UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Elettronica

Stima dello stato di carica di batterie per moto elettriche ad alte prestazioni

Estimation of the state of charge of batteries for high-performance electric motorcycles

Relatore:

Chiar.mo
Prof. Massimo Conti

Correlatore:

Rino Valente

Laureando:

Michele Grilli

Anno Accademico 2022/2023

Sommario

Introduzione

Nel corso degli ultimi anni la mobilità ha subito una decisa rivoluzione con l'introduzione di veicoli mossi da motori elettrici.

Nonostante la messa in produzione ormai assodata di diversi veicoli nel mercato, rimangono ancora inesplorate molte soluzioni riguardanti la stima dei parametri dei sistemi utilizzati.

Con l'emergere del riscaldamento globale e di varie condizioni meteorologiche estreme, il problema del cambiamento climatico e la carenza di approvvigionamento energetico hanno attirato ancora una volta l'attenzione diffusa nella società [1]. I veicoli elettrici hanno i vantaggi di bassa rumorosità, basso inquinamento, basso costo e facile manutenzione, che sono gradualmente diventati necessità per gli spostamenti quotidiani delle persone [2, 3]. La batteria agli ioni di litio è stata ampiamente applicata nei veicoli elettrici grazie alla lunga durata, all'alta densità di energia, all'alta tensione nominale e al basso tasso di autoscarica [4, 5].

Sono state svolte numerose ricerche mirate a comprendere e a stimare efficacemente il comportamento delle batterie utilizzate come forma di stoccaggio energetico e in molte di esse ha ricoperto un ruolo di spicco l'utilizzo del filtro di Kalman come strumento per quantizzare lo stato di carica (SOC) e per lo stato di salute (SOH) delle celle.

In questo lavoro verranno esaminate le peculiarità e le difficoltà per la progettazione di un sistema di stima del SOC per batterie utilizzate in moto elettriche ad alte prestazioni.

Dopo una prima parte sullo stato dell'arte riguardanti i sistemi di stima sviluppati fino ad oggi e sulle problematiche collegate al compito di collegare la misura dei parametri di una batteria a delle misure fisiche, verrà introdotto nel capitolo 2 il funzionamento della batteria agli ioni di litio e saranno spiegate le principali terminologie adoperate.

Nel capitolo 3 sarà effettuata una panoramica dei principali modelli matematici per le batterie e sarà trattato un approfondimento sul modello scelto per questo lavoro.

Il capitolo 4 è dedicato ai sistemi di stima utilizzati in questa tesi, con una particolare attenzione al filtro di Kalman, metodo scelto per questo lavoro.

Nel capitolo 5 sono descritti gli strumenti adoperati per creare il modello della batteria e per eseguire il fitting dei parametri. Saranno mostrati i vari schemi adoperati per le simulazioni con una relativa descrizione generale.

Il capitolo 6 parla delle simulazioni eseguite con il modello creato in Simulink e delle stime dei parametri effettuate.

Il capitolo 7 mostra una panoramica sul processo di esportazione del codice in linguaggio C per l'inserimento nel microcontrollore.

La parte relativa ai test eseguiti e ai commenti dei risultati ottenuti dal filtro di Kalman nel microcontrollore è contenuta nel capitolo 8; sono riportati i grafici delle simulazioni e sono descritti i test effettuati, oltre ad una parte relativa ad una successiva ottimizzazione.

Le conclusioni di questo lavoro sono inserite nel capitolo 9.

1 Stato dell'Arte

La stima dello stato di carica (SOC) per le batterie agli ioni di litio comporta l'utilizzo di tecniche avanzate che sfruttano modelli matematici, apprendimento automatico e analisi dei dati in tempo reale. Spesso molte queste tecniche di stima vengono combinate assieme per carcare di ottenere risultati sempre più accurati.

Le batterie agli ioni di litio (Li-ion) sono oggi la soluzione di accumulo di energia più promettente per i veicoli elettrici (EV) e per molti altri sistemi grazie alle loro prestazioni superiori, come l'alta densità di energia, la capacità di ricarica rapida, la lunga durata ciclica, l'ampio intervallo di temperature operative ecc. [6]. Come fonte di alimentazione unica nei veicoli elettrici, le batterie agli ioni di litio necessitano di un monitoraggio efficiente da parte di un sistema di gestione della batteria (BMS), per garantire uno sfruttamento sicuro e ottimale dell'energia immagazzinata [7]. Un parametro critico e importante del BMS è lo stato di carica (SOC), definito come una misura della quantità di energia residua rispetto alla capacità nominale della batteria. L'attento monitoraggio del SOC è fondamentale per evitare il sovraccarico o la scarica profonda della batteria e, garantire un funzionamento affidabile e sicuro del veicolo [8]. Il Coulomb Counting(CC) è un metodo di stima del SOC semplice e diretto che si basa su un'integrazione nel tempo delle correnti di

carica e scarica della batteria [9, 10]. Tuttavia, richiede un'accurata conoscenza del valore iniziale del SoC e una calibrazione periodica della capacità. Inoltre, è un algoritmo a circuito aperto e potrebbe essere soggetto a derive a lungo termine a causa di disturbi incerti ed errori di misurazione [6]. Molti metodi indiretti sono stati sviluppati per superare le carenze della tecnica CC. Sono tipicamente classificati in metodi basati sui dati (DD) e basati su modelli. I metodi DD considerano la batteria come una scatola nera, cioè non hanno bisogno di una conoscenza accurata del suo modello fisico o dei parametri interni. In [11], è stata proposta una rete neurale ricorrente (RNN) guidata dinamicamente per stimare il SOC e lo stato di salute (SOH) della batteria. La superiorità rispetto al percettrone multistrato (modello di rete neurale artificiale che mappa insiemi di dati in ingresso in un insieme di dati in uscita appropriati) convenzionale è stata dimostrata in termini di precisione nella stima del SOC. Tuttavia, l'apprendimento della RNN può fallire con la dipendenza a lungo termine [12]. In [12] il SOC è stato stimato utilizzando una rete di memoria a lungo termine impilata. L'errore di stima del SOC massimo assoluto ottenuto era nell'intervallo del 2%.

In [13], è stato sviluppato un modello basato su Support Vector Machine per stimare il SOC di una batteria al litio ferro manganese fosfato. L'errore di previsione del SOC ottenuto viene mantenuto al di sotto del 6%, con un valore

quadratico medio (RMS) della radice nell'intervallo dello 0,6%. Oltre alla stima del SOC, sono stati sviluppati anche vari modelli DD per costruire strumenti basati sull'intelligenza artificiale per la produzione di batterie e la diagnosi della vita, come discusso nel documento di revisione [14]. Sebbene i metodi DD abbiano dimostrato prestazioni soddisfacenti, il loro processo di apprendimento richiede una grande quantità di dati di carica-scarica per approssimare la relazione non lineare tra il SOC e le variabili di input. Inoltre, i metodi DD possono anche non essere convergenti quando i dati di addestramento non coprono completamente la regione operativa effettiva [15]. I metodi basati su modelli utilizzano un modello analitico che dovrebbe descrivere accuratamente il comportamento non lineare della batteria. Vengono usati filtri di osservazione non lineari per stimare il SOC. I metodi più popolari sono il filtro di Kamlan (KF) [16], l'osservatore Luenberger [15], l'osservatore H∞ [17], il filtro antiparticolato [18], il filtro a struttura variabile liscia (SVSF) [19], l'osservatore in modalità scorrevole [20]. Due tipi di modelli sono comunemente usati, vale a dire i modelli elettrochimici (EM) e i modelli di circuiti equivalenti (ECM). EM utilizza equazioni differenziali alle derivate parziali accoppiate e variabili nel tempo per descrivere i fenomeni fisico-chimici all'interno della batteria come diffusione, cinetica elettrochimica ecc. Ad esempio, in [21] il SOC è stimato utilizzando un osservatore di iniezione di

uscita e un modello EM ridotto della batteria basato su particelle singole. In [17], il modello di impedenza elettrochimica è stato utilizzato con l'osservatore H∞ per stimare il SOC di una batteria agli ioni di litio. In [22], è stato proposto un modello ibrido elettrochimico-termico-rete neurale (ETNN) per ridurre la complessità di calcolo. Il sottomodello elettrochimico-termico (ETSM) comprende un modello a singola particella e un modello termico concentrato per prevedere rispettivamente la tensione del terminale della batteria e la temperatura interna. Successivamente, una rete neurale feed-forward viene aggiunta a questo ETSM per migliorare le sue prestazioni in condizioni di temperatura estreme e grandi correnti. Il modello ETNN viene utilizzato con l'UKF (Unscented Kalman Filter) per ottenere una co-stima del SOC e dello stato di temperatura. L'errore quadratico medio (RMSE) della stima del SOC viene mantenuto al di sotto dell'1% a temperature diverse. Certamente i metodi EM sono accurati grazie alla loro maggiore capacità di catturare la dinamica e la cinetica del trasferimento di carica all'interno della batteria [23]. Tuttavia, non sono preferiti per l'implementazione in tempo reale a causa della loro complessità e del maggiore onere computazionale [24]. L'ECM utilizza componenti elettrici di base come resistori, condensatori e fonti di tensione per descrivere la caratteristica della batteria [25]. Sono semplici e facili da implementare, il che li rende molto adatti per le applicazioni online [26].

Tuttavia, i parametri ECM non sono costanti e variano con molti fattori interni ed esterni come la temperatura, il livello di SOC, l'invecchiamento e il C-rate [27]. Il comportamento della batteria è anche non lineare, in particolare nella regione di SOC basso [28]. Inoltre, le misurazioni di tensione e corrente sono seriamente influenzate da rumori sconosciuti e interferenze elettromagnetiche (EMI). Queste carenze rendono difficile l'identificazione dei parametri della batteria e peggiorano la precisione dei metodi di stima SOC basati su modelli. Considerando questo, è importante eseguire una stima affidabile dei parametri ECM in diverse condizioni per monitorare accuratamente il SOC della batteria. Diversi metodi sono stati proposti nella letteratura recente per identificare i parametri della ECM. In [29], gli autori hanno utilizzato un adattamento di funzione esponenziale per identificare i parametri di una ECM di secondo ordine a cinque temperature ambiente. Il SOC è stato stimato con il filtro Kalman esteso (EKF) e l'RMSE massimo è del <2% a diversi livelli di temperatura. In [27], gli autori hanno applicato il metodo non lineare del minimo quadrato per estrarre i parametri di una ECM del secondo ordine da test di caratterizzazione della potenza ibrida dell'impulso, condotti a vari livelli di temperatura. L'errore viene mantenuto entro il 2% nella maggior parte degli scenari operativi. In [30], i parametri della batteria sono stimati a 25°C utilizzando un'interpolazione sp-line tra i valori ottenuti rispettivamente a 20 e

30°C. Su questa base, l'UKF è stato utilizzato per stimare il SOC. L'errore massimo a varie temperature viene mantenuto entro il 3%. In [23], gli autori hanno utilizzato la struttura vincente per modellare la funzione OCV-SOC e l'algoritmo RLS (recursive least square con extended-kernel iterativo) per stimare i parametri di una ECM del secondo ordine. Il metodo viene testato con l'EKF e l'RMSE massimo del SOC stimato e viene mantenuto entro lo 0,56%. Il problema chiave dei metodi convenzionali di identificazione dei parametri basati su RLS è la loro sensibilità al rumore di misura [28]. A questo proposito, è stato sviluppato un metodo RLS basato su procedure di valutazione delle informazioni in [28] per superare questa limitazione. Gli autori hanno utilizzato la matrice di informazioni di Fisher e il limite inferiore di Cramer-Rao per valutare le informazioni misurate. Il metodo è stato testato con l'EKF e convalidato utilizzando un ECM di primo ordine. L'RMSE massimo del SOC stimato è compreso tra lo 0,6% e diverse condizioni di temperatura.

Un'ulteriore variante dell'EKF consiste nell'introduzione all'interno del modello della batteria del fenomeno di isteresi.

In [31] è riportato che le caratteristiche di isteresi delle batterie agli ioni di litio siano dovute all'intercalazione degli ioni di litio negli elettrodi di carbonio e LiMn2O4, che porta alla differenza di equilibrio tra carica e scarica della batteria. In [32] viene proposto un modello di EKF con isteresi modellata sul

OCV seguendo un profilo Preisach, i risultati hanno dimostrato un notevole aumento delle prestazioni sulla stima dello stato di carica.

La scelta finale sul metodo di stima adoperato per questo lavoro è stata fortemente determinata dai dati disponibili ed è confluita sul filtro di Kalman esteso con l'utilizzo del modello di Thevenin del primo ordine.

Nella tabella 1.1 sono riportati i principali metodi descritti in questo capitolo.

Nome Metodo	**Complessità**	**Vantaggi**	**Svantaggi**
Coulomb Counting	Semplice	Versatilità e integrazione	Soggetto a derive causate dalla mancanza di aggiornamento dei parametri
Filtro di Kalman	Moderata	Sufficientemente e computazionalmente leggero	Necessaria un'accurata stima dei parametri del modello interno
Intelligenza artificiale	Complessa	Buoni risultati di stima	Necessaria una grande mole di dati per l'allenamento della rete
Metodi EM	Complessa	Buoni risultati di stima	Richiede molti calcoli, difficile da usare per stime online

Tabella 1.1 Confronto fra metodi principali di stima

2 Batterie agli Ioni di Litio

Una batteria agli ioni di litio è un dispositivo di immagazzinamento dell'energia che funziona attraverso una reazione legata agli ioni di litio.

Grazie alla promozione di nuovi veicoli elettrici, l'industria delle batterie agli ioni di litio ha inaugurato il suo periodo di boom consentendo un rapido sviluppo e mostrando un grande potenziale.

Esistono diverse modalità che sfruttano il principio di funzionamento di questo tipo di batterie e molte varianti riguardano un differente tipo di materiali utilizzati nella struttura delle singole celle.

Nel 1980, il gruppo di ricerca di John B. Goodenough scoprì per la prima volta che il litio cobaltato può essere usato come materiale catodico.

Nel 1997, il professor Goodenough riferì che il litio ferro fosfato è un cristallo con una struttura di olivina ordinata, in cui l'esaedro di ferro-ossido è collegato tra loro sullo stesso lato. Questa struttura bilancia perfettamente l'energia redox degli elettrodi di litio ferro fosfato con la mobilità degli ioni di litio nel cristallo, e le sue caratteristiche possono soddisfare i requisiti delle batterie agli ioni di litio di potenza, con la sua capacità, le prestazioni del ciclo e la sicurezza ovviamente migliorate. Inoltre, questo materiale è più economico e più sicuro del litio cobaltato quindi è stato molto popolare nei successivi 20 anni [33].

2.1 Principio di Funzionamento delle Batterie agli Ioni di Litio

Il meccanismo di funzionamento di una batteria agli ioni di litio è comune a quello delle altre batterie, cioè l'energia chimica della batteria viene convertita in energia elettrica e inviata al circuito esterno durante il processo di scarica. Quindi, la batteria agli ioni di litio viene ripristinata al suo stato originale attraverso l'elettrificazione inversa, data da una sorgente esterna nella fase di ricarica; durante quest'ultimo processo, gli ioni di litio e gli elettroni nel catodo sono separati dal reticolo, mentre gli ioni di litio migrano verso l'anodo nell'elettrolita e gli elettroni migrano verso l'anodo attraverso il circuito esterno venendo infine incorporati nel materiale anodico. La scarica è opposta alla carica quando gli elettroni guidano dispositivi elettronici attraverso circuiti esterni.

2.2 Capacità

La capacità della batteria rappresenta la quantità di elettricità scaricata dalla batteria in determinate condizioni, che è uguale alla corrente di scarica moltiplicata per il tempo di scarica, di solito in Ah.

La capacità della batteria è suddivisa in capacità effettiva e capacità teorica. La capacità effettiva si riferisce alla quantità di elettricità che può essere fornita in un determinato sistema di scarica, mentre la capacità teorica si riferisce alla

quantità di elettricità fornita da tutti i principi attivi che partecipano alle reazioni della batteria. La capacità di scarica diminuisce con l'aumentare della profondità di scarica [34]. Maggiore è il tasso di scarica, più velocemente diminuisce la capacità della batteria. Un tasso di scarica eccessivo può danneggiare la batteria. La migliore temperatura di lavoro di una batteria agli ioni di litio è compresa tra i 20 °C e i -40 °C, infatti temperature più alte o più basse possono compromettere temporaneamente la capacità di una batteria, quindi, il degrado delle prestazioni della batteria è influenzato anche dalla temperatura.

2.3 Tasso di Autoscarica

Il tasso di autoscarica si riferisce alla capacità di mantenimento dell'elettricità immagazzinata della batteria in determinate condizioni quando la batteria è in uno stato di circuito aperto. Ci sono sempre impurità nei materiali, che inevitabilmente portano all'autoscarica.

Maggiore è la temperatura ambiente, maggiore è l'attività dei materiali elettrochimici, più intense le reazioni interne e più evidente è l'autoscarica chimica. Le batterie agli ioni sono facilmente influenzate dall'inquinamento atmosferico e dall'umidità dell'aria. L'elevata umidità dell'aria e le particelle conduttive possono portare ad un aumento della conduttività.

La tendenza generale è che una maggiore carica della batteria agli ioni di litio comporta una maggior differenza di potenziale, le reazioni interne alla chimica della batteria saranno quindi più intense e questo andrà ad aumentare il tasso di autoscarica.

Diverse impurità influenzano il tasso di autoscarica. Dopo aver caricato la batteria, le impurità metalliche situate tra il catodo e il separatore vengono ossidate e dissolte nell'elettrolita dal catodo per generare ioni metallici, che vengono continuamente ridotti e depositati sulla superficie dell'anodo. I dendriti metallici si depositano lentamente perforando il separatore ed entrando in contatto con il catodo, formando punti di cortocircuito e aumentando poi l'autoscarica [35].

2.4 Densità di Energia

La densità di energia si riferisce al rapporto tra l'energia che può essere caricata e la massa o il volume del mezzo di accumulo di energia per un dato dispositivo. La densità di energia di massa è uguale alla capacità della batteria divisa per il peso della batteria, con Wh / kg come unità. La densità di energia del volume è uguale alla capacità della batteria diviso il volume di scarica, con Wh/l come unità. La velocità di scarica di una batteria agli ioni di litio si riferisce al valore

di corrente richiesto quando la batteria rilascia la sua capacità nominale entro il tempo specificato.

Le dimensioni della batteria influiscono sulla densità di energia. Aumentare la proporzione di materiali attivi positivi migliorerà la densità energetica. Sono necessari più materiali attivi negativi per accogliere gli ioni di litio e immagazzinare energia. Rispetto alla quantità totale costante di materiale attivo positivo, solo quando il maggior numero possibile di ioni di litio viene deintercalato dal catodo, cioè la capacità specifica viene aumentata, è possibile migliorare la densità di energia [33].

2.5 Densità di Potenza

La densità di potenza si riferisce al rapporto tra la potenza di uscita della batteria rapportata al suo peso, questa quantità è solitamente espressa in W/kg. Maggiore è la densità di energia, maggiore è la densità di potenza [33].

3 Modello Matematico della Batteria

Per il calcolo dello stato della batteria è necessario l'utilizzo di un modello matematico accurato che ne simuli il comportamento.

Ad oggi esistono numerosi modelli matematici che stimano il comportamento delle batterie; i modelli di circuito equivalenti delle batterie agli ioni di litio includono principalmente il modello Rint [36], il modello di Thevenin del primo ordine [37, 38], il modello di Thevenin del secondo ordine [39], il modello PNGV [40]e il modello GNL [41]. Sulla base delle ricerche precedenti [42], i vantaggi e gli svantaggi di ciascun modello sono riassunti, come mostrato nella Tab 3.1.

Nome Metodo	**Complessità**	**Vantaggi**	**Svantaggi**	**Precisione**
Modello di Rint	Semplice	Versatile per via della semplicità	Non riflette le dinamiche della batteria	Bassa
Modello di Thevenin del primo ordine	Relativamente semplice	Simula bene le dinamiche della Batteria	Cambi di corrente non sono presi in considerazione	Bassa
Modello di Thevenin del secondo ordine	Più complesso	Simulazione molto buona delle dinamiche della batteria	Stima complessa dei parametri	Moderata
Modello PNGV	Molto Semplice	Simula bene le dinamiche della Batteria	L'errore di modifica dell'OCV è grande	Moderata
Modello GNL	Complesso	Riflette accuratamente le dinamiche della batteria con una buona accuratezza	La struttura è complessa e i parametri difficili da stimare	Molto alta

Tabella 3.1 Comparazione di modelli di batterie comuni

Prendendo in considerazione i dati sperimentali su cui è stato fatto il fitting è stato scelto di utilizzare il modello di Thevenin del primo ordine, poiché non è stato possibile reperire le informazioni necessarie alla modellazione di elementi circuitali superiori al primo ordine.

La scelta di questo modello, tuttavia, ha consentito un notevole risparmio nelle risorse computazionali, rendendo versatile l'esportazione del codice su piattaforma embedded.

3.1 Modello di Thevenin del Primo Ordine

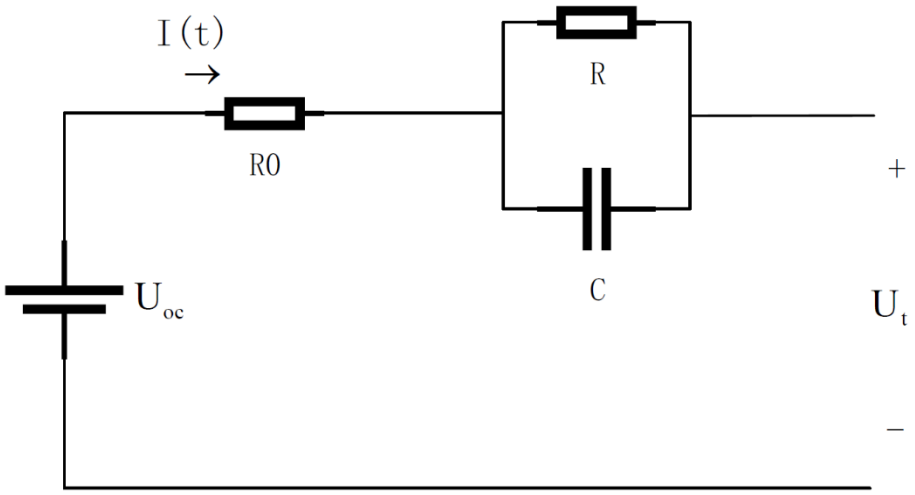

Figura 3.1 Diagramma del circuito equivalente del modello di Thevenin del primo ordine

Il modello di Thevenin del primo ordine introduce un loop RC basato sul modello Rint [42]. Il loop RC aggiunto può simulare meglio il lento cambiamento della tensione del terminale della batteria introducendo un

transitorio esponenziale del primo ordine che permetta la simulazione del recovery effect, ovvero riflettere il processo di polarizzazione della batteria, in modo che il modello possa rappresentare accuratamente la carica e la scarica della batteria rispetto al modello Rint. La struttura specifica è illustrata nella Figura 3.1.

Secondo le leggi di Kirchhoff possiamo ricavare le equazioni 3.1 e 3.2 come:

$$U_t = U_{ocv} - I(t) * R_0 - U \quad (3.1)$$

$$\frac{dU}{dt} = \left(-\frac{1}{RC}\right) * U + \frac{1}{C} * I(t) \quad (3.2)$$

Dove U_t è la tensione ai capi della batteria, U_{ocv} è la tensione a circuito aperto, R_0 rappresenta la resistenza interna, R e C sono rispettivamente la resistenza e la capacità di polarizzazione e U rappresenta la caduta di tensione prodotta dal gruppo RC.

4 Metodi di stima dello Stato di Carica

Come già descritto nel capitolo sullo stato dell'arte, esistono vari metodi di stima del SOC, in questo capitolo verranno approfonditi i metodi che saranno poi utilizzati nella trattazione di questo lavoro.

4.1 Coulomb Counting

La stima del SOC mediante Coulomb Counting si basa sulla misurazione della corrente e sull'integrazione di tale corrente nel tempo.

Lo stato di carica (SOC) di una cella della batteria non può essere misurato direttamente e viene stimato da altre misurazioni e parametri noti. Ciò porta a errori nella stima del SOC e all'impossibilità di sfruttare appieno la capacità della cella.

Il Coulomb Counting fornisce una variazione relativa del SOC e non un SOC assoluto. Se si misura la corrente in un determinato passaggio temporale, si ha una misura del numero di Ah che sono rimasti o sono stati ricevuti dalla batteria. Con il Coulomb Counting, il SOC può essere calcolato come il rapporto tra i Coulomb rimanenti e la capacità della batteria che si presume sia nota. L'approccio di Coulomb Counting alla stima del SOC può essere approssimato come segue:

$$SOC(t) = SOC(t-1) + \frac{\Delta_t i(t)}{3600 C_{batt}} \quad (4.1)$$

Dove SOC(t) indica lo stato di carica all'istante t, Δ_t è il tempo di campionamento, $i(t)$ è la corrente che scorre nella batteria all'istante t e C_{batt} è la capacità della batteria espressa in Ampere per ora (Ah).

