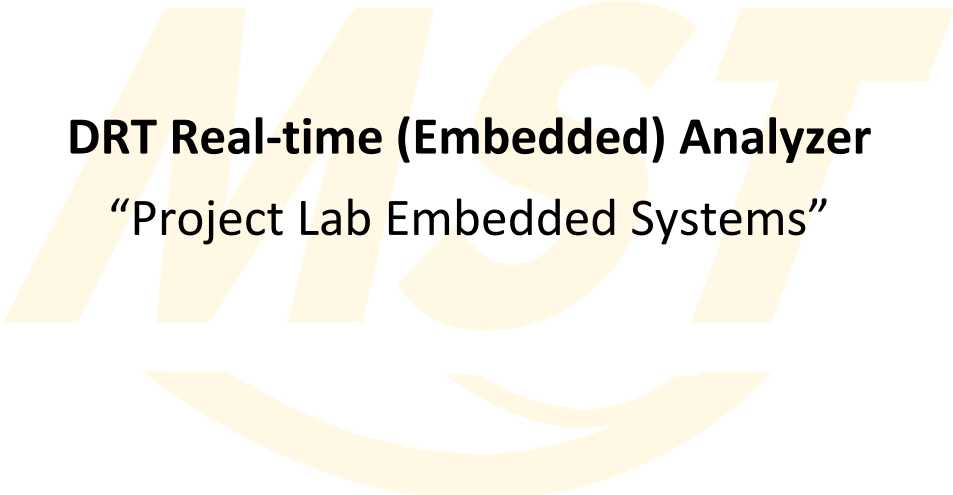


TECHNISCHE UNIVERSITÄT
CHEMNITZ

Department of Electrical Engineering and
Information Technology

Chair of Measurement and Sensor Technology



DRT Real-time (Embedded) Analyzer

“Project Lab Embedded Systems”

Guide:	Ahmed Yahia Kallel
Group:	14
Members:	Ankit Ashok Jaiswal Meghana Vishal Vasant Mhasawade
Date:	10 th August 2018

Contents

Contents	2
Abstract	3
INTRODUCTION	4
1.1 Motivation	4
1.2 Member Responsibilities	5
PROJECT DESCRIPTION	6
2.1 Overview	6
2.2 Software.....	6
2.2.1 Algorithm overview	6
2.2.2 Algorithm Results	9
2.2.3 Algorithm Choosing	23
2.3 Hardware	25
2.3.1 STM32 F4	25
2.3.2 Arduino Uno	26
2.3.3 Liquid Crystal Display (LCD)	27
2.3.4 Device under test (DUT)	28
PROTOTYPE.....	29
3.1 Simulation	31
RESULTS	33
CONCLUSION	35
REFERENCES	i

Abstract

Characterization of batteries is very crucial for applications of battery. This is very important to get the status of the battery, mainly the state-of-charge (SoC), or the remaining charging in the battery, as well as state-of-health (SoH), or how much efficient capacity can the battery handle compared to the factory capacity. Several techniques could be deployed and have been developed. Impedance spectroscopy is a widely known techniques for characterization of batteries. However, by itself, the impedance spectroscopy cannot provide an image of the SoC or SoH of the battery, which requires other additional steps. The most prominent method is modeling, in which an equivalent circuit is used to fit the impedance spectrum of the battery, and the different values of the circuit are used to return the SoC and the SoH. However, model-based approaches have a lot of shortcomings, as the model is dependent on the material used in the battery, as well as different external factors mainly related to battery setup. Another approach is a modeless technique, primarily the Distribution of relaxation times (DRT). The impedance spectra is used as a feed to calculate a vector named distribution of relaxation times (DRT). This DRT spectrum is the time domain representation of the linear battery behaviour. In contrast to the impedance spectra, it can be used to directly simulate the linear response of the battery. In this project special attention is given to the linear iterative algorithms which are employed to solve the mathematically formulated problem of DRT. This report also gives user an idea of the mathematics behind the algorithms used. The behaviour of several algorithms is checked against 2 types of DUTs and results of simulations are analysed.

CHAPTER 1

INTRODUCTION

The performance of the battery is strongly dependent on parameters like state of health, state of charge of battery. And Impedance plays a vital role in calculations of above mentioned parameters. It is very difficult to figure out the impedance of composite structures of battery viz. electrodes. For automotive applications of battery systems, there are special requirements. There are various methods for simulation and modeling of lithium ion batteries. We follow the distribution of relaxation times approach and thereby bridge the gap between mathematical models and coarse equivalent circuit models. It combines straight-forward parameterization and a structure that is readily implementable on microcontrollers with a scalable degree of detail.

For the analysis of impedance spectra of solid oxide fuel cells (SOFC), the method of the distribution of relaxation times has been successfully applied for years and has recently been applied for the analysis of lithium ion cells. Its major benefit is the better separation of processes with different time constants compared to the Nyquist or Bode plot. An impedance spectrum can be written as integral equation, where $g(\tau)$ represents the distribution of relaxation times:

$$Z(\omega) = R_0 + R_{pol} \int_0^{\infty} \frac{g(\tau)}{1+j\omega\tau} d\tau \quad \text{....equation (1)}$$

The method of the distribution of relaxation times has been successfully applied for years for the analysis of impedance spectra of solid oxide fuel cells (SOFC) and has recently been applied for the analysis of lithium ion cells. Its major benefit is the better separation of processes with different time constants compared to the Nyquist or Bode plot.

1.1 Motivation

Performance of a battery is strongly dependent on parameters like state-of-health, state-of-charge of battery. Several methods have been developed in order to estimate back the state-of-health (SoH) and state-of-charge (SoC) of the battery. The process of characterization of the battery parameters is called as parameterization. There are main two methods of Impedance Spectrum parameterization.

- I. Model based techniques
- II. Model less techniques

Model based techniques viz. Impedance spectroscopy, curve fitting is dependent of output of excited design under test (DUT). Output of the excited DUT is projected onto equivalent model. The accuracy of model is crucial for Impedance spectroscopy. If model is not well chosen then impedance spectroscopy results in wrong predictions and resulting fitting error is very high. A change in the setup (different cables, different electrodes) directly affects the Impedance Spectrum and whole model has to be recalibrated.

This serious limitation hints at need to model less approach. Distribution of Relaxation Times (DRT) being one of the model-less approach solves the abovementioned problem. There are several methods for the time domain simulation of battery parameters but those are limited to finite number of RC elements. Infinite number of RC elements in differential forms the base of concept of distribution of relaxation times. The distribution of relaxation times (DRT) links the impedance in the frequency domain to the time domain by using the Hilbert transform. DRT is based on some iterative constructive algorithms. Input of DRT is impedance spectrum of DUT which can be easily figured out by exciting the DUT with signal of known amplitude. DRT decomposes the impedance of DUT into parallel RC components, hence eliminates the need of any mathematical or physical models. DRT also reduces the chances of wrong parameterization by inaccurate model.

The method of the distribution of relaxation times has been successfully applied for years for the analysis of impedance spectra of solid oxide fuel cells (SOFC) and has recently been applied for the analysis of lithium ion cells. Its major benefit is the better separation of processes with different time constants compared to the Nyquist or Bode plot.

1.2 Member Responsibilities

Table 1: Member Responsibilities

Member	Role
Ankit Ashok Jaiswal	Prototype : HW architecture, Ammeter
Meghana	Algorithm Analysis, Prototype: Excitation Signal generation
Vishal Vasant Mhasawade	Algorithm analysis, Simulation

CHAPTER 2

PROJECT DESCRIPTION

2.1 Overview

Embedded DRT analyzer system is the embedded solution to analyse the distribution of relaxation time of device under test (DUT). Scope of project involves simulating the behaviour of several algorithms when impedance spectrum is fed as an input to DRT algorithm. Scope of the system also include comparing different DRT algorithms which are shown below. The comparison is done on the basis of the factors impacting the performance of algorithm and estimation of DUT in the terms of resistance and capacitance.

Project also proposes the embedded solution in order to comply with simulation results. To demonstrate the embedded solution, project includes signal generating unit (STM32F407) and a DUT whose output is measured on Arduino. With measurements of input voltage to DUT and output current from DUT the impedance is calculated by generating FFT of both the signals signal. Generated FFT is then fed back to the simulation part of algorithms and algorithm results are observed over real world data. The general workflow of algorithm is depicted in figure 1.

2.2 Software

2.2.1 Algorithm overview

Algebraic iterative reconstruction (AIR) methods are popular methods for computing regularized solutions to discretization of linear inverse problems,

$$Ax \approx b \quad \text{....equation (2)}$$

Where $A \in \mathbb{R}^{m \times n}$

These methods rely on semi-convergence where the number of iterations acts as regularization parameter. A MATLAB package AIR TOOLS with a collection of such methods.

Algebraic iterative reconstruction methods take the overall form of either sequential and simultaneous versions—plus their block variants. We use the following acronyms:

- Algebraic reconstruction technique (ART): Methods that sequentially involve one row at a time. The original algorithm is due to Kaczmarz.
- Column-action reconstruction technique (CART): Methods that sequentially involve one column at a time. These methods are sometimes referred to as column-relaxation methods.
- Simultaneous iterative reconstruction technique (SIRT): Methods that simultaneously involve all the rows at a time and, therefore, are based on matrix multiplications.

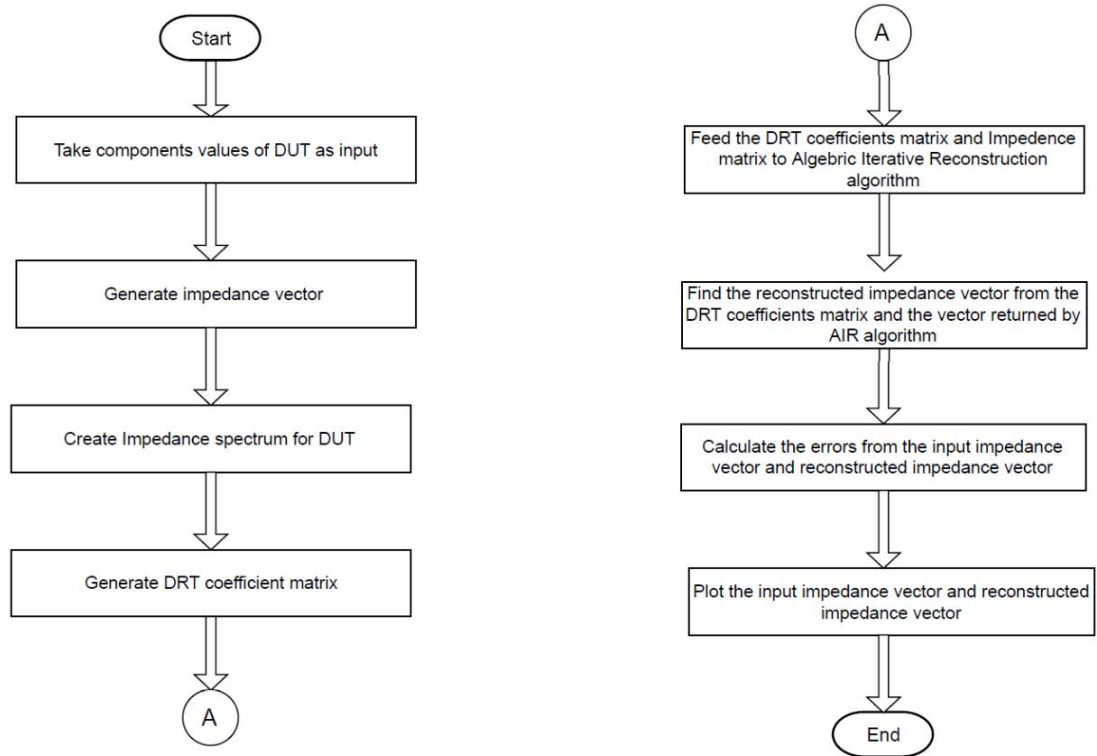


Figure 2: General Flowchart of DRT Algorithm

Table 2: Classification of DRT algorithms

Algorithm Category	Functions	Description
ART methods	<ul style="list-style-type: none"> art kaczmarz randkaczmarz syskaczmarz 	<ul style="list-style-type: none"> General interface for all the ART methods. Kaczmarz's method with cyclic row sweep. Kaczmarz's method with random row selection. Kaczmarz's method with "symmetric" (top→bottom→top) row sweep.
CART methods	<ul style="list-style-type: none"> cart columnaction 	<ul style="list-style-type: none"> General interface for the CART methods. Column-action method with cyclic column sweep.
SIRT methods	<ul style="list-style-type: none"> Sirt cav Cimmino Drop Landweber sart 	<ul style="list-style-type: none"> General interface for all the SIRT methods. Component averaging (CAV) method. Cimmino's method. Diagonally relaxed orthogonal projection. (DROP) method. Landweber's method. Simultaneous algebraic reconstruction technique (SART).

