UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Elettronica

Sviluppo di procedure MATLAB per la caratterizzazione tramite EIS di batterie agli ioni di litio e la gestione automatica delle fasi di carica e scarica

Development of MATLAB procedures for the EIS characterization of lithium-ion batteries and the automated management of charging and discharging phases

Candidato:
Samuele Paparelli

Relatore:
Prof. Orcioni Simone

Anno Accademico 2025-2026

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Elettronica

Sviluppo di procedure MATLAB per la caratterizzazione tramite EIS di batterie agli ioni di litio e la gestione automatica delle fasi di carica e scarica

Development of MATLAB procedures for the EIS characterization of lithium-ion batteries and the automated management of charging and discharging phases

Candidato:
Samuele Paparelli

Relatore:
Prof. Orcioni Simone

Anno Accademico 2025-2026

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea in Ingegneria Elettronica
Via Brecce Bianche – 60131 Ancona (AN), Italy

Ai miei genitori che mi hanno sempre aiutato.

Abstract

Nowadays, one of the greatest technological revolutions in the transportation and renewable energy sectors is driven by the development of lithium-ion batteries. The impact of this technology is highly evident in portable electronics, which have become lighter and longer-lasting thanks to the high energy density and excellent cycle life of these batteries, as well as in the transport sector, where they play a crucial role in the energy transition. Furthermore, they are essential for energy storage in renewable power systems. However, since batteries are subject to degradation and aging over time, it is necessary to study these processes through various methodologies. In particular, this study focuses on the analysis and implementation of MATLAB functions for the characterization of the charging, discharging, and Electrochemical Impedance Spectroscopy (EIS) of Sanyo and Panasonic NCR18650BL lithium-ion batteries, using the Keysight N6705C instrument equipped with N6781A and N6783A modules. The developed functions are divided into two main categories: those for charge and discharge management, which implement the two-quadrant (PS2Q) mode with Constant Current (CC) and Constant Voltage (CV) phase control via internal or external data logging, and those for EIS characterization, which leverage the arbitrary waveform (ARB) mode of the N6781A module to generate current sinusoids in the 10 mHz – 1000 Hz range while simultaneously acquiring the battery's voltage and current response signals. The experimental results obtained for the validation of these functions are reported in Chapter 5.

Sommario

Al giorno d'oggi, una delle più grandi rivoluzioni tecnologiche nei settori dei trasporti e delle energie rinnovabili è guidata dallo sviluppo delle batterie agli ioni di litio. L'impatto di questa tecnologia è fortemente evidente nell'elettronica portatile, che è diventata più leggera e duratura grazie all'elevata densità energetica e all'eccellente ciclo di vita di queste batterie, così come nel settore dei trasporti, dove esse svolgono un ruolo cruciale nella transizione energetica. Inoltre, sono essenziali per l'accumulo di energia nei sistemi energetici rinnovabili. Tuttavia, poiché le batterie sono soggette a degradazione e invecchiamento nel tempo, è necessario studiare questi processi attraverso diverse metodologie. In particolare, questo studio si concentra sull'analisi e sull'implementazione di funzioni MATLAB per la caratterizzazione di carica, scarica e spettroscopia di impedenza elettrochimica (EIS) di batterie al litio Sanyo e Panasonic NCR18650BL, utilizzando lo strumento Keysight N6705C dotato di moduli N6781A e N6783A. Le funzioni sviluppate si dividono in due categorie principali: quelle per la gestione di carica e scarica, che implementano la modalità a due quadranti (PS2Q) con controllo delle fasi a corrente costante (CC) e tensione costante (CV) tramite data logger interni o esterni, e quelle per la caratterizzazione EIS, che sfruttano la modalità a forma d'onda arbitraria (ARB) del modulo N6781A per generare sinusoidi di corrente nell'intervallo 10 mHz – 1000 Hz, acquisendo simultaneamente i segnali di risposta in tensione e corrente della batteria. I risultati sperimentali ottenuti per la validazione di queste funzioni sono riportati nel Capitolo 5.

Indice

Capitolo 1.

Richiami teorici

1.1. Batterie agli ioni di Litio

Il litio è un metallo caratterizzato dalla presenza di un solo elettrone nel livello energetico più esterno, che tende a cedere facilmente. Quando l'atomo cede questo elettrone, assume una carica positiva e si trasforma nello ione litio [1].

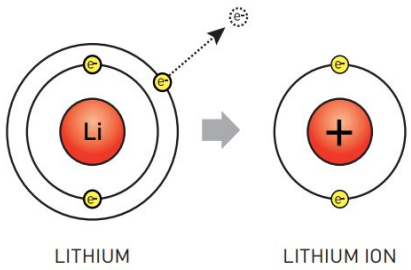

Figura 1.1.: Formazione dello ione di litio. Tratta da: [1]

.

L'impiego delle batterie agli ioni di litio è oggi fondamentale grazie alla loro elevata densità energetica e alla notevole leggerezza. I componenti fondamentali di una batteria sono [2] [3]:

- **Catodo:** è l'elettrodo positivo.
- **Anodo:** è l'elettrodo negativo.
- **Elettrolita:** è la sostanza contenente ioni di litio che mette in contatto chimico e elettrico i due elettrodi.
- **Separatore:** serve a isolare fisicamente l'anodo e il catodo uno dall'altro per consentire lo scambio interno degli ioni di litio ma impedire il passaggio diretto degli elettroni, evitando così il cortocircuito. Solitamente si tratta di una membrana polimerica microporosa.

In Tab.1.1 vengono confrontati i materiali di anodo a catodo delle varie composizioni chimiche delle batterie a litio.

In Fig.1.2 viene mostrata la struttura completa della batteria.

Tabella 1.1.: Confronto dei materiali elettrodici (catodo e anodo) per le principali chimiche di batterie agli ioni di litio [3]

.

Chimica	**Catodo (Elettrodo Positivo)**	**Anodo (Elettrodo Negativo)**
LCO	Ossido di cobalto e litio ($LiCoO_2$)	Grafite
LMO	Ossido di manganese e litio ($LiMn_2O_4$)	Grafite
LFP / LPO	Fosfato di ferro e litio ($LiFePO_4$)	Grafite
NMC	Una combinazione variabile di Nichel, Manganese e Cobalto (es. proporzione classica 1-1-1 o con riduzione di cobalto per limitare i costi)	Grafite
NCA	Ossido di nichel, cobalto e alluminio	Grafite
LTO	Solitamente NMC o LMO	Titanato di litio (sostituisce la grafite per migliorare drasticamente la sicurezza e la velocità di ricarica)

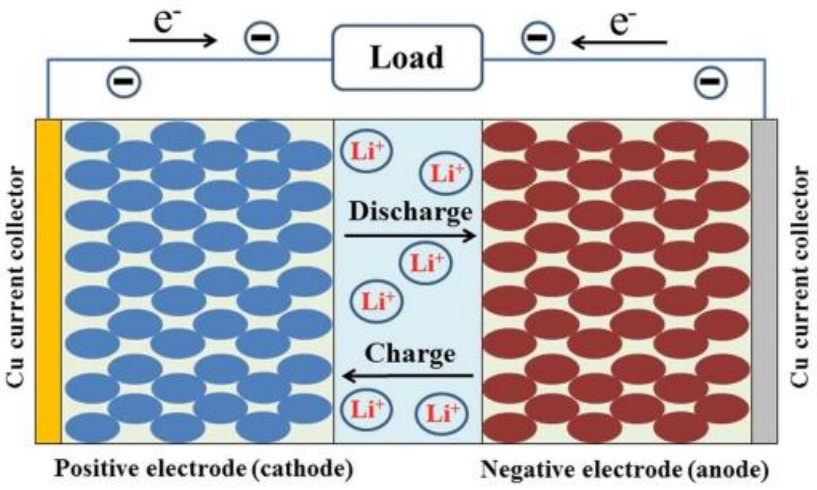

Figura 1.2.: Batteria agli ioni di litio. Tratta da: [3].

.

Celle NCR18650BL

Nei test sono state utilizzate le celle NCR18650BL. Sono della chimica NCA e della famiglia delle celle cilindriche di diametro 18 mm e di lunghezza 65 mm.

Le caratteristiche tecniche principali sono riportate in Tab.1.2.

Tabella 1.2.: Specifiche tecniche principali della cella Sanyo NCR18650BL.[4]

Parametro	**Valore**	**Note / Condizioni di test**
Capacità nominale (Tipica)	3350 mAh	Valore di riferimento con scarica a 0.65 A a 25 °C
Tensione nominale	3.6 V	–
Tensione di carica	4.20 ± 0.03 V	–
Tensione di scarica minima	2.5 V	Discharging End Voltage
Corrente di scarica continua massima	Fino a 4.875 A	Con protezione termica

Figura 1.3.: Celle NCR18650BL

.

1.2. Carica scarica delle batterie

1.2.1. C-rate

Il tempo necessario per caricare o scaricare una batteria dipende da un coefficiente chiamato C-rate, un valore definito in proporzione alla capacità nominale della batteria stessa. Teoricamente, un rate pari a 1C indica che la cella viene caricata o scaricata completamente in un'ora; di conseguenza, una batteria da 1 Ah sottoposta a una corrente di 1C fornirà 1 A per 60 minuti. Se la carica o la scarica avvengono, ad esempio a 0.5C, la corrente si dimezza e il tempo raddoppia. In Tab.1.3 si confrontano a livello teorico il C-rate e il tempo di carica o scarica. [5]

Tabella 1.3.: Relazione teorica tra il valore di C-rate e il rispettivo tempo di carica o scarica completa [5].

C-rate	**Tempo teorico di carica / scarica**
5 C	12 min
2 C	30 min
1 C	1 h
0.5 C (o $C/2$)	2 h
0.2 C (o $C/5$)	5 h
0.1 C (o $C/10$)	10 h
0.05 C (o $C/20$)	20 h

1.2.2. Fasi CC e CV

Le batterie agli ioni di litio si considerano completamente cariche quando, durante la fase di tensione costante (CV), la corrente di carica decresce al di sotto di una soglia predefinita.

La fase iniziale del ciclo prevede una carica a corrente costante (CC), durante la quale la tensione aumenta progressivamente fino a raggiungere il valore limite di fine carica. Per le celle impiegate nei test (vedere Tab.1.2), tale soglia è pari a 4.2 V. L'impiego di correnti elevate, corrispondenti a valori di C-rate maggiori, consente di abbreviare i tempi necessari per completare questa prima fase.

Una volta raggiunto il valore di tensione di fine carica, quest'ultima viene mantenuta costante: ha così inizio la fase a tensione costante (CV). In questa fase, la corrente inizia a decrescere progressivamente. La carica si considera completa quando la corrente scende al di sotto di una soglia compresa tra il 3% e il 5% della capacità nominale della batteria [6]. I risultati di una carica sono riportati in Fig.5.1.

1.2.3. SOC

Lo stato di carica (SOC, *State of Charge*) indica la quantità di energia residua immagazzinata nella cella e viene tipicamente espresso come un valore adimensionale compreso tra 0 e 1, o in formato percentuale (es. 100%). Per la stima del SOC, la maggior parte dei dispositivi impiega il metodo del *Coulomb Counting*. Tale tecnica si basa sul calcolo dell'integrale nel tempo della corrente in ingresso e in uscita, misurando di fatto il flusso netto di carica (espresso in Ampere-secondo o Ampere-ora).

$$\mathrm{SoC}(t) = \mathrm{SoC}(t_0) - \frac{1}{C_\mathrm{n}} \int_{t_0}^{t} \eta \cdot i(t)\,\mathrm{d}t \tag{1.1}$$

Ci sono altri metodi per stimare il SOC come quello della tensione (Voltage Method) e quello con la spettroscopia di Impedenza (Impedance Spectroscopy). [7].

1.3. EIS

L'acronimo EIS sta per *Electrochemical Impedance Spectroscopy* (Spettroscopia di Impedenza Elettrochimica). Si tratta di una tecnica diagnostica non invasiva che consiste nell'applicare uno stimolo elettrico alla batteria ed effettuare una scansione in frequenza, generalmente partendo da valori alti fino a valori molto bassi. Nei test eseguiti, le scansioni esplorano intervalli compresi tra 1 Hz e 1000 Hz, oppure tra 10 mHz e 1 Hz. L'analisi teorica dell'EIS si basa comunemente sulla visualizzazione dei dati tramite il diagramma di Nyquist: questa rappresentazione costituisce un profilo caratteristico che riflette lo stato di salute e le dinamiche interne della cella. Nel diagramma, l'asse orizzontale indica la parte reale dell'impedenza, corrispondente alla risposta puramente resistiva della batteria, mentre l'asse verticale ne indica la parte immaginaria, ovvero la risposta reattiva (capacitiva o induttiva).

Un aspetto fondamentale dell'EIS è la possibilità di poter separare i diversi fenomeni chimico-fisici interni alla cella semplicemente variando la frequenza dello stimolo. Il grafico di Nyquist viene tipicamente interpretato suddividendo la curva in tre regioni principali, procedendo da sinistra verso destra. [8]

- **Regione ad alta frequenza:** collocata nella parte sinistra del diagramma, riflette le dinamiche legate alla migrazione ionica. Tale area è governata in via esclusiva dai contributi puramente resistivi della cella.
- **Regione a media frequenza:** collocata nella parte centrale del diagramma, evidenzia il processo di trasferimento di carica. Rappresenta la reazione elettrochimica vera e propria, che avviene sulla superficie degli elettrodi; un suo allargamento o una sua deformazione è indice di anomalie, perdita di capacità o degradazione dei materiali.
- **Regione a bassa frequenza:** collocata nella porzione destra del diagramma, riflette il fenomeno della diffusione degli ioni all'interno del materiale attivo.

Capitolo 2.

Strumentazione

2.1. Keysight N6705C

Il modello N6705C è classificato come un "DC Power Analyzer", ovvero un mainframe multifunzione da laboratorio. È progettato con un'architettura modulare dotata di 4 slot, in grado di ospitare da uno a quattro moduli di potenza DC o moduli di carico elettronico. Integra in un solo strumento le capacità di un massimo di quattro alimentatori avanzati, un multimetro digitale (DMM), un oscilloscopio, un generatore di forme d'onda arbitrarie e un data logger. [9] [10]

Tabella 2.1.: Specifiche generali del KeysightN6705C: caratteristiche di potenza, interfacce di comunicazione, memoria e tempi di risposta.

Parametro	**Valore / Specifica**
1. Caratteristiche di Potenza e Terminali	
Potenza totale max. disponibile per i moduli	600 W
Corrente max. nominale (terminali frontali)	20 A
2. Interfacce e Controllo Remoto	
LAN	10/100/1000 Base-T Ethernet
USB	USB 2.0 (USBTMC488)
GPIB	Interfaccia SCPI-1993, IEEE 488.2
Web Server	Integrato (richiede un browser web)
3. Memoria e Tempi di Risposta	
Memoria flash interna (salvataggio dati)	4 GB
Tempo di elaborazione del comando	$\leq$ 1 ms

Figura 2.1.: KeysightN6705C Mainframe

.

2.1.1. Modulo N6781A

Il Keysight N6781A è un'unità di sorgente/misura (SMU) dotata di una rete di potenza a quadranti multipli, con modalità separate per la priorità di tensione e corrente. [9] [10]

Nelle seguenti tabelle vengono mostrate le specifiche tecniche principali del modulo.