Il Coulomb Counting richiede che il SOC iniziale sia corretto per riuscire a calcolare la stima; è inoltre noto che l'errore di misurazione commesso sulla corrente influirà sul SOC calcolato.

Un altro punto debole risiede nell'incertezza sulla conoscenza della capacità della batteria: quest'ultima cambia a causa della temperatura, dell'età, ecc. L'incertezza nella capacità della batteria influenzerà il SOC calcolato con il Coulomb Counting [43].

4.2 Filtro di Kalman

Il filtro di Kalman è uno stimatore del problema lineare quadratico, esegue cioè il compito di stimare istantaneamente lo stato di un sistema lineare, dinamico, utilizzando misure lineari correlate allo stato ma perturbate dalla presenza di rumore bianco [44].

È stato sviluppato da Rudolf Emil Kalman negli anni '60 ed è ampiamente utilizzato in diversi campi, tra cui ingegneria, navigazione, automazione, intelligenza artificiale e altre applicazioni in cui è necessario stimare lo stato di un sistema basandosi su misurazioni soggette a incertezza.

L'obiettivo principale del filtro di Kalman è quello di ridurre il rumore e migliorare la precisione della stima dello stato del sistema, combinando le informazioni provenienti da due fonti principali: la previsione dello stato basata sul modello matematico del sistema e le misurazioni in ingresso. L'idea fondamentale è quella di pesare le informazioni provenienti da ciascuna fonte in base alla loro incertezza relativa.

Il filtro di Kalman opera in due fasi principali: la fase di previsione e la fase di correzione. Durante la fase di previsione, utilizza il modello matematico del sistema per stimare lo stato futuro, tenendo conto anche dell'incertezza associata a questa predizione. Successivamente, durante la fase di correzione, dopo aver ottenuto nuove misurazioni dal sistema, il filtro di Kalman corregge la stima precedente tenendo conto sia delle misurazioni che delle previsioni fatte nella fase di predizione. L'idea principale è quella di dare più peso alle misurazioni con maggiore affidabilità (minore rumore) e meno peso alle misurazioni meno affidabili. In sostanza, il filtro di Kalman tiene conto sia delle informazioni a priori basate sul modello del sistema che delle informazioni misurate per ottenere una stima ottimale dello stato del sistema, considerando l'incertezza e il rumore associati alle misurazioni.

L'efficacia del filtro di Kalman dipende dalla corretta modellizzazione del sistema e dalla stima accurata delle incertezze coinvolte. È noto per la sua

capacità di fornire stime ottimali, nel senso della minimizzazione della varianza dell'errore, date le condizioni di incertezza iniziale e il rumore di misurazione.
Il filtro di Kalman assume che il sistema segua un modello lineare e che l'errore sia distribuito in modo gaussiano. In situazioni in cui il sistema è fortemente non lineare o in presenza di rumori non gaussiani, potrebbero essere necessarie varianti più avanzate del filtro, come il filtro di Kalman esteso (EKF).
Il filtro di Kalman è lo strumento più utilizzato per problemi di stima dinamici in cui lo stato e le misure sono affetti da rumori stocastici di cui è nota la caratterizzazione statistica, come nel caso in esame.

4.2.1 Filtro di Kalman Lineare

Il funzionamento del filtro di Kalman è basato sulla minimizzazione della varianza dell'errore tra lo stato stimato e lo stato reale del sistema.
Al fine di implementare il filtro ottimo bisogna avere a disposizione delle caratterizzazioni statistiche accurate, in termini di media e varianza, dei rumori presenti nel sistema e dell'errore di misura.
Si consideri il seguente sistema lineare tempo-variante:

$$\dot{x} = F(t)x + B(t)u + G(t)w \quad (4.2)$$

$$y_k = Hx_k + v_k \quad (4.3)$$

dove x rappresenta il vettore degli stati, u quello degli ingressi, y le misure delle uscite, w un rumore bianco a media nulla sugli stati con densità spettrale Q e

matrice di distribuzione G, e v un rumore bianco a media nulla sulle uscite con matrice di covarianza R.

Si consideri inizialmente solo l'equazione che descrive l'evoluzione dello stato del sistema. Essendo il termine di rumore impredicibile, è possibile scrivere l'equazione differenziale che descrive la propagazione della stima ottima dello stato come:

$$\dot{\hat{x}} = F(t)\,\hat{x} + B(t)u \quad (4.4)$$

Si può quindi definire l'errore di stima come:

$$\varepsilon = x - \hat{x} \quad (4.5)$$

E la sua propagazione come:

$$\dot{\varepsilon} = F(t)\varepsilon + G(t)w \quad (4.6)$$

La varianza dell'errore sullo stato è definita dalla relazione seguente:

$$P \equiv E\{\varepsilon\varepsilon^T\} \quad (4.7)$$

La cui propagazione è data da:

$$\dot{P} = E\{\varepsilon\dot{\varepsilon}^T\} + E\{\dot{\varepsilon}\varepsilon^T\} = FP + PF^T + GQG^T \quad (4.8)$$

nella quale per compattezza è stata omessa la dipendenza dal tempo, risolvibile ponendo una condizione iniziale $P(t_0) = P0$.

Si definisca ora la seguente forma discretizzata della correzione della stima dello stato:

$$\hat{x}_k^+ = \hat{x}_k^- + K_k[y_k - H_k\hat{x}_k^-] \quad (4.9)$$

nella quale i segni - e + indicano gli istanti di tempo subito prima e subito dopo una misurazione, mentre il pedice k denota l'istante di tempo discreto t_k. Il vettore $\hat{x}_k^-$ deriva dalla propagazione della stima definita dalla (4.4).

Sfruttando la (4.9) è possibile scrivere la relazione (4.7) come:

$$P_k^+ = E\{(x_k - \hat{x}_k^+)(x_k - \hat{x}_k^+)^T\} \ (4.10)$$

Sostituendo le equazioni (4.3) e (4.9) nella (4.10) si ottiene la seguente equazione che descrive l'aggiornamento della matrice P in tempo discreto:

$$P_k^+ = [I - K_k H_k]\, P_k^-\, [I - K_k H_k]^T + K_k R_k K_k^T \ (4.11)$$

Dove P_k^- si ottiene dalla (4.8) ed il guadagno ottimo K_k si ottiene minimizzando la seguente cifra di costo:

$$J_k = trP_k^+ \ (4.12)$$

La cui soluzione è data da:

$$K_k = P_k^- H_k^T [H_k P_k^- H_k^T + R_k]^{-1} (4.13)$$

Infine, sostituendo la (4.13) nella (4.11) si arriva ad una forma più compatta della propagazione in tempo discreto della matrice P, descritta dalla relazione seguente:

$$P_k^+ = [I - K_k H_k] P_k^- \ (4.14).$$

Prima di concludere è utile osservare che, nel caso in cui il sistema sia tempo invariante, è possibile utilizzare una forma statica della matrice P.

La motivazione risiede nel fatto che, per questa classe di sistemi, sotto opportune ipotesi la matrice P converge ad un valore costante. In questo caso

anche il guadagno K del filtro converge ad un valore costante ed è quindi possibile trascurare il transitorio iniziale andando ad utilizzare direttamente i valori di regime nell'algoritmo del filtro di Kalman, riducendo notevolmente il carico computazionale.

Il valore di regime della matrice `P` può essere calcolato sfruttando quella che viene definita equazione algebrica di Riccati (ARE dall'inglese Algebric Riccati Equation).

4.2.2 Filtro di Kalman Esteso

Nella pratica si incontrano spesso problemi di stima dello stato che coinvolgono modelli non lineari. È questo il caso della stima dello stato di carica di una batteria.

Dato che il filtro di Kalman illustrato nella sezione precedente è valido esclusivamente per sistemi lineari ed è quindi impossibile utilizzarlo direttamente per la stima dello stato di sistemi non lineari, è importante fornire un'alternativa valida per poter affrontare il problema della stima anche su questa classe di sistemi.

A tale scopo si consideri ad esempio il generico sistema non lineare:

$$\dot{x}(t) = f(x(t), t) + G(t)w(t) \quad (4.15)$$

Il metodo più comune per affrontare il problema della stima dello stato per questa classe di sistemi è l'utilizzo di quello che viene chiamato filtro di Kalman esteso. Esso si basa sull'algoritmo del filtro di Kalman, ma apporta alcune modifiche, discusse nel seguito, che lo rendono adatto anche a sistemi non lineari.

Come nel caso lineare, l'obiettivo del filtro è quello di fornire una stima dello stato che sia il più vicina possibile allo stato vero del sistema.

Il primo passo da compiere per poter sfruttare in parte la teoria già vista nella Sezione 4.1.1 è di definire una linearizzazione dello stato del sistema (4.15):

$$\dot{x} \approx f(\hat{x}) + \left.\frac{df}{dx}\right|_{\hat{x}} (x - \hat{x}) \quad (4.16)$$

Il filtro di Kalman esteso utilizza l'equazione che descrive la dinamica non lineare per la propagazione dello stato, ma sfrutta la (4.16) per la correzione, ovvero per il calcolo delle matrici P e K, così come visto per il filtro di Kalman. Ogni volta che viene effettuata la propagazione il sistema viene linearizzato nuovamente. In questo modo anche il guadagno K e la matrice di varianza dell'errore P devono essere aggiornate.

Il filtro di Kalman esteso prevede dunque un calcolo online della matrice di varianza P e del guadagno K, a differenza di quanto accade per i sistemi lineari tempo-invarianti per i quali è possibile un calcolo a priori di questi parametri.

In Tabella 4.1 è riportato l'algoritmo che riassume il funzionamento del filtro di Kalman esteso, il pedice k indica il tempo discreto, gli apici + e - indicano gli istanti di tempo subito prima e subito dopo l'aggiornamento della variabile.

Modello	$\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t), t\right) + G(t)\mathbf{w}(t)$, con $\mathbf{w} \sim WN(\mathbf{0}, Q)$ $\mathbf{y}_k = \mathbf{h}\left(\mathbf{x}_k\right) + \mathbf{v}_k$, con $\mathbf{v}_k \sim WN(\mathbf{0}, R_k)$
Inizializzazione	$\hat{\mathbf{x}}(t_0) = \mathbf{x}_0,\ P(t_0) = P_0$
Propagazione	$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), t)$ $\dot{P}(t) = F(t)P(t) + P(t)F^T(t) + G(t)Q(t)G^T(t)$ $F \equiv \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big\vert_{\hat{\mathbf{x}}}$
Guadagno	$K_k = P_k^- H_k^T \left[H_k P_k^- H_k^T + R_k\right]^{-1}$ $H_k \equiv \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\Big\vert_{\hat{\mathbf{x}}_k^-}$
Aggiornamento	$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k[\mathbf{y}_k - \mathbf{h}(\hat{\mathbf{x}}_k^-)]$ $P_k^+ = [I - K_k H_k]P_k^-$

Tabella 4.1 Algoritmo EKF.

5 Matlab e Simulink

Per estrarre i parametri della batteria e simularne il comportamento è stato creato un modello attraverso il software Matlab creato dalla MathWorks.

Matlab è una piattaforma di programmazione e calcolo numerico utilizzata per l'analisi di dati e la creazione di modelli e algoritmi.

All'interno di questo software è possibile installare l'estensione Simulink, un ambiente di sviluppo con diagrammi a blocchi, utilizzato per progettare sistemi con modelli multidominio, effettuare simulazioni prima di passare all'hardware e procedere alla distribuzione.

Matlab mette a disposizione numerosi pacchetti aggiuntivi ideati per esigenze di progettazione in differenti ambiti ingegneristici e di progettazione.

5.1 Simulink Design Optimization

Simulink Design Optimization è un'estensione che contiene funzioni, strumenti interattivi e blocchi per analizzare e regolare i parametri del modello. È possibile determinare la sensibilità del modello, adattare il modello ai dati di test e regolarlo in modo che risponda ai requisiti. Sfruttando tecniche quali la simulazione Monte Carlo e la progettazione di esperimenti, è possibile esplorare il proprio spazio di progettazione e calcolare l'influenza dei parametri sul comportamento del modello.

È possibile pre-elaborare i dati di test, stimare automaticamente i parametri del modello e convalidare i risultati delle stime.

Per migliorare le caratteristiche di progettazione dei sistemi come il tempo di risposta, la larghezza di banda e il consumo di energia, è possibile ottimizzare congiuntamente i parametri degli impianti fisici e i guadagni algoritmici o dei controller. Questi parametri possono essere regolati in base a requisiti specifici nel dominio del tempo e in quello della frequenza.

Questo strumento è stato utilizzato nella prima fase del progetto per estrapolare i parametri della batteria ed ottimizzarli al fine di ottenere un modello quanto più preciso nella stima.

5.2 Simulink Coder

Simulink Coder genera ed esegue codici C e C++ da modelli Simulink, grafici Stateflow e funzioni MATLAB. Il codice sorgente generato può essere utilizzato per applicazioni in tempo reale e non, tra cui l'accelerazione della simulazione, la prototipazione rapida e l'hardware-in-the-loop testing.

Al fine di inserire il codice del filtro di Kalman all’interno di una piattaforma embedded è stata utilizzata questa funzione per generare codice C direttamente dal modello per poi utilizzarla per dei test in tempo reale.

5.3 *Modelli Simulink*

Per estrapolare i parametri da inserire nel filtro di Kalman è stato creato un primo modello da utilizzare con una funzione di stima, successivamente i parametri sintetizzati nel primo step sono stati inseriti in un secondo modello che comprendeva anche il filtro di Kalman per valutarne le prestazioni, infine è stato creato un modello finale del filtro ottimizzato per l'esportazione di codice in linguaggio C, per l'utilizzo su piattaforma embedded e test sul campo.

5.3.1 *Modello per la stima dei Parametri*

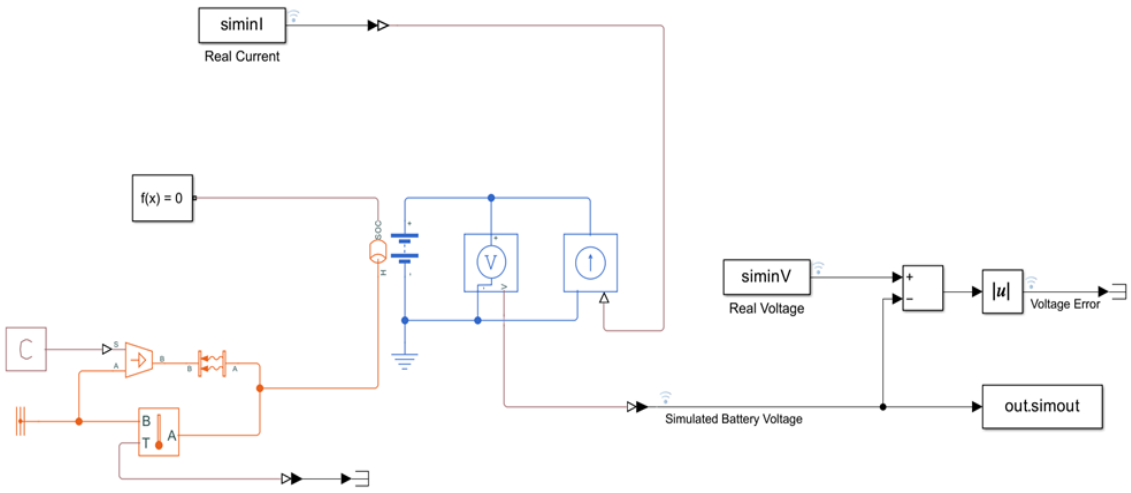

Figura 5.1 Modello Simulink della batteria per la stima dei parametri

Il modello utilizzato per stimare i parametri si compone di una batteria il cui comportamento è descritto da un circuito di Thevenin del primo ordine,

collegato ad un segnale di corrente proveniente direttamente dal Workspace Matlab.

Come riferimento per la simulazione è stato previsto anche l'inserimento di un segnale di tensione collegato all'andamento di corrente; i parametri della batteria sono stati impostati dal Workspace Matlab.

La tensione di uscita della batteria viene prelevata e confrontata con il segnale della tensione acquisita durante i test in tempo reale, ottenendo anche un segnale che rappresenti l'errore commesso.

In Figura 5.1 è mostrato il diagramma Simulink della batteria.

Il modello così composto è stato utilizzato per ottimizzare la tensione di uscita della batteria con quella acquisita dai dati sperimentali attraverso l'estensione "Simulink Design Optimization".

5.3.2 Modello per la simulazione del filtro di Kalman

Il modello ottimizzato è stato inserito in questo secondo file Simulink per testare l'introduzione del filtro di Kalman.

Come segnali d'ingresso sono stati impostate le tensioni e le correnti registrate su campo, sono stati inseriti dei log per monitorare gli errori commessi nelle varie grandezze stimate, compreso il SOC acquisito durante i test.

Nel modello è presente un selettore che serve a commutare la tensione in ingresso al filtro di Kalman, utilizzato per valutarne le prestazioni sia con il segnale simulato dalla batteria che con il segnale reale.

Il filtro di Kalman ha al suo interno un modello di Thevenin del primo ordine e condivide la caratterizzazione dei parametri con la batteria simulata.

Nella figura 5.2 è riportato lo schema Simulink.

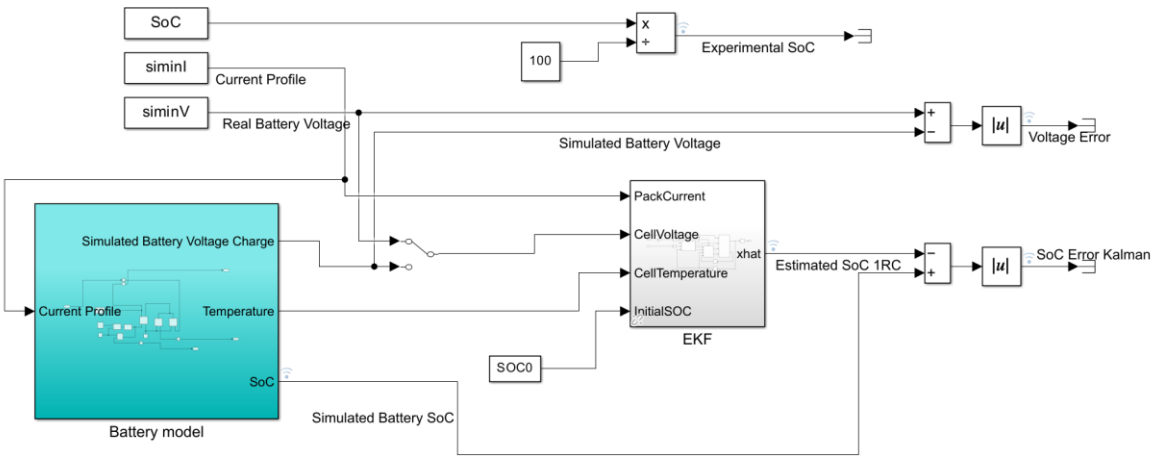

Figura 5.2 Modello Simulink con filtro di Kalman per la stima del SOC

5.3.3 Modello per l'esportazione del filtro di Kalman

L'ultimo modello Simulink generato è stato utilizzato per rifinire i dettagli relativi all'esportazione di codice.

È stato creato un file a parte per avere un controllo diretto sulla definizione dei tipi di variabili usate e per ripulire tutte le parti superflue.

Come mostrato in figura 5.3, sono presenti, oltre al modello del filtro, anche gli ingressi che simulano le correnti e le tensioni; questo è stato fatto per permettere di valutare il codice anche con altri test eseguiti sul filtro stesso e di riportarne poi le caratteristiche in simulazione ed effettuare, ove necessario, correzioni.

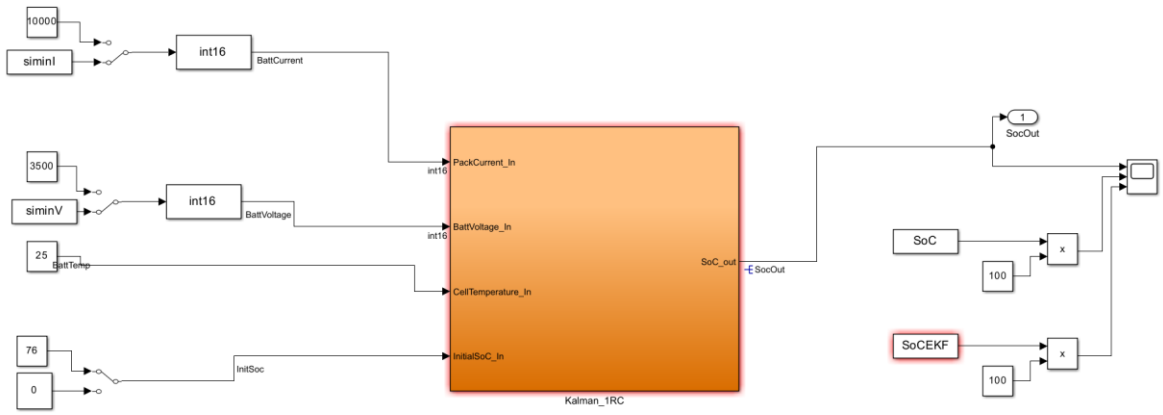

Figura 5.3 Modello del filtro di Kalman per l'esportazione su piattaforma embedded

In figura 5.4 viene mostrato l'interno del blocco arancione di figura 5.3, dove sono esplicitate le conversioni di tipo per il funzionamento del filtro e alcuni scaling delle grandezze utilizzate, questo perché il dispositivo con cui il filtro è stato interfacciato aveva già delle specifiche ben definite e il nuovo codice è stato adattato per il funzionamento.

All' interno del blocco EKF di figura 5.4 è presente il modello sviluppato nel corso delle precedenti simulazioni.

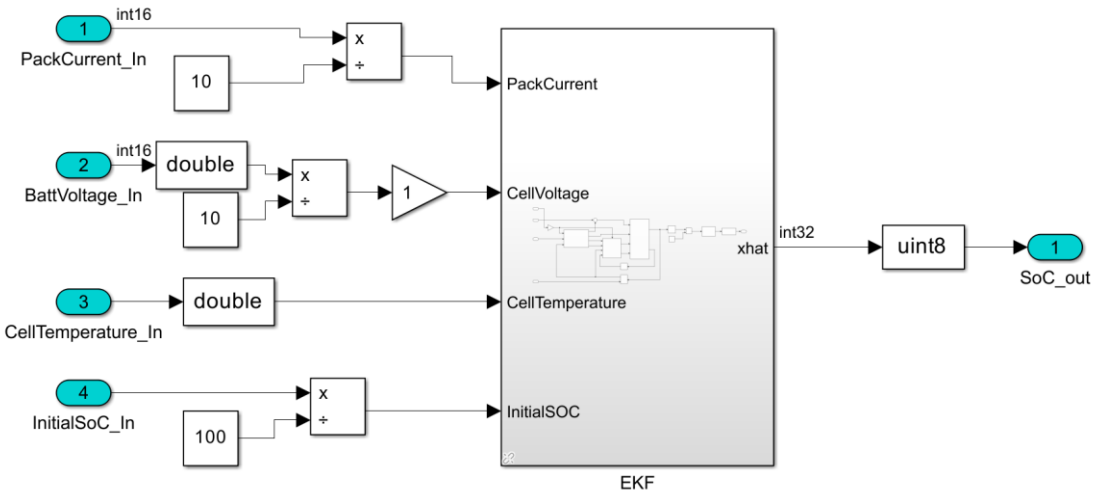

Figura 5.4 Dettaglio sulle conversioni di tipo nel modello Simulink

5.4 Script Matlab

I modelli Simulink descritti nelle sezioni precedenti utilizzavano un codice per la definizione dei parametri e per l'inizializzazione.

All'interno dello script vengono prima importati i file di log per definire i segnali registrati durante le simulazioni; questi segnali vengono processati e adattati per l'utilizzo tramite timetable e poi passati al Simulink come ingressi per le simulazioni; a titolo di esempio viene riportata una sezione del codice per l'importazione:

```
A = readtable ("Foglio1-2.xlsx");
        I_Profile = (table2array(A(:,4)));
        V_Profile = (table2array(A(:,3)));
        time = (table2array(A(:,1)));
        time = time';

        siminI = timeseries(I_Profile,time);
        siminV = timeseries(V_Profile,time);
        max_step = time(2);
```

Per poter utilizzare vari file sorgente è stata prevista una struttura di tipo "switch-case" con una variabile associata per la definizione del caso.

Nella seconda sezione del codice sono state definite le caratteristiche della batteria e i vettori da inserire nelle lookup-table per la caratterizzazione degli andamenti delle grandezze del modello di Thevenin in funzione del SOC.

Vista la diversità dei file contenenti i dati acquisiti, anche in questo caso è stata inserita una struttura "switch-case" e sono stati testati diversi modelli nella fase iniziale del progetto.

Ognuno dei casi contiene una definizione di un vettore che definisce i punti dello stato di carica, un vettore per la tensione a circuito aperto, uno per la resistenza interna della batteria e due ultimi vettori rispettivamente per la prima resistenza di polarizzazione e per la costante di tempo.