All iterative methods in this package {see table 2} take as input the coefficient matrix A (which is typically sparse) and the right-hand side vector b . It is possible, in all the iterative

methods, to include an orthogonal projection PC on the convex set C. We provide general box constraints described by the set,

$$C = [l_1, u_1] \times [l_2, u_1] \times \dots \times [l_n, u_n] \quad \text{....equation (3)}$$

where the lower (l) and upper bounds (u) are allowed to be $-\infty$ and $+\infty$, respectively.

The three row-action ART methods in this package involve an update of the iteration vector x of the form

$$x \leftarrow P_C \left(x + \omega \frac{b_i - a_i^T x}{\|a_i\|_2^2 + \alpha} a_i \right), \quad 1 \leq i \leq m, \quad \text{....equation (4)}$$

where b_i is the i^{th} component of the right-hand side,
 a_i^T is the i^{th} row of the coefficient matrix, and
 ω is a relaxation parameter which is either constant or decreases with the iterations (not to be confused with ω , the DRT angular frequency).

Moreover, α is a damping parameter that stabilizes the iterations for small row norms; we choose

$$\alpha = \text{damp} \cdot \max_i \|a_i\|_2^2, \quad \text{....equation (5)}$$

where damp is a user-specified parameter (default to zero).

What distinguishes the different methods are the strategy for selecting the i^{th} row at the k^{th} sweep over the rows, we provide one column-action CART method that uses an update of the form,

$$x_j \leftarrow P_C \left(x_j + \omega \frac{c_j^T (b - A x)}{\|c_j\|_2^2 + \alpha} \right), \quad 1 \leq j \leq n, \quad \text{....equation (6)}$$

where x_j is the j^{th} component of x , and
 c_j is the j^{th} column of the coefficient matrix.

We use α similar to before. The columns are selected in a cyclic fashion, in our implementation of CART, it is possible to use the “flagging” strategy from where update steps are skipped if the update is small; this can save a substantial amount of computational work.

For all ART and CART methods, to guarantee asymptotic convergence (for $k \rightarrow \infty$), the relaxation parameter must satisfy

$$0 < \omega < 2 \quad \text{....equation (7)}$$

The default value is $\omega = 1$ for ART and $\omega = 0.25$ for CART.

If the system (1) is inconsistent, then a constant relaxation parameter ω leads to cyclic asymptotic convergence for Kaczmarz's method, and to avoid this, one must use a diminishing step size. Therefore, in the ART methods, the user can supply a function (instead of a constant) with a diminishing parameter, e.g., $\omega = 1/\sqrt{l}$ where l counts the total number of row updates.

All five SIRT methods in this package involve updates of the form,

$$x \leftarrow P_C \left(x + \omega_k D A^T M (b - A x) \right), \quad k = 1, 2, 3, \dots, \quad \text{....equation (8)}$$

where ω_k is the relaxation parameter in iteration k while D and M are diagonal matrices with positive diagonal elements and, hence, symmetric.

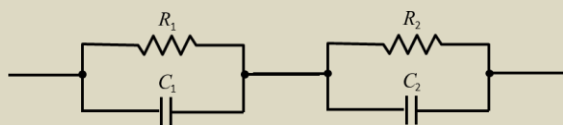
2.2.2 Algorithm Results

We have assumed the following parameters to feed into the available algorithms. This is to determine which algorithm is most suitable for our application.

Assumptions for Simulation:

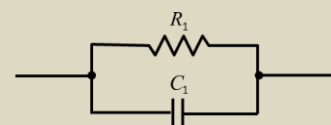
For 2-RC DUTs:

$R_1 = 10 \text{ m}\Omega$; $R_2 = 100 \text{ m}\Omega$; $C_1 = 1 \text{ mF}$; $C_2 = 10 \text{ mF}$
Frequency = 159.1549 Hz, 131.533 Hz
Number of Points (npts): Ranges between 0 and 100.



For RC DUTs:

$R = 10 \text{ m}\Omega$; $C = 100 \text{ mF}$
Frequency = 159.1549 Hz
Number of Points (npts): 40



2.2.2.1 CAV

a. For 2-RC Circuit:

Number of Iterations: 100000

Table 3: CAV result for 2-RC DUT

Number of points (npts)	Time Taken (in seconds)	Mean Absolute Error (Ω)	Root Mean Square Error (Ω)	R squared
10	5.6255	$4.3243 \cdot 10^{-4}$	0.0154	0.9951
30	7.8041	$1.5282 \cdot 10^{-5}$	$5.3788 \cdot 10^{-4}$	≈ 1.0000
70	10.4625	$1.1762 \cdot 10^{-5}$	$4.1659 \cdot 10^{-4}$	≈ 1.0000
100	12.4901	$1.4182 \cdot 10^{-5}$	$5.0324 \cdot 10^{-4}$	≈ 1.0000

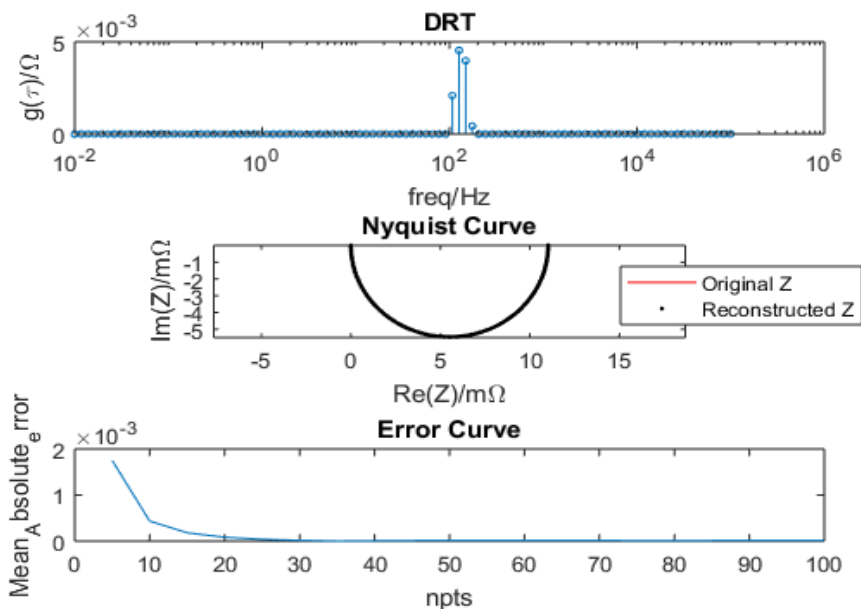


Figure 3(a): CAV result for 2-RC DUT

b. For RC DUT:

Number of Iterations: 100000

Time Taken: 4.9108 seconds

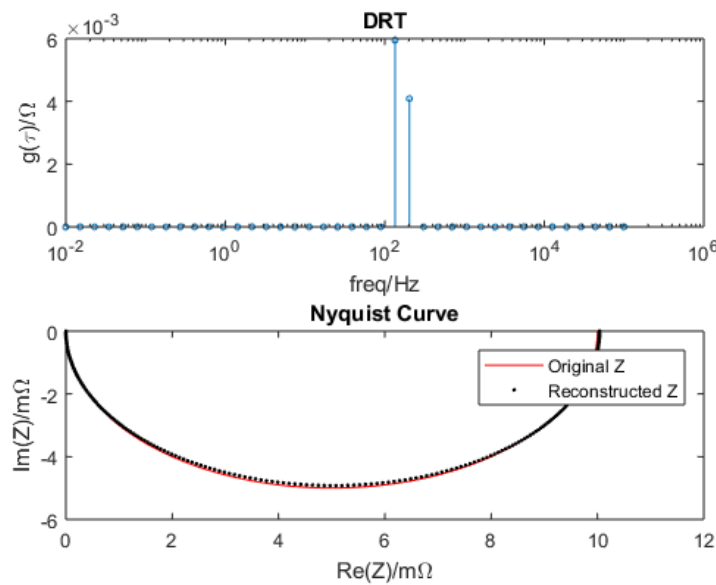


Figure 3(b): CAV result for RC DUT

2.2.2.2 CIMMINO

a. For 2-RC Circuit:

Number of Iterations: 100000

Table 4: CIMMINO result for 2-RC DUT

Number of points (npts)	Time Taken (in seconds)	Mean Absolute Error (Ω)	Root Mean Square Error (Ω)	R squared
10	11.0402	$1.4182 \cdot 10^{-5}$	$5.0324 \cdot 10^{-4}$	≈ 1.0000
30	6.5251	$1.5282 \cdot 10^{-5}$	$5.3788 \cdot 10^{-4}$	≈ 1.0000
70	9.3231	$1.1762 \cdot 10^{-5}$	$4.1659 \cdot 10^{-4}$	≈ 1.0000
100	11.0402	$1.4182 \cdot 10^{-5}$	$5.0324 \cdot 10^{-4}$	≈ 1.0000

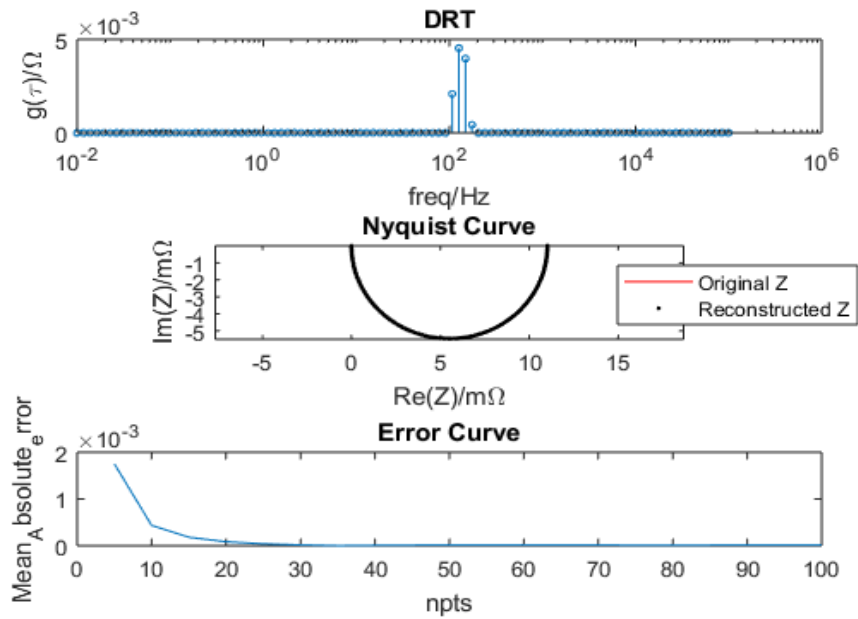


Figure 4(a): CIMMINO result for 2-RC DUT

b. For RC DUT:

Number of Iterations: 100000
Time Taken: 4.6639 seconds

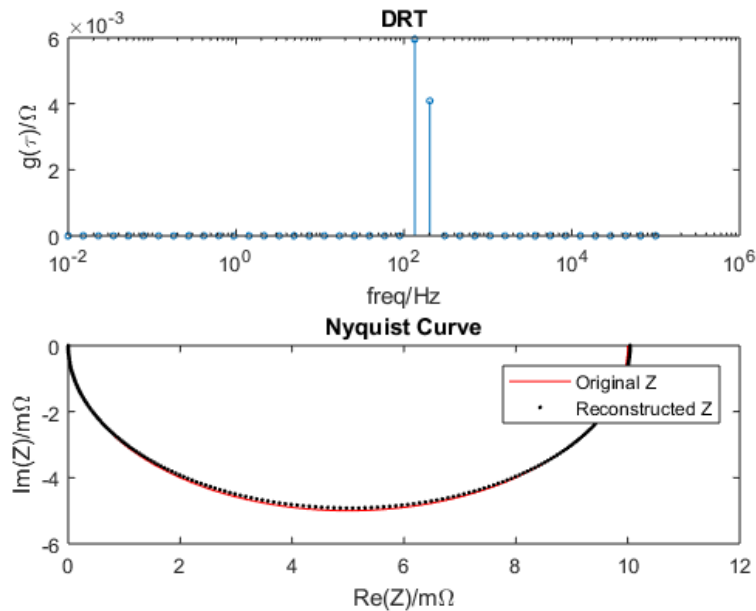


Figure 4(b): CIMMINO result for RC DUT

2.2.2.3 COLUMN ACTION

a. For 2-RC Circuit:

Number of Iterations: 100000

Table 5: COLUMN ACTION result for 2-RC DUT

Number of points (npts)	Time Taken (in seconds)	Mean Absolute Error (Ω)	Root Mean Square Error (Ω)	R squared
10	2.9470	$3.1273 \cdot 10^{-4}$	0.0138	0.9951
30	6.0869	$6.0313 \cdot 10^{-5}$	0.0027	0.9998
70	22.5666	$3.4363 \cdot 10^{-5}$	0.0015	0.9999
100	31.6129	$2.9028 \cdot 10^{-5}$	0.0013	≈ 1.0000

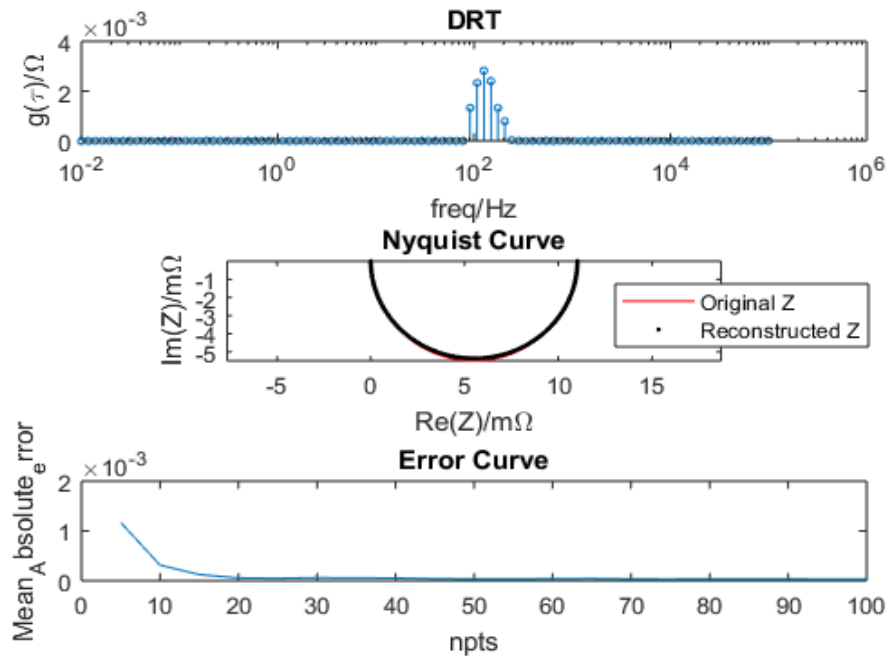


Figure 5(a): COLUMN ACTION result for 2-RC DUT

b. For RC DUT:

Number of Iterations: 100000

Time Taken: 7.0421 seconds

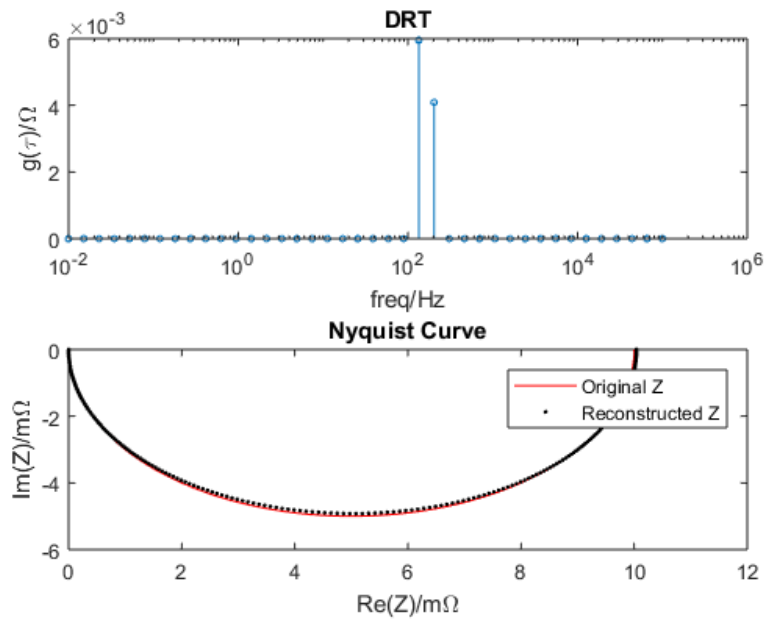


Figure 5(b): COLUMN ACTION result for RC DUT

2.2.2.4 DROP

a. For 2-RC Circuit:

Number of Iterations: 100000

Table 6: DROP result for 2-RC DUT

Number of points (npts)	Time Taken (in seconds)	Mean Absolute Error (Ω)	Root Mean Square Error (Ω)	R squared
10	3.3744	$4.3243 \cdot 10^{-4}$	0.0154	0.9951
30	8.1036	$1.5282 \cdot 10^{-5}$	$5.3788 \cdot 10^{-4}$	≈ 1.0000
70	10.5729	$1.1762 \cdot 10^{-5}$	$4.1659 \cdot 10^{-4}$	≈ 1.0000
100	12.4545	$1.4182 \cdot 10^{-5}$	$5.0324 \cdot 10^{-4}$	≈ 1.0000

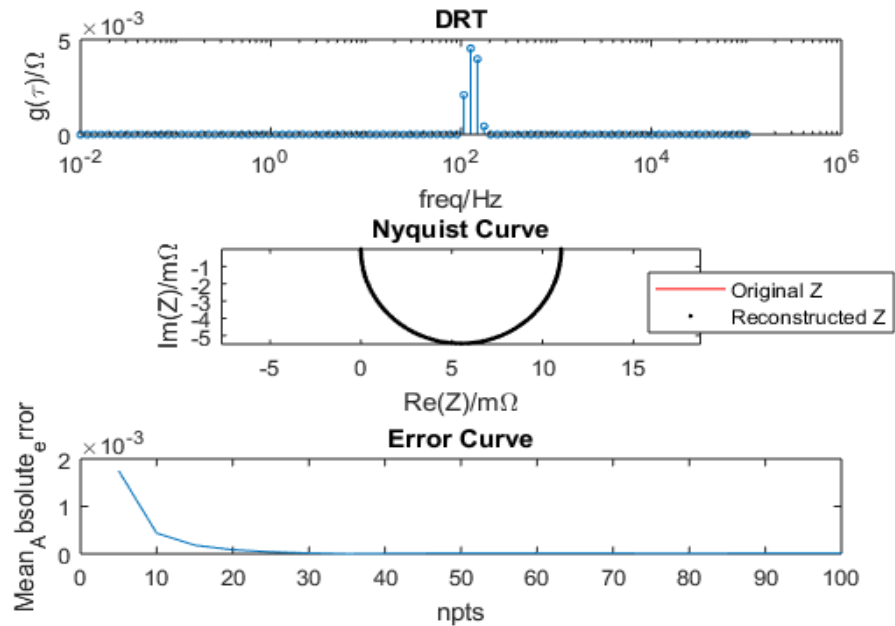


Figure 6(a): DRT result for 2-RC DUT

b. For RC DUT:

Number of Iterations: 100000

Time Taken: 6.3394 seconds

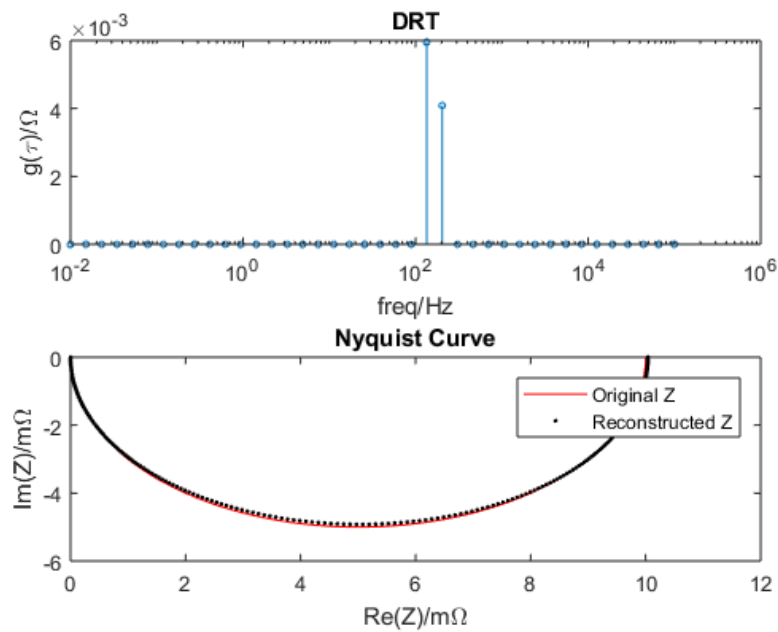


Figure 6(b): DRT result for RC DUT

2.2.2.5 KACZMARZ

a. For 2-RC Circuit:

Number of Iterations: 10000

Table 7: KACZMARZ result for 2-RC DUT

Number of points (npts)	Time Taken (in seconds)	Mean Absolute Error (Ω)	Root Mean Square Error (Ω)	R squared
10	65.6813	0.0020	0.0764	0.9749
30	68.9540	$9.9085 \cdot 10^{-5}$	0.0038	0.9999
70	50.7822	$6.6395 \cdot 10^{-5}$	0.0026	0.9999
100	77.1887	$5.1600 \cdot 10^{-5}$	0.0020	≈ 1.0000

