{4.50}$$

We note that, for a well designed symmetric line:

$R_a = R_b = R_c = R$; $L_{aa} = L_{bb} = L_{cc} = L_S$ = Self inductance of each phase of the three-phase line and $L_{ab} = L_{bc} = L_{ca} = L_M$ = Mutual inductance between phases of the three-phase line

Hence we can write:

$$v_a = (Rx)i_a + (L_S x)\left(\frac{di_a}{dt}\right) + (L_M x)\left(\frac{di_b}{dt}\right) + (L_M x)\left(\frac{di_c}{dt}\right) \tag{4.51}$$

$$v_a = (Rx)i_a + (L_S x)\left(\frac{di_a}{dt}\right) + (L_M x)\frac{d}{dt}(i_b + i_c) \tag{4.52}$$

Now, we have $i_a + i_b + i_c = 3\ i_0 \rightarrow (i_b + i_c) = 3i_0 - i_a$

$$v_a = (Rx)i_a + (L_S x)\left(\frac{di_a}{dt}\right) + (L_M x)\frac{d}{dt}(3i_0 - i_a) \tag{4.53}$$

$$v_a = (Rx)i_a + (L_S x)\left(\frac{di_a}{dt}\right) + 3xL_M\frac{di_0}{dt} - xL_M\left(\frac{di_a}{dt}\right) \tag{4.54}$$

$$v_a = (\mathrm{R}x)i_a + x\left(L_S - L_M\right)\left(\frac{di_a}{dt}\right) + 3xL_M\frac{di_0}{dt} \tag{4.55}$$

We have positive and zero sequence inductance of the line L_1 and L_0 given by:

$$L_1 = L_S - L_M \tag{4.56}$$

$$L_0 = L_S + 2\ L_M \tag{4.57}$$

$$L_0 - L_1 = 3\ L_M \tag{4.58}$$

$$v_a = (Rx)i_a + x(L_1)\left(\frac{di_a}{dt}\right) + x(L_0 - L_1)\frac{di_0}{dt} \tag{4.59}$$

$$v_a = (Rx)i_a + x(L_1)\frac{d}{dt}\left(i_a + \left(\frac{L_0 - L_1}{L_1}\right)i_0\right) \tag{4.60}$$

$$v_a = x\,R\ i_a + xL_1\frac{d}{dt}(i_a + ki_0) \tag{4.61}$$

where,
$$k = \left(\frac{L_0 - L_1}{L_1}\right) \tag{4.62}$$

Hence for a three-phase line, for *a-g* fault we get an equation which is exactly of the form $v(t) = Ri(t) + L\frac{di(t)}{dt}$; albeit the terms in the i(t) and $\frac{di(t)}{dt}$ are different but known currents. Hence we can use the differential equation algorithm developed for single phase line.

For catering to '*b-g*' and '*c*-g' faults we can similarly prove that:

$$v_b = xRi_b + x\,L_1\frac{d}{dt}(i_b + ki_0) \tag{4.63}$$

$$v_c = x\,R\ i_c + x\,L_1\frac{d}{dt}(i_c + k\ i_0) \tag{4.64}$$

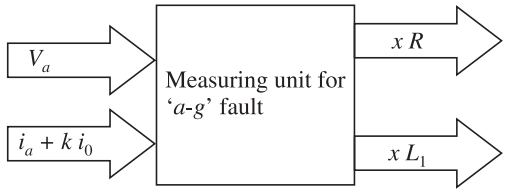


Figure 4.14 Input and output of '*a*–*g*' ground fault measuring unit.

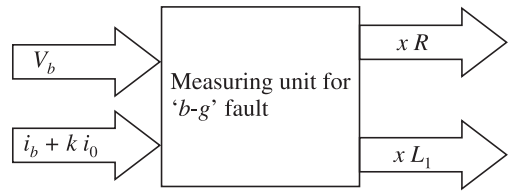


Figure 4.15 Input and output of '*b*–*g*' ground fault measuring unit.

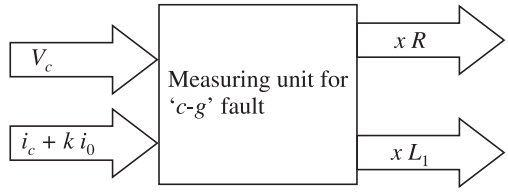


Figure 4.16 Input and output of '*c*–*g*' ground fault measuring unit.

4.7.3 Phase Fault Protection of Three-Phase Line Using Phase Quantities

Consider a phase '*a*' to phase '*b*' fault at a distance '*x*' from the relaying end as shown in Figure. 4.17.

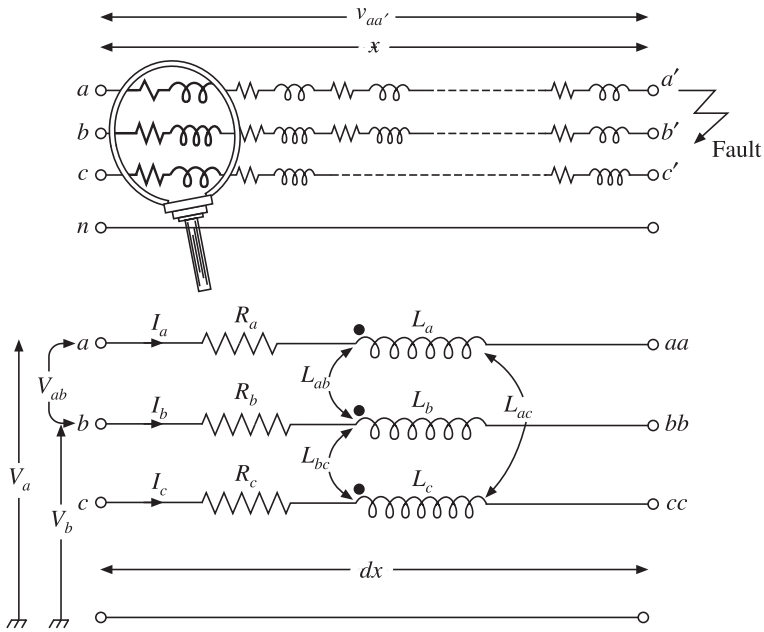


Figure 4.17 Phase '*a*' to phase '*b*' phase fault: Development of relaying quantities.

We can write:

$$v_a = x(R_a)i_a + x(L_a)\frac{di_a}{dt} + x\,L_{ab}\frac{di_b}{dt} + x\,L_{ac}\frac{di_c}{dt} \tag{4.65}$$

$$v_b = x(R_b)i_b + x(L_{ab})\frac{di_a}{dt} + x\,L_b\frac{di_b}{dt} + x\,L_{bc}\frac{di_c}{dt} \tag{4.66}$$

$$v_a - v_b = x(R_a)i_a - x(R_b)i_b + \left(x(L_a)\frac{di_a}{dt} - x(L_{ab})\frac{di_a}{dt}\right) + \left(x(L_{ab})\frac{di_b}{dt} - x(L_b)\frac{di_b}{dt}\right) + \left(x\left(L_{ac}\right)\frac{di_c}{dt} - x(L_{bc})\frac{di_c}{dt}\right) \tag{4.67}$$

$$v_a - v_b = x\,R_a\left(i_a - \frac{R_b}{R_a}i_b\right) + x(L_a - L_{ab})\frac{di_a}{dt} + x(L_{ab} - L_b)\frac{di_b}{dt} + x(L_{ac} - L_{bc})\frac{di_c}{dt} \tag{4.68}$$

For a fully transposed line it can be assumed that $L_{ac} \cong L_{bc}$, hence the last term in the above expression vanishes, giving:

$$V_a - V_b = x\,R_a\left(i_a - \frac{R_b}{R_a}i_b\right) + x(L_a - L_{ab})\left(\frac{di_a}{dt} + \left(\frac{L_{ab} - L_b}{L_a - L_{ab}}\right)\frac{di_b}{dt}\right) \tag{4.69}$$

$$V_a - V_b = x\,R_a\left(i_a - \frac{R_b}{R_a}i_b\right) + x(L_a - L_{ab})\left(\frac{di_a}{dt} - \left(\frac{L_b - L_{ab}}{L_a - L_{ab}}\right)\frac{di_b}{dt}\right) \tag{4.70}$$

$$V_a - V_b = x\,R_a\left(i_a - \frac{R_b}{R_a}i_b\right) + x(L_a - L_{ab})\frac{d}{dt}\left(i_a - \left(\frac{L_b - L_{ab}}{L_a - L_{ab}}\right)i_b\right) \tag{4.71}$$

$$V_a - V_b = x\,R_a(i_p) + x(L_a - L_{ab})\frac{di_q}{dt} \tag{4.72}$$

where $i_p = i_a - \frac{R_b}{R_a}i_b$ and $i_q = i_a - \left(\frac{L_b - L_{ab}}{L_a - L_{ab}}\right)(i_b)$ (4.73)

Note that for a symmetric line $\frac{L_b - L_{ab}}{L_a - L_{ab}}$ and $\frac{R_b}{R_a}$ will be equal to 1 and the above equation reduces to:

$$V_a - V_b = x\,R_a(i_a - i_b) + x(L_a - L_{ab})\frac{d(i_a - i_b)}{dt} \tag{4.74}$$

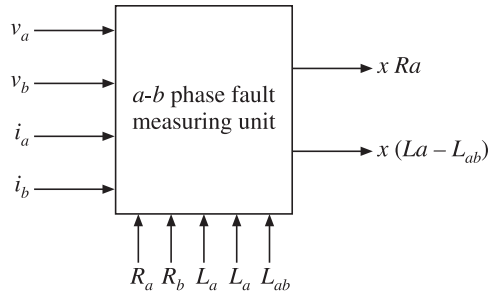


Figure 4.18 Inputs and outputs of *a-b* phase fault measuring unit.

4.7.4 Phase Fault Protection of Three-Phase Lines Using Sequence Quantities

We have seen that during '*a-b*' phase fault, the voltage ($V_a - V_b$) can be expressed as:

$$V_a - V_b = x\, R_a(i_a - i_b) + x(L_a - L_{ab})\frac{d(i_a - i_b)}{dt} \tag{4.75}$$

$$V_a - V_b = x\, R_a(i_p) + x(L_a - L_{ab})\frac{di_q}{dt} \tag{4.76}$$

where $i_p = i_a - \frac{R_b}{R_a} i_b$ and $i_q = i_a - \left(\frac{L_b - L_{ab}}{L_a - L_{ab}}\right)(i_b)$ (4.77)

Note that for a symmetric line, $\frac{L_b - L_{ab}}{L_a - L_{ab}}$ and $\frac{R_b}{R_a}$ will be equal to 1 and the above equation reduces to:

$$V_a - V_b = x\, R_a(i_a - i_b) + x(L_a - L_{ab})\frac{d(i_a - i_b)}{dt} \tag{4.78}$$

We have: $R_a = R$, $L_a = L_S$, the self-inductance and $L_{ab} = L_M$, the mutual inductance

$$V_a - V_b = x\, R(i_a - i_b) + x(L_S - L_M)\frac{d(i_a - i_b)}{dt} \tag{4.79}$$

The positive sequence inductance L_1 can be written as: $L_1 = L_S - L_M$

$$V_a - V_b = x\, R(i_a - i_b) + x(L_1)\frac{d(i_a - i_b)}{dt} \tag{4.80}$$

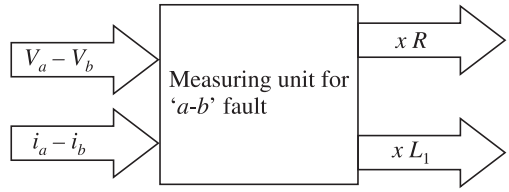


Figure 4.19 Inputs and outputs of measuring unit for *a-b* phase fault.

4.8 Elimination of Specific Harmonics by Integration between Selected Limits

Again consider the basic differential equation between voltage and current at the relay location

$$v = R\,i_p + L\frac{di_q}{dt} \tag{4.81}$$

Referring to Figure 4.20, if we choose the first limits of integration as 0 to α and second limit as (π/3) to ((π/3) + α) we get the following two simultaneous equations:

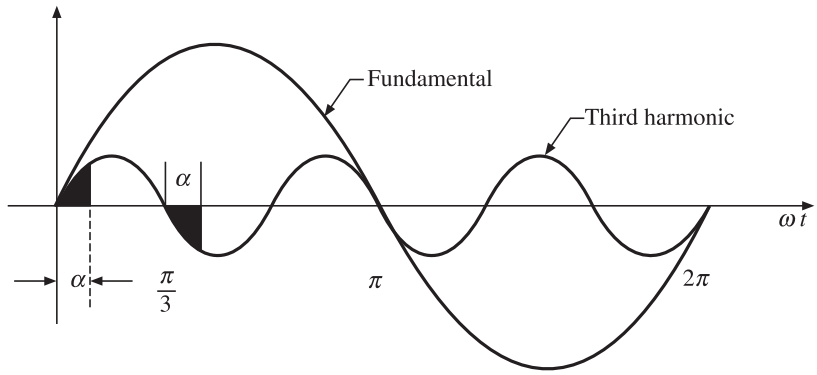


Figure 4.20 Elimination of third harmonic by choosing proper limits of integration.

$$\int_0^{\alpha/\omega} v\,dt = R\int_0^{\alpha/\omega} i_p dt + L\int_0^{\alpha/\omega} di_q \tag{4.82}$$

$$\int_{\pi/3\omega}^{\left(\frac{\pi}{3}+\alpha\right)/\omega} v\,dt = R\int_{\pi/3\omega}^{\left(\frac{\pi}{3}+\alpha\right)/\omega} i_p dt + L\int_{\pi/3\omega}^{\left(\frac{\pi}{3}+\alpha\right)/\omega} di_q \tag{4.83}$$

Adding the above two equations, we get:

$$\int_0^{\alpha/\omega} vdt + \int_{\pi/3\omega}^{\left(\frac{\pi}{3}+\alpha\right)/\omega} vdt = R\left(\int_0^{\alpha/\omega} i_p dt + \int_{\pi/3\omega}^{\left(\frac{\pi}{3}+\alpha\right)/\omega} i_p dt\right) + L\left[i_q\left[\frac{\propto}{\omega}\right] - i_q[0] + i_q\left[\frac{\frac{\pi}{3}+\propto}{\omega}\right] - i_q\left[\frac{\pi}{3\omega}\right]\right] \tag{4.84}$$

Similarly, we can choose another interval β to generate another equation:

$$\int_0^{\beta/\omega} vdt + \int_{\pi/3\omega}^{\left(\frac{\pi}{3}+\beta\right)/\omega} vdt = R\left(\int_0^{\beta/\omega} i_p dt + \int_{\pi/3\omega}^{\left(\frac{\pi}{3}+\beta\right)/\omega} i_p dt\right) + L\left[i_q\left[\frac{\beta}{\omega}\right] - i_q[0] + i_q\left[\frac{\frac{\pi}{3}+\beta}{\omega}\right] - i_q\left[\frac{\pi}{3\omega}\right]\right] \tag{4.85}$$

Let us make the following substitutions:

$$E = \int_0^{\alpha/\omega} vdt + \int_{\pi/3\omega}^{\left(\frac{\pi}{3}+\alpha\right)/\omega} vdt; \quad A = \left(\int_0^{\alpha/\omega} i_p dt + \int_{\pi/3\omega}^{\left(\frac{\pi}{3}+\alpha\right)/\omega} i_p dt\right) \tag{4.86}$$

$$B = \left[i_q\left[\frac{\propto}{\omega}\right] - i_q[0] + i_q\left[\frac{\frac{\pi}{3}+\propto}{\omega}\right] - i_q\left[\frac{\pi}{3\omega}\right]\right] \tag{4.87}$$

$$F = \int_0^{\beta/\omega} vdt + \int_{\pi/3\omega}^{\left(\frac{\pi}{3}+\beta\right)/\omega} vdt; \; C = \left(\int_0^{\beta/\omega} i_p dt + \int_{\pi/3\omega}^{\left(\frac{\pi}{3}+\beta\right)/\omega} i_p dt\right) \tag{4.88}$$

$$D = \left[i_q\left[\frac{\beta}{\omega}\right] - \mathrm{i_q}[0] + i_q\left[\frac{\frac{\pi}{3}+\beta}{\omega}\right] - i_q\left[\frac{\pi}{3\omega}\right]\right] \tag{4.89}$$

Then we can express the two equations in matrix form as:

$$E = R\,A + L\,B \tag{4.90}$$

$$F = R\,C + L\,D \tag{4.91}$$

$$\begin{bmatrix} \mathrm{E} \\ \mathrm{F} \end{bmatrix} = \begin{bmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{C} & \mathrm{D} \end{bmatrix} \begin{bmatrix} \mathrm{R} \\ \mathrm{L} \end{bmatrix} \tag{4.92}$$

Hence, R and L can be found out as:

$$\begin{bmatrix} R \\ L \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} \begin{bmatrix} E \\ F \end{bmatrix} \tag{4.93}$$

Review Questions

1. Derive the differential equation algorithm.
2. What is the difference between solution of DEA by numerical derivative and numerical integration?
3. Do we model the signal at the relay location or the power system in differential equation algorithm?
4. How do we model the system in differential equation algorithm?
5. Can lumped series model of an EHV transmission line be justified?
6. What are the problems inherent in the numerical derivative approach?
7. How numerical integration approach is better than numerical derivative approach?
8. Derive the trapezoidal method of numerical integration.
9. What is the curve underlying the trapezoidal rule?
10. Derive the Simpson's 1/3 rule for numerical integration.
11. What is the underlying curve for the Simpson's 1/3 rule?
12. State names of other methods of numerical integration.
13. Explain how the distance protection of a three-phase line is organised?
14. Why positive sequence impedance is preferred for setting of the distance relay?
15. What do you mean by ground faults?
16. What do you mean by phase faults?
17. Derive the inputs to be given to a ground fault distance measuring unit.
18. Derive the inputs to be given to a phase fault distance measuring unit.
19. What is the difference between a distance measuring unit and a distance relay?
20. What is the typical value of the zero sequence compensation factor $(L_0–L_1)/(L_1)$?
21. State the relationship between L_S, L_M and L_1, L_2, L_0.
22. What do you mean by transposed line?
23. What do you mean by fully transposed line?
24. What do you mean by symmetric line?
25. What is the motivation for 'integration between specific limits' method of implementing the DEA?

5

Algorithms Based on Least Squared Error (LSQ)

5.1 LSQ Technique

First step in any relaying or control activity is measurement. It is a fact of life that measurement is never perfect. It always involves errors. We can never completely eliminate errors. Thus, it is impossible to find the true value. We have to live with errors.

Gurudev Ravindranath Tagore's observation about error is very pertinent:

"If you shut the door on error, truth also will be shut out"

Thus, measurement is the art of keeping the baby while throwing out the bath water! We can think of errors as gateways to measurement rather than hindrance to measurement.

While we cannot eliminate errors, we can certainly minimise them. The branch of mathematics known as *statistics* provides us with very powerful methods of minimising errors. *LSQ technique* is one such method of minimising the errors. It is to be noted that even though we do not know the magnitude of the error, we have a good idea about its statistical nature. And this is fortunate, because otherwise we would have been left groping in the dark with a blind-fold. Knowledge of statistical nature of error neither removes the blind-fold nor the darkness but provides us with the blind-man's stick.

The statistical nature of error that we will assume is:

- random with zero mean
- Gaussian
- additive (and not multiplicative)

Before delving into numerical relaying algorithms based on LSQ technique, we will see in some detail theory behind it and its relationship with average and pseudo-inverse.

The least squared error (LSQ) also called as *least sum of squares of errors* (LSE) can be traced back to Gauss who first used it to predict the orbits of planets. It has been widely used over the years and offers us a tool for estimating values when the signal is subject to noise.

5.2 Average is Best Value in LSQ Sense

Right from our early school days, we have been taught to trust the average of readings obtained in an experiment more than the individual observation. Why is average so universally accepted as the true value?

The following investigation will throw some light on the relationship between 'average value' and value estimated by minimising the sum of squares of errors.

In order to find value of an unknown '*x*', let us say, two measurements were made. During each measurement we got a different result.

Let, $x = 2$ … first measurement
and $x = 3$ … second measurement

Now, we are faced with the problem of estimating the true value. Of course we cannot know the true value. We can only minimise the error. We will try to minimise the sum of squares of errors instead of the individual errors. We acknowledge that both measurements include additive errors. Thus, we can write the measurements as:

$x = 2 + e_1$ … e_1 is the error in the first measurement
$x = 3 + e_2$ … e_2 is the error in the second measurement

Which gives:

$$e_1 = x - 2$$
$$e_2 = x - 3$$

Hence, we can write the sum of the squares of the errors, denoted by SSE as:

$$\text{SSE} = e_1^2 + e_2^2 = (x-2)^2 + (x-3)^2$$

We are interested in finding the value of *x* which minimises the sum of the squares of errors. Hence, this value will be that for which, the following two conditions are met:

(i) $\dfrac{d(\text{SSE})}{dx} = 0$ and

(ii) $\dfrac{d^2(\text{SSE})}{dx^2} = \text{positive}$

Let us find the value of '*x*' for which the first derivative of SSE w.r.t. '*x*' is zero, i.e.,

$$\frac{d}{dx}((x-2)^2 + (x-3)^2) = 0 \text{ and } \frac{d^2}{dx^2}((x-2)^2 + (x-3)^2) > 0$$

$$2(x-2) + 2(x-3) = 0$$

$$2x - 4 + 2x - 6 = 0 \Rightarrow 4x - 10 = 0 \Rightarrow 4x = 10 \Rightarrow x = \frac{10}{4} \Rightarrow x = \frac{5}{2} = 2.5$$

Further, $\frac{d^2}{dx}((x-2)^2+(x-3)^2)=\frac{d}{dx}(4x-10)=4$ which is >0

Hence, $x=\frac{5}{2}=2.5$ is the value at which the sum of the squares of errors is indeed minimum. Now, it is easy to see that $x=\frac{5}{2}=\frac{3+2}{2}=2.5$

Thus, the estimated value with minimum error that we arrived at by minimising the sum of the squares of the errors is the same as the mean or the average value. Hence, there is greater significance to the average value as a value which minimises the sum of squares of errors, which is the closest that we can approach the truth.

5.3 LSQ and Pseudo-Inverse

Now, we will show how the problem illustrated above can be elegantly handled by using pseudo-inverse method. We write the two equations $x = 2$ and $x = 3$ as matrix equation in the following form:

$$\begin{bmatrix}1\\1\end{bmatrix}_{2\times1}[x]_{1\times1}=\begin{bmatrix}2\\3\end{bmatrix}_{2\times1}$$

In this system of equations, we have a situation where we have more equations than the number of unknowns. This is called an *over-determined system*. We cannot invert the coefficient matrix $\begin{bmatrix}1\\1\end{bmatrix}_{2\times1}$ since it is not square. In order to convert the coefficient matrix into square, we pre-multiply both sides by its transpose, which is [1 1], thus we get:

$$\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}[x]=\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}2\\3\end{bmatrix}$$

If we multiply both sides by $\left[\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\right]^{-1}$ then we get:

$$\left[\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\right]^{-1}\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}[x]=\left[\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\right]^{-1}\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}2\\3\end{bmatrix}$$

$$[1][x]=\left[\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\right]^{-1}\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}2\\3\end{bmatrix}$$

Note that $\left[\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\right]^{-1}=[2]^{-1}=\begin{bmatrix}\frac{1}{2}\end{bmatrix}=[0.5]$

$$[x] = \left[\frac{1}{2}\right][1 \quad 1]\begin{bmatrix}2\\3\end{bmatrix}$$

$$[x] = \left[\frac{1}{2}\right][1 \quad 1]\begin{bmatrix}2\\3\end{bmatrix}$$

$$[x] = \begin{bmatrix}\frac{1}{2} & \frac{1}{2}\end{bmatrix}\begin{bmatrix}2\\3\end{bmatrix}$$

$$[x] = \left[\frac{2}{2} + \frac{3}{2}\right]$$

$$[x] = \left[\frac{5}{2}\right] = 2.5$$

Representing the problem in symbolic fashion, we started with the matrix equation:

$$[A]_{2\times1}\,[x]_{1\times1} = [c]_{2\times1}$$

Pre-multiplying both sides by $[A]^T$

$$[A]^T[A]_{2\times1}[x]_{1\times1} = [A]^T\,[c]_{2\times1}$$

Pre-multiplying both sides by $\{[A]^T[A]_{2\times1}\}^{-1}$

$$\{[A]^T[A]_{2\times1}\}^{-1}\,\{[A]^T[A]_{2\times1}\}[x]_{1\times1} = \{[A]^T[A]_{2\times1}\}^{-1}[A]^T[c]_{2\times1}$$

$$[I]_{1\times1}[x]_{1\times1} = \{[A]^T[A]_{2\times1}\}^{-1}[A]^T[c]_{2\times1}$$

$$[x]_{1\times1} = [A^+][c]_{2\times1}$$

where $[A^+] = \{[A]^T[A]_{2\times1}\}^{-1}[A]^T$ is called the *pseudo-inverse* of the rectangular matrix A whose number of rows is greater than number of columns. Pseudo-inverse has properties very similar to inverse. Some of the properties of the pseudo-inverse are given below:

(i) $(P\,P^+)\,P = P$
(ii) $(P^+\,P)\,P^+ = P^+$
(iii) $(P^+\,P) = (P^+\,P)^T$
(iv) $P\,P^+ = (P\,P^+)^T$

Pseudo-inverse can also be determined for under-determined system as:

$$[A]^T\,\{[A]_{2\times1}[A]^T\}^{-1}$$

5.4 Relation between LSQ and Pseudo-Inverse

We present a more formal proof that pseudo-inverse technique amounts to minimising in the LSE sense. We will use the symbolism of statistical signal processing to denote:

[Z] is a column vector of measurements
$[\hat{x}]$ $\hat{x}$ is the vector of estimates of true value of [x] of which [Z] is the measurement

[*v***]** is the vector of noise terms (errors)
[*H*] is the matrix representing the process

The measurement process is defined by the following equation:

$$[Z]_{m\times 1} = [H]_{m\times n}\ [\hat{x}]_{n\times 1} + [v]_{m\times 1}$$

Here our aim is to find $\hat{x}$ such that error is minimised. We can write

$$[v]_{m\times 1} = [Z]_{m\times 1} - [H]_{m\times n}\ [\hat{x}]_{n\times 1}$$

Sum of squares of errors can be written as:

$$\text{SSE} = (v_1^2) + (v_2^2) + \cdots + (v_m^2) = [v_1 \ldots v_m]\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} = [v^T][v]$$

$$[v^T][v] = ([Z] - [H][\hat{x}])^T([Z] - [H][\hat{x}])$$

Noting that: $-([Z] - [H][\hat{x}])^T = (Z^T - \hat{x}^T H^T)$

$$[v^T][v] = (Z^T - \hat{x}^T H^T)([Z] - [H][\hat{x}])$$

$$(\text{SSE})_{1\times 1} = (Z^T[Z] - [\hat{x}]^T[H^T][Z] - Z^T[H][\hat{x}] + [\hat{x}]^T[H^T][H][\hat{x}])$$

Note: $[\hat{x}]^T[H^T][Z] = Z^T[H][\hat{x}]$

$$\text{SSE} = (Z^T[Z] - 2[\hat{x}]^T[H^T][Z] + [\hat{x}]^T[H^T][H][\hat{x}])$$

As already seen $d(\text{SSE})/d\hat{x} = 0$ and $d^2(\text{SSE})/d(\hat{x})^2 > 0$ for minimising the sum of squares of errors. Thus, differentiating the SSE and equating it to zero gives:

$$\frac{d(\text{SSE})}{d(\hat{x})} = (-2[H^T][Z] + 2[H^T][H][\hat{x}]) = 0$$

$$([H^T][H][\hat{x}]) = [H^T][Z]$$

$$([[H^T][H]]^{-1}[H^T][H][\hat{x}]) = [[H^T][H]]^{-1}[H^T][Z]$$

$$[\hat{x}] = \left[[H^T][H]\right]^{-1}[H^T][Z]$$

$$[\hat{x}] = [H]^{+}[Z]$$

where $[H]^{+} = [[H^T][H]]^{-1}$ is the pseudo-inverse of *H*.

Thus, minimising the sum of squares of errors directly involves pseudo-inverse.

Further, let us find out if $\dfrac{d^2(\text{SSE})}{d\hat{x}^2} = \text{positive?}$

then $$\frac{d^2(\text{SSE})}{dx^2} = [H^T][H]$$

Now, square matrix of the type $[H^T][H]$ is always positive definite. Hence both, the first order and second order conditions for minima are met.

Hence, we can say that the pseudo-inverse method minimises in the LSE or LSQ sense. Thus, we are justified in calling the algorithm by Sachdev as an LSQ/LSE algorithm.

5.5 LSQ Algorithm by Sachdev [50]

In the classical curve fitting technique as shown in Figure 5.1, the measured values of the function are given by Y_0, Y_1, Y_2... corresponding to the independent variable values x_0, x_1, x_2,.... Assuming that the true values of the function $f(x)$ are y_0, y_1, y_2,... then the errors are:

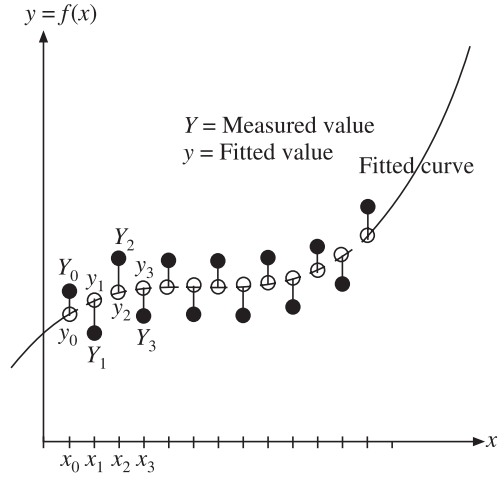


Figure 5.1 Curve fitting using LSQ Technique.

$e_1 = y_0 - Y_0$; $e_1 = y_1 - Y_1$; $e_2 = y_2 - Y_2$. We search for a function $y = f(x)$ which minimises the sum of squares of errors, SSE, instead of minimising the individual errors.

$$\text{SSE} = \sum_i e_i^2 = (e_0)^2 + (e_1)^2 + (e_2)^2 + \cdots$$

Sachdev's algorithm is based on the concept of curve fitting. On the basis of apriori knowledge, we know the nature of the waveform of the fault current but we do not know exact values of the parameters. We know that the fault current will consist of a decaying DC offset, a fundamental and a few harmonics. Thus, we try to fit the samples of fault current to the model given by:

$$y(t) = f(t) = I_0\, e^{-t/\tau} + I_{m1} \sin(\omega_0 t + \theta_0) + I_{m3} \sin(3\omega_0 t + \theta_3) + I_{m5} \sin(5\omega_0 t + \theta_5) + \cdots$$

Assuming, for the sake of illustration, that the fault current consists of:

(i) a decaying DC offset

(ii) no even harmonics
(iii) harmonics up to third harmonic

We can write the expression for the current as:

$$y(t) = f(t) = I_0\, e^{-t/\tau} + I_{m1} \sin(\omega_0 t + \theta_0) + I_{m3} \sin(3\omega_0 t + \theta_3)$$

Using the Taylor's series expansion of e^x, we get:

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots$$

writing ($I_0 e^{-t/\tau}$) using only three terms of the expansion as: $I_0 e^{-t/\tau} \cong I_0 - \frac{I_0 t}{\tau} + \frac{I_0 t^2}{2\tau^2}$.

Thus, the model of the fault current with which we will be trying to fit the samples of the fault current becomes:

$$y(t) = f(t) = I_0 - \frac{I_0 t}{\tau} + \frac{I_0 t^2}{2\tau^2} + I_{m1} \sin(\omega_0 t + \theta_1) + I_{m3} \sin(3\omega_0 t + \theta_3)$$

Further we will expand the $\sin(\omega_0 t + \theta_0)$ and $\sin(3\omega_0 t + \theta_3)$ terms to get:

$$i(t) = I_0 - \frac{I_0 t}{\tau} + \frac{I_0 t^2}{2\tau^2} + I_{m1} \sin(\omega_0 t) \cos(\theta_1) + I_{m1} \cos(\omega_0 t) \sin(\theta_1) + I_{m3} \sin(3\omega_0 t) \cos(\theta_3) + I_{m3} \cos(3\omega_0 t) \sin(\theta_3)$$

By Rearranging, we get:

$$i(t) = I_0 - \frac{I_0 t}{\tau} + \frac{I_0 t^2}{2\tau^2} + I_{m1} \cos(\theta_1) \sin(\omega_0 t) + I_{m1} \sin(\theta_1) \cos(\omega_0 t) + I_{m3} \cos(\theta_3) \sin(3\omega_0 t) + I_{m3} \sin(\theta_3) \cos(3\omega_0 t)$$

Again by rearranging, we get:

$$i(t) = I_0 + I_{m1} \cos(\theta_1) \sin(\omega_0 t) + I_{m1} \sin(\theta_1) \cos(\omega_0 t) + I_{m3} \cos(\theta_3) \sin(3\omega_0 t) + I_{m3} \sin(\theta_3) \cos(3\omega_0 t) - \frac{I_0 t}{\tau} + \frac{I_0 t^2}{2\tau^2}$$

Let us denote $i(t)$ since it is a sample value by $s(t)$.

$$s(t) = I_0 + I_{m1} \cos(\theta_1) \sin(\omega_0 t) + I_{m1} \sin(\theta_1) \cos(\omega_0 t) + I_{m3} \cos(\theta_3) \sin(3\omega_0 t) + I_{m3} \sin(\theta_3) \cos(3\omega_0 t) - \frac{I_0 \tau}{\tau} + \frac{I_0 \tau^2}{2\tau^2}$$

In the above equation we have seven unknowns. Hence, we need to generate a total of seven equations to be able to solve for the seven unknowns. This is shown below with unknown quantities in each equation shown in bold.

$$S(t_1) = 1.\ \boldsymbol{I_0} + \sin(\omega_0 t_1)\ \boldsymbol{I_{m1}}\ \boldsymbol{\cos(\theta_1)} + \cos(\omega_0 t_1)\ \boldsymbol{I_{m1}}\ \boldsymbol{\sin(\theta_1)}$$
$$+ \sin(3\omega_0 t_1)\ \boldsymbol{I_{m3}}\ \boldsymbol{\cos(\theta_3)} + \cos(3\omega_0 t_1)\ \boldsymbol{I_{m3}}\ \boldsymbol{\sin(\theta_3)} - t_1\left(\frac{\boldsymbol{I_0}}{\boldsymbol{\tau}}\right) + (t_1)^2\left(\frac{\boldsymbol{I_0}}{\boldsymbol{2\tau^2}}\right)$$

$$S(t_2) = 1.\ \boldsymbol{I_0} + \sin(\omega_0 t_2)\ \boldsymbol{I_{m1}}\ \boldsymbol{\cos(\theta_1)} + \cos(\omega_0 t_2)\ \boldsymbol{I_{m1}}\ \boldsymbol{\sin(\theta_1)}$$
$$+ \sin(3\omega_0 t_2)\ \boldsymbol{I_{m3}}\boldsymbol{\cos(\theta_3)} + \cos(3\omega_0 t_2)\ \boldsymbol{I_{m3}}\ \boldsymbol{\sin(\theta_3)} - t_2\left(\frac{\boldsymbol{I_0}}{\boldsymbol{\tau}}\right) + (t_2)^2\left(\frac{\boldsymbol{I_0}}{\boldsymbol{2\tau^2}}\right)$$

... ...

$$S(t_7) = 1.\ \boldsymbol{I_0} + \sin(\omega_0 t_7)\ \boldsymbol{I_{m1}}\ \boldsymbol{\cos(\theta_1)} + \cos(\omega_0 t_7)\boldsymbol{I_{m1}}\ \boldsymbol{\sin(\theta_1)}$$
$$+ \sin(3\omega_0 t_7)\ \boldsymbol{I_{m3}}\ \boldsymbol{\cos(\theta_3)} + \cos(3\omega_0 t_7)\boldsymbol{I_{m3}}\ \boldsymbol{\sin(\theta_3)} - t_7\left(\frac{\boldsymbol{I_0}}{\boldsymbol{\tau}}\right) + (t_7)^2\left(\frac{\boldsymbol{I_0}}{\boldsymbol{2\tau^2}}\right)$$

In each equation there are seven knowns, which can be pre-computed. For example, in the first equation these are:

$a_{11} = 1$, $a_{12} = \sin(\omega_0 t_1)$, $a_{13} = \cos(\omega_0 t_1)$, $a_{14} = \sin(3\omega_0 t_1)$, $a_{15} = \cos(3\omega_0 t_1)$, $a_{16} = -t_1$ and $a_{17} = t_1^2$

The seven unknowns are:

$x_1 = I_0$, $x_2 = I_{m1}\cos(\theta_1)$, $x_3 = I_{m1}\sin(\theta_1)$, $x_4 = I_{m3}\cos(\theta_3)$,

$x_5 = I_{m3}\sin(\theta_3)$, $x_6 = \left(\frac{I_0}{\tau}\right)$, $x_7 = \left(\frac{I_0}{2\tau^2}\right)$

The amplitudes and phase angles of the fundamental and the third harmonic can be found as shown below:

$$\boldsymbol{I_{m1} = \sqrt{x_2^2 + x_3^2}} \qquad \boldsymbol{I_{m3} = \sqrt{x_4^2 + x_5^2}}$$

$$\boldsymbol{\theta_1 = \tan^{-1}\left(\frac{x_3}{x_2}\right)} \qquad \boldsymbol{\theta_3 = \tan^{-1}\left(\frac{x_3}{x_4}\right)}$$

We will simplify the representation by writing as:

$$s(1) = a_{11}\ x_1 + a_{12}\ x_2 + a_{13}\ x_3 + a_{14}\ x_4 + a_{15}\ x_5 + a_{16}\ x_6 + a_{17}\ x_7$$
$$s(2) = a_{21}\ x_1 + a_{22}\ x_2 + a_{23}\ x_3 + a_{24}\ x_4 + a_{25}\ x_5 + a_{26}\ x_6 + a_{27}\ x_7$$
$$s(3) = a_{31}\ x_1 + a_{32}\ x_2 + a_{33}\ x_3 + a_{34}\ x_4 + a_{35}\ x_5 + a_{36}\ x_6 + a_{37}\ x_7$$
$$s(4) = a_{41}\ x_1 + a_{42}\ x_2 + a_{43}\ x_3 + a_{44}\ x_4 + a_{45}\ x_5 + a_{46}\ x_6 + a_{47}\ x_7$$
$$s(5) = a_{51}\ x_1 + a_{52}\ x_2 + a_{53}\ x_3 + a_{54}\ x_4 + a_{55}\ x_5 + a_{56}\ x_6 + a_{57}\ x_7$$
$$s(6) = a_{61}\ x_1 + a_{62}\ x_2 + a_{63}\ x_3 + a_{64}\ x_4 + a_{65}\ x_5 + a_{66}\ x_6 + a_{67}\ x_7$$
$$s(7) = a_{71}\ x_1 + a_{72}\ x_2 + a_{73}\ x_3 + a_{74}\ x_4 + a_{75}\ x_5 + a_{76}\ x_6 + a_{77}\ x_7$$

Using the matrix notation:

$$\begin{bmatrix} s(1) \\ s(2) \\ s(3) \\ s(4) \\ s(5) \\ s(6) \\ s(7) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & a_{37} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & a_{67} \\ a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & a_{57} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

For simplicity, we will write the equation as follows:

$$[S]_{7\times1} = [A]_{7\times7}\,[X]_{7\times1}$$

Since [*A*] matrix is square we can invert it and write:

$$[X]_{7\times1} = [A^{-1}]_{7\times7}\,[S]_{7\times1}$$

If we take '*m*' number of samples such that '*m*' > 7 then [*A*] matrix becomes rectangular and we can use pseudo-inverse to solve for the unknowns.

$$[S]_{m\times1} = [A]_{m\times7}[X]_{7\times1}$$
$$[X]_{7\times1} = [A^{+}]_{7\times m}\,[S]_{m\times1}$$

Figure 5.2 shows how LSQ algorithm can be used to implement any distance relay.

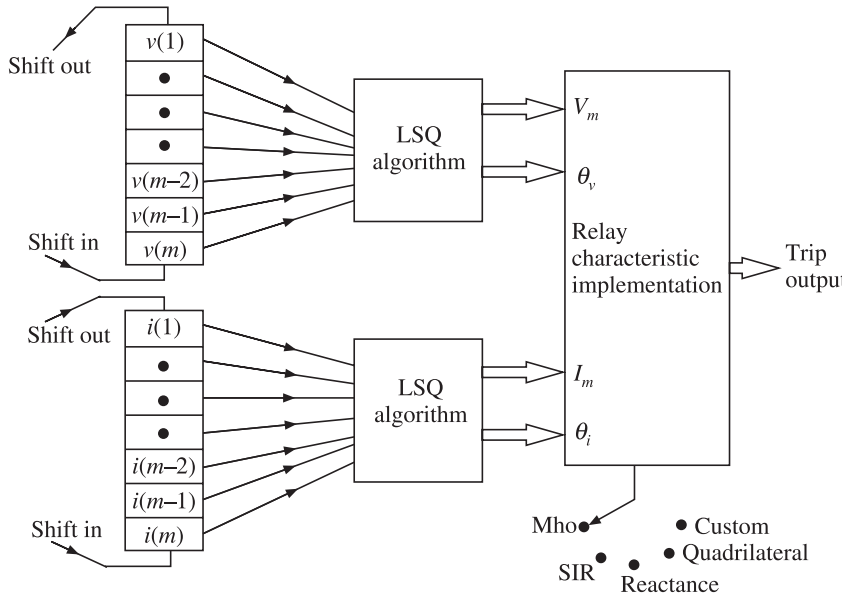


Figure 5.2 Implementation of LSQ algorithm.

5.6 MATLAB Implementation of LSQ Algorithm

```
% LSQ algorithm by Sachdev
% Algo is based on modelling of the signal.
% If we model the signal to contain
% decaying DC offset, funda and 3rd harmonic :-
% i(t)= I0 exp(-t/tau)+ Im1 sin(w0 t + theta1 ) + Im3 sin(3
w0 t + theta3)
% Unknowns are : Im1 , theta1 , Im3 , theta3 = 4 unknowns
% DC offset is modelled using three terms of expansion = 3 unknown
% Total num of unknowns in this case = 7 unknowns
% Therefore we need at least 7 sampling instants to form 7 equations
% [s] = [a ] * [ X ]
% [X ] = inv(a) *  [s] if [a] is not square then
% [X ] = pinv(a) * [s]
clear;clc
% ---------------------------------------------------------------
%----------------------------- Settings -----------------------
%---------------------------------------------------------------
samples = input(‘ Enter the no. of samples  = ‘)              ;
f                  = 50                                      ;
Vm                 = 110                                     ;
I0                 = 10                                      ;
theta1             = pi/3                                    ;
theta3             = pi/6                                    ;
R                  = 1                                       ;
XbyR               = 20                                      ;
MaxHarmonic        = 3                                       ;
%---------------------------------------------------------------
%-----------------------   End of Settings -------------------
%---------------------------------------------------------------
fsig_max           = f * MaxHarmonic                         ;
OverSamplingFactor = MaxHarmonic *20                         ;
%                      2 is the minimum value                ;
fsamp          = 2* fsig_max * OverSamplingFactor            ;
delta_t            = 1/fsamp                                 ;
X                  = R * XbyR                                ;
Zmag               = sqrt(R^2+ X^2)                          ;
w0                 = 2*pi*f                                  ;
L                  = X / w0                                  ;
tau                = L/R                                     ;
Im1                = Vm/ Zmag                                ;
Im3                = Im1/3                                   ;
%---------------------------------------------------------------
```

```
% NumOfUnknowns will depend upon model of signal assumed
  NumOfUnknowns = 7                                                    ;
%----- Sampling the signal to generate samples for processing
by LSQ ----
for col=1:NumOfUnknowns
      for row=1:samples
         t          = row*delta_t;
s(row,1)=  I0*exp(-t/tau)+Im1*sin(w0*t+theta1)+Im3*sin(3*w0*t+the
ta3);
a(row,1)= 1                                                            ;
         a(row,2)= sin(   w0 * t);
         a(row,3)= cos(   w0 * t);
         a(row,4)= sin(3*w0 * t);
         a(row,5)= cos(3*w0 * t);
         a(row,6)= t;
         a(row,7)= t*t;
      end
end
% --------Samples of signal generated   --------
% -------  LSQ Algorithm Starts ------------
   x                 = pinv(a)*s;
% -------  ------------------------  ------------
% x              = inv(a) * s if a is 7 x 7
   x2                = x(2,1);
   x3                = x(3,1);
   x4                = x(4,1);
   x5                = x(5,1);

   IIm1              = sqrt( (x2^2) + (x3^2) );
   theta1_calc   = atan2(x3,x2);
   IIm3              = sqrt( (x4^2) + (x5^2) );
   theta3_calc   = atan2(x5,x4);
   err_Im1         = ((IIm1-Im1) / (Im1))*100 ;
   err_theta1     = ( (theta1_calc-theta1)/ ( theta1)) * 100;
   fprintf(‘ Peak Value of Funda=%f Estimated Peak of Funda=%f
\n’,Im1 , IIm1);
    fprintf(‘ theta1                        =%f Estimated theta1
=%f \n’,theta1,theta1_calc);
  fprintf(‘ Error in Estimation of Funda Peak              =%f
%% \n’,err_Im1   );
  fprintf(‘ Error in Estimation of Phase Ang. of Funda     =%f
%% \n’,err_theta1);
    fprintf(‘ Im3=%f IIm3=%f theta3=%f theta3_calc=%f\n’,Im3 ,
```

```
IIm3,theta3,theta3_calc)
   plot(s)
%-------------------------------------------------------------------
%----------------------   End of Program ----------------------
%-------------------------------------------------------------------%
```

```
Enter the no. of samples = 9
Peak Value of Funda = 5.493138 Estimated Peak of Funda = 5.468352
theta1      = 1.047198, Estimated theta1    = 1.044800
Error in Estimation of Funda Peak       = –0.451218 %
Error in Estimation of Phase Ang. of Funda  = –0.228963 %
Im3 = 1.831046 IIm3 = 1.831287 theta3 = 0.523599 theta3_calc = 0.523735
>>
```

Review Questions

1. Show how average value is related with LSQ technique.
2. Show how pseudo-inverse is related with LSQ technique.
3. Give the geometric significance of LSQ technique.
4. Show that Fourier coefficients fit the given waveform using the Fourier series in the LSQ sense.
5. Define left pseudo-inverse and right psudo-inverse.
6. When are we required to use left pseudo-inverse?
7. In order to apply the LSQ technique, we must have apriori knowlege about the nature of the signal. Discuss.
8. It is known apriori that a certain fault current contains decaying DC offset, fundamental and fifth harmonic. Develop a LSQ algorithm to extract the various frequency components from the fault current signal.
9. What are the various variations of the LSQ technique?
10. What is total least squares?
11. What is recursive least squares?
12. How Kalman filter is related with least squares method?
13. What is weighted least squares?

6

Discrete Fourier Transform

6.1 Introduction to Concept of Spectrum

Mankind has always been fascinated by periodically repeating events like sun-rise and sun-set, onset of various seasons, phases of the moon, tides in the sea, celestial events like eclipses, sun-spots, and sighting of comets. Thus, curiosity about periodic events predates scientific study of the subject.

Sir Issac Newton introduced the scientific term '*spectrum*' using the Latin word for an image. Modern English language has the word 'spectre' meaning 'ghost' or 'apparition' and the corresponding adjective '*spectral*'. Every school-boy knows what 'VIBGYOR' shown in Figure 6.1 stands for in case of light and 'GAMUT' in case of musical sounds.

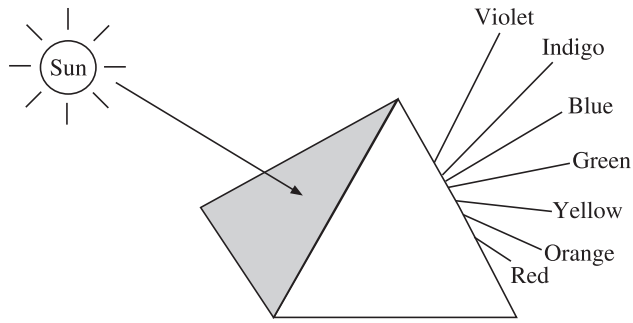


Figure 6.1 Concept of spectrum.

For the sake of completeness, we list the frequencies of various visible colours (in terra-hertz) and audible sounds (in Hz) as shown in Tables 6.1 and 6.2 respectively, which by the way, shows that we are almost deaf as well as blind since we can perceive only a small fraction of sound and light spectra.

Deaf we may be, but it can be seen from Table 6.2 that all of us, whether we are musically trained or not, have the ability to correctly decode the ratio of two frequencies. The ratio of frequency of adjacent (pure) notes on the piano is exactly $2^{1/7}$, so that the eighth note (first note in the next octave) is double in frequency as compared to the first note! In fact that is the origin of the word octave (Latin for eight). This is a tribute to our ears, which over millions of years, evolved as specialised instruments for doing real-time frequency analysis of very high complexity. In fact our ears have evolved dedicated hardware to do frequency analysis before sending the signal to the brain.

Thus, sound, which is information in time domain, encoded in time varying pressure wave in air, is actually converted into a frequency domain signal by the 'mechanism' of the ear. What we finally perceive is not the instantaneous variations of pressure but a 'tone', i.e., a frequency with a certain magnitude which we perceive as weak or strong. This is exactly the job of an instrument called *spectrum analyser*. Thus, we have a spectrum analyser for sound in the audible frequency range of 20 Hz to 20 kHz.

It is interesting to note that it is also possible to trick our ears into believing what we want it to believe if we program the sounds correctly. We can create the illusion of a stereo sound or 'surround sound' by manipulating the time of arrival of sound signals. Similarly, we can substantially reduce perception of noise in a recording by using some more tricks like 'Dolby™ techniques'.

Similarly our eyes perceive the time domain light signals as 'colours', i.e., frequencies with their respective strengths (lightness or darkness of the colour) before passing it to the brain. We cannot tell the instantaneous value of the light signal. Thus, our eyes constitute a spectrum analyser working in the visible frequency range.

This is a fascinating topic but we are constrained to leave it here and concentrate on electrical engineering!

Table 6.1 Spectrum of light

←Infra-red	**← Visible Spectrum →**							**→Ultra-violet**
	Red	Orange	Yellow	Green	Blue	Indigo	Violet	
< 400 THz	400 THz	484 THz	508 THz	526 THz	631 THz	651 THz	668 THz	> 668 THz
	700 nm						400 nm	

Table 6.2 Spectrum of sound

←Infra-sonic	**← Audible spectrum →**								**→Ultra-sonic**
	Sa	Re	Ga	Ma	Pa	Dha	Ni	Sa	
	Do	Re	Mi	Fa	So	La	Ti	Do	
Less than 20 Hz	470 Hz	$Sa \times 2^{1/7}$ Hz	$Re \times 2^{1/7}$ Hz	$Ga \times 2^{1/7}$ Hz	$Ma \times 2^{1/7}$ Hz	$Pa \times 2^{1/7}$ Hz	$Dha \times 2^{1/7}$ Hz	470×2 Hz	Greater than 20 kHz

Today our thinking is so much dominated by digital computers that it is difficult to imagine Fourier analysis being done by any other method than by using digital computers. However,

using computers for Fourier analysis is of comparatively recent origin. For tens of years, scientists have done Fourier analysis using mechanical, acoustic or optical methods. Prior to 1965 such devices were in wide use for Fourier analysis. However, with the publication of the FFT algorithm by Cooley and Tuckey in 1965, all other methods have been superseded by digital computer oriented methods.

6.1.1 Generalised Fourier Series

The generalised Fourier series is based on the concept of orthogonal functions. Let $\phi_0(t), \phi_1(t), \phi_2(t), \phi_3(t), \ldots, \phi_n(t)$ be a set of mutually orthogonal functions. Orthogonality is defined as follows:

Two functions $f(t)$ and $g(t)$ are orthogonal in the interval 'a' to 'b' if

$$\int_a^b f(t)g(t) = 0$$

with the additional property that

$$\int_a^b f(t)\,f(t) = K \quad \text{and} \quad \int_a^b g(t)g(t) = K$$

if $K = 1$ the functions are called *orthonormal*.

According to the theory of generalised Fourier series, we can express a given function as a linear combination of the above orthogonal functions as follows:

$$f(t) = C_0\phi_0(t) + C_1\phi_1(t) + C_2\phi_2(t) + C_3\phi_3(t) + \cdots + C_n\phi_n(t) + \cdots + C_\infty\phi_\infty(t)$$

Generalised Fourier analysis consists in finding the values of the coefficients C_0, C_1, C_2,..., C_n. On the other hand synthesis works in opposite direction, i.e., given coefficients, we can synthesise a time domain function as illustrated in Figure 6.2.

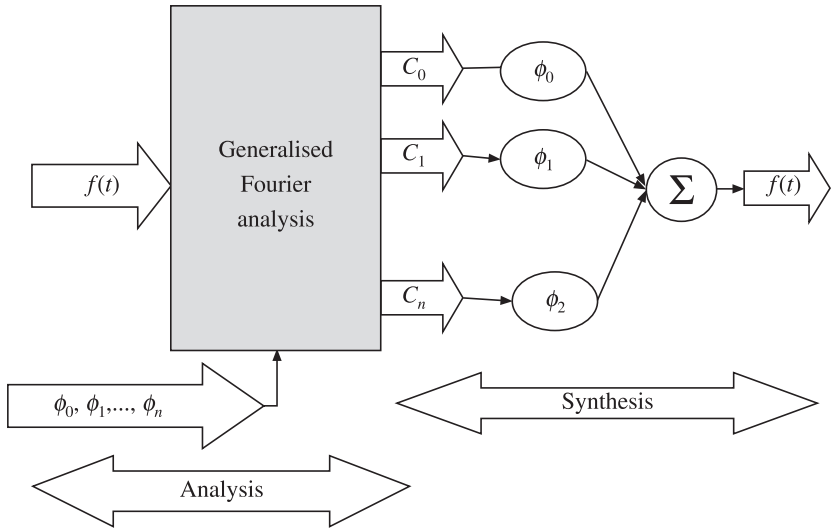


Figure 6.2 Basic idea of frequency analysis and synthesis.