L'ultima sezione del codice è stata riservata alla definizione delle variabili del filtro di Kalman come la covarianza del rumore di processo e la covarianza dell'errore di misura.

```
%% Kalman Filter
R    = 0.7 ;              % Covariance of the measurement noise, R
SOC0 = 0.5;              % Initial SOC for estimator

Q3    = [1e-4 0; 0 1e-4 ]; % Covariance of the process noise,
P03   = [1e-5 0; 0 1 ];% Initial state error covariance, P0
```

6 Simulazioni e Risultati

In questo capitolo verranno descritti i dettagli delle simulazioni svolte, procedendo nella stesura con l'ordine adoperato durante il progetto.

Per prima cosa verranno descritte le modalità di stima dei parametri da inserire nel filtro, successivamente verranno descritti i test con il relativo setup per la verifica dei risultati con i parametri inseriti nel filtro di Kalman.

In ognuna di queste sezioni verranno riportati i risultati ottenuti.

La batteria da cui sono stati estrapolati i dati è composta da 84 collegamenti in serie di due celle in parallelo, la capacità totale è di 64Ah e la tensione di utilizzo varia dai 350V ai 290V.

6.1 Stima dei Parametri

Il primo step è stato quello di ottimizzare i dati ottenuti da una prima rudimentale stima acquisita dal datasheet delle celle componenti la batteria.

Poiché l'architettura della batteria è nota, la prima approssimazione per la stima dei parametri è stata eseguita moltiplicando le tensioni della cella ai vari stati di carica per il numero di celle collegate in serie. Per gli altri elementi componenti il modello della batteria, sono stati invece ipotizzati dei valori costanti conformi ai valori di tensione, da utilizzare come punto di partenza.

Nella tabella 6.1 sono riportati i valori utilizzati per il calcolo dei valori iniziali della tensione a circuito aperto.

I risultati ottenuti da questa prima approssimazione sono stati poi adattati ai livelli di SOC registrati nei dati sperimentali.

SOC	***Charge OCV***	***Discharge OCV***
100	4.248V	4.272 V
97	4.233 V	4.228 V
95	4.210 V	4.202 V
90	4.152 V	4.143 V
85	4.094 V	4.084 V
80	4.036 V	4.025 V
75	3.979 V	3.968 V
70	3.925 V	3.914 V
65	3.872 V	3.860 V
60	3.822 V	3.809 V
55	3.762 V	3.750 V
50	3.713 V	3.701 V
45	3.685 V	3.673 V
40	3.664 V	3.653 V
35	3.648 V	3.637 V
30	3.634 V	3.623 V
25	3.620 V	3.607 V
20	3.603 V	3.578 V
15	3.572 V	3.547 V
10	3.532 V	3.511 V
5	3.481 V	3.467 V
0	3.442 V	3.437 V

Tabella 6.1 Valori di tensione ai vari livelli di SOC forniti dal datasheet

Per la R0 è stato impostato un valore costante di 0.1Ω e per la R1 un valore di 0.01Ω, il valore di tau è stato inizialmente fissato a 299.23s.

I dati su cui sono state svolte le ottimizzazioni sono una unione di diversi test eseguiti su un'unica batteria già montata su un veicolo.

I parametri utilizzati per la prima simulazione sono riportati in tabella 6.2.

SOC	***OCV***	***R0***	***R1***	***Tau1***
100	356.8V	0.1Ω	0.01Ω	299.23s
97	355.6 V	0.1Ω	0.01Ω	299.23s
95	353.4 V	0.1Ω	0.01Ω	299.23s
90	348.8 V	0.1Ω	0.01Ω	299.23s
85	343.9 V	0.1Ω	0.01Ω	299.23s
80	339.0 V	0.1Ω	0.01Ω	299.23s
75	334.2 V	0.1Ω	0.01Ω	299.23s
70	329.7 V	0.1Ω	0.01Ω	299.23s
65	325.3 V	0.1Ω	0.01Ω	299.23s
60	321.0 V	0.1Ω	0.01Ω	299.23s
55	316.0 V	0.1Ω	0.01Ω	299.23s
50	311.9 V	0.1Ω	0.01Ω	299.23s
45	309.5 V	0.1Ω	0.01Ω	299.23s
40	307.8 V	0.1Ω	0.01Ω	299.23s
35	306.4 V	0.1Ω	0.01Ω	299.23s
30	305.3 V	0.1Ω	0.01Ω	299.23s
25	304.0 V	0.1Ω	0.01Ω	299.23s
20	302.7 V	0.1Ω	0.01Ω	299.23s
15	300.0 V	0.1Ω	0.01Ω	299.23s
10	296.7 V	0.1Ω	0.01Ω	299.23s
5	292.4 V	0.1Ω	0.01Ω	299.23s
0	289.1 V	0.1Ω	0.01Ω	299.23s

Tabella 6.2 Parametri utilizzati come stima iniziale

6.1.1 Risultati dell'Ottimizzazione

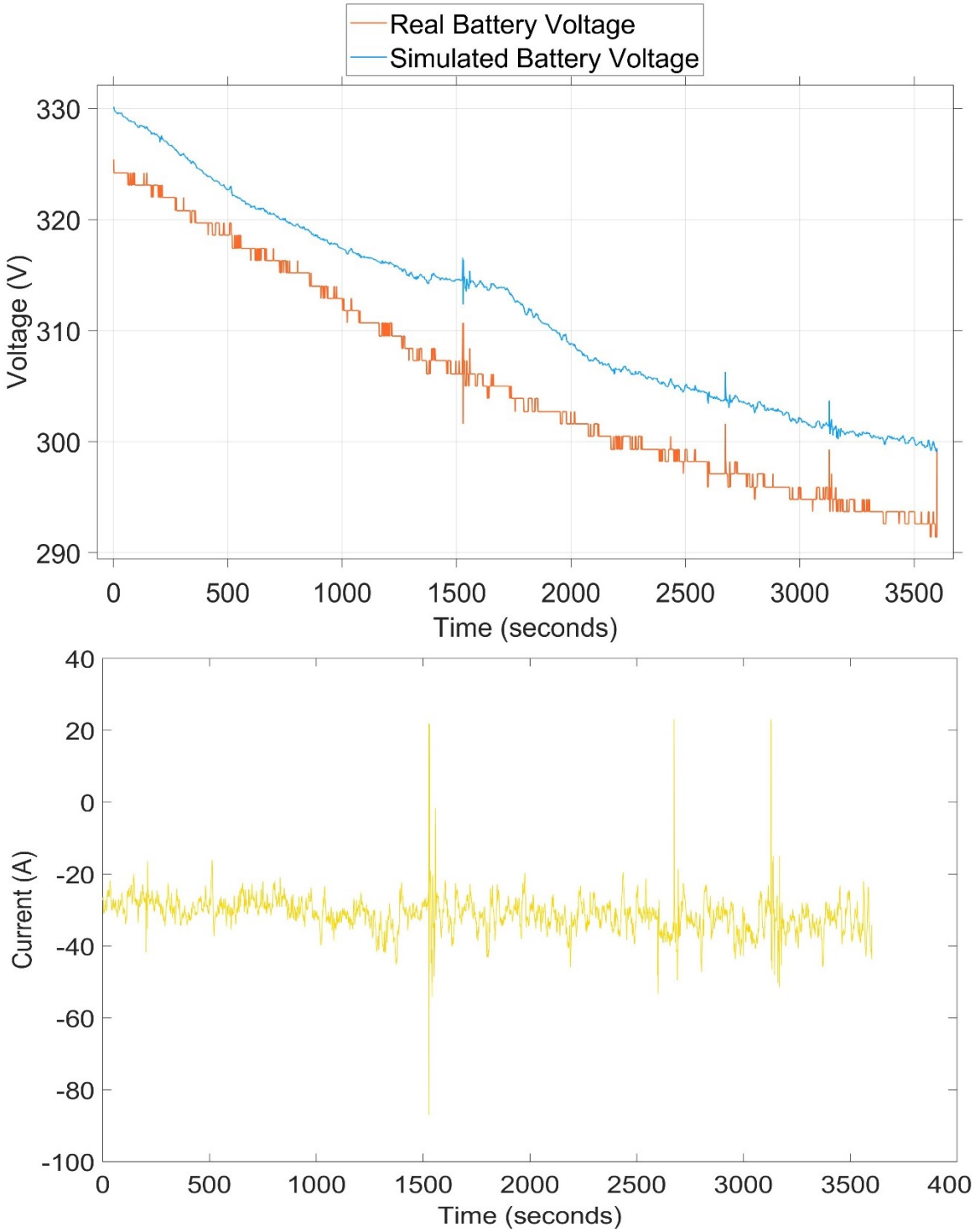

Figura 6.1 Andamento della tensione della batteria simulata e reale prima dell'ottimizzazione e relativo profilo di corrente

La prima approssimazione della batteria ottenuta con i dati presenti nel datasheet (riportati in figura 6.1) stima la tensione della batteria con un discreto errore.

Le tensioni riportate sono notevolmente più alte rispetto a quelle misurate durante i test e le cadute di tensione date dalle forti correnti in fase di accelerazione e recupero sono molto poco accentuate rispetto ai dati di riferimento.

Nella figura 6.1 è stato riportato per completezza anche l'andamento della corrente utilizzata per la stima dei parametri; nei grafici successivi è stato utilizzato sempre questo segnale come input del modello della batteria.

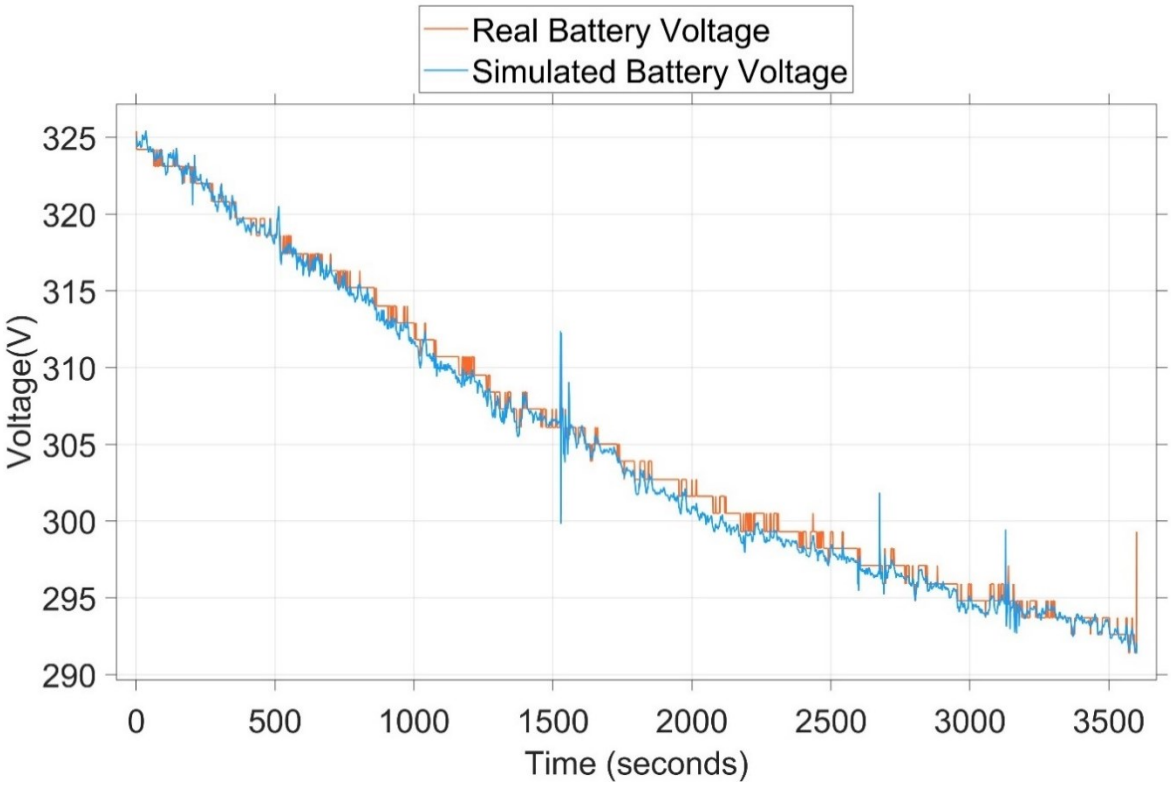

Figura 6.2 Andamento della tensione della batteria simulata e reale dopo l'ottimizzazione

La situazione riportata in figura 6.2 è sicuramente migliore; i valori di tensione dopo l'ottimizzazione della curva sono allineati con i dati sperimentali nelle fasi di rilassamento della batteria e le escursioni di tensione dovute alle forti correnti sono proporzionali.

La figura 6.3 mostra, infine, l'errore commesso nella stima delle tensioni dai due modelli simulati; l'andamento è stato ottenuto sottraendo al valore registrato nella batteria reale il valore ottenuto dal modello della batteria simulata, questo valore è stato poi posto in valore assoluto.

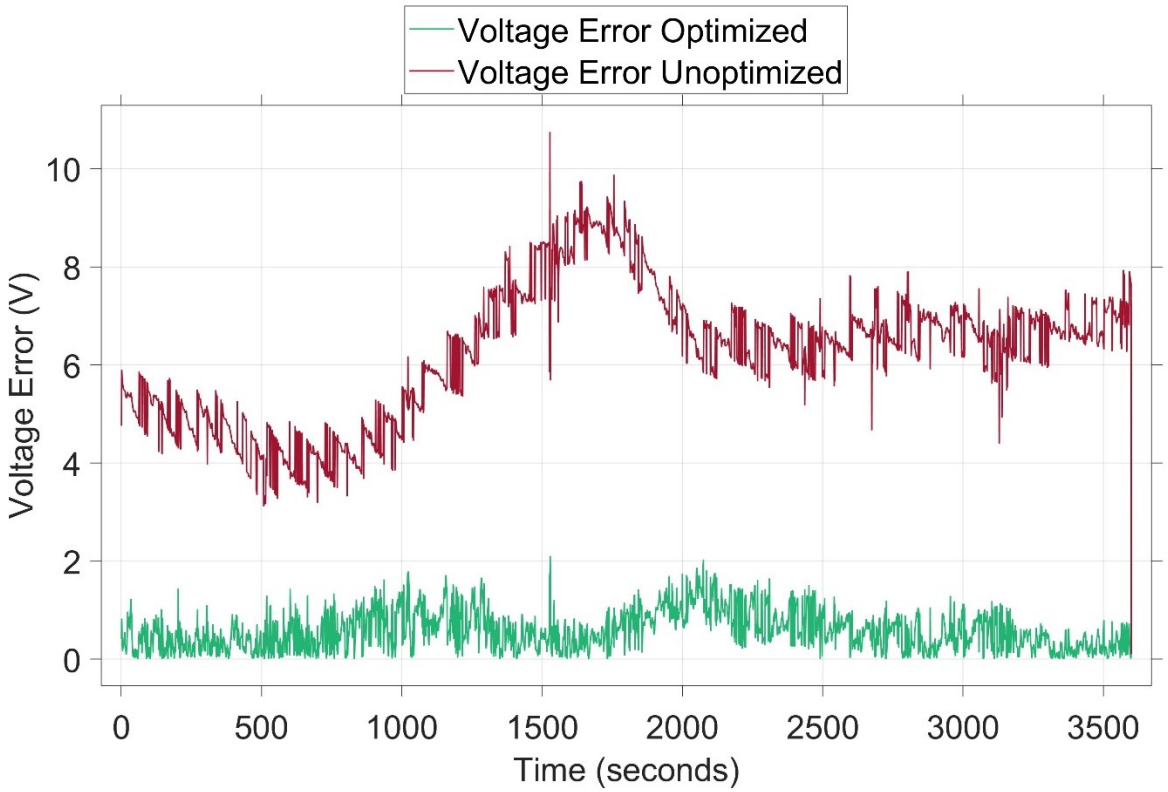

Figura 6.3 Confronto tra gli errori commessi dai due modelli nella rappresentazione della tensione.

La differenza tra le due curve si può notare sia nei tratti senza forti correnti, dove l'errore tende quasi a zero nel caso della curva ottimizzata, sia nei valori di picco dove le oscillazioni sono notevolmente ridotte.

Nella tabella 6.3 sono riportati i risultati dell'ottimizzazione.

SOC	***OCV***	***R0***	***R1***	***Tau1***
100	352.8V	0.129Ω	0.018Ω	299.23s
97	348.18 V	0.129Ω	0.018Ω	299.23s
95	340.0V	0.129Ω	0.018Ω	299.23s
90	336.42 V	0.129Ω	0.018Ω	299.23s
85	335.16 V	0.129Ω	0.018Ω	299.23s
80	333.9 V	0.129Ω	0.018Ω	299.23s
77	329.5 V	0.1652Ω	0.009Ω	393.58s
70	322.9 V	0.1784Ω	0.009Ω	313.94s
66	317.9 V	0.1532Ω	0.009Ω	299.23s
62	314.8 V	0.1716Ω	0.018Ω	294.63s
55	308.7 V	0.094Ω	0.0648Ω	217.94s
50	304.9 V	0.1109Ω	0.008Ω	299.23s
45	301.5 V	0.0896Ω	0.008Ω	1890.1s
38	299.2 V	0.0916Ω	0.0521Ω	164.44 s
33	298.1 V	0.0922Ω	0.010Ω	213.05s
28	295.2 V	0.0828Ω	0.009Ω	299.23s
25	293.5 V	0.129Ω	0.018Ω	299.23s
20	290.9 V	0.129Ω	0.018Ω	299.23s
15	288.27 V	0.129Ω	0.018Ω	299.23s
10	286.4 V	0.129Ω	0.018Ω	299.23s
5	280.7 V	0.129Ω	0.018Ω	299.23s
0	278.2 V	0.129Ω	0.018Ω	299.23s

Tabella 6.3 Parametri ottimizzati

6.2 Modello con Filtro di Kalman

I dati ottenuti dal precedente fitting sono stati inseriti nel modello mostrato in figura 5.2 per valutare le prestazioni del filtro. Inizialmente è stata applicata al filtro la tensione del modello simulato della batteria, successivamente sono stati valutati i risultati con l'utilizzo della tensione proveniente dalle misure.

Il primo test è servito da base per verificarne il funzionamento con i parametri stimati, mentre il secondo test ha valutato la risposta ad una situazione il più possibile fedele all'ambiente in cui verrà utilizzato.

In questo passaggio è stato possibile, inoltre, adattare i parametri sul rumore di misura delle grandezze utilizzate dal filtro.

Oltre alla differenza nelle stime dello stato di carica tra il filtro di Kalman e il Coulomb Counting calcolato dal modello Simulink della batteria, verrà analizzato l'andamento della tensione stima dal filtro di Kalman tramite il modello di batteria interno al filtro e confrontato con l'andamento della tensione applicata.

6.2.1 Risultati dei test sul filtro di Kalman

La figura 6.4 mostra le stime dello stato di carica acquisite tramite tre diversi metodi: Coulomb Counting del BMS della moto, Coulomb Counting della Batteria Simulata e Filtro di Kalman.

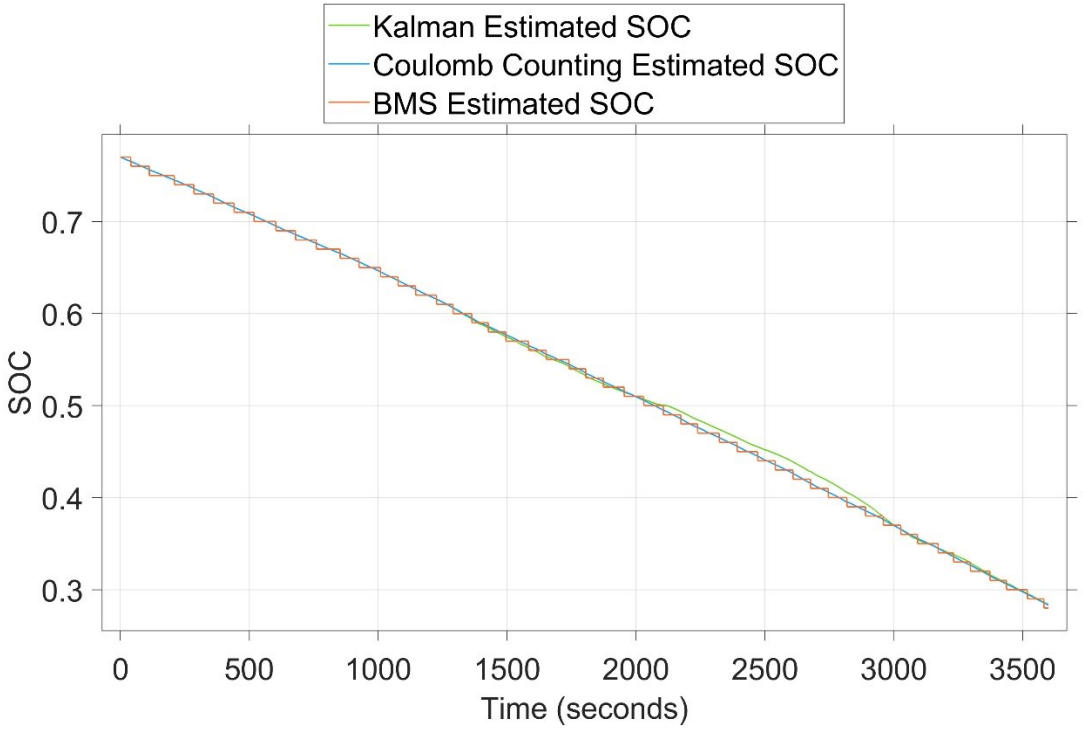

Figura 6.4 Stima del SOC con filtro di Kalman collegato alla tensione della batteria simulata

L'immagine mostra che non ci sono grandi differenze nell'andamento della stima nei tre metodi: applicando al filtro la tensione della batteria simulata, la stima riesce a seguire fedelmente l'andamento calcolato dal modello.

Lo stesso setup ma valutato con la tensione acquisita dai dati sperimentali è riportato in figura 6.5; a differenza del caso precedente viene commesso dal filtro un errore leggermente maggiore poiché esistono differenze tra il modello matematico e il comportamento reale della batteria.