Tabella 2.2.: Specifiche tecniche del modulo Keysight N6781A: valori nominali, precisione di programmazione/misura e ripple.

Parametro	**Valori massimi / Specifiche**
1. Valori Nominali di Uscita	
Potenza	20 W
Tensione	20 V / 6 V
Corrente	± 1 A / ± 3 A
2. Precisione Principale	
Programmazione Tensione	0.025% + 1.8 mV
Programmazione Corrente	0.04% + 0.3 mA
Misurazione Tensione	0.025% + 1.2 mV
Misurazione Corrente	0.03% + 0.25 mA
3. Ripple e Rumore (PARD)	
CV picco-picco	12 mV
CV rms	1.2 mV

Figura 2.2.: Rappresentazione del modulo di precisione Keysight N6781A.

2.1.2. Modulo N6783A

Il Keysight N6783A è un modulo di potenza DC di base a 2 quadranti progettato per la carica e la scarica delle batterie.[9] [10]

Nelle seguenti tabelle vengono mostrate le specifiche tecniche più rilevanti del modulo.

Tabella 2.3.: Specifiche tecniche del modulo Keysight N6783A: valori nominali, precisione di programmazione/misura e caratteristiche di ripple.

Parametro	**Valore / Specifica**
1. Valori Nominali di Uscita	
Potenza	24 W
Tensione	8 V
Corrente	± 3 A
2. Precisione di Programmazione	
Tensione	0.1% + 10 mV
Corrente	0.1% + 1.8 mA
3. Precisione di Misurazione	
Tensione	0.05% + 5 mV
Corrente	0.1% + 600 μA
4. Ripple e Rumore (PARD)	
Tensione (CV)	8 mV / 1.5 mV
Corrente (CC)	N/A / 2 mA

2.2. Linguaggio SCPI

Lo SCPI (acronimo di Standard Commands for Programmable Instruments) viene definito come un linguaggio di comando per strumenti basato su testo ASCII progettato specificamente per le apparecchiature di test e misurazione. [9]

Il linguaggio è diviso in due macro-categorie:

Comandi comuni (Common Commands IEEE 488.2):

Sono definiti dallo standard IEEE 488.2 per svolgere funzioni di interfaccia generiche (es. reset, lettura dello stato e sincronizzazione). Sono composti da tre lettere e sono sempre preceduti da un asterisco (ad esempio: *RST, *IDN?). [9]

Comandi di sottosistema (Subsystem Commands):

Eseguono funzioni specifiche per il controllo del dispositivo e sono strutturati ad albero, scendendo in percorsi gerarchici dal nodo "radice" in giù. [9]

Esempio grafico delle istruzioni `SOURce:EMULation PS2Q` e `SOURce:EMULation BATTery` del modulo N6781A:

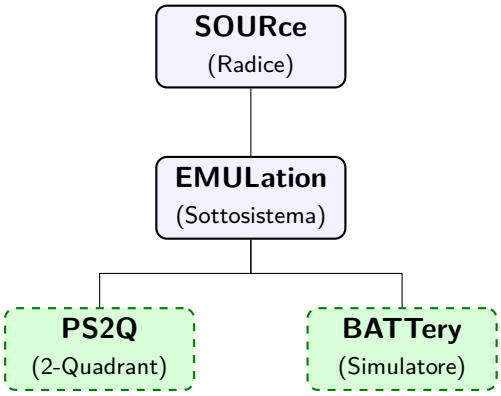

Figura 2.3.: Rappresentazione ad albero della gerarchia dei comandi SCPI di sottosistema.

Parole chiave (Key words)

Costituiscono le istruzioni vere e proprie riconosciute dallo strumento. Possono essere inviate in forma abbreviata per velocizzare la scrittura del codice (solitamente indicate con le sole lettere maiuscole della sintassi) o in forma estesa per una migliore leggibilità del programma.[9]

Interrogazioni (Queries)

Se si fa seguire una parola chiave da un punto interrogativo (?), il comando di impostazione si trasforma in una richiesta di lettura (es. VOLTage? per leggere la tensione impostata in quel momento).[9]

Regole di sintassi e separatori

I "due punti" (:) servono a separare i diversi livelli delle parole chiave all'interno dell'albero gerarchico.

Gli spazi vuoti sono obbligatori per separare il comando dal suo primo parametro.

Le virgole (,) separano i parametri adiacenti quando un comando ne richiede più di uno.

Il "punto e virgola" (;) separa stringhe di comandi diverse inviate in un unico blocco, permettendo di raggruppare le istruzioni in un solo invio.[9]

Esempio:

```
SENSe:DLOG:FUNC:CURRent␣ON;DLOG:PERiod␣0.1<NL>
```

2.3. Modalità a due quadranti PS2Q

Per le funzioni di carica e scarica viene utilizzata la modalità a due quadranti (PS2Q) dei moduli N6781A e N6783A, impostata in priorità di tensione (Voltage Priority). La modalità a due quadranti consente allo strumento di lavorare sia come sorgente (erogando corrente) sia come carico elettronico (assorbendo corrente). L'operazione è limitata a due quadranti con tensione positiva: `+V/+I` (erogazione) e `+V/-I` (assorbimento), un grafico viene riportato in Fig. 2.4.

È possibile programmare lo strumento per lavorare sia in voltage priority che in current priority. Lo strumento in modalità voltage priority controlla la tensione in uscita attraverso un anello di feedback a tensione costante il quale mantiene la tensione di uscita al valore desiderato confrontando la tensione ai capi della batteria e quella impostata come target [9]. Senza la definizione dei limiti di corrente lo strumento imposta la massima corrente possibile per raggiungere la tensione target. Definendo un limite di corrente (Corrente di compliance), è possibile gestire il C-rate nella fase CC (Constant Current) della carica/scarica della batteria. In questa fase lo strumento lavora come generatore di corrente. Una volta che la batteria raggiunge la tensione target, lo strumento lavora come un generatore di tensione e inizia la fase CV (Constant Voltage) e la corrente diminuisce progressivamente seguendo l'assorbimento della batteria.

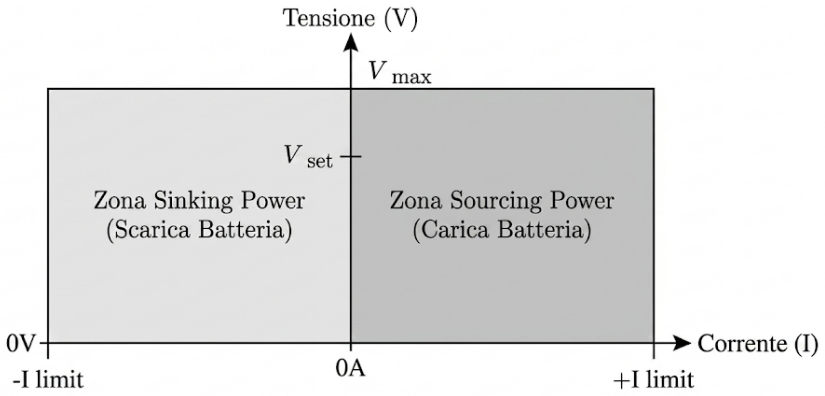

Figura 2.4.: Grafico del voltage priority per i moduli N6781A e N6783A. Riadattato da: [9]

.

2.4. External data logger

L'external data logger non può essere programmato dal pannello frontale. Una volta avviato, la gestione è completamente remota.

Ha una memoria buffer di tipo FIFO che deve essere svuotata periodicamente per evitare la sovrascrittura dei dati.

Il tempo massimo prima che il buffer si riempia è di circa 20 secondi, con il tempo di campionamento minimo pari a 102.4 µs, il numero di punti massimo disponibile è pari a circa 195.000 per un singolo parametro.

Può misurare diversi parametri in base a risoluzione e tempo di integrazione scelto (vedere Tab.2.5). In base al modulo utilizzato questi parametri vengono misurati simultaneamente o in modo alternato (vedere Tab.2.6).

Rispetto all'internal data logger, l'external è indipendente per ogni canale e quindi per ogni comando necessita di specificare come parametro il canale.

Le differenze tra l'internal data logger e l'external data logger sono riportate in Tab.2.4.

Procedura per la programmazione dell'external data logger:

- **Comandi per attivare le misurazioni:**
  Per la tensione: `SENS:ELOG:FUNC:VOLT ON` e per la corrente: `SENS:ELOG:FUNC:CURR ON`.
- **Comandi per il range:**
  Per la tensione: `SENS:ELOG:VOLT:RANG` e per la corrente: `SENS:ELOG:CURR:RANG`.
- **Periodo di integrazione:**
  Comando per specificare periodo di integrazione: `SENS:ELOG:PER` e per la risoluzione `SENS:SWE:TINT:RES`.
- **Inizializzazione:**
  Comando per selezionare il trigger immediato: `TRIG:ELOG:SOUR IMM` per inizializzare e avviare il logger: `INIT:ELOG` e `TRIG:ELOG`.
- **Comando per il recupero dei dati:**
  Il comando per il recupero dei dati è: `FETC:ELOG? <numero di record>`, in un record ci sono le misurazioni di corrente e tensione in base ai parametri scelti.
- **Interruzione:**
  Comando per lo spegnimento: `ABOR:ELOG`

Tabella 2.4.: Confronto tra Built-in Data Logger e External Data Logger [10]

Function	**Built-in Data Logger**	**External Data Logger**
Data viewing	Optimized for viewing the measurements on the power analyzer's display.	No front panel view or front panel control.
Data Storage	Stores the measurements to an internal file. Can be left unattended for an extended time and results can be viewed afterward.	Buffers measurements for about 20 seconds and requires that the computer periodically reads measurements to prevent the power analyzer's buffer from overflowing. The computer needs to provide the data storage function.
Measurement Resources	Allocates all the measurement resources of ALL outputs, even if data logging is only enabled on some of the outputs.	Runs independently on each output. Some outputs can be running an external data log, while the remaining outputs can be used in front panel control or used for other SCPI functions.
Interleaved Mode	Interleaved mode allows the data logger to log both voltage and current on power modules that have only one measurement converter.	Interleaved mode is not available. If a power module has only one measurement converter, then either voltage or current can be logged, but not both.
Logging Rate	Can log data at up to 20.48 microseconds for one parameter.	Can log data at up to 102.4 microseconds for one parameter with data format = real.

Tabella 2.5.: Tempi in base al numero di parametri e alla risoluzione [10]

1 parameter (Voltage or Current), with 20 μs resolution	102.4 μs
2 parameters (Voltage and Current), with 20 μs resolution	204.8 μs
4 parameters (Voltage+Vmin+Vmax+Current), with 20 μs resolution	409.6 μs
8 parameters with 40 μs resolution	819.2 μs
16 parameters with 40 μs resolution	1638.4 μs
24 parameters with 40 μs resolution	2457.6 μs

Tabella 2.6.: Confronto delle caratteristiche di data logging tra i moduli [10]

Caratteristica (Feature)	**Modulo N6781A**	**Modulo N6783A-BAT**
SCPI external data logging	•	•
Simultaneous voltage/current data logging	•	—
Interleaved voltage/current data logging	—	•

Capitolo 3.

Carica della batteria

3.1. Introduzione

Per la carica della batteria a litio sono state create delle funzioni MATLAB capaci di comunicare con lo strumento Keysight N6705C e di impostare i moduli N6781A e N6783A in modo tale da controllare la fase CC e la fase CV della carica o scarica. Le funzioni `chargeN6781A_Vdes`, `chargeN6783A_Vdes`, `chargeN6781A_ageing` e `chargeN6783A_ageing` utilizzano la modalità a due quadranti (PS2Q) (vedere 2.3) per portare la batteria da un valore di tensione desiderato.

3.2. chargeN6781A_Vdes

La funzione `chargeN6781A_Vdes` riportata in List.A.1.1 è una routine che gestisce in modo autonomo la carica e la scarica di una batteria, portando la batteria ad un valore di tensione desiderato.
Utilizza i comandi SCPI per configurare il modulo N6781A e l'internal data logger dello strumento.

Impostazioni e algoritmi utilizzati

Nel codice della funzione vengono impostati i parametri del modulo N6781A per la carica o scarica e i parametri per l'acquisizione dei dati con l'internal data logger. Viene anche implementato un algoritmo per il controllo della corrente di cutoff.

- **Impostazioni data logger:**

  Si impostano le misurazioni di tensione e corrente attraverso i comandi: `SENS:DLOG:FUNC:VOLT ON` e `SENS:DLOG:FUNC:CURR ON`.
  Per impostare la durata del datalogger si utilizza il comando: `SENS:DLOG:TIME 36000`, vengono impostati 36000 secondi come parametro per avere un tempo molto lungo in quanto, successivamente, verrà disattivato manualmente attraverso il comando `:ABORT:DLOG` alla fine della carica/scarica.
  Si imposta il periodo di campionamento attraverso il comando: `SENS:DLOG:PER`.
  Il data logger viene attivato prima impostando il trigger immediato con il comando: `:TRIG:DLOG:SOUR IMM` e poi inizializzandolo mediante il comando `INIT:DLOG "internal:<nome file>"`.

- **Impostazione dei parametri di carica:**

  Viene impostata la modalità a due quadranti attraverso il comando `:SOUR:EMUL PS2Q` e si imposta la voltage priority mediante il comando `:SOUR:FUNC VOLT`, per configurare il valore della tensione desiderato si utilizza il comando `:SOUR:VOLT`. Per definire i limiti di corrente positivo e negativo si utilizza il comando `:SOUR:CURR:LIM`.

- **Avvio della carica:**

  Il canale viene attivato attraverso il comando `:OUTP:STAT ON`.

- **Monitoraggio della corrente:**

  L'acquisizione della corrente avviene ciclicamente attraverso un algoritmo basato sul comando `FETC:DLOG:CURR?`. Il comando interroga lo strumento per ottenere il valore medio della corrente in una finestra di tempo delimitata da due marker posizionati attraverso i comandi: `SENS:DLOG:MARK1:POIN` (marker 1) e `SENS:DLOG:MARK2:POIN` (marker 2). I dati delle misurazioni della corrente vengono estratti dai record salvati dal data logger interno allo strumento. Il valore della corrente misurata viene confrontato ciclicamente con quello di soglia.

- **Spegnimento:**

  Il canale viene disattivato attraverso il comando `:OUTP:STAT OFF` una volta che la corrente ha raggiunto il valore di soglia.

3.3. chargeN6783A_Vdes

La funzione `chargeN6783A_Vdes` riportata in List.A.1.2 è una routine che gestisce in modo autonomo la carica e la scarica di una batteria, portando la batteria ad un valore di tensione desiderato. Permette di gestire contemporaneamente i canali 3 e 4. Utilizza i comandi SCPI per configurare il modulo N6783A e l'internal data logger dello strumento.

Impostazioni e algoritmi utilizzati

Nel codice della funzione vengono impostati i parametri del modulo N6783A per la carica o scarica e i parametri per l'acquisizione dei dati con l'internal data logger. Viene anche implementato un algoritmo per il controllo della corrente di cutoff.