Now, how to find the coefficients C_0, C_1, C_2 etc.? This is where the property of orthogonality becomes extremely useful. If we multiply both sides of above equation by ϕ_0 and integrate between '*a*' and '*b*', we get:

$$\int_a^b f(t)\phi_0(t)dt = \int_a^b C_0\,\phi_0^2(t)dt + C_1\int_a^b \phi_0(t)\phi_1(t)dt + \cdots + C_n\int_a^b \phi_0(t)\phi_n(t)dt + \cdots$$

Because of orthogonality, all integrals of the type

$$\int_a^b \phi_m(t)\phi_n(t)dt = 0 \quad \text{and}$$

$$\int_a^b \phi_n(t)\phi_n(t)dt = \int_a^b \phi_n^2(t)dt = k_0 \text{ (for orthonormal functions } k_0 = 1).$$

Thus, we get:

$$\int_a^b f(t)\phi_0(t)dt = k_0\,C_0$$

$$C_0 = \frac{1}{k_0}\int_a^b f(t)\phi_0(t)dt$$

In general,

$$C_n = \frac{1}{k_n}\int_a^b f(t)\phi_n(t)dt$$

Thus, multiplying by $\phi_n(t)$ and integrating between '*a*' and '*b*' has the effect of filtering out all the other terms except the '*n*' the term.

6.1.2 Dirichlet Conditions

The conditions under which it is possible to write Fourier series for a periodic function are known as *Dirichlet conditions*.

They require that the periodic function in each period:

1. Have a finite number of discontinuities
2. Have a finite number of maxima and minima
3. Be absolutely convergent, i.e., $\int_0^T |f(t)|\,dt < \infty$

Fortunately, most signals encountered in practice obey Dirichlet conditions.

6.1.3 Fourier Series in Trigonometric Form

The Fourier series can be written in the trigonometric form as:

$$f(t) = a_0 + a_1\cos(\omega_0 t) + a_2\cos(2\omega_0 t) + a_3\cos(3\omega_0 t) + \cdots + a_n\cos(n\omega_0 t)$$
$$+ b_1\sin(\omega_0 t) + b_2\sin(2\omega_0 t) + b_3\sin(3\omega_0 t) + \cdots + b_n\sin(n\omega_0 t)$$

This can also be written in compact notation as:

$$f(t) = a_0 + \sum_{n=1}^{\infty}(a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t))$$

Note that the above form is '*phase angle free*'. The phase angle is not explicitly stated but is encoded in the ratio of b_n and a_n. Another form which is sometimes used is a cosine wave with phase angle, explicitly stated. To develop that form consider:

$$C_n \cos(n\omega_0 t + \theta_n)$$

i.e., $$C_n \cos(n\omega_0 t + \theta_n) = C_n \cos(n\omega_0 t)\cos(\theta_n) - C_n \sin(n\omega_0 t)\sin(\theta_n)$$

or, $$C_n \cos(n\omega_0 t + \theta_n) = C_n \cos(\theta_n)\cos(n\omega_0 t) - C_n \sin(\theta_n)\sin(n\omega_0 t)$$

Let $$a_n = C_n \cos(\theta_n) \text{ and } b_n = -C_n \sin(\theta_n)$$

Let $$C_0 = a_0, \text{ then } C_n = \sqrt{(a_n^2) + (b_n^2)} \text{ and } \theta_n = -\tan^{-1}\left(\frac{b_n}{a_n}\right) \text{ (Note: – ve sign)}$$

Then, $$C_n \cos(n\omega_0 t + \theta_n) = a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)$$

Hence, alternate representation of the Fourier series will be:

$$f(t) = C_0 + \sum_{n=0}^{\infty} C_n \cos(n\omega_0 t + \theta_n)$$

The coefficients of the Fourier series can be found by evaluating the following integrals:

$$a_0 = \frac{1}{T}\int_0^T f(t)dt$$ … is the average value of the function over a cycle

$$a_n = \frac{2}{T}\int_0^T f(t)\cos(n\omega_0 t)dt$$

$$b_n = \frac{2}{T}\int_0^T f(t)\sin(n\omega_0 t)dt$$

The above simple expressions for the coefficient become possible because of the orthogonal nature of the following set of functions, which form the 'basis' functions of Fourier series:

$\{1, \cos(\omega_0 t), \cos(2\omega_0 t), \cos(3\omega_0 t), \ldots, \cos(n\omega_0 t), \sin(\omega_0 t), \sin(2\omega_0 t), \sin(3\omega_0 t), \ldots, \sin(n\omega_0 t)\}$

The orthogonal relationship between the 'basis' functions is captured in Table 6.3.

Table 6.3 Orthogonal relationship between basis functions

	1 dt	**$\sin(m\omega_0 t)dt$**	**$\sin(n\omega_0 t)dt$**	**$\cos(m\omega_0 t)dt$**	**$\cos(n\omega_0 t)dt$**
$\int_0^T \sin(m\omega_0 t)$	zero	$\frac{T}{2}=\pi$	zero	zero	zero
$\int_0^T \sin(n\omega_0 t)$	zero	zero	$\frac{T}{2}=\pi$	zero	zero
$\int_0^T \cos(m\omega_0 t)$	zero	zero	zero	$\frac{T}{2}=\pi$	zero
$\int_0^T \cos(n\omega_0 t)$	zero	zero	zero	zero	$\frac{T}{2}=\pi$

6.1.4 Fourier Series in Exponential Form

We have the trigonometric form of Fourier series as:

$$f(t)=a_0+a_1\cos(\omega_0 t)+a_2\cos(2\omega_0 t)+a_3\cos(3\omega_0 t)+\cdots+a_n\cos(n\omega_0 t)$$
$$+b_1\sin(\omega_0 t)+b_2\sin(2\omega_0 t)+b_3\sin(3\omega_0 t)+\ldots+b_n\sin(n\omega_0 t)$$
$$f(t)=a_0+\sum_{n=1}^{\infty}(a_n\cos(n\omega_0 t)+b_n\sin(n\omega_0 t))$$

By De Movire's theorem we have:

$$\cos(n\omega_0 t)=\frac{1}{2}(e^{jn\omega_0 t}+e^{-jn\omega_0 t}) \text{ and } \sin(n\omega_0 t)=\frac{1}{2j}(e^{jn\omega_0 t}-e^{-jn\omega_0 t})$$

Hence, we can write $f(t)$ as:

$$f(t)=a_0+\sum_{n=1}^{\infty}a_n\left[\frac{(e^{jn\omega_0 t}+e^{-jn\omega_0 t})}{2}\right]+b_n\left[\frac{(e^{jn\omega_0 t}-e^{-jn\omega_0 t})}{2j}\right]$$

$$f(t)=a_0+\sum_{n=1}^{\infty}\left[\left(\frac{a_n-jb_n}{2}\right)e^{jn\omega_0 t}+\left(\frac{a_n+jb_n}{2}\right)e^{-jn\omega_0 t}\right]$$

Let $a_0=C_0, \frac{a_n-jb_n}{2}=C_n^{\sim}$ and $\frac{a_n+jb_n}{2}=C_{-n}^{\sim}$

$$\left|C_n^{\sim}\right|=\frac{1}{2}\sqrt{a_n^2+b_n^2}=\frac{1}{2}C_n$$

where C_n is the coefficient in the cosine-phase angle form of Fourier series. $f(t)$ can now be expressed in compact form

$$f(t) = \sum_{n=-\infty}^{\infty} [C_n^{\sim} e^{jn\omega_0 t}]$$

Where the coefficients $C_n^{\sim}$ are obtained by:

$$C_n^{\sim} = \frac{1}{T}\int_0^T f(t)\cos(n\omega_0 t)dt - j\frac{1}{T}\int_0^T f(t)\sin(n\omega_0 t)dt$$

$$C_n^{\sim} = \frac{1}{T}\int_0^T f(t)\left[\cos(n\omega_0 t) - j\sin(n\omega_0 t)\right]dt$$

$$C_n^{\sim} = \frac{1}{T}\int_0^T f(t)e^{-jn\omega_0 t}\,dt$$

6.2 Fourier Coefficients are Best in LSQ Sense

Fourier analysis can be visualised in terms of 'curve fitting'. Given an arbitrary waveform, we are trying to find which sine and cosine curves can be used to 'fit the given curve'. The Fourier coefficients encode the magnitudes and phases of these sines and cosines.

We have the Fourier series:

$$f(t) = a_0 + \sum_{n=1}^{\infty}(a_n\cos(n\omega_0 t) + b_n\sin(n\omega_0 t))$$

Fourier analysis is a search for the values of a_n and b_n coefficients which fit the given curve with the help of sines and cosines. We should try to find the values of a_n and b_n which are optimal in the LSQ sense. Assuming 'N' number of terms in the series, we can express the error as:

$$\text{error} = f(t) - \left[a_0 + \sum_{n=1}^{N}(a_n\cos(n\omega_0 t) + b_n\sin(n\omega_0 t))\right]$$

Sum of squared errors can be written as:

$$\text{SSE} = \int_0^T \left[f(t) - \left[a_0 + \sum_{n=1}^{N}(a_n\cos(n\omega_0 t) + b_n\ \sin(n\omega_0 t))\right]\right]^2 dt$$

For minimising the SSE, we impose the conditions that:

$$\frac{d(\text{SSE})}{da_0} = 0;\ \frac{d(\text{SSE})}{da_n} = 0;\ \text{and}\ \frac{d(\text{SSE})}{db_n} = 0$$

$$\frac{d(\text{SSE})}{da_0} = 0 = \int_0^T 2\left[f(t) - \left[a_0 + \sum_{n=1}^{N}(a_n\cos(n\omega_0 t) + b_n\ \sin(n\omega_0 t))\right]\right]dt\ (-1)$$

$$0 = \int_0^T 2\, f(t)dt - 2a_0 \int_0^T dt - 2\int_0^T \sum_{n=1}^{N} (a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t))dt$$

Note that each of the term within the summation sign integrates to zero. Hence, we are left with:

$$0 = \int_0^T 2\, f(t)dt - 2a_0 \int_0^T dt$$

$$0 = \int_0^T f(t)dt - a_0 \int_0^T dt$$

$$a_0 \int_0^T dt = \int_0^T f(t)dt$$

$$a_0 T = \int_0^T f(t)dt$$

$$a_0 = \frac{1}{T}\int_0^T f(t)dt$$ .... which is the same expression derived earlier. Hence, a_0 is best in the LSQ sense.

To find the best value of a_n: $\frac{d\,\text{SSE}}{d\,a_n} = 0$

$$\frac{d(\text{SSE})}{da_n} = 0 = \int_0^T 2\left[f(t) - \left[a_0 + \sum_{n=1}^{N} (a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)) \right] \right] (\cos(n\omega_0 t))dt$$

$$0 = \int_0^T f(t)\cos(n\omega_0 t)dt - a_0 \int_0^T \cos(n\omega_0 t)dt - a_n \int_0^T (\cos(n\omega_0 t))^2 dt$$

$$- \int_0^T \left[\sum_{n=1}^{N} b_n \sin(n\omega_0 t)\cos(n\omega_0 t) \right] dt - \int_0^T \left[\sum_{m=1, m\neq n}^{N} b_n \cos(m\omega_0 t)\cos(n\omega_0 t) \right] dt$$

All these integrals evaluate to zero:

$$a_0 \int_0^T \cos(n\omega_0 t)dt = 0$$

$$\int_0^T \left[\sum_{n=1}^{N} b_n \sin(n\omega_0 t)\cos(n\omega_0 t) \right] dt = \int_0^T \left[\sum_{m=1, m\neq n}^{N} b_n \cos(m\omega_0 t)\cos(n\omega_0 t) \right] dt = 0$$

and $\int_0^T (\cos(n\omega_0 t))^2 dt = \frac{T}{2}$; thus we are left with:

$$0 = \int_0^T f(t)\cos(n\omega_0 t)dt - a_n \frac{T}{2}$$

$$a_n = \frac{2}{T}\int_0^T f(t)\cos(n\omega_0 t)dt$$ …which is the same expression as obtained earlier.

Similar analysis gives:

$$b_n = \frac{2}{T}\int_0^T f(t)\sin(n\omega_0 t)dt$$

which is the same expression as obtained earlier. Thus a_n and b_n are indeed optimal in LSQ sense. Thus, if we do curve fitting with the restriction that only sines and cosines are to be used to fit the curve, we again arrive at Fourier coefficients!

6.3 Summary of Fourier Series

1. Trigonometric Form of Fourier Series (Phase Angle Free Form)

$$f(t) = a_0 + \sum_{n=1}^{\infty}(a_n\cos(n\omega_0 t) + b_n\sin(n\omega_0 t))$$

$a_0 = \frac{1}{T}\int_0^T f(t)dt$	$a_n = \frac{2}{T}\int_0^T f(t)\cos(n\omega_0 t)dt$	$b_n = \frac{2}{T}\int_0^T f(t)\sin(n\omega_0 t)dt$

2. Trigonometric Form of Fourier Series (Phase Angle Free Form)

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n\cos(n\omega_0 t) + b_n\sin(n\omega_0 t))$$

$a_n = \frac{2}{T}\int_0^T f(t)\cos(n\omega_0 t)dt$	$b_n = \frac{2}{T}\int_0^T f(t)\sin(n\omega_0 t)dt$

n = 1 to ∞

3. Trigonometric Form of Fourier Series with Cosine Wave and Phase Angle

$$f(t) = C_0 + \sum_{n=0}^{\infty} C_n\cos(n\omega_0 t + \theta_n)$$

$C_0 = a_0$	$C_n = \sqrt{(a_n^2) + (b_n^2)}$	$\theta_n = -\tan^{-1}\left(\frac{b_n}{a_n}\right)$ (Note: –ve sign)

4. Exponential Form of Fourier Series

$f(t) = \sum_{n=-\infty}^{\infty} [C_n^{\sim} e^{jn\omega_0 t}]$	Complex Coeff. $C_n^{\sim} = \left\|C_n^{\sim}\right\| e^{j\theta_n}$ $\theta_n = -\tan^{-1}\left(\frac{b_n}{a_n}\right)$ (Note: –ve sign)	
$a_0 = C_0$	$\frac{a_n - jb_n}{2} = C_n^{\sim}$	$\frac{a_n + jb_n}{2} = C_{-n}^{\sim}$
$C_n^{\sim} = \frac{1}{T}\int_0^T f(t)e^{-jn\omega_0 t}dt$		

6.4 Applications of Fourier Analysis

Fourier analysis is useful in a very large number of fields. In fact the double helix form of DNA was discovered in 1962 through X-ray diffraction techniques and Fourier analysis. NASA improves the clarity and detail of picture of celestial objects taken in space by means of Fourier analysis. Fourier analysis has found application in plasma physics, semiconductor physics, microwave acoustics, seismography, oceanography, radar mapping, medical imaging, chemical analysis and biology to name a few areas.

6.5 Fourier Series for a Recurring Pulse

Figure 6.3 shows a recurring or periodic pulse with a time period of T second and amplitude of V_0. For the sake of simplicity, pulse is taken symmetric about the $t = 0$ point. The width of the pulse is 'a' second. We will use this pulse as a vehicle for transition from Fourier series to Fourier transform.

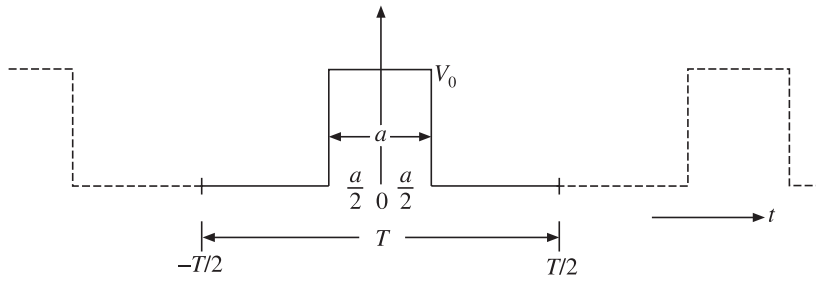


Figure 6.3 Recurring or periodic pulse.

The pulse is defined by:

$$f(t) = V_0; -\frac{a}{2} < t < \frac{a}{2}$$
$$= 0; -\frac{T}{2} < t < -\frac{a}{2}$$
$$= 0; \frac{a}{2} < t < \frac{T}{2}$$

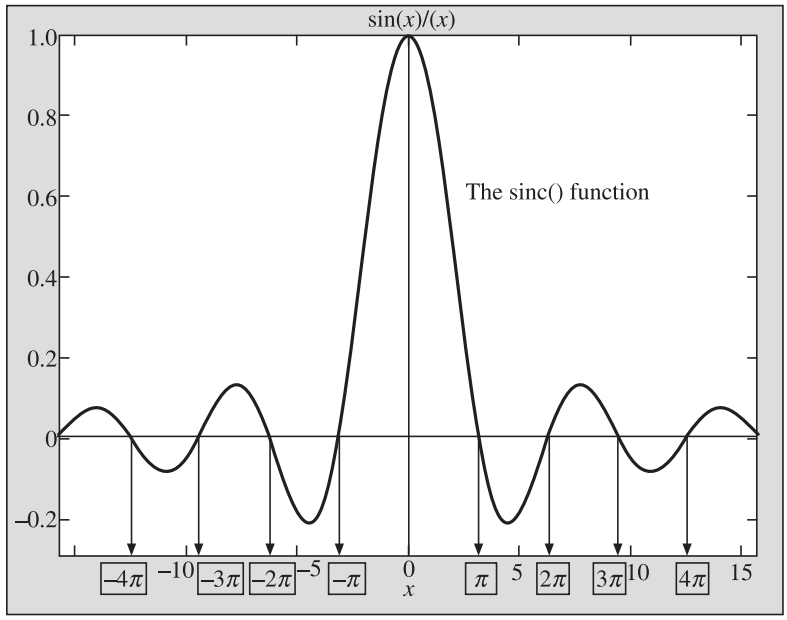


Figure 6.4 The sinc( ) function drawn using MATLAB ezplot(sin(x)/(x)).

First let us find the Fourier series representation for the pulse and plot its amplitude and phase spectrum. The complex Fourier coefficients $C_n^{\sim}$ are given by:

$$C_n^{\sim} = \int_0^T f(t)\, e^{-jn\omega_0 t} dt$$

$$C_n^{\sim} = \frac{V_0}{T} \int_{-a/2}^{a/2} e^{-jn\omega_0 t}\, dt$$

$$C_n^{\sim} = \frac{V_0}{T}\left[\frac{e^{-jn\omega_0 t}}{-jn\omega_0}\right]_{-a/2}^{a/2} = \frac{V_0}{T(-jn\omega_0)}\left[\frac{e^{-jn\omega_0 a/2} - e^{jn\omega_0 a/2}}{1}\right]$$

$$C_n^{\sim} = \frac{1}{T}\frac{V_0}{n2\pi f_0}\frac{2}{1}\frac{e^{jn\omega_0 a/2} - e^{-jn\omega_0 a/2}}{2j}$$

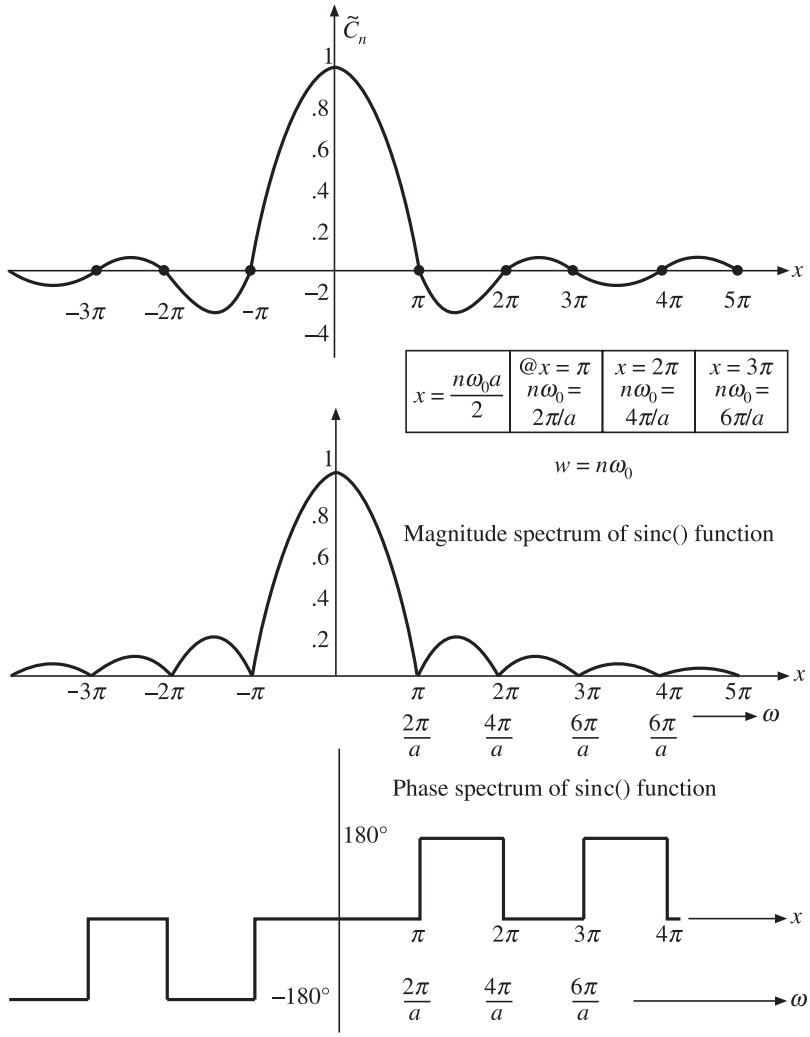


Figure 6.5 The sinc() function shown in more detail.

$C_n^{\sim} = \frac{V_0}{n\pi} \frac{e^{jn\omega_0 a/2} - e^{-jn\omega_0 a/2}}{2j}$; multiplying and dividing by $\left(n\frac{\omega_0 a}{2}\right)$, we get:

$$C_n^{\sim} = \frac{V_0}{n\pi} \frac{n}{1} \frac{\omega_0 a}{2} \frac{\sin(n\omega_0 a/2)}{(n\omega_0 a/2)}$$

$$C_n^{\sim} = \frac{V_0\, a}{T} \frac{\sin(n\omega_0 a/2)}{(n\omega_0 a/2)}$$

$$C_n^{\sim} = \frac{V_0 a}{T} \frac{\sin(x)}{(x)} \text{ where } x = \frac{n\omega_0 a}{2}$$

$$C_n^{\sim} = \frac{V_0 a}{T}\text{sinc}(x)$$

where $\text{sinc}(x)$ is defined as $\frac{\sin(x)}{(x)}$. It can be shown that the value of $\frac{\sin(x)}{(x)}$ is 1 when $x = 0$.

The '*c*' in sinc() comes from the Latin 'cardinalis'. The full name of the sinc() function is sinus-cardinalis. The plot of sinc() function using MATLAB ezplot() function is shown in Figure 6.4. In Figure 6.5 plot of $C_n^{\sim}$ and magnitude and phase spectrum of sinc() function are shown.

6.6 Recurring Pulse to Non-recurring Pulse

Because of the constraints imposed by finite computing power we cannot process infinite duration or (infinitely) *recurring signals*. The signals that we process, thus, can be said to exist during a finite window of time. We are forced to ignore everything outside this window. Thus, even if we are processing recurring signals they appear to be non-recurring! Since the Fourier theory is valid for infinitely recurring signals, strictly speaking, we cannot apply it to any of power system signals. However, Fourier technique is too valuable a tool to let go of. What, then, should we do?

We modify the Fourier theory itself so that it can be applied to non-recurring signals. We do this by using a simple mathematical trick. Consider what will happen as we stretch the time period *T* to infinity while keeping the 'on time' '*a*' constant as shown in Figure 6.6. Under such a situation there will be no more instances of the pulse and we are left with a single, non-recurring pulse of duration '*a*'.

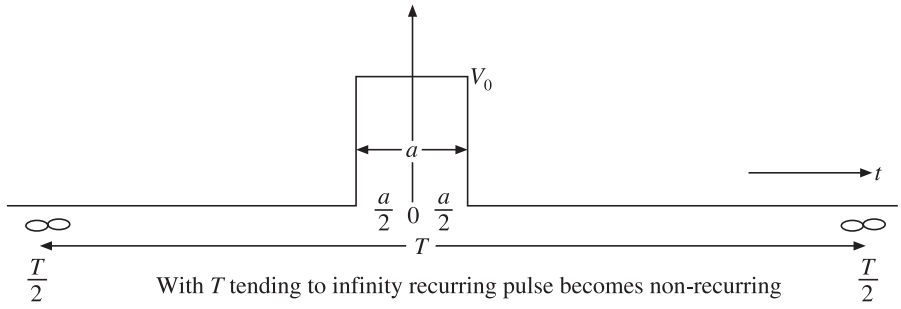


Figure 6.6 Transition from recurring to non-recurring pulse as $T \to \infty$.

It can be shown that as $T \to \infty$, the discrete frequency spectrum remains the same in shape but assumes the envelope of the spectrum of the recurring pulse. The envelope of $\frac{\sin(n\omega_0 a/2)}{(n\omega_0 a/2)}$ becomes $\frac{\sin(\omega a/2)}{(\omega a/2)}$. Note that in the first expression we have 'ω_0', a discrete frequency whereas in the second expression we have 'ω' the continuous frequency variable.

The mathematical treatment of the non-recurring pulse can thus be derived from the recurring pulse by letting the time period *T* approach ∞ in the limit. The changes that take place in the representation as we transit from recurring to non-recurring pulse are listed in Table 6.4.

Table 6.4

	Recurring pulse	**Single pulse**
Time period	Time period T is finite	Time period $T \to \infty$
Frequency	Fundamental frequency is $(1/T)$	Fundamental frequency f_{funda} is zero
Nature of frequency spectrum	Spectrum is discrete (spaced at f_{funda} from each other)	Spectrum is continuous (spaced at zero Hz from each other)
Summation/ Integration	Fourier sum $f(t) = \sum_{n=-\infty}^{\infty} C_n^{\sim} e^{jn\omega_0 t}$	Fourier integral $f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(j\omega) e^{j\omega t} dt$
Coefficients	$C_n^{\sim} = \frac{1}{T} \int_0^T f(t) e^{-jn\omega_0 t} dt$	$F(j\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$

Last row of Table 6.4 shows that the difference between the frequency spectrum of the recurring and non-recurring versions of a pulse is not in the shape. Both versions have the same shape. The difference is that the recurring pulse has discrete (discontinuous) frequency spectrum while the non-recurring pulse has continuous frequency spectrum. Thus, we can make the following observations:

1. The shape of the continuous amplitude and phase spectra for a non-recurring $f(t)$ is identical with the envelopes of the amplitude and phase line spectra for the same pulse recurring.
2. All frequencies are present in the continuous amplitude spectrum since $F(j\omega)$ is defined for all ω from $-\infty$ to $+\infty$.
3. Lightning, which is a very short duration pulse in time, makes its presence felt irrespective of the frequency of the radio station to which we may be tuned. (We do not hear the lightening crashes in our mobile phones because, we are not listening to the raw audio signal, the phone processes the audio signal to filter out noise using DSP techniques.)

We draw an important inference from the above discussion that in case of non-recurring pulse, we have to use Fourier integral instead of the Fourier summation. Fourier integral is also popularly known as *Fourier transform*. Note that the Fourier transform that has been described is a continuous transform. As can be expected, this continuous transform will have to give way to discrete transform. But this will be taken up in the next section.

6.7 Discrete Fourier Transform Development [5, 13, 19, 22, 33, 52, 61]

We have the exponential form of the Fourier series

$$f(t) = \sum_{n=-\infty}^{\infty} [C_n^{\sim} e^{jn\omega_0 t}]$$

where the coefficients $C_n^{\sim}$ are obtained by:

$$C_n^{\sim} = \frac{1}{T}\int_0^T f(t)\cos(n\omega_0 t)dt - j\frac{1}{T}\int_0^T f(t)\sin(n\omega_0 t)\,dt$$

$$C_n^{\sim} = \frac{1}{T}\int_0^T f(t)[\cos(n\omega_0 t) - j\sin(n\omega_0 t)]dt$$

$$C_n^{\sim} = \frac{1}{T}\int_0^T f(t)e^{-jn\omega_0 t}\,dt$$

Table 6.5

Fourier sum (Recurring signal)	**Fourier integral (Windowed signal)**
$f(t) = \sum_{n=-\infty}^{\infty} C_n^{\sim} e^{j\omega_0 n t}$	$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(j\omega)e^{j\omega t}dt$
Fourier Coefficients	**Fourier Coefficients**
$C_n^{\sim} = \frac{1}{T}\int_0^T f(t)e^{-jn\omega_0 t}dt$	$F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$

We develop the DFT by noting the expression for $F(j\omega)$ for windowed signal:

$$F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$$

Note that ω refers to angular frequency of the signal in the continuous domain, i.e., ω_{signal}. $\omega_{\text{signal}} = 2\pi f_{\text{signal}}$, where f_{signal} is the signal frequency in the continuous domain. Note that time 't' can take only discrete values $t = n\Delta t$; where $\Delta t = 1/f_{\text{sampling}}$ in the discrete domain and f_{signal} can take only discrete values corresponding to 'mf_{funda}'. Thus changing over to the discrete domain where $X(m)$ represents the Fourier coefficient at the mth harmonic frequency, we can write the discrete version of the Fourier integral as:

$$X(m) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi f_{\text{signal}}\, n\Delta t}$$

$$X(m) = \sum_{n=0}^{N-1} x(n)e^{\frac{-j2\pi m\, f_{\text{funda}} n}{f_{\text{sampling}}}}$$

Since we are taking 'N' samples, it implies that $f_{\text{sampling}} = N\, f_{\text{funda}}$

$$X(m) = \sum_{n=0}^{N-1} x(n)e^{\frac{-j2\pi m\, f_{\text{funda}} n}{N f_{\text{funda}}}}$$

Thus, we get the expression for DFT as:

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

We summarise the transition from CFT to windowed DFT as shown in Table 6.6.

Table 6.6 Transition of windowed DFT from CFT

Fourier sum (recurring continuous signal)	**Fourier Integral (windowed continuous signal)**	**Discrete Fourier Inverse (windowed sampled signal)**
$f(t) = \sum_{n=-\infty}^{\infty} C_n^{\sim} e^{j\omega_0 n t}$	$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(j\omega) e^{j\omega t} dt$	$x(n) = \frac{1}{N} \sum_{m=0}^{N-1} X(m) e^{\frac{j2\pi mn}{N}}$
Fourier coefficients	**Fourier coefficients**	**Fourier coefficients**
$C_n^{\sim} = \frac{1}{T} \int_0^T f(t) e^{-jn\omega_0 t} dt$	$F(j\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$	$X(m) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi mn}{N}}$

In the DFT expression, the following points are worth noting:

1. N = Number of samples
2. m = index for frequency domain sample, i.e., the order of the harmonic
   Both 'm' and 'n' run from 0 to (N–1) and n = index for time domain sample
3. The multiplier 1/N is included in IDFT but not in DFT definition
4. $X(m)$ is the frequency domain variable, the complex Fourier coefficient, representing the frequency content at the 'mth' harmonic frequency
5. $x(n)$ is the time domain variable, 'nth' sample of the signal

Note, however this does not mean that maximum order of harmonic in 'N–1'. The highest order of harmonic is limited to 'N/2'. Harmonics from '(N/2) + 1' to 'N–1' are in fact complex conjugate of harmonics from '(N/2)–1' to '1'.

6.8 Implicit Assumption behind Windowed Fourier Transform

Implicit assumption in DFT is that there are infinite repetitions of the window, both, before as well as after the window. Fourier analysis cannot but represent the signal in any other form. Thus, if we collect eight signal samples of 1,1,1,1,1,1,1,1 in the window, then implicit assumption causes DFT to add infinite repetitions of the window, both before and after the window, as shown in Figures 6.7 and 6.8.

In Figure 6.7, the implicit assumption causes DFT to infer that the signal is a DC signal. This is represented on the frequency spectrum as a single line at a frequency of zero hertz.

In Figure 6.8, the implicit assumption causes DFT to represent the signal as a sinc() function in the frequency domain.

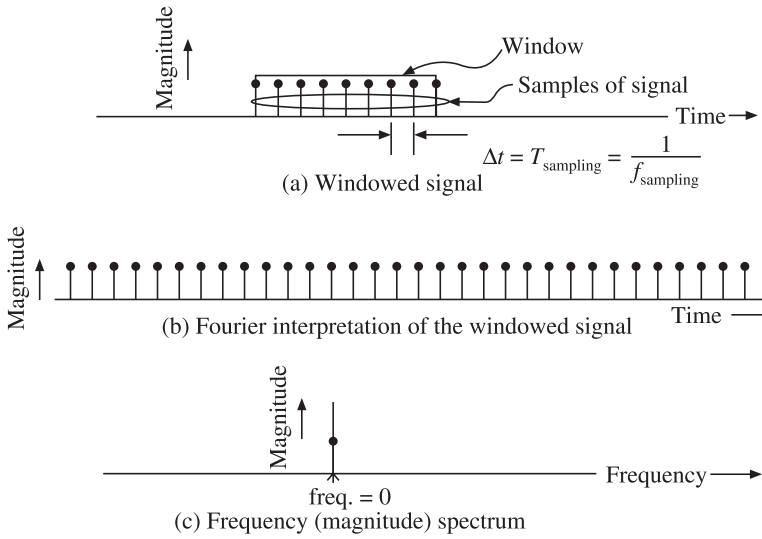


Figure 6.7 Fourier interpretation of windowed signal.

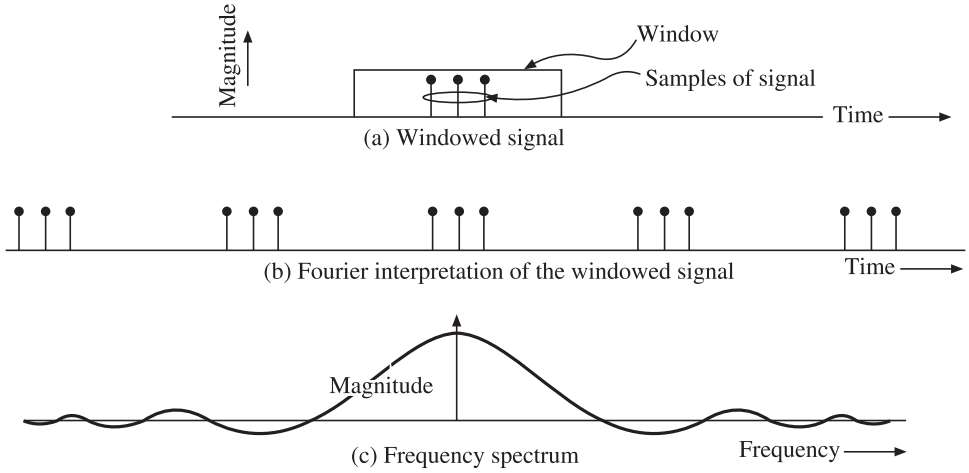


Figure 6.8 Another example of Fourier interpretation of windowed signal.

6.9 Meaning of '*N*' Samples Collected in a Window

Let us say, we have a window in which (N = 12), samples of a signal sampled at 600 Hz have been collected. This statement contains a lot of information! As per the implicit assumption in DFT, it automatically means that these 12 samples constitute a full cycle of the signal. Hence, time period of the signal, T_{signal} can be written as:

$$T_{\text{signal}} = N\,T_{\text{sampling}} = N\Delta t = N\frac{1}{f_{\text{sampling}}} = \frac{12}{600} = \frac{1}{50}\text{s}$$

Hence, frequency of the signal $f_{\text{signal}} = 1/T_{\text{signal}} = 50$ Hz.

It is worth noting that Fourier analysis assumes that we know the frequency of the signal apriori because otherwise we would not have been able to decide the sampling frequency. The primary purpose of Fourier analysis is not frequency measurement. Its purpose is to estimate magnitudes of various frequency components. The frequencies of these components automatically are tied up with the sampling frequency and therefore known beforehand (apriori). Thus we can write:

$$T_{\text{funda}} = N\,T_{\text{sampling}} \;\Rightarrow\; \frac{1}{f_{\text{funda}}} = N\frac{1}{f_{\text{sampling}}} \;\Rightarrow\; f_{\text{sampling}} = Nf_{\text{funda}} \;\Rightarrow\; f_{\text{funda}} = \frac{f_{\text{sampling}}}{N}$$

This is illustrated in Figure 6.9.

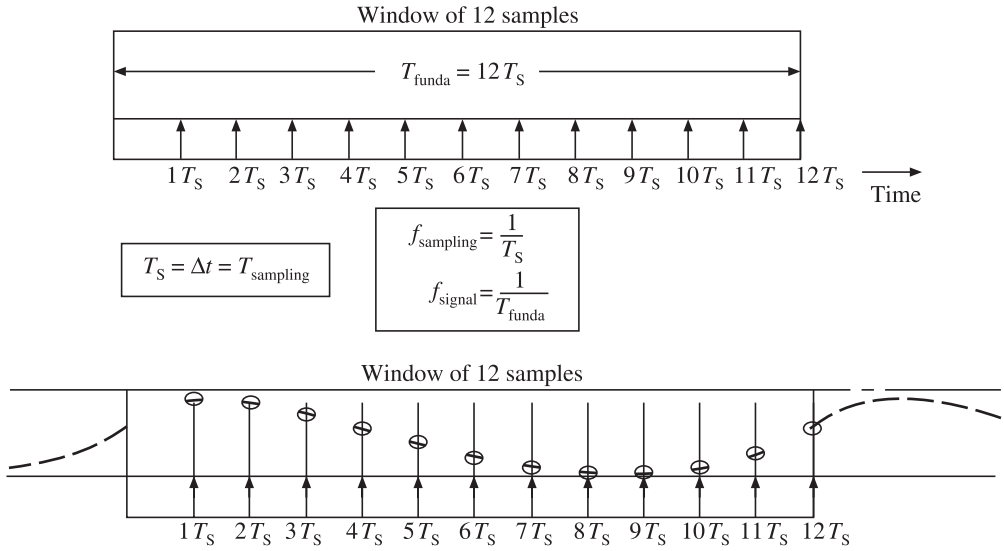


Figure 6.9 Meaning of collecting 12 samples in the window.

6.10 Redundancy of DFT

All the '*N*' number of DFT terms are not unique! Only half the terms are unique. The second half of the DFT terms are complex conjugates of corresponding first half of the terms. This will become clear when we try to find the relationship between $X(m)$ and $X(N - m)$.

We have:

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

Hence, we can write $X(N - m)$ as:

$$X(N-m) = \sum_{n=0}^{N-1} x(n)e^{\frac{-j2\pi(N-m)n}{N}}$$

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

$$X(N-m)=\sum_{n=0}^{N-1}x(n)e^{-j2\pi n}e^{\frac{+j2\pi mn}{N}}$$

Note that $e^{-j2\pi n}=1$ for $n=0, 1, 2, 3,\ldots$

Hence,

$$X(N-m)=\sum_{n=0}^{N-1}x(n)e^{\frac{+j2\pi mn}{N}}$$

Thus, the only difference between $X(m)$ and $X(N-m)$ is the sign of the exponents, which is reversed. Hence, $X(N-m)$ terms are complex conjugate of $X(m)$ terms. Thus, the magnitudes of $X(N-m)$ and $X(m)$ will be identical but their phase angles will be negative of each other. Hence, we can write:

$X(N-m)=\{X(m)\}^*$ or $\{X(N-m)\}^*=X(m)$ ← We are assuming here that the signal $x(n)$ is real.

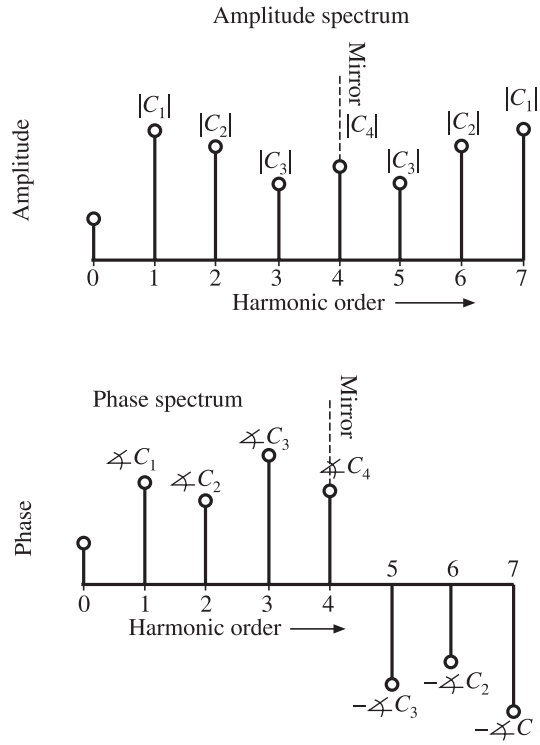


Figure 6.10 Spectrum of DFT.

This is depicted in Figure 6.10 as well as in Table 6.7.

Table 6.7

For $N = 8$; assuming that the signal $x(n)$ is real.							
$m = 0$	$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$	$m = 6$	$m = 7$
$N - m = 8$	$N - m = 7$	$N - m = 6$	$N - m = 5$	$N - m = 4$	$N - m = 3$	$N - m = 2$	$N - m = 1$
$X(0) = X(8)^*$	$X(1) = X(7)^*$	$X(2) = X(6)^*$	$X(3) = X(5)^*$	$X(4) = X(4)^*$	$X(5) = X(3)^*$	$X(6) = X(2)^*$	$X(7) = X(1)^*$

Note: $X(8)$ does not exist and $X(0)$ is always real. Since $X(4)^* = X(4)$, $X(4)$ is also real. Remaining Fourier coefficients are complex.

6.11 DFT as a Mapping

Mathematicians sometimes look at DFT purely as a transformation of one vector consisting of real samples of signal into another vector consisting of the complex values of the coefficients. Thus, we have 'N' real numbers at the input of this process and we get 'N' complex numbers at the output of this process as shown in Figure 6.11 for $N = 8$.

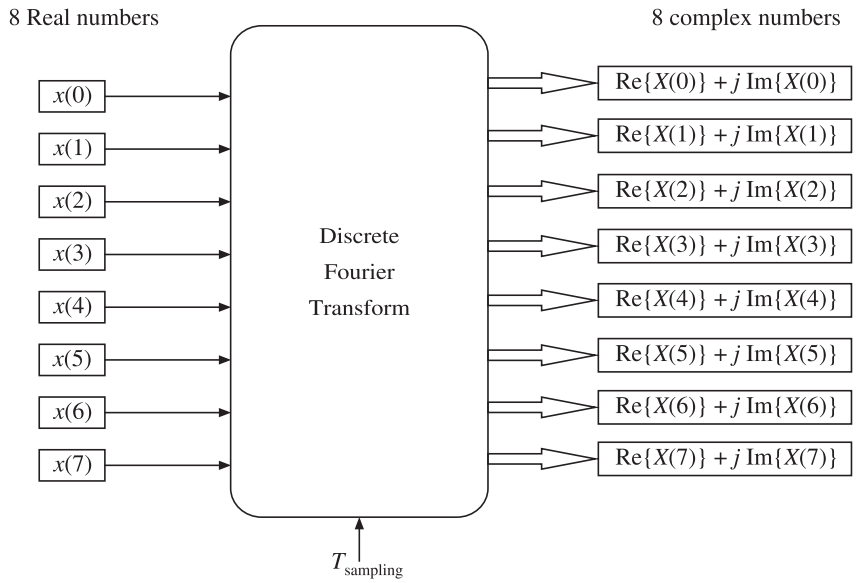


Figure 6.11 DFT as a mapping of 'N' real numbers into N complex numbers.

6.12 Numerical Calculation of DFT Coefficients

EXAMPLE 6.1 A power system signal with nominal frequency of 50 Hz was sampled at 600 Hz. The following 12 samples were obtained.

Sample no.	0	1	2	3	4	5
Value	253.000	275.2891	161.2628	77.0000	74.0526	65.7109
Sample no.	6	7	8	9	10	11
Value	33.0000	31.9212	35.9474	–11.0000	–29.2628	89.0788

Find the values of:

1. the DC offset,
2. magnitude and phase angle of fundamental, and
3. magnitude and phase angle of second harmonic component.

Solution We shall demonstrate the solution of the above numerical problem by (a) hand calculation, (b) through a spreadsheet and (c) by using a MATLAB program and by executing fft(), the MATLAB function for DFT .

(a) Computation of DFT by Hand

In the expression for DFT; putting $N = 12$ and running 'n' from 0 to '$N - 1$':

$$\underset{m=0}{\overset{m=N-1}{X(m)}} = \sum_{n=0}^{n=N-1} x(n)e^{-j\,2\pi mn/N}$$

(i) For DC component $m = 0$

$$X(0) = \sum_{n=0}^{n=11} x(n)e^{-j2\pi 0\frac{n}{12}} = x(0) + x(1) + x(2) + x(3) + x(4) + x(5) + x(6) + x(7) + x(8) + x(9) + x(10) + x(11)$$

$X(0) = 253.000 + 275.2891 + 161.2628 + 77.0000 + 74.0526 + 65.7109 + 33.0000 + 31.9212 + 35.9474 – 11.0000 – 29.2628 + 89.0788 = 1056$

> ***Note:*** The DFT formula does not include the multipliers of '1/N' for DC term and '2/N' for other terms. This is to speed-up the DFT calculation by avoiding an extra division. The additional division is time consuming and hence avoided. The result is not wrong because we can always factor in the 1/N and 2/N terms. Sometimes these may also get cancelled out during division. For example, if we are finding impedance then 2/N factor in voltage and 2/N factor in current will get cancelled.

Actual value of DC offset is $(1/N)^* X(0) = (1/12)^*(1056) = 88$ (Answer)

(ii) For fundamental $m = 1$ and $n = 0$ to 11 in the general expression for DFT:

$$X(1) = \sum_{n=0}^{n=11} x(n)e^{-j\,2\pi 1n/12} = \sum_{n=0}^{n=11} x(n)e^{-j\,\pi n/6} = \sum_{n=0}^{n=11} x(n)e^{-j\,(\frac{\pi}{6})n}$$

$$X(1)=\sum_{n=0}^{n=11}x(n)e^{-j\left(\frac{\pi}{6}\right)n}=x(0)e^{-j\left(\frac{\pi}{6}\right)0}+x(1)e^{-j\left(\frac{\pi}{6}\right)1}+x(2)e^{-j\left(\frac{\pi}{6}\right)2}+x(3)e^{-j\left(\frac{\pi}{6}\right)3}$$
$$+x(4)e^{-j\left(\frac{\pi}{6}\right)4}+x(5)e^{-j\left(\frac{\pi}{6}\right)5}+x(6)e^{-j\left(\frac{\pi}{6}\right)6}+x(7)e^{-j\left(\frac{\pi}{6}\right)7}+x(8)e^{-j\left(\frac{\pi}{6}\right)8}+x(9)e^{-j\left(\frac{\pi}{6}\right)9}$$
$$+x(10)e^{-j\left(\frac{\pi}{6}\right)10}+x(11)e^{-j\left(\frac{\pi}{6}\right)11}$$

On the CASIO FX-991ES and similar calculators, we can easily evaluate the above equation by noting that $re^{j\theta}=r\angle\theta$. Hence, we can write the expression for $X(1)$ as follows:

$$X(1)=\sum_{n=0}^{n=11}x(n)e^{-j\left(\frac{\pi}{6}\right)n}=x(0)+x(1)\angle\left(\frac{-1.\pi}{6}\right)+x(2)\angle\left(\frac{-2.\pi}{6}\right)+x(3)\angle\left(\frac{-3.\pi}{6}\right)$$
$$+x(4)\angle\left(\frac{-4.\pi}{6}\right)+x(5)\angle\left(\frac{-5.\pi}{6}\right)+x(6)\angle\left(\frac{-6.\pi}{6}\right)+x(7)\angle\left(\frac{-7.\pi}{6}\right)+x(8)\angle\left(\frac{-8.\pi}{6}\right)$$
$$+x(9)\angle\left(\frac{-9.\pi}{6}\right)+x(10)\angle\left(\frac{-10.\pi}{6}\right)+x(11)\angle\left(\frac{-11.\pi}{6}\right)$$

Converting radians to degrees for convenience:

$$X(1)=\sum_{n=0}^{n=11}x(n)e^{-j\left(\frac{\pi}{6}\right)n}=x(0)+x(1)\angle-30^\circ+x(2)\angle-60+x(3)\angle-90+x(4)\angle-120$$
$$+x(5)\angle-150+x(6)\angle-180+x(7)\angle-210+x(8)\angle-240+x(9)\angle-270+x(10)\angle$$
$$-300+x(11)\angle-330$$

$$X(1)=\sum_{n=0}^{n=11}x(n)e^{-j\left(\frac{\pi}{6}\right)n}=253+275.2891\angle-30^\circ+161.2628\angle-60^\circ+77.0000\angle-90^\circ$$
$$+74.0526\angle-120^\circ+65.7109\angle-150^\circ+33.0000\angle-180^\circ+31.9212\angle-210^\circ+35.9474\angle$$
$$-240^\circ-11.0000\angle-270^\circ-29.2628\angle-300^\circ+89.0788\angle-330^\circ$$

The steps to calculate the above for CASIO FX-991ES calculator are as follows:

1. Press [**Shift**] + [**Mode/Setup**] and then [3] for degrees
2. Press [Mode/Setup] and then [2] for complex mode
3. Press [253] + [275.2891] [Shift][(–)] when you do this ∠ appears on the LCD then enter –30°
4. The LCD shows the following string for steps up till now:
   253 + 275.2891 ∠ 30° press [+]
5. Go on entering other terms in a similar fashion finally pressing [=]

6. If all is well you will see the answer on the LCD as:

$$461.999789 - 396.0000809i$$

$$\frac{2}{N}C_1 = \frac{2}{12}(461.999789 - 396.0000809\,i) = (a_1 - jb_1)$$

Hence,

$a_1 = (461.999789/6) = 76.99996483 \cong 77$ (Answer)

and $b_1 = (396.000809/6) = 66.00001348 \cong 66$ (Answer)

(iii) To find a_2 and b_2 of 2nd harmonic for $m = 2$ and $n = 0$ to 11 in the general expression for DFT:

$$X(2) = \sum_{n=0}^{n=11} x(n)e^{-j2\pi 2n/12} = \sum_{n=0}^{n=11} x(n)e^{-j\pi n/3} = \sum_{n=0}^{n=11} x(n)e^{-j\left(\frac{\pi}{3}\right)n}$$

$$X(2) = \sum_{n=0}^{n=11} x(n)e^{-j\left(\frac{\pi}{3}\right)n} = x(0)e^{-j\left(\frac{\pi}{3}\right)0} + x(1)e^{-j\left(\frac{\pi}{3}\right)1} + x(2)e^{-j\left(\frac{\pi}{3}\right)2} + x(3)e^{-j\left(\frac{\pi}{3}\right)3}$$

$$+x(4)e^{-j\left(\frac{\pi}{3}\right)4} + x(5)e^{-j\left(\frac{\pi}{3}\right)5} + x(6)e^{-j\left(\frac{\pi}{3}\right)6} + x(7)e^{-j\left(\frac{\pi}{3}\right)7} + x(8)e^{-j\left(\frac{\pi}{3}\right)8}$$

$$+x(9)e^{-j\left(\frac{\pi}{3}\right)9} + x(10)e^{-j\left(\frac{\pi}{3}\right)10} + x(11)e^{-j\left(\frac{\pi}{3}\right)11}$$

$$X(2) = \sum_{n=0}^{n=11} x(n)e^{-j\left(\frac{\pi}{3}\right)n} = x(0) + x(1)\angle\left(\frac{-1.\pi}{3}\right) + x(2)\angle\left(\frac{-2.\pi}{3}\right) + x(3)\angle\left(\frac{-3.\pi}{3}\right)$$

$$+ x(4)\angle\left(\frac{-4.\pi}{3}\right) + x(5)\angle\left(\frac{-5.\pi}{3}\right) + x(6)\angle\left(\frac{-6.\pi}{3}\right) + x(7)\angle\left(\frac{-7.\pi}{3}\right) + x(8)\angle\left(\frac{-8.\pi}{3}\right)$$

$$+x(9)\angle\left(\frac{-9.\pi}{3}\right) + x(10)\angle\left(\frac{-10.\pi}{3}\right) + x(11)\angle\left(\frac{-11.\pi}{3}\right)$$

$$X(2) = \sum_{n=0}^{n=11} x(n)e^{-j\left(\frac{\pi}{3}\right)n} = 253 + 275.2891\angle -60° + 161.2628\angle -120° + 77.0000\angle -180°$$

$$+ 74.0526\angle -240° + 65.7109\angle -300° + 33.0000\angle -360° + 31.9212\angle -420° + 35.9474\angle$$

$$-480° - 11.0000\angle -540° - 29.2628\angle -600° + 89.0788\angle -660°$$

Result on the calculator will be:

$$X(2) = 330 - 264.00005011i$$

$$\frac{2}{N}C_2 = \frac{2}{12}(330 - 264.00005011i) = (a_2 - jb_2)$$

Hence,

$$a_2 = \frac{330}{6} = 55 \quad \text{(Answer)}$$

$$b_2 = \left(\frac{264.00005011}{6}\right) = 44 \quad \text{(Answer)}$$

Let us tabulate the results:

DC $a_0 = 88$	Fundamental		Second harmonic	
	$a_1 = 77$	$b_1 = 66$	$a_2 = 55$	$b_2 = 44$
	$\lvert c_1\rvert = \sqrt{(a_1)^2 + (b_1)^2}$		$\lvert c_2\rvert = \sqrt{(a_2)^2 + (b_2)^2}$	
	$\lvert c_1\rvert = \sqrt{(77)^2 + (66)^2}$		$\lvert c_2\rvert = \sqrt{(55)^2 + (44)^2}$	
	$\lvert c_1\rvert = 101.414999$		$\lvert c_2\rvert = 70.43436661$	
	$\theta_1 = \tan^{-1}(b_1/a_1)$		$\theta_2 = \tan^{-1}(b_2/a_2)$	
	$\theta_1 = \tan^{-1}(66/77)$		$\theta_2 = \tan^{-1}(44/55)$	
	$\theta_1 = 40.60129465°$		$\theta_2 = 38.65980825°$	

(b) A spreadsheet for DFT calculation

The samples of the signal given in Example 6.1 are entered in an Excel spreadsheet. The first three harmonics are computed in the spreadsheet as shown in Figure 6.12. It can be seen that the results of the spreadsheet match with hand calculation.