Nella figura 6.6 è riportato l'andamento dell'errore commesso dal filtro di Kalman calcolato come valore assoluto;

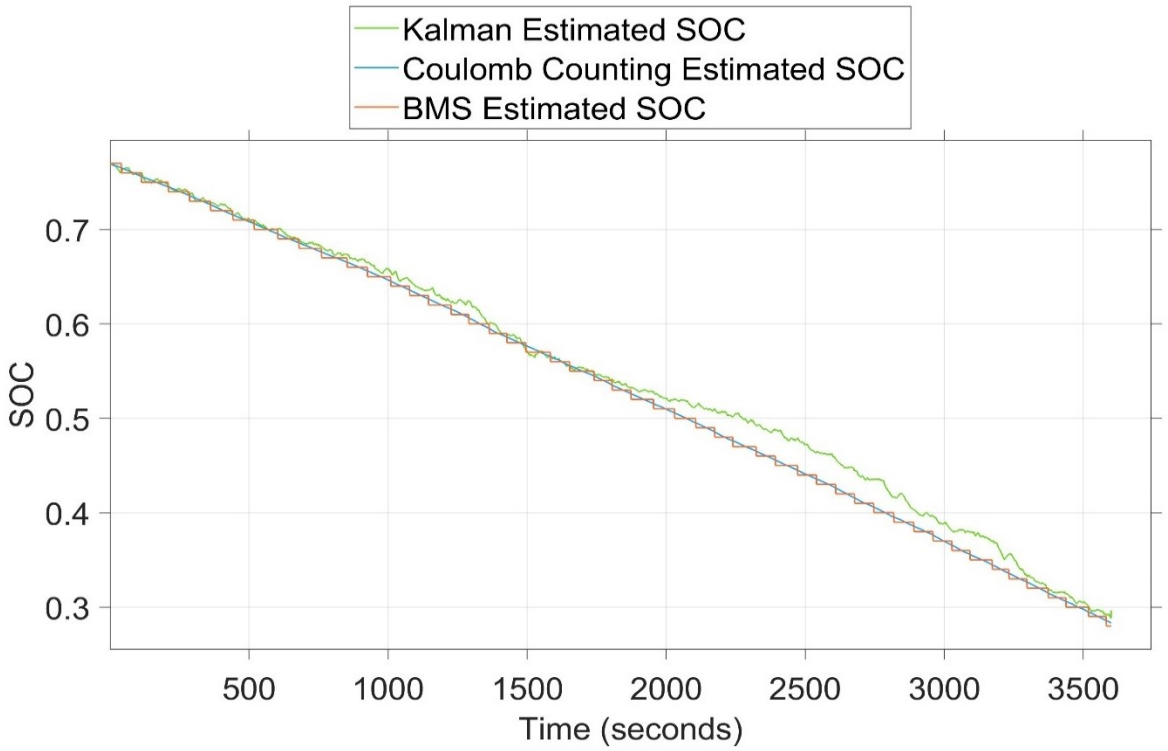

Figura 6.5 Stima del SOC con filtro di Kalman collegato alla tensione sperimentale

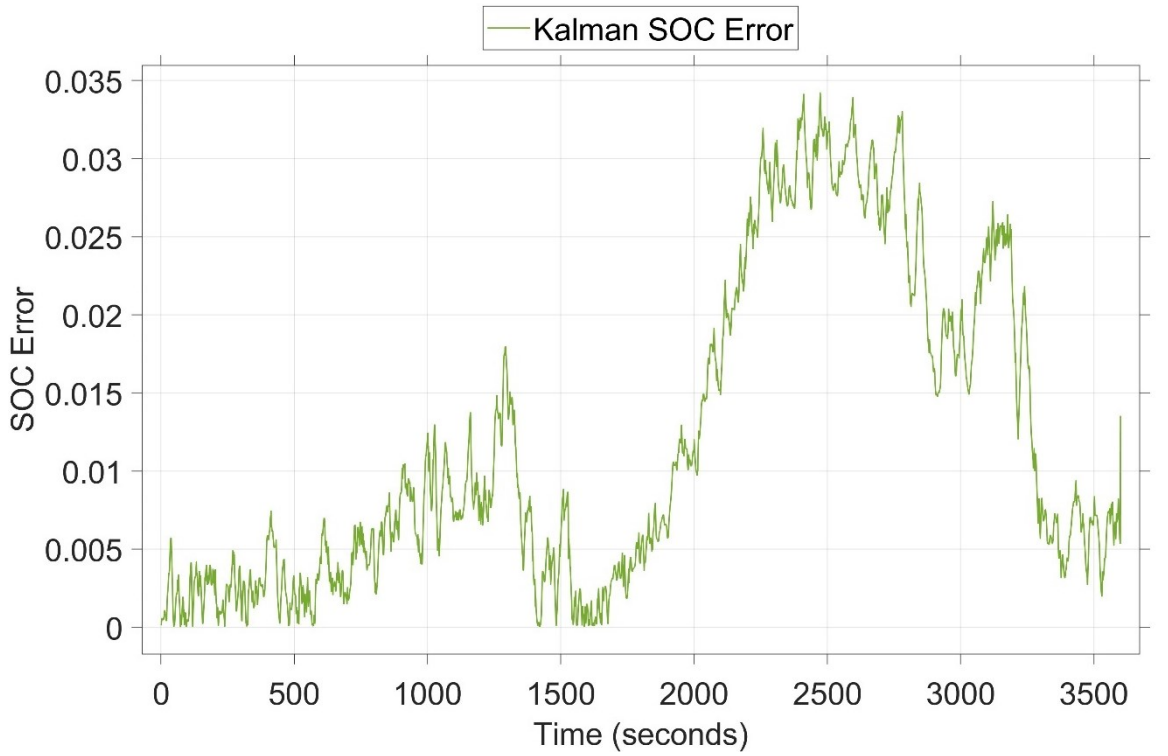

Figura 6.6 Errore assoluto commesso dal filtro di Kalman

il confronto è stato effettuato con la misura data dal SOC della batteria simulata, poiché i dati acquisiti dal BMS della moto hanno un andamento a gradini, mentre la curva descritta dalla simulazione è continua e coincide con i risultati misurati.

L'errore commesso non supera durante la simulazione il 3,5% e comunque riesce a correggersi tornando a livelli più contenuti.

Come ultimo approfondimento è stato valutato l'andamento della tensione stimata all'interno del filtro di Kalman dal modello.

In figura 6.7 è riportato il confronto fra la tensione misurata dalla batteria reale e la tensione stimata dal filtro di Kalman per la batteria.

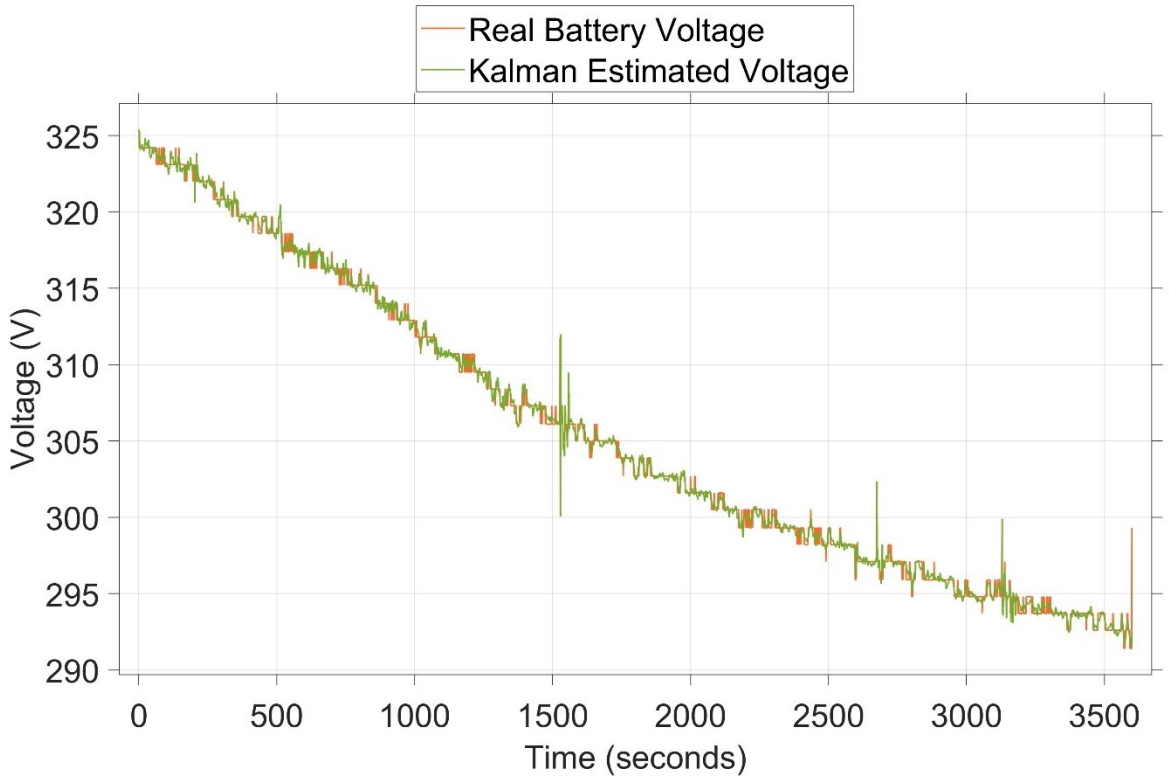

Figura 6.7 Confronto fra tensione sperimentale e tensione stimata dal filtro di Kalman

La curva di tensione stimata dal filtro non differisce dal riferimento sperimentale e riesce a seguire bene le varie oscillazioni e picchi.

7 Esportazione del codice su Scheda

L'ultimo passo fatto per testare il funzionamento del filtro progettato è stato quello di inserire il codice C generato tramite Simulink in una scheda programmabile.

Attraverso il Software proprietario della scheda è stato possibile completare le operazioni di migrazione e di debug e di eseguire dei semplici test per verificare il corretto trasferimento e fare alcune verifiche sul funzionamento prima dei test su campo.

7.1 STM32CubeIDE

STM32CubeIDE è uno strumento di sviluppo multi-piattaforma, che fa parte dell'ecosistema software STM32Cube.

STM32CubeIDE è una piattaforma di sviluppo C/C++ avanzata con funzionalità di configurazione delle periferiche, generazione di codice, compilazione e debug per microcontrollori e microprocessori STM32. Si basa sul framework Eclipse®/CDT™ e sulla toolchain GCC per lo sviluppo e GDB per il debug. Permette l'integrazione delle centinaia di plugin esistenti che completano le funzionalità dell'IDE Eclipse®.

STM32CubeIDE integra le funzionalità di configurazione e creazione del progetto STM32 di STM32CubeMX per offrire un'esperienza di strumenti

“tutto in uno” e risparmiare tempo di installazione e sviluppo. Dopo la selezione di un MCU o MPU STM32 vuoto, o di un microcontrollore o di un microprocessore preconfigurato dalla selezione di una scheda o dalla selezione di un esempio, il progetto viene creato e il codice di inizializzazione generato. In qualsiasi momento durante lo sviluppo, l'utente può tornare all'inizializzazione e alla configurazione delle periferiche o del middleware e rigenerare il codice di inizializzazione senza alcun impatto sul codice utente. STM32CubeIDE include analizzatori di build e stack che forniscono all'utente informazioni utili sullo stato del progetto e sui requisiti di memoria.

7.2 Scheda STM32F407VGT6

La famiglia STM32F407xx si basa sul core RISC a 32 bit Arm® Cortex-M4® ad alte prestazioni che opera a una frequenza fino a 168 MHz. Il core Cortex-M4 è dotato di un'unità a virgola mobile (FPU) a precisione singola che supporta tutte le istruzioni e i tipi di dati di elaborazione e dati a precisione singola Arm. Implementa inoltre un set completo di istruzioni DSP e un'unità di protezione della memoria (MPU) che migliora la sicurezza delle applicazioni.

La famiglia STM32F407xx incorpora memorie embedded ad alta velocità (memoria Flash fino a 1 Mbyte e fino a 192 Kbyte di SRAM), fino a 4 Kbyte

di SRAM di backup e una vasta gamma di I/O avanzati e periferiche collegate a due bus APB, tre bus AHB e una matrice di bus multi-AHB a 32 bit.

Tutti i dispositivi offrono tre ADC a 12 bit, due DAC, un RTC a bassa potenza, dodici timer generici a 16 bit tra cui due timer PWM per il controllo del motore, due timer a 32 bit per uso generale e un generatore di numeri casuali (RNG). Sono inoltre dotati di interfacce di comunicazione standard e avanzate.

7.3 Codice linguaggio C

Il codice inserito nella scheda STM32F407VGT6, si può concettualmente dividere in due sezioni che integrate fra loro andranno a comporre il risultato finale.

La prima parte riguarda l'esportazione del codice sorgente dal modello Simulink; i file generati contengono tutte le librerie necessarie al filtro per essere eseguito e i parametri definiti per simulare il comportamento della batteria. Il file principale è denominato "Kalman_1RC.c" e contiene tutta la programmazione del filtro.

La seconda parte di codice riguarda la configurazione con l'interfaccia CAN-Bus; per accedere ai dati in tempo reale durante i test è stato necessario introdurre questa parte di codice che rileva direttamente i dati di tensione e corrente misurati dal BMS della moto.

I dati Acquisiti dal CAN-Bus sono elaborati dal filtro, assegnati tramite il codice “mainTask.c” che restituisce anche il risultato della stima da parte del filtro di Kalman.

Per l’esportazione su schermo seriale è stata predisposta anche un’ulteriore funzione, tramite la quale è stato possibile visualizzare l’andamento dello stato di carica in tempo reale e confrontare i risultati con le altre grandezze estrapolate. Il codice permetteva inoltre di salvare i dati acquisiti durante i test; quest’ultimo dettaglio è stato utile per riprodurre in simulazione i risultati ottenuti.

Tutto il codice è gestito dal sistema operativo FreeRTOS dove i task per l’esportazione su seriale e il task per il codice del filtro sono inizializzati.

Il codice finale ha un’occupazione di 158Kb, rendendolo idoneo ad essere inserito nel microcontrollore con una memoria massima di 1MB.

L’esecuzione del codice del filtro di Kalman ha una durata inferiore al tempo designato per la stampa su seriale ogni 100ms.

Il codice generato in questa maniera riesce quindi a soddisfare le caratteristiche per l’inserimento nel microcontrollore scelto.

Nella figura è riportato uno schema a blocchi che spiega l’esecuzione del codice e in Appendice è riportato il codice del filtro utilizzato nel microcontrollore.

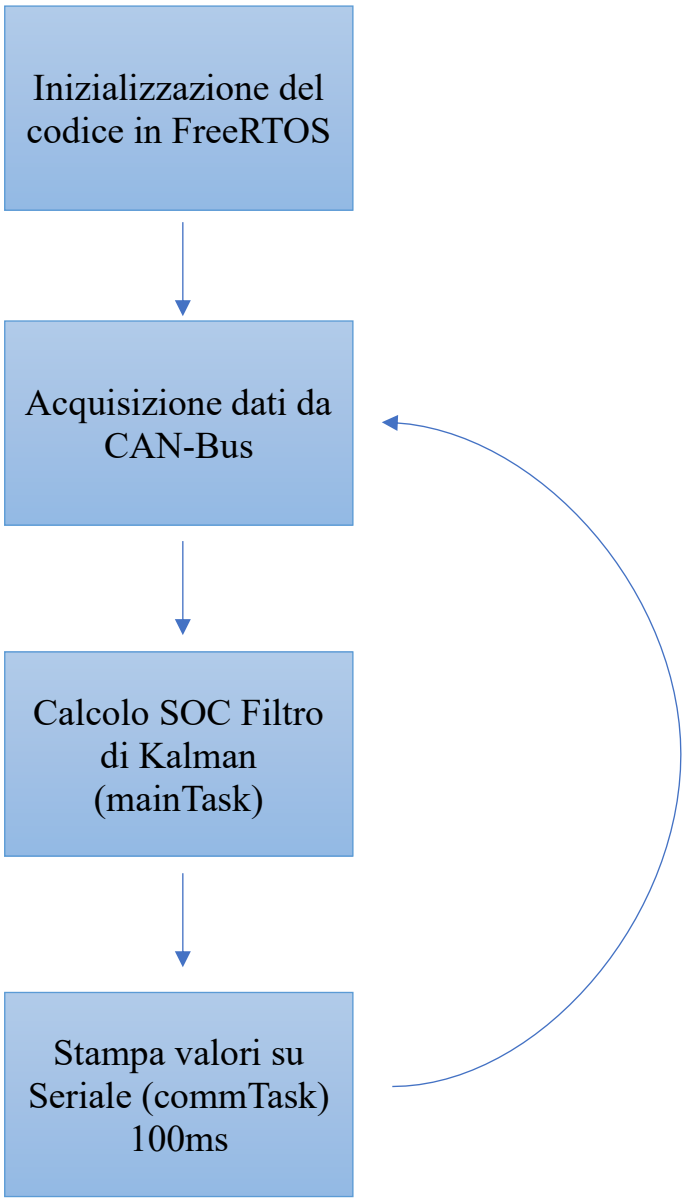

Figura 7.1 Schema a blocchi semplificato dell'esecuzione del codice in linguaggio c

8 Test Eseguiti sul Microcontrollore e Risultati

In questo capitolo verranno descritti i dettagli dei test e delle simulazioni svolte, dopo l'esportazione del codice in linguaggio C all'interno del microcontrollore. Saranno mostrati i test eseguiti sulla moto e verranno confrontati con i risultati ottenuti dal progetto Simulink.

8.1 Esportazione su microcontrollore e test sulla moto

Dopo aver verificato il funzionamento del filtro, è stato predisposto l'ultimo step prima dell'esportazione del codice su microcontrollore.

Il modello di figura 5.3 racchiude lo stesso filtro di Kalman sviluppato negli step precedenti al quale sono stati modificati i tipi delle variabili di ingresso e di uscita per essere compatibile ai dati acquisiti dal BMS della moto.

Questo modello Simulink è stato utilizzato per esportare il codice sviluppato e successivamente per verificare i dati acquisiti durante le simulazioni sulla moto e fare dei paragoni con il filtro inserito nel microcontrollore.

Il codice del filtro è stato convertito in linguaggio C ed inserito nel microcontrollore dove tramite un collegamento OBD è stato possibile far funzionare il filtro in tempo reale.

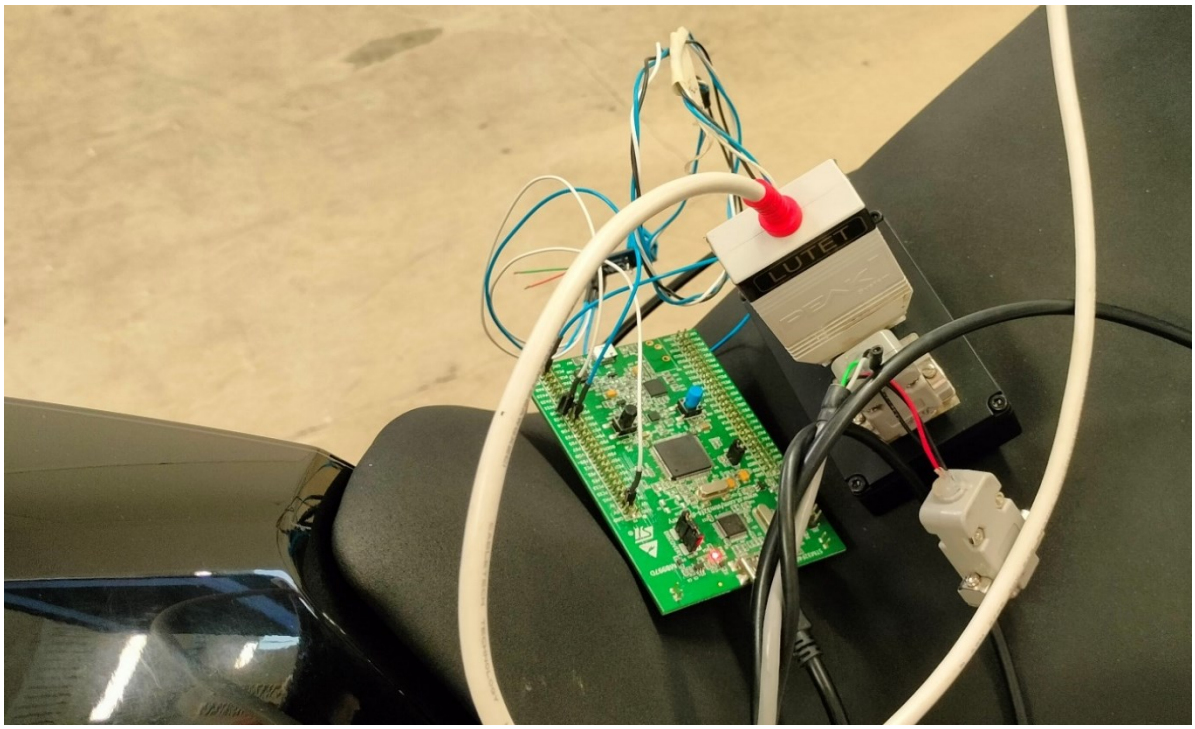

Figura 8.1 Microcontrollore collegato all'OBD della moto per i test del filtro

In figura 8.1 è mostrato il controllore collegato alla moto durante uno dei test effettuati; dal BMS venivano acquisiti e visualizzati in tempo reale i dati di tensione, corrente, SOC stimato dal BMS della moto e SOC stimato dal filtro di Kalman.

La moto durante i test era posizionata su un banco di prova atto a simulare la resistenza della strada sulla gomma posteriore, la ruota anteriore era bloccata e la moto asscurata al banco in modo che non potesse uscire dal dispositivo di test.

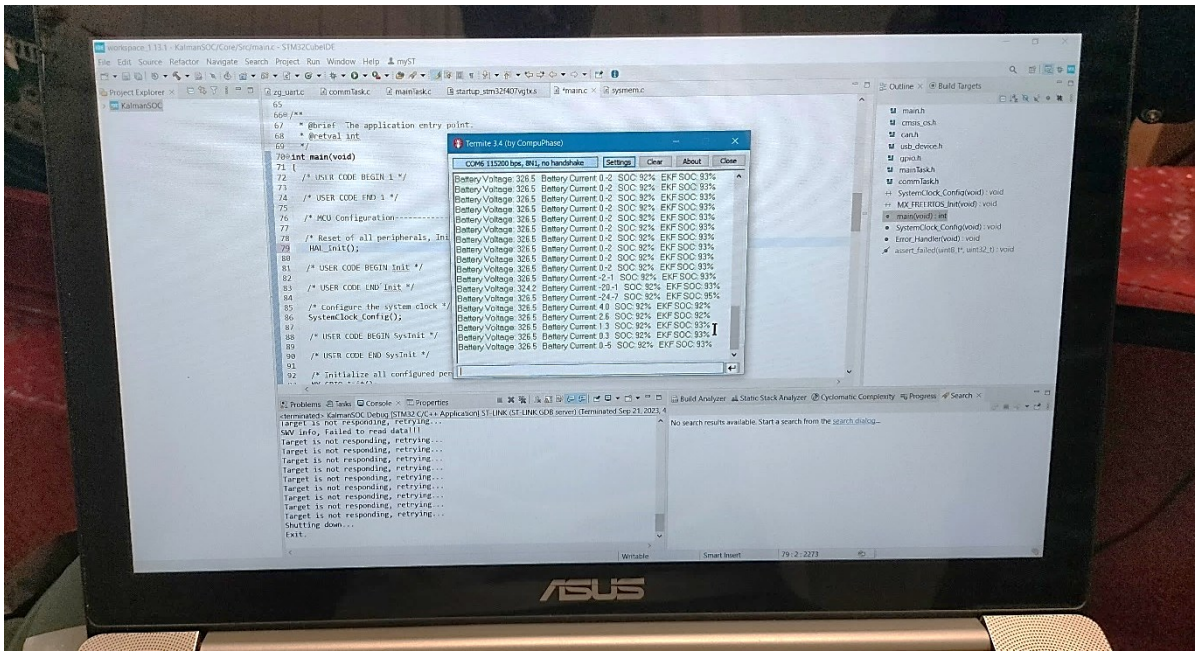

Figura 8.2 Visualizzazione dei dati in tempo reale

Le prove effettuati miravano a simulare diverse condizioni per la moto con caratteristiche più o meno stressanti per i componenti; sono state effettuate in ordine non definito brusche accelerate seguite da tempi di rilassamento graduali, brusche accelerate seguite da brusche interruzioni, accelerate graduali e rilasci bruschi dell'acceleratore, accelerate graduali e rilasci graduali e acelerazioni sotto sforzo (che simulino eventuali cambi di pendio nel percorso); ognuno di questi comportamenti veniva interrotto da tempi più o meno lunghi di rilassamento della batteria.

La batteria presente nella moto aveva una costruzione differente rispetto a quella usata per le ottimizzazioni, la struttura era composta di 81 collegamenti in serie di due celle in parallelo.

I dati inseriti nel microcontrollore sono gli stessi dell'ottimizzazione ma le tensioni sono state scalate; in tabella 8.1 sono riportati i valori.

SOC	***OCV***	***R0***	***R1***	***Tau1***
100	340.2V	0.129Ω	0.018Ω	299.23s
97	329.95V	0.129Ω	0.018Ω	299.23s
95	327.66V	0.129Ω	0.018Ω	299.23s
90	324.4V	0.129Ω	0.018Ω	299.23s
85	323.19V	0.129Ω	0.018Ω	299.23s
80	321.58V	0.129Ω	0.018Ω	299.23s
77	316.18V	0.1652Ω	0.009Ω	393.58s
70	311.12 V	0.1784Ω	0.009Ω	313.94s
66	307.0V	0.1532Ω	0.009Ω	299.23s
62	304.66V	0.1716Ω	0.018Ω	294.63s
55	297.67V	0.094Ω	0.0648Ω	217.94s
50	293.52V	0.1109Ω	0.008Ω	299.23s
45	290.73V	0.0896Ω	0.008Ω	1890.1s
38	288.51V	0.0916Ω	0.0521Ω	164.44 s
33	287.45V	0.0922Ω	0.010Ω	213.05s
28	284.65V	0.0828Ω	0.009Ω	299.23s
25	283.0V	0.129Ω	0.018Ω	299.23s
20	280.51V	0.129Ω	0.018Ω	299.23s
15	277.97V	0.129Ω	0.018Ω	299.23s
10	276.17V	0.129Ω	0.018Ω	299.23s
5	270.67V	0.129Ω	0.018Ω	299.23s
0	268.26V	0.129Ω	0.018Ω	299.23s

Tabella 8.1 Parametri utilizzati nel EKF del Microcontrollore

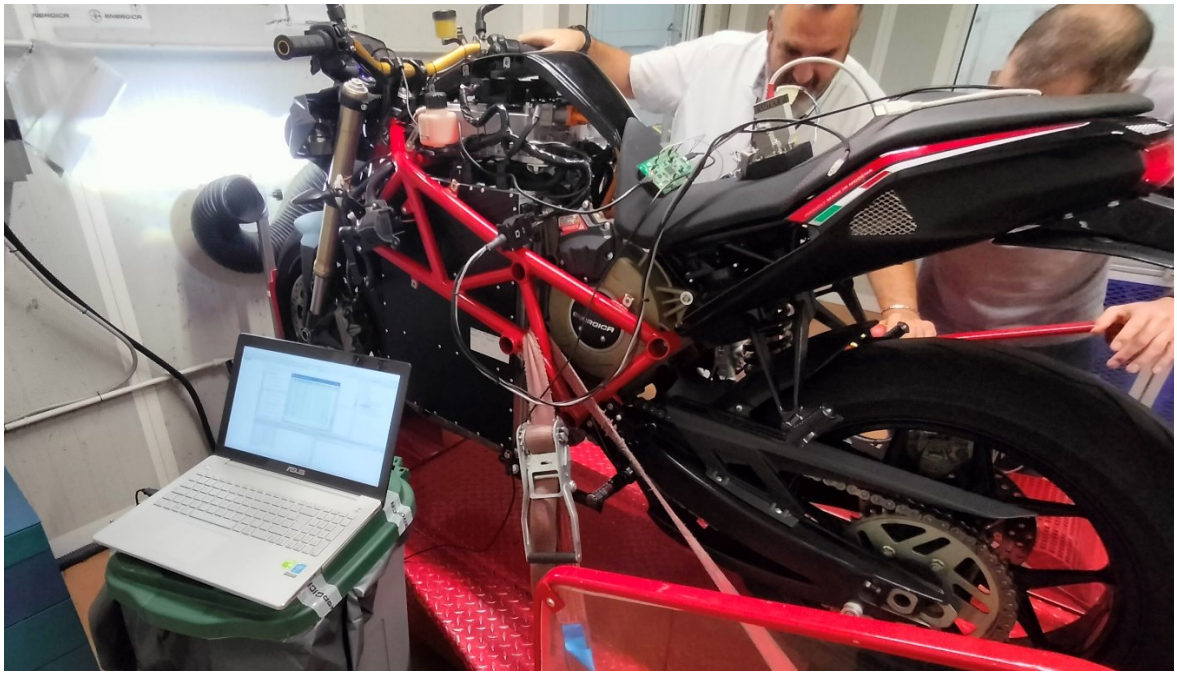

Figura 8.3 Banco di prova dei test effettuati sulla moto

8.2 Risultati dei test e simulazioni successive

Poiché tutti i test sono stati effettuati all'interno di una singola scarica, sono stati raggruppati in un unico segnale e successivamente saranno visualizzati insieme.