- **Impostazioni data logger:**

  Si impostano le misurazioni di tensione e corrente attraverso i comandi: `SENS:DLOG:FUNC:VOLT ON` e `SENS:DLOG:FUNC:CURR ON`.
  Per impostare la durata del datalogger si utilizza il comando: `SENS:DLOG:TIME`

`36000`, vengono impostati 36000 secondi come parametro per avere un tempo molto lungo in quanto, successivamente, verrà disattivato manualmente attraverso il comando `:ABORT:DLOG` alla fine della carica/scarica.
Si imposta il periodo di campionamento attraverso il comando: `SENS:DLOG:PER`.
Il data logger viene attivato prima impostando il trigger immediato con il comando: `:TRIG:DLOG:SOUR IMM` e poi inizializzandolo mediante il comando `INIT:DLOG "internal:<nome file>"`.

- **Impostazione dei parametri di carica:**

  A differenza del modulo N6781A, per il modulo N6783A non viene utilizzato il comando per impostare la modalità a due quadranti (PS2Q) in quanto nativa del modulo. Operando in voltage priority, viene configurato il valore della tensione di uscita desiderato mediante il comando `:SOUR:VOLT` e, successivamente, vengono definiti i limiti di corrente positivo e negativo attraverso il comando `:SOUR:CURR:LIM` per il valore positivo e `:SOUR:CURR:LIM:NEG` per il valore negativo, nel codice viene utilizzato lo stesso parametro sia per la corrente limite positiva sia per quella negativa. Il modulo N6783A può assorbire al massimo 2 A, se viene impostato un limite di corrente negativo superiore a questo, lo strumento reimposta il limite al suo valore massimo.

- **Avvio della carica:**

  Il canale viene attivato attraverso il comando `:OUTP:STAT ON`.

- **Monitoraggio della corrente:**

  L'acquisizione della corrente avviene ciclicamente attraverso un algoritmo basato sul comando `FETC:DLOG:CURR?`. Il comando interroga lo strumento per ottenere il valore medio della corrente in una finestra di tempo delimitata da due marker posizionati attraverso i comandi: `SENS:DLOG:MARK1:POIN` (marker 1) e `SENS:DLOG:MARK2:POIN` (marker 2). I dati delle misurazioni della corrente vengono estratti dai record salvati dal data logger interno allo strumento. Il valore della corrente misurata viene confrontato ciclicamente con quello di soglia.

- **Spegnimento:**

  Il canale viene disattivato attraverso il comando `:OUTP:STAT OFF` nel momento in cui la corrente raggiunge il valore di soglia.

3.4. chargeN6781A_ageing

La funzione `chargeN6781A_ageing` riportata in List.A.1.3 è una routine che gestisce in modo autonomo la carica e la scarica di una batteria, portando la batteria ad un valore di tensione desiderato. Utilizza i comandi SCPI per configurare il modulo N6781A e l'external data logger dello strumento.

Impostazioni e algoritmi utilizzati

Nel codice della funzione vengono impostati i parametri del modulo N6781A per la carica o scarica e i parametri per l'acquisizione dei dati con l'external data logger. Viene anche implementato un algoritmo per il controllo della corrente di cutoff.

- **Impostazioni external data logger:**

  Viene utilizzato, per evitare conflitti, il comando `SENS:ELOG:FUNC:CURR/VOLT OFF` per disabilitare la misurazione del logger su tensione e corrente. Poi viene impostata la risoluzione attraverso il comando `SENS:SWE:TINT:RES RES20`. I range massimi di tensione e corrente mediante `SENS:ELOG:CURR/VOLT:RANG MAX`. Vengono attivate le misurazioni per corrente e tensione attraverso i comandi `SENS:ELOG:FUNC:CURR/VOLT ON`. Poi viene impostato il periodo di campionamento con il comando `SENS:ELOG:PER`. Viene usato il comando `FORM:DATA REAL` per salvare i dati delle acquisizioni in binario e il comando `FORM:BORD SWAP` per mettere i byte nell'ordine di lettura corretto per MATLAB. Infine viene inizializzato con il comando `INIT:ELOG` e disattivato con il comando `:ABOR:ELOG`. Prima dell'inizializzazione, viene impostato il trigger immediato attraverso il comando `:TRIG:ELOG:SOUR IMM`.

- **Impostazione dei parametri di carica:**

  I comandi e i parametri utilizzati per il modulo N6781A sono gli stessi utilizzati dalla funzione `chargeN6781A_Vdes` (vedere 3.2).

- **Avvio della carica:**

  Il canale viene attivato attraverso il comando `:OUTP:STAT ON`.

- **Monitoraggio della corrente:**

  L'acquisizione della corrente avviene ciclicamente attraverso un algoritmo basato sul comando `FETC:ELOG? <numero di record>`. Il comando interroga lo strumento per ottenere un determinato numero di record. In ogni record ci sono i valori di tensione o di corrente prelevati dal buffer. Il valore della corrente misurata viene confrontato ciclicamente con quello di soglia.

- **Spegnimento:**

  Il canale viene disattivato attraverso il comando `:OUTP:STAT OFF` nel momento in cui la corrente raggiunge il valore di soglia.

3.5. chargeN6783A_ageing

La funzione `chargeN6783A_ageing` riportata in List.A.1.4 è una routine che gestisce in modo autonomo la carica e la scarica di una batteria, portando la batteria ad un valore di tensione desiderato. Permette di gestire contemporaneamente i canali 3 e 4. Utilizza i comandi SCPI per configurare il modulo N6783A e l'external data logger dello strumento.

Impostazioni e algoritmi utilizzati

Nel codice della funzione vengono impostati i parametri del modulo N6783A per la carica o scarica e i parametri per l'acquisizione dei dati con l'external data logger. Viene anche implementato un algoritmo per il controllo della corrente di cutoff.

- **Impostazioni external data logger:**

  Viene utilizzato, per evitare conflitti, il comando `SENS:ELOG:FUNC:CURR/VOLT OFF` per disabilitare la misurazione del logger su tensione e corrente. Poi viene impostata la risoluzione attraverso il comando `SENS:SWE:TINT:RES RES20`. I range massimi di tensione e corrente mediante `SENS:ELOG:CURR/VOLT:RANG MAX`. Vengono attivate le misurazioni per corrente e tensione attraverso i comandi `SENS:ELOG:FUNC:CURR/VOLT ON`. Poi viene impostato il periodo di campionamento con il comando `SENS:ELOG:PER`. Viene usato il comando `FORM:DATA REAL` per salvare i dati delle acquisizioni in binario e il comando `FORM:BORD SWAP` per mettere i byte nell'ordine di lettura corretto per MATLAB. Infine viene attivato con il comando `INIT:ELOG` e disattivato con il comando `:ABOR:ELOG`. Prima dell'inizializzazione, viene impostato il trigger immediato attraverso il comando `:TRIG:ELOG:SOUR IMM`.

- **Impostazione dei parametri di carica:**

  I comandi e i parametri utilizzati per il modulo N6783A sono gli stessi utilizzati dalla funzione `chargeN6783A_Vdes` (vedere 3.3).

- **Avvio della carica:**

  Il canale viene attivato attraverso il comando `:OUTP:STAT ON`.

- **Monitoraggio della corrente:**

  L'acquisizione della corrente avviene ciclicamente attraverso un algoritmo basato sul comando `FETC:ELOG? <numero di record>`. Il comando interroga lo strumento per ottenere un determinato numero di record. In ogni record ci sono i valori di tensione o di corrente prelevati dal buffer. A differenza del modulo N6781A, il modulo N6783A non permette di misurare tensione e corrente simultaneamente (vedere 2.4). Il valore della corrente misurata viene confrontato ciclicamente con quello di soglia.

- **Spegnimento:**

  Il canale viene disattivato attraverso il comando `:OUTP:STAT OFF` nel momento in cui la corrente raggiunge il valore di soglia.

Capitolo 4.

EIS

4.1. Introduzione

Al fine di eseguire la caratterizzazione EIS, sono state sviluppate apposite funzioni MATLAB in grado di interfacciarsi con il mainframe Keysight N6705C, equipaggiato con il modulo N6781A. L'obiettivo di queste funzioni è sfruttare appieno le capacità dello strumento come SMU (Source Measure Unit), generando i segnali sinusoidali per la stimolazione della batteria e acquisendo simultaneamente i valori di tensione e corrente di risposta.

4.2. Funzione charge_soc

La funzione `charge_soc` riportata in List.A.5 è una routine che porta la batteria ad un valore di state of charge (SOC) desiderato. Utilizza la modalità arbitraria (ARB) e la modalità a due quadranti (PS2Q) del modulo N6781A in priorità di corrente. Tale funzione si rivela fondamentale per preparare la batteria alle caratterizzazioni EIS a diversi stati di carica.

Impostazioni e algoritmi utilizzati

- **Impostazioni del modulo:**

  Il modulo viene impostato in modalità a due quadranti attraverso il comando `:SOUR:EMUL PS2Q` e in priorità di corrente attraverso il comando `FUNC CURR`. Poi viene impostata la modalità arbitraria in priorità di corrente attraverso il comando `ARB:FUNC:TYPE CURR`. Con il comando `ARB:FUNC:SHAP STEP` si comunica allo strumento che il segnale deve essere un gradino, e il valore di corrente viene impostato con `ARB:CURR:STEP:STAR:LEV`, il parametro è un valore negativo in quanto la corrente deve essere assorbita dallo strumento. Viene impostata la durata del segnale con il comando `ARB:CURR:STEP:END:TIM` come parametro viene impostato un secondo e si mantiene il segnale al valore impostato con il comando `ARB:TERM:LAST ON`.

- **Calcolo della carica:**

  Prima la batteria viene caricata al 100% attraverso la funzione `chargeN6781A_ageing`, vedere ListA.1.3. Poi, in base al parametro che viene inserito come SOC desiderato, viene calcolato il tempo per rimuovere il quantitativo di carica necessario per raggiungere quel valore.

- **Spegnimento:**

  Una volta terminato il tempo di scarica, viene posta la corrente pari a zero con il comando `CURR 0` e disattivato il canale con `OUTP OFF`.

4.3. Funzione SinGen_multi

La funzione `SinGen_multi`, riportata in Lis.A.6, si occupa di generare le sinusoidi di corrente per l'identificazione della batteria. L'algoritmo prende in ingresso i parametri relativi al range di frequenze, al numero di frequenze, al tempo di campionamento minimo, all'ampiezza della sinusoide e al numero di campioni per ogni frequenza, al fine di generare i segnali corrispondenti. Tutte le forme d'onda generate vengono infine salvate in una struttura dati chiamata `freqs`. Alcuni esempi sono riportati nella Tab.4.1.

Tabella 4.1.: Scenari di configurazione dei parametri della struttura p per l'algoritmo SinGen.

Parametro	**Descrizione / Unità**	**Caso Standard**	**Bassa Frequenza**
`p.fmin`	Freq. minima [Hz]	1	0.01
`p.fmax`	Freq. massima [Hz]	1000	1
`p.Nf`	Numero totale frequenze	30	10
`p.Ts`	Periodo campionamento [s]	204,8e-6	0.1
`p.amp`	Ampiezza sinusoidi [A]	1	0.5
`p.Np`	Numero punti per frequenza	10000	512

L'algoritmo si basa sul calcolo del tempo di campionamento e del numero di sinusoidi per ciascuna frequenza, adattando tali parametri in base al numero di punti impostato. Il relativo diagramma di flusso è illustrato in Fig. 4.1.

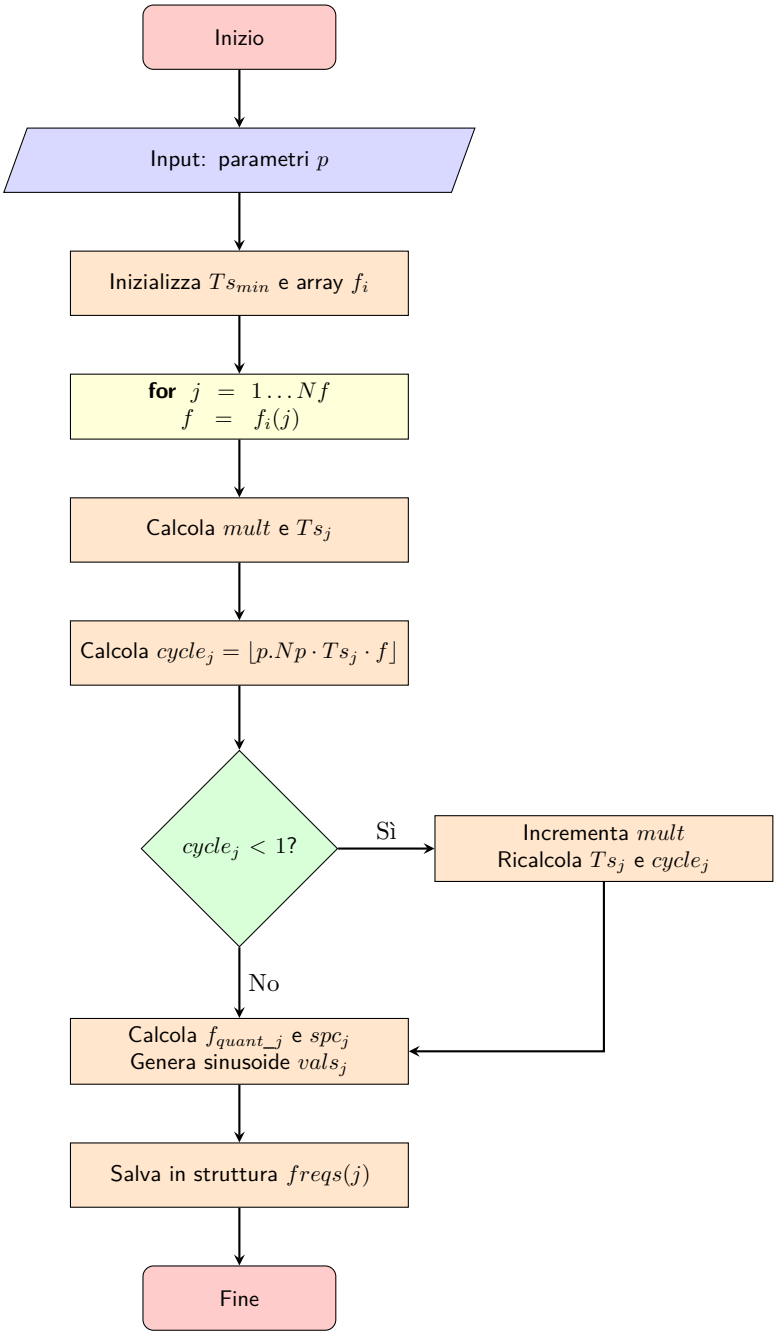

Figura 4.1.: Diagramma a blocchi dell'algoritmo SinGen.

4.4. Funzione measure

La funzione `measure`, riportata nel List. A.7, esegue l'identificazione della batteria utilizzando la modalità arbitraria (ARB) dello strumento per la generazione delle sinusoidi e l'external data logger per l'acquisizione dei dati.

Impostazioni e algoritmo utilizzato
La funzione configura lo strumento per caricare i valori delle sinusoidi tramite la modalità ARB. Per ciascuna frequenza, viene erogato il segnale di corrente e vengono simultaneamente acquisiti i valori di tensione e corrente in risposta dalla batteria.