The answers obtained from the spreadsheet are shown in the tabular form:

DC	a_1	b_1	a_2	b_2	a_3	b_3
88	77	66	55	44	33	22

(c) MATLAB program for DFT calculation

```
% Numerical computatio of DFT in MATLAB
clear , clc
fsig = 50 ;
N = 12 ;
fs = N * fsig ;
dt = 1/fs ;
% ------------------------------------------------------------
%---------- Samples of signal ----------------------------
%------ x(o) to x(11) are written as x(1) to x(12) since
matrix /vector index of zero
% ----- is not allowed in MATLAB ---------------------------
x(1)= 253.0000 ; x(2) = 275.2891 ; x(3) = 161.2628;
x(4)= 77.0000 ; x(5) = 74.0526 ; x(6) = 65.7109 ;
```

Spreadsheet for 12 point DFT									
			w =	314.15927	rad				
			f =	50	Hz	Samp/cycle ==>	12	F samp=>	600
			DC	a1	b1	a2	b2	a3	b3
		Values Taken ===>	99	30	40	6	8	3	4
	v(t) = DC + a1 * cos(wt) + b1 * sin(w t)+ a2* cos(2*wt) + b2 *sin(2*wt)+ a3* cos(3* wt) + b3 *sin(3* w t)								
wt(deg)	SampNo	t	Samples of v	v*cos(wt)	V*sin(wt)	v*cos(2wt)	v*sin(2wt)	v*cos(3wt)	v*sin(3wt)
0	0	0.0000	138.0000	138.0000	0.0000	138.0000	0.0000	138.0000	0.0000
30	1	0.0017	158.9090	137.6192	79.4545	79.4545	137.6192	0.0000	158.9090
60	2	0.0033	149.5692	74.7846	129.5307	-74.7846	129.5307	-149.5692	0.0000
90	3	0.0050	129.0000	0.0000	129.0000	-129.0000	0.0000	0.0000	-129.0000
120	4	0.0067	111.7128	-55.8564	96.7461	-55.8564	-96.7461	111.7128	0.0000
150	5	0.0083	93.0910	-80.6192	46.5455	46.5455	-80.6192	0.0000	93.0910
180	6	0.0100	72.0000	-72.0000	0.0000	72.0000	0.0000	-72.0000	0.0000
210	7	0.0117	58.9474	-51.0500	-29.4737	29.4737	51.0500	0.0000	-58.9474
240	8	0.0133	56.2872	-28.1436	-48.7461	-28.1436	48.7461	56.2872	0.0000
270	9	0.0150	57.0000	0.0000	-57.0000	-57.0000	0.0000	0.0000	57.0000
300	10	0.0167	66.4308	33.2154	-57.5307	-33.2154	-57.5307	-66.4308	0.0000
330	11	0.0183	97.0526	84.0500	-48.5263	48.5263	-84.0500	0.0000	-97.0526
		Sum=	1188	180	240	36	48	18	24
		Computed	99	30	40	6	8	3	4
			DC	a1	b1	a2	b2	a3	b3
		Error	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
			Mags	50.0000		10.0000		5.0000	
			Ph(rad)	0.9273		0.9273		0.9273	
			Ph(deg)	53.1301		53.1301		53.1301	

Figure 6.12 12 point DFT computation using Excel spreadsheet.

```
   x(7)= 33.0000 ; x(8) = 31.9212 ; x(9) = 35.9474 ;
   x(10)= -11.0000 ; x(11) = -29.2628 ; x(12)= 89.0788 ;
   %---------------- DFT calculation ---------------------------
   % Initialization of coeffs. to zero before start of
calculations
   for m = 0:N-1
    X(m+1)= 0 ;
   end
   %------------- Actual Caluculation of DFT ------------------
   for m = 0:N-1
    for n=0:N-1
    X(m+1) = X(m+1) + x(n+1) * exp(-j*2*pi*m*n/N);
    end
   end
   % ------------------------------------------------------------
   % Check calculated values against MATLAB's FFT function
   X_FFT = fft(x);
   for m = 0:N-1
    DFT_Error(m+1) = abs(X(m+1) - X_FFT(m+1));
   end
   max_error = max(DFT_Error)
   fprintf('Calculated values of DFT\n')
   for m = 0:N-1
    fprintf(' X(%d) =%f +j %f \n',m+1,real(X(m+1)), imag(X(m+1)));
    fprintf(' X_FFT(%d)=%f +j %f \n\n',m+1,real(X_FFT(m+1)),imag(X_
FFT(m+1)));
   end
    % Check using MATLAB's built in FFT() function
   a0_calc = X_FFT(1)/N ;
   a1_calc = real(X_FFT(2)*(2/N)) ;
   b1_calc = -imag(X_FFT(2)*(2/N)) ;
   a2_calc = real(X_FFT(3)*(2/N)) ;
   b2_calc = -imag(X_FFT(3)*(2/N)) ;
   a3_calc = real(X_FFT(4)*(2/N)) ;
   b3_calc = -imag(X_FFT(4)*(2/N)) ;
   fprintf(' a0_calc=%f \n',a0_calc )
   fprintf(' a1_calc=%f \n',a1_calc )
   fprintf(' b1_calc=%f \n',b1_calc )
   fprintf(' a2_calc=%f \n',a2_calc )
   fprintf(' b2_calc=%f \n',b2_calc )
   fprintf(' a3_calc=%f \n',a3_calc)
   fprintf(' b3_calc=%f \n',b3_calc )
   ---- Output of the program --------------------
```

```
max_error = 5.8524e-013

Calculated values of DFT
 X(1) = 1056.000000 + j 0.000000
 X_FFT(1) =1056.000000 + j 0.000000

 X(2) = 461.999979 + j -396.000081
 X_FFT(2) = 461.999979 + j -396.000081

 X(3) = 330.000000 + j -264.000050
 X_FFT(3) =330.000000 + j -264.000050

 X(4) = 198.000000 +j -132.000000
 X_FFT(4) = 198.000000 +j -132.000000

 X(5) =-0.000000 +j -0.000173
 X_FFT(5)=0.000000 +j -0.000173

 X(6) =0.000021 +j 0.000081
 X_FFT(6)=0.000021 +j 0.000081

 a0_calc = 88.000000
 a1_calc = 76.999996
 b1_calc = 66.000013
 a2_calc = 55.000000
 b2_calc = 44.000008
 a3_calc=33.000000
 b3_calc=22.000000
```

6.13 Sliding DFT Algorithm [35, 41]

Computing DFT is not an one-time affair. We have to continuously compute DFT as samples keep streaming-in. Fortunately, considerable amount of computational burden can be saved since DFT calculation during any window is related to DFT, already computed for the previous window. This becomes possible because, in the two consecutive windows, we are operating on two sets of samples which are almost same, except for the newest sample that we include and the oldest sample that we discard. We will modify the expression for DFT to account for the fact that we have to implement the DFT in a continuous fashion over contagious windows. The modified expression for DFT during 'k^{th}' window is as follows:

$$X_k(m) = \sum_{n=k}^{n=N+k-1} x(n) e^{\frac{-j2\pi(n-k)m}{N}}$$

where '*k*' is the window number and '*m*' is the order of the DFT; $m = 0 \rightarrow$ DC, $m = 1 \rightarrow$ fundamental etc.

Let us run through a few values of the indices to verify that the basic DFT expression remains same.

Let $N = 8$

window '*k*'	$n = k$ to $(N + k-1)$	$(n-k)$
0	$0 \rightarrow 7$	$0 \rightarrow 7$
1	$1 \rightarrow 8$	$0 \rightarrow 7$
2	$2 \rightarrow 9$	$0 \rightarrow 7$
3	$3 \rightarrow 10$	$0 \rightarrow 7$
4	$4 \rightarrow 11$	$0 \rightarrow 7$

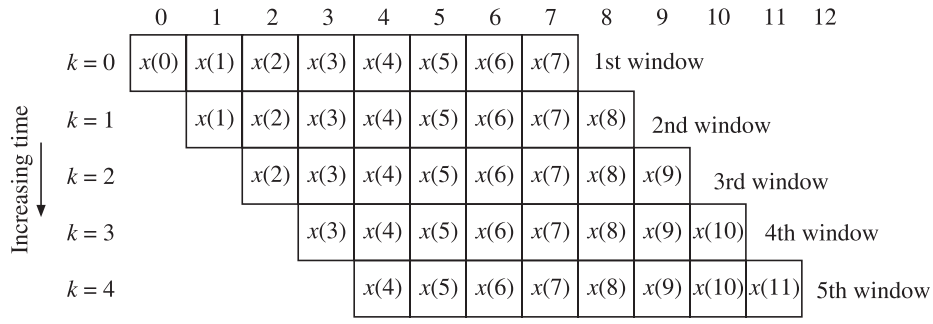


Figure 6.13 Sliding DFT.

DFT during the (k)th window can be written as:

$$X_k(m) = \sum_{n=k}^{n=N+k-1} x(n) e^{\frac{-j2\pi(n-k)m}{N}}$$

DFT for the $(k + 1)$th window can then, be written as:

$$X_{k+1}(m) = \sum_{n=k+1}^{n=N+k} x(n) e^{\frac{-j2\pi(n-k-1)m}{N}}$$

If we start the above summation from $n = k$, instead of $n = k + 1$, we will get an extra term:

$$x(k) e^{\frac{-j2\pi(-1)m}{N}} = x(k) e^{\frac{j2\pi m}{N}}$$

Thus, if we modify the lower limit of DFT for $(k + 1)$th window to $(n = k)$ then we will get the above extra term, hence we can keep the sum unaltered by subtracting $x(k) e^{\frac{j2\pi m}{N}}$ from

the sum with lower limit ($n = k$) instead of ($n = k + 1$). Hence, we can write it as:

$$X_{k+1}(m) = \left\{ \left(\sum_{n=k}^{n=N+k} x(n) e^{\frac{-j2\pi(n-k-1)m}{N}} \right) - x(k) e^{\frac{j2\pi m}{N}} \right\}$$

If we take out the last term of the above summation corresponding to $n = (N + k)$, we will have to reduce the upper index of the summation to $(N + k - 1)$ and the term corresponding to $n = (N + k)$ will be:

$$x(N+k) e^{\frac{-j2\pi(N+k-k-1)m}{N}} = x(N+k) e^{\frac{-j2\pi(N-1)m}{N}}$$

$$x(N+k) e^{\frac{-j2\pi(N-1)m}{N}} = x(N+k) e^{\frac{-j2\pi(N)m}{N}} e^{\frac{j2\pi m}{N}}$$

$$= x(N+k) e^{-j2\pi m} e^{\frac{j2\pi m}{N}}$$

$= x(N+k) e^{\frac{j2\pi m}{N}}$ since $e^{-j2\pi m}$ is equal to 1 for all values of m.

Hence, we can write:

$$X_{k+1}(m) = \left\{ \left(\sum_{n=k}^{n=N+k-1} x(n) e^{\frac{-j2\pi(n-k-1)m}{N}} \right) + x(N+k) e^{\frac{j2\pi m}{N}} - x(k) e^{\frac{j2\pi m}{N}} \right\}$$

$$X_{k+1}(m) = \left\{ \left(\sum_{n=k}^{n=N+k-1} x(n) e^{\frac{-j2\pi(n-k)m}{N}} e^{\frac{j2\pi m}{N}} \right) + x(N+k) e^{\frac{j2\pi m}{N}} - x(k) e^{\frac{j2\pi m}{N}} \right\}$$

$$X_{k+1}(m) = e^{\frac{j2\pi m}{N}} \left\{ \left(\sum_{n=k}^{n=N+k-1} x(n) e^{\frac{-j2\pi(n-k)m}{N}} \right) + x(N+k) - x(k) \right\}$$

$$X_{k+1}(m) = e^{\frac{j2\pi m}{N}} \left\{ X_k(m) + x(N+k) - x(k) \right\}$$

$$\text{DFT}(m)_{k+1} = e^{\frac{j2\pi m}{N}} \left\{ \text{DFT}(m)_k + \left[\text{Newest sample} - \text{Oldest sample} \right] \right\}$$

Consider the sliding DFT computation of an undistorted sine wave shown in Figure 6.14. In the first window, the samples from $x(0)$ to $x(7)$ will be used for computing $\text{DFT}(1)_1$. Next we slide the widow by one sample discarding $x(0)$ and including $x(8)$. Let us find how the $\text{DFT}(1)_2$ is related with $\text{DFT}(1)_1$.

$$\text{DFT}(1)_2 = e^{\frac{j2\pi 1}{8}} \left\{ \text{DFT}(1)_1 + [x(8) - x(0)] \right\}$$

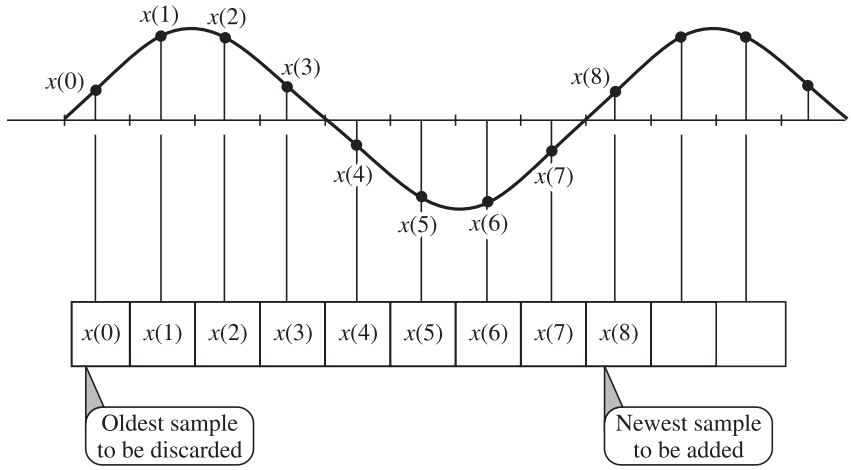


Figure 6.14 Sliding DFT computation.

However, in this case, since $x(8) = x(0)$, magnitude of DFT remains unaffected but there is an apparent phase shift of $\pi/4$ radians or 45° as the window slides along even if the signal is stationary. Thus:

$$\text{DFT}(1)_2 = e^{\frac{j\pi}{4}}\left\{\text{DFT}(1)_1\right\}$$

This is an unwanted artefact and provides motivation for modifying the DFT further as described in the next section.

6.13.1 Modified DFT

The recursive DFT expression shows that with every new computation of the sliding window, a phase shift of $j2\pi m/N$ is added to the phasor. Thus, there is a strong motivation for modifying the DFT to get rid of this apparent phase shift. The DFT can be modified as follows:

Computation of the 'mth' order DFT during the $(k + 1)$th window is given by:

$$X_{k+1}(m) = e^{\frac{j2\pi m}{N}}\left\{X_k(m) + x(N+k) - x(k)\right\}$$

Consider the computation of the fundamental, i.e., $m = 1$:

$$X_{k+1}(1) = e^{\frac{j2\pi}{N}}\left\{X_k(1) + x(N+k) - x(k)\right\}$$

It can be seen that DFT during $(k + 1)^{\text{th}}$ window accumulates a phase shift of $e^{\frac{j2\pi(k+1)}{N}}$

We can get rid of this phase shift of $e^{\frac{j2\pi(k+1)}{N}}$ in the computation of the $(k + 1)$th window by multiplying with $e^{\frac{-j2\pi(k+1)}{N}}$. Hence, we define a modified DFT $\widehat{X_{k+1}}(1)$ as:

$$\widehat{X_{k+1}}(1) = X_{k+1}(1)e^{\frac{-j2\pi(k+1)}{N}}$$

Multiplying both sides of equation for DFT by $e^{\frac{-j2\pi(k+1)}{N}}$, we get:

$$X_{k+1}(1)e^{\frac{-j2\pi(k+1)}{N}} = e^{\frac{-j2\pi(k+1)}{N}} e^{\frac{j2\pi}{N}} \left\{X_k(1) + x(N+k) - x(k)\right\}$$

$$\widehat{X_{k+1}}(1) = e^{\frac{-j2\pi(k)}{N}} \left\{X_k(1) + x(N+k) - x(k)\right\}$$

$$\widehat{X_{k+1}}(1) = \left\{ X_k(1)e^{\frac{-j2\pi(k)}{N}} + [x(N+k) - x(k)]e^{\frac{-j2\pi(k)}{N}} \right\}$$

Using the definition for the modified DFT, it can be seen that:

$$X_k(1)e^{\frac{-j2\pi(k)}{N}} = \widehat{X_k}(1)$$

Hence, we can write the modified recursive sliding DFT as:

$$\widehat{X_{k+1}}(1) = \left\{ \widehat{X_k}(1) + \left[x(N+k) - x(k)\right]e^{\frac{-j2\pi(k)}{N}} \right\}.$$

6.13.2 MATLAB Program for Sliding Modified Recursive DFT Algorithm

```
% Sliding Modified Recursive DFT Algorithm
% First 12 samples are generated from 200*sin(2*pi*50*t+pi/4)
% Next 12 samples are generated from 1000*sin(2*pi*50*t)
% sampling is done at 12 samples/cycle of 50 Hz i.e., at
600 Hz
% Sliding Modified Recursive DFT is applied to track the
signal magnitude

clear , clc
fsig = 50 ; T=1/fsig;
N = 12 ;
fs = N * fsig ;
dt = 1/fs ;
% ------------------------------------------------------------
%---------- Lets generate samples of signl ----------------
Im1 = 200 ; Im2 = 1000;
phi1 = 45*(pi/180) ; phi2 = 0 ;
n = 0 ;
for t=0:dt:(T-dt)
  n = n+1;
```

```
   x(n)=Im1*sin(2*pi*fsig*t+phi1);
   end

   fprintf('\nAt the end of first loop n = %f \n', n )

   fprintf('First Twelve Samples\n')
   disp(x(1:12))
   % first 12 samples are from 50Hz signal with peak value of 200
   for t = T:dt:(2*T-dt)
     n = n+1 ;
     x(n)= Im2*sin(2*pi*fsig*t+phi2);
   end

   fprintf('Next 12 Samples\n')
   disp(x(13:24))
   % Next 12 samples are from 50 Hz signal with peak value
of 1000
   %-----------------------------------------------------------
   %---------------- DFT calculation ----------------------------
   % Initialization of coeffs. to zero before start of
calculations
   for m = 0:1 % m is harmoic number 0=dc , 1 = fundamental
     X(m+1)= 0 ;
   end

   %----- Plotting of samples of signal-------------------------
     subplot(2,1,1)
     stem(x)
     xmin1=0;xmax1=25;ymin1=-1000;ymax1=1001;
     axis([xmin1 xmax1 ymin1 ymax1])
     grid on
     xlabel('Sample Number -->')
     ylabel('Sample value')
     title('Composite Signal ( discontinuity at sample no.13)')
   % ------------------------------------------------------------
   %------------- Actual Caluculation of DFT -----------------
   m = 1 % Only fundamental
     for n=0:N-1
     X(m+1)= X(m+1)+ ( x(n+1)*exp(-j*2*pi*m*n/N));
     end
   % ---------------------------------------------------
     fprintf('Calculated value of DFT\n')
   % Let us represent the sliding windowed DFT with X_hat(window
no. = k)
```

```
% Note X(1) is DC component and X(2) is fundamental
X_hat(1)=X(2);
  for k=1:12
  Newest = x(N+k);
  Oldest = x(k) ;
 X_hat(k+1)= X_hat(k)+( (Newest - Oldest)*(exp(-j*2*pi*k/N)));
  end

X_hat_mag =(2/N)*abs(X_hat)
% ----- Plotting of Sliding DFT -----
subplot(2,1,2)
stem(X_hat_mag)
xmin1=0;xmax1=13;ymin1=0;ymax1=1001;
axis([xmin1 xmax1 ymin1 ymax1])
grid on
xlabel('Sliding Window Number --->')
ylabel('Sliding DFT Magnitude')
title('Sliding DFT')
% ------------------ End of program ------------------------
```

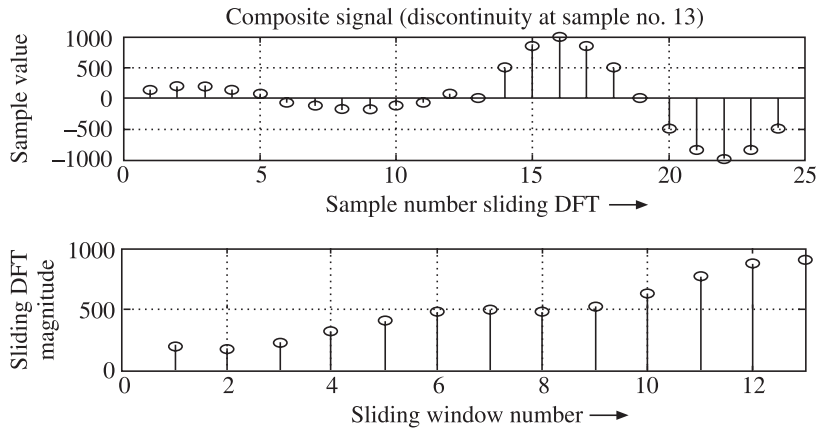


Figure 6.15 Sliding modified recursive DFT algorithm in MATLAB.

Review Questions

1. What are the conditions to be fulfilled by a signal for being represented by Fourier series?
2. What are the Dirichlet conditions?
3. State Fourier series in trigonometric form without phase angle.

4. State Fourier series in trigonometric form as a cosine wave with phase angle.
5. State Fourier series in trigonometric form as a sine wave with phase angle.
6. State the relationship between coefficients of Fourier series in Q.4 with Fourier series in Q.3.
7. State the relationship between coefficients of Fourier series in Q.5 with Fourier series in Q.3.
8. Derive expression for Fourier series in complex exponential form.
9. State the relationship between coefficients of Fourier series in Q.8 with Fourier series in Q.3.
10. Prove that the a_n and b_n coefficients of the Fourier series are optimal in LSQ sense.
11. Compare the expressions for CFT and DFT and point out similarities between them.
12. If the DFT has 'N' frequency terms, what is the highest harmonic frequency in terms of multiples of fundamental?
13. In a protective relay hardware, 8 samples at a sampling frequency of 1000 Hz are collected in the DFT window. What is the frequency of the fundamental, 2nd, 3rd, 4th harmonic etc.?
14. All samples inside the DFT window have same magnitude. What frequency content is indicated?
15. Only one sample is inside the DFT window, at the centre is non-zero, all other samples are zero. What frequency content is indicated?
16. If there are N samples in time domain, how many samples will be there in the frequency domain?
17. What is the effect of increasing number of samples in a given window on the precision of DFT?
18. What is special about $X(0)$?
19. What is special about $X(N/2)$; assuming N is even.
20. Explain the statement: all 'N' DFT coefficients are not unique.
21. If $x(n)$ are all real, is it guaranteed that $X(n)$ will all be real?
22. Why conventional Fourier series theory fails when we try to apply it to non-recurring waveform?

7

FFT and Goertzel Algorithm

7.1 What is the Motivation for FFT?

The expression for DFT is normally written in the form shown below:

$$X(m)_{m=0}^{m=N-1} = \sum_{n=0}^{n=N-1} x(n)e^{-j2\pi mn/N}$$

Let us represent $e^{-j2\pi/N} = W_N$ (Note that negative sign is included in W_N) and rewrite expression for DFT as shown below:

$$X(m)_{m=0}^{m=N-1} = \sum_{n=0}^{n=N-1} x(n)W_N^{mn}$$

For $N = 8$, we can write:

$$X(m)_{m=0}^{m=7} = \sum_{n=0}^{n=7} x(n)W_8^{mn}$$

This is actually a matrix equation since we have a $N \times 1$ column vector on LHS and product of a $(N \times N)$ matrix with a $(N \times 1)$ column vector on the RHS. Hence, we can write the equation as:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} W_8^{0\times0} & W_8^{0\times1} & W_8^{0\times2} & W_8^{0\times3} & W_8^{0\times4} & W_8^{0\times5} & W_8^{0\times6} & W_8^{0\times7} \\ W_8^{1\times0} & W_8^{1\times1} & W_8^{1\times2} & W_8^{1\times3} & W_8^{1\times4} & W_8^{1\times5} & W_8^{1\times6} & W_8^{1\times7} \\ W_8^{2\times0} & W_8^{2\times1} & W_8^{2\times2} & W_8^{2\times3} & W_8^{2\times4} & W_8^{2\times5} & W_8^{2\times6} & W_8^{2\times7} \\ W_8^{3\times0} & W_8^{3\times1} & W_8^{3\times2} & W_8^{3\times3} & W_8^{3\times4} & W_8^{3\times5} & W_8^{3\times6} & W_8^{3\times7} \\ W_8^{4\times0} & W_8^{4\times1} & W_8^{4\times2} & W_8^{4\times3} & W_8^{4\times4} & W_8^{4\times5} & W_8^{4\times6} & W_8^{4\times7} \\ W_8^{5\times0} & W_8^{5\times1} & W_8^{5\times2} & W_8^{5\times3} & W_8^{5\times4} & W_8^{5\times5} & W_8^{5\times6} & W_8^{5\times7} \\ W_8^{6\times0} & W_8^{6\times1} & W_8^{6\times2} & W_8^{6\times3} & W_8^{6\times4} & W_8^{6\times5} & W_8^{6\times6} & W_8^{6\times7} \\ W_8^{7\times0} & W_8^{7\times1} & W_8^{7\times2} & W_8^{7\times3} & W_8^{7\times4} & W_8^{7\times5} & W_8^{7\times6} & W_8^{7\times7} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix}$$

Thus, we can see that an 8-point DFT involves $8 \times 8 = 64$ multiplications. In general an N-point DFT involves N^2 multiplications. As N goes on increasing, N^2 increases in an exponential manner. Since multiplication takes time to perform on the DSP microprocessor, the computation time of DFT sharply increases as number of sample points increase. Therefore, it is highly profitable to somehow cut-down the number of multiplications. The following observation provided a solution to the problem of cutting down the number of multiplications. An N-point DFT requires N^2 multiplications and an $(N/2)$ point DFT requires $(N^2/4)$ multiplications. If, somehow, we could breakup an N-point DFT as a simple function of two $(N/2)$ point DFTs then the number of multiplications required would be $(N^2/4) + (N^2/4) = 2(N^2/4) = N^2/2$ multiplications instead of N^2 multiplications. Thus, we would be able to reduce the number of multiplications from N^2 to $N^2/2$, i.e., multiplications are reduced to half. Now, we can apply the same argument and break down $N/2$ point DFT in terms of two $N/4$ point DFTs and get further reduction in number of multiplications. Let us apply this divide and conquer policy to an 8-point DFT, as shown below:

- 8-point DFT direct implementation $\rightarrow 8^2 \rightarrow 64$ multiplications
- 8-point DFT $\rightarrow$ [4-point DFT] + [4-point DFT] $\rightarrow 16 + 16 = 32$ multiplications
- 8-point DFT $\rightarrow$ [2-point] + [2-point] DFT + [2-point] + [2-point] DFT $\rightarrow 4 + 4 + 4 + 4 = 16$ multiplications

Thus, by breaking down the DFT into smaller and smaller DFTs, we could reduce the number of time-consuming multiplication operations from 64 to 16. 'N' is very large, typically 1024, in case of image processing applications and saving becomes substantial. We did not count the number of additions involved. It can be shown that the number of additions required also goes down substantially if we apply the approach discussed above. This process of breaking down the DFT into smaller and smaller DFTs is called *decimation*. If we do decimation-in-time domain, the DFT algorithm is called *decimation-in-time FFT*. The alternative approach is decimation-in-frequency. Thus, FFT is nothing but a method of implementing DFT so as to speed it up. FFT is not different from DFT as far as the input and output are concerned as shown in Table 7.1.

Table 7.1 DFT and FFT

	Multiplications	**Additions**
DFT	N^2	$N(N-1)$
FFT	$N/2 \log_2 N$	$N \log_2 N$

Table 7.2 shows the substantial saving in computational effort offered by FFT assuming $N = 2^{13} = 8192$.

Table 7.2 Substantial saving in computational effort offered by FFT

$N = 8192$	**Multiplications**	**Additions**
DFT	$N^2 = 67108864$	$N(N-1) = 67100672$
FFT	$N/2 \log_2 N = 53248$	$N \log_2 N = 106496$
Saving	$\frac{\text{DFT multiplications}}{\text{FFT multiplications}} \cong 1260$	$\frac{\text{DFT additions}}{\text{FFT additions}} \cong 630$

The FFT algorithm was invented by J.W. Cooley and J.W. Tuckey in 1965. In those days, digital computer capabilities were rather limited in terms of processing speed and memory storage etc. Hence, FFT algorithm had a dramatic impact upon the scientific community. In fact, it started a revolution in signal processing. Many applications where one would not think of applying FFT (because of the high cost) could now benefit from FFT. Eventually, all other methods of doing Fourier analysis were superseded by digital computer based FFT. It is interesting to note that like most other mathematical inventions even FFT can be traced back to Carl Fredrich Gauss, the 18th century mathematical genius.

Consider an 8-point DFT. We have taken out the relevant portion of the matrix equation for extraction of fundamental. Another way to look at it is as a dot product or inner product of two vectors. One vector consists of the 8 weights and the other vector is composed of the 8 samples in time domain. Weights for computation of fundamental can be visualised as shown in Figure 7.1.

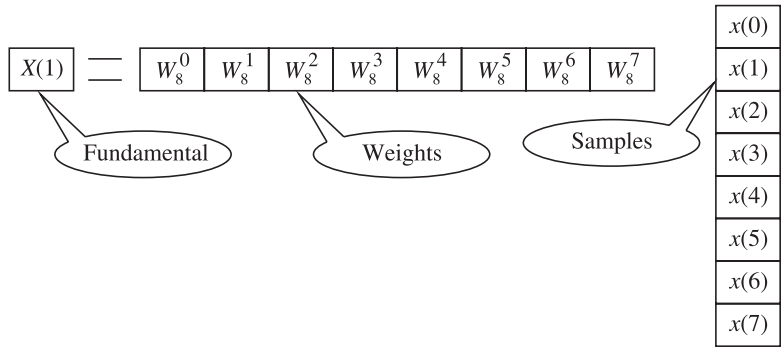


Figure 7.1 Visualisation of weights for computation of fundamental in an 8-point DFT (*contd.*).

Visualisation of weights for 8-point DFT

$$W_N = e^{\frac{-j2\pi}{N}}; N = 8$$

$$W_N = e^{\frac{-j2\pi}{8}} = e^{\frac{-j\pi}{4}} = e^{-j45^\circ}$$

Note: Positive angles are measured in anti-clockwise direction and negative angles, therefore are measured in clockwise direction.

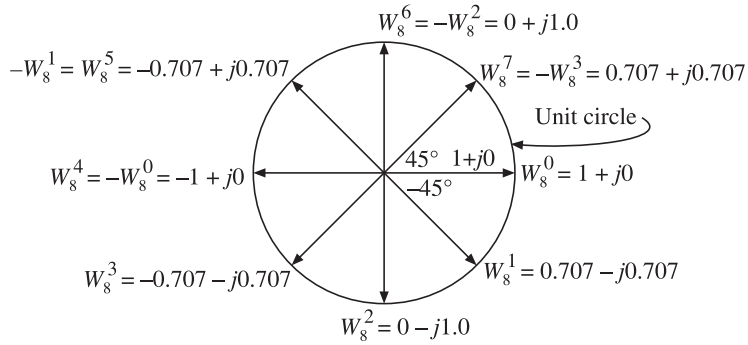


Figure 7.1 Visualisation of weights for computation of fundamental in an 8-point DFT.

7.2 FFT by Decimation-in-Time (DIT) [5, 13, 19, 22]

The idea of decomposing a bigger DFT into smaller ones is called *decimation*. It is shown conceptually in Figure 7.2 for an 8-point DFT. Note that a 1-point DFT of a signal is the signal itself, so it is a trivial operation, hence most of the books stop at the 2-point DFT.

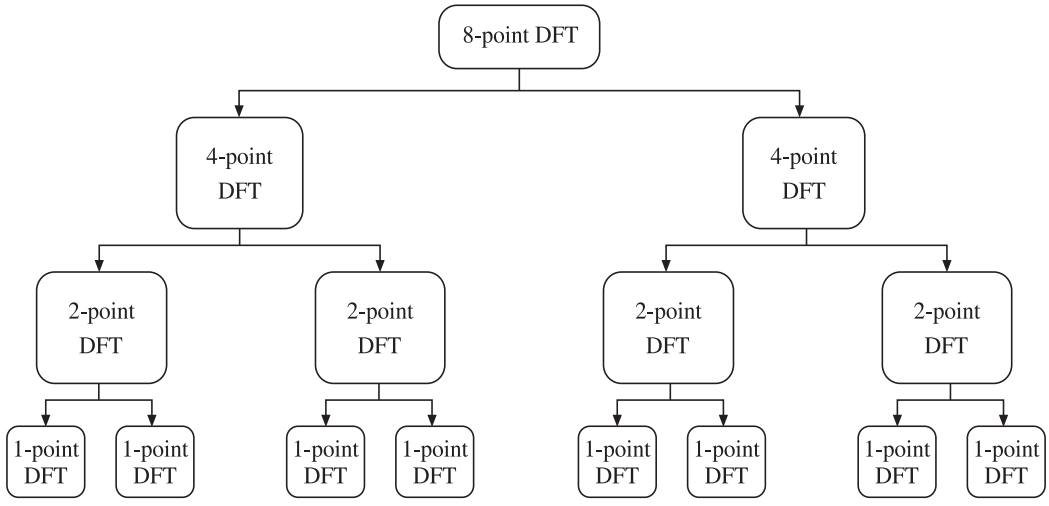


Figure 7.2 Decimation of an 8-point DFT.

In the decimation-in-time approach, we express the *N*-point DFT as a combination of *N*/2 point DFT of even indexed samples and another *N*/2 point DFT of odd indexed samples. Note that for *N*/2 to be an integer *N* must be even. Further it helps if *N* is a power of 2. Mixing of even indexed and odd indexed samples in this fashion is perfectly valid since in a summation the order in which terms are summed does not matter. The idea is shown conceptually in Figure 7.3

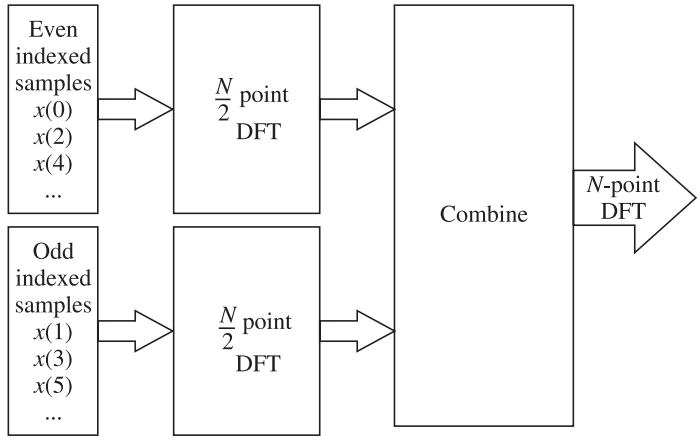


Figure 7.3 Combining two *N*/2 point DFTs.

To develop the decimation-in-time FFT algorithm, we start with the expression for DFT:

$$X(m)_{m=0}^{m=N-1} = \sum_{n=0}^{n=N-1} x(n)e^{-j2\pi mn/N} = \sum_{n=0}^{n=N-1} x(n)W_N^{mn}$$

Decomposing the sum as that of even indexed and odd indexed time domain samples:

$$X(m)_{m=0}^{m=N-1} = \sum_{n=0}^{n=\frac{N}{2}-1} x(2n)e^{-j2\pi m(2n)/N} + \sum_{n=0}^{n=\frac{N}{2}-1} x(2n+1)e^{-j2\pi m(2n+1)/N}$$

$$X(m)_{m=0}^{m=N-1} = \sum_{n=0}^{n=\frac{N}{2}-1} x(2n)e^{-j2\pi m(2n)/N} + e^{-j(2\pi m)/N} \sum_{n=0}^{n=\frac{N}{2}-1} x(2n+1)e^{-j2\pi m(2n)/N}$$

Consider the first part of the sum. By comparing it with standard DFT expression, we can conclude that, the expression represents an *N*/2 point DFT of the even indexed time domain samples.

$$\sum_{n=0}^{n=\frac{N}{2}-1} x(2n)e^{-j2\pi m\frac{(2n)}{N}} \Leftarrow \text{This is the } \frac{N}{2} \text{ point DFT of even indexed samples}$$

Consider the summation in the second part of the expression for DFT. Again it can be seen that it is another *N*/2 point DFT, but this time, of the odd indexed time domain samples.

$$\sum_{n=0}^{n=\frac{N}{2}-1} x(2n+1)e^{-j2\pi m(2n)/N} \Leftarrow \text{This is the } \frac{N}{2} \text{ point DFT of odd indexed samples}$$

Recall the definition of W_N; $W_N = e^{-j2\pi/N}$. Hence the remaining term, i.e., $e^{-j(2\pi m)/N}$ can be written as W_N^m. Thus, what we have done is the decomposition of the N-point DFT as:

[mth term of N-point DFT] = [$N/2$ point DFT(even indexed samples)]+ W_N^m [$N/2$ point. DFT(odd indexed samples)]

Thus, N-point DFT is not simply a sum of two $N/2$ point DFTs, we have to use a weighting factor of W_N^m before combining the two DFTs. Using more compact notation we can write:

$$e^{-j2\pi m(2n)/N} = e^{-j2\pi mn/\left(\frac{N}{2}\right)} = W_{N/2}^{mn}$$

and hence the expression for DFT becomes:

$$X(m)_{m=0}^{m=N-1} = \sum_{n=0}^{n=\frac{N}{2}-1} x(2n)W_{N/2}^{mn} + W_N^m \sum_{n=0}^{n=\frac{N}{2}-1} x(2n+1)W_{N/2}^{mn}$$

Further saving in labour is possible by noting the relationship between:

$$X(m); m = 0 \text{ to } \frac{N}{2}-1 \Rightarrow X(0) \text{ to } X\left(\frac{N}{2}-1\right) \text{and}$$

$$X\left(m+\frac{N}{2}\right); m = 0 \text{ to } \frac{N}{2}-1 \Rightarrow X\left(\frac{N}{2}\right) \text{to } X(N-1)$$

$$X\left(m+\frac{N}{2}\right)_{m=0}^{m=\frac{N}{2}-1} = \sum_{n=0}^{n=\frac{N}{2}-1} x(2n)W_{N/2}^{\left(m+\frac{N}{2}\right)n} + W_N^{\left(m+\frac{N}{2}\right)} \sum_{n=0}^{n=\frac{N}{2}-1} x(2n+1)W_{N/2}^{\left(m+\frac{N}{2}\right)n}$$

Consider $W_{N/2}^{\left(m+\frac{N}{2}\right)n}$

$$W_{N/2}^{\left(m+\frac{N}{2}\right)n} = W_{N/2}^{(mn)} W_{N/2}^{\left(\frac{N}{2}\right)n} = W_{N/2}^{(mn)} e^{\frac{-j2\pi nN/2}{N/2}}$$

$$e^{\frac{-j2\pi nN/2}{N/2}} = e^{-j2\pi n} = 1 \text{ (always)}$$

Hence,

$$W_{N/2}^{\left(m+\frac{N}{2}\right)n} = W_{N/2}^{(mn)}$$

Similarly, consider $W_N^{\left(m+\frac{N}{2}\right)}$

$$W_N^{\left(m+\frac{N}{2}\right)} = W_N^{(m)} W_N^{\left(\frac{N}{2}\right)}$$

Consider $W_N^{\left(\frac{N}{2}\right)}$

$$W_N^{\left(\frac{N}{2}\right)} = e^{\left(\frac{-j2\pi N/2}{N}\right)} = e^{-j\pi} = -1$$

Hence, $$W_N^{\left(m+\frac{N}{2}\right)} = -W_N^{(m)}$$

i.e., weights for the second half of DFT terms are negative of those for the first half. Thus, we can write:

$$X\left(m+\frac{N}{2}\right)_{m=0}^{m=\frac{N}{2}-1} = \sum_{n=0}^{n=\frac{N}{2}-1} x(2n) W_{N/2}^{(m)n} - W_N^{(m)} \sum_{n=0}^{n=\frac{N}{2}-1} x(2n+1) W_{N/2}^{(m)n}$$

We can rewrite the above expression as:

$$X(m)_{m=(\frac{N}{2})}^{m=(N-1)} = \sum_{n=0}^{n=\frac{N}{2}-1} x(2n) W_{N/2}^{(m)n} - W_N^{(m)} \sum_{n=0}^{n=\frac{N}{2}-1} x(2n+1) W_{N/2}^{(m)n}$$

Collecting the expressions for the DFT together:

$$X(m)_{m=0}^{m=\left(\frac{N}{2}\right)-1} = \sum_{n=0}^{n=\frac{N}{2}-1} x(2n) W_{N/2}^{mn} + W_N^m \sum_{n=0}^{n=\frac{N}{2}-1} x(2n+1) W_{N/2}^{mn}$$

$$X(m)_{m=\left(\frac{N}{2}\right)}^{m=(N-1)} = \sum_{n=0}^{n=\frac{N}{2}-1} x(2n) W_{N/2}^{mn} - W_N^{(m)} \sum_{n=0}^{n=\frac{N}{2}-1} x(2n+1) W_{N/2}^{mn}$$

If we let $$\sum_{n=0}^{n=\frac{N}{2}-1} x(2n) W_{N/2}^{mn} = A_m = \text{DFT of all even indexed samples}$$

and $$\sum_{n=0}^{n=\frac{N}{2}-1} x(2n+1) W_{N/2}^{mn} = B_m = \text{DFT of all odd indexed samples}$$

then we can write $$X(m)_{m=0}^{m=\left(\frac{N}{2}\right)-1} = A(m) + W_N^m\, B(m)$$

and $$X(m)_{m=N/2}^{m=N-1} = A(m) - W_N^m\, B(m)$$

Hence for an 8-point DFT, we can write the following:

$$X(0) = A(0) + W_8^0\, B(0)$$
$$X(1) = A(1) + W_8^1\, B(1)$$
$$X(2) = A(2) + W_8^2\, B(2)$$
$$X(3) = A(3) + W_8^3\, B(3)$$

and the second half of the terms as:

$$m = 0;\ X\left(m + \frac{8}{2}\right) = X(0 + 4) = X(4) = A(0) - W_8^0\, B(0)$$

$$m = 1;\ X\left(m + \frac{8}{2}\right) = X(1 + 4) = X(5) = A(1) - W_8^1\, B(1)$$

$$m = 2;\ X\left(m + \frac{8}{2}\right) = X(2 + 4) = X(6) = A(2) - W_8^2\, B(2)$$

$$m = 3;\ X\left(m + \frac{8}{2}\right) = X(3 + 4) = X(7) = A(3) - W_8^3\, B(3)$$

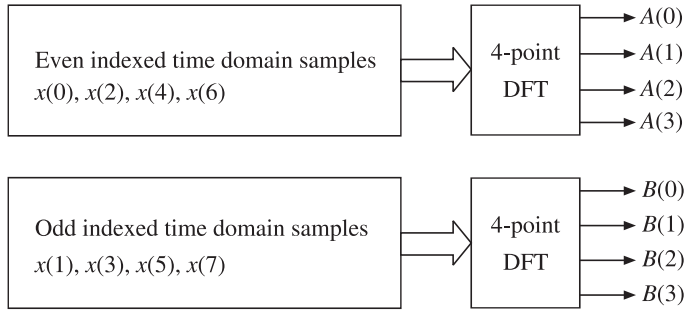


Figure 7.4 Development of FFT (decimation-in-time).

We take a look at the whole process in a step by step manner in Figures 7.5 and 7.6.

We can continue to follow the same mathematical logic to break down the 4-point DFT into a combination of two 2-point DFTs and each 2-point DFT as a combination of two 1-point DFTs. However, instead, we illustrate graphically, in Figures 7.7 and 7.8 how a 4-point DFT is synthesised from two 2-point DFTs. Figure 7.9 shows the synthesis of a 2-point DFT using two 1-point DFTs. Figure 7.10 depicts the concept of a 'butterfly'. Finally Figure 7.11 combines all these steps into a full-fledged flow-chart of an 8-point FFT with decimation-in-time.

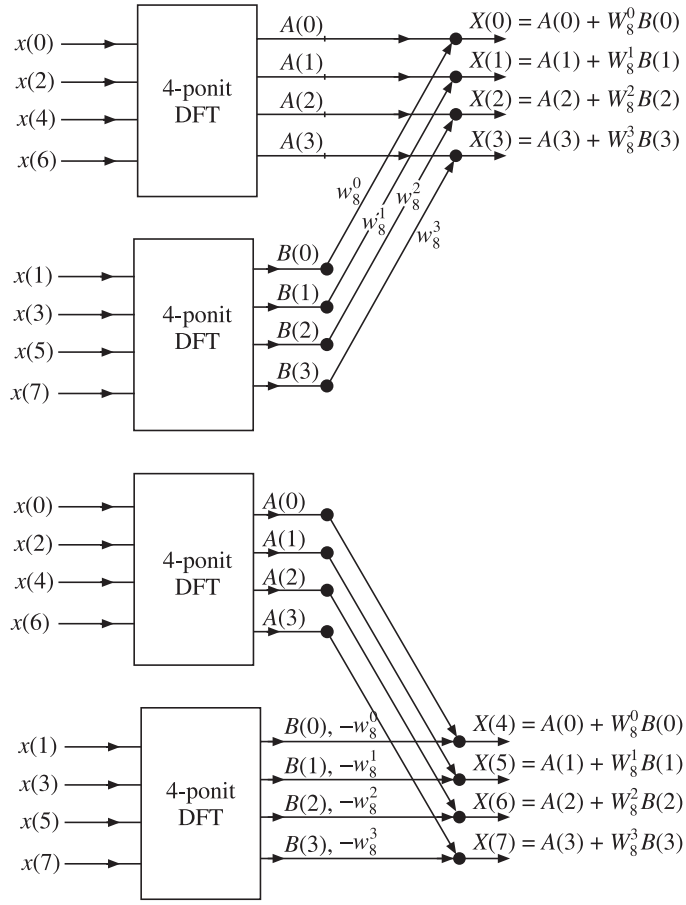


Figure 7.5 Development of FFT(decimation-in-time) in a step-by-step manner.

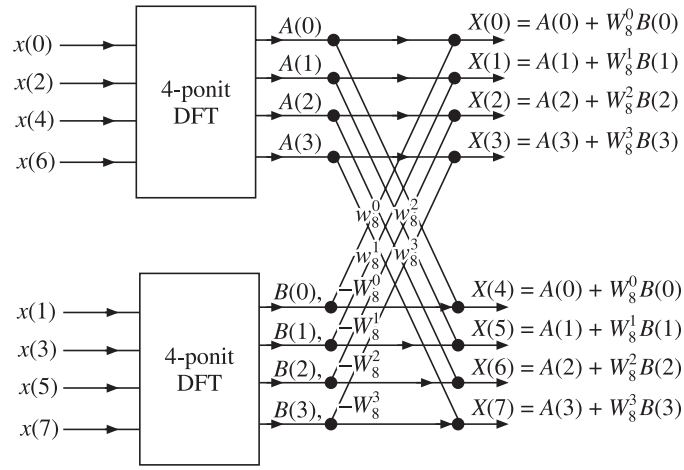


Figure 7.6 Development of FFT (decimation-in-time) continued.

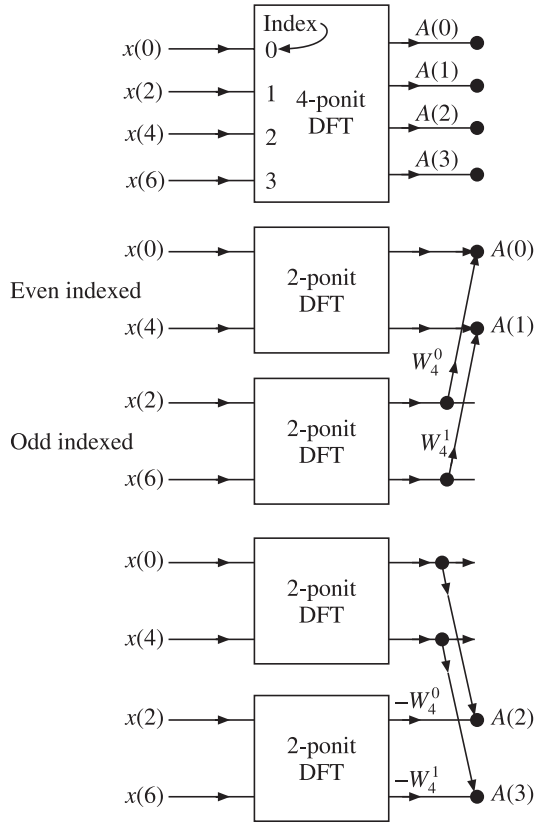


Figure 7.7 Development of FFT (decimation-in-time) continued.

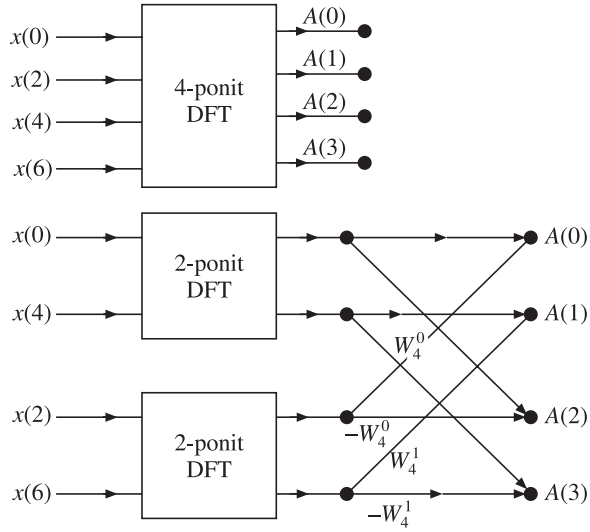


Figure 7.8 Development of FFT (decimation-in-time) continued.

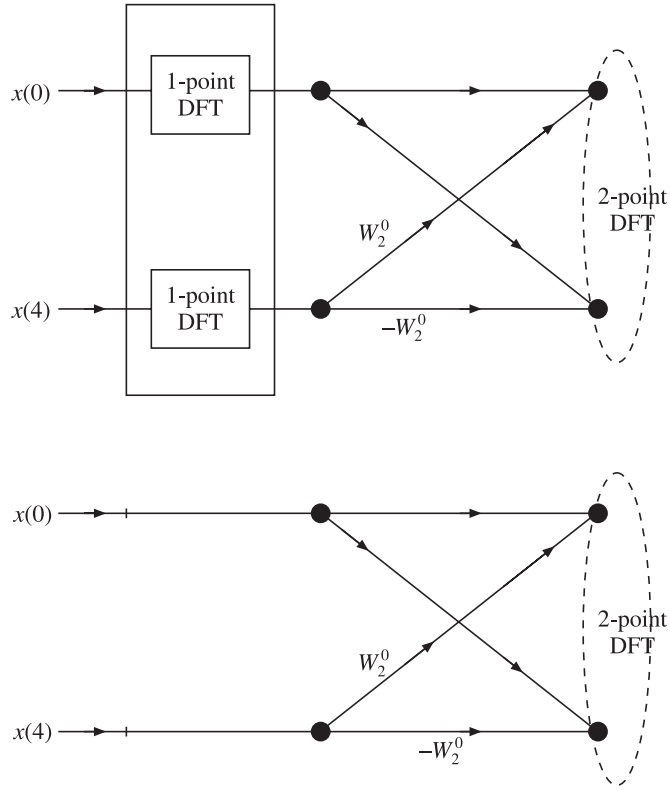


Figure 7.9 Development of FFT (decimation-in-time) continued.

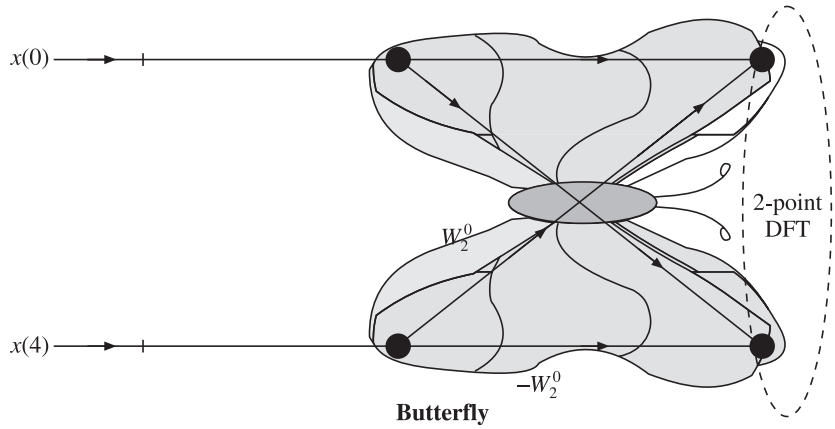


Figure 7.10 Concept of 'butterfly'.

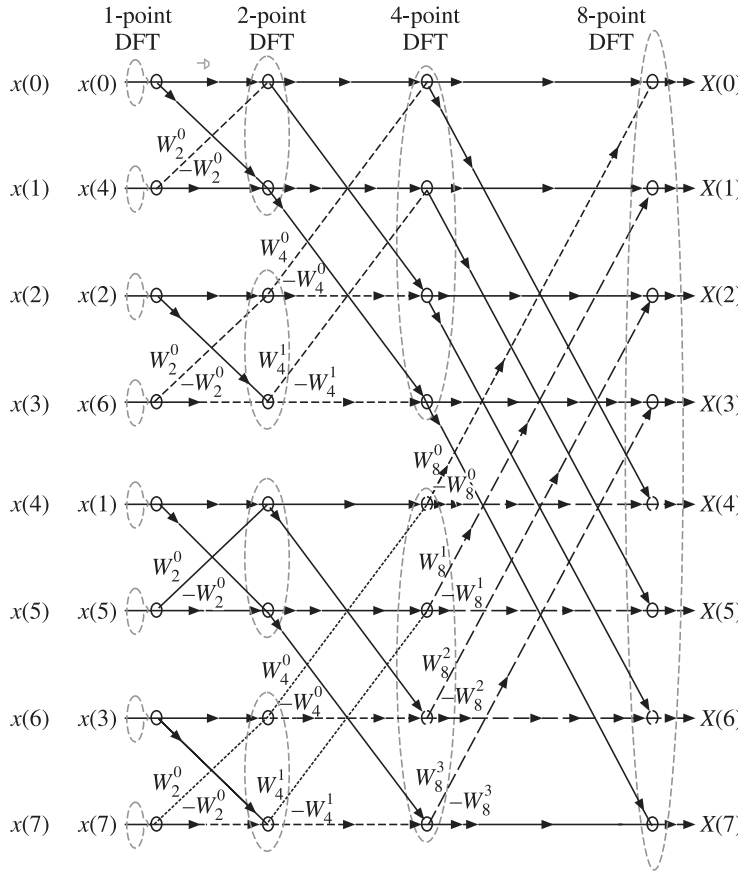


Figure 7.11 The FFT (decimation-in-time) signal flow graph.

It is to be noted that the original sequence of the indices which was in natural progression, i.e.,

$$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$$

has become scrambled. It is now, $0 \rightarrow 4 \rightarrow 2 \rightarrow 6 \rightarrow 1 \rightarrow 5 \rightarrow 3 \rightarrow 7$.