Gli andamenti di tensione e corrente acquisiti durante il test con la moto sono riportati in figura 8.4.

Nella figura sono stati riportati in modo da poter mettere a confronto i valori di corrente con i relativi valori in tensione. I dati della tensione sono quantizzati dal BMS, per questo l'andamento della prima curva è a gradini.

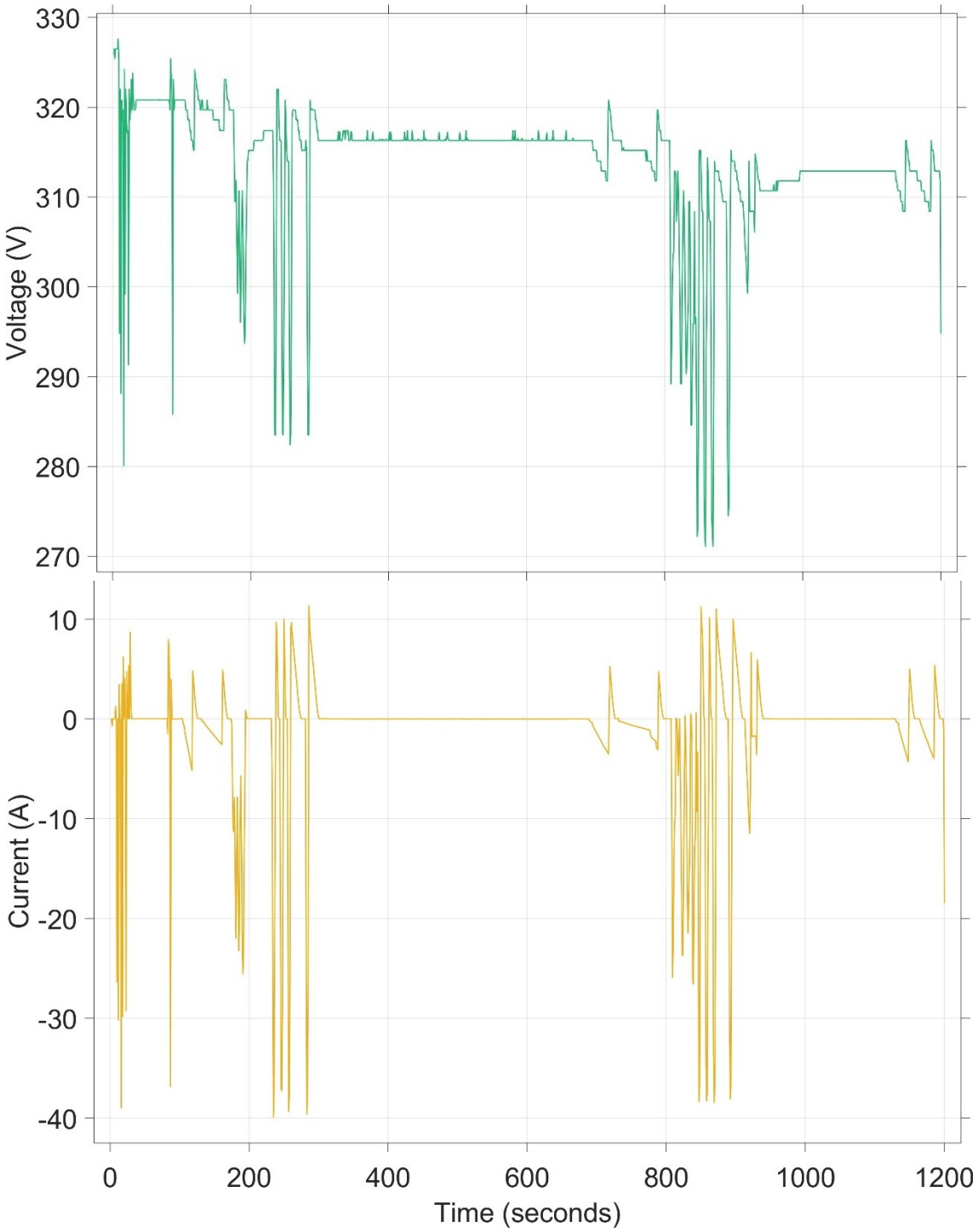

Figura 8.4 Profili di tensione e corrente registrati durante le prove con la moto sul banco

8.2.1 SOC BMS Vs EKF Simulink

La prima valutazione effettuata riguarda il confronto tra il SOC stimato dal BMS della moto (tramite Coulomb Counting) e la stima effettuata dal filtro di Kalman simulato.

In Figura 8.5 sono riportati gli andamenti della tensione inserita nel filtro di Kalman e la tensione acquisita dal test; i due profili sono molto differenti (nonostante l'aggiunta di una costante di correzione).

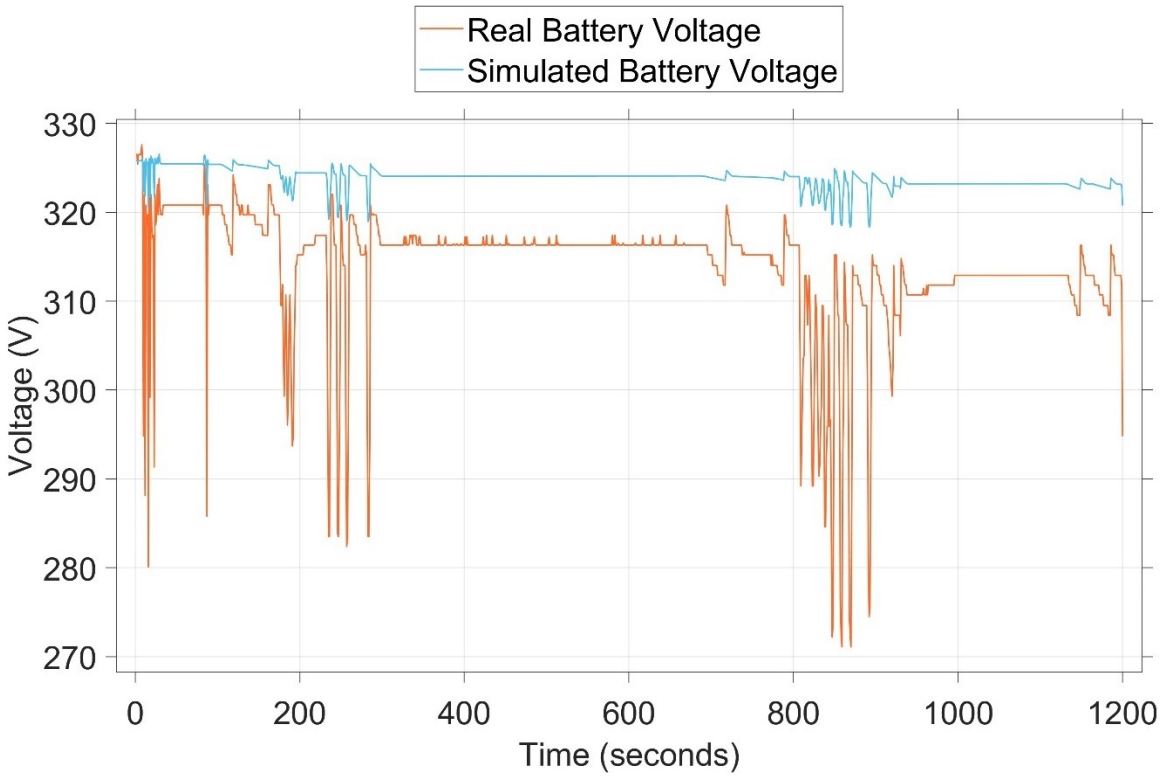

Figura 8.5 Confronto fra curva di tensione inserita nell'EKF Simulink e tensione del test.

In figura 8.6 è invece riportato il confronto delle due stime

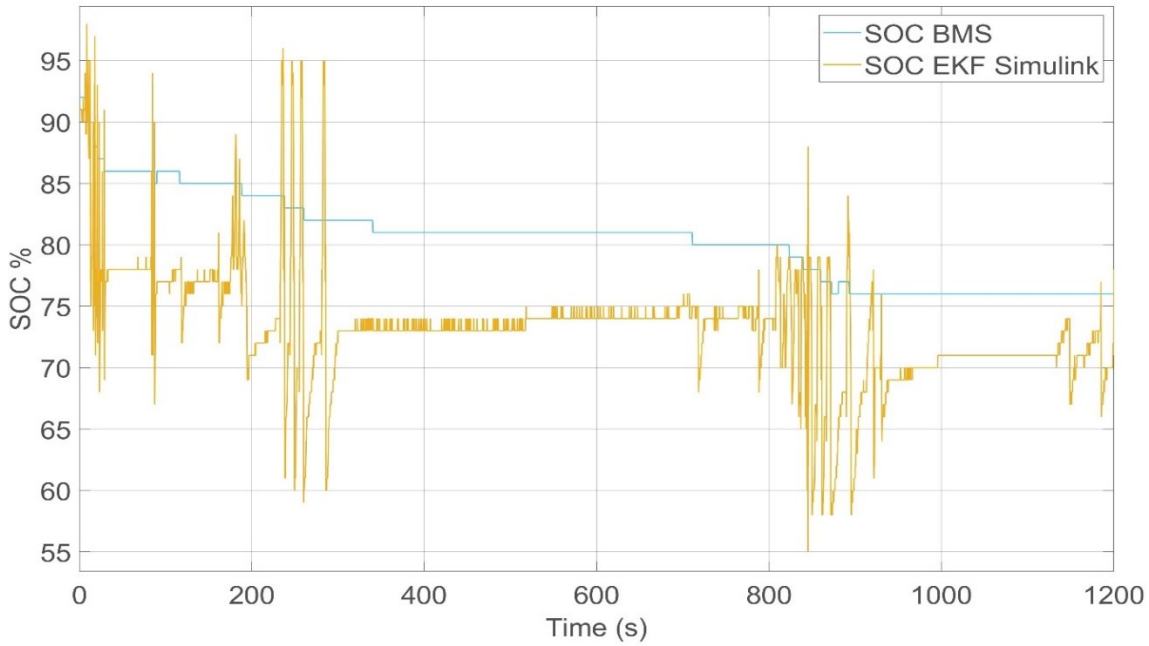

Figura 8.6 Confronto fra SOC stimato dal BMS della moto e filtro di Kalman simulato

Le due curve hanno un aspetto simile nell'andamento ma su livelli differenti.

In figura 8.7 è riportato, il grafico dell'errore commesso dal filtro simulato.

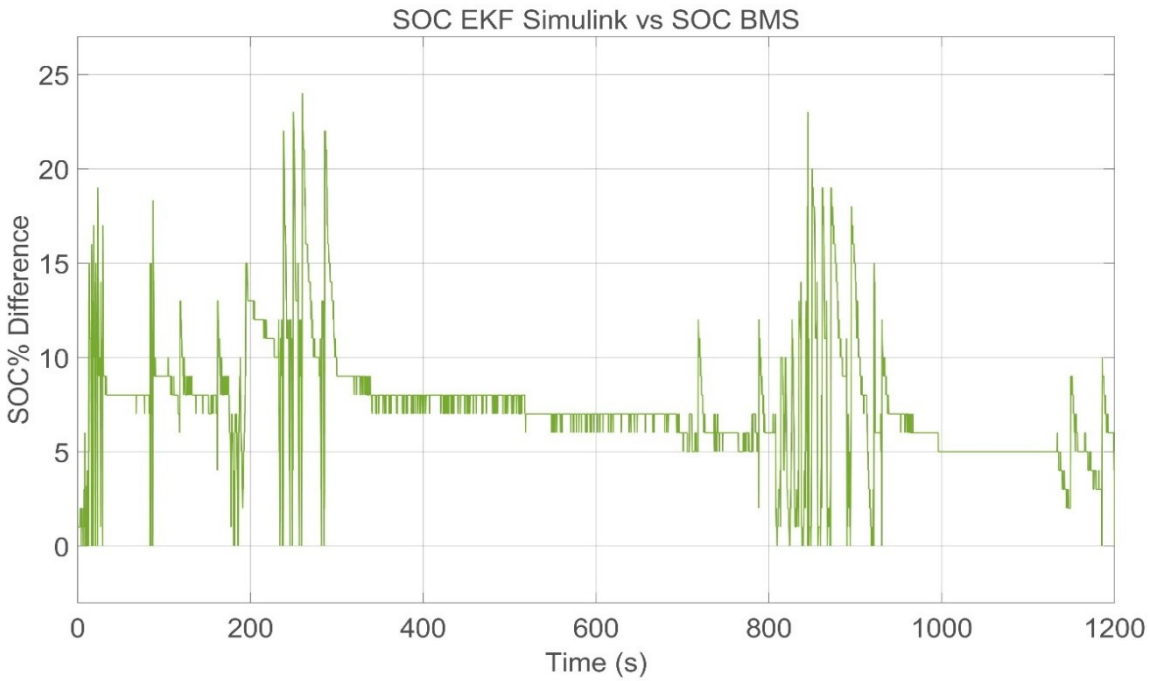

Figura 8.7 Errore percentuale del filtro simulato confrontato con il SOC del BMS

8.2.2 SOC BMS Vs EKF Microcontroller

Questa seconda valutazione è stata effettuata fra il filtro inserito nel microcontrollore e l'algoritmo di Coulomb Counting presente nel BMS della moto.

Le curva di figura 8.8 descrivono l'andamento delle due grandezze: la differenza nella stima rimane simile al caso precedente, nonostante il filtro del microcontrollore riesca a mantenere un andamento simile a quello ottenuto dal Coulomb Counting del BMS.

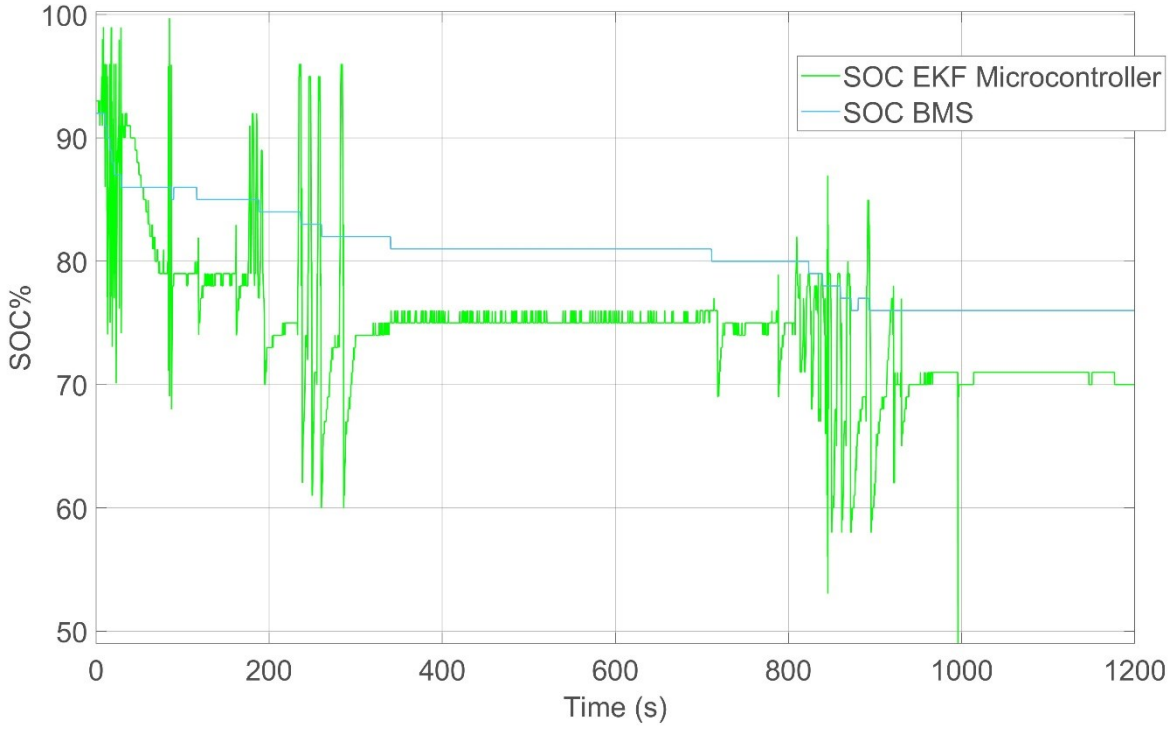

Figura 8.8 Confronto fra SOC stimato dal filtro di Kalman nel microcontrollore e SOC stimato dal BMS

Se si esamina l'errore commesso dal filtro del microcontrollore si può notare una lieve diminuzione rispetto al caso ideale del filtro simulato; la curva di

figura 8.9 mostra la differenza fra la stima del BMS e la stima del microcontrollore con il filtro di Kalman.

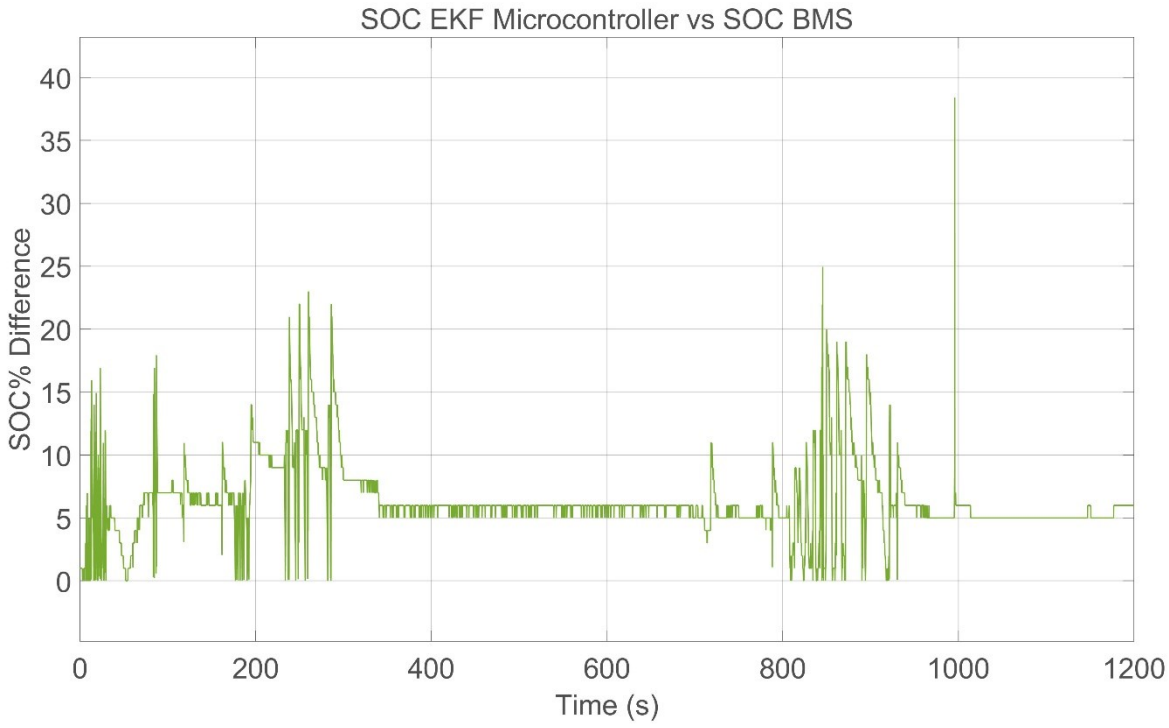

Figura 8.9 Errore percentuale del SOC stimato dal BMS confrontato con il SOC del filtro di Kalman nel microcontrollore

Un altro aspetto che è stato valutato durante la simulazione riguarda come il filtro di Kalman del modello adatti la propria tensione interna a quella estrapolata dalla batteria; come fatto nelle simulazioni del capitolo 6, è stato esaminato l'andamento della tensione stimata da Kalman.

La figura 8.10 riporta il confronto fra le due tensioni: il filtro riesce a correggere molto bene il suo valore, come visto durante la simulazione, arrivando alla piena convergenza dei valori.

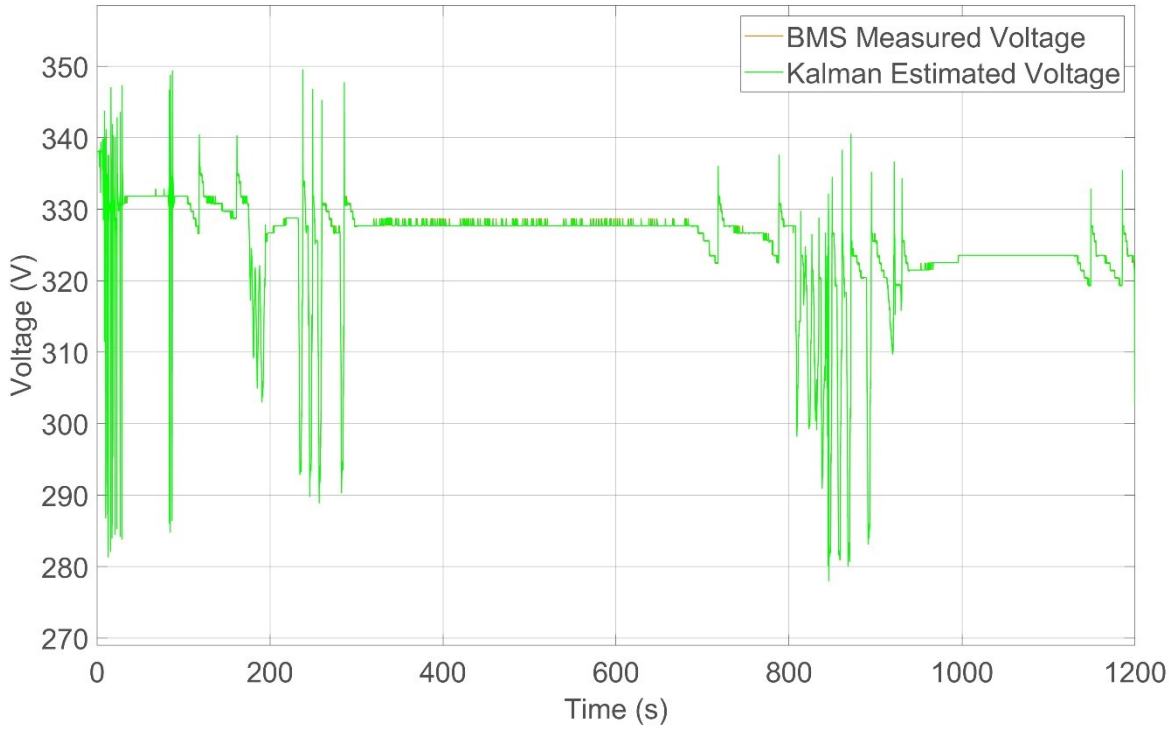

Figura 8.10 Confronto fra tensione stimata da Kalman e tensione misurata

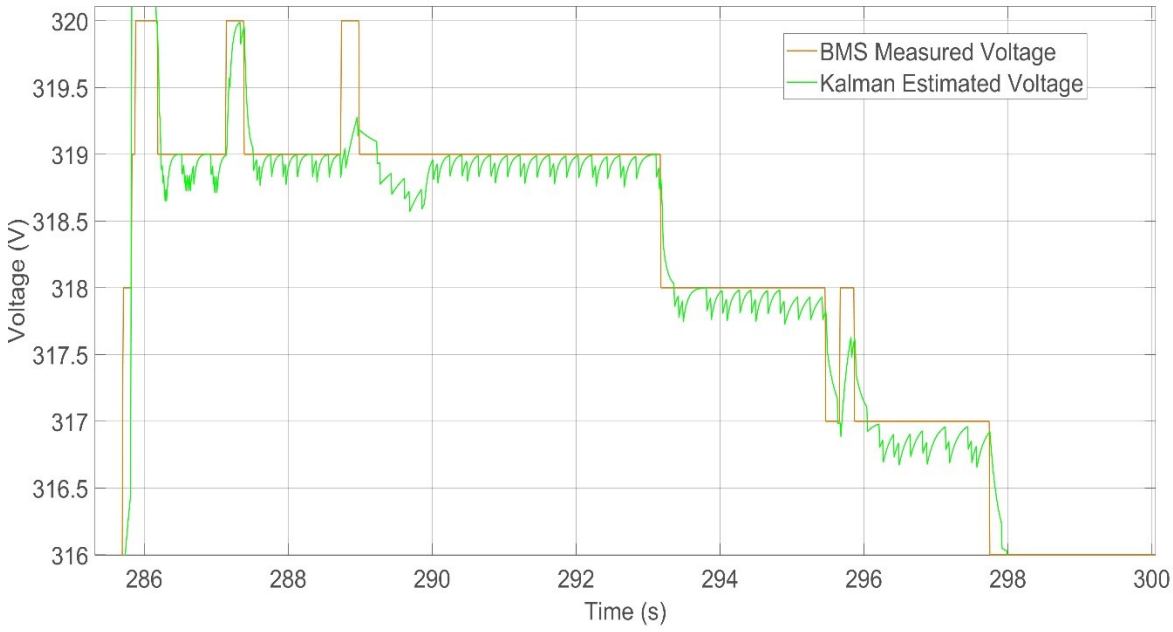

Figura 8.11 Dettaglio del Confronto fra tensione stimata da Kalman e tensione misurata

Nella figura 8.11 vengono riportati gli stessi dati di figura 8.10 ingranditi per evidenziare le correzioni effettuate dal filtro di Kalman e per mostrare come la tensione del modello stimata dall'algoritmo si corregga in funzione del segnale di riferimento.