- **Impostazioni External logger:** Le impostazioni del logger sono le stesse utilizzate per le funzioni `chargeN6781A_ageing` e `chargeN6783A_ageing` (vedere List. A.3 e A.4). Nello specifico, la configurazione ricalca quella del modulo N6781A, poiché vengono abilitate le misurazioni sia per la corrente che per la tensione. Per i comandi SCPI vedere 2.4.

- **Impostazioni modulo N6781A e ARB:** Il modulo viene configurato in modalità due quadranti (PS2Q) e con priorità di corrente inviando i comandi `:SOUR:EMUL PS2Q` e `FUNC CURR`. Successivamente, si abilita la funzione ARB specificando la natura in corrente del segnale tramite `ARB:FUNC:TYPE CURR` e definendo la forma d'onda con `ARB:FUNC:SHAP CDW`. Infine, i campioni che compongono la sinusoide vengono caricati nella memoria dello strumento utilizzando il comando `SOUR:ARB:CURR:CDW:LEV`.

- **Identificazione:** Mediante un ciclo iterativo, per ciascuna delle frequenze selezionate viene attivata la generazione ARB per erogare le sinusoidi di corrente di stimolo. Simultaneamente, vengono acquisiti i valori di tensione e corrente restituiti dalla batteria in risposta alla sollecitazione.

- **Salvataggio dati:** Al termine delle misurazioni, i dati acquisiti vengono memorizzati all'interno di due distinte matrici (rispettivamente per la tensione e per la corrente). In ciascuna struttura, ogni colonna è associata a una determinata frequenza di test, mentre le righe contengono l'evoluzione temporale dei campioni registrati.

4.5. Funzione plot_eis

La funzione `plot_eis`, riportata in List.A.8, elabora i valori di tensione e corrente acquisiti per determinare l'impedenza complessa mediante analisi FFT per ciascuna frequenza di test. Al termine del calcolo, la funzione genera e formatta automaticamente il diagramma di Nyquist relativo alla sessione di misura.

- **Estrazione dati:** per ogni frequenza estrae il valore dei dati di tensione e corrente.
- **Calcolo FFT:** viene calcolata la trasformata di Fourier (FFT) per entrambi i segnali. L'algoritmo identifica l'indice di picco spettrale fidx all'interno dello spettro di corrente, che corrisponde alla frequenza di stimolo applicata.
- **Calcolo dell'impedenza:** l'impedenza viene determinata calcolando il rapporto tra fasori di tensione e corrente. Tale calcolo restituisce un valore complesso che contiene sia l'informazione sull'ampiezza che sulla fase del segnale.
- **Generazione del diagramma di Nyquist:** I valori complessi ottenuti vengono normalizzati (moltiplicati per 1000 per rappresentarli in $m\Omega$) e riportati sul piano complesso, dove l'asse delle ascisse rappresenta la parte reale $\Re(Z)$ e quello delle ordinate la parte negativa della parte immaginaria $-\Im(Z)$.

Capitolo 5.

Risultati

5.1. Carica e scarica

In Fig.5.1 viene rappresentato il risultato di una carica effettuata con il modulo N6781A attraverso la funzione `chargeN6781A_Vdes` sulla cella 2. Come si può notare dal grafico la batteria è stata portata alla tensione di 4 V partendo dalla di tensione di 3.6 V. La tensione è indicata con la curva blu e la corrente con la curva arancione, si può notare la fase CC dove la corrente rimane costante e la tensione aumenta e rispettivamente la fase CV dove la corrente diminuisce e la tensione rimane costante. Nella fase CV la corrente arriva al valore di soglia che in questo caso è stato impostato a 75 mA.

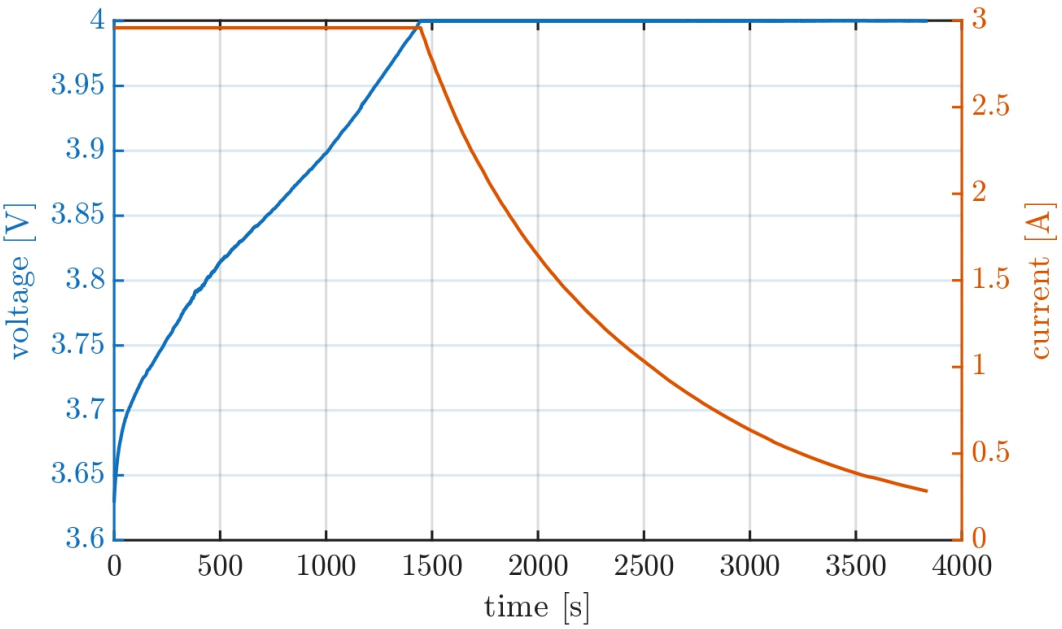

Figura 5.1.: Carica della cella 2 da 3.6 V a 4 V.

.

In Fig.5.2 viene rappresentato il risultato della scarica effettuata con il modulo N6781A attraverso la funzione `chargeN6781A_Vdes` sulla cella 2. La tensione in questo caso è stata portata da 3.8 V a 3.6 V. Anche qui si possono notare le fasi CC e CV e il valore di corrente di soglia pari a 0.9 A.

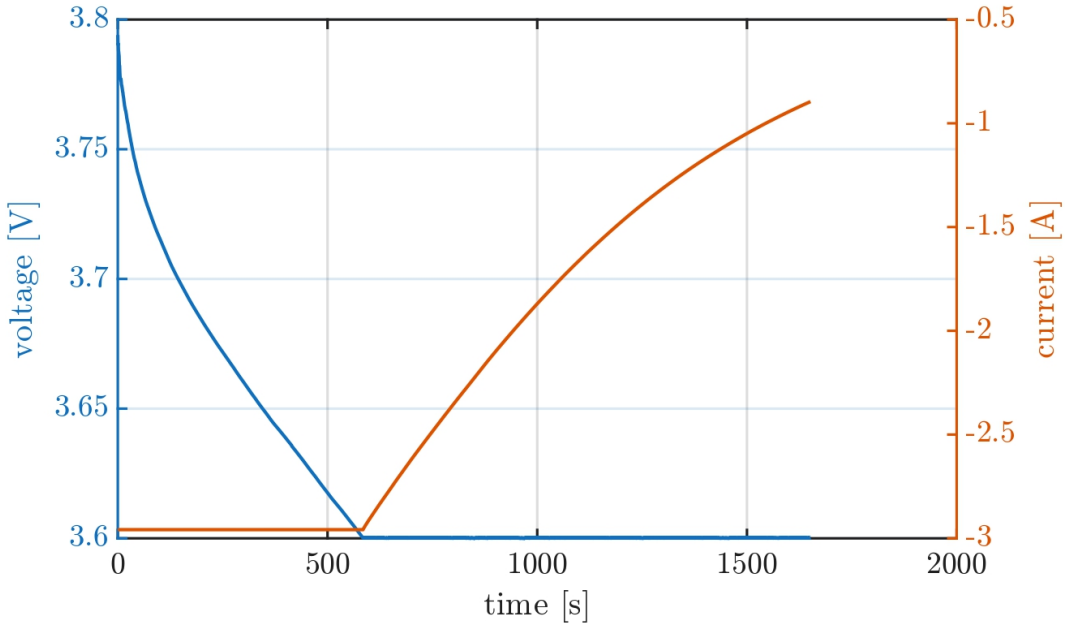

Figura 5.2.: Scarica della cella 2 da 3.8 V a 3.6 V.

.

In entrambi i processi di carica e scarica, il transitorio tra la fase a corrente costante (CC) e quella a tensione costante (CV) risulta netto e del tutto privo di sbalzi di tensione (overshoot). Tale stabilità conferma l'efficacia della configurazione dello strumento in modalità a due quadranti (PS2Q) e dell'impostazione di voltage priority gestite dagli script. Infine, l'algoritmo di monitoraggio della corrente di soglia interviene in modo accurato, disattivando l'uscita dello strumento all'esatto raggiungimento del valore limite.

5.2. EIS

Ai fini della caratterizzazione EIS della cella 2, sono stati analizzati differenti stati di carica (SOC 1.0, 0.9 e 0.4) su due distinti intervalli di frequenza (1 Hz – 1000 Hz e 10 mHz – 1 Hz). L'integrità del segnale acquisito è documentata mediante l'analisi nel dominio del tempo delle forme d'onda sinusoidali e la successiva rappresentazione nel dominio della frequenza (FFT), le quali testimoniano la corretta sincronizzazione tra l'algoritmo di generazione e l'acquisizione tramite il logger.

La Figura 5.3 mostra il diagramma di Nyquist relativo alla cella 2, analizzata a uno stato di carica (SOC) del 100%. Il segnale di eccitazione è stato generato mediante la funzione `SinGen_multi`, impostando una scansione su 30 frequenze (Nf = 30) distribuite nell'intervallo compreso tra 1 Hz e 1000 Hz. La curva ottenuta permette di distinguere chiaramente le diverse regioni caratteristiche del comportamento elettrochimico della cella all'interno del range selezionato: nello specifico, si osserva la zona ad alta frequenza, dominata dalla resistenza ohmica e dai fenomeni di migrazione ionica, e la regione a media frequenza, caratterizzata dal processo di trasferimento di carica all'interfaccia elettrodo-elettrolita.

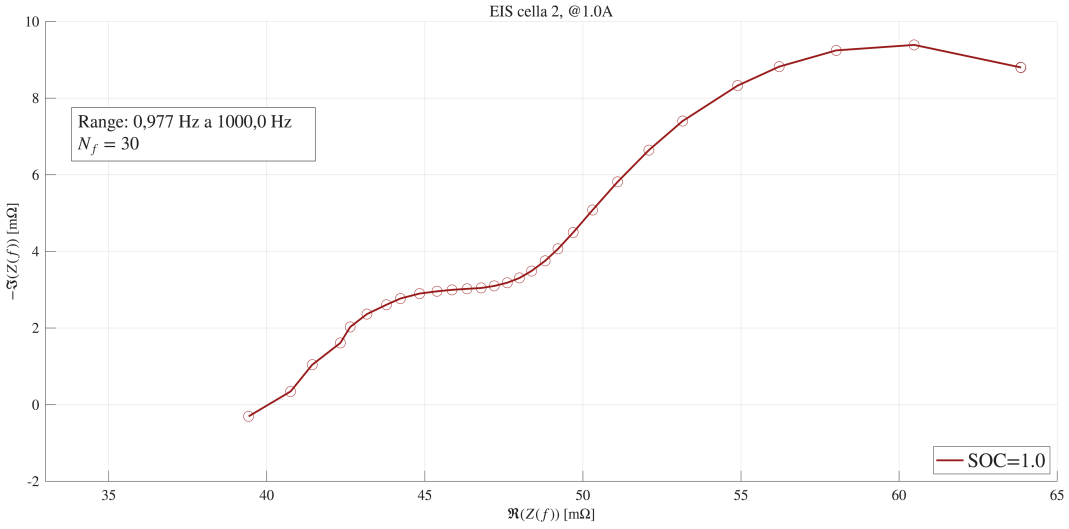

Figura 5.3.: Grafico di Nyquist della cella 2 con SOC=1. Range 1 Hz – 1000 Hz.

.

Nella Tabella 5.1 sono riportate le frequenze di test generate dalla funzione `SinGen_multi`; tali valori sono stati distribuiti secondo una progressione logaritmica per garantire un'adeguata risoluzione spettrale su tutto il range di interesse e per minimizzare gli effetti di spectral leakage durante l'analisi FFT.

Tabella 5.1.: Elenco delle frequenze utilizzate per la caratterizzazione EIS (Nf = 30).

Indice	**Freq. [Hz]**	**Indice**	**Freq. [Hz]**
1	0.98	16	35.16
2	0.98	17	44.92
3	1.46	18	57.13
4	1.95	19	72.75
5	2.44	20	92.29
6	2.93	21	117.19
7	3.91	22	148.44
8	4.88	23	188.48
9	6.35	24	239.26
10	8.30	25	303.71
11	10.74	26	385.25
12	13.67	27	489.26
13	17.09	28	620.61
14	21.97	29	787.60
15	27.83	30	1000.00

La Figura 5.4 illustra le forme d'onda sinusoidali di tensione e corrente acquisite per diverse frequenze di test. Come si può osservare, per ogni frequenza sono stati eseguiti molteplici cicli; in particolare, il numero di periodi acquisiti aumenta proporzionalmente all'aumentare della frequenza, al fine di garantire un'adeguata risoluzione temporale e una maggiore precisione nell'analisi spettrale.

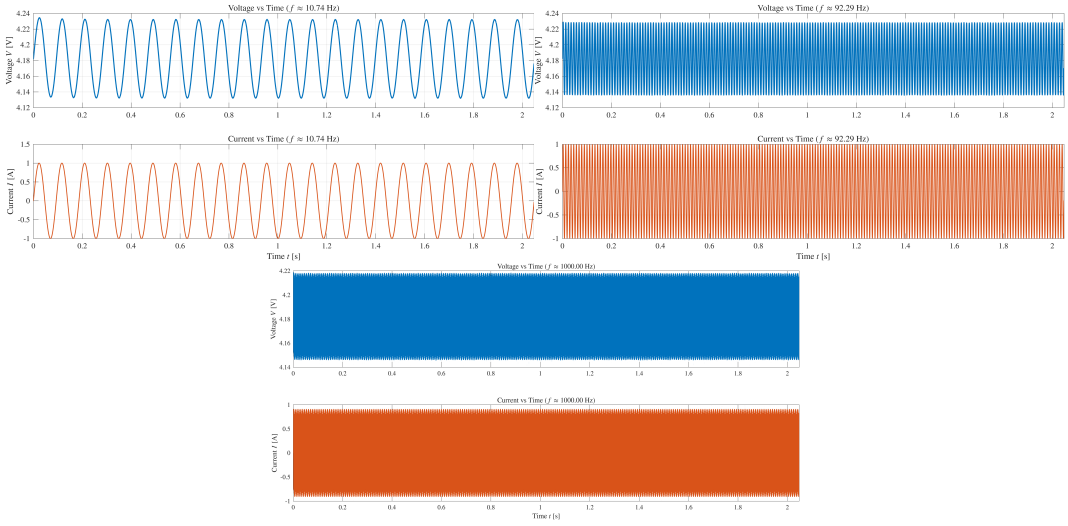

Figura 5.4.: Sinusoidi di tensione e corrente delle frequenze 10 Hz, 90 Hz e 1000 Hz.