Table 7.3 shows the binary representation of the indices.

Table 7.3 Binary representation of indices

Decimal	0	1	2	3	4	5	6	7
Binary	000	001	010	011	100	101	110	111
Bit reversed	000	100	010	110	001	101	011	111
Decimal	0	4	2	6	1	5	3	7

Thus, it is seen that the sequence of indices for the FFT can be easily generated by '*bit reversal*' of the naturally ordered indices. To take advantage of this property, DSP

microprocessors include facilities for 'bit reverse' addressing. Note that bit-reversal is not the same as 'bit complementation'.

7.2.1 Numerical Problem on FFT (DIT)

In order to appreciate the simplicity and beauty of FFT, it is worthwhile to solve a numerical problem by hand calculation.

EXAMPLE 7.1 Find the 8-point FFT(DIT) of the following signal samples collected in a window of 8-samples.

$x(0) = 650.0000$ $x(1) = 391.4214$ $x(2) = -350.0000$ $x(3) = 274.2641$
$x(4) = -150.0000$ $x(5) = 108.5786$ $x(6) = 50.0000$ $x(7) = -574.2641$

Solution We can solve the problem in a 'visual manner' by writing the calculations on the signal flow graph of the FFT(DIT) as shown in Figure 7.12.

The answers can be seen to be:

$X(0) = 400.0000$ $X(1) = 400 - j399.8792$
$X(2) = 800.0000 - j800.0000$ $X(3) = 1199.9390 - j1199.8790$
$X(4) = 0.0$ $X(5) = 1199 + j1199.879$
$X(6) = 800.0000 + j800.0000$ $X(7) = 400.0000 + j399.8792$

Thus, it can be seen that $X(1) = X(7)^*$, $X(2) = X(6)^*$ and $X(3) = X(5)^*$ while there are no conjugate entries for $X(0)$ and $X(4)$.

7.3 Fast Fourier Transform by Decimation-in-Frequency

To develop the decimation-in-frequency FFT algorithm, we again start with the expression for DFT:

$$X(m)_{m=0}^{m=N-1} = \sum_{n=0}^{n=N-1} x(n)e^{-j2\pi mn/N} = \sum_{n=0}^{n=N-1} x(n)W_N^{mn}$$

Decomposing the sum as that of first-half and second-half of the time domain samples:

$$X(m)_{m=0}^{m=N-1} = \sum_{n=0}^{n=\frac{N}{2}-1} x(n)e^{-j2\pi m(n)/N} + \sum_{n=0}^{n=\frac{N}{2}-1} x\left(\frac{N}{2}+n\right)e^{-j2\pi m\left(\frac{N}{2}+n\right)/N}$$

$$e^{-j2\pi m(n)/N} = W_N^{mn}$$

$$e^{-j2\pi m\left(\frac{N}{2}+n\right)/N} = e^{-j2\pi mN/2N}\, e^{-j2\pi mn/N}$$

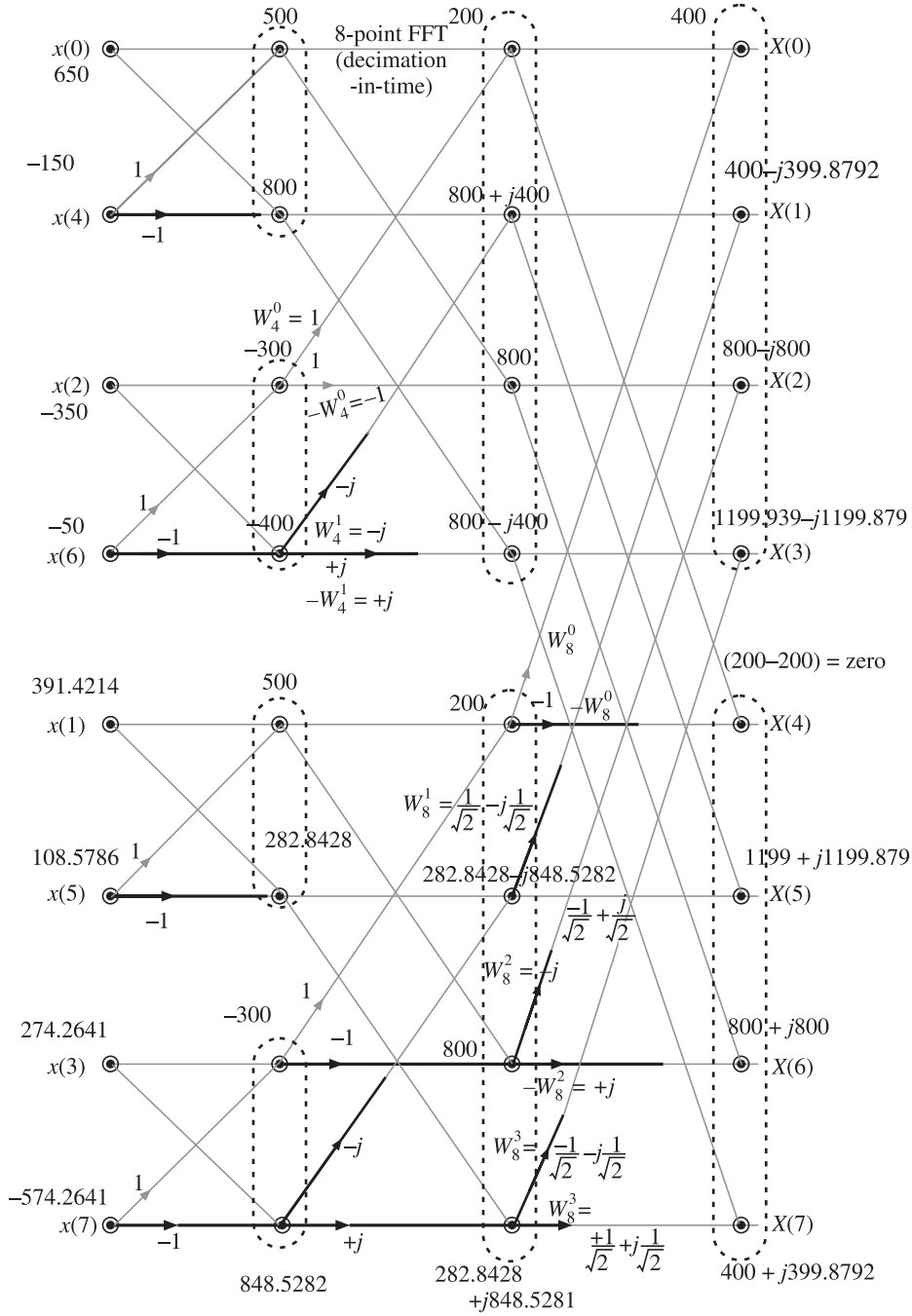


Figure 7.12 Numerical problem on FFT(DIT).

$e^{-j2\pi m\left(\frac{N}{2}+n\right)/N} = e^{-j2\pi m/2}\, e^{-j2\pi mn/N}$; in the first term the constant 2 in numerator is not cancelled with that in the denominator so that the term remains in standard form $W_N = e^{-j2\pi/N}$. Hence,

$$e^{-j2\pi m\left(\frac{N}{2}+n\right)/N} = W_2^m W_N^{mn}$$

$$X(m)_{m=0}^{m=N-1} = \sum_{n=0}^{n=\frac{N}{2}-1} x(n)W_N^{mn} + W_2^m \sum_{n=0}^{n=\frac{N}{2}-1} x\left(\frac{N}{2}+n\right)W_N^{mn}$$

In order to introduce further simplification by way of recursion, consider the even indexed terms of the DFT, i.e., $X(2m)$:

$$X(2m)_{m=0}^{m=N/2-1} = \sum_{n=0}^{n=\frac{N}{2}-1} x(n)W_N^{(2m)n} + W_2^{(2m)} \sum_{n=0}^{n=\frac{N}{2}-1} x\left(\frac{N}{2}+n\right)W_N^{(2m)n}$$

$$X(2m)_{m=0}^{m=N/2-1} = \sum_{n=0}^{n=\frac{N}{2}-1} x(n)W_{N/2}^{mn} + W_2^{(2m)} \sum_{n=0}^{n=\frac{N}{2}-1} x\left(\frac{N}{2}+n\right)W_{N/2}^{mn}$$

What is $W_2^{(2m)}$?

$$W_2^{(2m)} = e^{\frac{-j2\pi 2m}{2}} = e^{-j2\pi m} = 1$$

Hence, we get:

$$X(2m)_{m=0}^{m=N/2-1} = \sum_{n=0}^{n=\frac{N}{2}-1} x(n)W_{N/2}^{mn} + \sum_{n=0}^{n=\frac{N}{2}-1} x\left(\frac{N}{2}+n\right)W_{N/2}^{mn}$$

We can club the summation as:

$$X(2m)_{m=0}^{m=N/2-1} = \sum_{n=0}^{n=\frac{N}{2}-1} \left[x(n) + x\left(\frac{N}{2}+n\right)\right]W_{N/2}^{mn}$$

The above is an $N/2$ point DFT of sum of signals from the two halves, i.e., for an 8-point DFT

$$[x(0) + x(4)],\ [x(1) + x(5)],\ [x(2) + x(6)] \quad \text{and} \quad [x(3) + x(7)]$$

But remember that the above expression gives only even indexed terms of the DFT. We need to find expression for remaining half of the DFT terms, i.e., the odd indexed terms of the DFT.

Consider the expression for DFT derived earlier:

$$X(m)_{m=0}^{m=N-1} = \sum_{n=0}^{n=\frac{N}{2}-1} x(n)W_N^{mn} + W_2^m \sum_{n=0}^{n=\frac{N}{2}-1} x\left(\frac{N}{2}+n\right)W_N^{mn}$$

Hence, consider odd indexed terms of the DFT:

$$X(2m+1)_{m=0}^{m=N/2-1} = \sum_{n=0}^{n=\frac{N}{2}-1} x(n)W_N^{(2m+1)n} + W_2^{(2m+1)} \sum_{n=0}^{n=\frac{N}{2}-1} x(\frac{N}{2}+n)W_N^{(2m+1)n}$$

What is $W_N^{(2m+1)n}$?

$$W_N^{(2m+1)n} = W_N^{2mn}W_N^n = W_{N/2}^{mn}W_N^n$$

What is $W_2^{(2m+1)}$?

$$W_2^{(2m+1)} = W_2^{2m}W_2^1 = e^{\frac{-j2\pi 2m}{2}} e^{\frac{-j2\pi 1}{2}} = e^{-j2\pi m}e^{-j\pi} = 1\cdot(-1) = -1$$

Hence, we get:

$$X(2m+1)_{m=0}^{m=N/2-1} = \sum_{n=0}^{n=\frac{N}{2}-1} x(n)W_{N/2}^{mn}W_N^n - \sum_{n=0}^{n=\frac{N}{2}-1} x\left(\frac{N}{2}+n\right)W_{N/2}^{mn}W_N^n$$

Simplifying further we get:

$$X(2m+1)_{m=0}^{m=N/2-1} = \sum_{n=0}^{n=\frac{N}{2}-1} \left[\left[x(n) - x\left(\frac{N}{2}+n\right)\right]W_{N/2}^{mn}\right]W_N^n$$

Alternate way of writing the above expression could be:

$$X(2m+1)_{m=0}^{m=N/2-1} = \sum_{n=0}^{n=\frac{N}{2}-1} \left[\left[x(n)W_N^n - x\left(\frac{N}{2}+n\right)W_N^n\right]W_{N/2}^{mn}\right]$$

The above is a 'weighted' $N/2$ point DFT of difference of signals from the two halves, weighted by factor W_N^n, i.e., for an 8-point DFT:

$$[x(0) - x(4)],\ [x(1) - x(5)],\ [x(2) - x(6)] \quad \text{and} \quad [x(3) - x(7)]$$

Thus, on the input (time domain) side we have to club the first half of samples with the second half of samples and on the output side (frequency domain) we get a sequence of even indexed DFT terms followed by odd indexed DFT terms. If we compare this with decimation-in-time FFT then by analogy we can call this as decimation-in-frequency FFT. Hence, the term called '*decimation-in-frequency FFT*'.

The process of breaking down the bigger DFT into DFT of sum and difference of smaller sequences is continued till there is only one sample left in the sequence. The idea can be

grasped intuitively by Figures 7.13 to 7.15. Finally Figure 7.16 shows the full-fledged signal flow graph for the FFT with decimation-in-frequency.

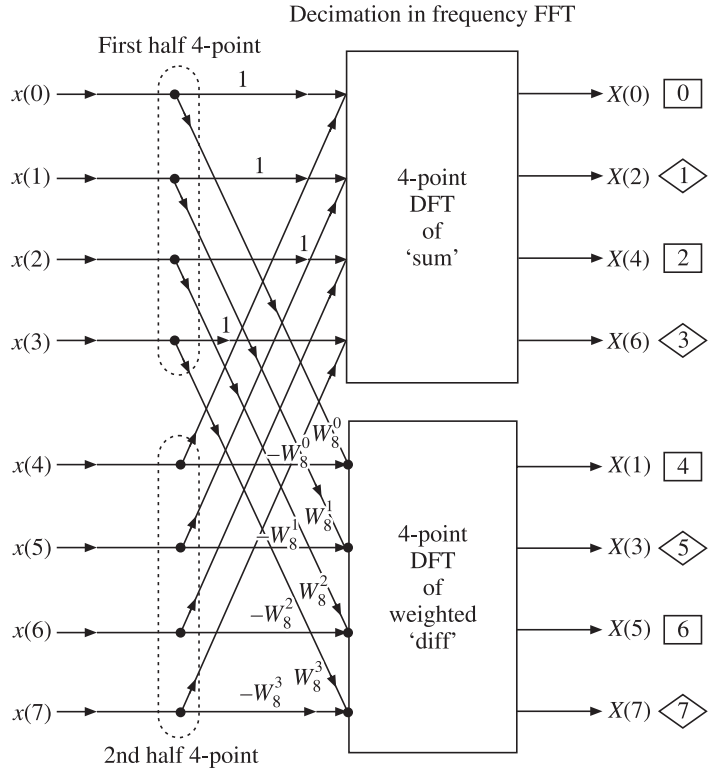


Figure 7.13 Development of decimation-in-frequency 8-point FFT.

The process of breaking down the DFT can be continued further till we get to one point DFT which is the signal itself. This is shown for an 8-point FFT in Figures 7.14 and 7.15.

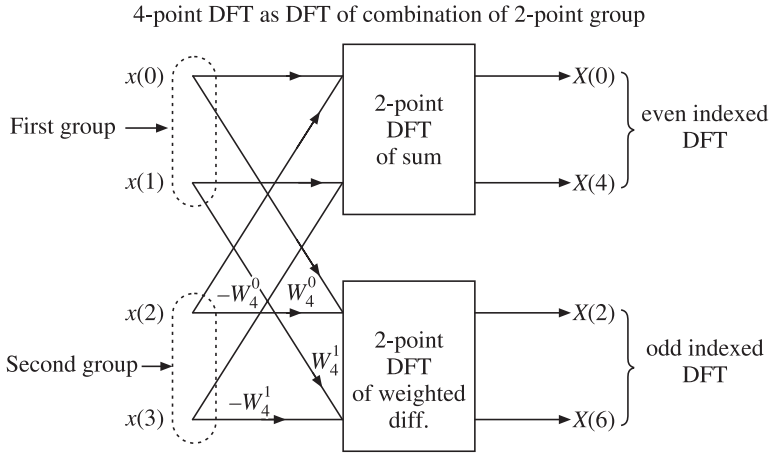


Figure 7.14 Breaking down a 4-point DFT into two number of 2-point DFTs.

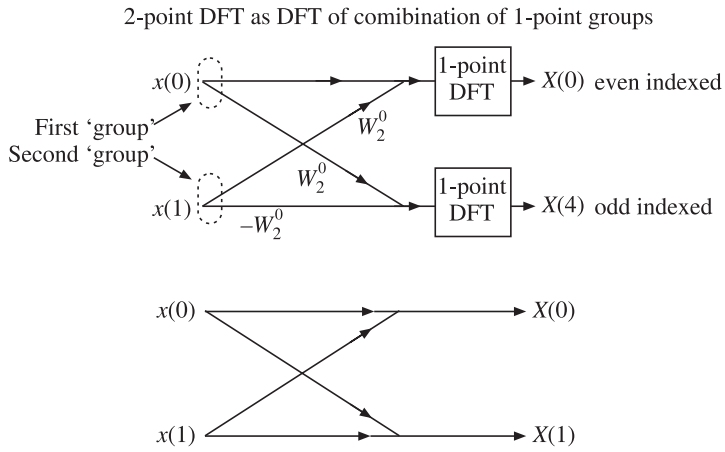


Figure 7.15 Further breaking down of 2-point DFT into 1-point DFTs.

7.3.1 Numerical Problem on FFT (DIF)

We repeat the problem given in Section 7.2.1 and solve it by decimation-in-frequency as shown in Figure 7.17 to get identical results.

7.4 MATLAB Implementation of FFT: Decimation-in-Time

```
% Program for testing my FFT DIT


clear, clc

x=[1 2 3 4 5 6 7 8];

XX = myFFTDIT(x)
YY = fft(x), % For checking results against MATLAB's fft()
function
% the user created function myFFTDIT() is include here for ease in
documentation
% function [X] = myFFTDIT(x)
% Implementation of decimation-in-time FFT
% N = numel(x);
% if (N == 1 )
% X = x ;
% else
% xeven = x(1:2:N-1);
% xodd = x(2:2:N);
% WNm=exp( (-j * 2 * pi / N)*(0:(N/2)-1 ) );
```

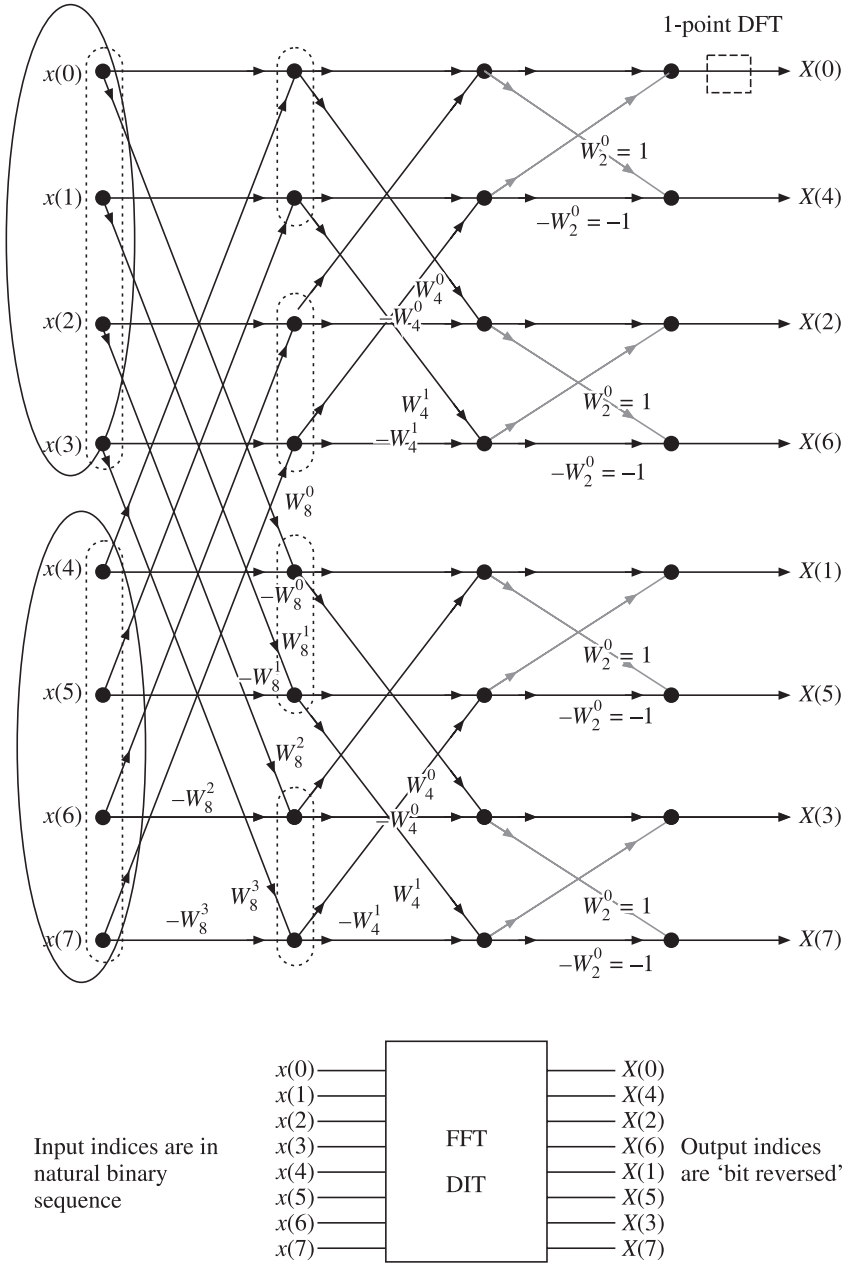


Figure 7.16 Fully developed 8-point FFT signal flow graph.

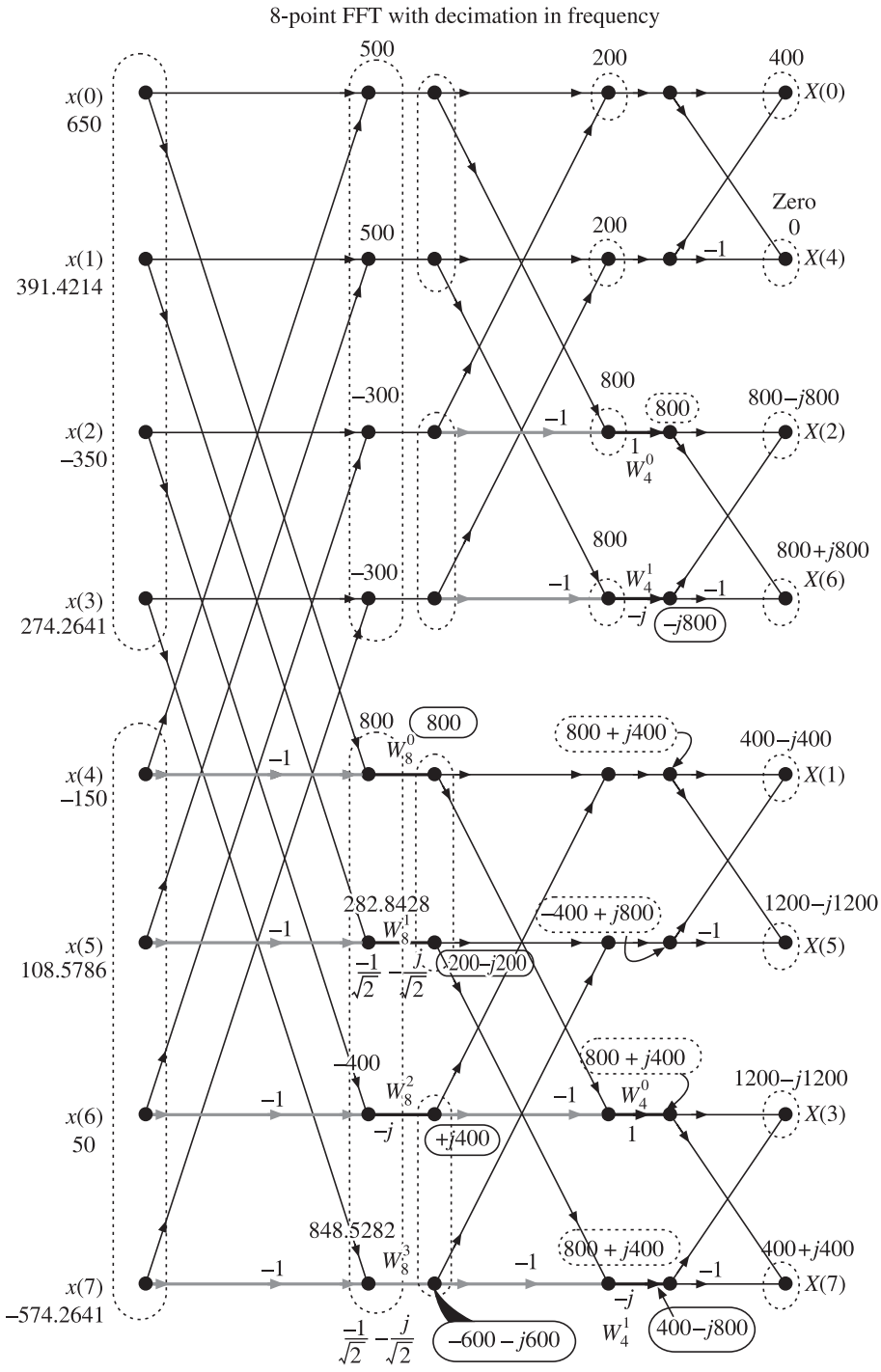


Figure 7.17 Numerical problem on FFT(DIF).

```
%
%  Am  =  myFFTDIT(xeven);
%  Bm  =  myFFTDIT(xodd);
%  X1  =  Am  +  Bm  .*  WNm;
%  X2  =  Am  -  Bm  .*  WNm;
%  X  =  [X1  X2]
%
%
%  end
```

```
XX=

Columns 1 through 6

36.0000 – 4.0000 + 9.6569i – 4.0000 + 4.0000i – 4.0000 + 1.6569i – 4.0000 – 4.0000 – 1.6569i

Columns 7 through 8

– 4.0000 – 4.0000i – 4.0000 – 9.6569i


YY=
Columns 1 through 6

36.0000 – 4.0000 + 9.6569i – 4.0000 + 4.0000i – 4.0000 + 1.6569i – 4.0000 – 4.0000 – 1.6569i

Columns 7 through 8

– 4.0000 – 4.0000i – 4.0000 – 9.6569i
```

7.5 MATLAB Implementation of FFT: Decimation-in-Frequency

```
% -------------- FFT, Decimation-in-Frequency Algorithm
% ---------------------------------------------------------------
%---------- The idea is to form two sequences of size N/2.
%---------- First sequence is formed by summing respective
elements of
%---------- first and second half.
%---------- Second sequence is formed by difference of the
above two, but
%---------- weighted by a weight W = exp( -j 2 pi n / N )
%----------- N/2 point DFT can be found for each of these
sequences ordered as
%----------- even indexed DFTs and odd indexed DFTs.
```

```
%------ This process is recursively continued till N/2 becomes 1
%------ 1-point DFT is the signal itself. This terminates the
recursion
%----------- Assumption is that N is a power of 2.

clear, clc

N = 8 ;
x =[ 1 2 3 4 5 6 7 8] ; % for testing
x
XX = myFFT(x);% FFT output in XX() is scrambled.

%-------------------- XXX() collects unscrambled FFT
```

```
XXX(1)=XX(1);% 0
XXX(5)=XX(2);% 4
XXX(3)=XX(3);% 2
XXX(7)=XX(4);% 6
```

```
XXX(2)=XX(5);% 1
XXX(6)=XX(6);% 5
XXX(4)=XX(7);% 3
XXX(8)=XX(8);% 7
```

Note:
Output of FFT
DIF is
scrambled.

```
XXX
YY = fft(x) % Compare the results with MATLABs fft() function

% The user defined function myFFT(x) is included here for ease
in documentation
% ----------------------------------------------------------------
% --------- function [ X ] = myFFT( x )
% ----------Decimation in Freq FFT Algo
% ----------------------------------------------------------------
% N = numel(x);
% if ( N == 1 )
% X = x ;
% else
%
% WNn = exp((-j*2*pi/N)*(0:(N/2)-1));
% first = x(1:(N/2))
% second= x((N/2)+1:N)
%
% sum = first + second ;
% diff = first - second ;
% diffW = diff .* WNn ;
```

```
% X1 = myFFT(sum);
% X2 = myFFT(diffW);
%
% X=[X1 X2];
%
%
% end
```

```
XXX = Columns 1 through 6

36.0000 –4.0000 + 9.6569i –4.0000 + 4.0000i –4.0000 + 1.6569i –4.0000 –4.0000 – 1.6569i

Columns 7 through 8

–4.0000 – 4.0000i –4.0000 – 9.6569i


YY = Columns 1 through 6

36.0000 –4.0000 + 9.6569i –4.0000 + 4.0000i –4.0000 + 1.6569i –4.0000 –4.0000 – 1.6569i

Columns 7 through 8

–4.0000 – 4.0000i –4.0000 – 9.6569i
```

7.6 Motivation for Goertzel Algorithm

This algorithm was published by Gerald Goertzel in 1958. Its utility stems from the fact that in many applications, we may not be interested in computing the entire frequency spectrum of a signal. In such situations the time and efforts taken by FFT in terms of computation resources may not be justifiable. For example, in majority of power system applications, one is interested in finding only the fundamental phasor, since the steady state theory is based on fundamental quantities. In some algorithms like transformer protection we may be interested in a few more harmonics like 2nd, 3rd and 5th but almost never in the full spectrum. Goertzel algorithm offers a more efficient method of computation in such circumstances. A popular application of this algorithm is recognition of the DTMF tones produced by the buttons pushed on a telephone keypad.

7.7 The Goertzel Algorithm [5, 13, 19, 22]

For the sake of development the Goertzel algorithm, we take the case of a 4-point DFT:

$$X(m)_{m=0}^{m=3} = \sum_{n=0}^{n=3} x(n) e^{\frac{-j2\pi mn}{4}}$$

Let $e^{\frac{-j2\pi}{4}} = W_4$ hence, $e^{\frac{-j2\pi mn}{4}} = W_4^{mn}$
Hence, we can write the 4-point DFT as:

$$X(m)_{m=0}^{m=3} = \sum_{n=0}^{n=3} x(n) W_4^{mn}$$

Let us say, we are interested in finding only the fundamental component; $X(1)$, then $m = 1$ and we can write:

$$X(1) = \sum_{n=0}^{n=3} x(n) W_4^{n}$$

expanding the summation:

$$X(1) = x(0)W_4^0 + x(1)W_4^1 + x(2)W_4^2 + x(3)W_4^3$$

since $W_4^0 = 1$; we can write:

$$X(1) = x(0) + x(1)W_4^1 + x(2)W_4^2 + x(3)W_4^3$$

$$X(1) = x(0) + W_4^1(x(1) + x(2)W_4^1 + x(3)W_4^2)$$

$$X(1) = x(0) + W_4^1(x(1) + W_4^1(x(2) + x(3)W_4^1))$$

$$X(1) = x(0) + W_4^1[x(1) + W_4^1\ (x(2) + W_4^1\ x(3))]$$

The algorithm can be implemented as shown in Figure 7.18.

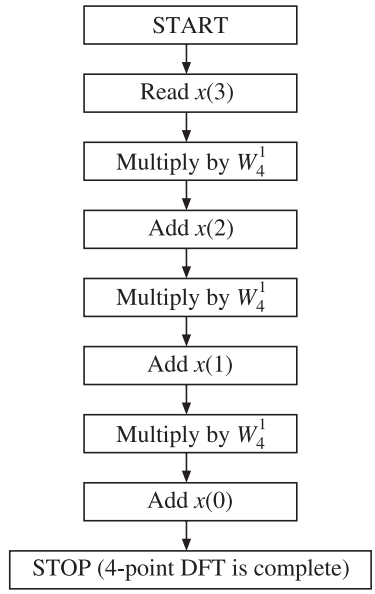


Figure 7.18 Flow chart of Goertzel algorithm for finding fundamental component of DFT.

The algorithm can be expressed more succinctly as shown in Figure 7.19.

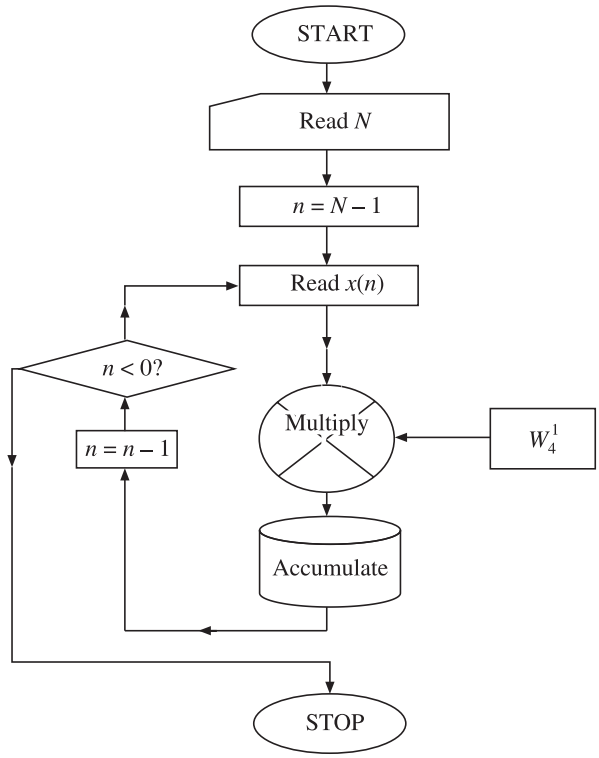


Figure 7.19 Flow chart of Goertzel algorithm for finding fundamental component of DFT.

We can also appreciate the Goertzel algorithm with the help of the signal flow graph shown in Figure 7.20.

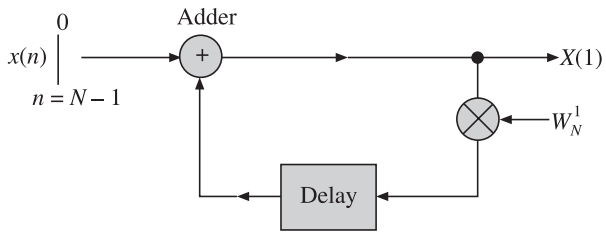


Figure 7.20 Signal flow graph of Goertzel algorithm.

We can run through the signal flow graph by following entries as in Table 7.4.

Table 7.4 Entries to signal flow graph

			Input to adder	Output	
Step	**Time**	***x*(*n*)**	**Output of delay block**	**Output of adder block**	**Input to delay block**
1	0	$x(3)$	0	$x(3)$	$W_4^1 x(3)$
2	1	$x(2)$	$W_4^1 x(3)$	$x(2) + W_4^1 x(3)$	$W_4^1(x(2) + W_4^1 x(3))$
3	2	$x(1)$	$W_4^1(x(2) + W_4^1 x(3))$	$x(1) + W_4^1(x(2) + W_4^1 x(3))$	$W_4^1(x(1) + W_4^1(x(2) + W_4^1 x(3)))$
4	3	$x(0)$	$W_4^1(x(1) + W_4^1(x(2) + W_4^1 x(3)))$	$x(0) + W_4^1(x(1) + W_4^1(x(2) + W_4^1 x(3)))$ (4-point fundamental DFT complete)	–

7.8 Implementation of Goertzel Algorithm in MATALB

```
%-----------------------------------------------------------------
%----------------- Goertzel Algorithm ------------------------
%-----------------------------------------------------------------
 clear,clc

N=8 ; WN1 = exp(-j*2*pi* 1 /N);

X1=0;
x=[0 1 2 3 4 5 6 7] ;


for k=N:-1:1
 X1= X1 * WN1 + x(k)
end

X1
%------- Lets check our result with MATLAB fft()
XX= fft(x);
XX1=XX(2) % First term of FFT is index 1 and not zero and is
the DC term, hence XX(2)

X1=7
```

X1 = 10.9497 – 4.9497i

X1 = 9.2426 – 11.2426i

X1 = 2.5858 – 14.4853i

X1 = –5.4142 – 12.0711i

X1 = –10.3640 – 4.7071i

X1 = –9.6569 + 4.0000i

X1 = –4.0000 + 9.6569i

X1 = –4.0000 + 9.6569i ← Final value of X1 found by Goertzel algorithm

XX1 = –4.0000 + 9.6569i ← Value of X1 found by MATLAB's FFT function

Review Questions

1. What is the motivation for development of the FFT algorithm?
2. For an N-point DFT, what is the number of multiplications and number of additions?
3. Explain the concept of decimation.
4. Show how you will break down a 16-point DFT.
5. What is the number of multiplications required for FFT of 16 real samples?
6. What is the number of additions required for the FFT of 16 real samples?
7. Why FFT had a dramatic impact on signal processing?
8. What is the MATLAB function for performing FFT?
9. Perform FFT of [1, 2, 3, 4] and comment upon it.
10. Sketch the signal [1 2 3 4] and its recurring version.
11. Derive the decimation-in-frequency FFT algorithm.
12. Sketch the signal flow graph for an 8-point decimation-in-frequency FFT algorithm.
13. Why is it called as decimation-in-frequency?
14. What is special about the sequence of output samples in DIF FFT?
15. Write binary codes for the indices of samples of the output in DIF FFT and show how they can be obtained from natural binary sequence.
16. What are the numerical values of the following weights?

$$[W_8]^{0,}\,[W_8]^{1,}\,[W_8]^2\,[W_8]^3$$

17. What are the numerical values of the following weights?

$$[W_4]^0, [W_4]^1$$

18. What are the numerical values of the following weights?

$$[W_2]^0$$

19. What is the butterfly in the decimation-in-frequency algorithm of FFT?

20. Convert the FFT-DIT algorithm into a MATLAB program.

21. What do you mean by the term 'decimation'?

22. What do you mean by 'decimation-in-time'?

23. What do you mean by decimation-in-frequency?

24. What is the difference between DFT and FFT?

25. A signal has 8 samples. What will be the number of samples in the output of the FFT?

26. Normally inputs to the FFT are real numbers. What can you say about the nature (real/complex) of FFT of such signals?

27. How will you extract the magnitude spectrum from the FFT?

28. How will you extract the phase spectrum from the FFT?

29. Is FFT algorithm implemented recursively suitable for real-time applications?

30. What is the motivation for development of Goertzel algorithm?

31. Develop Goertzel algorithm for second harmonic extraction.

32. Develop Goertzel algorithm for '*m*th' harmonic extraction.

33. Calculate the number of multiplications and additions required by the Goertzel algorithm.

8

Windowing and Spectral Leakage

8.1 The Fourier Machine

Fourier transform can be depicted as a machine extracting the various frequency components and collecting them into individual bins as depicted in Figure 8.1. If the sampling frequency

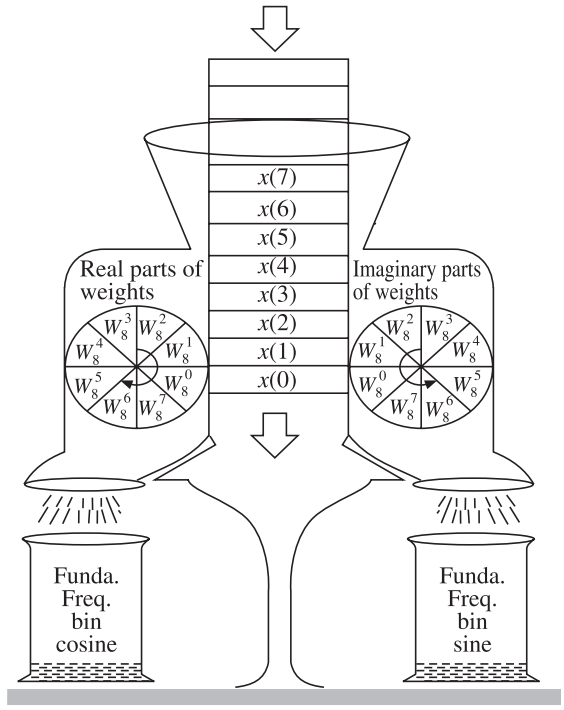


Figure 8.1 The Fourier machine.

is an integral multiple of the signal frequency and the window length is adjusted to contain integral number of cycles of the signal then the various frequency components get collected in the correct bin. However, if the sampling frequency is not an integral multiple of the signal frequency or if the window length is such that it does not contain integer number of cycles of the signal, all the bins collect something, even if the signal has only single frequency component. Thus, the frequency components are said to 'leak' into those bins where they are not supposed to show up.

8.2 Windowing without Spectral Leakage

In order to perform signal processing, we have to pick samples of signal for a finite duration, say T_{window}. Mathematically, this is equivalent to multiplying the signal with a rectangular window of magnitude equal to 1 inside the window and zero outside the window. Thus, the signal that we process is not the original signal but product of the 'original signal' and the 'window signal'. It is therefore natural that the spectral characteristics of the 'windowed' signal will be different from that of the original signal. Since there is no escape from windowing, we have to live with the fact that the spectrum of the windowed signal will be different from that of the original signal. The next best thing that we can do is to strive for a window which causes minimum contamination in the spectral content of the windowed signal from that of the original signal. This motivates the study of various window shapes.

The following points should be kept in mind in order to appreciate the topic of windowing and spectral leakage.

1. Windowing cannot be avoided.
2. It is helpful to think in terms of frequency 'bin', which is easy to imagine if we visualise the 'Fourier machine' which is a mechanistic way of looking at the Fourier transform.
3. By windowing we are introducing a new signal, i.e., the 'window signal' into the picture in addition to the signal being processed. It is natural to expect that this additional signal will show up in some way. In order that, windowing has least undesirable effect, we will have to give very careful thought to the shape of the window. Intuitively, the gentler the slope of the window at the edges, the better it is.
4. In time domain, windowing is simple multiplication between the samples of the signal and samples of the window function.
5. In frequency domain, windowing amounts to convolution between the frequency spectra of signal and frequency spectra of window function.
6. Because of convolution in frequency domain, it can be expected that frequency spectrum of the windowed signal will 'smeared'. Thus, a solitary line in the spectrum of a signal may get transformed into a broader spectrum. This is obviously going to cause confusion in the frequency domain, thus, leading to errors.
7. Rectangular window causes a drastic effect in time domain, because it multiplies everything outside the window by zero. Hence, equally drastic effect in the frequency domain can be expected.

8. Sampled data system has 'vision' only at integral multiples of $f_{sampling}/N$, i.e., at analysis frequencies of $mf_{sampling}/N$; where 'm' is from 0 to $N/2$. Thus, we have vision only at discrete points in the frequency domain, up to a maximum frequency of $f_{sampling}/2$ at $m = N/2$. At all other points in the frequency domain, we remain blind.
9. Frequency response of DFT is zero at integral multiples of fundamental. And these are the points at which the DSP system has vision.
10. Frequency response of DFT is non-zero at non-integral multiples of fundamental, but at these points DSP does not have vision.

It is to be noted that we are performing the mathematical operation of multiplication between signal and window in the time domain. Multiplication in time domain has effect of convolution in frequency domain. Hence, spectrum of *time windowed* signal can be found by convolution of spectra of signal and window function.

Spectrum of the rectangular function is the well-known sinc() function. Hence, the spectrum of the rectangular windowed signal is obtained by convolution of sinc() function with spectrum of the signal.

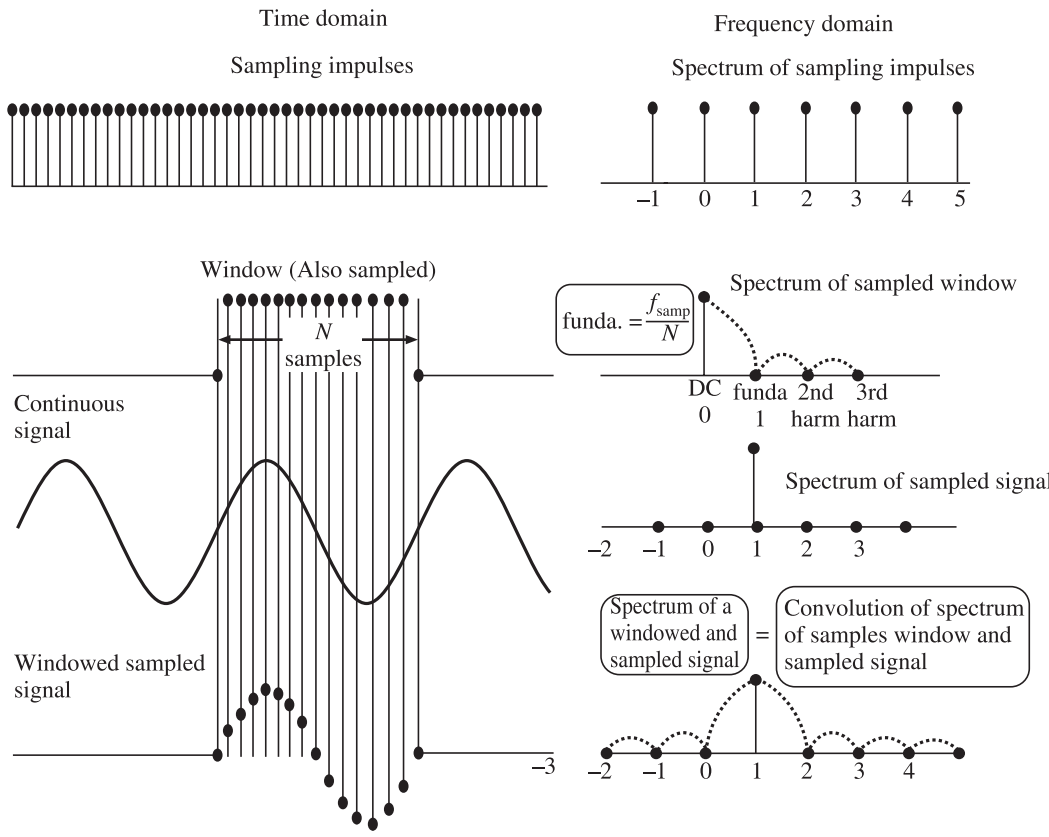


Figure 8.2 The sampled and windowed signal in time and frequency domains.

Figure 8.2 shows that in spite of sampling and windowing, the spectrum of the signal has remained intact! Careful observation of Figures 8.2 and 8.3 will reveal that this feat was possible only because of the following:

1. Selection of the window size to match one complete cycle of the signal.
2. Selecting sampling frequency as an integral multiple of signal frequency.

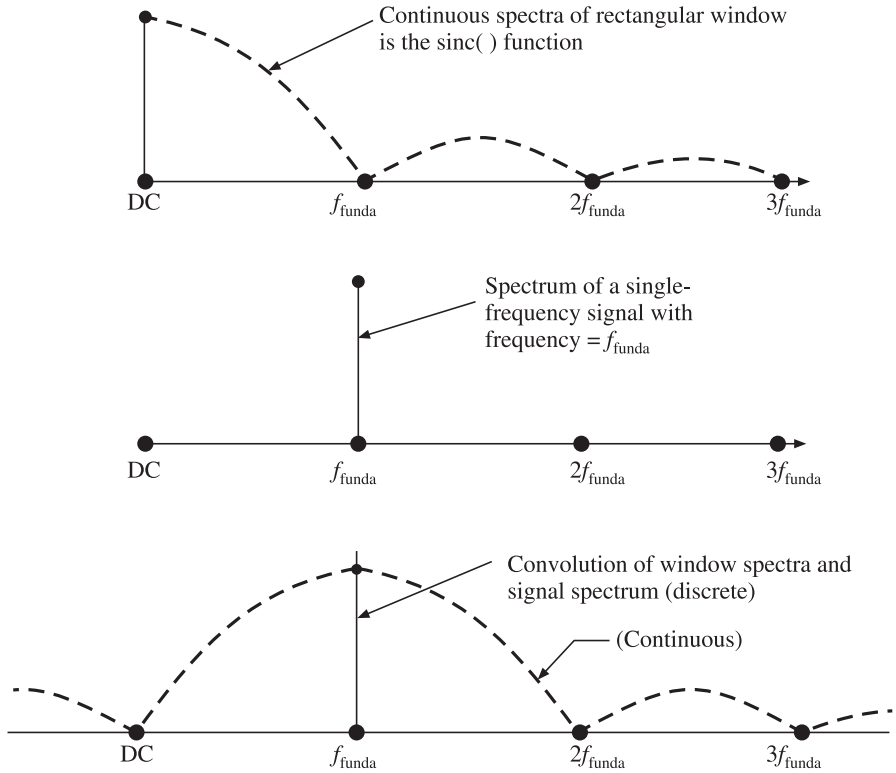


Figure 8.3 Convolution between window spectra and signal spectrum.

The convolution operation between the window spectrum and signal spectrum can be verified with the help of Figure 8.4.

It can now be easy to appreciate that in real life these conditions can seldom, if ever, be satisfied. This raises the natural question: What will happen if:

1. a fraction of cycle of signal appears in the window?
2. sampling frequency is not an integral multiple of the signal frequency?

The answer to both above questions is that in such a circumstance there will be spectral leakage. We take up spectral leakage in the next section.

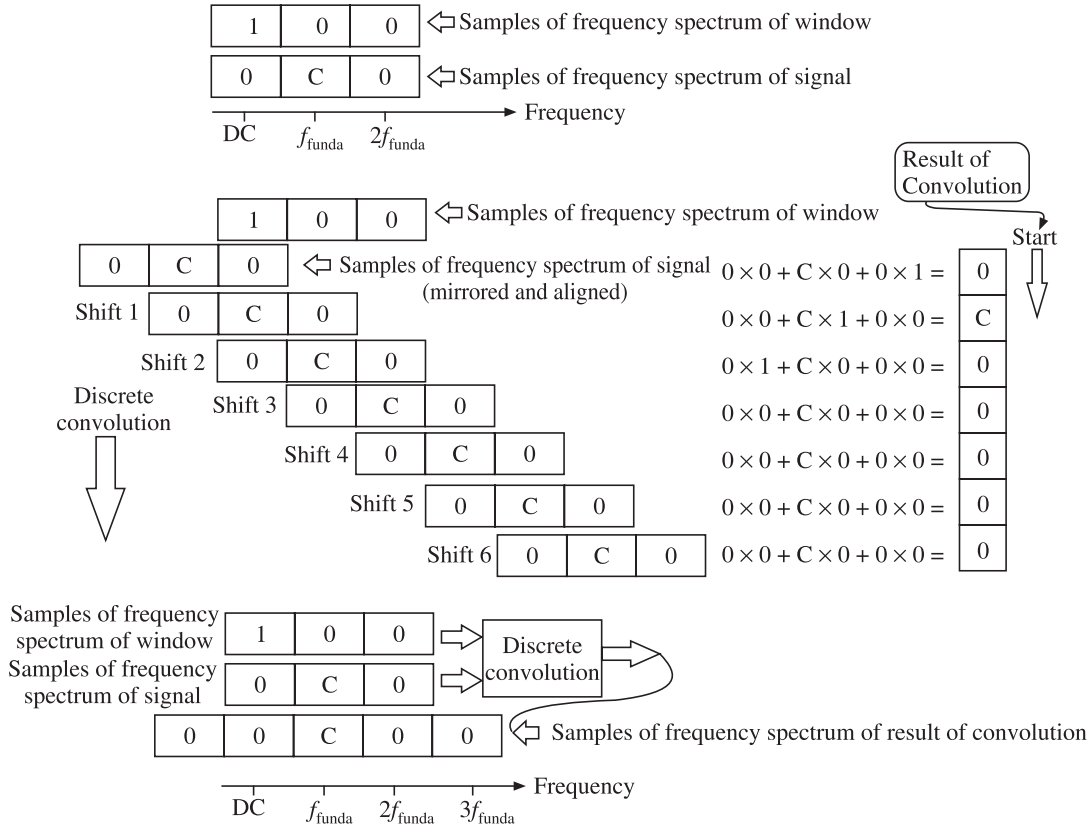


Figure 8.4 Discrete convolution between window spectra and signal spectra.

8.3 Windowing Leading to Spectral Leakage [5, 13, 19, 22]

Any phenomena in signal processing, in general, can be viewed from two alternate viewpoints: from frequency domain viewpoint and from time domain viewpoint. Both the views give different insight into the phenomena, making its understanding clearer and deeper. Let us first view the spectral leakage from time domain viewpoint.

8.3.1 Spectral Leakage from Time Domain Viewpoint

Consider the windowing of an undistorted 50 Hz signal. If we adjust the window length to exactly 20 ms as shown in Figure 8.5 then there is exactly one cycle of the signal inside the window. The implied windowed signal which is obtained by extending the signal, on both sides, outside the window, also happens to be a smooth undistorted signal. Hence, additional spurious frequency components are not generated. Next consider that the window remains fixed at 20 ms but the signal frequency drops to 45 Hz. Note that the signal is undistorted. It can be seen from Figure 8.5 that the implied windowed signal is no longer undistorted and severe

distortions have appeared at the window edges! This is bound to cause additional frequency components in the frequency spectrum of the windowed signal. These additional spurious frequency components are referred to as '*spectral leakage*' in the signal processing literature.

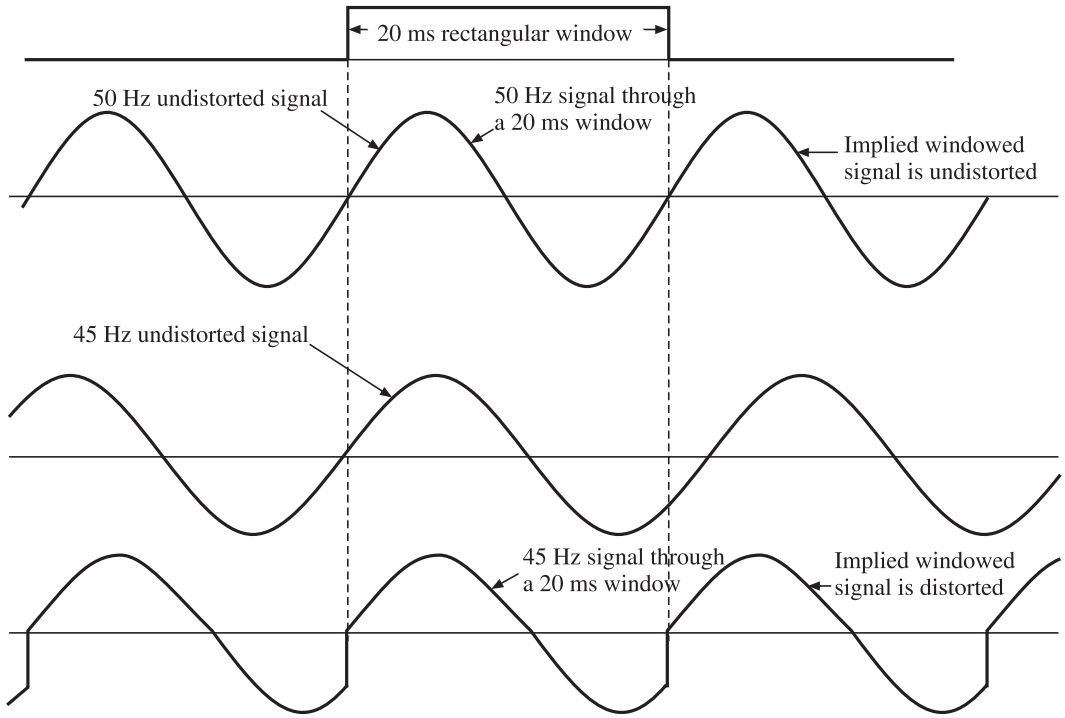


Figure 8.5 Spectral leakage from time domain viewpoint.