8.2.3 SOC EKF Simulink Vs EKF Microcontroller

I dati estrapolati dal filtro inserito nel microcontrollore sono stati confrontati con i dati del filtro simulato; la differenza delle due curve nella stima dello stato di carica non risulta particolarmente elevata, testimoniando una buona trasposizione del filtro dall'ambiente di simulazione a quello di test su microcontrollore.

Le differenze sono date dalla trasposizione in codice C delle Lookup Table necessarie per la definizione dei parametri.

Sono da evidenziare i comportamenti del filtro durante i transitori di utilizzo dove erano applicate forti correnti; l'algoritmo ha reagito sottostimando il livello di carica della batteria adattandosi rapidamente al livello di tensione registrato.

Il picco negativo registrato all'istante temporale di 1000s è frutto di un test sulla reattività del filtro qualora venisse forzato con un valore differente da quello stimato.

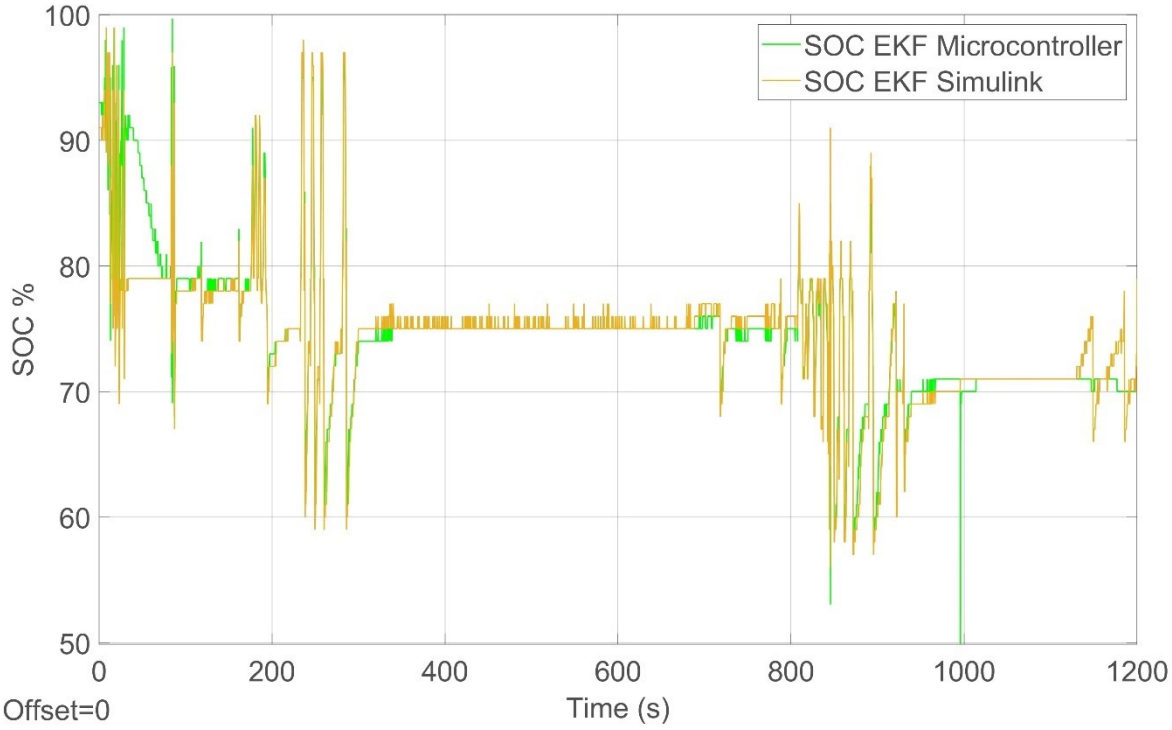

Figura 8.12 Confronto fra SOC stimato dal filtro di Kalman nel microcontrollore e filtro di Kalman simulato

Il rapido recupero dimostra un'eccellente flessibilità dell'algoritmo anche in condizioni fortemente sfavorevoli come l'assegnazione esterna di un valore casuale.

Nella figura 8.13 sono riportati gli andamenti delle due curve OCV presenti nei rispettivi filtri. La curva di partenza è la stessa ma la trasposizione in linguaggio C ha introdotto delle lievi differenze nell'andamento che spiegano la differenza nelle stime.

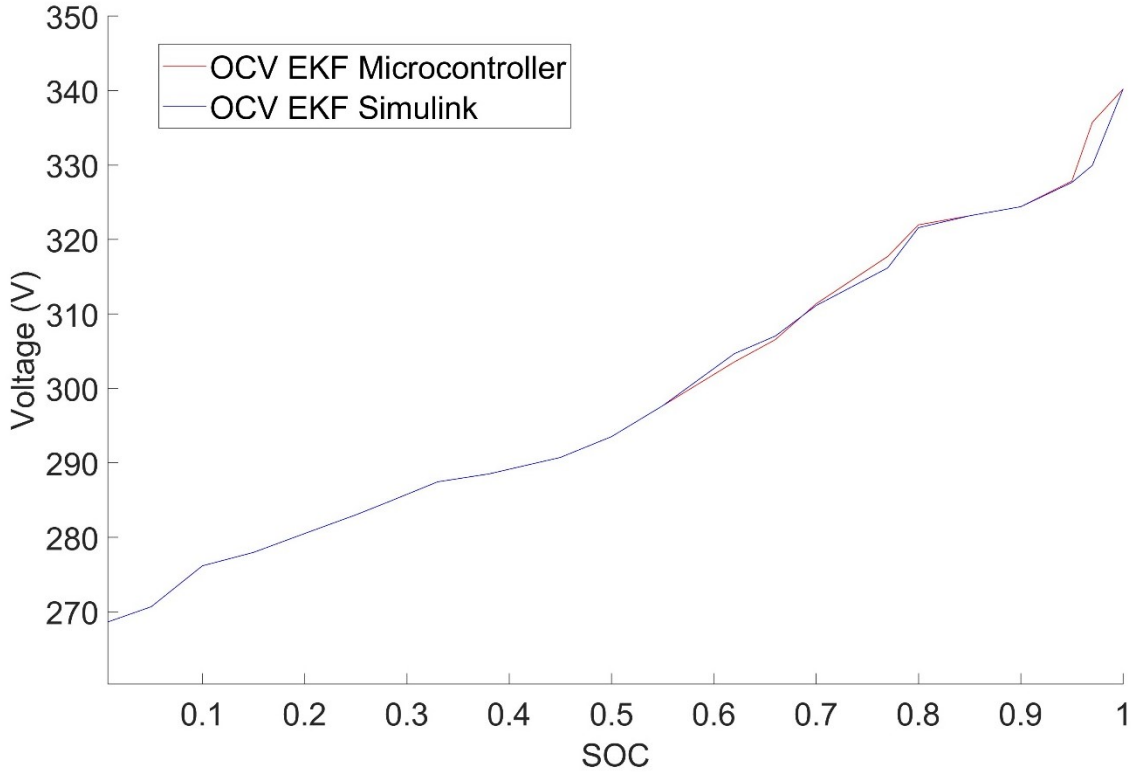

Figura 8.13 Confronto fra la curva OCV del filtro simulato con l'OCV del filtro di Kalman nel microcontrollore

8.2.4 Ottimizzazione finale

Dopo aver valutato le prestazioni del filtro sulla moto ed aver constatato che le curve ottenute dal modello Simulink e quelle prese dal microcontrollore per lo stato di carica sono simili, sono stati ottimizzati i parametri del modello sulle nuove curve di tensione ottenute. La curva di tensione della batteria simulata confrontata con la curva di tensione ottenuta dai dati sperimentali è riportata nella figura 8.14.

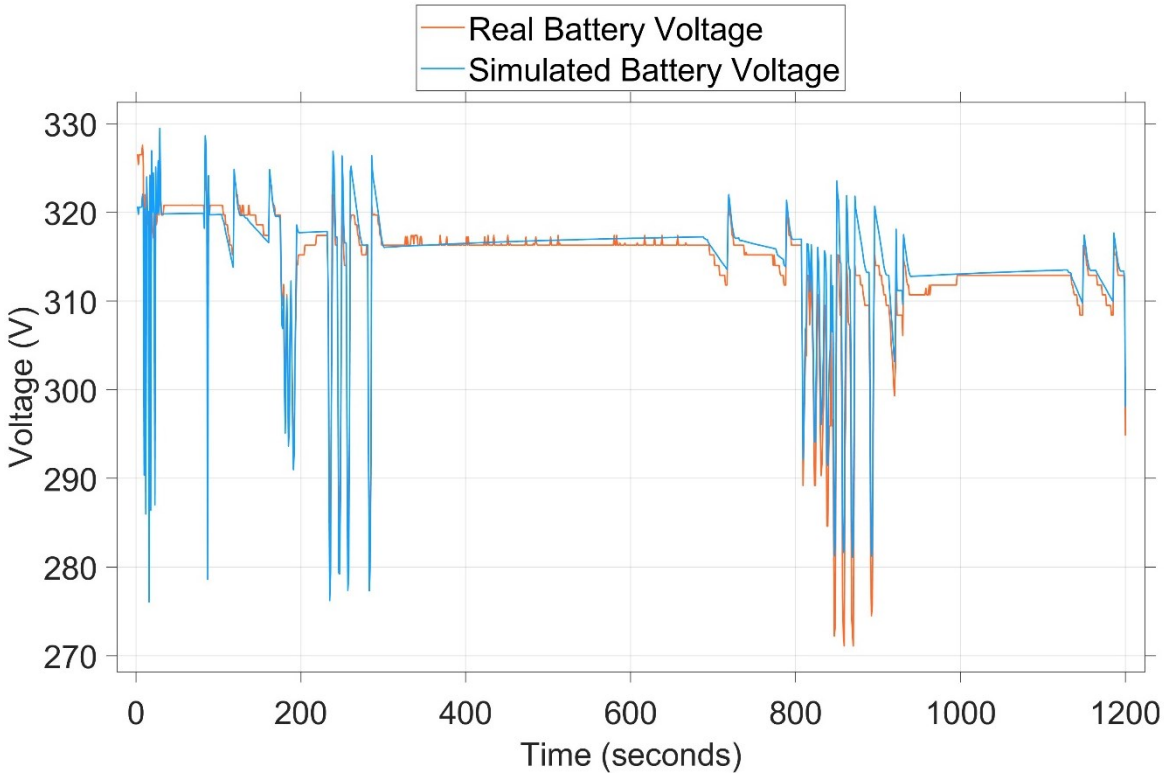

Figura 8.14 Curva di tensione ottimizzata su dati di test

In figura 8.15 è mostrato un ingrandimento della figura 8.14.

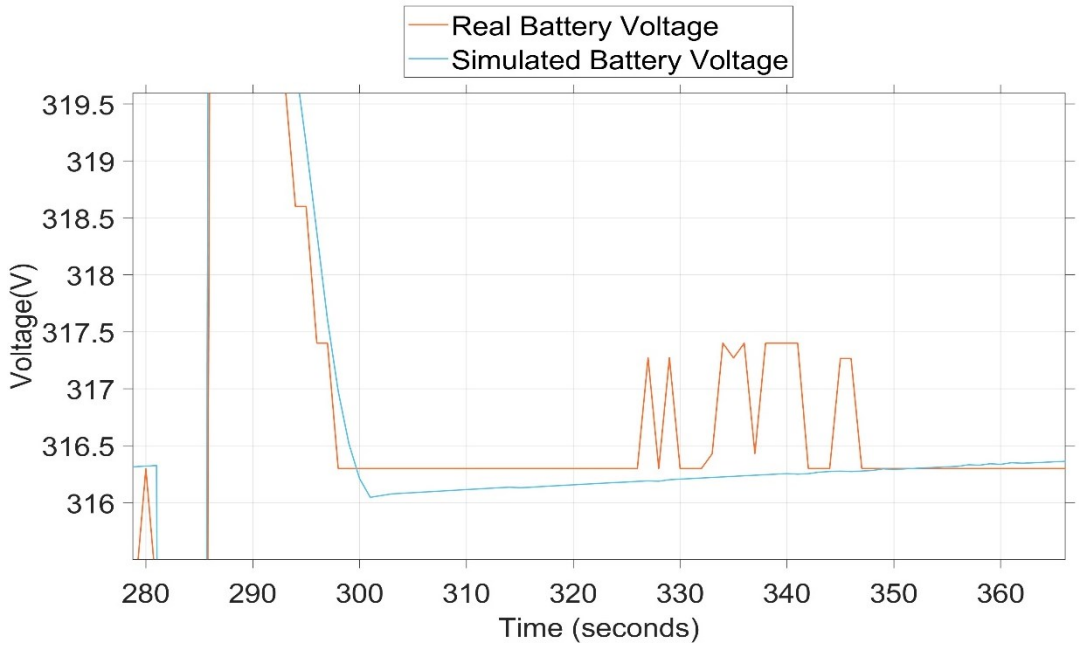

Figura 8.15 Dettaglio della curva di tensione ottimizzata su dati di test

I parametri aggiornati sono poi stati inseriti all'interno dell'EKF Simulink per valutarne i risultati.

La tabella 8.2 riporta i valori ottenuti con l'ottimizzazione fatta sulle ultime curve ottenute.

SOC	***OCV***	***R0***	***R1***	***Tau1***
100	334.0V	1.1457Ω	1.00Ω	299.23s
97	327.0V	1.1457Ω	1.00Ω	299.23s
95	322.09V	1.1457Ω	1.00Ω	299.23s
92	320.58V	0.8362Ω	0.6291Ω	299.23s
86	315.26V	0.8030Ω	0.44Ω	299.23s
82	311.0V	0.8690Ω	0.53Ω	299.23s
76	302.0V	0.7960Ω	0.061Ω	393.58s
72	298.0 V	0.8330Ω	0.063Ω	313.94s
64	297.5V	0.8990Ω	0.463Ω	299.23s
61	297.0V	0.6940Ω	0.463Ω	294.63s
57	296.1V	0.5545Ω	0.463Ω	217.94s
50	294.0V	0.4480Ω	0.521Ω	299.23s
45	291.5V	0.4580Ω	0.1Ω	1890.1s
38	289.0V	0.4610Ω	0.09Ω	164.44 s
33	287.75V	0.4140Ω	0.18Ω	213.05s
28	285.5V	0.6450Ω	0.18Ω	299.23s
25	284.0V	0.6450Ω	0.18Ω	299.23s
20	281.5V	0.6450Ω	0.18Ω	299.23s
15	280.68V	0.6450Ω	0.18Ω	299.23s
10	278.28V	0.6450Ω	0.18Ω	299.23s
5	274.44V	0.6450Ω	0.18Ω	299.23s
0	268V	0.6450Ω	0.18Ω	299.23s

Tabella 8.2 Parametri dell'ottimizzazione su curve di test

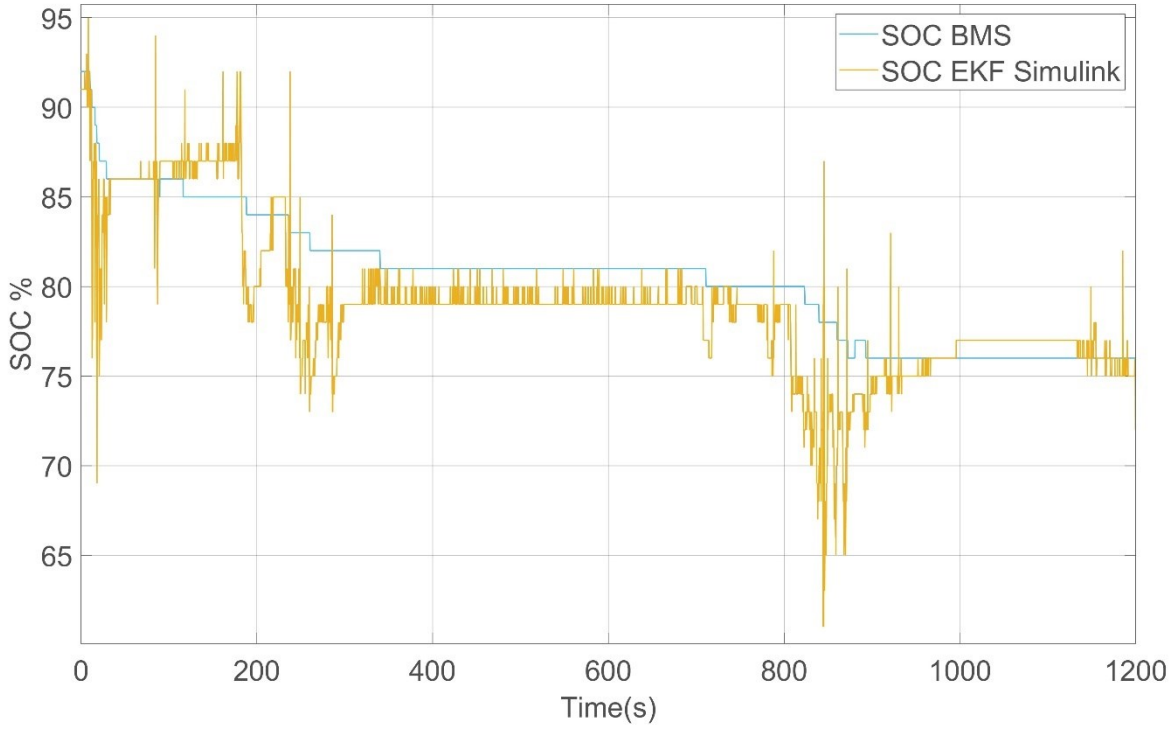

Figura 8.16 Confronto fra SOC estrapolato dal BMS della moto e SOC stimato dall'EKF Simulink

In figura 8.16 sono riportate le curve di confronto tra il SOC stimato dall'EKF Simulink e il SOC stimato dal BMS della moto, registrato durante le prove.

Si può notare come il modello ora abbia un andamento molto più simile a quello stimato tramite Coulomb Counting dal BMS, fatta sempre eccezione per le zone transitorie di forte corrente.

Il grafico sull'errore di figura 8.17 evidenzia che i due metodi stimano quasi lo stesso valore durante la scarica portando l'errore commesso nelle zone di rilassamento della batteria a valori compresi tra l'1% e il 2%.

Questi ultimi risultati dimostrano come l'utilizzo di un modello accurato possa garantire una stima più accurata nei parametri della batteria.

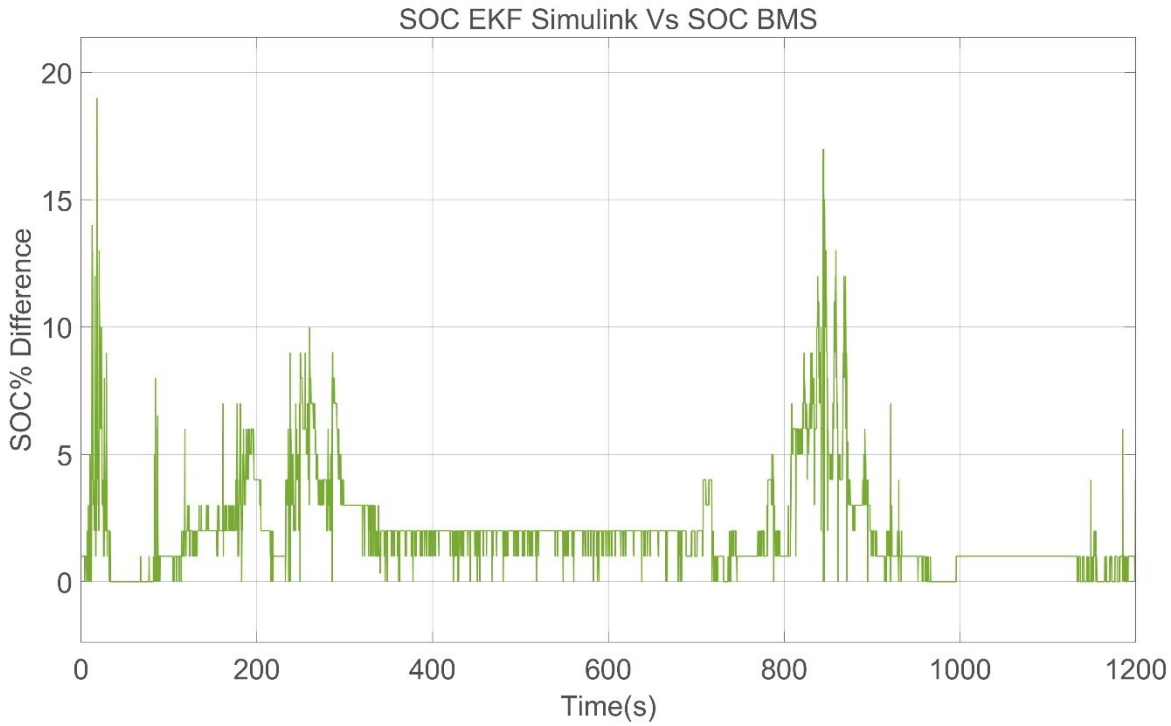

Figura 8.17 Differenza tra il SOC stimato dall'EKF Simulink e dal BMS dopo l'ottimizzazione dei parametri

9 Conclusioni

Il lavoro svolto in questa tesi ha permesso di studiare le problematiche principali presenti nello sviluppo di un sistema di stima della carica di batterie per veicoli elettrici.

La progettazione di un filtro di Kalman esteso, di un modello di Thevenin del primo ordine per la rappresentazione del comportamento della batteria ha evidenziato nei risultati degli esperimenti svolti, come sia necessario utilizzare un modello quanto più accurato per descrivere le dinamiche interne di una batteria agli ioni di litio e come poi questo modello rifletta le sue lacune in una successiva stima dei parametri della batteria.

Nonostante la sintesi di un modello che nelle simulazioni rispetti l'andamento delle tensioni reali di funzionamento del veicolo, appaiono in fase di applicazione all'interno del filtro, altre problematiche legate al recupero della tensione della batteria durante le fasi transitorie di utilizzo che compromettono la stima da parte del filtro di Kalman.

L'esportazione del codice dall'ambiente di simulazione al microcontrollore ha inoltre messo in evidenza come i dati estrapolati dal microcontrollore vadano rielaborati in maniera da renderli utilizzabili per l'inserimento in un BMS e successivamente leggibili da un utente medio nella consultazione dello stato di carica.

Le prestazioni ottenute dal filtro di Kalman, sebbene notevolmente limitate dall'utilizzo di un modello matematico semplice della batteria, confermano l'efficienza dell'algoritmo per gli scopi di stima della carica e non solo.

Ulteriori sviluppi di questo lavoro possono dirigersi verso l'utilizzo di modelli più complessi che replichino il comportamento della batteria anche durante i transitori dovuti alle forti correnti; per ottenere questi dati è necessario stimarli a partire da curve di tensione e corrente con una risoluzione maggiore e che non presentino compressioni dovute ai mezzi di acquisizione, in particolare sarebbe idoneo l'uso di una curva HPPC per la definizione dei vari livelli di carica.

La stima dei parametri può essere inoltre migliorata andando ad indagare anche il degrado della batteria stessa durante i cicli di utilizzo introducendo la misura dello stato di salute (SOH) e la stima dell'aumento della resistenza interna.