In Figura 5.5 è riportata l'analisi spettrale di tensione e corrente per tutte le frequenze analizzate. Il grafico evidenzia la presenza di picchi netti in corrispondenza delle frequenze di test, confermando la precisa corrispondenza tra i segnali di stimolo impostati e la risposta misurata dal sistema.

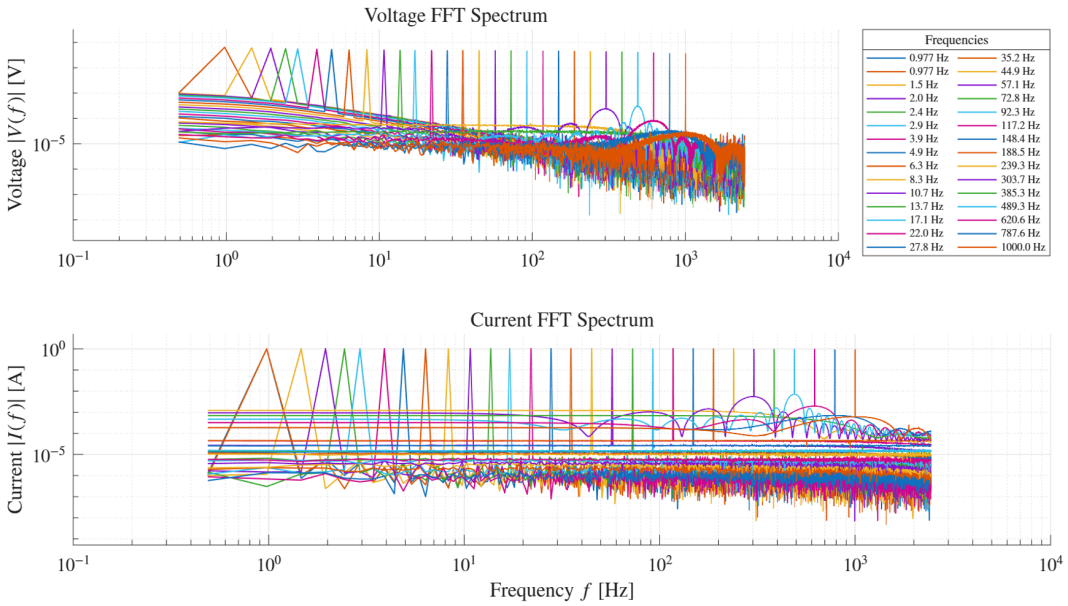

Figura 5.5.: Spettro delle sinusoidi di tensione e corrente di tutte le frequenze. Range 1 Hz a 1000 Hz

.

La Figura 5.6 presenta il diagramma di Nyquist relativo al range di basse frequenze (10 mHz – 1 Hz), fondamentale per l'analisi dei fenomeni diffusivi all'interno della cella. In tale grafico è chiaramente distinguibile l'impedenza di Warburg, caratterizzata dalla tipica pendenza lineare a 45 gradi. La coerenza dei segnali acquisiti in questo intervallo è documentata nelle Figure 5.7 e 5.8, che riportano rispettivamente le forme d'onda nel dominio del tempo e l'analisi spettrale di tensione e corrente. Infine, la Tabella 5.2 elenca le frequenze di test impostate per questa analisi.

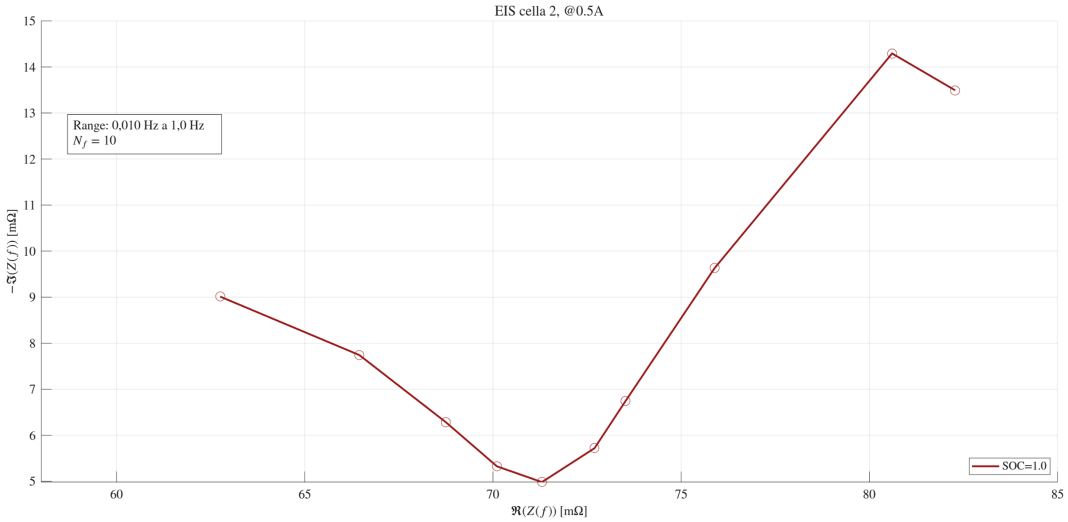

Figura 5.6.: Grafico di Nyquist della cella 2. Range 10 mHz – 1 Hz.

.

Tabella 5.2.: Elenco delle frequenze utilizzate per il range a bassa frequenza (10 mHz – 1 Hz).

Indice	**Frequenza [Hz]**
1	0.010
2	0.010
3	0.020
4	0.039
5	0.059
6	0.117
7	0.215
8	0.352
9	0.586
10	0.996

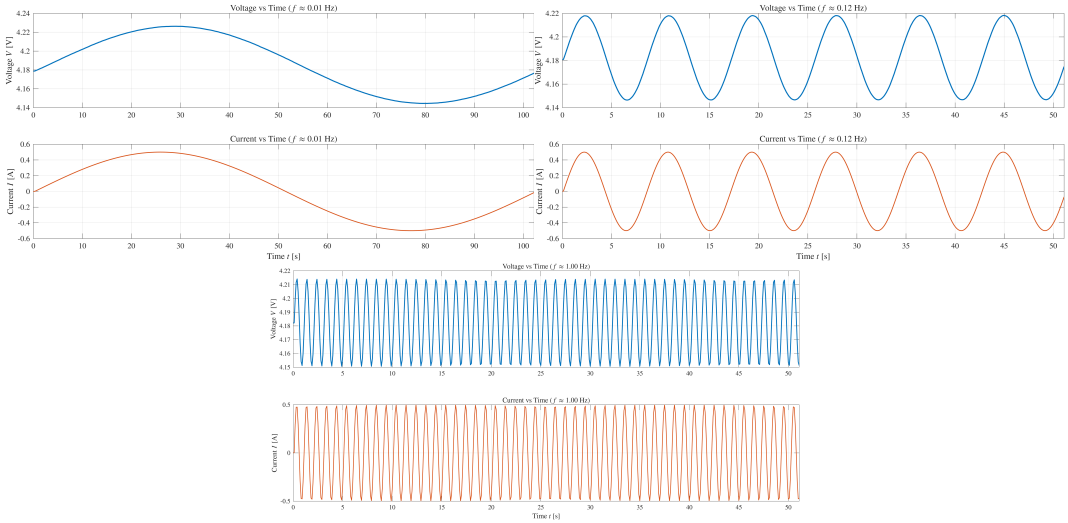

Figura 5.7.: Sinusoidi di tensione e corrente per le frequenze 1 Hz, 0.010 Hz e 0.12 Hz.

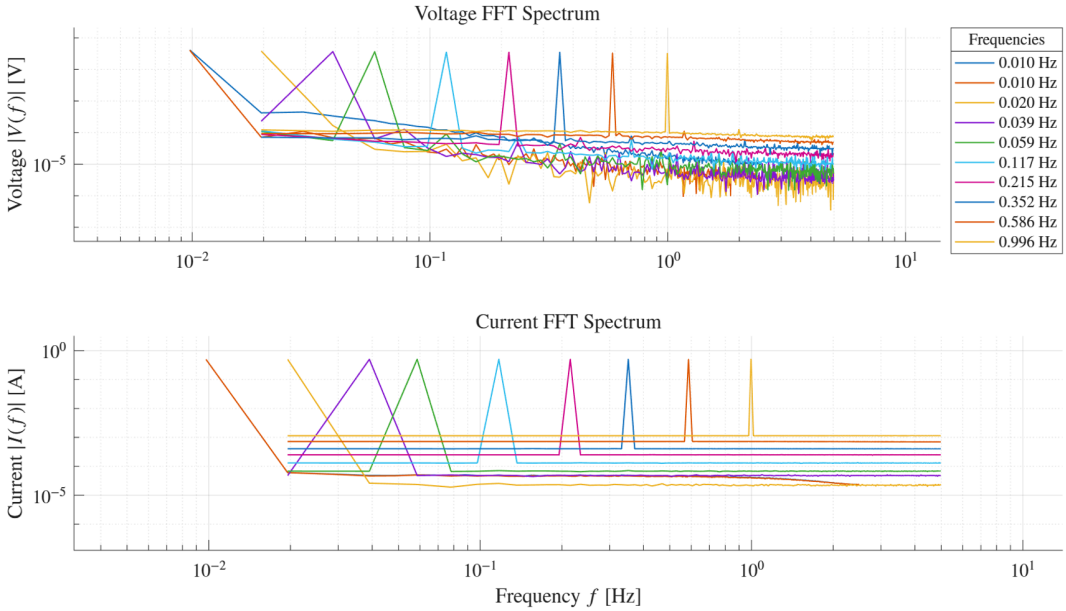

Figura 5.8.: Spettro delle sinusoidi di tensione e corrente di tutte le frequenze. Range 10 mHz a 1 Hz.

.

La Figura 5.9 mostra il diagramma di Nyquist della cella 2, analizzata a uno stato di carica (SOC) del 90% nell'intervallo di frequenza compreso tra 1 Hz e 1000 Hz. Rispetto alla condizione precedente, si osserva un'evoluzione del profilo della curva, indicativa di una variazione del comportamento elettrochimico della cella. Le Figure 5.10 illustrano i relativi spettri di tensione e corrente, mentre la Tabella 5.3 riporta l'elenco delle frequenze di test adottate per la presente sessione sperimentale.

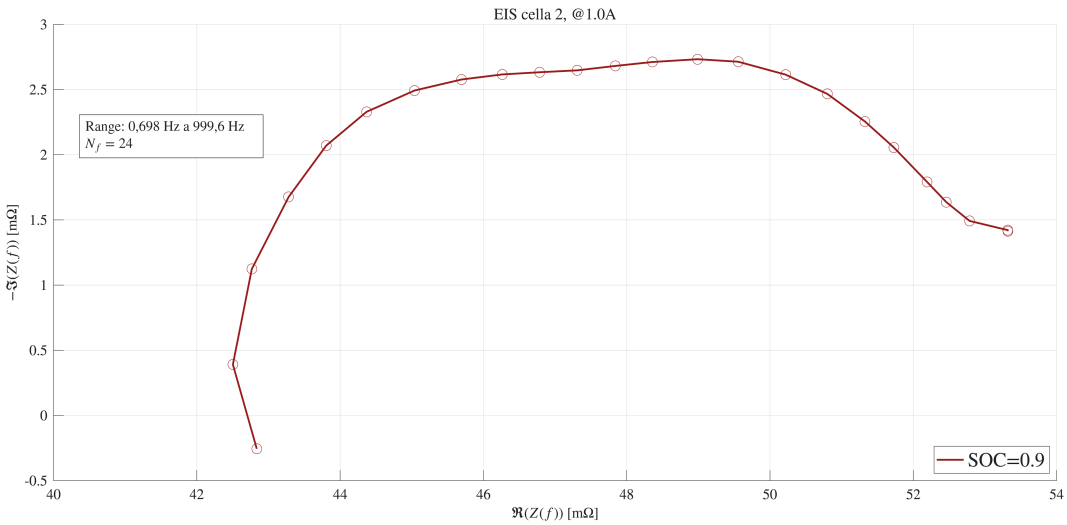

Figura 5.9.: Grafico di Nyquist della cella 2 con un SOC pari a 0.9.

.

Tabella 5.3.: Elenco delle frequenze di test per la caratterizzazione EIS. SOC=0.9

Indice	**Freq. [Hz]**	**Indice**	**Freq. [Hz]**
1	0.70	13	36.27
2	0.70	14	49.53
3	1.40	15	66.96
4	2.09	16	89.98
5	2.79	17	122.07
6	4.19	18	164.62
7	5.58	19	222.52
8	7.67	20	300.64
9	10.46	21	405.97
10	14.65	22	548.27
11	19.53	23	740.09
12	27.20	24	999.58

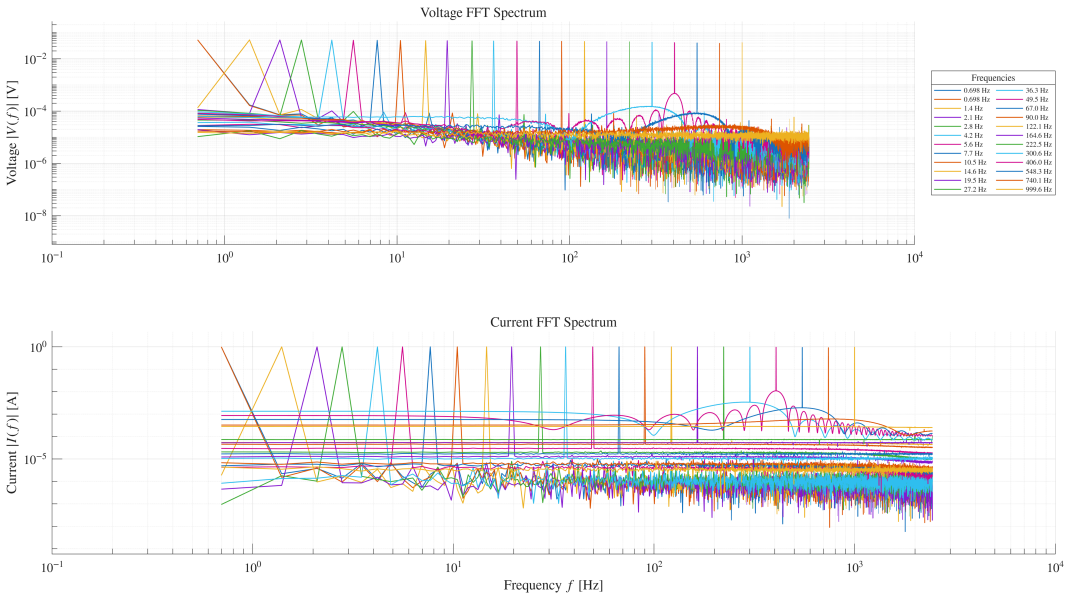

Figura 5.10.: Spettro delle sinusoidi di tensione e corrente di tutte le frequenze. Range 1 Hz a 1000 Hz per un SOC = 0.9.

.

La Figura 5.11 riporta il diagramma di Nyquist della cella 2, analizzata a uno stato di carica (SOC) pari al 40%. Il grafico illustra l'evoluzione del profilo della curva in condizioni di basso livello di carica; il relativo set di frequenze di test è specificato nella legenda del grafico stesso.