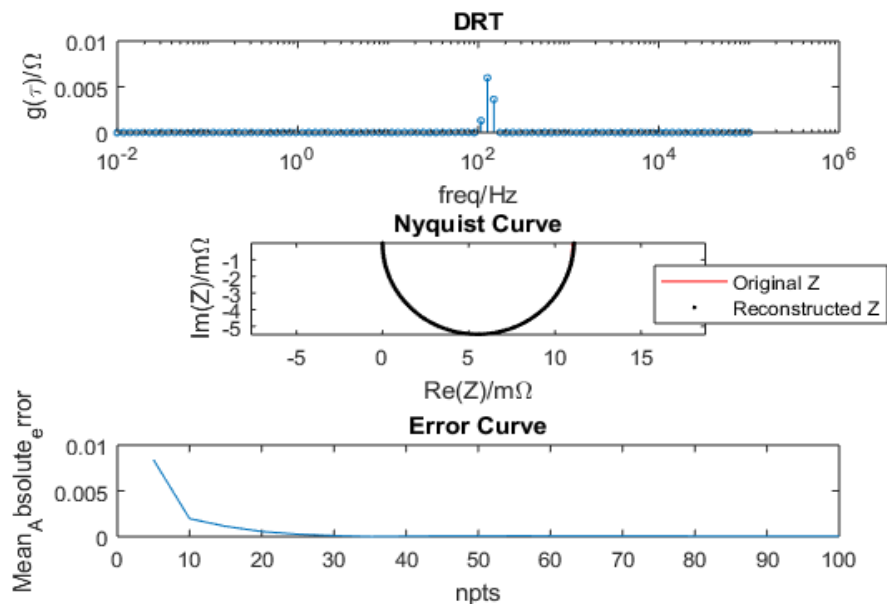


Figure 7(a): KACZMARZ result for 2-RC DUT

b. For RC DUT:

Number of Iterations: 80

Time Taken: 0.4580 seconds

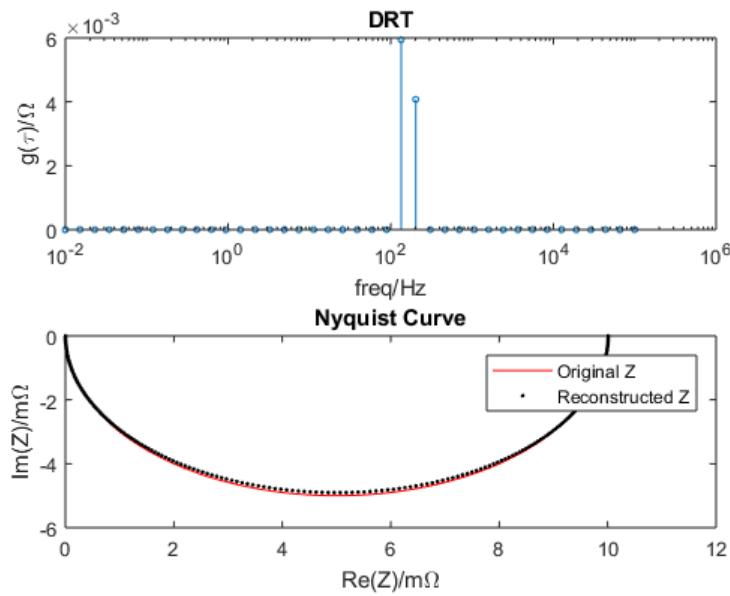


Figure 7(b): KACZMARZ result for RC DUT

2.2.2.6 Non-negative Linear Least-Square (LSQNONNEG)

a. For 2-RC Circuit:

Table 8: LSQNONNEG result for 2-RC DUT

Number of points (npts)	Number of Iterations	Time Taken (in seconds)	Mean Absolute Error(Ω)	Root Mean Square Error (Ω)	R squared
10	2	1.538393	$3.1273 \cdot 10^{-4}$	0.0138	0.9951
30	4	1.538393	$1.0053 \cdot 10^{-5}$	$4.5083 \cdot 10^{-4}$	≈ 1.0000
70	5	3.889735	$7.1200 \cdot 10^{-6}$	$3.2453 \cdot 10^{-4}$	≈ 1.0000
100	6	5.562340	$2.7527 \cdot 10^{-6}$	$1.2502 \cdot 10^{-4}$	≈ 1.0000

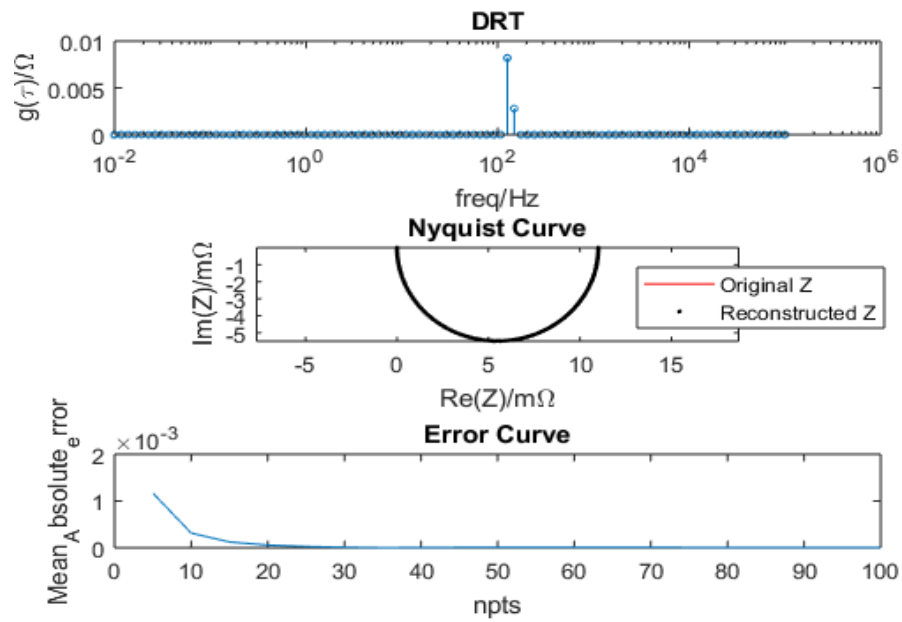


Figure 8(a): LSQNONNEG result for 2-RC DUT

b. For RC DUT:

Number of Iterations: 4

Time Taken: 0.129237 seconds

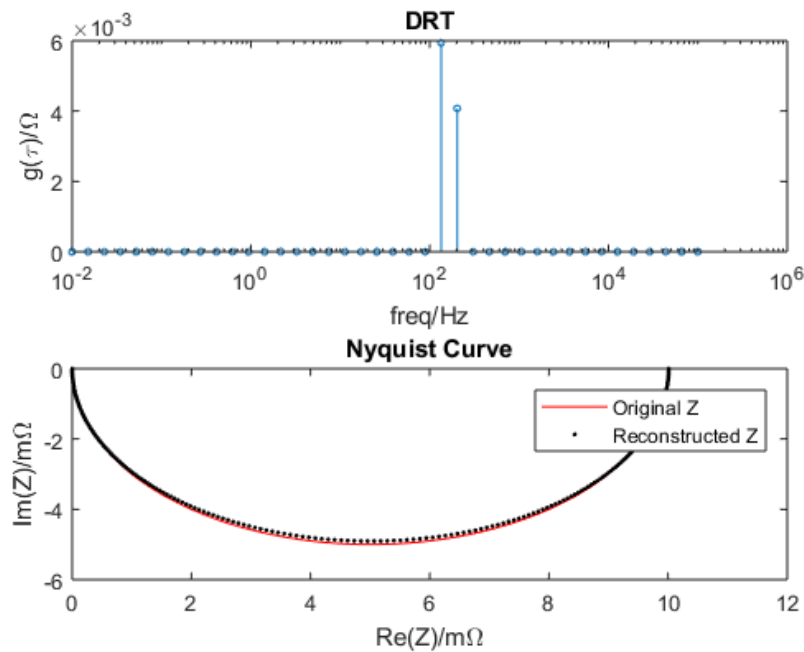


Figure 8(b): LSQNONNEG result for RC DUT

2.2.2.7 RANDOM KACZMARZ

a. For 2-RC Circuit:

Number of Iterations: 10000

Table 9: RANDOM KACZMARZ result for 2-RC DUT

Number of points (npts)	Time Taken (in seconds)	Mean Absolute Error(Ω)	Root Mean Square Error (Ω)	R squared
10	140.1517	$2.8953 \cdot 10^{-4}$	0.0144	0.9946
30	148.9213	$9.6697 \cdot 10^{-6}$	$4.8768 \cdot 10^{-4}$	≈ 1.0000
70	201.9920	$6.5915 \cdot 10^{-6}$	$3.5031 \cdot 10^{-4}$	≈ 1.0000
100	214.2109	$8.9128 \cdot 10^{-6}$	$2.9216 \cdot 10^{-4}$	≈ 1.0000

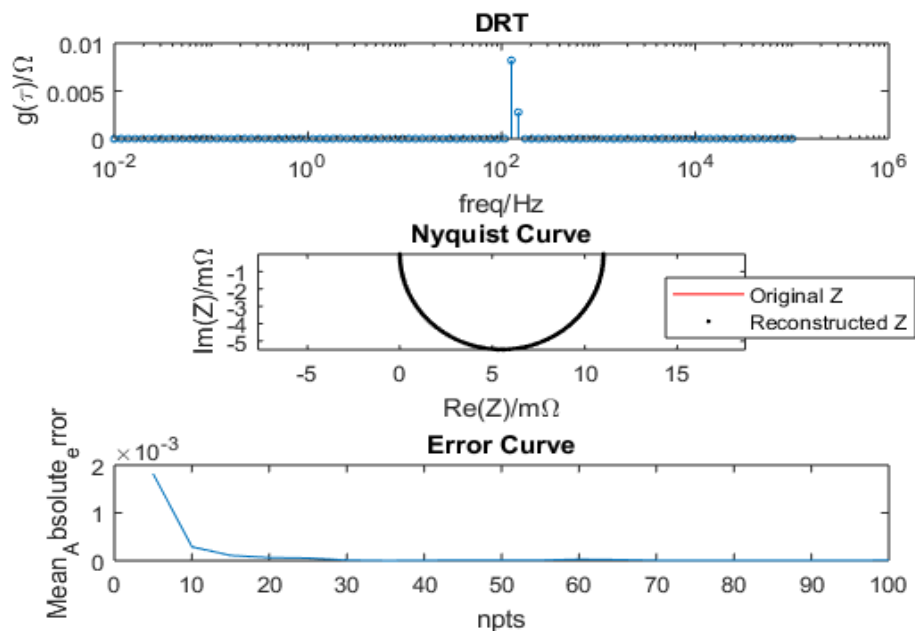


Figure 9(a): RANDOM KACZMARZ result for 2-RC DUT

b. For RC DUT:

Number of Iterations: 100000

Time Taken: $1.1343 \cdot 10^3$ seconds

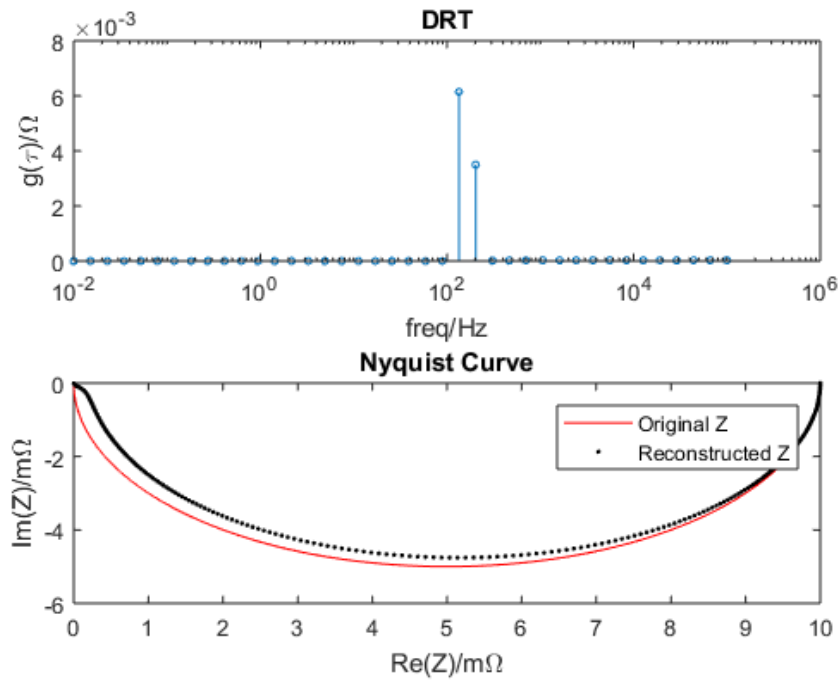


Figure 9(b): RANDOM KACZMARZ result for RC DUT

2.2.2.8 SART

a. For 2-RC Circuit:

Number of Iterations: 100000

Table 10: SART result for 2-RC DUT

Number of points (npts)	Time Taken (in seconds)	Mean Absolute Error(Ω)	Root Mean Square Error (Ω)	R squared
10	3.4242	$3.7136 \cdot 10^{-4}$	0.0142	0.9950
30	7.9466	$1.2687 \cdot 10^{-5}$	$4.7644 \cdot 10^{-4}$	≈ 1.0000
70	10.5200	$1.1971 \cdot 10^{-5}$	$4.5135 \cdot 10^{-4}$	≈ 1.0000
100	12.3817	$1.4251 \cdot 10^{-5}$	$5.3939 \cdot 10^{-4}$	≈ 1.0000

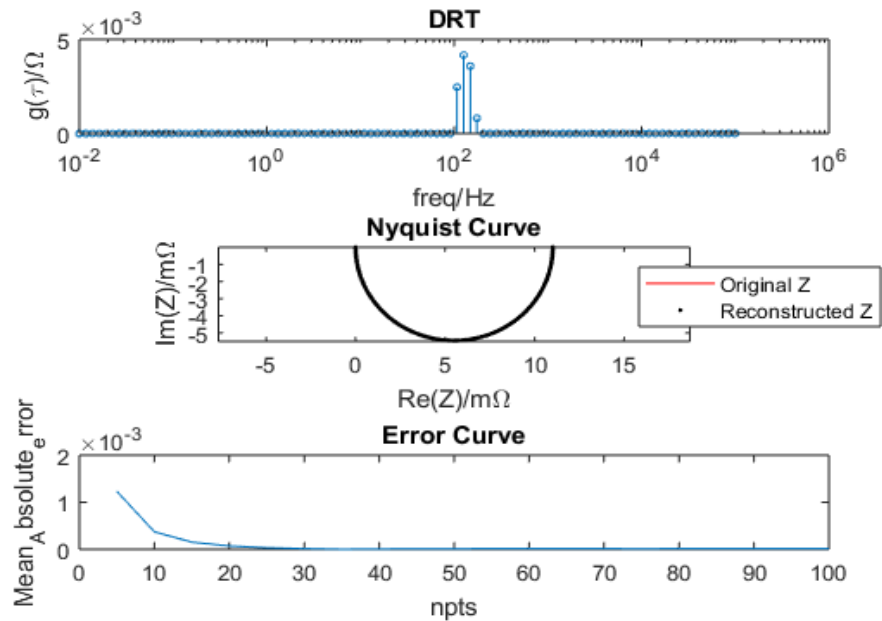


Figure 10(a): SART result for 2-RC DUT

b. For RC DUT:

Number of Iterations: 100000

Time Taken: 4.7291 seconds

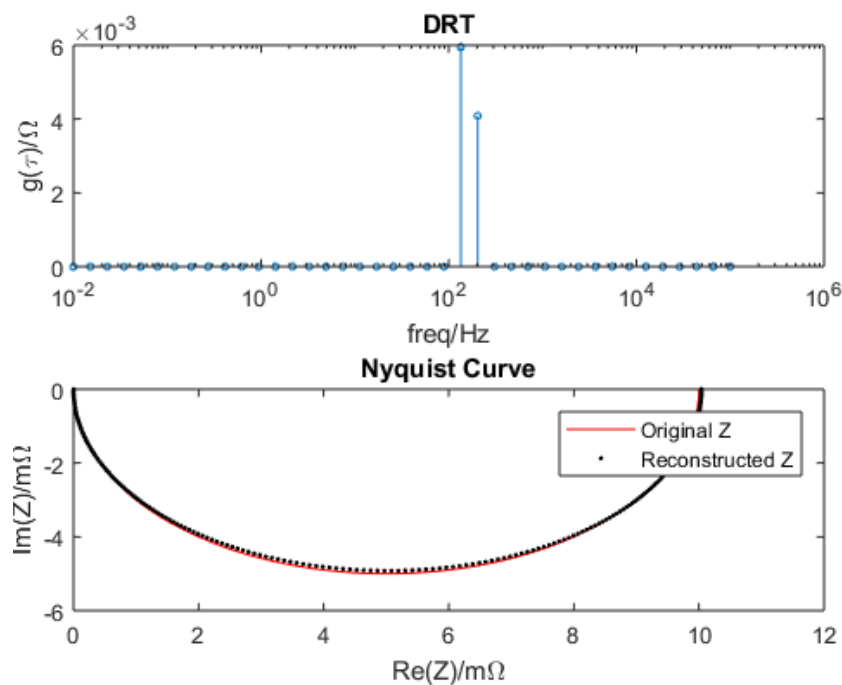


Figure 10(b): SART result for RC DUT

2.2.2.9 SYSTEMATIC KACZMARZ

a. For 2-RC Circuit:

Number of Iterations: 10000

Table 11: SYSTEMATIC KACZMARZ result for 2-RC DUT

Number of points (npts)	Time Taken (in seconds)	Mean Absolute Error(Ω)	Root Mean Square Error (Ω)	R squared
10	3.4242	0.0010	0.0440	0.9797
30	65.4650	$4.1347 \cdot 10^{-5}$	0.0018	≈ 1.0000
70	69.8505	$1.0913 \cdot 10^{-4}$	0.0049	0.9999
100	46.9542	$9.4852 \cdot 10^{-5}$	0.0042	0.9999

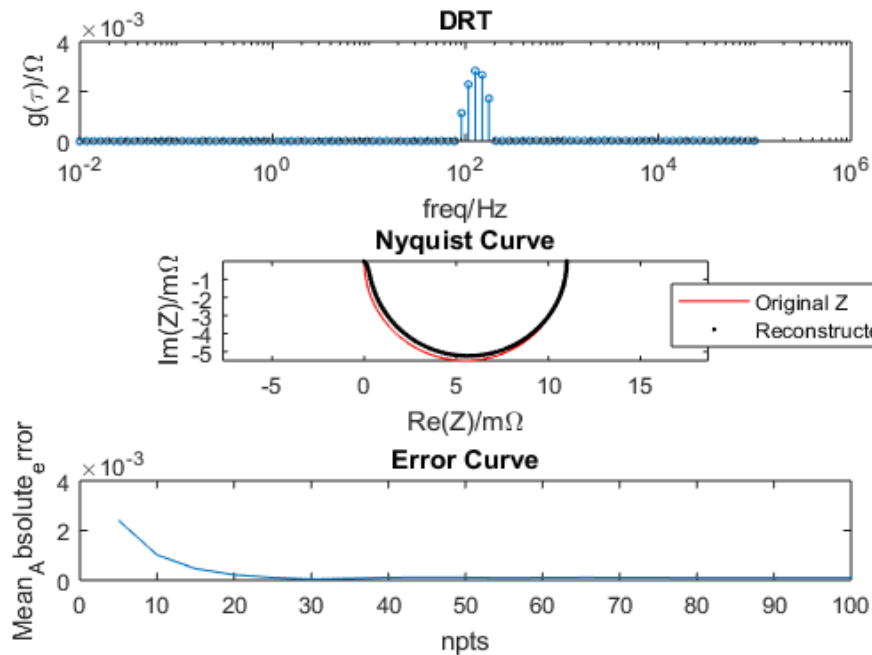


Figure 11(a): SYSTEMATIC KACZMARZ result for 2-RC DUT

b. For RC DUT:

Number of Iterations: 100000

Time Taken: 419.8459 seconds

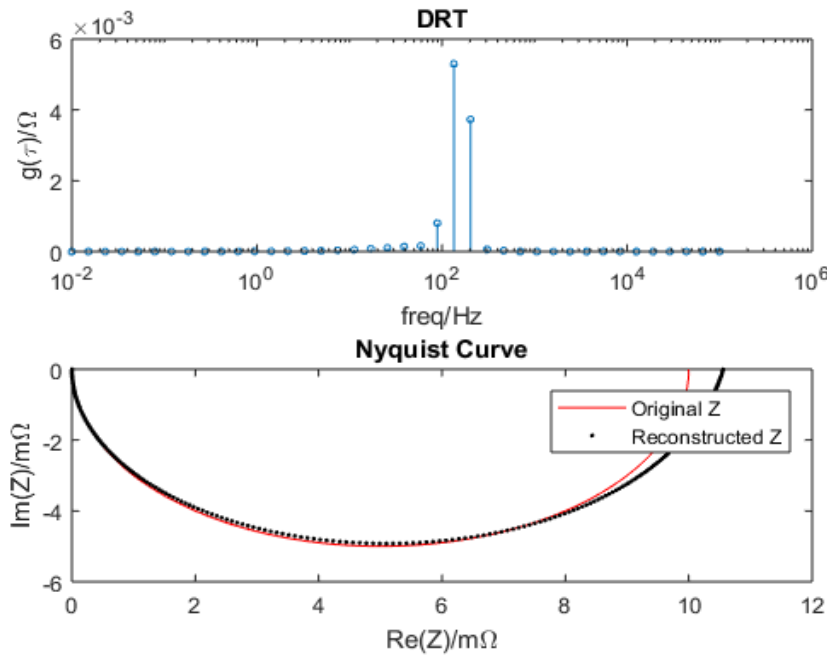


Figure 11(b): SYSTEMATIC KACZMARZ result for RC DUT

2.2.3 Algorithm Choosing

In the previous section, we came across various algorithms to determine DRT. We consider two parameters for deciding the ideal algorithm for our application: Execution Time and Mean Absolute Error. But these values vary with respect to the number of points in the frequency vector (npts). From observation, higher the npts (i.e., 100), better the reconstruction (see figure 12). Now plotting the execution time and error for npts 100 for all the algorithms, as shown in figure below.