8.3.2 Spectral Leakage from Frequency Domain Viewpoint

Figure 8.6 shows that when the window length is exactly equal to time period (T) of the signal then there is no spectral leakage. However, if there is a mismatch between the window length and signal time period then there is spectral leakage. It can be seen from Figure 8.6 that a 50 Hz signal (T = 20 ms) viewed through a 20 ms rectangular window does not cause any spectral leakage, however a 45 Hz (T = 1/45 = 22.2222 ms) undistorted signal processed through the same rectangular window of 20 ms causes spectral leakage in all the bins. In practice it is not possible to match the window length precisely with the fundamental time period of the signal. Hence, some degree of spectral leakage is unavoidable. The next best thing that we can do is to search for a window shape which will minimise the spectral leakage.

From Figure 8.5, it is clear that the basic cause of spectral leakage is the abrupt change in the signal magnitude at the edges of the rectangular window. Smoother window edges give rise to lesser leakage. In the literature on signal processing, many window shapes have been reported, however the theme common to all window shapes is to smoothen out the transition

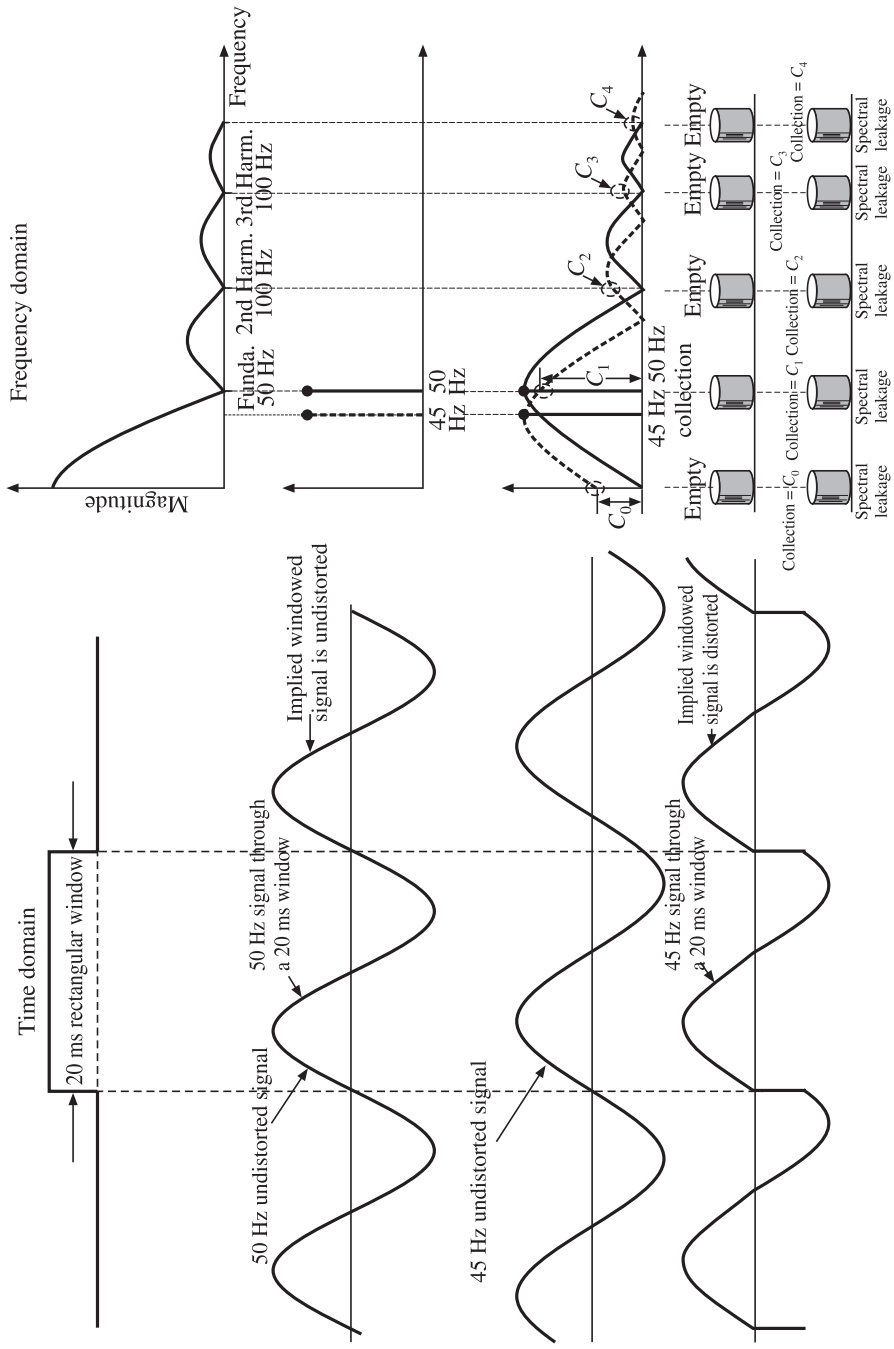


Figure 8.6 Frequency domain view of spectral leakage.

to zero at the edges of the window. If we look at the spectrum of a window function, we will notice that in general there is an inverse relationship between width of main lobe and height of side lobe, i.e., side lobe ripple. Ideally we would like the width of main lobe to be zero and side lobe ripple to be zero as well, which makes it an impulse at 0 Hz. However, such a window shape (impulse) in frequency domain corresponds to infinite window length in time domain, obviously an impractical proposition. Thus, we have to live with ill effects of windowing, and the best that we can do is to choose a window shape which will minimise spectral leakage. Table 8.1 lists a few popularly used windows.

Table 8.1 Popularly used windows

1.	Hanning window	$\sum_{n=0}^{N-1} w(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{N}\right)$
2.	Hamming window	$\sum_{n=0}^{N-1} w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N}\right)$
3.	Triangular window	$\sum_{n=0}^{N/2} w(n) = \left(\frac{n}{N/2}\right)$
		$\sum_{n=\frac{N}{2}+1}^{N-1} w(n) = 2 - \left(\frac{n}{N/2}\right)$

Many other windows are reported in the literature, which are named after their inventors. Some of these are: Bartlett, Blackmann, Chebyshev, Gaussian, Kaiser, Nuttal, Parzen and Tuckey.

Figure 8.7 shows three windows in time and frequency domain. It can be observed that in the frequency domain, the peak of rectangular window is shown normalised to 1. This is known as *processing gain* or *coherent gain* of the window. This is in fact the DC signal gain of the window. For all other windows, the gain is reduced because the window smoothly goes to zero near the boundaries. The processing gain of the triangular window and the Hamming window are 0.5 and 0.5328 respectively. This is a known bias on spectral amplitudes of the windowed signal.

8.4 Windowed and Sampled Signal

Figures 8.8 and 8.9 show a signal which is windowed and sampled by using a triangular and Hamming window respectively. It can be seen that in the middle of the window, the amplitude is remains relatively unchanged while at the edges the signal magnitude is smoothly brought to zero.

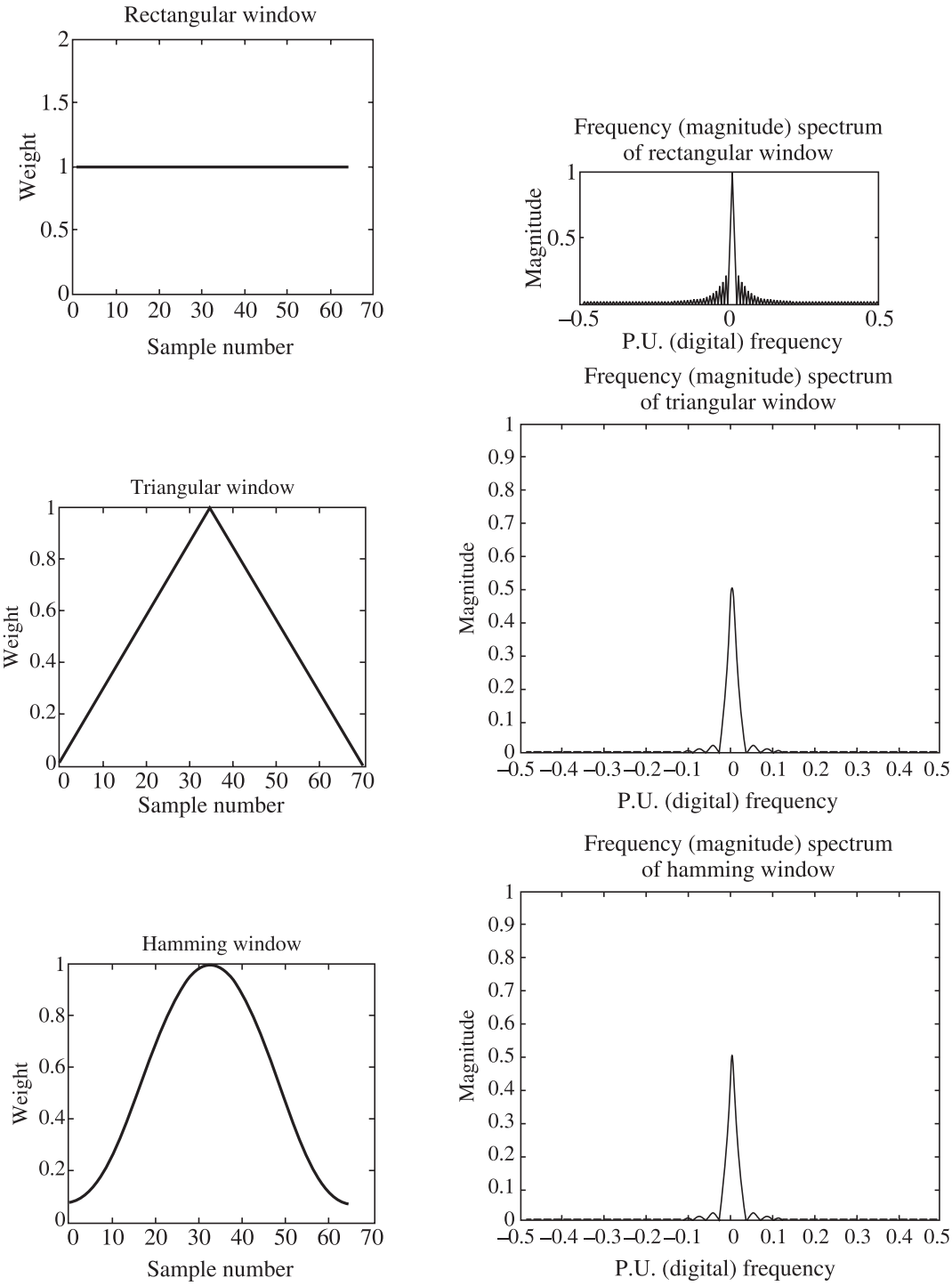


Figure 8.7 Various windows and their spectra.

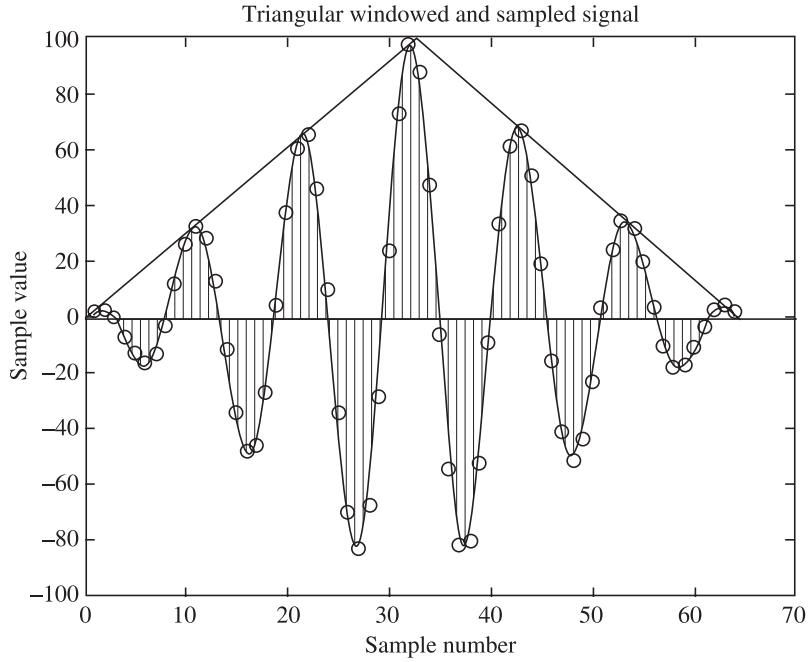


Figure 8.8 Triangular windowed and sampled signal.

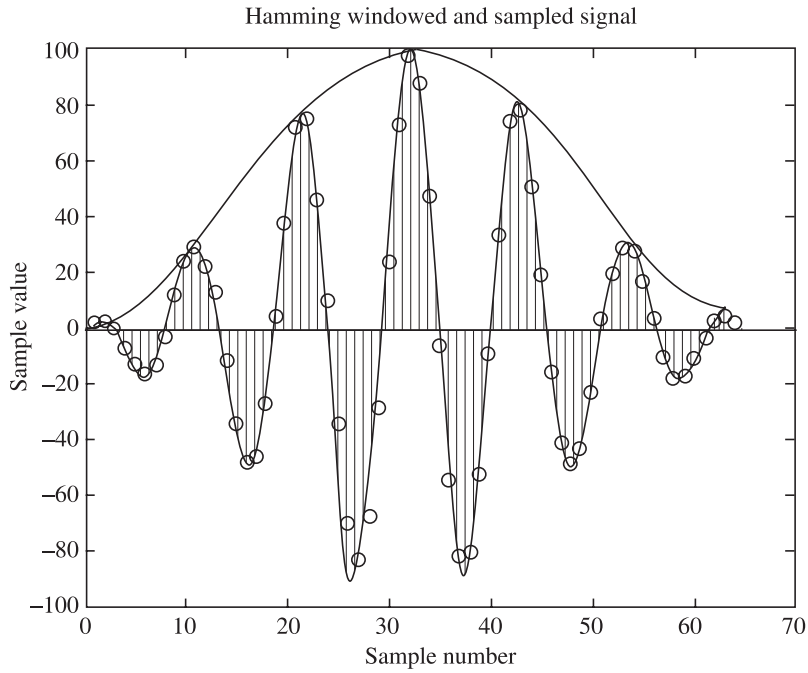


Figure 8.9 Hamming widowed and sampled signal.

REVIEW QUESTIONS

1. Explain the concept of frequency 'bin'.
2. What factors fix the position of the 'bin' on the frequency axis?
3. What do you mean by 'spectral leakage'?
4. In practise is it possible to avoid spectral leakage?
5. When is it possible to have a situation, where, in spite of rectangular window, there is no spectral leakage?
6. What do you mean by windowing?
7. Why windowing is unavoidable?
8. Why rectangular shape gives rise to a 'drastic' window?
9. List various window shapes and their mathematical functions.
10. Discuss: In time domain ideal window is that with infinite length.
11. Discuss: In frequency domain the ideal window is an impulse (The dual of the above statement).
12. Define processing gain of window (also known as coherent gain) and list its value for various windows.

9

Introduction to Digital Filtering

9.1 Introduction to Filters

What is common among the following:

1. Your sun-glasses
2. Water/Tea/Coffee dispensing machine
3. TV or any kind of radio receiver including your mobile telephone
4. Spike buster
5. DC power supply
6. Noise cancelling microphone
7. Lightening arrestor
8. Mosquito net
9. An antenna
10. Tendency of students to tune-out teacher's words or advice?

A little thought will convince you that all these things involve the basic act of filtering. But what is filtering? We intuitively understand filtering. Giving preferential treatment to one entity as compared to some other entity could be one general definition. Thus, we can see that the concept of filtering is ubiquitous. *Filtering* is not limited to the domain of electrical engineering but can take place in almost any domain like chemical, optical or even psychological! In this chapter we are going to concentrate upon a special version of filters called *digital filters*.

Everywhere you see, you come across filters. Life will be very difficult without filters. We are going to study the 'digital' versions of electrical filters. The speciality of these digital filters is that they receive a stream of numbers as input and give out another stream of numbers as output according to some algorithm. Thus, in the true spirit of GIGO philosophy, it is 'numbers-in' and 'numbers-out'.

Thus, the world around us is full of filters. Filters help us to make sense out of the deluge of information that floods us (the system). With everything going digital, it was inevitable that filters will also take on a digital avatar! So we are confronted with digital filters.

9.2 What is a Digital Filter?

The term *digital filter* refers to a computational process by which a sequence of numbers, usually samples of an analog signal is transformed into a second sequence of numbers as shown in Figure 9.1.

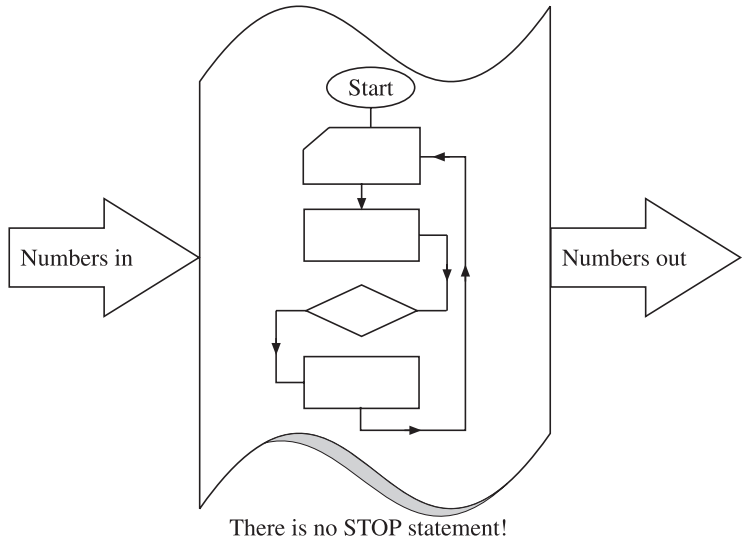


Figure 9.1 Digital filter as a number-in, numbers-out system.

It can be proved that the computational process that goes on inside the box corresponds to some kind of filtering operation like high-pass, low-pass, band-pass or band-stop filtering and differentiating, integrating, convolution or co-relation. In fact every act of processing on time domain samples creates some or the other effect in the frequency domain, be it intentional or unintentional. We can also implement other types of digital filters which cannot be classified into one of the basic four types listed above, such as *adaptive filters* used for noise cancelling and beam steering. For example we can devise an adaptive digital filter for picking out an EEG signal buried deep in 50 Hz line noise. We can design a filter for predicting tomorrows share price index or the electrical load on the power system during the next hour and so on. Keeping in mind that an antenna is a spatial filter, we can design it in such a way as to have a null in the direction of an interfering signal, without, of course physically moving the antenna! (Adaptive beam steering).

Thus, any mathematical operation is an act of filtering. Even if you do not do anything to samples of a signal and just pass them from input to output, it is an 'all-pass' filtering operation.

9.3 Why Digital Filtering?

Before we venture into detailed discussion about digital filters, it will be pertinent to ask; what motivates us to go for digital filtering? There are certain advantages of using digital filtering which are listed in Table 9.1.

Table 9.1 Advantages of digital filter over analog filter

	Attribute	Analog filter	Digital filter
1	Consistency	Analog filters exhibit deterioration in their performance because of drift in the component values with time and temperature.	Once designed the digital filters exhibit rock-steady performance as there is no ageing effect and temperature plays no role.
2	Repeatability	Every piece of the filter has slightly different characteristics from the prototype because of component tolerances. Hence, every piece needs to be tested and checked.	Once designed the billionth piece will perform exactly as the prototype. Hence, performance is perfectly repeatable and each 'instance' of the filter need not be checked for performance!
3	Precision	It is limited by the precision of the analog parts. Increasing precision beyond a point becomes very costly and hence impractical.	Precision can be increased to an arbitrarily high value (at least theoretically) by increasing the number of bits used to represent the values. This can be done either in hardware by using a CPU with large word length or can be done in software.
4	Cost	They turn out to be costly as compared to digital filters.	Because of advancements in VLSI techniques, they are turning out to be more and more economical. As per Moore's law, capabilities of semiconductors are doubling every 18 months. This is continuously driving the costs in downward direction.
5	Very low frequency	It becomes problematic to process very low frequency signals as component values and physical size of components becomes impractical.	No such limitation in case of digital filters. They can handle low frequency signals with ease.

However, there are some drawbacks of digital filters. These are listed in Table 9.2.

Table 9.2 Disadvantages of digital filter over analog filter

	Attribute	Analog filters	Digital filters
1	Very high frequency signals	Analog filters can process very high frequency signals upto the gigahertz range.	It is difficult to process signals in the gigahertz range (on a real time basis) as clock speeds need to be in tens and hundreds of gigahertz range. At present such high clock speeds are not available.
2	Power consumption	Low power consumption	High power consumption as compared to analog technique. Further, power consumption goes on increasing with clock speed.
3	Simplicity	Analog filters are simple to visualise.	Digital filters are more complex as compared to analog filters.

9.4 FIR and IIR Filters [5, 13, 19, 22]

Digital filters are classified as finite impulse response (FIR) and infinite impulse response (IIR). The basic philosophies of FIR and IIR are shown in Table 9.3, and Table 9.4 shows the comparison between FIR and IIR filters.

Table 9.3 Basic philosophies of FIR and IIR

FIR filter	IIR filter
Present output = *function of* (present input and '*p*' number of past inputs)	Present output = *function of* (present input and '*Q*' number of past inputs, and '*P*' number of past outputs)
Coefficients remain constant w.r.t. time	Coefficients remain constant w.r.t. time

Table 9.4 Comparison between FIR and IIR Filters

	Characteristics	FIR	IIR
1	Number of computations to get the same filtering effect	Most	Least
2	Non-recursive implementation possible	Yes	No
3	Recursive implementation possible	Yes	Yes
4	Can simulate analog filter prototype	No	Yes
5	Linear phase	Yes, provided impulse response (coefficients) exhibit either even or odd symmetry	No
6	Stability	Guaranteed if implemented non-recursively	Has to be ensured by design
7	Amount of hardware memory required	Most	Least
8	Sensitivity to filter quantization	Very low	High for direct form but can be reduced by cascade or parallel form
9	Probability of over-flow	Very low	Can be high for direct form but can be reduced by cascade form or parallel form.
10	Difficulty of quantization noise analysis	Least	Most
11	Hardware filter complexity	Simple	Moderate
12	Ease of design	Simple	Moderately complex
13	Adaptive filtering possible	Yes	Yes
14	Suitability for use in numerical protective relays	More suitable since it quickly forgets past (pre-fault) inputs and therefore is more affected by post fault condition.	Less suitable since it remembers all past inputs, therefore responds to pre-fault conditions.

The IIR filters can be naturally implemented using recursion because of the inherent feedback. Hence, they are sometimes referred to in the literature as recursive filters. There is no apparent recursion in the FIR filters and are naturally implemented using non-recursive procedures. Hence, they are referred to in the literature as non-recursive filter. However, it can be shown that both FIR as well as IIR filters can be expressed recursively or non-recursively. Hence, it is better to classify them as FIR and IIR instead of recursive and no-recursive.

FIR filters are the simplest of all. Hence, we first take up the explanation of FIR filters.

In FIR filters as shown in Table 9.3, the present sample of output is produced by linearly combining the present input with a finite number of past input samples. FIR filters do not use any of the past output samples, they use past input samples only.

Present FIR filter output = f[present input, past inputs]

$y(n) = f[x(n), x(n-1), x(n-2), \ldots, x(n-k)]$

Figure 9.2 shows a block diagram representation of FIR and IIR filters. The block diagram also suggests the hardware that will be required in case the filter is realised in hardware.

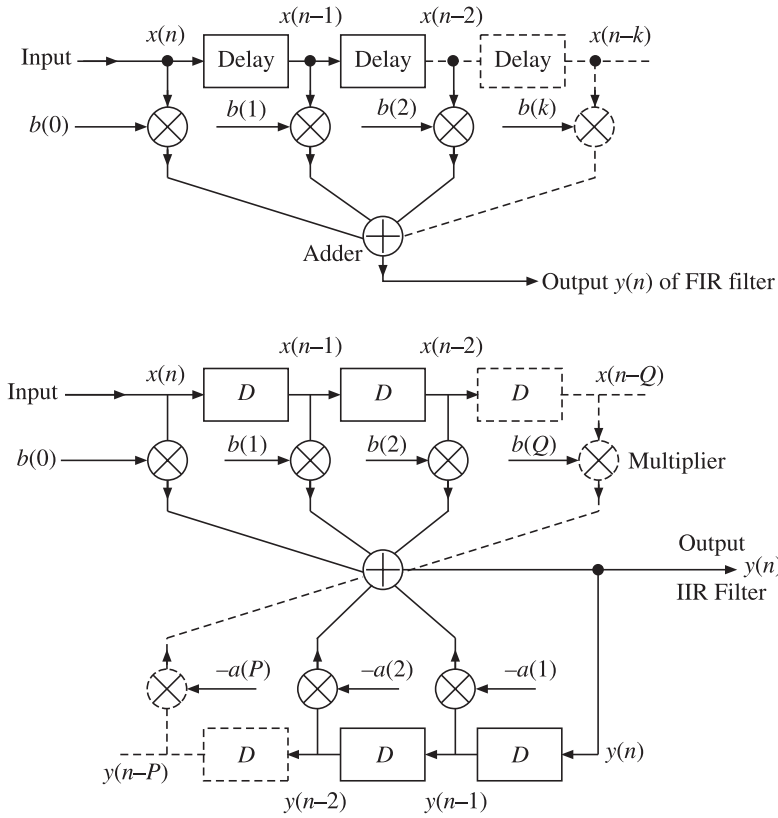


Figure 9.2 Block diagrams of FIR and IIR filters suggesting hardware implementation.

Thus, the block-diagram serves a dual purpose of a flowchart as well as hardware diagram of the digital filter! It can be seen that multiplier, adder, delay (MAD) and memory are the basic elements from which any digital filter of the FIR or IIR type can be constructed.

The outputs of the FIR and IIR filters, as can be seen from the block diagram, will be given by:

$$\text{FIR} \rightarrow y(n) = b(0)\,x(n) + b(1)\,x(n-1) + b(2)\,x(n-2) + \cdots + b(k)\,x(n-k)$$

$$\text{IIR} \rightarrow y(n) = b(0)\,x(n) + b(1)\,x(n-1) + b(2)\,x(n-2) + \cdots + b(Q)\,x(n-Q)$$
$$- a(1)\,y(n-1) - a(2)y(n-2) - \cdots - a(P)\,y(n-P)$$

9.5 Running Average as FIR Filter: 5-Point Running Average Filter

We will first get familiar with the symbolism associated with digital filters and then take up simple mathematical analysis to show that the whatever processing is occurring can indeed be called as a *filter*.

Consider a FIR filter whose output is the running average of present input sample and previous four input samples. Thus, we can write the present output sample $y(n)$ as:

$$y(n) = \frac{x(n) + x(n-1) + x(n-2) + x(n-3) + x(n-4)}{5}$$

Which can be written as:

$$y(n) = \frac{1}{5}x(n) + \frac{1}{5}x(n-1) + \frac{1}{5}x(n-2) + \frac{1}{5}x(n-3) + \frac{1}{5}x(n-4)$$

Which can be depicted in the form of a signal flow graph as shown in Figure 9.3.

The questions that naturally arise are:

- Why is it called a filter? Does it work like a filter?
- Which kind of filter?
- Why is it classified as FIR?

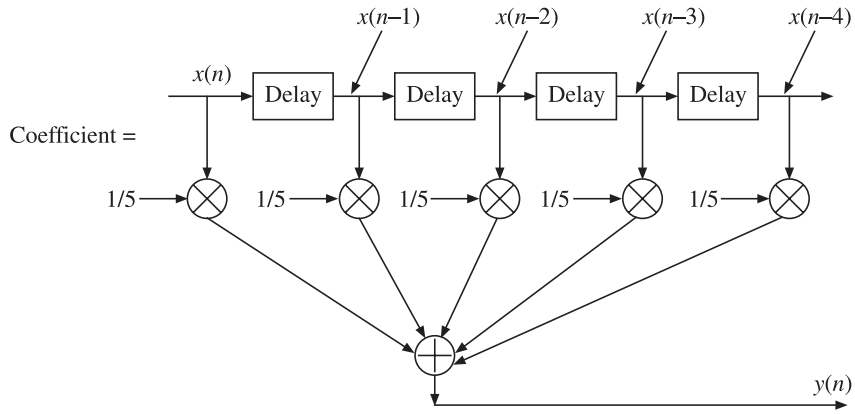


Figure 9.3 Block diagram of 5-point running average filter.

We will answer all the above questions as we progress. Let us write the equation for $y(n)$ as follows:

$$y(n) = h(0)\,x(n) + h(1)\,x(n-1) + h(2)\,x(n-2) + h(3)\,x(n-3) + h(4)\,x(n-4)$$

Which can be written in the compact form as:

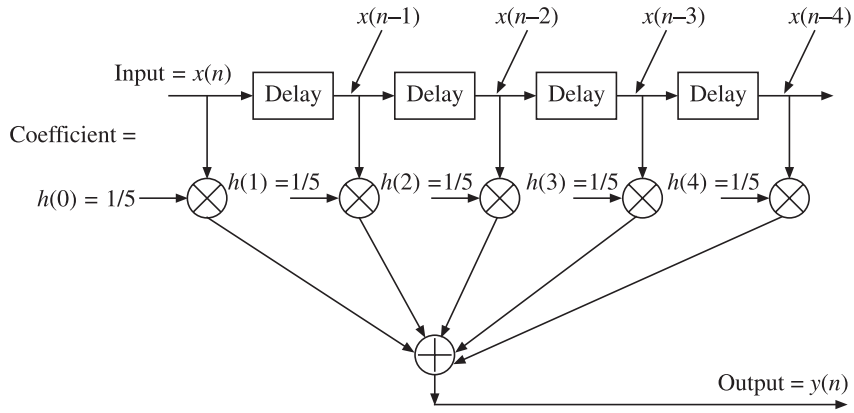
$$y(n) = \sum_{k=0}^{k=4} h(k)x(n-k)$$

Does the above equation look familiar? It surely does, as we can easily see that it is the operation of convolution. It is a convolution between two sequences $h(\ )$ and $x(\ )$. This is a very important result because it gives us a procedure for implementing digital filter and provides an insight into its working.

9.5.1 Impulse Response of 5-Point Running Average Filter

Let us investigate the working of the FIR filter by applying a unit impulse signal to it. The impulse signal is a very useful signal for the purpose of 'interrogating' various digital systems, since it reveals the information about the inner working of the digital system.

Figure 9.4 shows a unit impulse applied to the filter. We can see that the response to the



	Input					Output	
Time	$x(n)$	$x(n-1)$	$x(n-2)$	$x(n-3)$	$x(n-4)$	$y(n)$	
0	1	0	0	0	0	$1/5 = h(0)$	Duration of impulse response
T_s	0	1	0	0	0	$1/5 = h(1)$	
$2T_s$	0	0	1	0	0	$1/5 = h(2)$	
$3T_s$	0	0	0	1	0	$1/5 = h(3)$	
$4T_s$	0	0	0	0	1	$1/5 = h(4)$	
$5T_s$	0	0	0	0	0	0	
$6T_s$	0	0	0	0	0	0	

Figure 9.4 Computation of impulse response of 5-point running average filter.

impulse lasts for a finite duration after which it becomes zero. Hence, it is named as finite (time) impulse response or FIR filter. We can also see that the impulse causes the coefficients of the filter to be outputted one-by-one. Since the coefficients are all that can be known about a filter, the impulse response reveals everything about the filter. The impulse response is sketched in Figure 9.5.

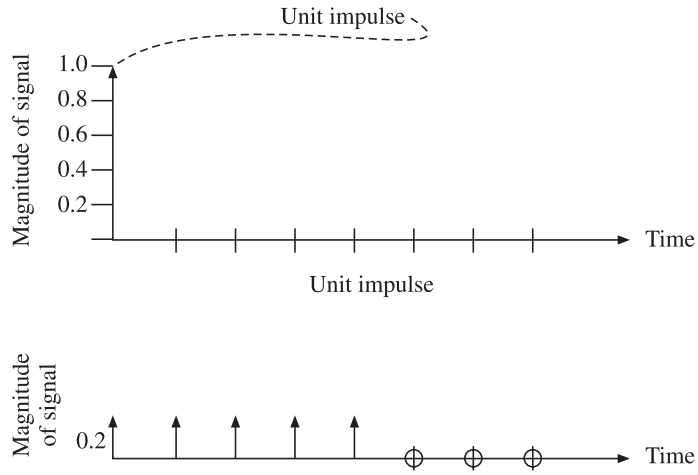


Figure 9.5 Impulse response of 5-point running average filter.

9.5.2 Frequency Response of a 2-Point Running Average FIR Filter

We have not yet proved that the running average results into low-pass filtering. So it is time now to investigate the 'frequency response' of the running average system. For the sake of simplicity, we will consider a running average system which takes average of just two samples; the present sample and the sample just previous to it. Hence, $y(n)$ can be written as:

$$y(n) = \frac{1}{2}x(n) + \frac{1}{2}x(n-1)$$

Let $x(n) = e^{j\omega_{\text{signal}}nT_s}$ be the signal with normalised amplitude (amplitude = 1) where ω_{signal} is the actual frequency of the signal and T_s is the sampling interval which is the reciprocal of the sampling frequency f_{sampling}. It will be much more convenient to express the frequency in per unit rather than actual frequency in radians, since all signal frequencies in DSP have to be limited to a value less than half the sampling frequency in hertz or to 'π' in terms of pu radian frequency. We define per-unit frequency as:

$$f_{\text{pu}} = \frac{f_{\text{signal,actual}}}{f_{\text{sampling}}} \Rightarrow f_{\text{pu,max}} = \frac{f_{\text{signal,actual,max}}}{f_{\text{sampling}}} = \frac{f_{\text{sampling}}/2}{f_{\text{sampling}}} = \frac{1}{2}$$

Hence, $\omega_{\text{pu, max}}$ can be written as:

$$\omega_{\text{pu,max}} = 2\pi\ f_{\text{pu,max}} = 2\pi\frac{1}{2} = \pi$$

Note that the following three are all equivalent:

$f_{\text{signal,max}} \rightarrow f_{\text{sampling}}/2 \rightarrow 0.5$ pu (maximum pu hertz freq.) $\rightarrow \pi$ (maximum pu radian freq.)

$$x(n) = e^{j2\pi f_{\text{signal}} n T_{\text{sampling}}} = e^{j2\pi f_{\text{signal}} n \frac{1}{f_{\text{sampling}}}} = e^{j2\pi\left(\frac{f_{\text{signal}}}{f_{\text{sampling}}}\right)n} = e^{j2\pi(f_{\text{pu}})n} = e^{j\omega_{\text{pu}} n}$$

Dropping the subscript pu from ω_{pu} but keeping in mind that ω is per unit radian frequency, we can write the expression for $x(n)$ as:

$$x(n) = e^{j\omega n}$$

where, ω is the pu radian frequency of the signal.

Since output is the average of present sample and previous samples, so we can write:

$$y(n) = \frac{1}{2}e^{j\omega n} + \frac{1}{2}e^{j\omega(n-1)}$$

$$y(n) = \frac{1}{2}e^{j\omega n} + \frac{1}{2}e^{j\omega n}\frac{1}{2}e^{-j\omega}$$

$$y(n) = \frac{1}{2}(1 + e^{-j\omega})e^{j\omega n}, \text{ but } e^{j\omega n} = x(n)$$

Hence we can write:

$$y(n) = \frac{1}{2}(1 + e^{-j\omega})\ \ x(n)$$

$$\frac{y(n)}{x(n)} = \frac{1}{2}(1 + e^{-j\omega}) = \vec{H}(\omega); \text{ which is complex}$$

Hence we can write:

$$\vec{H}(\omega) = \left|\vec{H}(\omega)\right| \angle \vec{H}(\omega)$$

$$\vec{H}(\omega) = \frac{1}{2}(1 + e^{-j\omega})$$

Taking out $e^{-j(\omega/2)}$ common, we get:

$$\vec{H}(\omega) = \frac{1}{2}(e^{j(\omega/2)} + e^{-j(\omega/2)})e^{-j(\omega/2)}$$

But we know that

$$\frac{1}{2}(e^{j(\omega/2)} + e^{-j(\omega/2)}) = \cos\left(\frac{\omega}{2}\right)$$

Hence, we can write:

$$\vec{H}(\omega) = \left[\cos\frac{\omega}{2}\right]e^{-j(\omega/2)}$$

Hence, the frequency versus magnitude response is a cosine curve and phase versus frequency is linear, with negative slope, in the fourth quadrant as shown in Figure 9.6. Thus, phase shift is always negative as the output of the filter always lags the input because of the delay inherent in the filter operation.

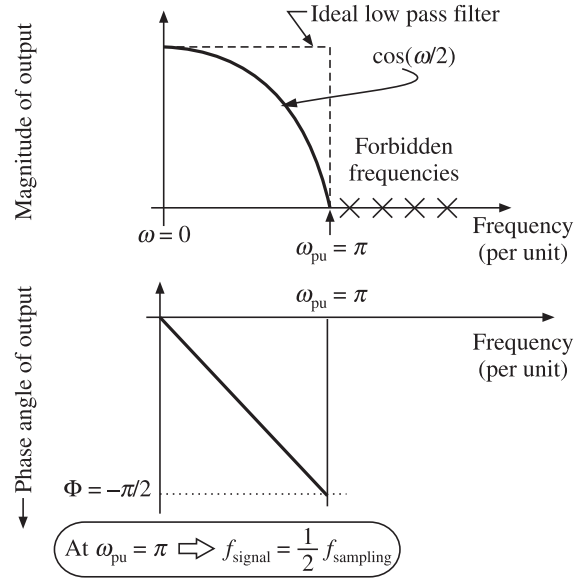


Figure 9.6 Frequency response (magnitude and phase response) of 2-point running average filter.

Consider signal samples corresponding to lowest and highest frequencies. At lowest frequency, i.e., $\omega = 0$

$$x(n) = e^{j\omega n} = [e^{j0\times0}\, e^{j0\times1}\, e^{j0\times2} e^{j0\times3} e^{j0\times4}]$$

$$= [1 \quad 1 \quad 1 \quad 1 \quad 1]$$

For the 2-point running average filter, we can find the output by running through the sequence and finding the average of every two consecutive points.

$$y(n) = \left[* \frac{(1+1)}{2} \frac{(1+1)}{2} \frac{(1+1)}{2} * \right]$$

$$y(n) = [* \;\; 1\; 1\; 1 \; *]$$

Thus, the lowest frequency signal, or DC is passed without any attenuation.

Now, let us first generate the samples of signal at highest possible per unit frequency of π.

$$x(n) = e^{j\pi n} = [e^{j\pi\times0} \quad e^{j\pi\times1} \quad e^{j\pi\times2} \quad e^{j\pi\times3} \quad e^{j\pi\times4}]$$

$$x(n) = e^{j\pi n} = [1 \quad -1 \quad 1 \quad -1 \quad 1]$$

Again, calculating the 2-point running average of the sequence, we get another sequence:

$$y(n) = \left[* \frac{(1-1)}{2} \quad \frac{(1-1)}{2} \quad \frac{(1-1)}{2} * \right]$$

Hence, we get:

$$y(n) = [* \; 0 \quad 0 \quad 0 \; *]$$

Thus, the signal at the highest (per unit) frequency is totally attenuated to zero.

9.5.3 Frequency Response of 2-Point Running Average Filter Using *Z*-transform

Let us analyse the 2-point running average filter using the Z-transform method. For the 2-point running average filter, we have:

$$y(n) = \frac{1}{2}x(n) + \frac{1}{2}x(n-1)$$

Taking Z-transform of both sides:

$$Y(z) = \frac{1}{2}X(z) + \frac{1}{2}z^{-1}X(z)$$

$$Y(z) = \frac{1}{2}(1 + z^{-1})X(z)$$

$$\frac{Y(z)}{X(z)} = H(j\omega) = \frac{1}{2}(1 + z^{-1})$$

Substituting $z = e^{j\omega}$

$$H(j\omega) = \frac{1}{2}(1 + e^{-j\omega})$$

$$H(j\omega) = \frac{1}{2}(e^{+j\omega/2}e^{-j\omega/2} + e^{-j\omega/2}\,e^{-j\omega/2})$$

$$H(j\omega) = \frac{1}{2}(e^{+j\omega/2} + e^{-j\omega/2})e^{-j\omega/2}$$

$$H(j\omega) = \frac{e^{+j\omega/2} + e^{-j\omega/2}}{2}e^{-j\omega/2}$$

$$H(j\omega) = \cos\left(\frac{\omega}{2}\right)e^{-j\omega/2}$$

$|H(\omega)| = \cos\left(\frac{\omega}{2}\right)$ is the magnitude response and $H(j\omega) = e^{-j\omega/2}$ is the phase angle response. These responses are exactly same as those derived without using Z-transform.

9.6 Frequency Response of a 3-Point Running Average Filter

Let us investigate the magnitude-frequency and phase-frequency response of a 3-point running average filter. For a 3-point running average filter we can write:

$y(n)$ = Average of present and last two samples

$$y(n) = \frac{1}{3}x(n) + \frac{1}{3}x(n-1) + \frac{1}{3}x(n-2)$$

Let $x(n) = e^{j\omega n}$
Hence, we can write $y(n)$ as:

$$y(n) = \frac{1}{3}e^{j\omega n} + \frac{1}{3}e^{j\omega(n-1)} + \frac{1}{3}e^{j\omega(n-2)}$$

$$y(n) = \frac{1}{3}e^{j\omega n} + \frac{1}{3}e^{j\omega n}e^{-j\omega} + \frac{1}{3}e^{j\omega n}e^{-j\omega 2}$$

$$y(n) = \frac{1}{3}[1 + e^{-j\omega} + e^{-j2\omega}]e^{j\omega n}$$

but $e^{j\omega n} = x(n)$; hence we can write:

$$y(n) = \frac{1}{3}[1 + e^{-j\omega} + e^{-j2\omega}]x(n)$$

$$\frac{y(n)}{x(n)} = H(j\omega) = \frac{1}{3}\left[1 + e^{-j\omega} + e^{-j2\omega}\right]$$

$$H(j\omega) = \frac{1}{3}\left[1 + \cos(\omega) - j\sin(\omega) + \cos(2\omega) - j\sin(2\omega)\right]$$

$$H(j\omega) = \frac{1}{3}\left[(1 + \cos(\omega) + \cos(2\omega)) - j(\sin(\omega) + \sin(2\omega))\right]$$

$$|H(j\omega)| = \sqrt{\frac{(1 + \cos(\omega) + \cos(2\omega))^2}{9} + \frac{(\sin(\omega) + \sin(2\omega))^2}{9}}$$

$$|H(j\omega)| = \frac{1}{3}\sqrt{(1 + \cos(\omega) + \cos(2\omega))^2 + (\sin(\omega) + \sin(2\omega))^2}$$

$$|H(j\omega)| = \frac{1}{3}\sqrt{\begin{array}{l}(1 + 2\cos(\omega) + (\cos(\omega))^2 + 2(1 + \cos(\omega)\cos(2\omega)) + (\cos(2\omega))^2 \\ \quad + (\sin(\omega) + \sin(2\omega))^2)\end{array}}$$

$$|H(j\omega)| = \frac{1}{3}\sqrt{\begin{array}{l}[1 + 2\cos(\omega) + (\cos(\omega))^2 + 2\cos(2\omega) + 2\cos(2\omega)\cos(\omega) \\ \quad + (\cos(2\omega))^2 + (\sin(\omega))^2 + 2\sin(2\omega)\sin(\omega) + (\sin(2\omega))^2]\end{array}}$$

$$|H(j\omega)| = \frac{1}{3}\sqrt{3 + 4\cos(\omega) + 2\cos(2\omega)}$$

Recall that for a 2-point running average filter the cut-off frequency was π which can be written as $2\pi/2$. One can expect that the cut-off frequency for a 3-point running average filter will be $2\pi/3$. Hence, substituting $\omega = 2\pi/3$, we get

$$|H(j)| = \frac{1}{3}\sqrt{3 + 4\cos(\omega) + 2\cos(2\omega)}$$

$$\left|H\left(j\frac{2\pi}{3}\right)\right| = \frac{1}{3}\sqrt{3 + 4\cos\left(\frac{2\pi}{3}\right) + 2\cos(2)\frac{2\pi}{3}}$$

$$\left|H\left(j\frac{2\pi}{3}\right)\right| = \frac{1}{3}\sqrt{3 + 4\frac{-1}{2} + 2\frac{-1}{2}}$$

$$\left|H\left(j\frac{2\pi}{3}\right)\right| = \frac{1}{3}\sqrt{3 - 2 - 1} = 0$$

To find $\measuredangle H(j\omega)$:

$$H(j\omega) = (1 + \cos(\omega) + \cos(2\omega)) - j(\sin(\omega) + \sin(2\omega))$$

$$\measuredangle H(j\omega) = -\tan^{-1}\left[\frac{(\sin(\omega) + \sin(2\omega))}{(1 + \cos(2\omega) + \cos(2\omega))}\right]$$

$$\measuredangle H(j\omega) = -\tan^{-1}\left[\frac{(\sin(\omega) + 2\sin(\omega)\cos(\omega))}{(1 + \cos(\omega) + (2\cos(\omega))^2 - 1)}\right]$$

$$\measuredangle H(j\omega) = -\tan^{-1}\left[\frac{(\sin(\omega) + 2\sin(\omega)\cos(\omega))}{(\cos(\omega) + (2\cos(\omega))^2)}\right]$$

$$\measuredangle H(j\omega) = -\tan^{-1}\left[\frac{(\sin(\omega)(1 + 2\cos(\omega)))}{(\cos(\omega)(1 + 2\cos(\omega)))}\right]$$

$$\measuredangle H(j\omega) = -\tan^{-1}\left[\frac{(\sin(\omega)}{\cos(\omega)}\right]$$

$$\measuredangle H(j\omega) = -\tan^{-1}\left[\tan(\omega)\right]$$

$$\measuredangle H(j\omega) = -\omega$$

Hence,

$$\measuredangle H\left(j\frac{2\pi}{3}\right) = -\frac{2\pi}{3}$$

The frequency response of the 3-point running average filter is shown in Figure 9.7. Note that at DC, i.e., $\omega = 0$;

$$\left|H(j\omega)\right| = \frac{1}{3}\sqrt{3+4\cos(\omega)+2\cos(2\omega)} = \frac{1}{3}\sqrt{3+4.1+2\ .1} = \frac{1}{3}\sqrt{9} = 1$$

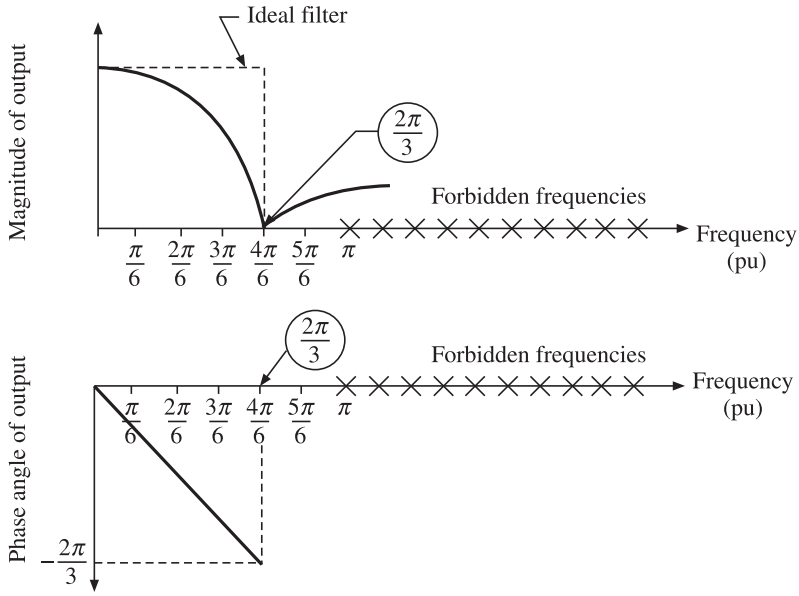


Figure 9.7 Frequency response of 3-point running average filter.

9.6.1 Frequency Response of 3-Point Running Average Filter Using *Z*-transform

Let us analyse the 3-point running average filter using the Z-transform method. For the 3 point running average filter, we have:

$$y(n) = \frac{1}{3}x(n) + \frac{1}{3}x(n-1) + \frac{1}{3}x(n-2)$$

Taking Z-transform of both sides:

$$Y(z) = \frac{1}{3}X(z) + \frac{1}{3}z^{-1}X(z) + \frac{1}{3}z^{-2}X(z)$$

$$Y(z) = \frac{1}{3}(1 + z^{-1} + z^{-2})X(z)$$

$$\frac{Y(z)}{X(z)} = H(\omega) = \frac{1}{3}(1 + z^{-1} + z^{-2})$$

Substituting $z = e^{j\omega}$

$$H(\omega) = \frac{1}{3}(1 + e^{-j\omega} + e^{-j2\omega})$$

Taking out common factor $e^{-j\omega}$, we get:

$$H(\omega) = \frac{1}{3}(e^{j\omega} + 1 + e^{-j\omega})e^{-j\omega}$$

$$H(\omega) = \frac{1}{3}\left(1 + 2\frac{e^{j\omega} + e^{-j\omega}}{2}\right)e^{-j\omega}$$

$$H(\omega) = \frac{1}{3}(1 + 2\cos(\omega))e^{-j\omega} = \text{Magnitude}\angle\text{angle}$$

Thus, magnitude of output = $\frac{1}{3}(1 + 2\cos(\omega))$ and phase angle = $-\omega$.

However, $(1 + 2\cos(\omega))$ cannot be taken as magnitude since as $\cos(\omega)$ becomes -1, the term becomes negative. Magnitude of $\frac{1}{3}(1 + 2\cos(\omega))$ can be written as:

$$|H(\omega)| = \frac{1}{3}\sqrt{(1 + 2\cos(\omega))^2}$$

$$|H(\omega)| = \frac{1}{3}\sqrt{(1 + 4\cos(\omega) + 4(\cos(\omega))^2}$$

$$|H(\omega)| = \frac{1}{3}\sqrt{(1 + 4\cos(\omega) + (2\cos(2\omega) + 2))}$$

$$|H(\omega)| = \frac{1}{3}\sqrt{((3 + 4\cos(\omega) + (2\cos(2\omega)))}$$

and $\angle H(\omega) = -\omega$

Note that we have used the following:

$$|1 + 2\cos(\omega)| = \sqrt{(1 + 2\cos(\omega))^2}$$

Let us find the frequency at which $|H(\omega)|$ is zero by solving the following:

$$1 + 2\cos(\omega) = 0$$

$$2\cos(\omega) = -1$$

$$\cos(\omega) = -\frac{1}{2}$$

$$\omega = \cos^{-1}\left(-\frac{1}{2}\right)$$

$$\omega = \frac{2\pi}{3} = 120^\circ \text{ (deg.)}$$

This is the same as derived earlier without using *Z*-transform.

9.7 Frequency Response of 4-Point Running Average Filter Using *Z*-transform

Output of the 4-point running average filter will be given by average of present sample taken with the last three samples:

$$y(n) = \frac{1}{4}x(n) + \frac{1}{4}x(n-1) + \frac{1}{4}x(n-2) + \frac{1}{4}x(n-3)$$

Taking Z-transform of both sides

$$Y(z) = \frac{1}{4}X(z) + \frac{1}{4}z^{-1}X(z) + \frac{1}{4}z^{-2}X(z) + \frac{1}{4}z^{-3}X(z)$$

$$Y(z) = \frac{1}{4}(1 + z^{-1} + z^{-2} + z^{-3})X(z)$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{4}(1 + z^{-1} + z^{-2} + z^{-3})$$

Substituting $z = e^{j\omega}$, we get

$$H(j\omega) = \frac{Y(z)}{X(z)} = \frac{1}{4}(1 + e^{-j\omega} + e^{-j2\omega} + e^{-j3\omega})$$

$$H(j\omega) = \frac{Y(z)}{X(z)} = \frac{1}{4}(1 + \cos(\omega) - j\sin(\omega) + \cos(2\omega) - j\sin(2\omega) + \cos(3\omega) - j\sin(3\omega))$$

$$H(j\omega) = \frac{Y(z)}{X(z)} = \frac{1}{4}[\{1 + \cos(\omega) + \cos(2\omega) + \cos(3\omega)\} - j\{\sin(\omega) + \sin(2\omega) + \sin(3\omega)\}]$$

$$|H(j\omega)| = \frac{1}{4}\sqrt{(1 + \cos(\omega) + \cos(2\omega) + \cos(3\omega))^2 + (\sin(\omega) + \sin(2\omega) + \sin(3\omega))^2}$$

$$\angle H(j\omega) = \tan^{-1}\left(\frac{[\sin(\omega) + \sin(2\omega) + \sin(3\omega)]}{[1 + \cos(\omega) + \cos(2\omega) + \cos(3\omega)]}\right)$$

As the expressions for magnitude and phase response of the 4-point running average digital filter is somewhat involved and not easy to sketch manually, let us write a small MATLAB script to study the magnitude and phase response as shown in Figure 9.8. The program is shown below:

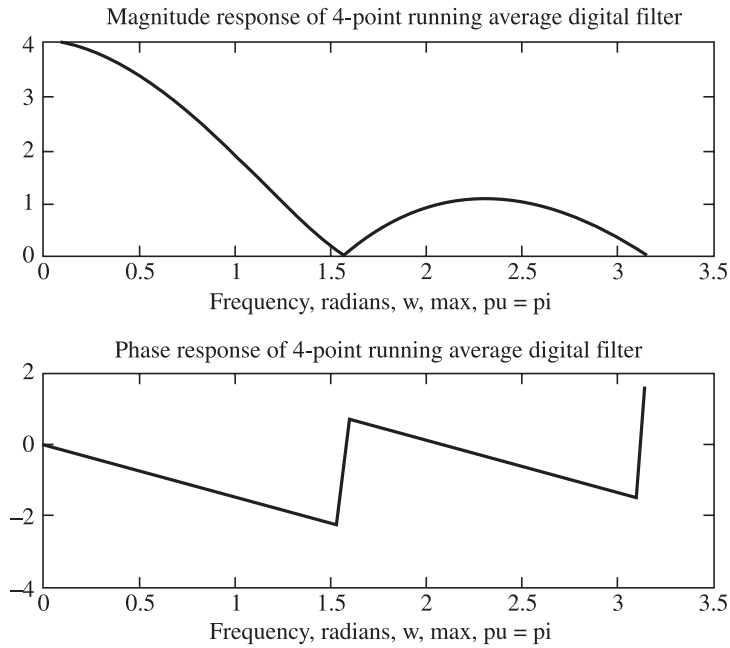


Figure 9.8 Frequency response of 4-point digital filter.

```
% ------------------------------------------------------------------
% -----  Freq  Response  of  4-Point  Running  Average  Digital  Filter
% ------------------------------------------------------------------
clear  ,  clc,clf
k=0;
step=pi/100;
for  w  =  0  :  step  :  pi
      k  =  k+1;
      freq(k)  =  w;
      x  =  1  +  cos(w)+  cos(2*w)+  cos(3*w);
      y  =        sin(w)  +  sin(2*w)  +  sin(3*w);
      H(k)  =sqrt(x*x  +  y*y);
      Ang_H(k)  =  atan2(-y,  x);

end
subplot(2,1,1)
plot(freq,H)
title('Magnitude  response  of  4-point  running  average  digital  filter')
xlabel('Frequency  ,  radians  ,   w,max,pu=pi')
subplot(2,1,2)
plot(freq,Ang_H)
title('Phase  response  of  4-point  running  average  digital  filter')
xlabel('Frequency  ,  radians  ,   w,  max  ,  pu  =  pi')
```

In the next section, we write a MATLAB program for plotting the frequency response of an *N*-point running average filter.