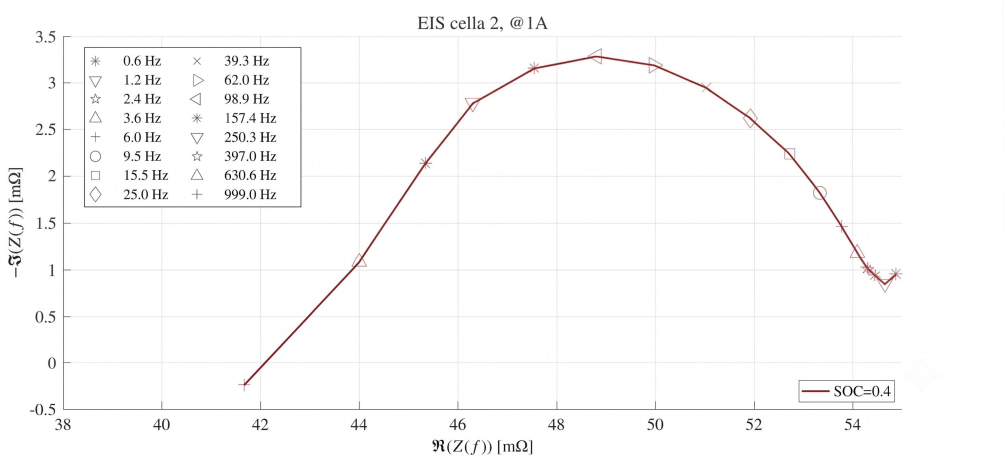

Figura 5.11.: Grafico di Nyquist della cella 2 con un SOC pari a 0.4.

.

Appendice A.

Listati funzioni MATLAB

A.1. Codici delle funzioni per la carica

A.1.1. Codice chargeN6781A_Vdes

Listing A.1: Codice della funzione chargeN6781A_Vdes. Licenza GNU GPL v3

```
function chargeN6781A_Vdes(DEV, B, OVP, channel, cutoff_curr, Tc,
    dlog_flag)

%function chargeN6781A_Vdes(DEV, B, OVP, channel, cutoff_curr, Tc)
%
% Charge/discharge routine to a desired voltage via Internal Data
    Logger (Dlog);
%
%% Inputs
%
% DEV       -> KEYSIGHT Device
% OVP       -> Set the value of the OVER VOLTAGE PROTECTION
%              [4.5 V Default]
% B         -> Battery parameters
% B.cap     -> mean capacity of the battery  [Ah]
% B.curr_C  -> The current at which operate in C unit
% B.v_max   -> desired battery voltage [V]
%              for backward compatibility the name of the field is not
    changed
% channel   -> Channel index [Channel 1 Default]
% cutoff_curr -> low current limit
% dlog_flag -> data logger flag, boolean (true = save file)

% Copyright (C) 2024, A. Goffi, N. Ceccarelli
% Copyright (C) 2025 G. Leandrini, S. Orcioni
% Copyright (C) 2026 S. Paparelli
%This program is free software: you can redistribute it and/or modify
%it under the terms of the GNU General Public License as published by
%the Free Software Foundation, either version 3 of the License, or
%(at your option) any later version.
%This program is distributed in the hope that it will be useful,
%but WITHOUT ANY WARRANTY; without even the implied warranty of
%MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
```

```
%GNU General Public License for more details.
%You should have received a copy of the GNU General Public License
%along with this program.  If not, see <http://www.gnu.org/licenses/>.

if(~exist('DEV','var'))
  error('Unspecified device', 'Error');
    end
if(~exist('OVP', 'var'))
    OVP=4.7;
    end
if(~exist('channel','var'))
    channel=1;
    end
if(~exist('cutoff_curr','var'))
    cutoff_curr = 0.075;
    end
if(exist('log_file','var'))
    % open CSV file for logging
    fid = fopen(log_file, 'w');
    fprintf(fid, 'Timestamp,Current (A)\n');
end
if(~exist('Tc','var'))
 Tc = 0.3;
    end
if(~exist('dlog_flag','var'))
        dlog_flag = true;
    end

curr = abs(B.curr_C)*B.cap;

%% DATALOGGER SETTINGS
% TURN OFF ALL DATALOGGING CHANNELS
writeline(DEV, sprintf('SENS:DLOG:FUNC:CURR OFF,(@1,2,3,4)'));
writeline(DEV, sprintf('SENS:DLOG:FUNC:VOLT OFF,(@1,2,3,4)'));

% ACTIVATE CURRENT DATALOGGER
writeline(DEV, sprintf('SENS:DLOG:FUNC:CURR ON,(@%d)',channel));
% ACTIVATE VOLTAGE DATALOGGER
writeline(DEV, sprintf('SENS:DLOG:FUNC:VOLT ON,(@%d)',channel));

% SET THE DATALOGGING DURATION
% %(it will be forced off at the end of the charge) (36000s = 10h)
writeline(DEV, sprintf('SENS:DLOG:TIME 36000'));

% SET THE DATALOGGER SAMPLING PERIOD
writeline(DEV, sprintf('SENS:DLOG:PER %f', Tc));

%% N6781A SETTINGS
disp('Setting Charge parameters');
% Set 2 quadrant power supply mode
writeline(DEV,sprintf(':SOUR:EMUL PS2Q, (@%d)',channel));
```

```
% Set operating in voltage priority
writeline(DEV,sprintf(':SOUR:FUNC VOLT, (@%d)',channel));
% Set target voltage
writeline(DEV,sprintf(':SOUR:VOLT %f, (@%d)', B.v_max, channel));
% Set maximum positive and negative current limit
writeline(DEV,sprintf(':SOUR:CURR:LIM %f, (@%d)', abs(curr),
    channel));
% Set the protection voltage
writeline(DEV,sprintf('VOLT:PROT:REM %f, (@%d)', OVP, channel));


%%  TURN ON DLOGGER (and start tic)
writeline(DEV, sprintf(':TRIG:DLOG:SOUR IMM'));
if dlog_flag

    d=string(datetime);
    d=strrep(d,":","-");
    d=strrep(d," ","-");
    writeline(DEV, sprintf('INIT:DLOG
        "internal:\\chargeN6781A-%s.dlog"',d));

else
    % Overwrites the default file
    writeline(DEV, 'INIT:DLOG "internal:\\default.dlog"');
end

tic
%% N6781A ON
writeline(DEV,sprintf(':OUTP:STAT ON, (@%d)',channel));
disp ('Charge started')
%%  CHARGING ALGORITHM
pause(10*Tc)
current_now = cutoff_curr + 1;
while (abs(current_now) > cutoff_curr)
    t1 = toc;
    writeline(DEV,sprintf('SENS:DLOG:MARK2:POIN %.1f',t1-Tc));
    writeline(DEV,sprintf('SENS:DLOG:MARK1:POIN %.1f',t1 - 5 * Tc));
    writeline(DEV,sprintf('FETC:DLOG:CURR? (@%d)',channel));
    current_now = str2double(readline(DEV));
    fprintf('Tempo: %.1f s | Corrente: %.4f A\n', t1, current_now);
    %disp(current_now)
    pause(10*Tc)
end
disp("CHARGE ENDED")

% deactivate the output
writeline(DEV,sprintf(':OUTP:STAT OFF, (@%d)', channel));
disp ('Charge completed')

if(exist('log_file','var'))
    fclose(fid);
```

```
end
%% TURN OFF DLOG

writeline(DEV, sprintf(':ABOR:DLOG'));

end
```

A.1.2. Codice chargeN6783A_Vdes

Listing A.2: Codice della funzione chargeN6783A_Vdes. Licenza GNU GPL v3

```
function chargeN6783A_Vdes(DEV, charge_v, charge_c, ...
    channel, cutoff_current, Tc, dlog_flag)

% function chargeN6783A_Vdes(DEV, charge_v, charge_c, ...
%     channel, cutoff_current, Tc, dlog_flag)
%
% Charge/discharge routine via Internal Data Logger (Dlog);
%
%% Inputs
% DEV             -> TCP client ID
% charge_voltage -> vector of desired voltages [V]
% charge_current -> vector of current compliances (positive) [A]
% cutoff_current -> charge stop current limit
% Tc              -> time between current measurements
% channel         -> vector of channels (only 3 or 4)
% dlog_flag       -> data logger flag, boolean (true = save file)

% Copyright (C) 2025 G. Leandrini, S. Orcioni
% Copyright (C) 2026 S. Paparelli
%This program is free software: you can redistribute it and/or modify
%it under the terms of the GNU General Public License as published by
%the Free Software Foundation, either version 3 of the License, or
%(at your option) any later version.
%This program is distributed in the hope that it will be useful,
%but WITHOUT ANY WARRANTY; without even the implied warranty of
%MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%GNU General Public License for more details.
%You should have received a copy of the GNU General Public License
%along with this program.  If not, see <http://www.gnu.org/licenses/>.

%% Default Parameters Check
if(~exist('cutoff_current','var'))
    cutoff_current = 0.075;
end

if(~exist('Tc','var'))
    Tc = 0.3;
end
if(~exist('dlog_flag','var'))
    dlog_flag = true;
end
```

```
channel_string = sprintf('(@%s)', join(string(channel), ','));


%% DATALOGGER SETTINGS AND N6783A-BAT SETTINGS
% SET THE DATALOGGING DURATION
% %(it will be forced off at the end of the charge) (36000s = 10h)
writeline(DEV, sprintf('SENS:DLOG:TIME 36000'));
% SET THE DATALOGGER SAMPLING PERIOD
writeline(DEV, sprintf('SENS:DLOG:PER %f', Tc));
% TURN OFF DATALOGGING CHANNELS
writeline(DEV, sprintf('SENS:DLOG:FUNC:CURR OFF,(@1,2,3,4)'));
writeline(DEV, sprintf('SENS:DLOG:FUNC:VOLT OFF,(@1,2,3,4)'));

N = length(channel);
for i = 1:N

    channel_num = channel(i);
    % min(i, end) ensures that if the loop index 'i' exceeds the
        array length,
    % it remains at the last available value (clamping the index).
    v_target = charge_v(min(i,end));
    c_target = abs(charge_c(min(i,end)));

    %% DATA LOGGER SETTINGS
    % ACTIVATE CURRENT DATALOGGER
    writeline(DEV, sprintf('SENS:DLOG:FUNC:CURR ON,(@%d)',
        channel_num));
    % ACTIVATE VOLTAGE DATALOGGER
    writeline(DEV, sprintf('SENS:DLOG:FUNC:VOLT ON,(@%d)',
        channel_num));
    % SELECT 3.06A CURRENT RANGE
    writeline(DEV, sprintf('SENS:DLOG:CURR:RANG MAX,(@%d)',
        channel_num));

    %%  N6783A-BAT SETTINGS

    %SET VOLTAGE PRIORITY
     writeline(DEV, sprintf('SOUR:FUNC VOLT,(@%d)', channel_num));
    % SET THE DESIRED VOLTAGE
    writeline(DEV, sprintf(':SOUR:VOLT %f, (@%d)', v_target,
        channel_num));
    % SET THE MAXIMUM POSITIVE CURRENT
    writeline(DEV, sprintf(':SOUR:CURR:LIM %f, (@%d)',
        abs(c_target), channel_num));
    % SET THE MAXIMUM NEGATIVE CURRENT
    writeline(DEV, sprintf(':SOUR:CURR:LIM:NEG %f, (@%d)',
        -abs(c_target), channel_num));

end
```

```
%% TURN ON DLOGGER (and start tic)
writeline(DEV, sprintf(':TRIG:DLOG:SOUR IMM'));

if dlog_flag

    d = string(datetime);
    d = strrep(d, ":", "-");
    d = strrep(d, " ", "-");

    writeline(DEV, sprintf('INIT:DLOG "internal:\\charge-%s.dlog"',
        d));
else
    % Overwrites the default file
    writeline(DEV, 'INIT:DLOG "internal:\\default.dlog"');
end

tic

%% TURN ON N6783A-BAT
% ACTIVATE N6783A-BAT OUTPUT
writeline(DEV, sprintf(':OUTP:STAT ON, %s', channel_string));
disp("CHARGE STARTED " + strjoin(string(channel), ','))

%% CHECKING MINIMUM CURRENT ALGORITHM
pause(10 * Tc)

channel_active = true(1, N);

try
    % Continue while any channel remains active (charge/discharge)
    while any(channel_active)
        t1 = toc;

        % Set markers once per cycle (common to all channels)
        writeline(DEV, sprintf('SENS:DLOG:MARK2:POIN %.1f', t1 - Tc));
        writeline(DEV, sprintf('SENS:DLOG:MARK1:POIN %.1f', t1 - 5 *
            Tc));

        for i = 1:N
            % If channel is active, measure current and check cutoff
                current
            if channel_active(i)
                channel_num = channel(i);

                % Measure current and save to current_now
                writeline(DEV, sprintf('FETC:DLOG:CURR? (@%d)',
                    channel_num));
                current_now = str2num(readline(DEV));

                % Measure voltage and save to voltage_now
```

```
                writeline(DEV, sprintf('FETC:DLOG:VOLT? (@%d)',
                    channel_num));
                voltage_now = str2double(readline(DEV));

                fprintf('[Tempo: %6.1f s] CH %d -> Tensione: %5.3f V
                    | Corrente: %7.4f A\n', ...
                        t1, channel_num, voltage_now, current_now);
                % Cutoff current check
                if abs(current_now) <= abs(cutoff_current)
                   % Set channel_active flag to false
                   channel_active(i) = false;
                   % Turn off channel i
                   writeline(DEV, sprintf(':OUTP:STAT OFF, (@%d)',
                       channel_num));
                   fprintf('CHARGE ENDED ON CHANNEL %d\n',
                       channel_num);
                end
            end
        end
        pause(10 * Tc);
    end

    disp("ALL CHARGES ENDED SUCCESSFULLY.");

    catch ME
    writeline(DEV, sprintf(':OUTP:STAT OFF, (@%s)', channel_string));
    fprintf('Errore alla riga %d: %s\n', ME.stack(1).line,
        ME.message);
end

    %% DLOG SHUTDOWN
    writeline(DEV, sprintf(':ABOR:DLOG'));

end
```