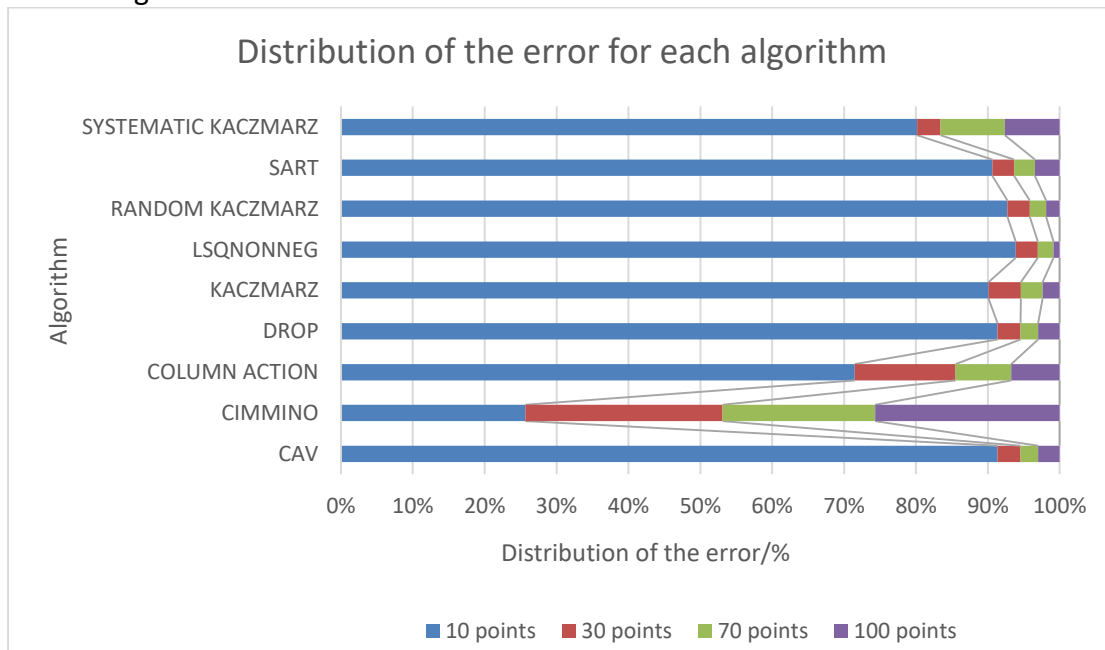


Figure 12: Distribution of the error per number of points and algorithm

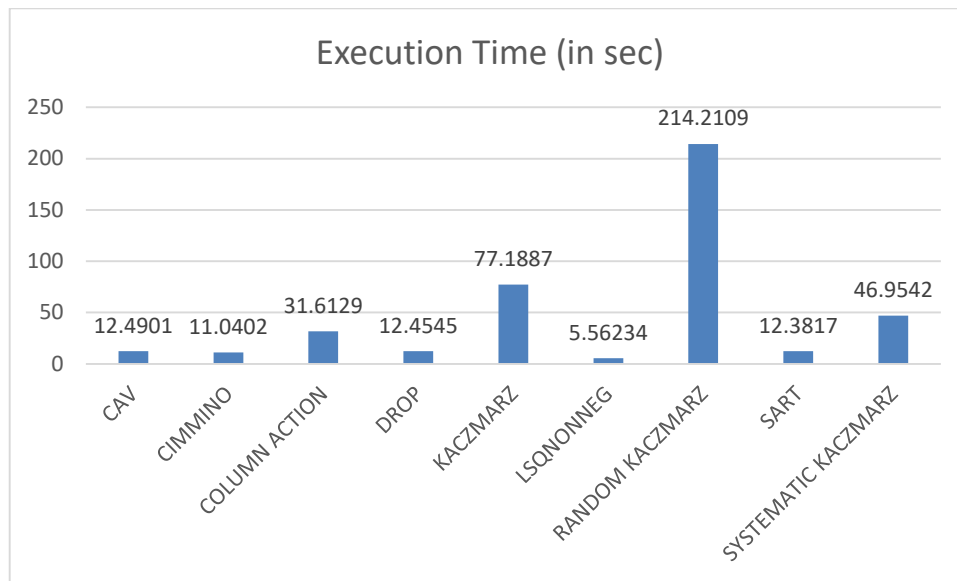


Figure 13: Plot of execution time for different DRT algorithms

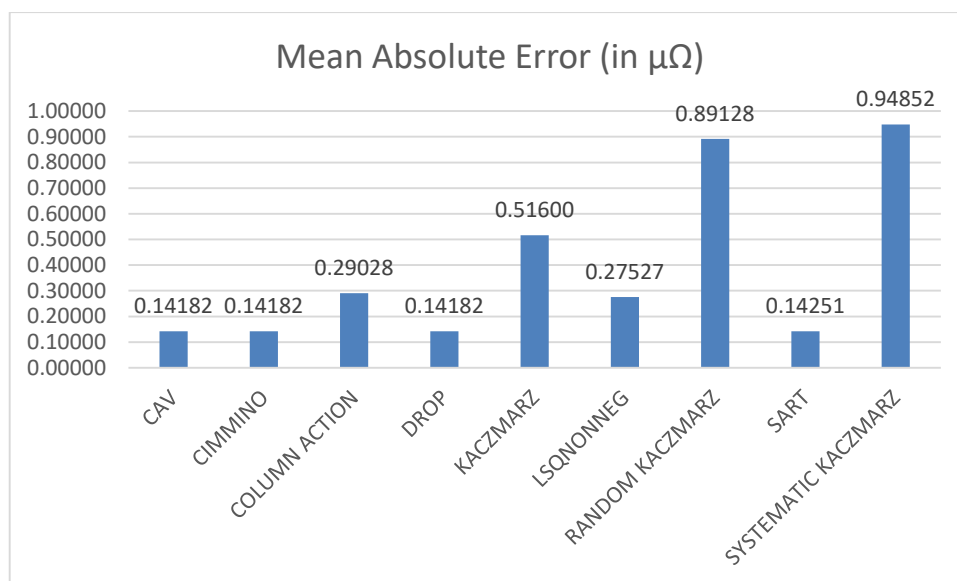


Figure 14: Plot of Mean Absolute Error for different DRT algorithms

From figure 13 and 14, Non-Negative Linear Least- Square (LSQNONNEG) method is most suitable.

In this project, we consider DUTs possibly having resistance and capacitance (RC), KACZMARZ is ideally used for RQ circuits. Though KACZMARZ is conceptually correct, mean absolute error and the execution time of this method is not good. The LSQNONNEG method gives a better practical model for the same circuit.

2.3 Hardware

2.3.1 STM32 F4

The STM32 F4-series is the first group of STM32 microcontrollers based on the ARM Cortex-M4F core. The F4 is pin-to-pin compatible with the STM32 F2-series and adds higher clock speed, 64 KB CCM static RAM, full duplex I²S, improved real-time clock, and faster ADCs.

Technical Specification

- Core: ARM Cortex-M4F core at a maximum clock rate of 84 / 100 / 168 / 180 MHz
- Memory: Static RAM consists of up to 192 KB general purpose, 64 KB core coupled memory (CCM), 4 KB battery-backed, 80 bytes battery-backed with tamper-detection erase.
- Flash: It consists of 512 / 1024 / 2048 KB general purpose, 30 KB system boot, 512 bytes one-time programmable (OTP), 16 option bytes.
- Each chip has a factory-programmed 96-bit unique device identifier number.
- Peripherals: Common peripherals included in all IC packages are USB 2.0 OTG HS and FS, two CAN 2.0B, one SPI + two SPI or full-duplex I²S, three I²C, four USART, two UART, SDIO for SD/MMC cards, twelve 16-bit timers, two 32-bit timers, two watchdog timers, temperature sensor, 16 or 24 channels into three ADCs, two DACs, 51 to 140 GPIOs, sixteen DMA, improved real-time clock (RTC), cyclic redundancy check (CRC) engine, random number generator (RNG) engine. Larger IC packages add 8/16-bit external memory bus capabilities.
- The STM32F4x7 models add Ethernet MAC and camera interface.
- The STM32F41x/43x models add a cryptographic processor for DES / TDES / AES, and a hash processor for SHA-1 and MD5.
- The STM32F4x9 models add a LCD-TFT controller.
- Oscillators consists of internal (16 MHz, 32 kHz), optional external (4 to 26 MHz, 32.768 to 1000 kHz).
- IC packages: WLCSP64, LQFP64, LQFP100, LQFP144, LQFP176, UFBGA176. STM32F429/439 also offers LQFP208 and UFBGA216.
- Operating voltage range is 1.8 to 3.6 volt.

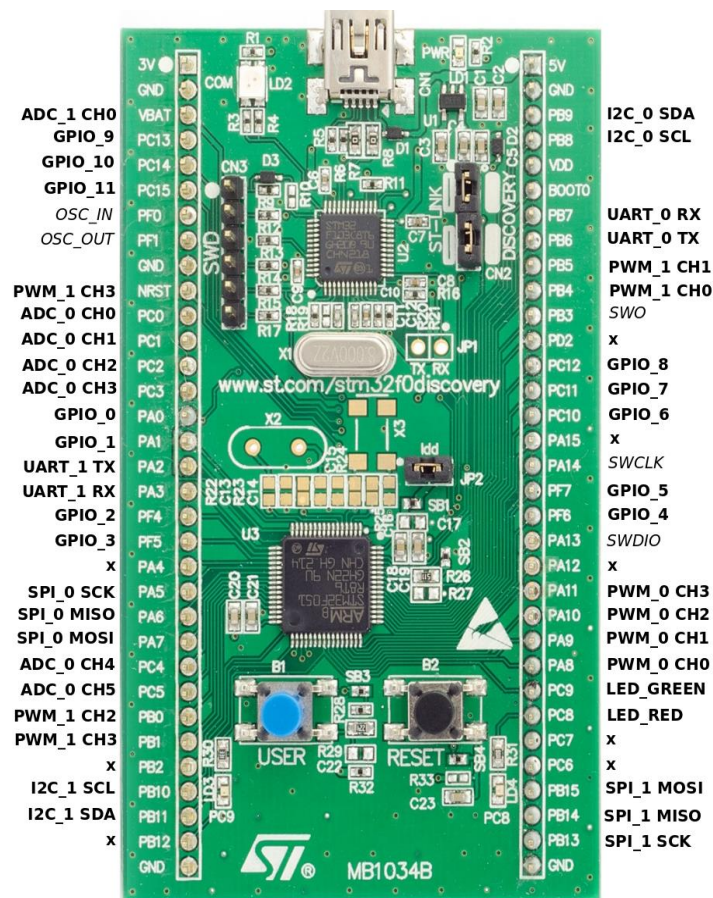


Figure 15: Schematic of STM32F407VGT6 microcontroller

2.3.2 Arduino Uno

The Arduino UNO is a widely used open-source microcontroller board based on the ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board features 14 Digital pins and 6 Analog pins. It is programmable with the Arduino IDE (Integrated Development Environment) via a type B USB cable. It can be powered by a USB cable or by an external 9 volt battery, though it accepts voltages between 7 and 20 volts. It is also similar to the Arduino Nano and Leonardo.

The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform. The ATmega328 on the Arduino Uno comes preprogrammed with a boot loader that allows to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol. The Uno also differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter. The Arduino UNO is generally considered the most user-friendly and popular board or the Arduino board series.

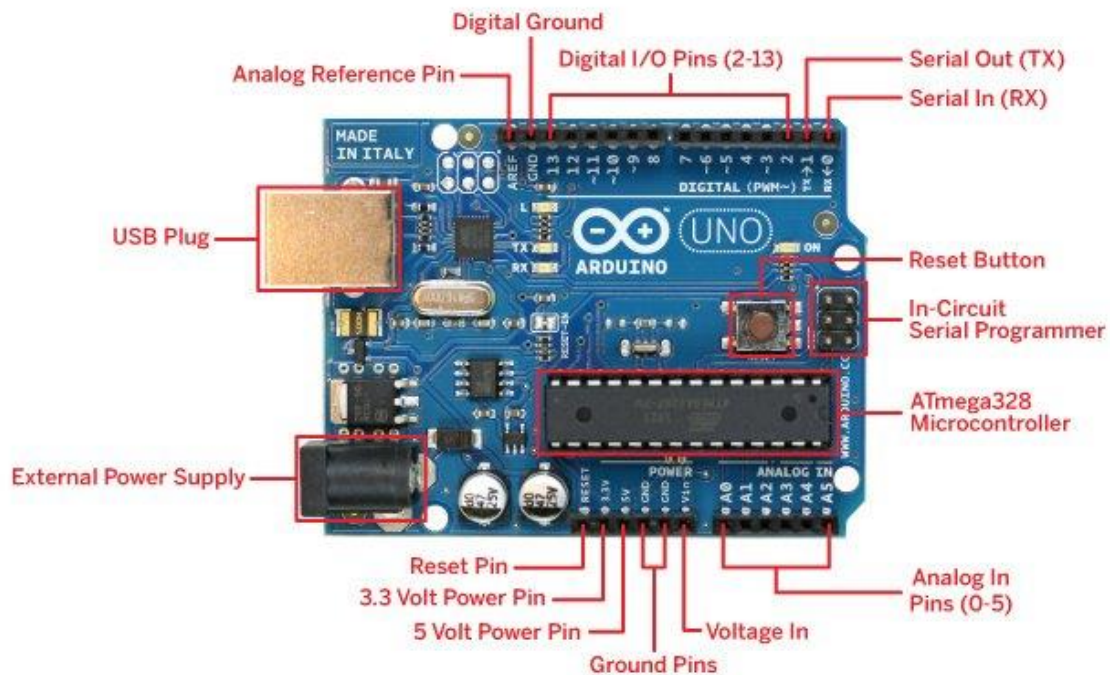


Figure 16: Schematic of Arduino Uno

Technical Specification

- Microcontroller: ATmega328P
- Operating Voltage: 5v
- Input Voltage: 7-20v
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 20 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB of which 0.5 KB used by boot loader
- SRAM: 2 KB
- EEPROM: 1 KB
- Clock Speed: 16 MHz
- Length: 68.6 mm
- Width: 53.4 mm
- Weight: 25 g

2.3.3 Liquid Crystal Display (LCD)

A liquid-crystal display (LCD) is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals. Liquid crystals do not emit light directly, instead using a backlight or reflector to produce images in color or monochrome. LCDs are available to display arbitrary images (as in a general-purpose computer display) or

fixed images with low information content, which can be displayed or hidden, such as preset words, digits, and seven-segment displays, as in a digital clock. They use the same basic technology, except that arbitrary images are made up of a large number of small pixels, while other displays have larger elements.