La stima dei parametri della batteria rimane ancora un argomento centrale nella ricerca di nuove tecnologie per la mobilità sostenibile in quanto consentirebbe, qualora affinata, una miglior gestione delle risorse presenti nel pianeta e favorendo una progressiva decarbonizzazione delle emissioni e contrastando il progressivo riscaldamento che affligge il pianeta.

Appendice

Codice generato del filtro di Kalman in linguaggio C utilizzato per i test sulla moto.

```
#include "Kalman_1RC.h"
#include <math.h>
#include "rt_nonfinite.h"
#include "rtwtypes.h"
#include <string.h>
#include "look2_binlxpw.h"
#include "Kalman_1RC_private.h"

/* Block signals (default storage) */
BlockIO_Kalman_1RC Kalman_1RC_B;

/* Block states (default storage) */
D_Work_Kalman_1RC Kalman_1RC_DWork;

/* External inputs (root inport signals with default storage) */
ExternalInputs_Kalman_1RC Kalman_1RC_U;

/* External outputs (root outports fed by signals with default
storage) */
ExternalOutputs_Kalman_1RC Kalman_1RC_Y;

/* Real-time model */
static RT_MODEL_Kalman_1RC Kalman_1RC_M_;
RT_MODEL_Kalman_1RC *const Kalman_1RC_M = &Kalman_1RC_M_;

/* Model step function */
void Kalman_1RC_step(void)
{
  real_T tmp[4];
  real_T tmp_0[4];
  real_T v1[4];
  real_T Product2_l;
  real_T Transpose1;
  real_T tmp_1;
  real_T tmp_2;
  real_T v1_0;
  real_T v1_idx_0;
  real_T v1_idx_1;
  int32_T i;
  int32_T i_0;

  /* Sum: '<S14>/Sum1' incorporates:
   *  Constant: '<S14>/Time constant'
```

```
   */
  Kalman_1RC_B.Sum1 = 200.0 - Kalman_1RC_B.Probe[0];

  /* RelationalOperator: '<S18>/Compare' incorporates:
   *  Constant: '<S18>/Constant'
   */
  Kalman_1RC_B.Compare = (Kalman_1RC_B.Sum1 <= 0.0);

  /* Logic: '<S14>/Logical Operator' */
  Kalman_1RC_B.LogicalOperator = (Kalman_1RC_B.Compare &&
    Kalman_1RC_ConstB.Compare);

  /* Product: '<S1>/Divide1' incorporates:
   *  Constant: '<S1>/Constant1'
   *  Inport: '<Root>/InitialSoC_In'
   */
  Kalman_1RC_B.Divide1 = Kalman_1RC_U.InitSoc / 100.0;

  /* SignalConversion generated from: '<S4>/Delay' */
  Kalman_1RC_B.TmpSignalConversionAtDelayInpor[0] =
Kalman_1RC_B.Divide1;
  Kalman_1RC_B.TmpSignalConversionAtDelayInpor[1] =
    Kalman_1RC_ConstB.Conversion_de;

  /* Delay: '<S4>/Delay' */
  if (Kalman_1RC_DWork.icLoad) {
    Kalman_1RC_DWork.Delay_DSTATE[0] =
      Kalman_1RC_B.TmpSignalConversionAtDelayInpor[0];
    Kalman_1RC_DWork.Delay_DSTATE[1] =
      Kalman_1RC_B.TmpSignalConversionAtDelayInpor[1];
  }

  /* Delay: '<S4>/Delay' */
  Kalman_1RC_B.Delay[0] = Kalman_1RC_DWork.Delay_DSTATE[0];
  Kalman_1RC_B.Delay[1] = Kalman_1RC_DWork.Delay_DSTATE[1];

  /* DataTypeConversion: '<S1>/Data Type Conversion2'
incorporates:
   *  Inport: '<Root>/CellTemperature_In'
   */
  Kalman_1RC_B.DataTypeConversion2 = Kalman_1RC_U.BattTemp;

  /* Lookup_n-D: '<S5>/2-D Lookup Table R1' incorporates:
   *  DataTypeConversion: '<S1>/Data Type Conversion2'
   */
  Kalman_1RC_B.R1 = look2_binlxpw(Kalman_1RC_B.Delay[0],
    Kalman_1RC_B.DataTypeConversion2,
rtCP_uDLookupTableR1_bp01Data,
    rtCP_uDLookupTableR1_bp02Data, rtCP_uDLookupTableR1_tableData,
    rtCP_uDLookupTableR1_maxIndex, 24U);

  /* Lookup_n-D: '<S5>/2-D Lookup Table C1' incorporates:
```

```
   *  DataTypeConversion: '<S1>/Data Type Conversion2'
   */
  Kalman_1RC_B.C1 = look2_binlxpw(Kalman_1RC_B.Delay[0],
    Kalman_1RC_B.DataTypeConversion2,
rtCP_uDLookupTableC1_bp01Data,
    rtCP_uDLookupTableC1_bp02Data, rtCP_uDLookupTableC1_tableData,
    rtCP_uDLookupTableC1_maxIndex, 24U);

  /* Product: '<S5>/Divide' */
  Kalman_1RC_B.Divide = Kalman_1RC_B.R1 * Kalman_1RC_B.C1;

  /* Product: '<S5>/Product2' */
  Kalman_1RC_B.Product2 = 1.0 / Kalman_1RC_B.Divide *
Kalman_1RC_ConstB.Ts;

  /* Math: '<S5>/Math Function'
   *
   * About '<S5>/Math Function':
   *  Operator: exp
   */
  Kalman_1RC_B.MathFunction = exp(Kalman_1RC_B.Product2);

  /* Assignment: '<S5>/Assignment' incorporates:
   *  DataTypeConversion: '<S11>/Conversion'
   */
  Kalman_1RC_B.Assignment[0] = Kalman_1RC_ConstB.Conversion_di[0];

  /* UnitDelay: '<S2>/Unit Delay - P' */
  Kalman_1RC_B.UnitDelayP[0] =
Kalman_1RC_DWork.UnitDelayP_DSTATE[0];

  /* Assignment: '<S5>/Assignment' incorporates:
   *  DataTypeConversion: '<S11>/Conversion'
   */
  Kalman_1RC_B.Assignment[1] = Kalman_1RC_ConstB.Conversion_di[1];

  /* UnitDelay: '<S2>/Unit Delay - P' */
  Kalman_1RC_B.UnitDelayP[1] =
Kalman_1RC_DWork.UnitDelayP_DSTATE[1];

  /* Assignment: '<S5>/Assignment' incorporates:
   *  DataTypeConversion: '<S11>/Conversion'
   */
  Kalman_1RC_B.Assignment[2] = Kalman_1RC_ConstB.Conversion_di[2];

  /* UnitDelay: '<S2>/Unit Delay - P' */
  Kalman_1RC_B.UnitDelayP[2] =
Kalman_1RC_DWork.UnitDelayP_DSTATE[2];
  Kalman_1RC_B.UnitDelayP[3] =
Kalman_1RC_DWork.UnitDelayP_DSTATE[3];

  /* Assignment: '<S5>/Assignment' */
```

```
Kalman_1RC_B.Assignment[3] = Kalman_1RC_B.MathFunction;

/* Math: '<S7>/Transpose' incorporates:
 *  Assignment: '<S5>/Assignment'
 */
Kalman_1RC_B.Transpose[0] = Kalman_1RC_B.Assignment[0];
Kalman_1RC_B.Transpose[1] = Kalman_1RC_B.Assignment[2];
Kalman_1RC_B.Transpose[2] = Kalman_1RC_B.Assignment[1];
Kalman_1RC_B.Transpose[3] = Kalman_1RC_B.Assignment[3];

/* Product: '<S7>/Product2' incorporates:
 *  Assignment: '<S5>/Assignment'
 *  Math: '<S7>/Transpose'
 *  UnitDelay: '<S2>/Unit Delay - P'
 */
tmp[0] = Kalman_1RC_B.UnitDelayP[0];
tmp_0[0] = Kalman_1RC_B.Transpose[0];
tmp[1] = Kalman_1RC_B.UnitDelayP[1];
tmp_0[1] = Kalman_1RC_B.Transpose[1];
tmp[2] = Kalman_1RC_B.UnitDelayP[2];
tmp_0[2] = Kalman_1RC_B.Transpose[2];
tmp[3] = Kalman_1RC_B.UnitDelayP[3];
tmp_0[3] = Kalman_1RC_B.Transpose[3];
Transpose1 = tmp[0];
tmp_1 = tmp[2];
tmp_2 = tmp[1];
v1_idx_0 = tmp[3];
i_0 = 0;
for (i = 0; i < 2; i++) {
  Product2_l = tmp_0[i_0];
  v1_idx_1 = Product2_l * Transpose1;
  v1_0 = tmp_0[i_0 + 1];
  v1_idx_1 += v1_0 * tmp_1;
  v1[i_0] = v1_idx_1;
  v1_idx_1 = Product2_l * tmp_2;
  v1_idx_1 += v1_0 * v1_idx_0;
  v1[i_0 + 1] = v1_idx_1;
  i_0 += 2;
}

tmp[0] = Kalman_1RC_B.Assignment[0];
tmp[1] = Kalman_1RC_B.Assignment[1];
tmp[2] = Kalman_1RC_B.Assignment[2];
tmp[3] = Kalman_1RC_B.Assignment[3];
Transpose1 = tmp[0];
tmp_1 = tmp[2];
tmp_2 = tmp[1];
v1_idx_0 = tmp[3];

/* Product: '<S7>/Product2' */
i_0 = 0;
for (i = 0; i < 2; i++) {
```

```
    v1_idx_1 = v1[i_0];
    Product2_l = v1_idx_1 * Transpose1;
    v1_0 = v1[i_0 + 1];
    Product2_l += v1_0 * tmp_1;
    Kalman_1RC_B.Product2_l[i_0] = Product2_l;
    Product2_l = v1_idx_1 * tmp_2;
    Product2_l += v1_0 * v1_idx_0;
    Kalman_1RC_B.Product2_l[i_0 + 1] = Product2_l;
    i_0 += 2;
  }

  /* Sum: '<S7>/Sum1' incorporates:
   *  Constant: '<S7>/Constant'
   *  Product: '<S7>/Product2'
   */
  Kalman_1RC_B.Sum1_h[0] = Kalman_1RC_B.Product2_l[0] + 1.0E-6;
  Kalman_1RC_B.Sum1_h[1] = Kalman_1RC_B.Product2_l[1] + 0.0;
  Kalman_1RC_B.Sum1_h[2] = Kalman_1RC_B.Product2_l[2] + 0.0;
  Kalman_1RC_B.Sum1_h[3] = Kalman_1RC_B.Product2_l[3] + 1.0E-6;

  /* Lookup_n-D: '<S5>/2-D Lookup Table dV0' incorporates:
   *  DataTypeConversion: '<S1>/Data Type Conversion2'
   */
  Kalman_1RC_B.dV0 = look2_binlxpw(Kalman_1RC_B.Delay[0],
    Kalman_1RC_B.DataTypeConversion2,
rtCP_uDLookupTabledV0_bp01Data,
    rtCP_uDLookupTabledV0_bp02Data,
    Kalman_1RC_ConstWithInitP.uDLookupTabledV0_tableData,
    rtCP_uDLookupTabledV0_maxIndex, 24U);

  /* SignalConversion generated from: '<S5>/Transpose1' */
  Kalman_1RC_B.TmpSignalConversionAtTranspose1[0] =
Kalman_1RC_B.dV0;
  Kalman_1RC_B.TmpSignalConversionAtTranspose1[1] =
    Kalman_1RC_ConstB.Conversion_ft;

  /* Math: '<S5>/Transpose1' */
  Transpose1 = Kalman_1RC_B.TmpSignalConversionAtTranspose1[0];
  Kalman_1RC_B.Transpose1[0] = Transpose1;

  /* Math: '<S3>/Transpose' incorporates:
   *  Math: '<S5>/Transpose1'
   */
  Kalman_1RC_B.Transpose_e[0] = Transpose1;

  /* Math: '<S5>/Transpose1' */
  Transpose1 = Kalman_1RC_B.TmpSignalConversionAtTranspose1[1];
  Kalman_1RC_B.Transpose1[1] = Transpose1;

  /* Math: '<S3>/Transpose' incorporates:
   *  Math: '<S5>/Transpose1'
   */
```

```
Kalman_1RC_B.Transpose_e[1] = Transpose1;

/* Product: '<S3>/Product' incorporates:
 *  Math: '<S3>/Transpose'
 *  Product: '<S3>/Product2'
 *  Sum: '<S7>/Sum1'
 */
tmp[0] = Kalman_1RC_B.Sum1_h[0];
tmp[1] = Kalman_1RC_B.Sum1_h[1];
tmp[2] = Kalman_1RC_B.Sum1_h[2];
tmp[3] = Kalman_1RC_B.Sum1_h[3];
tmp_1 = Kalman_1RC_B.Transpose_e[0];
tmp_2 = Kalman_1RC_B.Transpose_e[1];
Transpose1 = tmp[0] * tmp_1;
v1_idx_1 = tmp[2] * tmp_2 + Transpose1;

/* Product: '<S3>/Product' */
Transpose1 = v1_idx_1;
Kalman_1RC_B.Product[0] = Transpose1;

/* Product: '<S3>/Product' incorporates:
 *  Product: '<S3>/Product2'
 */
Transpose1 = tmp[1] * tmp_1;
Transpose1 += tmp[3] * tmp_2;

/* Product: '<S3>/Product' */
Kalman_1RC_B.Product[1] = Transpose1;

/* Product: '<S3>/Product2' incorporates:
 *  Math: '<S5>/Transpose1'
 */
v1_idx_0 = v1_idx_1;
v1_idx_1 = Transpose1;
tmp_2 = Kalman_1RC_B.Transpose1[0];
Transpose1 = tmp_2 * v1_idx_0;

/* Product: '<S3>/Product1' incorporates:
 *  Product: '<S3>/Product'
 */
tmp_2 = Kalman_1RC_B.Product[0];
tmp_1 = tmp_2;

/* Product: '<S3>/Product2' incorporates:
 *  Math: '<S5>/Transpose1'
 */
tmp_2 = Kalman_1RC_B.Transpose1[1];
Transpose1 += tmp_2 * v1_idx_1;

/* Product: '<S3>/Product1' incorporates:
 *  Product: '<S3>/Product'
 */
```

```
  tmp_2 = Kalman_1RC_B.Product[1];

  /* Product: '<S3>/Product2' */
  Kalman_1RC_B.Product2_lm = Transpose1;

  /* Sum: '<S3>/Sum2' incorporates:
   *  Constant: '<S3>/Constant'
   */
  Kalman_1RC_B.Sum2 = Kalman_1RC_B.Product2_lm + 0.7;

  /* Product: '<S3>/Divide' */
  Transpose1 = Kalman_1RC_B.Sum2;

  /* Product: '<S3>/Divide' incorporates:
   *  DataTypeConversion: '<S9>/Conversion'
   */
  Kalman_1RC_B.Divide_j = Kalman_1RC_ConstB.Conversion_d /
Transpose1;

  /* Product: '<S3>/Product1' */
  Transpose1 = Kalman_1RC_B.Divide_j;

  /* DataTypeConversion: '<S1>/Data Type Conversion1'
incorporates:
   *  Inport: '<Root>/BattVoltage_In'
   */
  Kalman_1RC_B.DataTypeConversion1 = Kalman_1RC_U.BattVoltage;

  /* Product: '<S1>/Divide2' incorporates:
   *  Constant: '<S1>/Constant2'
   */
  Kalman_1RC_B.Divide2 = Kalman_1RC_B.DataTypeConversion1 / 10.0;

  /* Gain: '<S1>/Gain' */
  Kalman_1RC_B.Gain = 1.0 * Kalman_1RC_B.Divide2;

  /* Lookup_n-D: '<S5>/2-D Lookup Table V0' incorporates:
   *  DataTypeConversion: '<S1>/Data Type Conversion2'
   */
  Kalman_1RC_B.V0 = look2_binlxpw(Kalman_1RC_B.Delay[0],
    Kalman_1RC_B.DataTypeConversion2,
rtCP_uDLookupTableV0_bp01Data,
    rtCP_uDLookupTableV0_bp02Data, rtCP_uDLookupTableV0_tableData,
    rtCP_uDLookupTableV0_maxIndex, 24U);

  /* Lookup_n-D: '<S5>/2-D Lookup Table R0' incorporates:
   *  DataTypeConversion: '<S1>/Data Type Conversion2'
   */
  Kalman_1RC_B.R0 = look2_binlxpw(Kalman_1RC_B.Delay[0],
    Kalman_1RC_B.DataTypeConversion2,
rtCP_uDLookupTableR0_bp01Data,
    rtCP_uDLookupTableR0_bp02Data, rtCP_uDLookupTableR0_tableData,
```

```
    rtCP_uDLookupTableR0_maxIndex, 24U);

  /* Product: '<S1>/Divide' incorporates:
   *  Constant: '<S1>/Constant'
   *  Inport: '<Root>/PackCurrent_In'
   */
  Kalman_1RC_B.Divide_g = (real_T)Kalman_1RC_U.BattCurrent / 10.0;

  /* Gain: '<S2>/Gain' */
  Kalman_1RC_B.Gain_a = (-1.0) * Kalman_1RC_B.Divide_g;

  /* Product: '<S5>/Product6' */
  Kalman_1RC_B.Product6 = Kalman_1RC_B.R0 * Kalman_1RC_B.Gain_a;

  /* Sum: '<S5>/Sum2' */
  Kalman_1RC_B.Sum2_m = (Kalman_1RC_B.V0 - Kalman_1RC_B.Product6)
-
    Kalman_1RC_B.Delay[1];

  /* Sum: '<S2>/Sum' */
  Kalman_1RC_B.Sum = Kalman_1RC_B.Gain - Kalman_1RC_B.Sum2_m;

  /* Product: '<S3>/Product1' */
  v1_idx_1 = tmp_1 * Transpose1;
  Kalman_1RC_B.Kk1[0] = v1_idx_1;

  /* Product: '<S3>/Product3' incorporates:
   *  Product: '<S3>/Product1'
   */
  tmp_1 = v1_idx_1;

  /* Product: '<S3>/Product1' */
  v1_idx_1 = tmp_2 * Transpose1;
  Kalman_1RC_B.Kk1[1] = v1_idx_1;

  /* Product: '<S3>/Product3' incorporates:
   *  Product: '<S3>/Product1'
   */
  tmp_2 = v1_idx_1;
  Transpose1 = Kalman_1RC_B.Sum;
  tmp_1 *= Transpose1;

  /* Product: '<S3>/Product3' */
  Kalman_1RC_B.Product3[0] = tmp_1;

  /* Product: '<S3>/Product3' */
  tmp_1 = tmp_2;
  tmp_1 *= Transpose1;

  /* Product: '<S3>/Product3' */
  Kalman_1RC_B.Product3[1] = tmp_1;
```

```
  /* Product: '<S7>/Product' incorporates:
   *  Assignment: '<S5>/Assignment'
   */
  tmp[0] = Kalman_1RC_B.Assignment[0];
  tmp[1] = Kalman_1RC_B.Assignment[1];
  tmp[2] = Kalman_1RC_B.Assignment[2];
  tmp[3] = Kalman_1RC_B.Assignment[3];
  tmp_1 = Kalman_1RC_B.Delay[0];
  tmp_2 = Kalman_1RC_B.Delay[1];
  v1_idx_1 = tmp[0] * tmp_1;
  v1_idx_1 += tmp[2] * tmp_2;

  /* Product: '<S7>/Product' */
  Kalman_1RC_B.Product_l[0] = v1_idx_1;

  /* Product: '<S7>/Product' */
  v1_idx_1 = tmp[1] * tmp_1;
  v1_idx_1 += tmp[3] * tmp_2;

  /* Product: '<S7>/Product' */
  Kalman_1RC_B.Product_l[1] = v1_idx_1;

  /* Sum: '<S5>/Sum' */
  Kalman_1RC_B.Sum_p = Kalman_1RC_ConstB.Conversion_f -
    Kalman_1RC_B.MathFunction;

  /* Product: '<S5>/Product4' */
  Kalman_1RC_B.Product4 = Kalman_1RC_B.Sum_p * Kalman_1RC_B.R1;

  /* SignalConversion generated from: '<S7>/Product1' */
  Kalman_1RC_B.TmpSignalConversionAtProduct1In[0] =
Kalman_1RC_ConstB.Gain;
  Kalman_1RC_B.TmpSignalConversionAtProduct1In[1] =
Kalman_1RC_B.Product4;

  /* Product: '<S7>/Product1' */
  Transpose1 = Kalman_1RC_B.Gain_a;
  v1_idx_1 = Kalman_1RC_B.TmpSignalConversionAtProduct1In[0];
  v1_idx_1 *= Transpose1;

  /* Product: '<S7>/Product1' */
  Kalman_1RC_B.Product1[0] = v1_idx_1;

  /* Sum: '<S7>/Sum' */
  v1_idx_1 += Kalman_1RC_B.Product_l[0];
  Kalman_1RC_B.Sum_g[0] = v1_idx_1;

  /* Sum: '<S3>/Sum' */
  v1_idx_1 += Kalman_1RC_B.Product3[0];
  Kalman_1RC_B.Sum_j[0] = v1_idx_1;

  /* Product: '<S7>/Product1' */
```

```
v1_idx_1 = Kalman_1RC_B.TmpSignalConversionAtProduct1In[1];
v1_idx_1 *= Transpose1;

/* Product: '<S7>/Product1' */
Kalman_1RC_B.Product1[1] = v1_idx_1;

/* Sum: '<S7>/Sum' */
v1_idx_1 += Kalman_1RC_B.Product_l[1];
Kalman_1RC_B.Sum_g[1] = v1_idx_1;

/* Sum: '<S3>/Sum' */
v1_idx_1 += Kalman_1RC_B.Product3[1];
Kalman_1RC_B.Sum_j[1] = v1_idx_1;

/* Assignment: '<S3>/Assignment' */
Kalman_1RC_B.Assignment_l[1] = v1_idx_1;

/* Saturate: '<S3>/Saturation' */
Transpose1 = Kalman_1RC_B.Sum_j[0];
v1_idx_1 = 0.0;
tmp_1 = 1.0;
if (Transpose1 > tmp_1) {
  /* Saturate: '<S3>/Saturation' */
  Kalman_1RC_B.Saturation = tmp_1;
} else if (Transpose1 < v1_idx_1) {
  /* Saturate: '<S3>/Saturation' */
  Kalman_1RC_B.Saturation = v1_idx_1;
} else {
  /* Saturate: '<S3>/Saturation' */
  Kalman_1RC_B.Saturation = Transpose1;
}

/* End of Saturate: '<S3>/Saturation' */

/* Assignment: '<S3>/Assignment' */
Kalman_1RC_B.Assignment_l[0] = Kalman_1RC_B.Saturation;

/* Product: '<S2>/Divide' incorporates:
 *  Constant: '<S2>/Constant'
 */
Kalman_1RC_B.Divide_d = Kalman_1RC_B.Assignment_l[0] * 100.0;

/* Gain: '<S6>/K' */
Kalman_1RC_B.K = 1.0 * Kalman_1RC_B.Divide_d;

/* DiscreteIntegrator: '<S20>/Integrator' */
if (Kalman_1RC_DWork.Integrator_IC_LOADING != 0) {
  Kalman_1RC_DWork.Integrator_DSTATE = Kalman_1RC_B.K;
  if (Kalman_1RC_DWork.Integrator_DSTATE >= 100.0) {
    Kalman_1RC_DWork.Integrator_DSTATE = 100.0;
  } else if (Kalman_1RC_DWork.Integrator_DSTATE <= 0.0) {
    Kalman_1RC_DWork.Integrator_DSTATE = 0.0;
```

```
    }
  }

  if (Kalman_1RC_B.LogicalOperator ||
      (Kalman_1RC_DWork.Integrator_PrevResetState != 0)) {
    Kalman_1RC_DWork.Integrator_DSTATE = Kalman_1RC_B.K;
    if (Kalman_1RC_DWork.Integrator_DSTATE >= 100.0) {
      Kalman_1RC_DWork.Integrator_DSTATE = 100.0;
    } else if (Kalman_1RC_DWork.Integrator_DSTATE <= 0.0) {
      Kalman_1RC_DWork.Integrator_DSTATE = 0.0;
    }
  }

  if (Kalman_1RC_DWork.Integrator_DSTATE >= 100.0) {
    Kalman_1RC_DWork.Integrator_DSTATE = 100.0;
  } else if (Kalman_1RC_DWork.Integrator_DSTATE <= 0.0) {
    Kalman_1RC_DWork.Integrator_DSTATE = 0.0;
  }

  /* DiscreteIntegrator: '<S20>/Integrator' */
  Kalman_1RC_B.Integrator = Kalman_1RC_DWork.Integrator_DSTATE;

  /* Saturate: '<S20>/Saturation' */
  Kalman_1RC_B.Saturation_g = Kalman_1RC_B.Integrator;

  /* DataTypeConversion: '<S2>/Data Type Conversion' */
  Transpose1 = floor(Kalman_1RC_B.Saturation_g);
  if (rtIsNaN(Transpose1) || rtIsInf(Transpose1)) {
    Transpose1 = 0.0;
  } else {
    Transpose1 = fmod(Transpose1, 4.294967296E+9);
  }

  /* DataTypeConversion: '<S2>/Data Type Conversion' */
  Kalman_1RC_B.DataTypeConversion = Transpose1 < 0.0 ? -
(int32_T)(uint32_T)
    -Transpose1 : (int32_T)(uint32_T)Transpose1;

  /* Outport: '<Root>/SoC_out' incorporates:
   *  DataTypeConversion: '<S1>/Cast'
   */
  Kalman_1RC_Y.SoC_out = (uint8_T)Kalman_1RC_B.DataTypeConversion;

  /* Product: '<S3>/Product4' incorporates:
   *  Math: '<S5>/Transpose1'
   *  Product: '<S3>/Product1'
   */
  tmp_1 = Kalman_1RC_B.Kk1[0];
  v1_idx_0 = Kalman_1RC_B.Transpose1[0];
  tmp_2 = Kalman_1RC_B.Kk1[1];
  v1_idx_1 = Kalman_1RC_B.Transpose1[1];
```

```
  /* Product: '<S3>/Product4' */
  Kalman_1RC_B.Product4_b[0] = tmp_1 * v1_idx_0;
  Kalman_1RC_B.Product4_b[1] = tmp_2 * v1_idx_0;
  Kalman_1RC_B.Product4_b[2] = tmp_1 * v1_idx_1;
  Kalman_1RC_B.Product4_b[3] = tmp_2 * v1_idx_1;

  /* Sum: '<S3>/Sum3' incorporates:
   *  DataTypeConversion: '<S8>/Conversion'
   *  Product: '<S3>/Product4'
   */
  Transpose1 = Kalman_1RC_ConstB.Conversion[0] -
Kalman_1RC_B.Product4_b[0];
  Kalman_1RC_B.Sum3[0] = Transpose1;

  /* Product: '<S3>/Product5' incorporates:
   *  Sum: '<S3>/Sum3'
   *  Sum: '<S7>/Sum1'
   */
  tmp[0] = Transpose1;
  tmp_0[0] = Kalman_1RC_B.Sum1_h[0];

  /* Sum: '<S3>/Sum3' incorporates:
   *  DataTypeConversion: '<S8>/Conversion'
   *  Product: '<S3>/Product4'
   */
  Transpose1 = Kalman_1RC_ConstB.Conversion[1] -
Kalman_1RC_B.Product4_b[1];
  Kalman_1RC_B.Sum3[1] = Transpose1;

  /* Product: '<S3>/Product5' incorporates:
   *  Sum: '<S3>/Sum3'
   *  Sum: '<S7>/Sum1'
   */
  tmp[1] = Transpose1;
  tmp_0[1] = Kalman_1RC_B.Sum1_h[1];

  /* Sum: '<S3>/Sum3' incorporates:
   *  DataTypeConversion: '<S8>/Conversion'
   *  Product: '<S3>/Product4'
   */
  Transpose1 = Kalman_1RC_ConstB.Conversion[2] -
Kalman_1RC_B.Product4_b[2];
  Kalman_1RC_B.Sum3[2] = Transpose1;

  /* Product: '<S3>/Product5' incorporates:
   *  Sum: '<S3>/Sum3'
   *  Sum: '<S7>/Sum1'
   */
  tmp[2] = Transpose1;
  tmp_0[2] = Kalman_1RC_B.Sum1_h[2];

  /* Sum: '<S3>/Sum3' incorporates:
```

```
   *  DataTypeConversion: '<S8>/Conversion'
   *  Product: '<S3>/Product4'
   */
  Transpose1 = Kalman_1RC_ConstB.Conversion[3] -
Kalman_1RC_B.Product4_b[3];
  Kalman_1RC_B.Sum3[3] = Transpose1;

  /* Product: '<S3>/Product5' incorporates:
   *  Sum: '<S3>/Sum3'
   *  Sum: '<S7>/Sum1'
   */
  tmp[3] = Transpose1;
  tmp_0[3] = Kalman_1RC_B.Sum1_h[3];

  /* MinMax: '<S14>/Max' incorporates:
   *  Constant: '<S14>/Time constant'
   */
  Transpose1 = fmax(Kalman_1RC_B.Probe[0], 200.0);

  /* MinMax: '<S14>/Max' */
  Kalman_1RC_B.Max = Transpose1;

  /* Fcn: '<S14>/Avoid Divide by Zero' */
  Kalman_1RC_B.AvoidDividebyZero = (real_T)(Kalman_1RC_B.Max ==
0.0) *
    2.2204460492503131e-16 + Kalman_1RC_B.Max;

  /* Sum: '<S6>/Sum1' */
  Kalman_1RC_B.Sum1_g = Kalman_1RC_B.K -
Kalman_1RC_B.Saturation_g;

  /* Product: '<S6>/1//T' */
  Kalman_1RC_B.uT = 1.0 / Kalman_1RC_B.AvoidDividebyZero *
Kalman_1RC_B.Sum1_g;

  /* Update for Delay: '<S4>/Delay' */
  Kalman_1RC_DWork.icLoad = false;

  /* Product: '<S3>/Product5' */
  Transpose1 = tmp_0[0];
  tmp_1 = tmp_0[1];
  tmp_2 = tmp_0[2];
  v1_idx_0 = tmp_0[3];
  for (i_0 = 0; i_0 < 2; i_0++) {
    /* Product: '<S3>/Product5' */
    Product2_l = tmp[i_0];

    /* Product: '<S3>/Product5' */
    v1_idx_1 = Product2_l * Transpose1;

    /* Product: '<S3>/Product5' */
    v1_0 = tmp[i_0 + 2];
```

```
    /* Product: '<S3>/Product5' */
    v1_idx_1 += v1_0 * tmp_1;

    /* Product: '<S3>/Product5' */
    Kalman_1RC_B.Product5[i_0] = v1_idx_1;

    /* Product: '<S3>/Product5' */
    v1_idx_1 = Product2_l * tmp_2;
    v1_idx_1 += v1_0 * v1_idx_0;

    /* Product: '<S3>/Product5' */
    Kalman_1RC_B.Product5[i_0 + 2] = v1_idx_1;

    /* Update for Delay: '<S4>/Delay' incorporates:
     *  Product: '<S3>/Product5'
     */
    Kalman_1RC_DWork.Delay_DSTATE[i_0] =
Kalman_1RC_B.Assignment_l[i_0];
  }

  /* Update for UnitDelay: '<S2>/Unit Delay - P' incorporates:
   *  Product: '<S3>/Product5'
   */
  Kalman_1RC_DWork.UnitDelayP_DSTATE[0] =
Kalman_1RC_B.Product5[0];
  Kalman_1RC_DWork.UnitDelayP_DSTATE[1] =
Kalman_1RC_B.Product5[1];
  Kalman_1RC_DWork.UnitDelayP_DSTATE[2] =
Kalman_1RC_B.Product5[2];
  Kalman_1RC_DWork.UnitDelayP_DSTATE[3] =
Kalman_1RC_B.Product5[3];

  /* Update for DiscreteIntegrator: '<S20>/Integrator' */
  Kalman_1RC_DWork.Integrator_IC_LOADING = 0U;
  if (!Kalman_1RC_B.LogicalOperator) {
    Kalman_1RC_DWork.Integrator_DSTATE += 0.01 * Kalman_1RC_B.uT;
    if (Kalman_1RC_DWork.Integrator_DSTATE >= 100.0) {
      Kalman_1RC_DWork.Integrator_DSTATE = 100.0;
    } else if (Kalman_1RC_DWork.Integrator_DSTATE <= 0.0) {
      Kalman_1RC_DWork.Integrator_DSTATE = 0.0;
    }
  }

  Kalman_1RC_DWork.Integrator_PrevResetState = (int8_T)
    Kalman_1RC_B.LogicalOperator;

  /* End of Update for DiscreteIntegrator: '<S20>/Integrator' */
}

/* Model initialize function */
void Kalman_1RC_initialize(void)
```

```
{
  /* Registration code */

  /* initialize non-finites */
  rt_InitInfAndNaN(sizeof(real_T));

  /* non-finite (run-time) assignments */
  Kalman_1RC_ConstWithInitP.uDLookupTabledV0_tableData[0] = rtNaN;
  Kalman_1RC_ConstWithInitP.uDLookupTabledV0_tableData[24] =
rtNaN;
  Kalman_1RC_ConstWithInitP.uDLookupTabledV0_tableData[48] =
rtNaN;

  /* initialize error status */
  rtmSetErrorStatus(Kalman_1RC_M, (NULL));

  /* block I/O */
  (void) memset(((void *) &Kalman_1RC_B), 0,
                sizeof(BlockIO_Kalman_1RC));

  /* states (dwork) */
  (void) memset((void *)&Kalman_1RC_DWork, 0,
                sizeof(D_Work_Kalman_1RC));

  /* external inputs */
  (void)memset(&Kalman_1RC_U, 0,
sizeof(ExternalInputs_Kalman_1RC));

  /* external outputs */
  Kalman_1RC_Y.SoC_out = 0U;

  /* Start for Probe: '<S14>/Probe' */
  Kalman_1RC_B.Probe[0] = 0.01;
  Kalman_1RC_B.Probe[1] = 0.0;

  /* InitializeConditions for Delay: '<S4>/Delay' */
  Kalman_1RC_DWork.icLoad = true;

  /* InitializeConditions for UnitDelay: '<S2>/Unit Delay - P' */
  Kalman_1RC_DWork.UnitDelayP_DSTATE[0] = 1.0E-5;
  Kalman_1RC_DWork.UnitDelayP_DSTATE[1] = 0.0;
  Kalman_1RC_DWork.UnitDelayP_DSTATE[2] = 0.0;
  Kalman_1RC_DWork.UnitDelayP_DSTATE[3] = 1.0;

  /* InitializeConditions for DiscreteIntegrator:
'<S20>/Integrator' */
  Kalman_1RC_DWork.Integrator_PrevResetState = 0;
  Kalman_1RC_DWork.Integrator_IC_LOADING = 1U;
}

/* Model terminate function */
void Kalman_1RC_terminate(void)
```

```
{
  /* (no terminate code required) */
}

/*
 * File trailer for generated code.
 *
 * [EOF]
 */
```