9.8 MATLAB Program for Plotting Frequency Response of N-Point Running Average Filter

```
%--- Frequency Response of N-point Running Average ( R.A.) Filter
% ------------------------------------
%--- For Nth   order filter
% x = 1+cos(w)+cos(2w)+...+cos(N-1)w
% y =    sin(w)+sin(2w)+...+sin(N-1)w
%    |H(jw)|    = sqrt( x*x + y*y )
%    Ang(H(jw) = atan( -y   x)

%         In this program first the formulas for x and y are built-up
%-       as strings. And then the strings are evaluated using the
%        eval( ) function.

%       Input :-   N = Number of points in the running average

%      Output :-  (1) Magnitude response of the N-point R.A. filter
%                 (2) Phase response of the N-point R.A. filter
%                 (3) Formuals leading to   |H(jw)|   and Ang(H(jw))


clear,clc,clf
%-------------------------------------------------------
%------ Enter the number of points in the R.A. filter ---
                       N = 10;
%-------------------------------------------------------

x= '1'
y='0'

w=1;
for n=1:N-1
    s=num2str(n);
    ss=strcat('+cos(',s,'','*','w',')')
    sss= strcat('+sin(',s,'','*','w',')')
x = [ x strcat(ss)]
y = [ y strcat(sss)]
end
x
y
step = pi/100;
k=0;
for w=0:step:pi
     k =k+1;
     ang(k)=w;
```

```
      xx=eval(x);
      yy=eval(y);
      magH(k)=(sqrt(xx*xx+yy*yy))/(N);
      phH(k)=atan2(-yy,xx)
end
ang_deg=(180/pi).*ang;
xmin=0;xmax=180;ymin=0;ymax=1;
xmin1=0;xmax1=180;ymin1=-pi;ymax1=pi;
subplot(2,1,1)
plot(ang_deg,magH)
axis([xmin xmax ymin ymax])
title('Magnitude Response of Running Average Filter')
xlabel(' Freuency in pu deg ( max=pi rad)')


subplot(2,1,2)
plot(ang_deg,phH)
axis([xmin1 xmax1 ymin1 ymax1])
title('Phase Response of Running Average Filter')
xlabel(' Freuency in pu deg (max =pi rad) ')
x;
y;
mag_of_H =['|H(jw)|      =''sqrt' '(' x   ')' '^2' '+ ' '('   y ')'
'^2'];
ph_of_H   =[ 'ang(H(jw) =''atan2' '(','-','(' y ,')','/',x ,')']

formula1=num2str(mag_of_H)
formula2=num2str(ph_of_H)
fprintf(' |H(jw)|         = %s \n'       ,formula1   )
fprintf(' Angle(H(jw)) = %s \n',formula2   )


text(1,10,formula1)
text(1,0.8,formula2)
text(1,11,'No. of points=')
text(40,11,num2str(N))
% subplot(2,1,2)
% text(1,
```

$$x = 1 + \cos(1*w) + \cos(2*w) + \cos(3*w) + \cos(4*w) + \cos(5*w) + \cos(6*w) + \cos(7*w) + \cos(8*w) + \cos(9*w)$$

$$y = 0 + \sin(1*w) + \sin(2*w) + \sin(3*w) + \sin(4*w) + \sin(5*w) + \sin(6*w) + \sin(7*w) + \sin(8*w) + \sin(9*w)$$

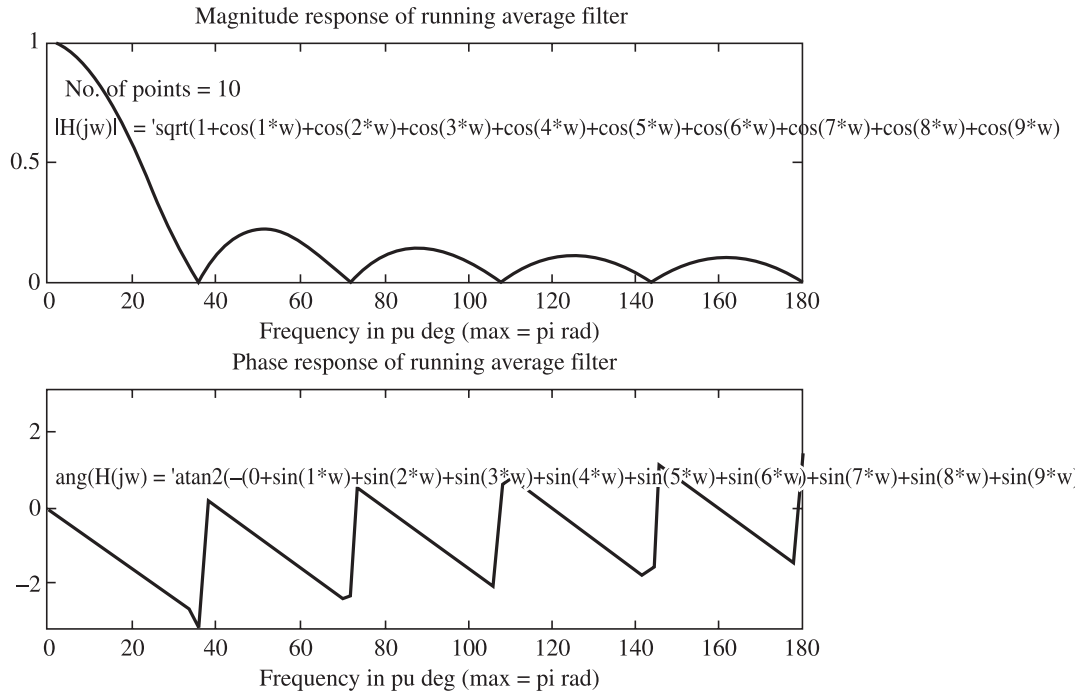


Figure 9.9 Frequency response of 10-point running average filter.

9.9 Introduction to Infinite Impulse Response (IIR) Filters

The infinite impulse response (IIR) filters are given by the following expression:

$$y(n) = \sum_{k=0}^{Q} b_k x(n-k) - \sum_{k=1}^{P} a_k y(n-k)$$

The above expression can be interpreted as:

Present output = Weighted sum of present input, past 'Q' input samples and past 'P' output samples

Thus, there is a feedback from output to input. We can construct a signal flow graph based on the above expression as shown in Figure 9.10. The weights denoted by b_k and a_k are called the coefficients of the filter. The *filter coefficients* entirely decide the filter behaviour; hence design of filter means finding 'b_k' and 'a_k' coefficients which will give the desire response.

We need to first answer the question: Why is it called ***infinite*** impulse response (IIR)?

Let us first make it clear that IIR stands for Infinite-time Impulse Response. Thus, the response to an impulse signal, at least theoretically, exists for infinite-time. The impulse signal sequence in the discrete domain is a finite signal of one-sample duration, which is preceded and succeeded by an infinite string of zeros as shown below for a unit-impulse:

$$[\text{from } \infty \rightarrow 0\,0\,0\,0\,1\,0\,0\,0\,0\,0 \rightarrow \text{to } \infty]$$

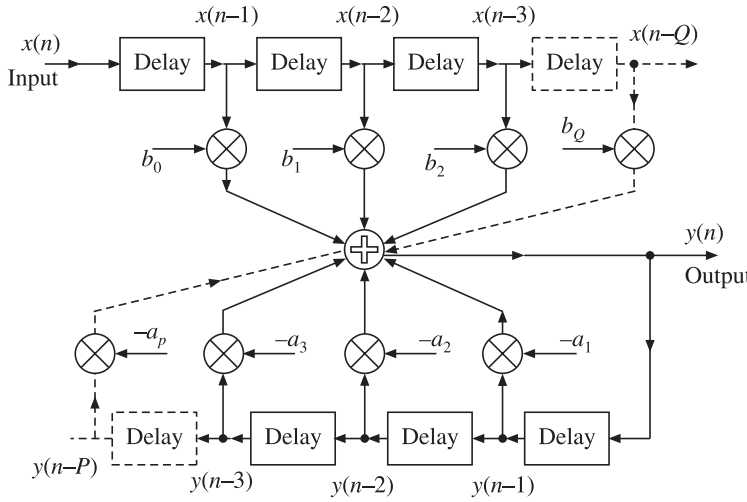


Figure 9.10 IIR filter block diagram.

The response of the IIR filter to the impulse will be of the form:

$$[\text{from } \infty \rightarrow 0\,0\,0\,0\; y_1\, y_2\, y_3\, y_4\, y_5 \ldots\ldots\ldots \rightarrow \text{to } \infty]$$

where y_1, y_2,… are all finite values. The basic cause for continuation of the impulse response for infinite time is the feedback that exists in the IIR filter. This feedback causes the signal to get recycled again and again even after the input has vanished. Note also that the basic reason for instability of IIR filters also stems from the feedback. FIR filters do not have feedback. Hence, impulse response of FIR filter lasts for a finite time and there is no possibility of the FIR filter becoming unstable.

We repeat the time domain expression for output of the IIR filter below, for convenience.

$$y(n) = \sum_{k=0}^{Q} b_k x(n-k) - \sum_{k=1}^{P} a_k y(n-k)$$

The block-diagram of Figure 9.10 can be used to directly implement an IIR filter in either hardware or software form. Hence, this method of implementation or the topology of the IIR filter is called '*direct form IIR filter*'.

Taking the Z-transform of the above equation, gives us:

$$Y(z) = \sum_{k=0}^{Q} b_k z^{-k} X(z) - \sum_{k=1}^{P} a_k z^{-k} Y(z)$$

Rearranging further gives:

$$Y(z) + Y(z)\sum_{k=1}^{P} a_k z^{-k} = X(z)\sum_{k=0}^{Q} b_k z^{-k}$$

$$Y(z)\left(1 + \sum_{k=1}^{P} a_k z^{-k}\right) = X(z)\sum_{k=0}^{Q} b_k z^{-k}$$

Thus, we get the expression for $H(z)$, the transfer function of the filter in frequency domain:

$$\frac{Y(z)}{X(z)} = H(z) = \frac{\sum_{k=0}^{Q} b_k z^{-k}}{\left(1 + \sum_{k=1}^{P} a_k z^{-k}\right)}$$

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + \cdots + b_Q z^{-Q}}{(1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + \cdots + a_P z^{-P})}$$

$$H(z) = \frac{\text{Polynomial of order } Q \text{ in } z}{\text{Polynomial of order } P \text{ in } z}$$

$$H(z) = \frac{\text{Factored polynomial of order } Q \text{ in } z}{\text{Factored polynomial of order } P \text{ in } z}$$

$$H(z) = \frac{(z - q_1)(z - q_2)(z - q_3)\ldots(z - q_Q)}{(z - p_1)(z - p_2)(z - p_3)\ldots(z - p_P)}$$

Note that we can express the transfer function in various ways. For example we can write:

$$H(z) = H_1(z) \cdot H_2(z) \ldots H_k(z)$$

The above form can be implemented using a cascade of lower order transfer functions.

Alternatively we can express $H(z)$ as:

$$H(z) = 1 + \sum_{i=1}^{k} H_I(z) + \sum_{i=1}^{l} H_{II}(z)$$

where $H_I(z)$ are 'k' first order transfer functions and $H_{II}(z)$ are 'l' second order transfer functions. This type of representation leads to parallel form of IIR filter realisation. A question that needs to be answered at this stage is: What is the motivation for alternate forms of implementation when we can implement the filter easily in the 'direct form'? The answer to this question is not intuitively obvious. The answer lies in the error analysis of various topologies and their sensitivity to quantization of coefficient values. It can be shown that direct form is not always the best, from the above consideration and cascade and parallel form prove to be better under certain situations. Hence, large numbers of topologies have been developed in the literature on digital filtering.

Review Questions

1. FIR stands for
   (a) Finite magnitude impulse response
   (b) Finite duration impulse response
   (c) Finite duration, finite magnitude impulse response
   (d) Finite magnitude, infinite duration impulse response.

2. FIR filter makes use of
   (a) Samples of past input only
   (b) Samples of past output only
   (c) Samples of past input as well as samples of past output
   (d) Samples of past input along with present sample of input.

3. IIR filter makes use of
   (a) Samples of past input only
   (b) Samples of past output only
   (c) Present sample, samples of past input as well as samples of past output
   (d) Samples of past input along with present sample of input.

4. What are the basic hardware elements required for implementation of FIR or IIR filters?

5. Discuss why FIR has finite-time and IIR has infinite-time impulse response.

6. Why IIR filters are referred to as recursive filters?

7. Why FIR filters are called non-recursive?

8. Show how an FIR filter can be implemented recursively.

9. FIR filters are referred to as filters with only zeros and a Qth order pole at the origin. Justify/Explain.

10. IIR filters have both poles as well as zeros. Explain.

11. FIR filters are free from possibility of becoming unstable. Explain.

12. Explain why IIR filters need to be carefully designed in order to avoid instability.

13. Explain intuitively how running average can have low-pass filtering effect.

14. By corollary, which mathematical operation will cause high-pass filtering?

15. What will be the effect of increasing the number of points in the running average?

16. Analyse a 4-point running average filter and find is cut-off frequency and phase shift at cut-off frequency.

17. Perform the analysis of 2-point and 3-point running average filters using Z-transform.

18. State whether the running average filter is of FIR type or IIR type.

19. Explain the concept of per unit frequency.

20. What is the maximum per unit radian frequency allowable in digital signal processing?

21. What is the maximum per unit hertz frequency allowable in digital signal processing?

22. What happens if a (pu) frequency greater than 'π' appears at the input of a digital processing system?

23. Write the equation in time domain for output of a 2-point running average filter.

24. Write the equation in time domain for output of a 3-point running average filter.

25. Write the equation in time domain for output of a *n*-point running average filter.

26. Convert above equations into Z-domain equations.

27. Convert above equations into transfer functions of respective filters.

28. Write a MATLAB script to plot the magnitude and frequency response of an '*n*-point' running average filter.

10

Digital Filter Design

10.1 Filter Specifications

The design of filters starts with specifications of the desired filter in terms of its:

(a) Magnitude Vs frequency response
(b) Phase Vs frequency response

In terms of magnitude Vs frequency response, the main types of the filters are:

(i) Low-pass (LP) filter
(ii) High-pass (HP) filter
(iii) Band-pass (BP) filter
(iv) Band-stop (BS) filter

FIR filters can be designed to exhibit linear phase response while linear phase response is not possible with IIR filters. Advantage of linear phase response is that it does not distort the signal waveform. The phase response of the digital filters is not specified as rigorously as the magnitude response.

Figure 10.1 shows ideal magnitude response of various filters.

Detailed filter specifications of a practical low-pass filter are shown in Figure 10.2.

The specifications of interest as shown in Figure 10.2 are pass band deviation δ_p, stop band deviation δ_s, pass-band edge frequency f_p and stop band edge frequency f_s. The peak pass band ripple is given as $A_{\text{pass}} = 20 \log(1 + \delta_p)$ dB and the minimum stop band attenuation is given as $A_{\text{stop}} = -20\log(\delta_s)$.

10.2 Relation between Linear Phase and Symmetry of Filter Coefficients for FIR Filter [52]

Before we delve into design procedure for FIR filters, the following points need to be kept in mind:

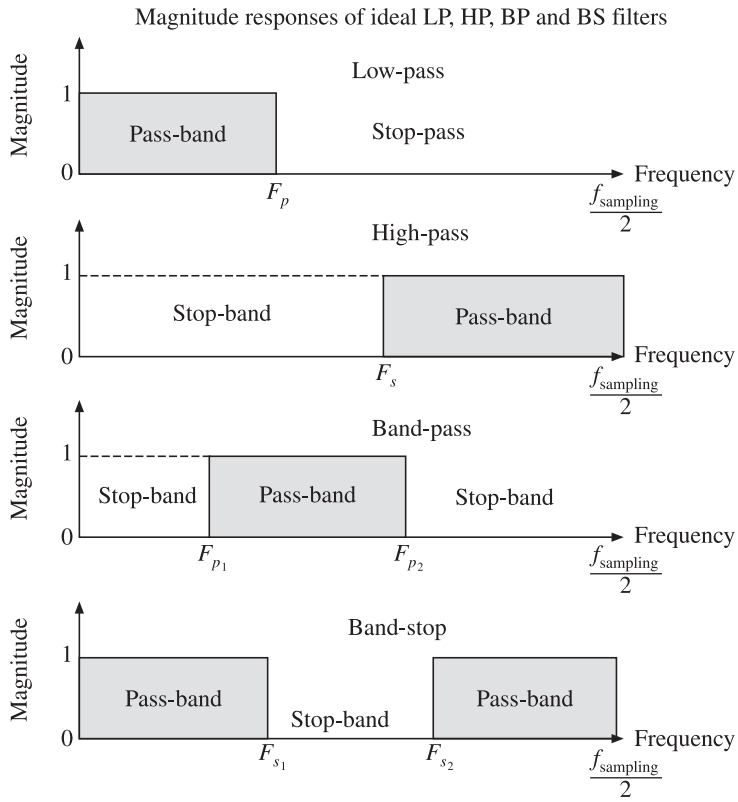


Figure 10.1 Ideal magnitude response of LP, HP, BP and BS filters.

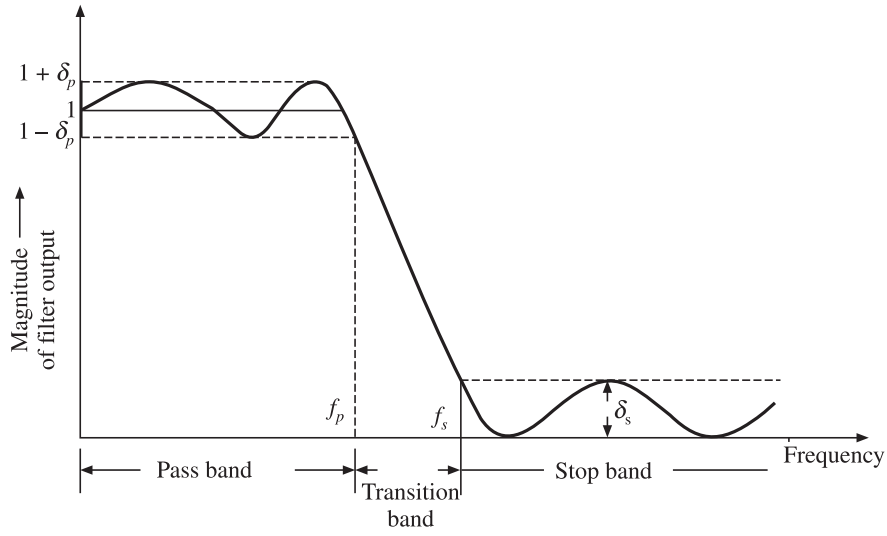


Figure 10.2 Detailed specifications of LP digital filter.

1. Infinite combinations of filter coefficients are possible; theoretically making it possible to design an infinite variety of filters.
2. However, since we are mostly interested in designing FIR filters with linear phase, certain symmetry constraints get imposed on the FIR filter coefficients.
3. It can be shown that constraint of linear phase demands that filter coefficients exhibit either even or odd symmetry.
4. Depending upon coefficient symmetry (even/odd) and filter order (even/odd), there are four types of filters labelled as type-I, type-II, type-III and type-IV as shown in Table 10.1 and Figure 10.3 .

Table 10.1 Four types of FIR filters

Filter Type	Impulse Response Symmetry	Filter Order M	Phase Offset	Amplitude Response	End-point Zeros	Candidate Filters
Type-I	Even	Even	Zero	Even	None	All
Type-II	Even	Odd	0	Even	$z = -1$	LP, BP
Type-III	Odd	Even	$\pi/2$	Odd	$z = \pm 1$	BP
Type-IV	Odd	Odd	$\pi/2$	Odd	$z = 1$	HP, BP

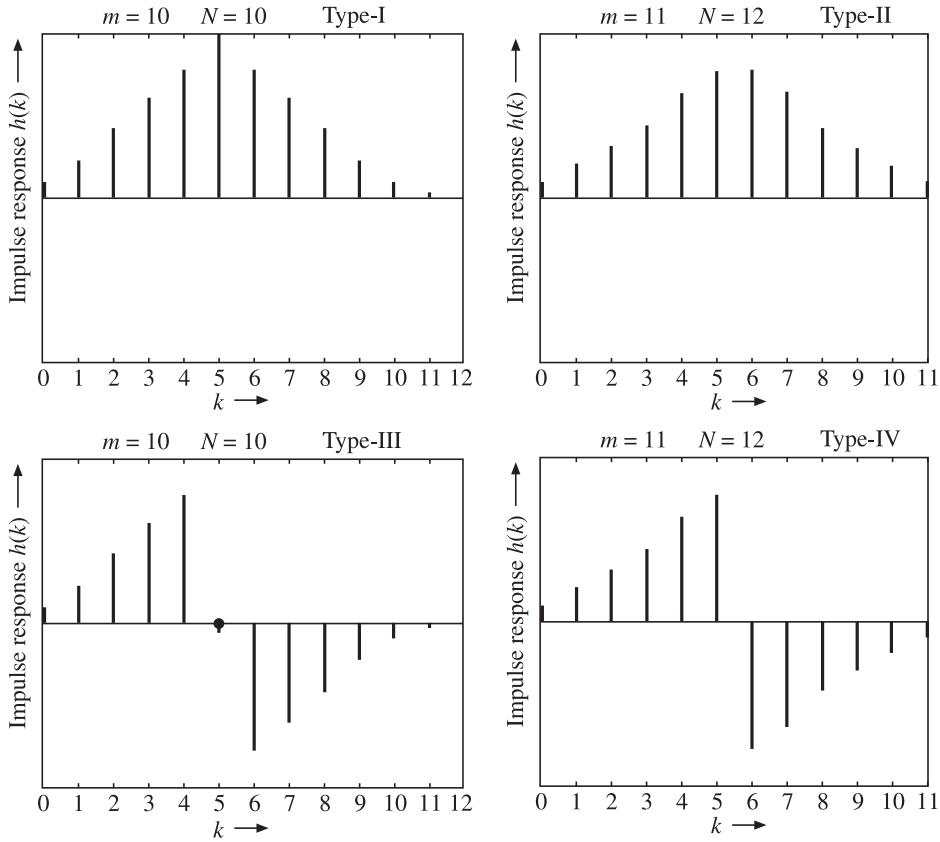


Figure 10.3 Impulse response symmetry patterns for type I, II, III and IV FIR filters.

10.3 Design of FIR Filter Using Frequency Sampling Method [13, 22, 52]

Let there be N samples of the entire frequency range from 0 to f_{sampling}, then the ith frequency sample will be located at a frequency of $f_i = i(f_{\text{sampling}}/N)$ with 'i' ranging from 0 to (N–1). The relation between impulse response and frequency response is as follows:
Frequency response = DFT (Impulse response)
Hence, taking inverse DFT, i.e., IDFT of both sides of above equation:

$$\text{IDFT(Frequency response)} = \text{IDFT (DFT (Impulse response))}$$
$$= \text{Impulse response} = \text{Coefficients of (FIR) filter}$$

Hence, the impulse response $h(k)$ can be written as:

$h(k)$ = IDFT($H(f_i)$); k = 0 to N–1; where $H(f_i)$ denotes the desired frequency response of the filter. Further, since we always want the filter to have linear phase, we have to place certain constraints on the symmetry that $h(k)$ should exhibit. Thus even before we find actual values of $h(k)$, we know apriori that it will have certain symmetry. Let us assume even symmetry about the mid point $k = m/2$. The frequency response of this type of filter can be written as:

$$H(f) = A_r(f)e^{-j\pi mfT}$$

where the magnitude response $A_r(f)$ is a real even function that specifies the desired magnitude response, i.e., $|A_r(f)| = A(f)$.

Hence, we can write the expression for IDFT as:

$$h(k) = \frac{1}{N}\sum_{i=0}^{N-1} H(f_i)e^{\frac{j2\pi ik}{N}}$$

> ***Note:*** The differences between DFT and IDFT formulae are as follows:
> 1. $1/N$ factor is included in IDFT formula. No such factor is in DFT formula.
> 2. Sign of exponent is positive in IDFT and negative in DFT formula.

We can write $h(k)$ as:

$$h(k) = \frac{1}{N}\sum_{i=0}^{N-1} A_r(f_i)e^{-j\pi f_i T}e^{\frac{j2\pi ik}{N}}$$

which can be written as:

$$h(k) = \frac{1}{N}\sum_{i=0}^{N-1} A_r(f_i)e^{\frac{-j\pi mif_sT}{N}}e^{\frac{j2\pi ik}{N}}$$

Where $f_s = f_{\text{sampling}} = 1/T$; hence $f_sT = 1$. Thus, we get:

$$h(k) = \frac{1}{N}\sum_{i=0}^{N-1} A_r(f_i)e^{\frac{-j\pi mi}{N}}e^{\frac{j2\pi ik}{N}}$$

$$h(k) = \frac{1}{N}\sum_{i=0}^{N-1} A_r(f_i) e^{\frac{j2\pi ik - j\pi mi}{N}}$$

$$h(k) = \frac{1}{N}\sum_{i=0}^{N-1} A_r(f_i) e^{\frac{j2\pi i(k-0.5m)}{N}}$$

$$h(k) = \frac{1}{N}\sum_{i=0}^{N-1} A_r(f_i)\left\{\cos\left(\frac{2\pi i(k-0.5m)}{N}\right) + j\sin\left(\frac{2\pi i(k-0.5m)}{N}\right)\right\}$$

We are designing a filter with all $h(k)$ values real. Hence, the j sin() terms cancel out, giving:

$$h(k) = \frac{1}{N}\sum_{i=0}^{N-1} A_r(f_i)\cos\left(\frac{2\pi i(k-0.5m)}{N}\right)$$

Further, treating the '$i = 0$' term separately, we can write:

$$h(k) = \frac{A_r(0)}{N} + \frac{1}{N}\sum_{i=1}^{N-1} A_r(f_i)\cos\left(\frac{2\pi i\ (k-0.5m)}{N}\right)$$

Using the symmetry property of DFT, i.e., the contributions of the ith and $(N–i)$th terms are the same. Thus, we can sum half of the terms and double the result. Further recalling that $b(k) = h(k)$ for FIR filters and noting that 'm' the order of the filter is given by:

$m = N - 1$; where 'N' is the number of time/frequency domain samples

$N = m + 1$; where 'm' is the order of the filter

We get the final expression for filter coefficients as:

$$_{k=0}^{k=m-1}[b_k] = \frac{A_r(0)}{(m+1)} + \frac{2}{(m+1)}\sum_{i=1}^{\text{floor}\left(\frac{m}{2}\right)} A_r(f_i)\cos\left(\frac{2\pi i\left(k-0.5m\right)}{m+1}\right)$$

Note that for type-I filter, the order of the filter 'm' is even, in which case floor (m/2) = m/2. But for type-II filter, 'm' is odd, hence m/2 will not be an integer but the floor() function of MATLAB will generate an integer, thus allowing the summation.

10.3.1 Numerical Problem on Design of FIR Filter Using Frequency Sampling Method

Let us illustrate the design of FIR filter using the frequency sampling method with the help of a numerical problem.

EXAMPLE 10.1 Design a low-pass FIR filter to mimic an ideal low-pass filter with a cut-off frequency of $f_{\text{sampling}}/4$. Assume order of the filter to be 20.

Solution Let us design a type-I filter. We have order of the filter $m = 20$. We have $m = N - 1$; where N = number of frequency domain samples. Hence $N = m + 1 = 20 + 1 = 21$. Thus, number of frequency domain samples = 21. Let us first sketch the desired frequency response of the filter as shown in Figure 10.4.

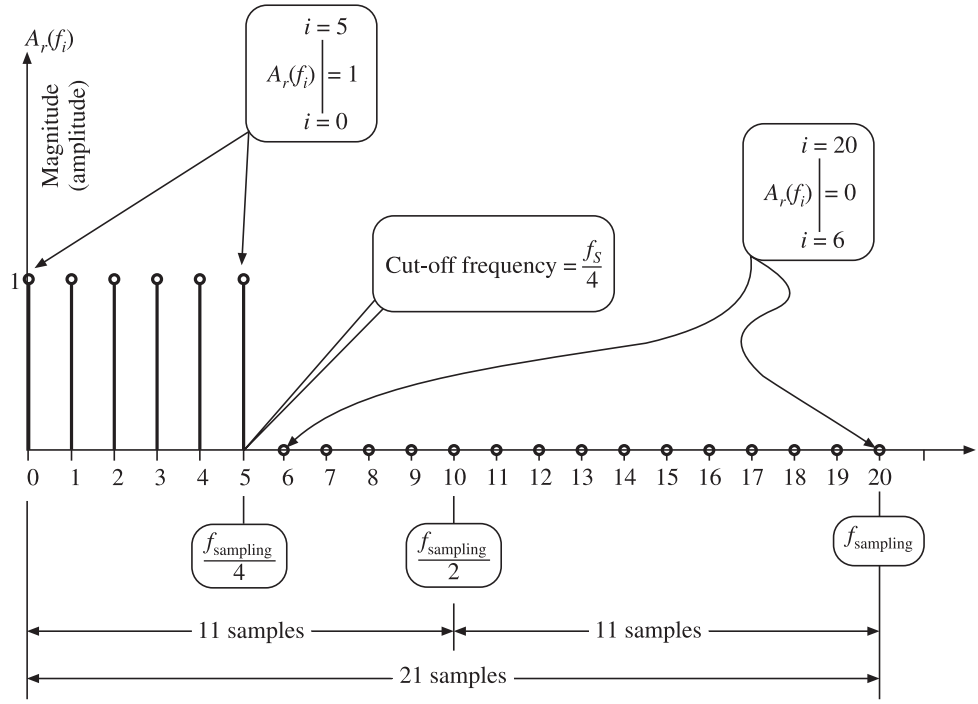


Figure 10.4 Frequency response of the desired low-pass filter.

For constructing Figure 10.4, following steps may be followed:

1. Draw 21 equally spaced points on the x-axis which represents frequency.
2. Mark this entire range from 0 to 20 as $f_{sampling}$.
3. Mark half of this range, at the 10th sample point as $f_{sampling}/2$.
4. Mark half of the above, at 5th sample point as $f_{sampling}/4$.

Now, given the above frequency response we are interested in finding out the $b(k)$ coefficients of the FIR filter in the block diagram of the filter shown in Figure 10.5.

Thus, we are given information in frequency domain and are required to transform it to time domain. Hence, inverse Fourier transform is called for as shown below:

Frequency domain $\rightarrow$ Time domain : Inverse Fourier transform
Time domain $\rightarrow$ Frequency domain : Fourier transform

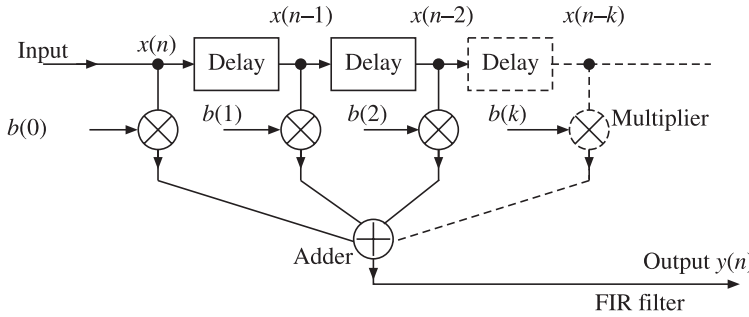


Figure 10.5 FIR filter block diagram.

We can write, for the $b(k)$ coefficients as:

$$[b(k)] = \frac{A_r(0)}{(m+1)} + \frac{2}{(m+1)} \sum_{i=1}^{\text{floor}\left(\frac{m}{2}\right)} A_r(f_i)\cos\left(\frac{2\pi i\ (k-0.5m)}{m+1}\right)$$

From the sketch of the desired frequency response we can see that:

$$A_r(f_i)\begin{cases} i=5 \\ i=0 \end{cases} = 1 \quad \text{and} \quad A_r(f_i)\begin{cases} i=20 \\ i=6 \end{cases} = 0$$

Further, we have $m = 20$, hence $0.5m = 10$. Thus, we can write the above expression as:

$$b(k) = \frac{1}{21} + \frac{2}{21}\sum_{i=1}^{10} A_r(f_i)\cos\left(\frac{2\pi i(k-10)}{21}\right)$$

Let us write a small MATLAB script to compute the filter coefficients and verify that the coefficients indeed implement the desired filter by finding the FFT of the coeffiients:

```
clear , clc
%-----------------------------------------
%---------Order of filter -----------------
m=20;
%-----------------------------------------
for k = 0 : m
      fpu(k+1)= k/m;
      b(k+1)=(1/(m+1));
    for ii =1:5
      b(k+1)=b(k+1) + (2/(m+1))* cos( 2 *pi*ii*(k-10)/(m+1))   ;
    end
```

```
    fprintf(‘  b(%d)=  %f  \n’,k,b(k+1))
end
%-------------------------------------------------------------------
subplot(2,1,1)
  stem(b)
  xlabel(‘sample  number’)
  ylabel(‘h(n)=  Impulse  response  =b(k)’)
  title(‘Design  of  low  pass  FIR  filter  with  fc  =  f/4    ‘)
subplot(2,1,2)
  stem(fpu,abs((fft(b))))   ← For checking the frequency response from
the  computed  coeffs.
  xlabel(‘Freuency  in  pu  ‘)
  ylabel(‘Magnitude  of  output  A(f)’)
```

b(0) = –0.032480	b(1) = 0.038188	b(2) = 0.028817
b(3) = –0.047619	b(4) = –0.026427	b(5) = 0.065171
b(6) = 0.024916	b(7) = –0.106999	b(8) = –0.024078
b(9) = 0.318607	b(10) = 0.523810	b(11) = 0.318607
b(12) = –0.024078	b(13) = –0.106999	b(14) = 0.024916
b(15) = 0.065171	b(16) = –0.026427	b(17) = –0.047619
b(18) = 0.028817	b(19) = 0.038188	b(20) = –0.032480

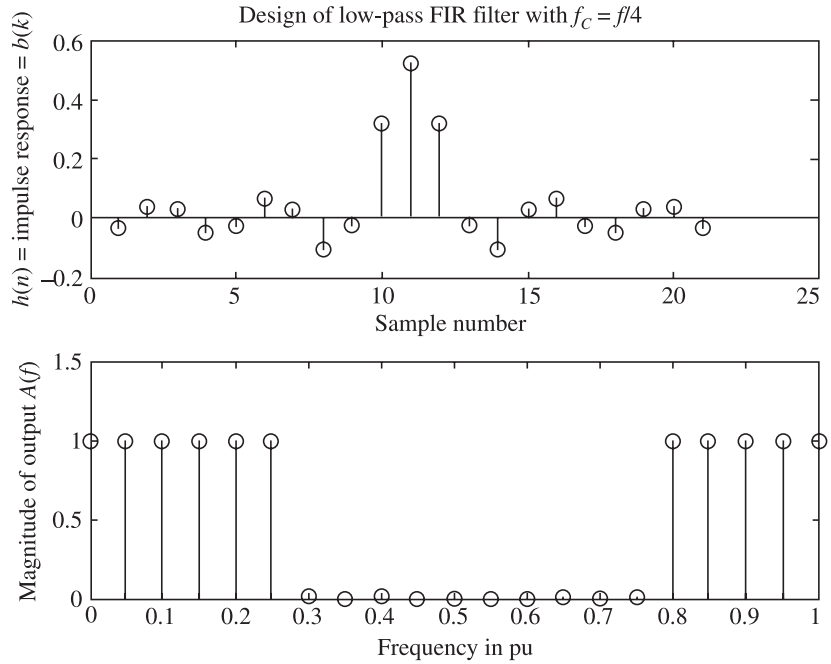


Figure 10.6 Design of FIR filter by frequency sampling method.

10.4 Bilinear Transformation

IIR filters are based upon analog filters. We start with an analog filter and transform its '*s*' domain transfer function into the '*z*' domain transfer function of the corresponding digital filter. Bilinear transformation is used for converting '*s*' domain transfer function into its '*z*' domain equivalent. Hence, we will first derive the bilinear transform.

We have $z = e^{sT}$ which can be written as:

$$z = e^{sT/2}e^{sT/2} = \frac{e^{sT/2}}{e^{-sT/2}} = \frac{1+\left(\frac{sT}{2}\right)+\frac{\left(\frac{sT}{2}\right)^2}{2!}+\cdots}{1-\left(\frac{sT}{2}\right)+\frac{\left(\frac{sT}{2}\right)^2}{2!}-\cdots}$$

If we neglect higher order terms, we can write:

$$z \cong \frac{1+(sT/2)}{1-(sT/2)} = \frac{2+sT}{2-sT}$$

Hence, we can write the following transformation as:

$$s = \frac{2}{T}\frac{(z-1)}{(z+1)}$$

10.4.1 Design of IIR Filter Using Bilinear Transformation

If we have an analog filter prototype whose '*s*' domain transfer function is known, we can transform it to '*z*' domain using bilinear transform. We can write time domain difference equation for the filter from the '*z*' domain transfer function very easily. Thus, we can implement an IIR filter. However, there is one more factor to be taken into account. As we migrate from '*s*' domain to '*z*' domain, the frequency gets transformed in a very non-linear fashion. The basic reason for this is that in the analog domain, frequency range is from zero to infinity, while, in the digital domain, the frequency range is from zero Hz to ($f_{sampling}$/2)Hz or on a per unit basis from zero to π per unit. Thus, when we do the bilinear transform, the entire frequency range from zero to infinity (Hz) is compressed into a range from zero Hz to ($f_{sampling}$/2)Hz or zero to π(pu Hz). This is known as *warping of the frequency*.

10.4.2 Warping of Frequency

To distinguish between frequency in the analog and digital domains, we use the following symbols:

Ω : Analog frequency
ω : Digital frequency

The relation between analog and digital frequencies can be derived as follows:
Noting that: $s = j\Omega$ and $z = e^{j\omega}$

We have the bilinear transform given by:

$$s = \frac{2}{T}\frac{(z-1)}{(z+1)} = \frac{2}{T}\frac{(1-z^{-1})}{(1+z^{-1})}$$

Hence, we can write:

$$s = j\Omega = \frac{2}{T}\frac{(1-e^{-j\omega})}{(1+e^{-j\omega})}$$

which can be simplified as follows:

$$j\Omega = \frac{2}{T}\frac{e^{-j\omega/2}}{e^{-j\omega/2}}\frac{(e^{j\omega/2}-e^{-j\omega/2})}{(e^{j\omega/2}+e^{-j\omega/2})}$$

$$j\Omega = \frac{2}{T}\frac{(e^{j\omega/2}-e^{-j\omega/2})}{(e^{j\omega/2}+e^{-j\omega/2})}$$

$$\Omega = \frac{2}{T}\frac{(e^{j\omega/2}-e^{-j\omega/2})}{2j}\frac{1}{(e^{j\omega/2}+e^{-j\omega/2})/2}$$

$$\Omega = \frac{2}{T}\frac{\sin\left(\frac{\omega}{2}\right)}{\cos\left(\frac{\omega}{2}\right)}$$

$$\Omega = \frac{2}{T}\tan\left(\frac{\omega}{2}\right)$$

$$\omega = 2\tan^{-1}\left(\frac{\Omega T}{2}\right)$$

$$\Omega_{\max} = \frac{2}{T}\tan\left(\frac{\omega_{\max}}{2}\right)$$

As digital frequency 'ω' reaches its maximum value, i.e., $\omega_{\max} = \pi$, it gets translated as:

$$\Omega_{\max} = \frac{2}{T}\tan\left(\frac{\pi}{2}\right) = \text{infinity}$$

Thus, the entire range of analog frequencies from 0 to ∞ gets compressed into digital frequencies from 0 to π; giving rise to the 'warping' effect as shown in Figure 10.7.

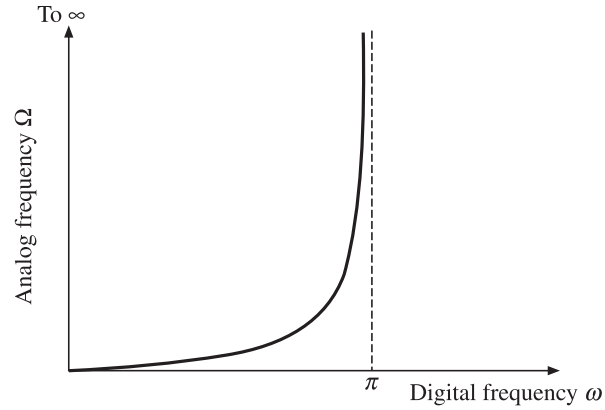


Figure 10.7 Warping of frequency during bilinear transformation.

10.4.3 Pre-Warping of Frequency During Design

We can defeat the warping by pre-compensating the frequency as follows:

The bilinear transform has the effect if we design at an analog frequency of Ω, we get effect at the digital frequency of $2\tan^{-1}(\Omega T/2)$. Hence, if we want effect at frequency of Ω, we should design at pre-warped frequency of $(2/T)\tan(\Omega/2)$. This will give effect at Ω since $2\tan^{-1}[(2/T)\tan(\Omega/2)\cdot(T/2)] = \Omega$.

To summarize:

(i) Pre-warped frequency at which filter is designed $= \dfrac{2}{T}\tan\left(\dfrac{\Omega}{2}\right)$

(ii) Effective (digital domain) frequency at which effect will be obtained = Ω

10.4.4 Step by Step Design of IIR Filter Using Bilinear Transformation

We can summarise the steps in the design of IIR filter using bilinear transform as follows:

1. Start with analog prototype of the filter to be designed. Widely used forms of analog filters are Butterworth, Chebyshev, Elliptical etc. Analog filter data books contain details about these filters.
2. Write down the 's' domain transfer function of the prototype analog filter.
3. Pre-warp the analog cut-off frequency Ω_C to $(2/T)\tan(\Omega_C/2)$.
4. Re-write the 's' domain transfer function of the prototype filter in terms of the pre-warped cut-off frequency.
5. Convert from 's' domain transfer function to 'z' domain transfer function using bilinear transformation, i.e., substitute

$$s = \frac{2}{T}\frac{(z-1)}{(z+1)} = \frac{2}{T}\frac{(1-z^{-1})}{(1+z^{-1})}$$

6. Express the $H(z)$ in terms of z^{-1} so that implementation in terms of delay elements becomes obvious.
7. Draw block diagram of the IIR filter in terms of multipliers, adders and delay elements. The filter can now be implemented either in software or hardware.

10.4.5 Review of Formulas Derived

We have derived the formulas as shown in Table 10.2.

Table 10.2

Bilinear transform	$s = \frac{2}{T}\frac{(z-1)}{(z+1)} = \frac{2}{T}\frac{(1-z^{-1})}{(1+z^{-1})}$
$\Omega \xrightarrow{\text{Analog frequency to digital frequency}} \omega$	$\omega = 2\ \tan^{-1}\left(\frac{\Omega T}{2}\right)$
$\omega \xrightarrow{\text{Digital frequency to analog frequency}} \Omega$	$\Omega = \frac{2}{T}\tan\left(\frac{\omega}{2}\right)$
Pre-warping	$\Omega \xrightarrow{\text{Pre-warping}} \frac{2}{T}\tan\left(\frac{\Omega}{2}\right)$

10.4.6 Numerical Problem on Design of IIR Filter

EXAMPLE 10.2 Design an IIR digital filter based on first order low-pass Butterworth filter prototype with 3 dB cut-off frequency $\omega_c = 0.25\ \pi$; by using bilinear transformation.

Solution First order low-pass Butterworth filter with cut-off frequency of Ω_c has the 's' domain transfer function $H_a(s) = 1(1 + s/\Omega_c)$.

As explained earlier if we design the filter to have a cut-off frequency of Ω_c, because of bilinear transform the actual cut-off frequency will be $(2/T)\tan^{-1}(\Omega/2)$.

Hence, we have pre-warp Ω_c to $(2/T)\tan(\Omega_c/2)$. Hence,

$$\Omega_c(\text{pre-warped}) = \frac{2}{T}\tan\left(\frac{0.25\pi}{2}\right) = \frac{2}{T}\tan\left(\frac{\pi}{8}\right) = \frac{2}{T}0.414 = \frac{0.828}{T}$$

Thus, the transfer function of the prototype filter which was $H_a(s) = 1(1 + s/\Omega_c)$ becomes:

$$H_a(s) = \frac{1}{1 + \frac{s}{\frac{0.828}{T}}} = \frac{1}{1 + \frac{sT}{0.828}}$$

Applying bilinear transformation, i.e., substituting, $s = \frac{2}{T}\left(\frac{z-1}{z+1}\right)$

$H(z) = \frac{1}{1+\left[\frac{2}{T}\frac{(z-1)}{(z+1)}\right]\frac{T}{0.828}}$; which can be simplified as shown in the following steps:

$$H(z) = \frac{1}{1+\frac{2(z-1)}{0.828(z+1)}}$$

$$= \frac{1}{\frac{0.828(z+1)+2(z-1)}{0.828(z+1)}}$$

$$= \frac{0.828(z+1)}{0.828z+0.828+2z-2}$$

$$= \frac{0.828(z+1)}{2.828z-1.172}$$

$$= \frac{0.828(z+1)}{2.828(z-1.172/2.828)}$$

$$= \frac{0.2927(z+1)}{(z-0.414)}$$

$$= \frac{0.2927(1+z^{-1})z}{(1-0.414z^{-1})z}$$

$$= \frac{0.2927(1+z^{-1})}{(1-0.414z^{-1})}$$

Now, let us think about implementing the filter. We have:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{0.2927(1+z^{-1})}{(1-0.414z^{-1})}$$

$$Y(z)(1-0.414z^{-1}) = X(z)0.2927(1+z^{-1})$$

$$Y(z) - 0.414z^{-1}Y(z) = 0.2927X(z) + 0.2927z^{-1}X(z)$$

Using the following inverse *Z*-transformations; where inverse *Z*-transform is denoted by Z^{-1}, i.e.,

$$Z^{-1}[z^{-1}\,Y(z)] = y(n-1);\ Z^{-1}[Y(z)] = y(n);\ Z^{-1}\,[z^{-1}X(z)] = x(n-1);\ Z^{-1}\,[X(z)] = x(n).$$

Thus, we can transform the above '*z*' domain equation into a time domain difference equation as shown below:

$$y(n) - 0.414y(n-1) = 0.2927x(n) + 0.2927x(n-1)$$

$$y(n) = 0.414y(n-1) + 0.2927x(n) + 0.2927x(n-1)$$

The above time domain equation can be directly implemented with the help of multiplier, adder and delay blocks, as shown in Figure 10.8.

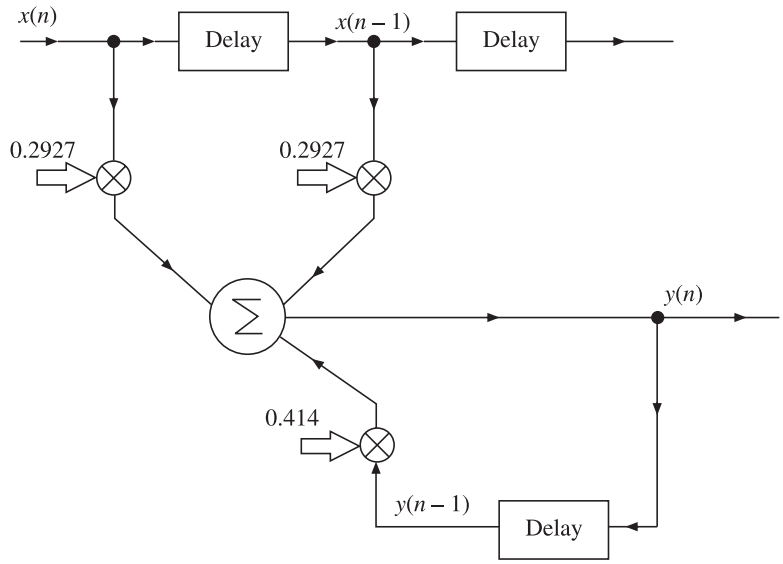


Figure 10.8 Numerical on design of IIR filter using bilinear transformation.

10.5 Design of a Digital Resonator Using Pole Zero Placement

We will now take up design of a digital resonator using placement of 'z' plane poles and zeros of the transfer function. The response of the resulting digital system is analogous to that of a tuned circuit. The motivation in taking up this exercise is to demonstrate that there are a variety of approaches available for designing digital filters.

10.5.1 Understanding the 'z' Plane

In order to understand the 'z' plane, let us start with the definition of 'z':

$$z = r\, e^{j\omega T_S}$$

We have $\omega = 2\pi f$; where 'f' is the frequency of the signal and $T_s = (1/f_s)$ where 'T_s' is the sampling time period which is the reciprocal of 'f_s' the sampling frequency. Hence, we can write:

$$z = re^{j2\pi f\frac{1}{f_s}}$$

$$z = r\,e^{j2\pi f_{pu}}$$

$$z = r\,e^{j\omega_{pu}}$$

Normally, the subscript '*pu*' is dropped from ω_{pu}. But we should keep in mind that 'ω' is per unit radian frequency whose maximum value is π. Thus,

$$z = r\,e^{j\omega}$$

Remembering that 'ω' is an angle, say 'θ' which varies from '0' to 'π'.

$$z = r\,e^{j\theta}$$

$$z = r\,e^{j\theta}\ (\text{angle from 0 to } \pi)$$

The frequency response is found by evaluating the transfer function $H(z)$ on the unit circle, i.e., $|z| = 1$ or $r = 1$. Hence, $z = e^{j\theta}$. Thus, we see that 'Ω' which is a linear distance (in the vertical direction) on the '*s*' plane, becomes an angle 'θ' on the '*z*' plane as shown in Figure 10.9. Thus, as frequency tends to infinity in the 's' domain, the angle tends to 'π' in the '*z*' domain as shown in Table 10.3.

Table 10.3 Comparison between 's' and '*z*' planes

Ω Analog frequency in 's' domain	ω Per unit frequency in the '*z*' domain	$z = r\,e^{j\omega}$
$\Omega \to 0$	$\omega \to 0$	$z = e^{j0} = 1$
$\Omega \to$ infinity	$\omega \to \pi$	$z = e^{j\pi} = -1$

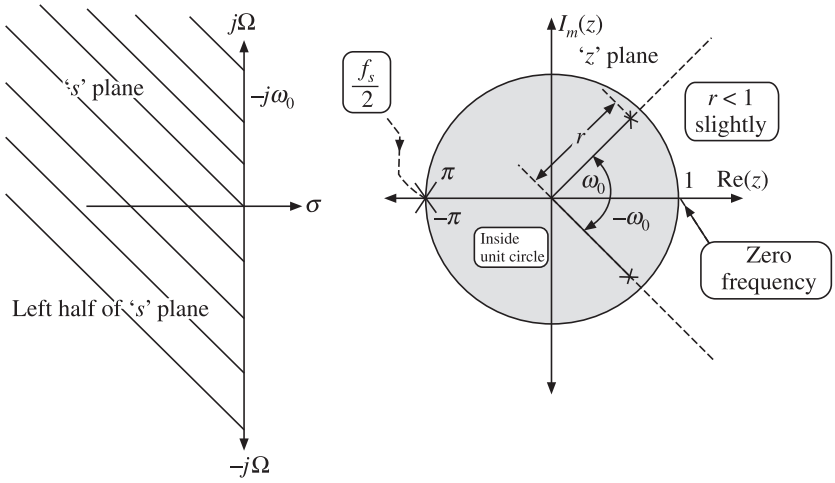


Figure 10.9 Mapping from 's' plane to 'z' plane.

Mapping of '*z*' plane from '*s*' plane from Figure 10.9 is shown in Table 10.4.

Table 10.4 Mapping of '*z*' plane from '*s*' plane

'*s*' plane	'*z*' plane
Left half	Inside unit circle
+/– $j\,\omega$ axis	Circumference of unit circle
Zero frequency	$z = 1$
Infinite frequency	$z = +/- \pi$ or $0.5\,f_s$

10.5.2 Placement of Pole

The aim is to design a digital resonator with a resonant frequency of F_0 as shown in Figure 10.10.

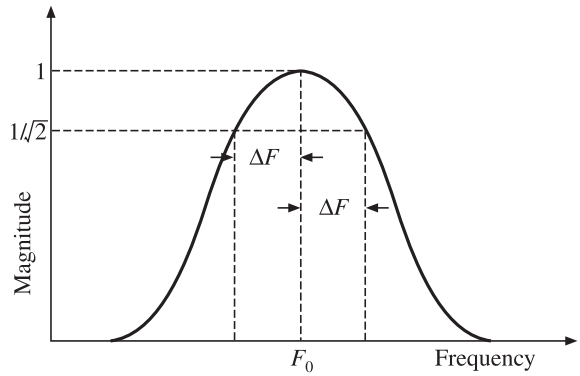


Figure 10.10 Frequency response of the digital resonator.

Observation of the desired frequency response shown in Figure 10.10 leads to the following conclusions:

1. The transfer function must contain a pole.
2. The pole should be at frequency $F_0 \rightarrow \theta_0$.
3. The pole should be located inside unit circle on the '*z*' plane, i.e., $r < 1$ for stability.

Now, the mapping is as follows:

$\Omega = 0$ to infinity ← analoge frequency (Hz)
$\omega = 0$ to π ← digital frequency (pu)
$f = 0$ to $f_s/2$ ← signal frequency (Hz)

Hence, we can write:

$$\theta_0 = \pi \frac{F_0}{f_s/2}$$

$$\theta_0 = \frac{2\pi F_0}{f_s}$$

The '*z*' domain transfer function must have a pole at $z = re^{j\theta_0}$, where $r < 1$

Now, since we want real coefficients, poles must always appear in conjugate pairs. Hence, we can now directly write the expression for the '*z*' domain transfer function as:

$$H(z) = \frac{\text{Numerator}}{(z - re^{j\theta_0})(z - re^{-j\theta_0})}$$

10.5.3 Placement of Zeros

Desired frequency response is zero at zero frequency as well as at very high frequencies. Thus, we have to locate zeros at zero frequency and infinite frequency. At some intermediate frequency, the response peaks because of the pole.