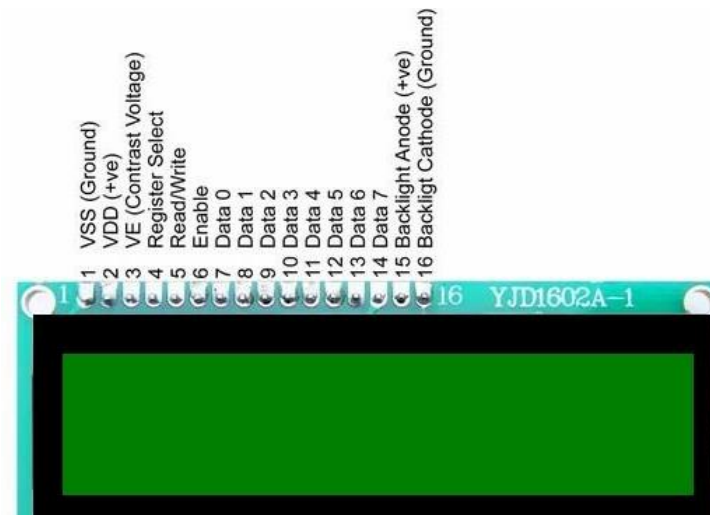


Figure 17: Schematic of a generic LCD

2.3.4 Device under test (DUT)

We have considered a simple RC circuit. But in real world applications, like battery characterization, the battery will be the actual device under test. The impedance is first determined and then electric circuit can be modelled based on the DRT generated. Thus DUT is a black box device, whose internal circuitry is not known priorly.

CHAPTER 3

PROTOTYPE

To generate saw-tooth wave, STM32 Developer Board was used. The generated wave is given to DUT. Then we take the value of voltage across the load resistance and measure current using Arduino.

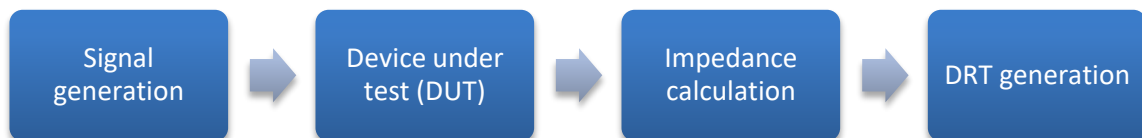


Figure 18: Basic Blocks of a DRT analyzer

Ammeter is used to measure current flow through any load or device. Arduino is used to measure current based on Ohm's law. It states that "the potential difference between two poles or terminals of a conductor is directly proportional to the amount of current pass through the same conductor" for constant of proportionality we use resistance, so here it comes the equation of ohm's law.

$$V = IR \quad \text{....equation (8)}$$

Where V = voltage across the conductor in Volt (v).

I = current pass through the conductor in Ampere (A).

R = resistance constant of proportionality in Ohm (Ω).

In order to find the current pass through the device we just rearrange the equation as below, or we can calculate with ohm's law calculator.

$$I = \frac{V}{R} \quad \text{....equation (9)}$$

This Ammeter circuit, in figure 16, consists of a resistor and LED as load. Resistor is connected in series to the LED that current flows through the load and voltage drops is determined from the resistor. The terminal V1, V2 are going to connect with the analog input of the Arduino.

In the ADC of Arduino that coverts the voltage into 10 bit resolution numbers from 0-1023. So we need to program to covert it in voltage value. Before that we need to know the minimal voltage that the ADC of Arduino can detect, that value is 4.88mV. We multiply the value from ADC with the 4.88mV and we get the actual voltage into the ADC. Learn more about the ADC of Arduino here.

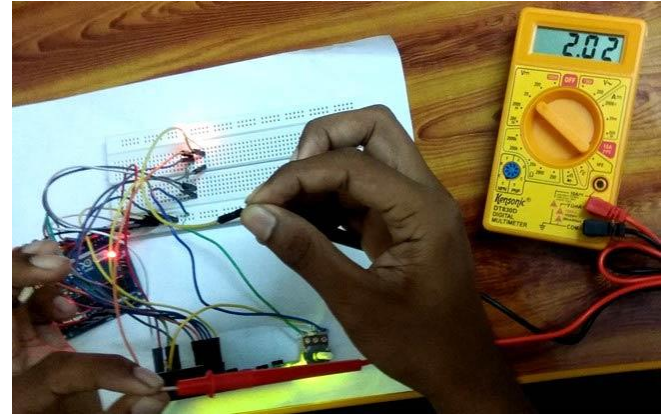
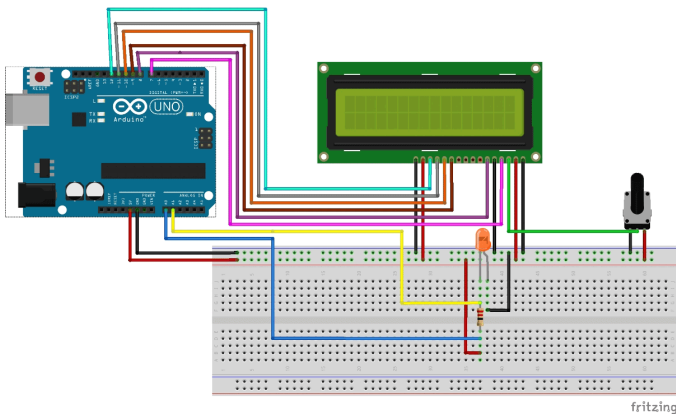


Figure 19: Schematic diagram of the Arduino Ammeter Circuit

This current is fed-back to STM32 and then we perform FFT operation on input voltage and the output current to calculate Impedance (Z). This value of Impedance (Z) is used to calculate the value of resistor and capacitor in the DUT circuit. In this project, we have deviated from this by performing the FFT and DRT calculations using MATLAB on computer.

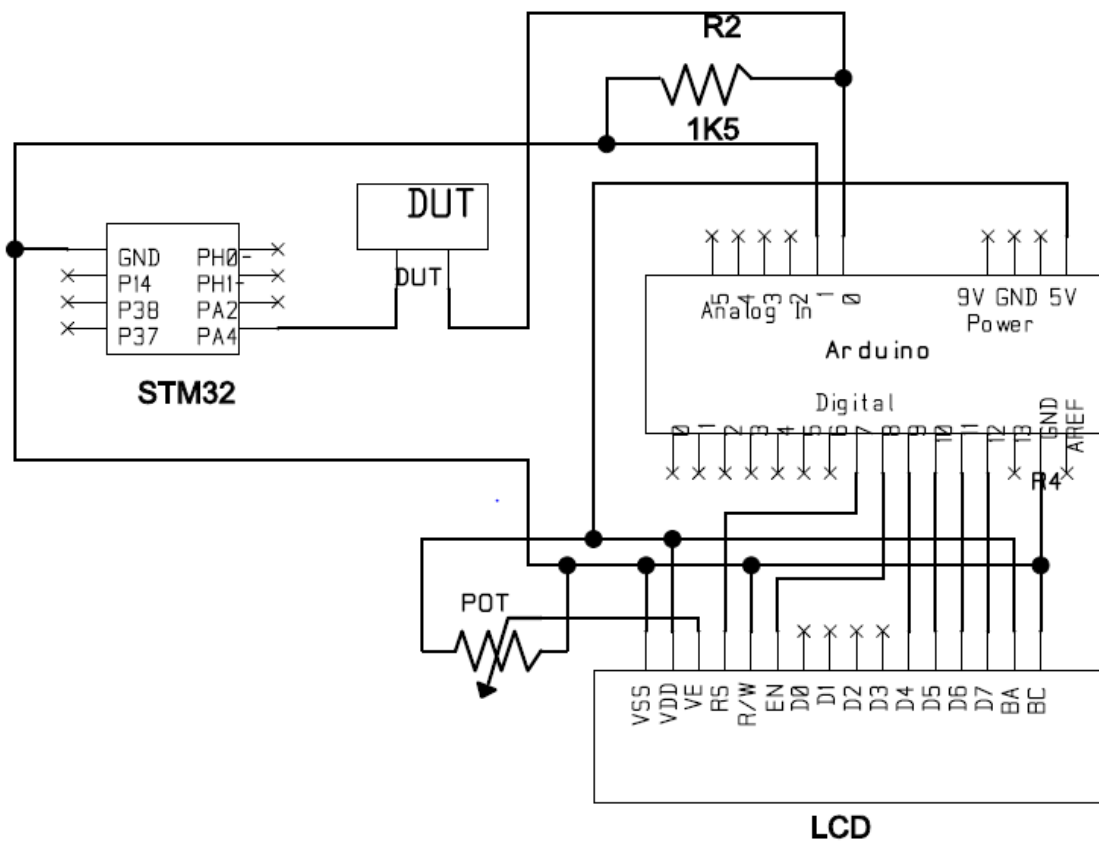


Figure 20: Schematic of DRT real-time analyzer Prototype

Thus, the output current from the Arduino and the output voltage from the STM32 was captured using a DSO and then fed to the MATLAB algorithm. The simulation and details are mentioned in section 3.1.

3.1 Simulation

In order to simulate the embedded DRT analyzer, STM 32F407 (Discovery board) is used as signal generator. Choice of excitation signal for DUT was crucial in establishing DRT simulation. Instead of using multi-sinewave we have explored DRT with sawtooth wave. Sawtooth being rich signal and can also be constructed summation of multiple sine wave we decided to experiment with Sawtooth as excitation signal instead of multi sinewave.

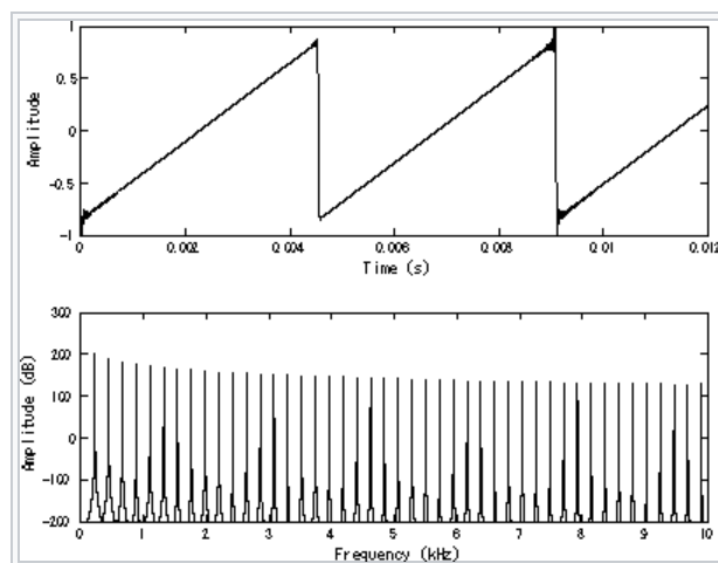


Figure 21: FFT of Sawtooth signal showing its amplitude spikes

Generated signal (with known voltage) is fed to DUT in order to measure the voltage drop across DUT. Output of DUT is fed to Arduino. Arduino is employed to measure the current from the voltage drop of DUT across the known value resistor.