A.1.3. Codice chargeN6781A_ageing

Listing A.3: Codice della funzione chargeN6781A_ageing. Licenza GNU GPL v3

```
function chargeN6781A_ageing(DEV, B, OVP, channel, cutoff_curr, Tc)

%function chargeN6781A_ageing(DEV, B, OVP, channel, cutoff_curr, Tc)
%
% Charge/discharge routine via External Data Logger (Elog);
% measurements are retrieved in real-time without local storage on
    the instrument.
%
%% Inputs
% DEV          -> KEYSIGHT Device
% OVP          -> Set the value of the OVER VOLTAGE PROTECTION
%                 [4.5 V Default]
% B            -> Battery parameters
```

```
% B.cap        -> mean capacity of the battery  [Ah]
% B.curr_C     -> The current at which operate in C unit
% B.v_max      -> desired battery voltage [V]
%                 for backward compatibility the name of the field is
    not changed
% channel      -> Channel index [Channel 1 Default]
% cutoff_curr -> low current limit
% Tc           -> Integration period of External Data Logger

% Copyright (C) 2026 S. Paparelli
%This program is free software: you can redistribute it and/or modify
%it under the terms of the GNU General Public License as published by
%the Free Software Foundation, either version 3 of the License, or
%(at your option) any later version.
%This program is distributed in the hope that it will be useful,
%but WITHOUT ANY WARRANTY; without even the implied warranty of
%MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%GNU General Public License for more details.
%You should have received a copy of the GNU General Public License
%along with this program.  If not, see <http://www.gnu.org/licenses/>.

if(~exist('DEV','var'))
  error('Unspecified device', 'Error');
end
if(~exist('OVP', 'var'))
    OVP=4.7;
end
if(~exist('channel','var'))
    channel=1;
end
if(~exist('cutoff_curr','var'))
    cutoff_curr = 0.075;
end

if(~exist('Tc','var'))
 Tc = 0.3;
end

curr = abs(B.curr_C)*B.cap;

%% ELOGGER SETTINGS
% TURN OFF ELOGGER FOR ALL CHANNELS
writeline(DEV, sprintf('SENS:ELOG:FUNC:CURR OFF, (@1,2,3,4)'));
writeline(DEV, sprintf('SENS:ELOG:FUNC:VOLT OFF, (@1,2,3,4)'));
% SET RESOLUTION
writeline(DEV, 'SENS:SWE:TINT:RES RES20');
% SELECT CURRENT RANGE
writeline(DEV, sprintf('SENS:ELOG:CURR:RANG MAX, (@%d)',channel));
% ACTIVATE CURRENT ELOGGER
writeline(DEV, sprintf('SENS:ELOG:FUNC:CURR ON, (@%d)',channel));
% ACTIVATE VOLTAGE ELOGGER
```

```
writeline(DEV, sprintf('SENS:ELOG:FUNC:VOLT ON, (@%d)',channel));
% SET THE ELOGGER INTEGRATION PERIOD
writeline(DEV, sprintf('SENS:ELOG:PER %f, (@%d)',Tc, channel));
% SET BINARY
writeline(DEV, 'FORM:DATA REAL');
% SET BYTE ORDER
writeline(DEV, 'FORM:BORD SWAP');
%% N6781A SETTINGS
% Set 2 quadrant power supply mode
writeline(DEV,sprintf(':SOUR:EMUL PS2Q, (@%d)',channel));
% Set operating in voltage priority
writeline(DEV,sprintf(':SOUR:FUNC VOLT, (@%d)',channel));
% Set target voltage
writeline(DEV,sprintf(':SOUR:VOLT %f, (@%d)', B.v_max, channel));
% Set maximum positive and negative current limit
writeline(DEV,sprintf(':SOUR:CURR:LIM %f, (@%d)', abs(curr),
    channel));
% Set the protection voltage
writeline(DEV,sprintf('VOLT:PROT:REM %f, (@%d)', OVP, channel));


%% TURN ON ELOGGER
% SELECT THE ELOG TRIGGER SOURCE
writeline(DEV, sprintf(':TRIG:ELOG:SOUR IMM, (@%d)', channel));
% INITATE ELOG
writeline(DEV, sprintf('INIT:ELOG (@%d)', channel));

%% N6781A ON
writeline(DEV,sprintf(':OUTP:STAT ON, (@%d)',channel));
disp ('Charge started')
%%  CHARGING ALGORITHM
% Time for two records
pause(2*Tc)
current_now = cutoff_curr + 1;
while (abs(current_now) > cutoff_curr)
    % Asking for ten record
    writeline(DEV, sprintf('FETC:ELOG? 10, (@%d)', channel));
    % Reading binary block (4 byte for single value)
    data = readbinblock(DEV, 'single');
    % Flush communication channel
    read(DEV, 1, 'uint8');
    % Verify data vector and save to current_now and voltage_now
    if ~isempty(data)
    current_now = data(1);
    fprintf('Current: %.4f A\n', current_now);
    voltage_now = data(2);
    fprintf('Voltage: %.4f V\n', voltage_now);
    end
    % Waiting for one record
    pause(Tc);
end
```

```
% Set outout channel off
writeline(DEV,sprintf(':OUTP:STAT OFF, (@%d)', channel));
disp ('Charge completed')

%% TURN OFF ELOG
writeline(DEV, sprintf(':ABOR:ELOG (@%d)', channel));
end
```

A.1.4. Codice chargeN6783A_ageing

Listing A.4: Codice della funzione chargeN6783A_ageing. Licenza GNU GPL v3

```
function chargeN6783A_ageing(DEV, charge_v, charge_c, channel, ...
    cutoff_current, Tc)

%function chargeN6783A_ageing(DEV, charge_v, charge_c, channel, ...
%      cutoff_current, Tc)
%
% Charge/discharge routine via External Data Logger (Elog);
% measurements are retrieved in real-time without local storage on
   the instrument
%
%% Inputs
% DEV            -> TCP client ID
% charge_v       -> desired battery voltages [V]
% charge_c       -> current compliances (positive) [A]
% cutoff_current -> charge stop current limit
% Tc             -> Integration period of External Data Logger
% channel        -> vector of channels (only 3 and 4)

% Copyright (C) 2026 S. Paparelli
%This program is free software: you can redistribute it and/or modify
%it under the terms of the GNU General Public License as published by
%the Free Software Foundation, either version 3 of the License, or
%(at your option) any later version.
%This program is distributed in the hope that it will be useful,
%but WITHOUT ANY WARRANTY; without even the implied warranty of
%MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%GNU General Public License for more details.
%You should have received a copy of the GNU General Public License
%along with this program.  If not, see <http://www.gnu.org/licenses/>.

if(~exist('cutoff_current','var'))
    cutoff_current = 0.075;
end

if(~exist('Tc','var'))
    Tc=0.3;
end

% TURN OFF ELOGGER FOR ALL CHANNELS
```

```
writeline(DEV, 'SENS:ELOG:FUNC:CURR OFF, (@1,2,3,4)');
writeline(DEV, 'SENS:ELOG:FUNC:VOLT OFF, (@1,2,3,4)');
% SET BINARY
writeline(DEV, 'FORM:DATA REAL');
% SET BYTE ORDER
writeline(DEV, 'FORM:BORD SWAP');

channel_string = sprintf('(@%s)', join(string(channel), ','));

N = length(channel);
for i = 1:N
%% ELOGGER SETTINGS
channel_num = channel(i);
% min(i, end) ensures that if the loop index 'i' exceeds the array
    length,
% it remains at the last available value (clamping the index).
v_target = charge_v(min(i,end));
c_target = abs(charge_c(min(i,end)));
% SET RESOLUTION
writeline(DEV, 'SENS:SWE:TINT:RES RES20');
% ACTIVATE CURRENT ELOGGER
writeline(DEV, sprintf('SENS:ELOG:FUNC:CURR ON, (@%d)', channel_num));
% SELECT CURRENT RANGE
writeline(DEV, sprintf('SENS:ELOG:CURR:RANG MAX, (@%d)',
    channel_num));
% SET THE ELOGGER INTEGRATION PERIOD
writeline(DEV, sprintf('SENS:ELOG:PER %f, (@%d)', Tc, channel_num));

%%  N6783A-BAT SETTINGS

%SET VOLTAGE PRIORITY
writeline(DEV, sprintf('SOUR:FUNC VOLT, (@%d)', channel_num));

% SET THE DESIRED VOLTAGE
writeline(DEV, sprintf(':SOUR:VOLT %f, (@%d)', v_target,
    channel_num));

% SET THE MAXIMUM POSITIVE CURRENT
writeline(DEV, sprintf(':SOUR:CURR:LIM %f, (@%d)', c_target,
    channel_num));

% SET THE MAXIMUM NEGATIVE CURRENT
writeline(DEV, sprintf(':SOUR:CURR:LIM:NEG %f, (@%d)',
    -abs(c_target), channel_num));

% SELECT THE ELOG TRIGGER SOURCE
writeline(DEV, sprintf(':TRIG:ELOG:SOUR IMM, (@%d)', channel_num));
end
%%  TURN ON ELOGGER
% INITATE ELOG
writeline(DEV, sprintf('INIT:ELOG %s', channel_string));
```

```
%%  TURN ON N6783A-BAT
% ACTIVATE N6783A-BAT OUTPUT
writeline(DEV,sprintf(':OUTP:STAT ON, %s', channel_string));
disp("CHARGE STARTED")

%%  CHECKING MINIMUM CURRENT ALGORITHM
% Time for two records
pause(2*Tc)
channel_active = true(1,N);
try
% Continue while any channel remains active (charge/discharge)
while any(channel_active)
    for i = 1:N
        % If channel is active, measure current and check cutoff
            current
        if channel_active(i) %
            channel_num = channel(i);
            % Asking for ten record
            writeline(DEV, sprintf('FETC:ELOG? 10, (@%d)',
                channel_num));
            % Reading binary block (4 byte for single value)
            data = readbinblock(DEV, 'single');
            fprintf('channel %d\n data length: %d\t', channel_num,
                length(data));
            % Flush communication channel
            read(DEV, 1, 'uint8');
            % Verify data vector and save to current_now
            if ~isempty(data)
                current_now = data(1);
                fprintf('Current: %.4f A\n', current_now);
            end
            % Cutoff current check
            if abs(current_now) <= abs(cutoff_current)
                % Set channel_active flag to false
                channel_active(i) = false;
                % Turn off channel i
                writeline(DEV,sprintf(':OUTP:STAT OFF,
                    (@%d)',channel_num));
                % Turn off elogger of channel i
                writeline(DEV, sprintf(':ABOR:ELOG (@%d)',
                    channel_num));
                fprintf('CHARGE ENDED ON CHANNEL %d\n', channel_num);
            end
        end
    end

    % Waiting for one record
    pause(Tc);
end
catch ME
```

```
    writeline(DEV, sprintf(':OUTP:STAT OFF, %s', channel_string));
    writeline(DEV, sprintf(':ABOR:ELOG %s', channel_string));
end
disp("ALL CHARGES ENDED");

end
```

A.2. Codici delle funzioni per l'EIS

A.2.1. Codice charge_soc

Listing A.5: Codice della funzione charge_soc. Licenza GNU GPL v3

```
function charge_soc(DEV, soc_target, B, channel)

% function charge_soc(DEV, soc_target, B, channel)
%
% Brings the battery State of Charge to the desired target level
%% Inputs
% DEV         -> KEYSIGHT Device
% soc_target -> Desired state of charge (SoC)
% B           -> Battery parameters
% channel     -> Channel index (Only 1 or 2)

% Copyright (C) 2026 S. Paparelli
%This program is free software: you can redistribute it and/or modify
%it under the terms of the GNU General Public License as published by
%the Free Software Foundation, either version 3 of the License, or
%(at your option) any later version.
%This program is distributed in the hope that it will be useful,
%but WITHOUT ANY WARRANTY; without even the implied warranty of
%MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%GNU General Public License for more details.
%You should have received a copy of the GNU General Public License
%along with this program.  If not, see <http://www.gnu.org/licenses/>.

% Charging parameters
B.v_max = 4.2;
cutoff_curr = 75e-3;
Tc = 0.3;
OVP = 4.7;
% Constant Current (CC) value calculation
currA = abs(B.curr_C)*B.cap;
% Full battery charge
chargeN6781A_ageing(DEV, B , OVP, channel, cutoff_curr, Tc)
% Charge to remove to reach target SoC
Q_remove = B.cap * (1-soc_target);
% Time required to remove charge (seconds)
T_discharge = (Q_remove / abs(currA))*3600;

%% N6781A Module Operation in Arbitrary (ARB) Mode

% Set instrument to 2-quadrant mode
writeline(DEV, sprintf(':SOUR:EMUL PS2Q, (@%d)', channel));
% Set instrument to "current priority mode"
writeline(DEV, sprintf('FUNC CURR, (@%d)', channel));
% Set ARB priority mode to current
writeline(DEV, sprintf('ARB:FUNC:TYPE CURR,(@%d)', channel));
% Specify the type of ARB --> STEP
```

```
writeline(DEV, sprintf('ARB:FUNC:SHAP STEP,(@%d)', channel));
% Current before step, set to 0
writeline(DEV, sprintf('ARB:CURR:STEP:STAR:LEV 0, (@%d)', channel));
% Set step current value
writeline(DEV, sprintf('ARB:CURR:STEP:END:LEV %f, (@%d)',
    -abs(currA), channel));
% Initial delay set to zero (disabled)
writeline(DEV, sprintf('ARB:CURR:STEP:STAR:TIM 0, (@%d)', channel));
% Set step duration to 1 second
writeline(DEV, sprintf('ARB:CURR:STEP:END:TIM 1, (@%d)', channel));
% Hold the last ARB point value (currA)
writeline(DEV, sprintf('ARB:TERM:LAST ON, (@%d)', channel));
% Execute once only
writeline(DEV, sprintf('ARB:COUNT 1, (@%d)', channel));
% Set the current control mode to ARB
writeline(DEV, sprintf('CURR:MODE ARB, (@%d)', channel));
% Set immediate trigger
writeline(DEV, 'TRIG:ARB:SOUR IMM');
% Channel output ON
writeline(DEV, sprintf('OUTP ON, (@%d)', channel));
% Prepare ARB trigger
writeline(DEV, sprintf('INIT:TRAN (@%d)', channel));
% ARB ON
writeline(DEV, sprintf('TRIG:TRAN (@%d)', channel));

d = duration(0, 0, T_discharge, 'Format', 'mm:ss');
fprintf('Discharging for %s\n', char(d));

% Wait for T_discharge to maintain the current step
pause(T_discharge);
% Current set to 0
writeline(DEV, sprintf('CURR 0, (@%d)', channel));
% Channel output OFF
writeline(DEV, sprintf('OUTP OFF, (@%d)', channel));

end
```

A.2.2. Codice SinGen_multi

Listing A.6: Codice della funzione SinGen_multi. Licenza GNU GPL v3

```
function freqs = SinGen_multi(p)
%
%function [str_vals, vals, fquant] = SinGen_multi(p)
%
% Generates a structure containing a sinusoid for each frequency.
%
% INPUT (p structure):
%   p.fmin    - minimum frequency [Hz]
%   p.fmax    - maximum frequency [Hz]
%   p.Nf      - number of frequencies
%   p.Np      - number of points per sinusoid
```

```
%   p.amp     - amplitude [A]
%   p.Ts      - sampling period [s]
%
% OUTPUT:
%   freqs     - structure array (1 x Nf) with fields:
%     .f_nom     - nominal frequency [Hz]
%     .f_quant   - actual quantized frequency [Hz]
%     .Ts        - sampling period [s]
%     .mult      - Ts_min multiplier
%     .Np        - number of points
%     .cycle     - actual integer cycles
%     .spc       - samples per cycle
%     .vals      - sine wave sample vector

% Copyright (C) 2026 S.Paparelli
%
%This program is free software: you can redistribute it and/or modify
%it under the terms of the GNU General Public License as published by
%the Free Software Foundation, either version 3 of the License, or
%(at your option) any later version.
%This program is distributed in the hope that it will be useful,
%but WITHOUT ANY WARRANTY; without even the implied warranty of
%MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%GNU General Public License for more details.
%You should have received a copy of the GNU General Public License
%along with this program.  If not, see <http://www.gnu.org/licenses/>.

% Minimum period
Ts_min = p.Ts;
% Frequencies array
fi = logspace(log10(p.fmin), log10(p.fmax), p.Nf);

for j = 1:p.Nf
    f = fi(j);

    % Minimum multiple to have at least 1 full cycle
    mult = max(1, ceil(1 / (p.Np * f * Ts_min)));

    Ts_j = mult * Ts_min;

    % Actual integer cycles
    cycle_j = floor(p.Np * Ts_j * f);

    % Ensures at least 1 cycle
    if cycle_j < 1
        mult    = mult + 1;
        Ts_j    = mult * Ts_min;
        cycle_j = floor(p.Np * Ts_j * f);
    end

    % Quantized frequency
```

```
    f_quant_j = cycle_j / (p.Np * Ts_j);

    % Sample per cycle
    spc_j = p.Np / cycle_j;

    % Generates sine wave samples
    t = (0:p.Np-1) * Ts_j;
    vals_j = p.amp * sin(2 * pi * f_quant_j * t);

    % Populates the structure
    freqs(j).f_nom    = f;
    freqs(j).f_quant  = f_quant_j;
    freqs(j).Ts       = Ts_j;
    freqs(j).mult     = mult;
    freqs(j).Np       = p.Np;
    freqs(j).cycle    = cycle_j;
    freqs(j).spc      = spc_j;
    freqs(j).vals     = vals_j;

    fprintf('%10.3f | %10.3f | %8.2f | %6d | %6d | %11.1f\n', ...
        f, f_quant_j, Ts_j*1e6, mult, cycle_j, spc_j);
end

fprintf('\nTotale frequenze: %d | Np per frequenza: %d\n', p.Nf,
    p.Np);
fprintf('Durata per frequenza: min=%.3f s  max=%.3f s\n', ...
    min([freqs.Np] .* [freqs.Ts]), ...
    max([freqs.Np] .* [freqs.Ts]));
end
```