Thus, we also have to ensure that $H(z)$ has a null at $\omega = 0$, i.e., $z = e^{j\omega} = 1$.

Hence, $z = 1$ is a zero of the transfer function.

We also have to ensure that $H(z)$ has a null at $\omega = \pi$, i.e., at the highest frequency.

$$z = e^{-j\pi} = -1$$

Hence, $z = -1$ is also a zero of the transfer function. Thus, the transfer function that we are trying to synthesise can be written as:

$$H(z) = \frac{b_0\ (z-1)(z+1)}{(z - re^{j\theta_0})(z - re^{-j\theta_0})}$$

$$H(z) = \frac{b_0(z^2 - 1)}{z^2 - z\,r(e^{j\theta_0} + e^{-j\theta_0}) + r^2}$$

$$H(z) = \frac{b_0(z^2 - 1)}{z^2 - 2\,z\,r(\cos(\theta_0)) + r^2}$$

If we design the gain to be unity at $\omega = \omega_0$, then it will be automatically less than 1 at all other frequencies. The gain has already been designed to be zero at $\omega = 0$ and $\omega = \pi$.

Imposing this constraint on $H(z)$, we can write:

$$\left|H(z)\right|_{z=e^{j\theta}} = 1 = \frac{b_0\left|e^{2j\theta_0} - 1\right|}{\left|e^{2j\theta_0} - 2e^{j\theta_0} r\cos\left(\theta_0\right) + r^2\right|}$$

which gives b_0 as:

$$b_0 = \frac{\left|e^{2j\theta_0} - 2re^{j\theta_0}\cos(\theta_0) + r^2\right|}{\left|e^{2j\theta_0} - 1\right|}$$

The value of '*r*' should be nearly equal to 1 and on the lower side. The empirical formula for '*r*' is given by:

$$r = 1 - \frac{\Delta F\,\pi}{f_s}$$

where ΔF is change in frequency from F_0 at which the gain is 3 dB down.

10.5.4 Expression for *H*(*z*), the Frequency Domain Transfer Function

$$H(z) = \frac{b_0(z^2 - 1)}{z^2 - 2zr(\cos(\theta_0)) + r^2}$$

$$\left|H(z)\right|_{z=e^{j\theta}} = 1 = \frac{b_0\left|e^{2j\theta_0} - 1\right|}{\left|e^{2j\theta_0} - 2e^{j\theta_0}r\cos(\theta_0) + r^2\right|}$$

$$b_0 = \frac{\left|e^{2j\theta_0} - 2re^{j\theta_0}\cos(\theta_0) + r^2\right|}{e^{2j\theta_0} - 1}$$

$$r = 1 - \frac{\Delta F\,\pi}{f_s}$$

10.5.5 Numerical Problem on Design of Digital Resonator

EXAMPLE 10.3 Design a digital resonator for a frequency of 200 Hz. Assume sampling frequency to be 1200 Hz and ΔF = 6 Hz.

Solution We have:

$$\theta_0 = \frac{2\pi F_0}{f_s} = 2\pi\frac{200}{1200} = \frac{2\pi}{6} = \frac{\pi}{3}$$

We have

$$r = 1 - \frac{\Delta F\,\pi}{f_s} = 1 - \frac{6\pi}{1200} = 0.9843$$

$$b_0 = \frac{\left|e^{2j\theta_0} - 2re^{j\theta_0}\cos(\theta_0) + r^2\right|}{\left|e^{2j\theta_0} - 1\right|}$$

$$b_0 = \frac{\left|e^{2j\theta_0} - 2(0.9843)e^{j\pi/3}\cos(\pi/3) + (0.9843)^2\right|}{\left|e^{2j\pi/3} - 1\right|}$$

$$b_0 = 0.0156$$

Hence, $H(z)$ can be written as:

$$H(z) = \frac{b_0(z^2 - 1)}{z^2 - 2zr(\cos(\theta_0)) + r^2}$$

$$= \frac{0.0156(z^2 - 1)}{z^2 - 2z\ (0.9843)(\cos(\pi/3)) + (0.9843)^2}$$

$$H(z) = \frac{0.0156(z^2 - 1)}{z^2 - 2z\,(0.9843)(0.5) + 0.9688}$$

$$= \frac{0.0156(z^2 - 1)}{z^2 - (0.9843)z + 0.9688}$$

$$= \frac{0.0156(1 - z^{-2})}{1 - (0.9843)z^{-1} + 0.9688z^{-2}}$$

$$= \frac{Y(z)}{X(z)} = \frac{0.0156(1 - z^{-2})}{1 - (0.9843)z^{-1} + 0.9688z^{-2}}$$

$$Y(z) - (0.9843)z^{-1}Y(z) + 0.9688z^{-2}Y(z) = 0.0156X(z) - 0.0156z^{-2}X(z)$$

Taking inverse *Z*-transform of the above equation, we get:

$$Y(z) - \rightarrow y(n)$$

$$z^{-1}Y(z) - \rightarrow y(n-1)$$

$$z^{-2}Y(z) - \rightarrow y(n-2)$$

$$X(z) - \rightarrow x(n)$$

$$z^{-2}X(z) - \rightarrow x(n-2)$$

$$y(n) - (0.9843)\,y(n-1) + 0.9688\,y(n-2) = 0.0156x(n) - 0.0156x(n-2)$$

$$y(n) = 0.0156x(n) - 0.0156x(n-2) + (0.9843)\,y(n-1) - 0.9688\,y(n-2)$$

The above difference equation leads to the 'direct form' implementation as shown in Figure 10.11.

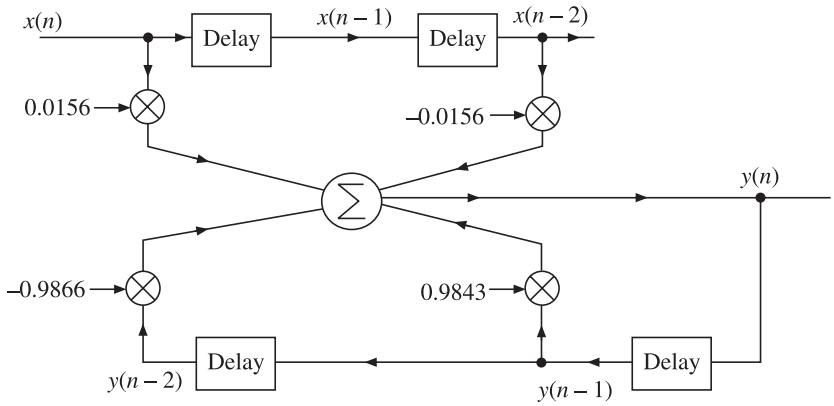


Figure 10.11 Direct form implementation of digital resonator (numerical problem).

REVIEW QUESTIONS

1. Sketch the ideal low-pass, high-pass, band-pass and band-stop filter magnitude responses.
2. Why phase response of digital filters is always negative?
3. What are the advantages of having linear phase response?
4. What constraints get imposed on the FIR filter impulse response (filter coefficients) when it is required to have linear phase response?
5. Describe the frequency sampling method of FIR filter design.
6. Compare FIR and IIR filters.
7. Which type of filter (FIR/IIR) is more suitable for numerical protective relaying application?
8. Design a low-pass FIR filter with a cut-off frequency of $f_{sampling}/6$. Assume order of the filter to be equal to 32.
9. In IIR filter, state whether the magnitude of the impulse response is infinite or its duration is infinite.
10. What is the basic reason for infinite impulse response?
11. Write expression for direct time domain implementation of the IIR filter.
12. Write the frequency domain transfer function, $H(z)$ of the IIR filter.
13. What are the various alternate ways in which $H(z)$ can be expressed?
14. What is the motivation for implementing the filter in other than 'direct form'?
15. Derive bilinear transformation.
16. What is the range of digital frequencies in Hz and in (pu) radians?
17. What do you mean by warping of frequency?
18. How frequency warping can be pre-compensated by pre-warping?
19. State transfer functions of low-pass Butterworth, Chebyshev and elliptical filters in the '*s*' domain.
20. Numerical problems:
    (i) Design an IIR digital filter based on first order low-pass Butterworth filter prototype with 3 dB cut-off frequency $\omega_c = 0.1\pi$; by using bilinear transformation.
    (ii) Design a digital resonator for a frequency of 100 Hz. Assume sampling frequency to be 1500 Hz and $\Delta F = 4$ Hz.

11

Synchrophasors

11.1 Introduction [1, 4, 7, 9, 12, 18, 32, 35, 36, 37]

Synchrophasors are marching inexorably into the field of power systems and seem to have caught the imagination of the protection engineers. They have become the mainstay of the so called smart-grid and penetrated considerably into the existing power system transmission network. They have made feasible the concept of wide area protection, control and monitoring. It is said that If SCADA system is akin to X-Ray then synchrophasor is the MRI! One single factor which has made the synchrophasor revolution possible is the affordability GPS technology.

Synchrophasors combine the concept of phasor with that of synchronisation in time. The phasors were conceptualised by Charles Proteus Steinmetz in 1893. In 1988 synchrophasor measurement units were invented by Arun G. Phadke and James S. Thorp at Virginia Tech University in the U.S.A. Thus, Steinmetz's technique of phasor calculation has evolved into the calculation of real time phasors that are synchronised to an absolute time reference distributed by the global positioning system (GPS). Early prototypes of the PMU were built at Virginia Tech. Macrodyne Corporation (U.S.A.) built the first commercial phasor measurement unit (PMU) in 1992 almost a hundred years after Steinmetz's conceptualisation of *phasor*. The concept of synchrophasor was standardised by IEEE in 1995 as IEEE Std 1334–1995 which was reaffirmed in 2001. A working group was established in 2001 to update the standard. The new synchrophasor standard was issued in 2005 as IEEE Std C37.118–2005. This was modified in 2011 and broken up into two standards C37.118.1 which specified the measurement part of the PMU and C37.118.2 which specified the communication aspects. The new standard now includes specifications of PMU output under transient conditions, about which its previous version was silent. In the following sections we will first visit the concept of phasor and then develop the idea of synchrophasor as defined by the IEEE.

11.2 The Phasor

An electrical signal represented by the equation:

$$v_x = \sqrt{2}V_{\text{RMS}} \cos(2\pi f t + \phi) \quad \text{where } \omega = 2\pi f$$

or

$$v_x = \sqrt{2}V_{\text{RMS}} \cos(\omega t + \phi)$$

can be written in exponential form as:

$$v_x = \text{Re}(\sqrt{2}V_{\text{RMS}} e^{j(\omega t + \phi)})$$

$$v_x = \text{Re}\{\sqrt{2}(V_{\text{RMS}} e^{j\phi}) e^{j\omega t}\}$$

$$\underbrace{v_x}_{\text{Instantaneous value}} = \text{Re}\left\{\sqrt{2}\,\underbrace{V_{\text{RMS}} e^{j\phi}}_{\text{Phasor}}\, e^{j\omega t}\right\}$$

Thus, while defining the phasor, we drop the term $e^{j\omega t}$ in the above expression and consider only the RMS value and phase angle to express the phasor $\vec{V}$ as:

$$\vec{V} = V_{\text{RMS}} e^{j\phi}$$

with the tacit understanding that frequency 'ω' is constant. In phasor calculations we replace the signal v_x by the phasor $\vec{V}$. Thus, we have made the reversible phasor transformation denoted by $\mathcal{P}(\ )$.

$$\mathcal{P}(v_x) \xrightarrow{\text{Yields}} \vec{V}$$

and the inverse operation is:

$$\mathcal{P}^{-1}(\vec{V}) \xrightarrow{\text{Yields}} v_x$$

$$\mathcal{P}(\sqrt{2}V_{\text{RMS}} \cos(\omega t + \phi)) \xrightarrow{\text{Yields}} V_{\text{RMS}} e^{j\phi}$$

and the inverse operation is:

$$\mathcal{P}^{-1}(V_{\text{RMS}} e^{j\phi}) \xrightarrow{\text{Yields}} \sqrt{2}V_{\text{RMS}} \cos(\omega t + \phi)$$

The reason, for popularity of phasors, is that in the steady state, it is very convenient to deal with the signal represented in the phasor form rather than in instantaneous form. We should note, therefore, that ***phasor* is inherently a steady state concept.** Conventional phasor representation is valid for single frequency undistorted sinusoids of constant amplitude, i.e., for **stationary** signals. In a power system, immediately after occurrence of faults or switching events or during a power swing, signals are not stationary. During these times, it is difficult to apply classical concept of phasor. However, the recent modification to the IEEE synchrophasor standard for synchrophasor C37.118.1, provides definition of the synchrophasor under non-stationary conditions, i.e., when both its amplitude and phase angle are being modulated by a low frequency signal.

Figure 11.1 shows, the snap-shot of a length V_m rotating at ω rad/s. As the length V_m rotates, its instant to instant projection along the horizontal axis is the instantaneous value of the phasor. If the length V_m happened to be aligned with the horizonal axis at 't = 0' then we associate

a phase angle of 0° with it. Thus, expressing the phasor as $\overrightarrow{V_1} = (V_m/\sqrt{2})\angle 0$. Suppose at time '$t$ = 0' the rotating length V_m happened to be at an angle 'ϕ' with the horizontal axis, then we associate a phase angle of ϕ° with it, thus, expressing the phasor as $\overrightarrow{V_2} = (V_m/\sqrt{2})\angle\phi$. Hence, the instantaneous value, the peak value, the RMS value and the phase angle are all linked with the same entity, i.e., the sinusoidal waveform, the phasor captures all this information in a compact form.

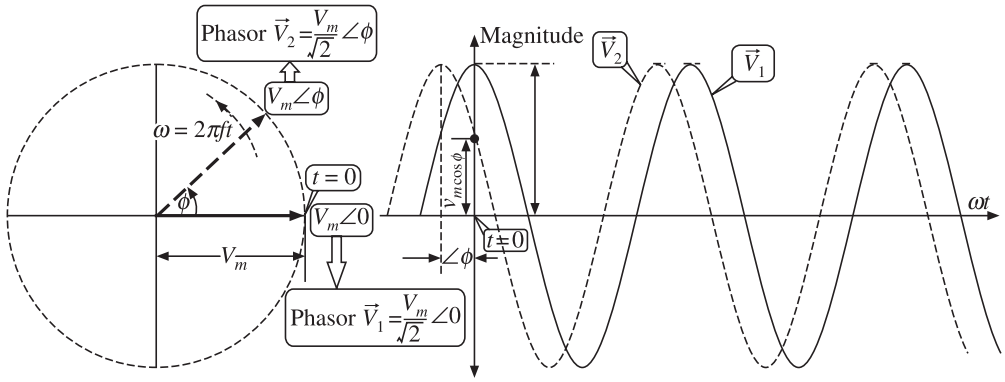


Figure 11.1 Phasor representation.

The phasor diagram is a snap-shot of the rotating phasor of magnitude $V_m/\sqrt{2}$ rotating with angular speed $\omega = 2\pi f$. Note that the instantaneous value of the signal is $\sqrt{2}$ times the projection of the rotating phasor on the horizontal axis.

11.3 The Synchrophasor as per IEEE Std C37.118.1-2011

The phase angle ϕ of the phasor is determined by comparing it with a fictitious cosine waveform which was at its peak at some absolute instant of time which is taken as reference (t = 0). The synchrophasor technology has become possible because of the ability, through GPS, to precisely refer to the t = 0 instant, all over the power system and at all times. The concept of a phasor is a strictly steady state concept. However, in practise we encounter situations where the amplitude as well as phase angle are continuously varying according to a low-frequency signal. The IEEE Standard for synchrophasor now covers both the steady-state and transient conditions. We first take up synchrophasor definition under steady state conditions followed by that under transient conditions.

11.3.1 The Synchrophasor in Steady State Conditions

Figure 11.2 shows the reference cosine wave, given by $v_x = V_m\cos(\omega t)$, i.e., a cosine wave with a phase angle of zero, with which any given 50 Hz wave will be compared for its synchrophasor representation. As defined in the IEEE Std C37.118-2011, the time count starts from t = 0 instant, which is defined as 00 hr : 00 min : 00 sec on January 1st, 1970. Precise time-keeping has been maintained from this instant with the help of highly precise 1 pulse per

second signal. At the start of every second (which can be traced back to this '$t = 0$' instant), a signal known as the '1 PPS' signal is broadcasted from the geographical positioning system satellites, and is available throughout the power system over widely separated geographical locations, indeed throughout the globe. A small fist sized GPS antenna connected to a receiver is used to pick up this signal and decode it. Figure 11.2 also shows the phasor diagram for the reference wave, which will be expressed mathematically as:

$$\overrightarrow{V_{\text{ref}}} = \frac{V_m}{\sqrt{2}} \angle 0 = \frac{V_m}{\sqrt{2}} e^{j0}$$

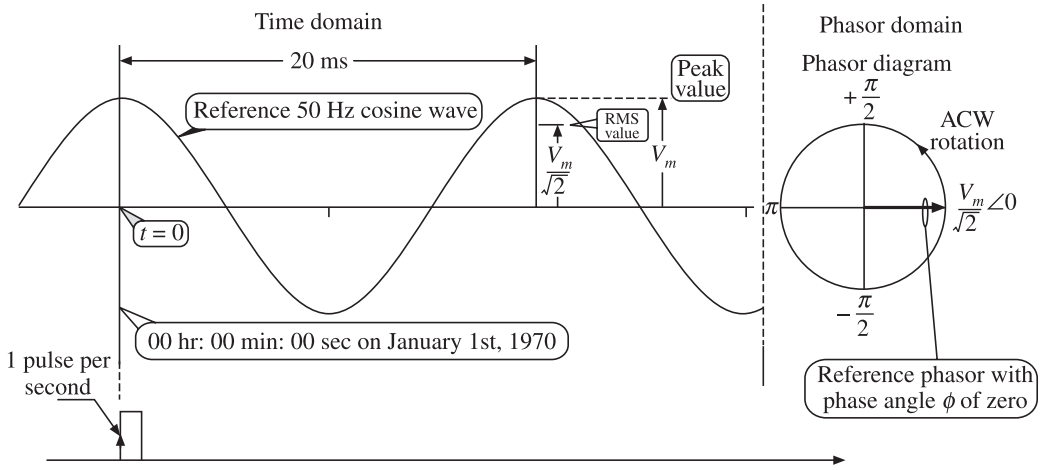


Figure 11.2 Reference cosine wave for synchrophasor representation.

Figure 11.3 shows a signal with amplitude of V_m and phase angle of ϕ leading (i.e., positive phase angle). It can be seen that the phase angle of the measured waveform is given by measuring the occurrence of its peak with respect the rising edge of the 1 PPS signal.

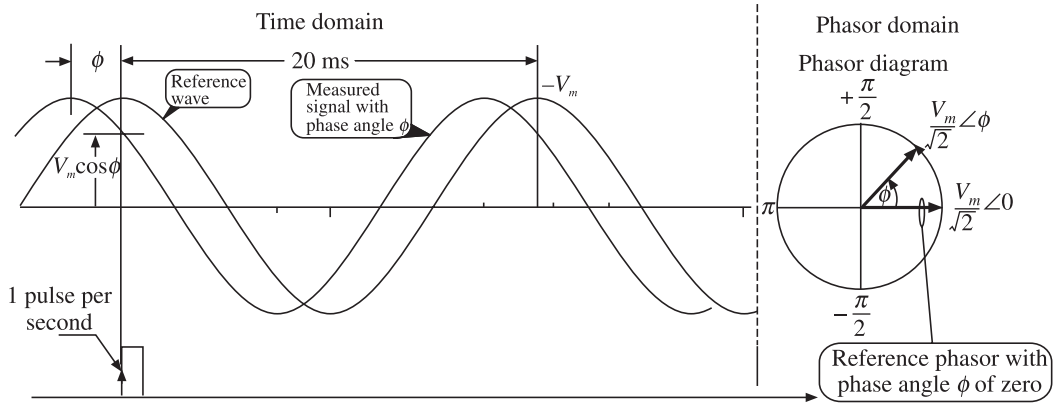


Figure 11.3 Measured signal with peak value V_m and phase angle ϕ.

11.3.2 The Synchrophasor in Transient Conditions

In the general case when the amplitude is a function of time expressed as $V_m(t)$ and the sinusoid frequency is also a function of time $f(t)$, we can define the function $g(t) = f(t) - f_0$ where f_0 is the nominal frequency and $g(t)$ is the difference between the actual and nominal frequencies. Note that frequency, by definition, is rate of change of phase, i.e., $\omega = d\phi/dt$

The sinusoid during transient condition can then be expressed as:

$$v(t) = V_m(t)\cos(\omega(t)t + \phi_o)$$

Note that $\{\omega(t)t\}$ represents the phase which continuously changes with time while ϕ_o is the fixed phase offset.

$$v(t) = V_m(t)\cos(\phi(t) + \phi_o)$$

We have

$$\omega = \frac{d}{dt}(\phi(t))$$

$$2\pi f = \frac{d(\phi(t))}{dt}$$

$$d(\phi(t)) = 2\pi f\, dt$$

$$\int d(\phi(t)) = 2\pi \int f\, dt$$

$$\text{Phase change with time} = 2\pi \int f\, dt$$

Thus, when the input frequency is varying, we can write the input signal as:

$$v(t) = V_m(t)\cos\left(2\pi \int f\, dt + \phi_o\right)$$

$$v(t) = V_m(t)\cos\left(2\pi \int (g(t) + f_0)dt + \phi_o\right)$$

$$v(t) = V_m(t)\cos\left(2\pi f_0 \int dt + 2\pi \int g(t)dt + \phi_o\right)$$

The synchrophasor representation for the above signal is given by:

$$\vec{V} = \frac{V_m(t)}{\sqrt{2}} e^{j2\pi \int g(t)dt + \phi_o}$$

Note that unlike the classical phasor, the synchrophasor magnitude as well as phase angle, both are now functions of time.

For the special case when $V_m(t) = V_m$ is constant and $g = \Delta f$ is a constant offset from the nominal frequency $\int g(t)dt = \int \Delta f\, dt = \Delta f\, t$, so the synchrophasor is simply:

$$\vec{V} = \frac{V_m}{\sqrt{2}} e^{j(2\pi \Delta f\, t + \phi_0)}$$

Note the $\vec{V}$ will now rotate at uniform rate Δf, the difference between the actual and off-nominal frequency.

11.3.3 Synchrophasor under Dynamic Conditions

Under dynamic conditions, the three phase signal may be represented by

$$v_a = V_m[1 + K_x \cos(\omega_m t)] \cos[\omega_0 t + K_a \cos(\omega_m t - \pi)]$$
$$v_b = V_m[1 + K_x \cos(\omega_m t)] \cos[\omega_0 t + K_a \cos(\omega_m t - \pi) - 2\pi/3]$$
$$v_c = V_m[1 + K_x \cos(\omega_m t)] \cos[\omega_0 t + K_a \cos(\omega_m t - \pi) + 2\pi/3]$$

where V_m is the amplitude of the input signal, ω_0 is the nominal power frequency, ω_m is the modulating frequency in radians per second and $f_m = \omega_m/2\pi$ is the modulating frequency in Hz, K_x is the amplitude modulation factor and K_a is the phase angle modulation factor. The phase 'a' positive sequence signal corresponding to the above 3-phase input is given by

$$V_{a1} = V_m[1 + K_x \cos(\omega_m t)]\cos[\omega_0 t + K_a \cos(\omega_m t - \pi)]$$

Thus, the PMU should produce a positive sequence phasor measurement of:

$$\overrightarrow{V_{a1}} = \frac{V_m[1 + K_x \cos(\omega_m t)]}{\sqrt{2}} \angle K_a \cos(\omega_m t - \pi)$$

Note that we can write, $t = nT$; where 'n' is integer and T is phasor reporting rate, giving positive sequence 'a' phase phasor as:

$$\overrightarrow{V_{a1}}(nT) = \frac{V_m[1 + K_x \cos(\omega_m nT)]}{\sqrt{2}} \angle K_a \cos(\omega_m nT - \pi)$$

It may be noted that during steady state the synchrophasor expression did not explicitly involve power system frequency ω_0 or time 't'. It was implicitly implied. However during dynamic conditions, ω_m, i.e., the rate at which the synchrophasor frequency is being modulated and nT, the discrete time at which synchrophasors are reported is explicitly a part of the synchrophasor expression.

11.3.4 Total Vector Error (TVE)

The theoretical values of the synchrophasor and that estimated by PMU may have differences in both magnitude as well as phase angle. The IEEE Standard combines both errors by defining total vector error (TVE). The TVE is defined as:

$$\text{TVE}(n) = \sqrt{\frac{\left(\hat{V}_r(n) - V_r(n)\right)^2 + \left(\hat{V}_i(n) - V_i(n)\right)^2}{(V_r(n))^2 + (V_i(n))^2}}$$

where $\hat{V}_r(n)$ and $\hat{V}_i(n)$ are the sequence of estimates given by the PMU under test and $V_r(n)$ and $V_i(n)$ are the sequence of theoretical values of the input signal.

As can be seen from Figure 11.4, the maximum error is 1% when the phase angle error is zero and maximum error in angle is $\tan^{-1}(.01OP/OP) = \tan^{-1}(.01) = 0.572938698 \cong 0.573°$.

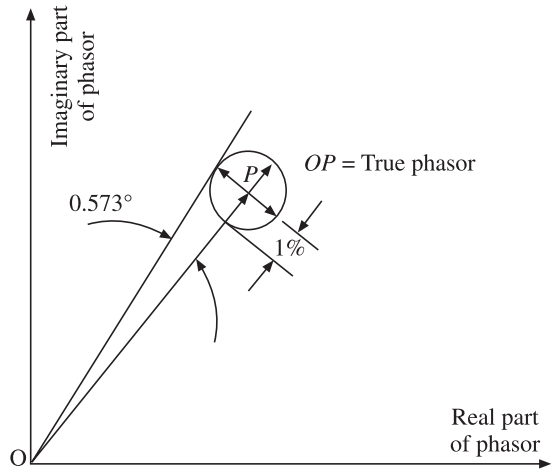


Figure 11.4 Total vector error.

11.3.5 Frequency and Rate of Change of Frequency (ROCOF)

As per the synchrophasor standard, the PMU must be capable of calculating and reporting frequency as well as rate of change of frequency (ROCOF). The following definitions are used for this purpose:

Given a sinusoidal signal:

$$x(t) = X_m \cos[\psi(t)]$$

The frequency is defined as:

$$f(t) = \frac{1}{2\pi}\frac{d\psi(t)}{dt}$$

and the rate of change of frequency (ROCOF) is defined as:

$$\text{ROCOF}(t) = \frac{df(t)}{dt}$$

Synchrophasors are always computed in relation to the system nominal frequency 'f_0'. If the argument of the cosine is represented as:

$$\psi(t) = \omega_0 t + \varphi(t) = 2\pi\left[f_0 t + \frac{\varphi(t)}{2\pi}\right]$$

then the frequency can be expressed as:

$$f(t) = f_0 + \frac{d}{dt}\left[\frac{\varphi(t)}{2\pi}\right] = f_0 + \Delta f(t)$$

where $\Delta f(t)$ is the deviation of frequency from nominal.

Hence, ROCOF can be written as:

$$\text{ROCOF}(t) = \frac{d^2}{dt^2}\left[\frac{\varphi(t)}{2\pi}\right] = \frac{d}{dt}\left[\Delta f(t)\right]$$

Frequency in phasor measurements may be reported as the actual frequency $f(t)$ or the deviation of frequency from nominal, $\Delta f(t)$. In steady state conditions, $\Delta f(t)$ can be represented as a scalar number.

11.4 Time Tagging (Stamping) of the Synchrophasor

What sets apart the synchrophasor from phasor is called 'time tag' or 'time stamp'. According to the IEEE Standard, the synchrophasor measurement is to be tagged with UTC (universal time coordinated) time corresponding to the time of measurement. This shall consist of three numbers, i.e., a second-of-century (SOC) count, a fraction-of-second count, and a time status value. The SOC count shall be a 4-byte binary count in seconds from midnight (00 hr:00 min) of January 1st, 1970 to the present second. Leap seconds shall be added or deleted from this count as necessary to keep it synchronised with UTC. Insertion of a leap second results in two successive seconds having the same SOC count which are differentiated by the leap second bit in the FRACSEC word as defined further. This SOC time stamp is the same as used by the UNIX operating system and is similar to those used by other computer operating systems including DOS, MAC OS and network time protocol (NTP).

The second shall be divided into an integer number of subdivisions by the TIME_BASE integer which is 2^{24}. The fraction of second = (fraction of second count (FRACSEC)/2^{24}). The fraction-of-second count shall be zero when it coincides with the 1s rollover. The structure of the time stamp is shown in Figure 11.5.

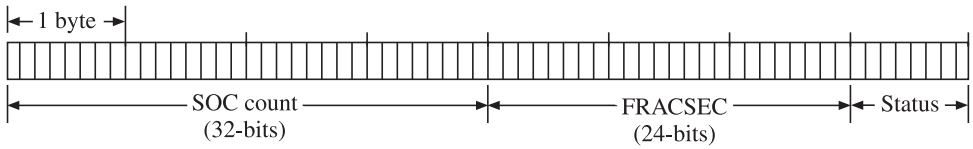


Figure 11.5 Structure of the synchrophasor time stamp.

The SOC counter started filling up at 00:00 on January 1st, 1970 and is getting incremented every second. One may wonder how long it will take for the counter to become full and hence over-run in the next second. It can be calculated as:

$$\frac{\text{Seconds}}{\text{Year}} = \left(\frac{\text{sec}}{\text{min}}\right)\left(\frac{\text{min}}{\text{hr}}\right)\left(\frac{\text{hr}}{\text{day}}\right)\left(\frac{\text{day}}{\text{year}}\right) = (60)(60)(24)(365) = 31536000$$

Capacity of the SOC counter = 2^{32} = 4294967296
Number of years after which SOC counter will become full:

$$\frac{\text{Capacity of SOC conuter}}{\text{Seconds/year}} = \frac{4294967296}{31536000} = 136.1925\ldots \text{ years.}$$

Thus, it will be 136 years from January 1st, 1970, i.e., 2106 A.D., by the time SOC counter over-runs. In all probability long before this happens, the technology will have changed and hopefully will not cause something like the infamous Y2K problem during roll-over from 1999 to 2000.

The exact number of seconds, including the fractional part, since January 1st, 1970 can be computed from the time stamp as:

$$\text{Exact number of seconds} = \text{SOC} + \frac{\text{FRACSEC}}{2^{24}}$$

The PMU has to communicate the measurements with other devices such as *phasor data concentrators*. Some protocol has to be used for such communications. The protocol allows for necessary identifying information, such as the PMU IDCODE and status to be embedded in data frame for proper interpretation of the measured data. Four message types are defined, i.e., data, configuration, header, and command. The first three message types are transmitted from the PMU and the last, i.e., command is received by the PMU. Configuration is a machine readable message describing the data that the PMU sends and providing calibration factors. Header is human-readable descriptive information sent from the PMU but provided by the user. Commands are machine-readable codes sent to the PMU for control and configuration. Information can be stored in any convenient form in the PMU but when transmitted it shall be formatted as frames. Figure 11.6 shows the configuration of a data frame sent out by a PMU.

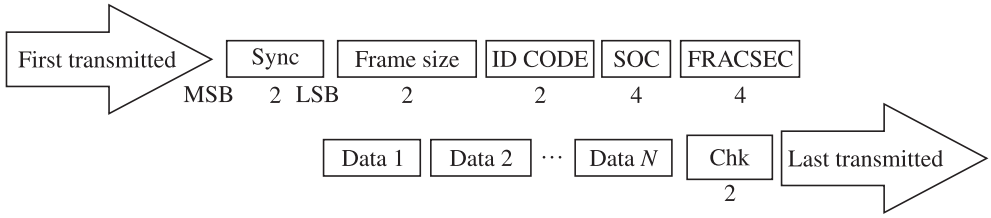


Figure 11.6 Data frame transmission order.

11.5 Dissemination of the Time Stamp: IRIG-B Standard

IRIG-B (Inter Range Instrumentation Group, 'B' is for utility industry) is fully described in the IRIG Standard 200. Time is provided once per second in seconds through day of year in a binary coded decimal (BCD) format and an optional second of the year count. IRIG-B signal may be provided in a level shift, a 1 kHz amplitude modulated signal or in a bi-phase Manchester modulated format. The IRIG-B amplitude modulated format is the most commonly used. The Manchester format is more compatible with fibre optic and digital systems and provides complete synchronisation without additional signals.

The time code format is:

<sync> SS: MM: HH: DDD <control> <binary seconds>
where <sync> is the on-time sync marker
SS is the second of the minute [00 to 59] (60 during leap second)
MM is the minute of the hour (00 to 59)
HH is the hour of the day in 24 hour format (00 to 23)

DDD is the day of the year (001 to 366)
<control> is a block of 27 binary control characters
<binary seconds> is a 17-bit second of the day in binary

11.6 Synchrophasor Reporting Rates

As per the IEEE Std C37.118, the phasor measurement unit (PMU) shall support data reporting (by recording or output) at sub-multiples of the nominal power-line (system) frequency. Standard reporting rates for the 50 Hz and 60 Hz systems are listed in Table 11.1.

Table 11.1 Synchrophasor data reporting rates

System frequency	50 Hz		60 Hz				
Reporting rates in frames per second	10	25	10	12	15	20	30

11.7 Block Diagram of the Synchrophasor Enabled Digital Relay or Phasor Measurement Unit (PMU)

Figure 11.7 shows the architecture of a synchrophasor enabled digital relay. It can be seen that it has been evolved from the block diagram of the digital relay given in Chapter 2, by adding the GPS clock and a phase locked oscillator locked to the 1 PPS signal given by the GPS clock. The GPS receiver also provides the DSP microprocessor with the time stamp using IRIG-B protocol.

A phasor measurement unit will have very similar organisation. To difference between a synchrophasor enabled digital relay and a PMU is that the PMU just computes the phasors, time stamps them, assembles the data frames and either stores them (records them) or transmits them to a remote entity which may be a phasor data concentrator or a digital computer, whereas a digital protective relay is tasked with taking a trip/restrain decision after computing the synchrophasors.

11.7.1 Two Alternatives for Sampling

We need to give careful thought to the effect of sampling frequency on the phasor computation, because in a real life power system the signal frequency is almost never exactly equal to the nominal frequency of 50 Hz (or 60 Hz). If we lock the sampling frequency to the nominal frequency of 50 Hz, as we do in synchrophasors, we will end up with inaccurate measurement due to the leakage effect. This inaccurate measurement will have to be corrected or 'compensated' by accounting for the actual signal frequency.

If we wish to get rid of this error (rather than compensating it) then we will have to lock the sampling to the actual power system frequency by actually measuring it. Thus in both cases extra work, than that indicated by the DFT algorithm is needed. Both the approaches are shown schematically in Figures 11.8 and 11.9. As already mentioned in synchrophasor

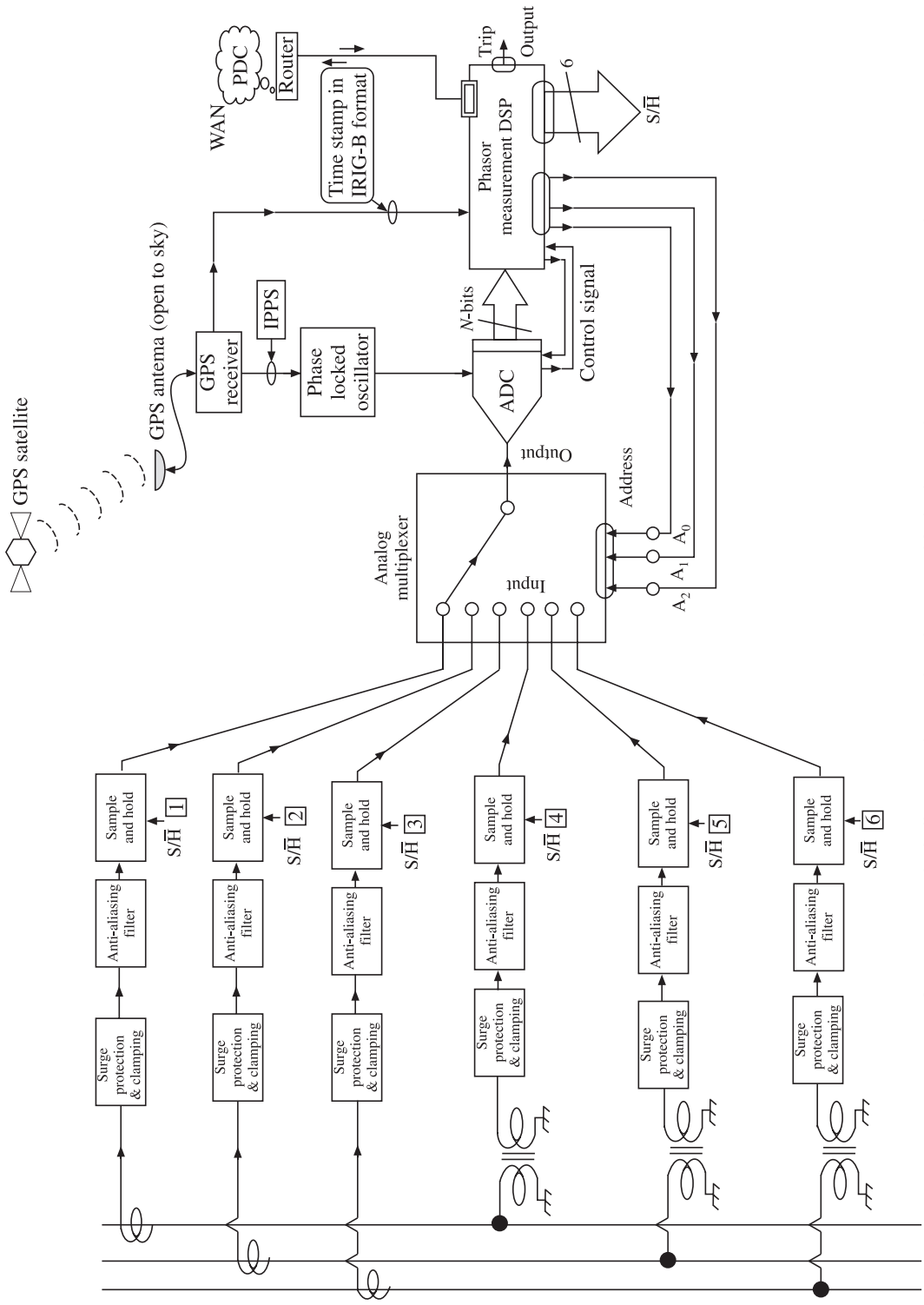


Figure 11.7 Block diagram of a synchrophasor enabled relay or phasor measurement unit.

approach, we keep the sampling frequency locked to a fictitious cosine wave whose positive peak synchronised with 00 hr: 00 min : 00 sec on January 1st, 1970 as shown in the figure.

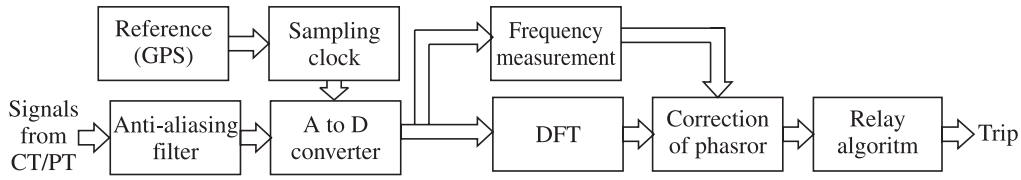


Figure 11.8 Sampling locked with a fictitious 50 Hz Signal via GPS.

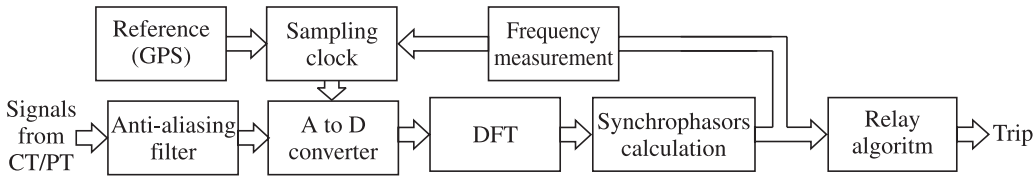


Figure 11.9 Sampling locked with system frequency.

11.8 Off-nominal Frequency Operation of the PMU

It is well-known that the power system frequency is never exactly equal to the nominal frequency of 50 Hz or 60 Hz. The power system frequency is kept very close to the nominal value and in the worst case can vary within a few hertz around the nominal frequency for a short duration. However, the sampling is kept locked to the fictitious 50 Hz cosine wave starting at $t = 0$ instant. This can be expected to cause some error in the DFT computation, which needs to be investigated and accounted for. We first demonstrate the effect of off-nominal frequency operation by carrying out a 4-point DFT computation by hand and then follow it up with a small MATLAB script and then develop analytical expression for the same.

11.8.1 A Demonstration of Error Caused by Off-nominal Frequency Operation

Let the nominal frequency be 50 Hz and the sampling be done at four times the nominal frequency, i.e., at 200 Hz. Hence, the sampling time period $= \Delta t = \dfrac{1}{f_{\text{sampling}}} = \dfrac{1}{200} = (5)10^{-3}$ s.

Now let us consider two phasors, both of value $1 \measuredangle 0$ but one having frequency of 50 Hz and another of 51 Hz and sample both of them at 200 Hz for one cycle thus drawing four samples of each. From these four samples let us evaluate the DFT and compare the computed and actual values, thus finding errors, if any. The samples of the two signals are shown in Table 11.2.

Table 11.2 Samples of 50 Hz and 51 Hz signal

Sample number n	$t = n\,\Delta t$	$v_{50} = (1)\cos(2\,\pi\,50\,n\,\Delta t)$	$v_{51} = (1)\cos(2\,\pi\,51\,n\,\Delta t)$
0	0	1	1
1	$(1)\cdot(5.10^{-3})$	0	–0.03141
2	$(2)\cdot(5.10^{-3})$	–1	–0.9980
3	$(3)\cdot(5.10^{-3})$	0	0.0941

The 4-point DFT of the 50 Hz signal is given as:

$$\sum_{n=0}^{3} v_{50}(n)e^{\frac{-j2\pi 1n}{4}} = \sum_{n=0}^{3} v_{50}(n)e^{\frac{-j\pi n}{2}} = \sum_{n=0}^{3} v_{50}(n) \angle (-90°(n))$$

$$V_{50} = v_{50}(0) \angle (-90°(0)) + v_{50}(1) \angle (-90°(1)) + v_{50}(2) \angle (-90°(2)) + v_{50}(3) \angle (-90°(3))$$

$$V_{50} = 1 \angle (-90°(0)) + 0 \angle (-90°(1)) - 1 \angle (-90°(2)) + 0 \angle (-90°(3))$$

$$V_{50} = 1 \angle ((0)) - 1 \angle (-180) = 2 \angle 0$$

Hence, actual (peak) value of $V_{50} = \frac{2}{N} V_{50} = \frac{2}{4} 2\angle 0 = 1\angle 0$.

Thus, there is no error, either in magnitude or in phase angle when the 50 Hz phasor is sampled at 4 × 50 Hz frequency. Let us do similar calculations on the samples of the 51 Hz signal as shown below:

$$\sum_{n=0}^{3} v_{51}(n)e^{\frac{-j2\pi 1n}{4}} = \sum_{n=0}^{3} v_{51}(n)e^{\frac{-j\pi n}{2}} = \sum_{n=0}^{3} v_{51}(n) \angle (-90°(n))$$

$$V_{51} = v_{51}(0) \angle (-90°(0)) + v_{51}(1) \angle (-90°(1)) + v_{51}(2) \angle (-90°(2)) + v_{51}(3) \angle (-90°(3))$$

$$V_{51} = 1 \angle (-90°(0)) - 0.03141 \angle (-90°(1)) - 0.9980 \angle (-90°(2)) + 0.0941 \angle (-90°(3))$$

$$V_{51} = 2.001965 \angle (3.5946)$$

Hence, actual (peak) value of $V_{51} = \frac{2}{N} V_{51} = \frac{2}{4} 2.001965\angle(3.5946) = 1.0009827\angle 3.5946$.

Thus, there is error, both in magnitude and in phase angle when the 51 Hz phasor is sampled at 4 × 50 Hz frequency. It can be seen that the error in magnitude is only 0.098% which is less than 0.1% while the error in phase angle is 3.5946°. Thus, phase angle error in synchrophasors when off-nominal frequency signals are encountered is more serious than the error in magnitude.

11.8.2 Analytical Expression for Effect of Off-nominal Frequency Operation

We can write the expression for phasor as:

$$\widehat{X} = \frac{\sqrt{2}}{N} \sum_{n=0}^{N-1} x[n\,\Delta t] e^{\frac{-j\,2\,\pi n}{N}}$$

where $\hat{\vec{X}}$ is the one-cycle phasor estimate,

$$\Delta t = \frac{1}{f_{\text{sampling}}} = \frac{1}{N f_{\text{nom}}}$$

where f_{nom} is the nominal frequency (50 Hz or 60 Hz). It is assumed that N is even. Let the sample belong to an off-nominal frequency 'f'. Hence, from the definition of a phasor we can express the sample as:

$$x[n\Delta t] = \sqrt{2}\ \text{Real}\,[\vec{X}\, e^{j2\pi f\, n\Delta t}]$$

where $\vec{X}$ is the 'true' phasor value of the samples.

$$x[n\Delta t] = \sqrt{2}\ \text{Real}\left[\vec{X}\, e^{\frac{j2\pi f\, n}{N f_{\text{nom}}}}\right]$$

The one cycle phasor estimate can then be written using the DFT equation for an off-nominal frequency signal as:

$$\hat{\vec{X}} = \frac{\sqrt{2}}{\text{N}} \sum_{n=0}^{N-1} \sqrt{2}\,\text{Real}\left[\vec{X}\, e^{\frac{j2\pi f\, n}{N f_{\text{nom}}}}\right] e^{\frac{-j2\pi n}{N}}$$

Now for any phasor $\vec{P}$, the $\text{Real}\,(\vec{P})$ can be written as

$$\text{Real}\,(\vec{P}) = \frac{\vec{P} + \vec{P}^*}{2}$$

Hence, we can write the above equation during kth window as:

$$\hat{\vec{X}} = \frac{2}{2N} \sum_{n=k}^{k+N-1} \left[\vec{X}\, e^{\frac{j2\pi f\, n}{N f_{\text{nom}}}} + \overrightarrow{X^*}\, e^{\frac{-j2\pi f\, n}{N f_{\text{nom}}}}\right] e^{\frac{-j2\pi n}{N}}$$

$$\hat{\vec{X}} = \frac{1}{N} \sum_{n=k}^{k+N-1} \left[\vec{X}\, e^{\frac{j2\pi f\, n}{N f_{\text{nom}}}}\right] e^{\frac{-j2\pi n}{N}} + \frac{1}{N} \sum_{n=k}^{\text{k+N}-1} \left[\overrightarrow{X^*}\, e^{\frac{-j2\pi f\, n}{N f_{\text{nom}}}}\right] \text{e}^{\frac{-j2\pi n}{N}}$$

$$\hat{\vec{X}} = \frac{1}{N} \vec{X} \sum_{n=k}^{k+N-1} \left[e^{\frac{j2\pi f\, n}{N f_{\text{nom}}} - \frac{j2\pi n}{N}}\right] + \frac{1}{N} \overrightarrow{X^*} \sum_{n=k}^{k+N-1} \left[e^{\frac{-j2\pi f\, n}{N f_{\text{nom}}} - \frac{j2\pi n}{N}}\right]$$

$$\hat{\vec{X}} = \frac{1}{N} \vec{X} \sum_{n=k}^{k+N-1} \left[e^{\frac{j2\pi\, n}{N}\left(\frac{f}{f_{\text{nom}}} - 1\right)}\right] + \frac{1}{N} \overrightarrow{X^*} \sum_{n=k}^{k+N-1} \left[e^{\frac{-j2\pi n}{N}\left(\frac{f}{f_{\text{nom}}} + 1\right)}\right]$$

$$\hat{\vec{X}} = \frac{1}{N} \vec{X} \sum_{n=k}^{k+N-1} \left[e^{\frac{j2\pi}{N}\left(\frac{f}{f_{\text{nom}}} - 1\right)}\right]^n + \frac{1}{N} \overrightarrow{X^*} \sum_{n=k}^{k+N-1} \left[e^{\frac{-j2\pi}{N}\left(\frac{f}{f_{\text{nom}}} + 1\right)}\right]^n$$

Let $n - k = q$, hence $(n = q + k)$ and when $n = k$; $n - k = q = 0$ and when $n = k + N - 1$; $n - k = q = N - 1$, hence we can write the above equation as:

$$\tilde{\vec{X}} = \frac{1}{N}\vec{X}\sum_{q=0}^{N-1}\left[e^{\frac{j2\pi}{N}\left(\frac{f}{f_{\text{nom}}}-1\right)}\right]^{q+k} + \frac{1}{N}\overrightarrow{X^*}\sum_{q=0}^{N-1}\left[e^{\frac{-j2\pi}{N}\left(\frac{f}{f_{\text{nom}}}+1\right)}\right]^{q+k}$$

$$\tilde{\vec{X}} = \frac{1}{N}\vec{X}\left[e^{\frac{j2\pi}{N}\left(\frac{f}{f_{\text{nom}}}-1\right)}\right]^{k}\sum_{q=0}^{N-1}\left[e^{\frac{j2\pi}{N}\left(\frac{f}{f_{\text{nom}}}-1\right)}\right]^{q} + \frac{1}{N}\overrightarrow{X^*}\left[e^{\frac{-j2\pi}{N}\left(\frac{f}{f_{\text{nom}}}+1\right)}\right]^{k}\sum_{q=0}^{N-1}\left[e^{\frac{-j2\pi}{N}\left(\frac{f}{f_{\text{nom}}}+1\right)}\right]^{q}$$

Using closed form summation formula:

$$\sum_{n=0}^{N-1}(e^{j\theta})^n = \frac{\sin\left[\frac{N\theta}{2}\right]}{\sin\left(\frac{\theta}{2}\right)}e^{j(N-1)\frac{\theta}{2}}$$

We can write the estimate of phasor for the off-nominal frequency signal as:

$$\text{First term} = \vec{X}\left[e^{\frac{j2\pi}{N}\left(\frac{f}{f_{\text{nom}}}-1\right)}\right]^{k}\frac{\sin\left[\frac{N}{2}\frac{2\pi}{N}\left(\frac{f}{f_{\text{nom}}}-1\right)\right]}{N\sin\left(\frac{1}{2}\frac{2\pi}{N}\left(\frac{f}{f_{\text{nom}}}-1\right)\right)}e^{j(N-1)\frac{1}{2}\frac{2\pi}{N}\left(\frac{f}{f_{\text{nom}}}-1\right)}$$

$$\text{Second term} = \overrightarrow{X^*}\left[e^{\frac{-j2\pi}{N}\left(\frac{f}{f_{\text{nom}}}+1\right)}\right]^{k}\frac{\sin\left[\frac{N}{2}\frac{-2\pi}{N}\left(\frac{f}{f_{\text{nom}}}+1\right)\right]}{N\sin\left(\frac{-1}{2}\frac{2\pi}{N}\left(\frac{f}{f_{\text{nom}}}+1\right)\right)}e^{j(N-1)\frac{-1}{2}\frac{2\pi}{N}\left(\frac{f}{f_{\text{nom}}}+1\right)}$$

Simplifying them as:

$$\text{First term} = \vec{X}\left[e^{\frac{j2\pi}{N}\left(\frac{f}{f_{\text{nom}}}-1\right)}\right]^{k}\frac{\sin\left[\frac{\pi}{1}\left(\frac{f}{f_{\text{nom}}}-1\right)\right]}{N\sin\left(\frac{\pi}{N}\left(\frac{f}{f_{\text{nom}}}-1\right)\right)}e^{j(N-1)\frac{\pi}{N}\left(\frac{f}{f_{\text{nom}}}-1\right)}$$

Noting that

$$\left[\left(\frac{f}{f_{\text{nom}}}-1\right)\right] = \left[\left(\frac{f-f_{\text{nom}}}{f_{\text{nom}}}\right)\right] = \left[\left(\frac{\omega-\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)\right] \text{ and } \left(\frac{f}{f_{\text{nom}}}+1\right) = \left(\frac{\omega+\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)$$

$$\text{First term} = \vec{X}\, e^{\frac{j2\pi k}{N}\left(\frac{\omega-\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)} \frac{\sin\left[\pi\left(\frac{\omega-\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)\right]}{N\sin\left(\frac{\pi}{N}\left(\frac{\omega-\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)\right)} e^{j(N-1)\frac{\pi}{N}\left(\frac{\omega-\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)}$$

Similarly,

$$\text{Second term} = \vec{X}\, e^{\frac{-j2\pi k}{N}\left(\frac{\omega+\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)} \frac{\sin\left[\pi\left(\frac{\omega+\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)\right]}{N\sin\left(\frac{\pi}{N}\left(\frac{\omega+\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)\right)} e^{-j(N-1)\frac{\pi}{N}\left(\frac{\omega+\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)}$$

Hence, expression for the off-nominal frequency DFT in terms of the true DFT $\vec{X}$ can be written as:

$$\hat{\vec{X}} = A\vec{X}\, e^{\frac{j2\pi k}{N}\left(\frac{\omega-\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)} + B\vec{X}^{*} e^{\frac{-j2\pi k}{N}\left(\frac{\omega+\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)}$$

where

$$A = \frac{\sin\left[\pi\left(\frac{\omega-\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)\right]}{N\sin\left(\frac{\pi}{N}\left(\frac{\omega-\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)\right)} e^{j(N-1)\frac{\pi}{N}\left(\frac{\omega-\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)}$$

and

$$B = \frac{\sin\left[\pi\left(\frac{\omega+\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)\right]}{N\sin\left(\frac{\pi}{N}\left(\frac{\omega+\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)\right)} e^{-j(N-1)\frac{\pi}{N}\left(\frac{\omega+\omega_{\text{nom}}}{\omega_{\text{nom}}}\right)}$$

The variation of magnitudes of '*A*' and '*B*' as f_{nom} changes from 45 Hz to 55 Hz, i.e., a change of ∓5 Hz as shown in Figures 11.10 and 11.11.