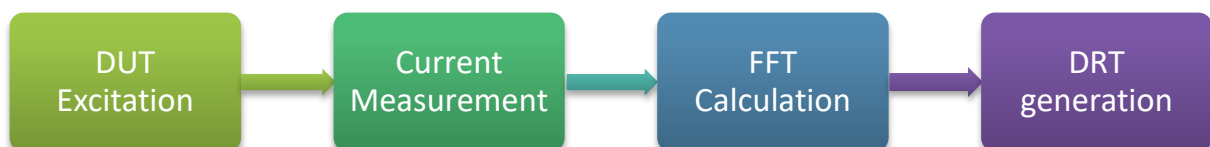


Figure 22: Simulation flow

Now as we are in possession of voltage and current data we can calculate the impedance. In order to calculate the impedance, we should calculate the FFT of the voltage and current signals as both of them were out of phase with each other.

After calculating the FFT of both of the signals we are calculating the impedance of the DUT from the relationship of $Z=U/I$.

At this stage we are transporting the real-world values of impedance to the DRT algorithm for simulation. DRT algorithm, as chosen in section 2.2.3, runs on the provided impedance and resolves the DRT problem.

CHAPTER 4

RESULTS

As stated in section 2.2.3, Non-Negative Linear Least Square algorithm performs better as compared to several algorithms when benchmarked with execution time and mean absolute error. Result of least non-negative linear least square when run on 2RC circuit:

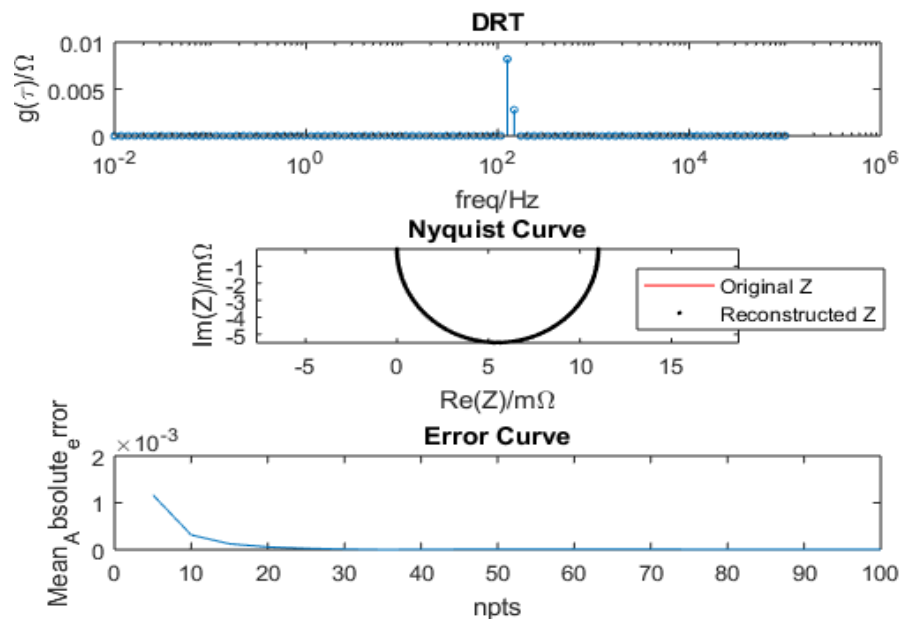


Figure 23: LSQNONNEG result for 2-RC DUT

Experiment Results:

Below shown sawtooth wave was recovered from the data taken via DSO.

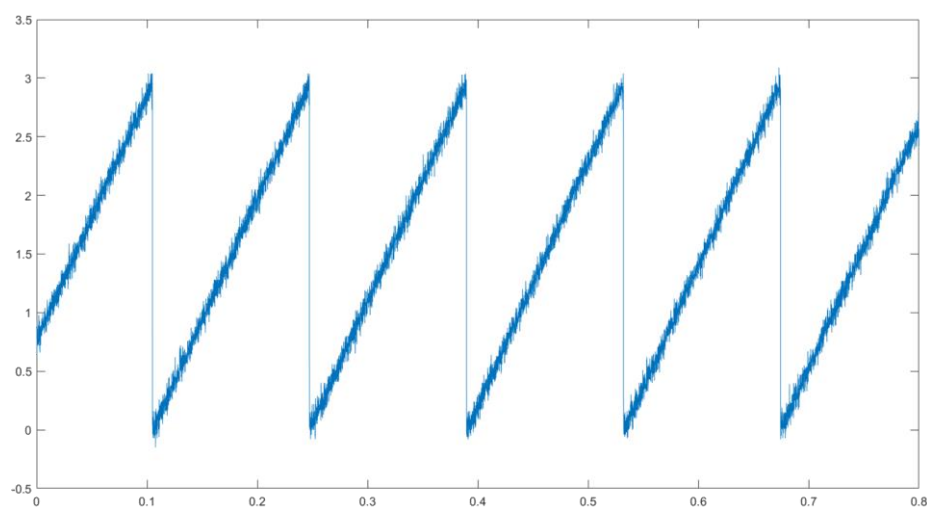


Figure 24: Sawtooth recovered from DSO data

FFT of input voltage (Sawtooth signal): - As STM is capable of generating 3.3v signal but signal recovered from DSO is noisy. Same can be seen from the FFT of the signal.

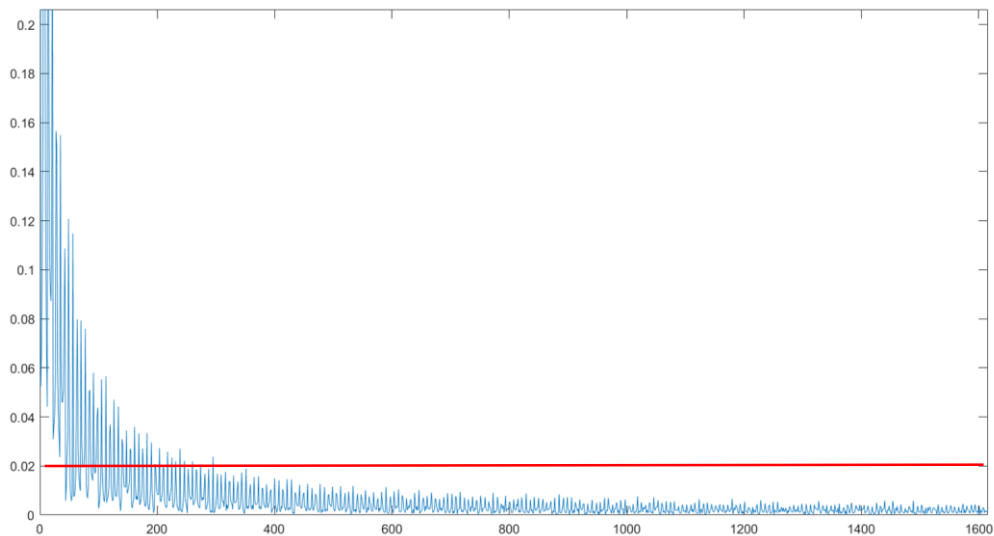


Figure 25: FFT output of the sawtooth signal

DRT and Nyquist plot of DUT with Non-negative linear least square algorithm. Due to presence of noisy input the Nyquist curve heavily differs from the ideal Nyquist curve of simulation.

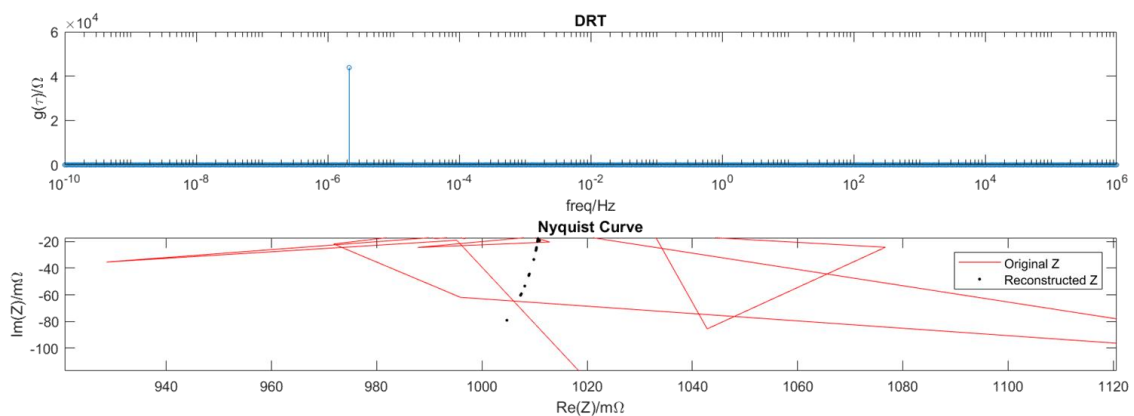


Figure 26: DRT and Nyquist Plot of the DUT

CHAPTER 5

CONCLUSION

The usage of batteries in all the fields is exponentially increasing. In order to enhance the performance of system which is using the battery high level of stress is put on batteries. This scenario gives rise to need of characterization of battery, an important step to get the state-of-charge and the health of the battery.

This project proposes the use of DRT technique instead of using model-based techniques. DRT improves on model-based techniques by eliminating the dependency of model and also saves considerable amount of time.

Among the ART, CART and SART algorithms categories, we propose the “non-negative linear square” for DUTs of RC composition. Although Kaczmarz algorithm family is being widely used in research of DRT. Non-negative least square algorithm betters the Kaczmarz algorithms when benchmarked with execution time and mean absolute error. We also assume that the higher the number of points of frequency vector, higher is the accuracy and reconstruction.

This project also explores the possibility of sawtooth as excitation signal. Sawtooth being rich signal instead of using multisine wave we have used sawtooth for prototyping. The simulation results showcased above show presence of lot of noise which casts doubt on use of sawtooth wave as an excitation signal.

REFERENCES

- [1] Jan Philipp Schmidt, Philipp Berg, Michael Schönleber, André Weber, Ellen Ivers-Tiffée, The Distribution of relaxation times as basis for generalized time-domain models for Li-ion batteries, Journal of Power Sources, Volume 221, 2013, Pages 70-77, ISSN 0378-7753.
- [2] P. Büschel, U. Tröltzsch and O. Kanoun, "Calculation of the distribution of relaxation times For characterization of the dynamic battery behavior," International Multi-Conference on Systems, Signals & Devices, Chemnitz, 2012, pp. 1-3.
- [3] Per Christian Hansen, Maria Saxild-Hansen, AIR Tools — A MATLAB package of algebraic Iterative reconstruction methods, Journal of Computational and Applied Mathematics, Volume 236, Issue 8, 2012, Pages 2167-2178, ISSN 0377-0427.
- [4] Longcong Chen, Gaiqin Liu, "Development of a Multi-Function Signal Generator Based on STM32 with High Performance," Spring Congress on Engineering and Technology, 2012.
- [5] <https://www.sciencedirect.com/science/article/pii/S0167273817307749>
- [6] <http://www.bio-logic.net/wp-content/uploads/AN60.pdf>
- [7] <https://www.st.com>.
- [8] <http://www.bio-logic.net/wp-content/uploads/QCircuits>.
- [9] https://en.wikipedia.org/wiki/Constant_phase_element.
- [10] <https://en.wikipedia.org/wiki/STM32>.
- [11] https://en.wikipedia.org/wiki/Arduino_Uno.