A.2.3. Codice measure

Listing A.7: Codice della funzione measure. Licenza GNU GPL v3

```
function [V_meas, I_meas, freqs] = measure(DEV, p, channel, path_out,
    soc, B, save_file)

%function [data_matrix, freq_idx, freqs] = measure(DEV, p, channel)
%
% Acquires voltage and current for each generated frequency.
%
% For each frequency defined in the parameters, this function loads
    the ARB
% sequence to the instrument, measures using the external data logger,
% and saves the synchronized voltage and current arrays into
    dedicated matrices.
% Optionally, it exports the data and parameters to a .mat file for
    future analysis.
%
% INPUT:
%   DEV       -> KEYSIGHT Instrument object (connection interface)
%   p         -> Parameters structure (must contain Np, Idtype, etc.)
%   channel   -> Instrument channel index (e.g., 1)
%   path_out  -> Destination path directory for the saved output file
%   soc       -> State of Charge (SOC) of the battery [0, 1]
%   B         -> Battery parameters structure (must contain B.Id)
%   save_file -> Boolean flag: if true, saves V_meas, I_meas, and
    structures into a .mat file
%
% OUTPUT:
%   V_meas    -> Matrix of Voltage measurements (each column
    represents a tested frequency)
%   I_meas    -> Matrix of Current measurements (each column
    represents a tested frequency)
%   freqs     -> Structure array (1 x Nf) containing frequency
    details and ARB waveforms

% Copyright (C) 2026 S.Paparelli
% This program is free software : you can redistribute it and / or
    modify
% it under the terms of the GNU General Public License as published by
% the Free Software Foundation , either version 3 of the License , or
% ( at your option ) any later version .
% This program is distributed in the hope that it will be useful ,
% but WITHOUT ANY WARRANTY ; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE . See the
% GNU General Public License for more details .
% You should have received a copy of the GNU General Public License
% along with this program . If not , see < http :// www . gnu . org /
    licenses / >.

% Generate sinusoids
```

```
freqs = SinGen_multi(p);

Nf = length(freqs);

for j = 1:Nf
    f     = freqs(j).f_quant;
    Ts_j  = freqs(j).Ts;
    Np_j  = freqs(j).Np;
    vals  = freqs(j).vals;



    durata_j = Np_j * Ts_j;

    %% N6781A SETTINGS
    % set 2-quadrant mode
    writeline(DEV, sprintf(':SOUR:EMUL PS2Q, (@%d)',channel));
    % set current priority mode
    writeline(DEV, sprintf('FUNC CURR, (@%d)', channel));

    %% Setup and load ARB
    writeline(DEV, sprintf('ARB:FUNC:TYPE CURR, (@%d)', channel));
    writeline(DEV, sprintf('ARB:FUNC:SHAP CDW, (@%d)', channel));
    writeline(DEV, sprintf('ARB:CURR:CDW:DWEL %f, (@%d)', Ts_j,
        channel));
    writeline(DEV, sprintf('ARB:TERM:LAST OFF, (@%d)', channel));
    writeline(DEV, sprintf('CURR:MODE ARB, (@%d)', channel));
    writeline(DEV, 'TRIG:ARB:SOUR IMM');

    %% Setup and load ARB

        % Set data as float32
        mywriteline(DEV, "FORM:DATA REAL");
        % Set byte order to little-endian (used by MATLAB)
        mywriteline(DEV, "FORM:BORD SWAP");
        data = single(vals);
        dataBytes = typecast(data, 'uint8');
        % Builds the SCPI prefix for binary block
        len = length(dataBytes);
        lenStr = num2str(len);
        N = num2str(length(lenStr));
        prefix = ['#' N lenStr];
        channel_cmd = sprintf(', (@%d)', channel);
        % Builds the complete SCPI command
        cmd = ['SOUR:ARB:CURR:CDW:LEV ', prefix, dataBytes,
            channel_cmd];
        % Send the command + binary data
        write(DEV, uint8(cmd), "uint8");

    pause(1);
```

```
%% Configure Elog
 % Turn off elogger for all channels
 writeline(DEV, sprintf('SENS:ELOG:FUNC:CURR OFF, (@1,2,3,4)'));
 writeline(DEV, sprintf('SENS:ELOG:FUNC:VOLT OFF, (@1,2,3,4)'));
 % Set resolution
 writeline(DEV, 'SENS:SWE:TINT:RES RES20');
 % Set range voltage and current measure
 writeline(DEV, sprintf('SENS:ELOG:VOLT:RANG MAX , (@%d)',
     channel));
 writeline(DEV, sprintf('SENS:ELOG:CURR:RANG MAX ,
     (@%d)',channel));
 % Activate voltage elogger
 writeline(DEV, sprintf('SENS:ELOG:FUNC:VOLT ON, (@%d)', channel));
 % Activate current elogger
 writeline(DEV, sprintf('SENS:ELOG:FUNC:CURR ON, (@%d)', channel));
 % Set elogger integration period
 writeline(DEV, sprintf('SENS:ELOG:PER %.8f, (@%d)', Ts_j,
     channel));
 % Set elogger immediate trigger
 writeline(DEV, sprintf('TRIG:ELOG:SOUR IMM, (@%d)', channel));

 %% CHANNEL OUTPUT ON
 writeline(DEV, sprintf('OUTP ON, (@%d)', channel));

 %% ELOG START
 writeline(DEV, sprintf('INIT:ELOG (@%d)', channel));
 pause(0.5);

 %% ARB START
 writeline (DEV, sprintf('INIT:TRAN (@%d)',channel));
 writeline (DEV, sprintf('TRIG:TRAN (@%d)', channel));
 % wait for ARB
 pause(durata_j + 1);

 %% Single FETCh at the end

 % The Elog buffer contains all ARB samples

 [V_raw, I_raw] = fetch(DEV, Np_j + 10000 , channel);

 startIdx = findDelay_1(vals, I_raw);
 endIdx = startIdx + Np_j - 1;

 V_meas(:,j) = V_raw(startIdx : endIdx);
 I_meas(:,j) = I_raw(startIdx : endIdx);

 %% Stop Elog and exit
 writeline(DEV, sprintf('ABOR:ELOG (@%d)', channel));
 writeline(DEV, sprintf('OUTP OFF, (@%d)', channel));
```

```
    pause(0.3);

end

if save_file

   %% Save data to .mat file
    pre_out = strcat(string(datetime('today')),"-soc-",num2str(soc),
        string(datetime('now', 'Format', '-HH_mm_ss')),"-Np-",
        num2str(p.Np));
    filename = sprintf('%c-B-%d-%s', p.Idtype, B.Id, pre_out);
    mat_filename = sprintf('%s/EIS-%s.mat', path_out, filename);

    % Save essential variables and data structures for post-processing
    save(mat_filename,'V_meas', 'I_meas', 'freqs', 'p', 'soc');

   fprintf('Data and parameters successfully saved in: %s\n',
      mat_filename);

end

end
```

A.2.4. Codice plot_eis

Listing A.8: Codice della funzione plot_eis. Licenza GNU GPL v3

```
function  plot_eis(p, soc, B, path_out, V_meas, I_meas, freqs)

%function plot_eis(p, soc, B, path_out, V_meas, I_meas, freqs)
%
% Calculates the impedance via FFT and plots the Nyquist diagram.
%
%   This function processes time-domain voltage (V_meas) and current
%   (I_meas) experimental data to calculate the complex impedance Z(f)
%   using the Fast Fourier Transform (FFT). Subsequently, it generates
%   a Nyquist plot (complex plane: real part vs. negative imaginary
    part)
%   and automatically exports it as a PDF file.
%%   Input
%   p         -> Test parameters structure (must contain p.amp and
    p.Idtype)
%   soc       -> State of Charge (SOC) of the battery, value between
    [0, 1]
%   B         -> Battery parameters structure (must contain B.Id)
%   path_out  -> Destination path to save the exported PDF plot
%   V_meas    -> Voltage measurements matrix (each column is a tested
    frequency)
%   I_meas    -> Current measurements matrix (each column is a tested
    frequency)
%   freqs     -> Structure containing details of the tested
    frequencies (e.g., f_quant)
```

```
%%   Output:
%   - Export of a formatted .pdf file containing the Nyquist diagram.

% Copyright (C) 2026 S.Paparelli
% This program is free software : you can redistribute it and / or
    modify
% it under the terms of the GNU General Public License as published by
% the Free Software Foundation , either version 3 of the License , or
% ( at your option ) any later version .
% This program is distributed in the hope that it will be useful ,
% but WITHOUT ANY WARRANTY ; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE . See the
% GNU General Public License for more details .
% You should have received a copy of the GNU General Public License
% along with this program . If not , see < http :// www . gnu . org /
    licenses / >.



markers = {'o'};
colors_ = get(gca,'colororder');
plts1 = [];
plts2 = [];

Nf = length(freqs);

%% Calculate Z impedance for each frequency
Z      = nan(1, Nf);
fquant = nan(1, Nf);

for j = 1:Nf
    % Extract the time-domain voltage and current samples for the
        current frequency
    V = V_meas(:, j);
    I = I_meas(:, j);
    % Compute the Fast Fourier Transform (FFT)
    Vfft = fft(V);
    Ifft = fft(I);
    % Identify the index of the dominant frequency component in the
        current signal
    [~, fidx] = max(abs(Ifft(1:floor(end/2))));
    % Calculate the complex impedance Z at the excitation frequency
    % Z(f) is derived as the ratio of the phasors of voltage and
        current
    Z(j)      = Vfft(fidx) / Ifft(fidx);
    fquant(j) = freqs(j).f_quant;


end
```

```
%% Plot
fig = figure('Color', 'white');
hold on;
FontSize = 16;

color_ = [0.6 0.1 0.1];


plt1 = plot(real(Z)*1000, -imag(Z)*1000, ...
    'LineWidth', 2, 'Color', color_);

% Markers for each frequency
plts2 = [];
for j = 1:length(fquant)
    marker_idx = mod(j-1, length(markers)) + 1;
    plt2_temp  = plot(real(Z (j))*1000, -imag(Z(j))*1000, ...
        markers{marker_idx}, ...
        'MarkerSize', FontSize - 5, ...
        'Color', color_);
    plts2 = [plts2, plt2_temp];
end

%% Axis format
ax = gca;
ax.FontSize             = FontSize;
ax.TickLabelInterpreter = 'latex';
xlabel('$\Re(Z(f))$ [m$\Omega$]', 'Interpreter', 'latex', 'FontSize',
    FontSize);
ylabel('$-\Im(Z(f))$ [m$\Omega$]', 'Interpreter', 'latex',
    'FontSize', FontSize);
t = sprintf("EIS cella %d, @%.1fA",B.Id,p.amp);
title(t,'Interpreter','latex');
grid on;

lgd = legend(plt1, sprintf('SOC=%.1f', soc), ...
    'Location', 'southeast', 'Interpreter', 'latex');
lgd.FontSize = 22;
%% Freqeuncies mini legend
lgd_str2 = arrayfun(@(f) sprintf('%.4f Hz', f), fquant, ...
    'UniformOutput', false);
lgd_str3 = markers(mod((1:length(fquant))-1, length(markers))+1);
x_lim = xlim;
xlim([x_lim(1)-2, x_lim(2)]);
%miniLegend_1(lgd_str3, lgd_str2, 'k', [0.15, 0.55, 0.20, 0.35],
    FontSize);
miniLegend_2(freqs, [0.15, 0.55, 0.20, 0.35], FontSize+4);

%% PDF Export
pre_out = strcat(string(datetime('today')),"-soc-",num2str(soc),
    string(datetime('now', 'Format', '-HH_mm_ss')));
```

```
filename = sprintf('%c-B-%d-%s', p.Idtype, B.Id, pre_out);
exportapp(fig, sprintf('%s/EIS-%s.pdf', path_out, filename));

end
```

Bibliografia

[1] The Royal Swedish Academy of Sciences. They created a rechargeable world, 2019. Accessed: 2026-06-09.

[2] Da Deng. Li-ion batteries: basics, progress, and challenges. *Energy Science & Engineering*, 3(5):385–418, 2015.

[3] Phuoc-Anh Le. A general introduction to lithium-ion batteries: From the first concept to the top six commercials and beyond. *J. Electrochem. Sci. Eng.*, 13(4):591–604, 2023.

[4] Sanyo Electric Co., Ltd. *Lithium-Ion Rechargeable Battery Model NCR18650BL Datasheet*. Sanyo Electric Co., Ltd., 2015. Specifiche tecniche del produttore.

[5] Isidor Buchmann. BU-402: What Is C-rate. `https://www.batteryuniversity.com/article/bu-402-what-is-c-rate/`, 2021. Battery University (Cadex Electronics), Accesso: 18-06-2026.

[6] Isidor Buchmann. BU-409: Charging Lithium-ion. `https://www.batteryuniversity.com/article/bu-409-charging-lithium-ion/`, 2021. Battery University (Cadex Electronics), Accesso: 18-06-2026.

[7] Isidor Buchmann. BU-903: How to Measure State-of-charge. `https://www.batteryuniversity.com/article/bu-903-how-to-measure-state-of-charge/`, 2021. Battery University (Cadex Electronics), Accesso: 18-06-2026.

[8] Isidor Buchmann. BU-915: Testing Battery with EIS. `https://www.batteryuniversity.com/article/bu-915-testing-battery-with-eis/`, 2024. Battery University (Cadex Electronics), Accesso: 19-06-2026.

[9] Keysight Technologies. *N6700 Series Low-Profile Modular Power System - Operating and Service Guide*. Keysight Technologies, 2020. Models N6700C, N6701C, N6702C and power modules.

[10] Keysight Technologies. *N6705C DC Power Analyzer - User's Guide*. Keysight Technologies, 2020.