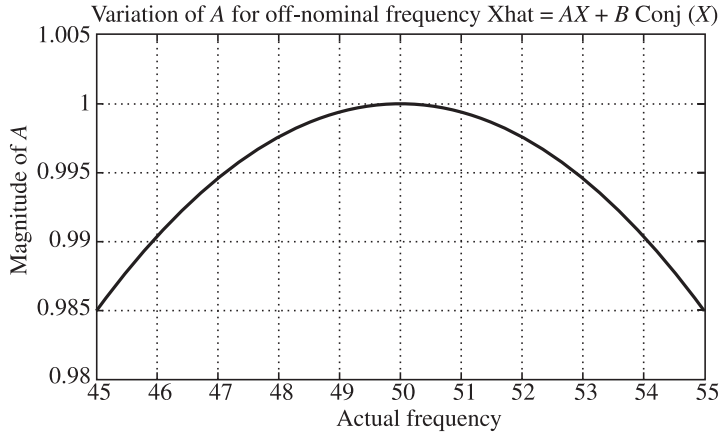


Figure 11.10 Variation of '*A*' for off-nominal frequency.

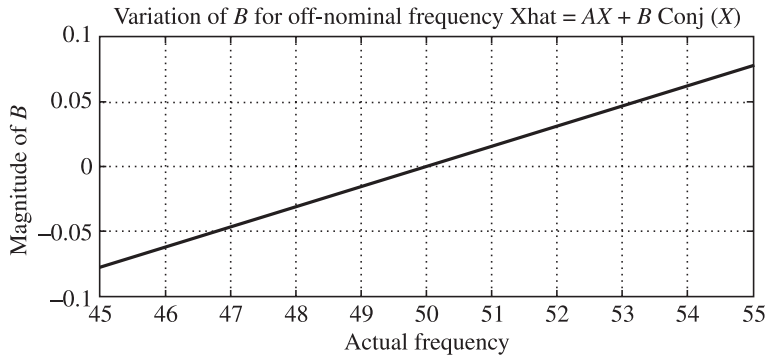


Figure 11.11 Variation of '*B*' for off-nominal frequency.

Further it can be shown that the locus of all phasor estimates, for a given phasor amplitude and variable phase angle which is a circle at the nominal frequency happens to be an ellipse at off-nominal frequency.

11.8.3 A MATLAB Script to Demonstrate Off-nominal Frequency DFT

In this MATLAB script we demonstrate the effect of off-nominal frequency upon the DFT magnitude calculation when the DFT computation is based on the samples drawn at multiples of nominal frequency. In this script we sample a 50 Hz, a 45 Hz and a 55 Hz signal at 4×50 Hz. We keep the amplitude of the signal constant but change its phase from 0 to 360°. As expected the 50 Hz signal traces out a circle but the 45 Hz and 55 Hz signals trace out ellipses. Thus, off-nominal frequency operation of DFT whose sampling is locked to nominal frequency gives rise to error in phasor calculation.

```
% Demonstration of off-nominal frequency operation of DFT
% Comparison between DFT of 50 Hz sig and signals of 45 Hz and 55 Hz
% all sampled at 4*50 Hz
clf , clear , clc
f0 = 50; f1 = 45; f2 = 55; N = 4;
fsamp = N*f0;
dT = 1/fsamp;
phase = 0 ;
ph_step = pi/36;
p = 0;
for phase = 0 : ph_step : 2*pi
 p = p+1;
 ph_angle(p) = phase;
 n = 0;
for k = 0 :N-1
 n = n+1;
 v50(n) = 100*cos((2*pi*f0*k*dT)+(phase));
 voff_nom1(n) = 100*cos((2*pi*f1*k*dT)+(phase));
 voff_nom2(n) = 100*cos((2*pi*f2*k*dT)+(phase));
end
```

```
v50;
voff_nom1;
V50spectrum = fft(v50);
V50_complex(p) = (2/N)*V50spectrum(2);
V50mag(p) = abs(V50_complex(p));

voff_nom1spectrum = fft(voff_nom1);
voff_nom1_complex(p) = (2/N)*voff_nom1spectrum(2);
voff_nom1mag(p) = abs(voff_nom1_complex(p));
err_mag(p) = voff_nom1mag(p)-100;
voff_nom1angle(p) = (angle(voff_nom1_complex(p)))*(180/pi);

voff_nom2spectrum = fft(voff_nom2);
voff_nom2_complex(p) = (2/N)*voff_nom2spectrum(2);
voff_nom2mag(p) = abs(voff_nom2_complex(p));
err_mag(p) = voff_nom2mag(p)-100;
voff_nom2angle(p) = (angle(voff_nom2_complex(p)))*(180/pi);
end


figure(1)
polar(ph_angle,V50mag)
title('Variation magnitude w.r.t. phase for 50Hz signal sampled at
4*50 Hz')
figure(2)
polar(ph_angle,voff_nom1mag)
title('Variation magnitude w.r.t. phase for 45Hz signal sampled at
4*50 Hz')
figure(3)
polar(ph_angle,voff_nom2mag)
title('Variation magnitude w.r.t. phase for 55Hz signal sampled at
4*50 Hz')
```

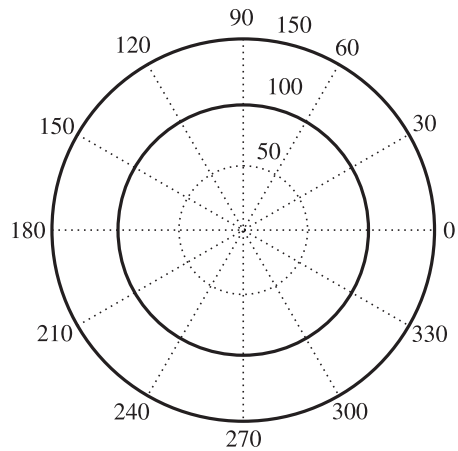


Figure 11.12 Variation of magnitude w.r.t. phase for 50 Hz signal sampled at 4×50 Hz.

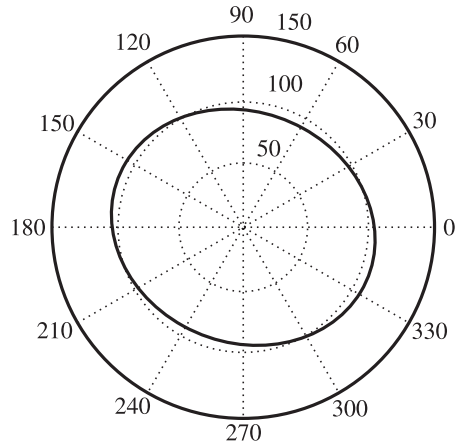


Figure 11.13 Variation of magnitude w.r.t. phase for 45 Hz signal sampled at 4×50 Hz.

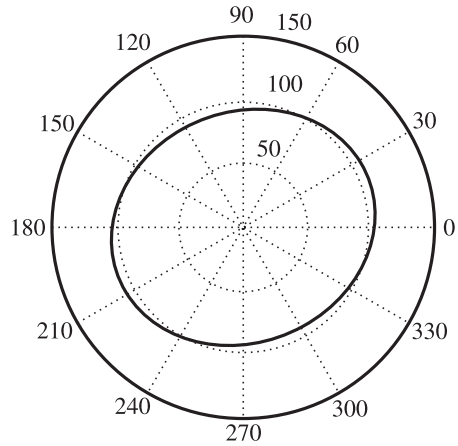


Figure 11.14 Variation of magnitude w.r.t. phase for 55 Hz signal sampled at 4×50 Hz.

11.9 Synchrophasor Applications

Power system is spread out over a very large geographical area. Normally majority of protective relays make use of locally available information. Carrier aided schemes make use of information at the remote end to take a more robust decision. Now, with proliferation of synchrophasor technology, information from diverse locations can be made available anywhere in the system to enhance the relaying decision. This opens up a large number of novel applications which can provide wide area protection and control. Some of these applications are as follows:

1. State estimation: The present state estimation approach consists in measuring active and reactive power flows and voltage magnitudes at substations, and then communicating them to a central site for processing. The data is acquired in a time window of seconds to minutes.

Thus, the calculated state is at best an approximation to an averaged system state. Hence, the estimates that are produced by the state estimation program are called *static state estimates*.

Synchronised phasor measurements of positive sequence bus voltage and currents most naturally lead to state estimates. In the utopian view of the power system, every location is equipped with synchrophasor measurements and information from all is available at a central location. Thus, we have is a real-time state measurement rather than state estimation. However, it is not necessary to have synchrophasor at every location to perform state estimation. It can be shown that we can take measurements from a few key locations and perform state estimation. This is possible because we can divide the network into 'observable locations' where synchrophasor are located and 'unobservable locations' where synchrophasor are not located but estimates can be indirectly calculated. From estimated values at observable locations, we can make close estimates at unobservable locations.

2. Power system control: Prior to synchrophasor measurements, all control in power systems used local measurements and a mathematical model of the larger power system was used to arrive at control decisions. It was recognised that such controllers are rarely optimum and could produce completely unacceptable response to system phenomenon when the models are inaccurate. Synchrophasor measurements can improve such control actions. Typical applications to problems where the control objectives were global in nature, i.e., for example an HVDC controller may be called upon to damp electromechanical oscillations between two widely separated areas of the power system. The high speed synchronised phasor measurements provided by PMU offers a very attractive alternative to the problem of power system controllers. Remote measurements could be brought to the controlled device at high speed, and the remote inputs used as feedback signals in the controller. Phasor data are being used increasingly by individual utilities to manage real-time grid operations.

3. Wide area power system protection: An additional protection area where phasor measurements play a role is that of adaptive security and dependability. The existing protection system has redundant primary protection coupled with multiple backup schemes. The resulting system is highly dependable in that virtually every fault is ultimately cleared. The trade-off is false trips which are tolerated. As the system has evolved, however, the fact that false trips during large disturbances exacerbate the disturbance and allow it to cascade has been recognised. The solution is to adaptively alter the security dependability balance during times of stress. Remote phasor measurements can be used to determine when this happens. The mechanism is to alter the relaying logic which normally lets any relay which sees fault to trip the breaker to logic that demands a vote such as two out of three. This controlled dependability-security system is an example of 'adaptive' relaying. Other adaptive relaying applications include distance relay zone adjustments, transformer protections, reclosing operations, out-of-step relays, etc. Another popular innovation in modern protection systems is the increasing use of remedial actions schemes (RAS) now renamed as system integrity protection (SIPS). These schemes use wide area measurements to identify system conditions under which extraordinary actions must be taken to avoid major system outages. Since the synchrophasor measurement provides very detailed and precise time synchronised measurements of the state of the electrical grid, the analysis of past electricity blackouts has indicated that monitoring synchrophasors has the potential to detect system instability much earlier than current supervisory control and

data acquisition (SCADA) systems. This will prevent large-scale blackouts by giving system operators more time to respond to disruptive events enabling them to take corrective action before system stability is affected. Synchrophasor projects involving the installation of hundreds of phasor measurement units (PMUs) are currently taking place throughout the globe.

4. Forensic analysis: The dictionary meaning of the word forensic is 'relating to the application of science to decide questions arising from crime or litigation'. In case of major system events like blackouts, the data provided by PMUs becomes the basis on which the disturbance can be analysed. Since, PMUs collect and store high volumes of high-speed time synchronised data about the conditions across an interconnection, those data can be compiled quickly and analysed to determine the sequence of events which caused the disturbance. Phasor data were essential in the recent investigations of blackouts in Europe and the USA.

5. Smart grid: At the transmission and generation level, synchrophasor systems are the single most effective technology element to realise and implement smart grid, because synchrophasor systems collect, distribute and analyse critical data and convert it in real time into information and insight that improve grid automation and operation.

Review Questions

1. Explain the concept of a phasor.
2. What is the difference between a phasor and a synchrophasor?
3. Discuss the importance of a global reference time signal for the synchrophasor.
4. What are the various ways in which standard time signal can be disseminated?
5. Describe the GPS system.
6. Describe the time stamp generated by the PMU.
7. What is the effect of off-nominal frequency operation of the power system upon the computation of the synchrophasor?
8. Calculate the error in calculation of phasor when the actual frequency is 51 Hz. Assume that the sampling clock is locked to 50 Hz and sampling at 24 samples per cycle of 50 Hz.
9. Repeat question 8 when the actual system frequency is 49 Hz.
10. What are the applications of the synchrophasor technique in a large power system?
11. What is the relationship of the synchrophasors with the state estimation of the power system?
12. How does the data generated by the PMUs help in forensic analysis of the power system event?
13. What are the standard synchrophasor reporting rates in a 50 Hz system?
14. For a synchrophasor reporting rate of 10 times per second, what is the maximum deviation in frequency that can be reported without error?
15. How frequency and rate of change of frequency are defined in the synchrophasor standard?
16. Explain the concept of total vector error (TVE).

12

Removal of DC Offset

12.1 Why Decaying DC Offsets are Created? [61]

A typical power system is predominantly inductive in nature. If we ignore the shunt capacitances, the system can be represented as a series R-L circuit. It is well known that whenever there is a switching or sudden change in an R-L circuit, it is accompanied by a transient which dies down at a rate decided by the time constant of the R-L circuit. For the sake of relaying, we model the system as shown in Figure 12.1. The voltage is expressed as $v(t) = V_m \sin(\omega t + \alpha)$; where α is the switching angle which decide the fault instant with respect to the zero crossing of the voltage waveform.

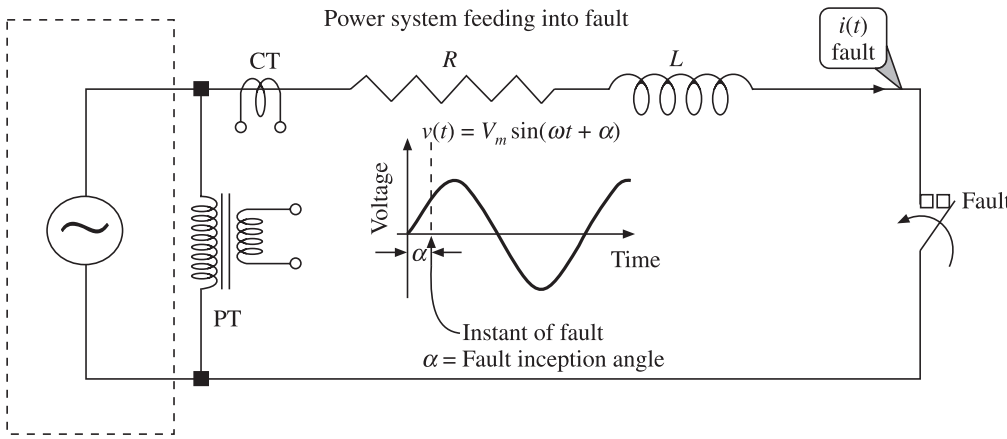


Figure 12.1 Model of power system.

The current in the above circuit is given by writing the KVL around the loop as:

$$v(t) = R\,i(t) + L\frac{di(t)}{dt}$$

Substituting the value of $v(t)$, i.e.,

$$V_m \sin(\omega t + \alpha) = Ri(t) + L\frac{di(t)}{dt}$$

Solving the above differential equation for $i(t)$ gives:

$$i(t) = I_m \sin(\omega t + \alpha - \phi) - I_m \sin(\alpha - \phi)e^{-t/\tau}$$

Where $I_m = \frac{V_m}{Z}$; $Z = \sqrt{R^2 + X^2}$; $\phi = \tan^{-1}\left(\frac{\omega L}{R}\right)$ and $\tau = \frac{L}{R}$

Thus, it can be seen from the expression for $i(t)$ that when $\alpha - \phi = \pm n\pi$; $n = 0, 1, 2,\ldots$, $\sin(\alpha - \phi) = \sin(\pm n\pi) = 0$, hence there will be no DC offset. The DC offset will be maximum when $\alpha - \phi = \pm n\pi/2$. It may be noted that $I_m \sin(\omega t + \alpha - \phi)$ is the steady state fault current and $I_m e^{-t/\tau} \sin(\alpha - \phi)$ is the transient fault current also called as the *decaying DC offset*. It is well known that decaying DC term will become insignificant after a time equal to 5 time constants after the instant of disturbance. Hence, DC offset is important only when the current values immediately after the fault are to be used as in case of high speed distance relays. For many slow applications, it may not be necessary to take cognizance of the DC offset. The way we have modelled the power system as shown in Figure 12.1, it is clear that there cannot be any DC offset in the voltage. However, in practise DC offset can exist in voltage as well.

Figure 12.2(a) shows steady state waveforms of voltage and current. The current lags the voltage by 90° in the steady state. In Figure 12.2(b), the instant of switching is chosen at the positive going zero crossing of voltage waveform giving rise to maximum DC offset in the current.

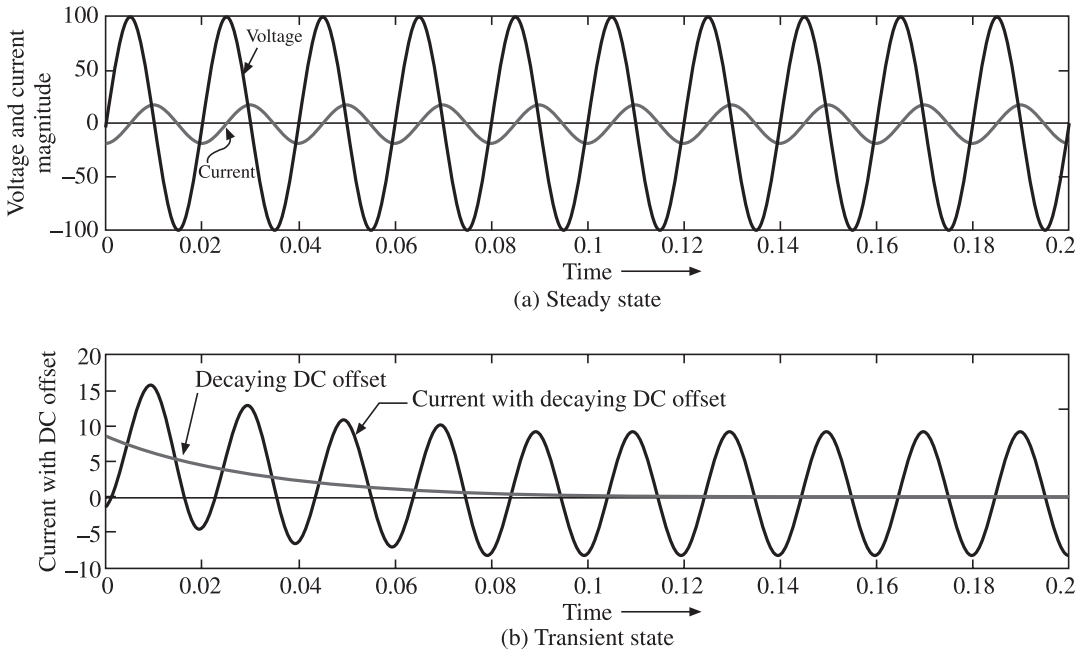


Figure 12.2 Decaying DC offset in current.

12.2 What is the Effect of Decaying DC Offsets up on Relays? [33, 34]

The expression for current immediately following the fault can be written as:

$$i(t) = i_{\text{AC}}(t) + i_{\text{DC}}(t)$$

where

$$i_{\text{AC}} = I_m \sin(\omega t + \alpha - \phi)$$

$$i_{\text{DC}} = -I_m e^{-t/\tau} \sin(\alpha - \phi)$$

Hence, the RMS value of the current can be written as:

$$I_{\text{RMS}} = \sqrt{I_{\text{AC}}^2 + I_{\text{DC}}^2}$$

Thus, it can be seen that the RMS value of current with DC offset is more than the RMS value of the pure AC current. This can cause fast acting over current as well as distance relays to over-reach. Further, presence of decaying DC offset is troublesome for the DFT algorithm. The classical DFT algorithm which is based on Fourier analysis gives correct results when a constant DC is present in the signal but gives erroneous results when decaying DC offset is present. This is because while a constant DC (in time domain) shows up at only zero frequency (in frequency domain), a decaying DC offset in the time domain has a wider frequency spectrum and thus corrupts all the DFT bins. This can be seen from the Fourier transform of $e^{-\alpha t}$, denoted by FT( ), as shown below:

$$\text{FT}(e^{-\alpha t}) = \frac{1}{\alpha + j\omega}$$

Numerous researchers have worked on this problem and there is a large body of literature on this topic. We discuss three approaches is the next sections. Interested reader is urged to follow the literature on this subject.

12.2.1 Mimic Impedance Method of Removal of DC Offset

Traditionally DC offset has been successfully dealt with by connecting mimic impedance in the CT secondary circuit which has the same L/R ratio as the primary circuit. Figure 12.3 shows the response of mimic impedance to the exponentially decaying part of the CT secondary current.

The net voltage developed across the mimic impedance can be written as:

$$v(t) = KR\, i(t) + KL \frac{di(t)}{dt}$$

substituting the decaying DC offset given by $I_0 e^{-t/\tau}$; we get:

$$v_{\text{mimic}} = KR\, I_0 e^{-t/\tau} + K\, L \frac{d\, I_0 e^{-t/\tau}}{dt}$$

$$v_{\text{mimic}} = KR\, I_0 e^{-t/\tau} - KL\, I_0 \frac{1}{\tau} e^{-t/\tau}$$

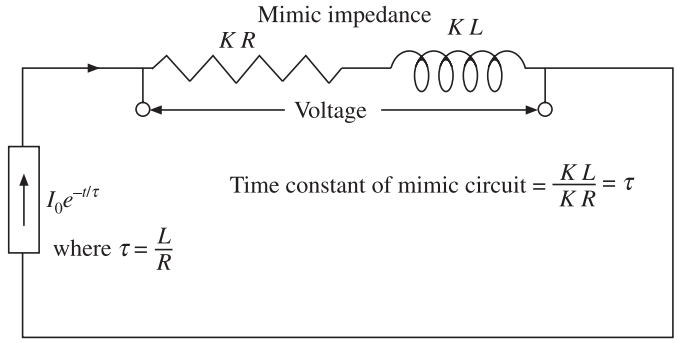


Figure 12.3 Response of mimic impedance to decaying DC offset.

Noting that $\tau = \dfrac{L}{R}$

$$v_{\text{mimic}} = KR I_0 e^{-t/\tau} - KL I_0 \frac{R}{L} e^{-t/\tau}$$

$$v_{\text{mimic}} = KR I_0 e^{-t/\tau} - KR I_0 \ e^{-t/\tau} = 0$$

Thus, the mimic impedance completely rejects the decaying DC offset. However, one must remember that this complete rejection was made possible because the mimic circuit parameters had been exactly matched with the time constant of the decaying DC current. This may not always be possible and may lead to incomplete rejection of the DC offset.

12.2.2 Immunity of LSQ Method to Decaying DC Offset

Readers will recall that in the derivation of the LSQ algorithm, we had assumed the signal consisting of a decaying DC offset in addition to the fundamental and harmonics. The curve-fitting done by the LSQ algorithm was under this explicit assumption. Hence, LSQ algorithm can successfully filter out the DC offset. Therefore, presence of decaying DC offset in the relaying signals is not detrimental to the operation of the LSQ algorithm.

12.2.3 Immunity of Differential Equation Algorithm to DC Offsets

In the differential equation algorithm, we use the instantaneous values of voltage and current samples. The equation $v(t) = R\,i(t) + L(di/dt)$ is valid without any regard to the presence of a decaying DC offset. Hence, methods based on solution of differential equation (DEA) inherently accommodate the DC offset.

12.3 Digital Mimic Impedance Based Filter [3]

In the mimic impedance method the CT secondary current signal is passed through the mimic impedance giving the voltage across the mimic impedance as:

$$v(t) = KR\,i(t) + KL\frac{di(t)}{dt}$$

the equation can be discretised as:

$$v(t_k) = KR\,i(t_k) + K\,L\frac{di(t_k)}{dt}$$

$$v(t_k) = KR\,i(t_k) + K\,L\left(\frac{i(t_k) - i(t_{k-1})}{\Delta t}\right)$$

Taking *Z*-transform of the above equation gives:

$$V(z) = K\,R\,I(z) + \frac{K\,L}{\Delta t}\,[I(z) - Z^{-1}I(z)]$$

$$V(z) = K\,R\,I(z) + \frac{K\,L\,I(z)}{\Delta t}\,[1 - Z^{-1}]$$

$$V(z) = K\,R\left\{I(z) + \frac{L\,I(z)}{R\,\Delta t}\,[1 - Z^{-1}]\right\};\ \text{Noting that}\ \frac{L}{R} = \tau$$

$$V(z) = K\,R\left\{I(z) + \frac{\tau\,I(z)}{\Delta t}\,[1 - Z^{-1}]\right\};\ \text{representing}\ \frac{\tau}{\Delta\tau} = \tau_1$$

$$V(z) = K\,R\,I(z)\{1 + \tau_1[1 - Z^{-1}]\}$$

$$\frac{V(z)}{I(z)} = K\,R\ \{1 + \tau_1[1 - Z^{-1}]\}$$

$$\frac{V(z)}{I(z)} = H(z) = K\,R\{(1 + \tau_1) - \tau_1 Z^{-1}]\}$$

which is written as:

$$\frac{V(z)}{I(z)} = H(z) = K_m\{(1 + \tau_1) - \tau_1 Z^{-1}]\}$$

where $K_m = K\,R$ can be thought of as a gain term.

The gain K_m is set in such a way that at rated frequency, the filter gain will be one. Substituting $Z = e^{j\,\omega T} = e^{j\,2\,\pi\,f\,T} = e^{j\,2\,\pi\,50\,T}$; we can write:

$$\text{Gain at } 50 \text{ Hz} = K_m\{(1 + \tau_1) - \tau_1 e^{-j\,2\pi\,50T}]\}$$

where T is the time period corresponding to the sampling frequency which can be expressed as $T = (1/f_{\text{sampling}})$. Hence,

$$K_m\{(1 + \tau_1) - \tau_1\,e^{-j\,2\,\pi\,50\,/f_{\text{sampling}}}\} = 1$$

$$K_m\left\{(1 + \tau_1) - \tau_1\cos\left(\frac{2\pi 50}{f_{\text{sampling}}}\right) + j\,\tau_1\sin\left(\frac{2\pi 50}{f_{\text{sampling}}}\right)\right\} = 1$$

$$K_m\left\{\sqrt{\left[(1 + \tau_1) - \tau_1\cos\left(\frac{2\pi 50}{f_{\text{sampling}}}\right)\right]^2 + \left[\tau_1\sin\left(\frac{2\pi 50}{f_{\text{sampling}}}\right)\right]^2}\right\} = 1$$

$$K_m = \frac{1}{\sqrt{\left[(1+\tau_1) - \tau_1 \cos\left(\frac{2\pi 50}{f_{\text{sampling}}}\right)\right]^2 + \left[\tau_1 \sin\left(\frac{2\pi 50}{f_{\text{sampling}}}\right)\right]^2}}$$

It can be easily appreciated that the mimic filter is a high-pass filter will therefore amplify any noise contained in the CT secondary current.

12.4 Method of Partial Sums for Filtering DC Offsets [12, 40]

A signal containing a decaying DC offset, fundamental and upto $(N/2 - 1)$ order harmonics can be represented in its discrete form as:

$$S_k = A_0 r^k + \sum_{m=1}^{\frac{N}{2}-1} A_m \sin\left(\frac{2\pi k}{N} m + \varphi_m\right)$$

where $k = 0$ to $(N - 1)$, $A_m = V_m/Z$, $A_0 = -A_m \sin(\alpha - \phi)$ and $r = e^{-(\Delta t/\tau)}$.

If we define a partial sum PS_1 as:

$$PS_1 = S_1 + S_3 + \cdots + S_{N-1}$$

then we can write PS_1 as:

$$PS_1 = \sum_{k=1}^{N/2} A_0 r^{2k-1} + \sum_{k=1}^{N/2} \sum_{m=1}^{((N/2)-1)} A_m \sin\left(\frac{2\pi(2k-1)}{N} m + \varphi_m\right)$$

It can be shown that:

$$\sum_{k=1}^{N/2} \sum_{m=1}^{((N/2)-1)} \sin\left(\frac{2\pi(2k-1)}{N} m + \varphi_m\right) = 0$$

Thus, giving

$$PS_1 = \sum_{k=1}^{N/2} A_0 r^{2k-1}$$

which can be written in the form highlighting the underlying geometric series as:

$$PS_1 = A_0 r^{-1} \sum_{k=1}^{N/2} r^{2k}$$

$$PS_1 = A_0 r^{-1} \sum_{k=1}^{N/2} (r^2)^k$$

Using closed form formula for summation of geometric series, we get:

$$PS_1 = A_0 r^{-1} \frac{(r^2)^{\frac{N}{2}+1} - r^2}{r^2 - 1}$$

$$PS_1 = A_0\, r^{-1}\, \frac{r^N r^2 - r^2}{r^2 - 1}$$

$$PS_1 = A_0\, \frac{r^N r - r}{r^2 - 1}$$

$$PS_1 = A_0\, \frac{r(r^N - 1)}{r^2 - 1}$$

Similarly, we can define another partial sum PS_2 as:

$$PS_2 = S_0 + S_2 + S_4 + \cdots + S_{N-2}$$

which can be expressed as:

$$PS_2 = \sum_{k=0}^{\frac{N}{2}-1} A_0\, r^{2k} + \sum_{k=0}^{\frac{N}{2}-1} \sum_{m=1}^{\frac{N}{2}-1} A_m \sin\left(\frac{2\pi(2k)}{N} m + \varphi_m\right)$$

Using similar reasoning, the second term in the above equation turns out to be zero, giving:

$$PS_2 = \sum_{k=0}^{\frac{N}{2}-1} A_0\, r^{2k}$$

Putting in the form of sum of geometric series as:

$$PS_2 = A_0 \sum_{k=0}^{\frac{N}{2}-1} (r^2)^k$$

$$PS_2 = A_0\, \frac{(r^2)^{N/2} - (r^2)^0}{r^2 - 1}$$

$$PS_2 = A_0\, \frac{r^N - 1}{r^2 - 1}$$

From equations for PS_1 and PS_2, we can write:

$$r = \frac{PS_1}{PS_2}$$

and

$$A_0 = \frac{r^2 - 1}{(r^N - 1)} PS_2$$

Having the values of 'r' and 'A_0', the samples can be modified as:

$$S_{k-\text{new}} = S_k - A_0 r^k;\ \ k = 0 \text{ to } N - 1$$

This, new set of samples, is free from decaying DC offset and can be used for further processing.

12.5 Characterising the Decaying DC Offset by Integration [6]

Let us express the fault current as shown below:

$$i(t) = I_0 e^{-t/\tau} + \sum_{k=1}^{p} (I_k \sin(k\omega_1 t + \theta_k))$$

where I_0 is the peak value of the DC offset, τ is the time constant of the decaying DC offset, 'k' is the harmonic order, I_k is the peak value of the kth harmonic and θ_k is the phase angle of the kth harmonic component and 'p' is the maximum harmonic order. If the above expression is integrated for one time period 'T' of the fundamental, then second term evaluates to zero, and we are left with integral of the decaying DC offset term as shown below:

$$\int_{t-T}^{t} i(t)dt = \int_{t-T}^{t} \left(I_0 e^{-t/\tau} + \sum_{k=1}^{p} \left(I_k \sin(k\omega_1 t + \theta_k) \right) \right) dt$$

$$= \int_{t-T}^{t} I_0 e^{-t/\tau} dt = -I_0 \tau \, [e^{-t/\tau}]_{t-T}^{t}$$

$$= -I_0 \tau \, (e^{-t/\tau} - e^{-(t-T)/\tau}) = -I_0 \tau (e^{-t/\tau} - e^{-t/\tau} e^{T/\tau})$$

$$= -I_0 \tau e^{-t/\tau} (1 - e^{T/\tau})$$

Let us call the above result of the integral as $Z(t)$. Hence,

$$Z(t) = -I_0 \tau e^{-t/\tau} (1 - e^{T/\tau})$$

Now, the expression for $Z(t + \Delta t)$, which is the integral of the DC component $Z(t)$ extended by a small interval of time Δt can be written as:

$$Z(t + \Delta t) = -I_0 \tau e^{-(t+\Delta t)/\tau} \, (1 - e^{T/\tau})$$

$$Z(t + \Delta t) = \underbrace{-I_0 \tau e^{-t/\tau} (1 - e^{T/\tau})}_{Z(t)} e^{-\Delta t/\tau}$$

Thus, we can write $Z(t + \Delta t)$ in terms of $Z(t)$ as:

$$Z(t + \Delta t) = Z(t) e^{-\Delta t/\tau}$$

Simplifying further,

$$-\frac{\Delta t}{\tau} = \ln \frac{Z(t + \Delta t)}{Z(t)}$$

Thus, the time constant of the decaying DC offset τ is found out as:

$$\tau = -\frac{\Delta t}{\ln \dfrac{Z(t + \Delta t)}{Z(t)}}$$

and I_0 can be found out as:

$$I_0 = \frac{Z(t)}{-\tau e^{-t/\tau}(1 - e^{T/\tau})}$$

Thus, once I_0 and τ are found out, we can completely characterise the decaying DC offset and compensate the raw signals to extract decaying offset free part of the signal as explained earlier.

Review Questions

1. Clearly state the difference between (constant) DC offset and decaying DC offset.
2. Sketch the frequency spectra of a decaying DC offset, thus explain why it corrupts all the DFT 'bins'.
3. DFT cannot filter out the effect of decaying DC offset while it does so for constant DC offset. Explain.
4. Describe how decaying DC offsets are created in AC power system.
5. What is the problem caused by existence of decaying DC offsets?
6. Explain whether decaying DC offset is present in output of CT or output of PT or both.
7. Out of current and voltage signals which one is more likely to exhibit decaying DC offset?
8. Explain the effect of decaying DC offset on OC relays.
9. Explain the effect of decaying DC offset on distance relays.
10. Explain the mimic impedance method of filtering decaying DC offset.
11. Explain the digital implementation of the mimic impedance for filtering out decaying DC offset.
12. Explain why the mimic impedance acts as a high-pass filter. What are the implications of this?
13. Explain the method of partial sums to filter out decaying DC offset.
14. Explain how DC offset can be characterised by integration over the fundamental time period, and this can be used to filter out the decaying DC offset.

Appendix

A.1 The GPS system

GPS (geographical positioning system) is a system of satellites owned and operated by the U.S. Department of Defense for the purpose of navigation throughout the world. Being a defense system it is very robust and is designed in such a way that it will provide navigation signal throughout the world without fail in the worst possible scenario. This includes all weather conditions, system degradation and interference. The launching of all the 24 satellites of the GPS system was completed in 1994. It became fully operational in 1995. As of December 2012 there are 32 satellites in eight orbital planes. At any location at least four satellites are in view at any time. Hence, even sites with restricted sky-view keep receiving the GPS signals.

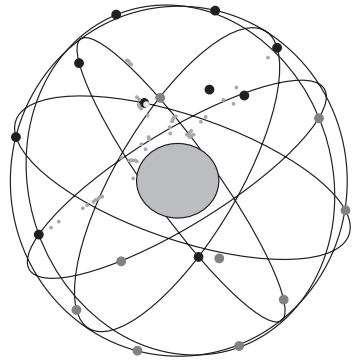


Figure A.1 Orbits of the GPS satellites.

Range and time is broadcast on three L-band frequencies. Time can be derived from the coarse/acquisition (C/A) code transmitted at 1575 MHz. All satellites broadcast at the same frequencies. Signals are encoded using code division multiple access (CDMA) allowing messages from individual satellites to be distinguished from each other based on unique encodings for each satellite (that the receiver must be aware of). Two distinct types of CDMA

encodings are used, i.e., the coarse/acquisition (C/A) code, which is accessible by the general public, and the precise (P(Y)) code, which is encrypted so that only the U.S. military can access it. The signal can be received by a simple omnidirectional antenna. The spread-spectrum technique that is used, makes the signal resistant to interference. At the heart of the GPS is a ground based cesium clock ensemble that itself is referenced to the universal coordinated time (UTC), which is the world time standard. Each satellite provides a correction to the UTC time that the receiver automatically applies to the outputs. With continuous adjustments, the time accuracy is limited only by short term signal reception whose basic accuracy is 0.2 μs. Baseline accuracy can be improved by advanced decoding and processing techniques. For general applications a 0.5 μs is a safer figure to depend on. The short term frequency accuracy of the received signal is of the order of 10E-11.

One might wonder how ordinary public throughout the world can access the U.S. Department of Defense satellite but access issues were largely resolved in the 1990's with agreements to add new civilian frequencies and culminating in termination of the selective availability signal degradation of the C/A code in the year 2000. The U.S. Department of Defence and Department of Transportation have committed to make GPS available to civilian users at all-time except in national emergency.

A.2 COMTRADE

COMTRADE (Common format for Transient Data Exchange for power systems) is a file format for storing oscillography and status data related to transient power system disturbances.

COMTRADE files are typically generated by Intelligent Electronic Devices (IEDs), such as a protective relay, in electrical substations. Analysis tools can download the COMTRADE file and calculate useful information related to the power system. COMTRADE files from multiple substations can be used collectively to perform forensic analysis of large scale power disturbance events (e.g., blackouts) to determine the root cause of the disturbance, help to improve system protection and guide future mitigation strategies.

The COMTRADE file format has been standardised by the Power System Relaying Committee (PSRC) of the IEEE Power & Energy Society as C37.111. The most widely used version of the COMTRADE standard is C37.111-1999. This version specified a file format that consisted of multiple file types designated by the assigned file extensions of *.CFG, *.INF, *.HDR, and *.DAT. The *.DAT file contains the digitised sample data in an ASCII text format. The *.CFG file contains configuration data as in the *.DAT file including information such as signal names, start time of the samples, number of samples, min/max values, and more. Only the *.CFG and *.DAT files were mandatory. Although the values of the digitised samples in the *.DAT file are viewable without the *.CFG file, the value of the data is greatly diminished as it would be very difficult to fully reconstruct the meaning of the data without the *.CFG file.

As applications for using COMTRADE grew the limitations of the file format began to cause problems. For instance, the North American Synchro Phasor Initiative (NASPI) sponsored by the North American Electric Reliability Corporation (NERC) wanted to use the COMTRADE file format to exchange synchrophasor data. The C37.111–1999 version of the COMTRADE format only supported ASCII encoded data, supported limited analog and digital status data

types, and did not convey any power system configuration information. This would have made large scale exchange of synchrophasor data more difficult. Beginning in 2011, the IEEE PSRC began work on a new version of the COMTRADE standard that has been published as C37.111-2013. This version of the standard supports a variety of new features including XML encoding configuration and data, a single file type (*.CFF) that combines all the C37.111-1999 types into a single file, additional data types for 32-bit binary and floating point values and a partial binary encoding to reduce file size.

The IEEE Power Quality Subcommittee of the IEEE Power & Energy Society has also developed a file format called the Power Quality Data Interchange Format (PQDIF) that is similar to COMTRADE in structure but is used primarily to convey power quality data instead of transient disturbance data.

There are numerous COMTRADE viewers available from many commercial companies as well as free viewers like Comcalc Pro, TOP, GTPPLOT and others.

A.3 ANSI/IEEE Standard Device Numbers

In industrial drawings, protective relays are generally referred to by standard device numbers. Letters are sometimes added to specify the application (IEEE Std C37.2-2008).

Device Numbers (the more commonly used ones are in bold)

1 – Master Element
2 – Time Delay Starting or Closing Relay
3 – Checking or Interlocking Relay
4 – Master Contactor
5 – Stopping Device
6 – Starting Circuit Breaker
7 – Rate of Change Relay
8 – Control Power Disconnecting Device
9 – Reversing Device
10 – Unit Sequence Switch
11 – Multifunction Device
12 – Overspeed Device
13 – Synchronous-speed Device
14 – Underspeed Device
15 – Speed or Frequency-Matching Device
16 – Data Communications Device
20 – Electrically Operated valve (solenoid valve)
21 – Distance Relay
23 – Temperature Control Device
24 – Volts Per Hertz Relay

25 – Synchronising or Synchronism-Check Device
26 – Apparatus Thermal Device
27 – Undervoltage Relay
30 – Annunciator Relay
32 – Directional Power Relay
36 – Polarity or Polarising Voltage Devices
37 – Undercurrent or Underpower Relay
38 – Bearing Protective Device
39 – Mechanical Conduction Monitor
40 – Field (over/under excitation) Relay
41 – Field Circuit Breaker
42 – Running Circuit Breaker
43 – Manual Transfer or Selector Device
46 – Reverse-phase or Phase-Balance Current Relay
47 – Phase-Sequence or Phase-Balance Voltage Relay
48 – Incomplete-Sequence Relay
49 – Machine or Transformer Thermal Relay
50 – Instantaneous Overcurrent Relay
51 – AC Time Overcurrent Relay
52 – AC Circuit Breaker
53 – Field Excitation Relay
55 – Power Factor Relay
56 – Field Application Relay
59 – Overvoltage Relay
60 – Voltage or Current Balance Relay
62 – Time-Delay Stopping or Opening Relay
63 – Pressure Switch
64 – Ground Detector Relay
65 – Governor
66 – Notching or jogging device
67 – AC Directional Overcurrent Relay
68 – Blocking or “out of step” Relay
69 – Permissive Control Device
74 – Alarm Relay
75 – Position Changing Mechanism
76 – DC Overcurrent Relay
78 – Phase-Angle Measuring Relay
79 – AC-Reclosing Relay

81 – Frequency Relay
83 – Automatic Selective Control or Transfer Relay
84 – Operating Mechanism
85 – Pilot Communications, Carrier or Pilot-Wire Relay
86 – Lockout Relay
87 – Differential Protective Relay
89 – Line Switch
90 – Regulating Device
91 – Voltage Directional Relay
92 – Voltage and Power Directional Relay
94 – Tripping or Trip-Free Relay
B – Bus
F – Field
G – Ground or generator
N – Neutral
T – Transformer

References

[1] Adamiak, Mark, Bogdan Kasztenny, and William Premerlani, "Synchrophasors: definition, measurement, and application," *Proceedings of the 59th Annual Georgia Tech Protective Relaying, Atlanta, GA*, 2005.

[2] Adelson, Ronald M., "Rapid power-line frequency monitoring," *Digital Signal Processing 12.1*, 2002: 1–11.

[3] Benmouyal, Gabriel, "Removal of DC-offset in current waveforms using digital mimic filtering," *Power Delivery, IEEE Transactions on 10.2,* 1995: 621–630.

[4] Benmouyal, Gabriel, E.O. Schweitzer, and Armando Guzman, "Synchronised phasor measurement in protective relays for protection, control, and analysis of electric power systems," 2004: 814–820.

[5] Bracewell, R.N., *The Fourier Transform and Its Applications*, McGraw-Hill, 1986.

[6] Cho, Yoon-Sung, et al., "An innovative decaying DC component estimation algorithm for digital relaying," *Power Delivery, IEEE Transactions on 24.1*, 2009: 73–78.

[7] De La Ree, Jaime, et al., "Synchronised phasor measurement applications in power systems," *Smart Grid, IEEE Transactions on 1.1,* 2010: 20–27.

[8] Elmore, Walter A., *Protective Relaying: Theory and Applications*, ABB, 1994.

[9] Ferrer, Altvue H.J. and E.O. Schweitzer, III (Eds.), *Modern Solutions for Protection, Control, and Monitoring of Electric Power Systems,* Schweitzer Engineering Laboratories, 2010.

[10] Gilbert, J.G. and R.J. Shovlin, "High speed transmission line fault impedance calculation using a dedicated minicomputer," *Power Apparatus and Systems, IEEE Transactions on 94.3,* 1975: 872–883.

[11] Gilchrist, G.B., G.D. Rockefeller, and E.A. Udren, "High-speed distance relaying using a digital computer I-system description," *Power Apparatus and Systems, IEEE Transactions on 3*, 1972: 1235–1243.

[12] Guo, Yong, Mladen Kezunovic, and Deshu Chen, "Simplified algorithms for removal of the effect of exponentially decaying DC-offset on the Fourier algorithm," *Power Delivery, IEEE Transactions on 18.3*, 2003: 711–717.

[13] Hays, H.H., *Schaum's Outline of Theory and Problems of Digital Signal Processing*, Tata McGraw-Hill, New Delhi, 2004.

[14] Holbert, Keith E., G.I. Heydt, and Hui Ni, "Use of satellite technologies for power system measurements, command, and control," *Proceedings of the IEEE 93.5*, 2005: 947–955.

[15] Hope, G.S. and V.S. Umamaheswaran, "Sampling for computer protection of transmission lines," *Power Apparatus and Systems, IEEE Transactions on* 5, 1974: 1522–1534.

[16] Horowitz, S H., A.G. Phadke, and J.S. Thorp, "Adaptive transmission system relaying," *Power Delivery, IEEE Transactions on 3.4*, 1988: 1436–1445.

[17] Horton, John W, "The use of Walsh functions for high speed digital relaying," IEEE Publication 75CH1034-8, 1975: 1–9.

[18] IEEE Standard for Synchrophasor Measurements for Power Systems, IEEE Std C37.118.1™ –2011 and IEEE Std C37.118.2™ –2011.

[19] Ifeachor, Emmanuel C. and B.W. Jervis, *Digital Signal Processing—A Practical Approach*, Pearson, 2007.

[20] Jeyasurya, B. and W.J. Smolinski, "Identification of a best algorihthm for digital distance protection of transmission lines," *Power Apparatus and Systems, IEEE Transactions on 10*, 1983: 3358–3369.

[21] Johns, A.T. and M.A. Martin, "Fundamental digital approach to the distance protection of EHV transmission lines," *Proceedings of the Institution of Electrical Engineers*, Vol. 125, No. 5, IET Digital Library, 1978.

[22] Lyons, Richard G., *Understanding Digital Signal Processing*, 3rd ed., Prentice Hall, 2010.

[23] Madhav Rao, T.S., *Power System Protection: Static Relays with Microprocessor Application*, 2nd ed., Tata McGraw-Hill, New Delhi, 2001.

[24] Mahmood, Mahir K., Janan E. Allos, and Majid AH Abdul-Karim, "Microprocessor implementation of a fast and simultaneous amplitude and frequency detector for sinusoidal signals," *Instrumentation and Measurement, IEEE Transactions on 34.3*, 1985: 413–417.

[25] Makino, J. and Y. Miki, "Study of operating principles and digital filters for protective relays with digital computer," IEEE PES Winter Power Meeting, 1975.

[26] Mann, B.J. and I.F. Morrison, "Digital calculation of impedance for transmission line protection," *Power Apparatus and Systems, IEEE Transactions on 1*, 1971: 270–279.

[27] Mann, B.J. and I.F. Morrison, "Relaying a three-phase transmission line with a digital computer," *Power Apparatus and Systems, IEEE Transactions on 2*, 1971: 742–750.

[28] Martin, K.E., et al., "IEEE Standard for synchrophasors for power systems,"*Power Delivery, IEEE Transactions on 13.1*, 1998: 73–77.

[29] Mason, C. Russell, *The Art and Science of Protective Relaying,* Wiley, 1956.

[30] McInnes, A.D. and I.F. Morrison, "Real time calculation of resistance and reactance for transmission line protection by digital computer," *IEEE Trans*, 1971: 16–23.

[31] McLaren, Peter Grant and M.A. Redfern, "Fourier-series techniques applied to distance protection," *Proceedings of the Institution of Electrical Engineers*, Vol. 122, No. 11, IET Digital Library, 1975.

[32] North American Synchrophasor Initiative: Synchrophasor System Benefits Factsheet: http://www.naspi.org.

[33] Paithankar, Y.G. and S.R. Bhide, *Fundamentals of Power System Protection,* PHI Learning, New Delhi, 2010.

[34] Paithankar, Y.G., *Transmission Network Protection, Theory and Practise*, CRC Press, 1997.

[35] Phadke, A.G. and J.S. Thorp, *Synchronised Phasor Measurements and Their Applications*, Springer, 2008.

[36] Phadke, A.G. and Bogdan Kasztenny, "Synchronised phasor and frequency measurement under transient conditions," *Power Delivery, IEEE Transactions on 24.1*, 2009: 89–95.

[37] Phadke, A.G. and J.S. Thorp, "History and applications of phasor measurements," *Power Systems Conference and Exposition, 2006, PSCE'06, 2006, IEEE PES*, IEEE, 2006.

[38] Phadke, A.G., et al., "Synchronised sampling and phasor measurements for relaying and control," *Power Delivery, IEEE Transactions on 9.1,* 1994: 442–452.

[39] Phadke, A.G., J.S. Thorp, and M.G. Adamiak, "A new measurement technique for tracking voltage phasors, local system frequency, and rate of change of frequency," *Power Apparatus and Systems, IEEE Transactions on* 5, 1983: 1025–1038.

[40] Phadke, A.G., T. Hlibka, and M. Ibrahim, "A digital computer system for EHV substations: analysis and field tests," *Power Apparatus and Systems, IEEE Transactions on 95.1*, 1976: 291–301.

[41] Phadke, Arun G. and James S. Thorp, *Computer Relaying for Power Systems*, 2nd ed., John Wiley & Sons, 2009.

[42] Pratap, Rudra, *Getting Started with MATLAB 7*, Oxford University Press (Indian Edition), 2006.

[43] Rahman, M.A. and B. Jeyasurya, "A state-of-the-art review of transformer protection algorithms," *Power Delivery, IEEE Transactions on 3.2*, 1988: 534–544.

[44] Ram, Badri, *Power System Protection and Switchgear*, Tata McGraw-Hill, 2011.

[45] Ramamoorty, M., "Application of digital computers to power system protection," *Journal of the Institution of Engineers*, India, 52, 1972: 235–238.

[46] Ranjbar, A.M. and B.J. Cory, "Algorithms for distance protection," *IEEE Conf. on Developments in Power System Protection,* Institution of Electrical Engineers, London, 1975.

[47] Ranjbar, A.M. and B.J. Cory, "An improved method for the digital protection of high voltage transmission lines," *Power Apparatus and Systems, IEEE Transactions on 94.2*, 1975: 544–550.

[48] Rockefeller, G.D. and E.A. Udren, "High-speed distance relaying using a digital computer II-test results," *Power Apparatus and Systems, IEEE Transactions on 3,* 1972: 1244–1258.

[49] Rockefeller, George D., "Fault protection with a digital computer," *Power Apparatus and Systems, IEEE Transactions on 4*, 1969: 438–464.

[50] Sachdev, M.S. and M.A. Baribeau, "A new algorithm for digital impedance relays," *Power Apparatus and Systems, IEEE Transactions on 6*, 1979: 2232–2240.

[51] Scarborough, James Blaine, *Numerical Mathematical Analysis,* Oxford and IBH Publishing, 1955.

[52] Schilling, Robert Joseph and Sandra L. Harris, *Introduction to Digital Signal Processing*, 2nd ed., Cengage Learning, 2011.

[53] Segui, Trinidad, et al., "Fundamental basis for distance relaying with parametrical estimation," *Power Delivery, IEEE Transactions on 16.1*, 2001: 99–104.

[54] Sidhu, T.S., et al., "Discrete-Fourier-transform-based technique for removal of decaying DC offset from phasor estimates," *IEEE Proceedings-Generation, Transmission and Distribution 150.6*, 2003: 745–752.

[55] Singh, Lankeshwar Prakash, *Digital Protection: Protective Relaying from Electromechanical to Microprocessor*, New Age International, 1997.

[56] Smolinski, W.J., "An algorithm for digital impedance calculation using a single PI section transmission line model," *Power Apparatus and Systems, IEEE Transactions on 5*, 1979: 1546–1551.

[57] Soman, S.A., "Power System Protection Course," NPTEL Website.

[58] Strang, Gilbert, *Linear Algebra and Its Applications*, 4th ed., Cengage Learning, 2006.

[59] Thorp, J.S. and A.G. Phadke, "A microprocessor based three-phase transformer differential relay," *Power Apparatus and Systems, IEEE Transactions on* 2, 1982: 426–432.

[60] Thorp, J.S., et al., "Limits to impedance relaying," *Power Apparatus and Systems, IEEE Transactions on 1*, 1979: 246–260.

[61] Van Valkenburg, M.E., *Network Analysis,* Prentice Hall Inc., 1974.

[62] Warrington, AR van C. (Ed.), *Protective Relays Their Theory and Practise*: Volume I, 1962.

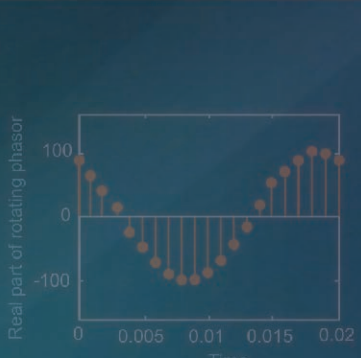
[63] Warrington, AR van C. (Ed.), *Protective Relays Their Theory and Practise*: Volume II, 1969.

[64] Wilson, Robert E., "Uses of precise time and frequency in power systems," *Proceedings of the IEEE* 79.7, 1991: 1009–1018.

[65] Yu, Sun-Li and Jyh-Cherng Gu, "Removal of decaying DC in current and voltage signals using a modified Fourier filter algorithm," *Power Delivery, IEEE Transactions on 16.3*, 2001: 372–379.

[66] Ziegler, Gerhard, *Numerical Distance Protection*, Siemens, Publicis, Germany, 2006.

Index



DIGITAL POWER SYSTEM PROTECTION

S.R. Bhide

Digital power system protection, as a subject, offers the use of computers in power line relaying which is the act of automatically controlling the power system via instrumentation and control devices. This book is an attempt to make a gentle introduction to the nitty-gritty of digital relays. Written in a simple, clear and student-friendly style, this text covers basics of digital processing of analog signals for the purpose of relaying. All important basic algorithms that are used in various types of digital relays have been explained. FIR and IIR filters have been presented in such a manner that students will be able to develop intuitive understanding. The book also covers DFT and FFT and synchrophasor technology in details. MATLAB programs and Excel simulations have been given to reinforce the comprehension of the algorithms.

This book has been thoroughly class-room tested and based on course notes which is primarily intended for undergraduate and postgraduate students of electrical engineering.

KEY FEATURES

- In-depth coverage of DSP fundamentals
- Pedagogical tools like figures, flowcharts, block diagrams and tables have been extensively used
- Review questions are given at the end of each chapter
- Extensive references to literature on power system protection

THE AUTHOR

S.R. BHIDE, Ph.D., is Associate Professor of Electrical Engineering at the Visvesvaraya National Institute of Technology, Nagpur, where he has been serving since 1984. He is a life member of ISTE. His special areas of interest include application techniques of Artificial Intelligence (AI) to power system protection schemes.

You may also be interested in

Electrical Power Systems: Analysis, Security and Deregulation, P. Venkatesh, B.V. Manikandan, S. Charles Raja and A. Srinivasan

***Power System Transients: A Statistical Approach*, 2nd ed.**, C.S. Indulkar, D.P. Kothari and K. Ramalingam

***Power Theft*, 3rd ed.**, G. Sreenivasan

ISBN:978-81-203-4979-7

9 788120 349797

www.phindia.com