Bibliografia

[1] S. W. J. Z. J. Q. Chao Wang, «A Novel BCRLS-BP-EKF Method for the State of Charge Estimation of Lithium-ion Batteries,» *International Journal of Electrochemical Science,* vol. 17, n. 220431, 2022.

[2] S. Z. H. C. X. L. C. Z. X. X. H. L. a. S. Y. Y. Hua, *International Journal of Energy Research,* vol. 44, p. 11059, 2020.

[3] S. I. R. K. C. M. J. R. D. K. S. M. H. A. S. I. M. S. S. A. Hasib, *Ieee Access,* vol. 9, p. 86166, 2021.

[4] N. S. H. J. A. D. a. A. I. O. M. Al Gabalawy, *International Journal of Energy Research,* vol. 45, p. 6708, 2021.

[5] S. W. H. J. C. F. a. X. X. L. Chen, *International Journal of Electrochemical Science,* vol. 16, p. 1, 2021.

[6] M. e. a. Hannan, «A review of lithium-ion battery state of charge estimation and management system in electric vehicle applications: challenges and recommendations.,» *Renew. Sustain. Energy Rev. 78,* p. 834–854, 2017.

[7] R. e. a. Xiong, «Critical review on the battery state of charge estimation methods for electric vehicles.,» *IEEE Access,* vol. 6, p. 1832–1843, 2018.

[8] M. e. a. Hannan, «State-of-the-art and energy management system of lithium-ion batteries in electric vehicle applications: issues and recommendations.,» *IEEE Access,* vol. 6, p. 19362–19378, 2018.

[9] M. e. a. Lazreg, «SoC estimation of Li-ion battery pack for light electric vehicles using enhanced coulomb counting algorithm.,» in *10th International Renewable Energy Congress (IREC)*, 2019.

[10] M. e. a. Lazreg, «Lithium-ion battery pack modeling using accurate OCV model: application for SoC and SoH estimation.,» in *4th International Conference on Intelligent Energy and Power Systems*, 2020.

[11] H. I. C. Chaoui, «State of charge and state of health estimation for lithium batteries using recurrent neural networks.,» *IEEE Trans. Veh. Technol.,* vol. 66, p. 8773–8783, 2017.

[12] F. e. a. Yang, «State-of-charge estimation of lithium-ion batteries via long short-term memory network.,» *IEEE Access,* vol. 7, p. 53792–53799, 2019.

[13] J. e. a. Anton, «Support vector machines used to estimate the battery state of charge.,» *IEEE Trans. Power Electron.,* vol. 28, p. 5919–5926, 2013.

[14] K. e. a. Liu, «Towards long lifetime battery: AI-based manufacturing and management.,» *IEEE/CAA J. Autom. Sin.,* vol. 9, p. 1139–1165, 2022.

[15] R. e. a. Zhang, «State of the art of lithium-ion battery SoC estimation for electrical vehicles.,» *Energies,* vol. 11, p. 1820, 2018 .

[16] S. M. B. H. M. Jemmali, «Pure hardware design and implementation on FPGA of an EKF based accelerated SoC estimator for a lithium-ion battery in electric vehicles.,» *IET Power Electron.,* vol. 15, p. 1004–1015, 2022.

[17] N. e. a. Chen, «Estimating the state-of-charge of lithium-ion battery using an H-infinity observer based on electrochemical impedance model.,» *IEEE Access,* vol. 8, p. 26878–26884, 2020.

[18] M. G. H. C. B. Ye, «A model-based adaptive state of charge estimator for a lithium-ion battery using an improved adaptive particle filter.,» *Appl. Energy,* vol. 190, p. 740–748, 2017.

[19] H. e. a. Afshari, «Reliable state of charge and state of health estimation using the smooth variable structure filter,» *Control Eng. Pract,* vol. 77, p. 1–14, 2018.

[20] F. e. a. Zhong, «An SoC estimation approach based on adaptive sliding mode observer and fractional order equivalent circuit model for lithium-ion batteries.,» *Commun. Nonlinear Sci. Numer. Simulat,* vol. 24, p. 127–144, 2015.

[21] N. e. a. Lotfi, «Critical review on the battery state of charge estimation methods for electric vehicles.,» *IEEE Trans. Control Syst. Technol.,* vol. 25, p. 1217–1230, 2017.

[22] F. e. a. Feng, «Co-estimation of lithium-ion battery state of charge and state of temperature based on a hybrid electrochemical-thermal-neural-network model.,» *J. Power Sources,* vol. 455, p. 227935, 2020.

[23] F. e. a. Naseri, «An enhanced equivalent circuit model with real-time parameter identification for battery state-of-charge estimation.,» *IEEE Trans. Ind. Electron.,* vol. 69, p. 3743–3751, 2022.

[24] J. M. N. S. H. Rivera-Barrera, «SoC estimation for lithium-ion batteries: review and future challenges.,» *Electronics,* vol. 6, p. 102, 2017.

[25] K. J. M. G. L. Khan, «Comparison of Li-ion battery equivalent circuit modelling using impedance analyser and Bayesian networks.,» *IET Electr. Syst. Transp.,* vol. 8, p. 197–204, 2018.

[26] B. P. K. H. S. Bairwa, «Investigation on lithium ion battery equivalent circuit models for dynamic load profiles.,» *Energy Storage,* vol. 3, p. 231, 2021.

[27] D. e. a. Mahboubi, «State of charge estimation for lithium-ion batteries based on square root sigma point Kalman filter considering temperature variations.,» *IET Electr. Syst. Transp.,* vol. 12, p. 1–16, 2022.

[28] X. e. a. Du, «An information appraisal procedure: endows reliable online parameter identification to lithium-ion battery model.,» *IEEE Trans. Ind. Electron.,* vol. 69, p. 5889–5899, 2021.

[29] Y. e. a. Huo, « Research on parameter identification and state of charge estimation of improved equivalent circuit model of Li-ion battery based on temperature effects for battery thermal management.,» *Int. J. Energy Res.,* vol. 44, p. 11583–11596, 2020.

[30] X. L. X. D. J. Wu, «State of charge estimation of lithium-ion batteries over wide temperature range using unscented Kalman filter.,» *IEEE Access,* vol. 6, p. 41993–42003, 2018.

[31] J. J. W. Z. a. S. M. S. Caiping Zhang, «Estimation of State of Charge of Lithium-Ion Batteries Used in HEV Using Robust Extended Kalman Filtering,» *Energies,* vol. 5, pp. 1098-1115, 2012.

[32] D. M. M. H. A. P.Venegasa, «Kalman filter and classical Preisach hysteresis model applied to the state of charge battery estimation,» *Computers andMathematicswithApplications,* vol. 118, pp. 74-84, 2022.

[33] Z. Lin, «Research Status of Cathode Materials for Lithium Ion Batteries,» *MATEC Web of Conferences,* vol. 382, 2023.

[34] S. B. H. X. S. Z. Q. K. e. a. Xie, «Aging-aware co-optimization of battery size,depth of discharge,and energy management for plug-in hybrid electric vehicles.,» *Power Sources,* vol. 250, n. 227638, 2020.

[35] P. C. C. J. Y. &. W. Z. Y. Pei, «Selfdischarge mechanism and measurement methods for lithium ion batteries,» *Journal of Tsinghua University (Science and Technology),* vol. 59, n. 1, pp. 53-65, 2019.

[36] D. G. R. e. a. Ramsey, «Comparison of Equivalent Circuit Battery Models for Energetic Studies on Electric Vehicles,» in *17th IEEE Vehicle Power and Propulsion Conference*, 2020.

[37] J. H. Y. L. K. L. H. F. C. Zhou, «Research on a High Precision Modeling Solution for Lithium-ion Batteries,» *Proceedings of the Chinese Society for Electrical Engineering,* vol. 39, n. 21, pp. 6394-6403, 2019.

[38] R. B. T. Milishchuk, «Thevenin-based Battery Model with Ageing Effects in Modelica,» in *In 21st IEEE Mediterranean Electrotechnical Conference*, 2022.

[39] Y. Q. S. L. G. Ji, «Simulation of Lithium-ion Battery Second-order RC Equivalent Circuit Model Based on Variable Resistance and

Capacitance,» *Journal of Central South University,* vol. 27, n. 09, pp. 2606-2613, 2020.

[40] S. G. D. e. a. Nejad, «A Systematic Review of Lumped-parameter Equivalent Circuit Models for Real-time Estimation of Lithium-ion Battery States,» *Journal of Power Sources,* vol. 316, pp. 183-196, 2016.

[41] X. G. Y. C. Y. e. a. Yan, «Electric Vehicle Battery State of Charge Estimation based on GNL Model Adaptive Kalman Filter,» *Journal of Physics: Conference Series,* vol. 1087, n. 5, 2018.

[42] Y. Xia, «Comparison and Analysis of SOC Estimation Based on First-Order and Second-Order Thevenin Battery Models Based on EKF,» *Academic Journal of Science and Technology,* vol. 6, n. 3, pp. 10-18, 2023.

[43] K. Movassagh, A. Raihan, B. Balasingam e K. Pattipati, «A Critical Look at Coulomb Counting Approach for State of Charge Estimation in Batteries,» *Energies ,* vol. 14, n. 4074, 2021.

[44] A. P. A. Mohinder S. Grewal, Kalman Filtering: Theory and Practice Using MATLAB, John Wiley & Sons, Inc, 2001.

Ringraziamenti

Come prima menzione voglio ringraziare il mio relatore, il professor Massimo Conti, che mi ha seguito durante tutto il tirocinio e durante la stesura della tesi, dandomi dei validissimi spunti e un allettante scorcio di che cosa possa essere la vita accademica; vorrei inoltre ringraziarlo per la gentilezza dimostrata anche nel trattare i più piccoli dettagli relativi al lavoro svolto.

Ringrazio il mio correlatore Rino Valente per aver eccellentemente fatto da tramite con l'Energica Motor Company e avermi accolto in azienda durante le giornate di tirocinio.

Ringrazio Giovanni Gherardi che mi ha aiutato e assistito oltre ciò che si potrebbe chiedere, nelle questioni tecniche relative al progetto.

Fatta questa parte più "tecnica", questa sezione riguarderà i miei cari.

Nella mia ultima tesi sono stato particolarmente sintetico nel ringraziare i miei affetti, questa volta proverò a fare qualcosa di meglio.

Inizio col ringraziare i miei genitori, Rosita e Roberto, per avermi dato la possibilità di giungere fino a questo punto del mio percorso di studi e di aver condiviso in maniera attiva questa importante parte della mia vita.

Ringrazio mia sorella Gaia per avermi reso partecipe della marea di turbe letterarie con cui è venuta in contatto e per aver reso in qualche modo

accattivante anche le materie umanistiche, distraendomi talvolta dai tecnicismi che regolano il mio mondo.

Ringrazio i miei nonni materni, Alfiera e Salvatore, custodi di affascinanti conoscenze sulla vita e senza i quali non avrei probabilmente fatto le scelte che ho fatto.

Ringrazio i miei nonni paterni, Mariola e Leonardo, in cui rivedo molti dei miei tratti caratteriali e la cui mancanza è avvenuta troppo prematuramente.

Ringrazio i miei zii, Alessandra e Michele, Fabio e Gemma, Matteo e Barbara, grazie ai loro hobby, al loro calore e "colore" sono venuto a contatto con i più svariati ambienti nel corso della mia vita.

Ovviamente la famiglia è importante ma la vita non sarebbe così piena senza ciò che alcuni hanno definito la "Family" (ovvero la cerchia di persone che qualcuno si sceglie); senza frapporre ulteriori indugi, è arrivato il momento di tuffarci nella giungla dei ringraziamenti relativi al gruppo di persone che fanno parte della mia vita, volenti o meno (perlopiù), in modo attivo e che nel corso degli anni si sono guadagnati un posto nel mio tempo.

Ringrazio Matteo Di Tommaso per essersi dimostrato ad ogni occasione un estremamente pedante (detto ovviamente in senso negativo) affidabile amico e per le innumerevoli telefonate fatte per organizzare le uscite in montagna che

hanno fatto sì che il mio percorso di studi durasse ameno 2 anni in più (ma lo rifarei).

Ringrazio Ismaela Avellino, che mi ha sempre fornito punti di vista nelle questioni al di fuori del mio modo di valutare le cose che mi hanno permesso di allargare (di poco vista la mia rigidità) i miei orizzonti cognitivi.

Ringrazio Giacomo Jack Cupido per avermi accompagnato durante una parte del mio percorso accademico e per avermi poi mostrato come qualcosa di imbevibile (gli alcolici oltre i 20°) possa diventare, per alcuni, addirittura arte.

Ringrazio Mark Giacchetti, la sua passione per la storia e per il fascino d'altri tempi lo rende una preziosa amicizia e una gradevole compagnia, soprattutto quando le riunioni avvengono attorno ad una bottiglia di vino (o Whisky dopo la conoscenza di Jack).

Ringrazio Marco Bilò, tra gli innumerevoli aneddoti che ho nei ricordi, la maggior parte derivano dalla tua (apparente) mancanza di senno, e mi hanno spinto, oltre a dei limiti che non avrei probabilmente valicato altrimenti.

Ringrazio Marco Montenovo, fonte inesauribile di idee pratiche che rivelano un notevole ingegno e compagnia sempre gradita, soprattutto quando un malcapitato animale è posto sui carboni ardenti.

Ringrazio Paolo Pincini, amico perso per qualche anno e poi ritrovato con il quale abbiamo condiviso innumerevoli giri motociclistici persi nelle strade del centro Italia.

Ringrazio Andrea Amoroso, Diego Brandoni, Matteo Burroni e Nicholas Balducci, con i quali ho condiviso immemori esperienze nella tarda adolescenza e oltre.

Ringrazio Diego Kaneda Pierini, ormai è chiaro che io sia un tuo fan, ma ciò che mi ha sempre meravigliato è il tuo modo di percepire il mondo che ti circonda con la curiosità di chi non ha mai perso la sua innocenza.

Ringrazio Marco Stacey Rinaldi, che con le sue massime mi ha dimostrato che significa padroneggiare la saggezza di chi ha passato i 27 anni.

Ringrazio Laura De Carlo per avermi aperto ad un mondo, che probabilmente non avrei mai preso in considerazione, che mi ha portato ad una tangibile crescita personale.

Ringrazio i miei amici della palestra, che nelle giornate di studio più intense erano sempre disponibili per una chiacchierata leggera che mi distraesse dai miei problemi.

Ringrazio la famiglia della mia ragazza che mi hanno accettato nella loro cerchia e incluso nelle loro attività.

Per ultima ringrazio la mia ragazza Noemi Magnaterra, che mi sopporta e supporta ogni giorno e con la quale condivido ogni mia vicissitudine e avventura, non potrei desiderare compagna migliore e anche potendo non vorrei.

Voglio concludere ringraziando tutti i miei amici ancora una volta, se la mia vita è un parco giochi è soprattutto merito vostro.

“Keep on Rotting in a Free World”

Michele