Politecnico di Torino

Laurea Magistrale in Ingegneria Elettronica

A.A. 2024/2025
Sessione di Laurea Ottobre 2025

Sviluppo di modelli circuitali per l'integrazione di celle fotovoltaiche e condensatori elettrochimici

Relatori:

LAMBERTI ANDREA (DISAT)
SPERANZA ROBERTO (Docente esterno)
ZACCAGNINI PIETRO (DISAT)

Candidati:

Andrea Iozzia (284660)

Ringraziamenti

Desidero esprimere il mio più sincero ringraziamento al professore Andrea Lamberti e ai dottorandi Roberto Speranza e Pietro Zaccagnini, per la guida, il supporto e i preziosi consigli fornitimi nel corso di questo lavoro di tesi.

Un doveroso ringraziamento va anche ai miei familiari e amici per la comprensione e il supporto morale datomi in questi anni di studio.

A tutti voi, il mio più sentito grazie.

Indice

1 Introduzione

Negli ultimi anni la crescente diffusione di dispositivi elettronici a basso consumo energetico ha aperto nuove prospettive per l'impiego di sistemi di alimentazione autonomi e di lunga durata, capaci di operare senza interventi esterni di manutenzione o sostituzione delle batterie.
Questa esigenza è particolarmente rilevante in applicazioni di sensoristica distribuita e di Internet of Things (IoT), dove la collocazione di questi sensori in ambienti difficilmente accessibili rendono onerosa la manutenzione periodica delle batterie tradizionali.

In tale contesto, l'integrazione tra fonti di energia rinnovabile e sistemi di accumulo rappresenta una soluzione promettente per garantire l'alimentazione continua di sensori, moduli di comunicazione o microcontrollori a basso consumo. Tra le possibili fonti energetiche, il fotovoltaico è una delle più interessanti grazie alla sua disponibilità diffusa, modularità e compatibilità con applicazioni su piccola scala. Tuttavia, in scenari indoor, la radiazione luminosa disponibile è di intensità ridotta e presenta uno spettro e una variabilità differenti rispetto alla luce solare diretta. Ciò comporta una potenza generata limitata e fortemente variabile nel tempo, rendendo necessario l'impiego di un dispositivo di accumulo energetico in grado di stabilizzare la tensione e assicurare un'alimentazione affidabile al carico.

Tra le diverse tecnologie di accumulo, i supercondensatori (o condensatori elettrochimici) si distinguono per la loro elevata densità di potenza, lunga vita ciclica, tempi di carica e scarica ridotti e buona efficienza energetica. Queste caratteristiche li rendono particolarmente adatti ad affiancare celle fotovoltaiche in sistemi intermittenti, dove il dispositivo deve accumulare rapidamente energia durante la fase di illuminazione e rilasciarla durante i periodi di oscurità o basso irraggiamento.

L'obiettivo di questo lavoro di tesi è lo sviluppo e la validazione di un modello completo in ambiente Matlab/Simulink capace di descrivere il comportamento di un sistema fotovoltaico accoppiato a un supercondensatore, destinato all'alimentazione di un carico a basso consumo (ad esempio un sensore wireless operante in condizioni indoor).
Il lavoro si articola in più fasi:

- inizialmente vengono analizzati i dati sperimentali della cella fotovoltaica al fine di determinare, mediante tecniche di fitting non lineare, i parametri caratteristici del modello elettrico equivalente;
- successivamente, il sistema completo (cella fotovoltaica + supercapacitore + carico) viene implementato e simulato per valutarne la capacità di immagazzinare e fornire energia in condizioni operative variabili di irraggiamento e carico;
- infine, vengono confrontati i risultati di simulazione con i dati sperimentali per verificare la coerenza del modello e individuare i limiti e le potenzialità della configurazione proposta.

I risultati ottenuti mostreranno come il modello sviluppato permetta di riprodurre con buona accuratezza il comportamento dinamico del sistema e possieda potenzialità predittive utili per la progettazione di dispositivi energeticamente autonomi.
In prospettiva, il lavoro fornisce una base metodologica per lo sviluppo di sistemi fotocapacitivi integrati per applicazioni in ambito IoT, monitoraggio ambientale, domotica e dispositivi biomedicali portatili, contribuendo al progresso verso l'elettronica "battery-free" e a basso impatto ambientale.

2 Obiettivi della tesi

2.1 Motivazioni

L'affidabilità dei sistemi di sensoristica distribuita dipende in larga misura dalla disponibilità di soluzioni di alimentazione autonome, stabili e durature. L'utilizzo di batterie tradizionali comporta limitazioni rilevanti, dovute alla necessità di sostituzione periodica e allo smaltimento a fine vita. In applicazioni a lungo termine, soprattutto in ambienti difficilmente accessibili, questi vincoli riducono la sostenibilità e l'efficienza del sistema.
Un approccio alternativo è rappresentato dall'impiego di fonti di energia rinnovabile integrate con sistemi di accumulo, capaci di garantire un flusso energetico continuo anche in presenza di sorgenti luminose intermittenti o variabili. Nel contesto specifico di questa tesi, si è scelto di indagare l'utilizzo di un sistema fotovoltaico accoppiato a un supercondensatore per l'alimentazione di un sensore, come può essere un sensore di temperatura operante in condizioni indoor.
L'analisi condotta ha come scopo principale quello di realizzare un tool di simulazione che permetta di fare delle predizioni quanto più realistiche di un sistema integrato di harvesting storage per ambienti indoor, riducendo notevolmente i tempi degli esperimenti e accelerandone quindi la validazione.

2.2 Cella fotovoltaica e supercondensatore

L'obiettivo principale della tesi è lo sviluppo e la validazione di un modello completo che comprenda i due elementi fondamentali del sistema:

- Cella fotovoltaica (PV): responsabile della conversione della radiazione luminosa in energia elettrica, caratterizzata a partire da dati sperimentali raccolti in condizioni di illuminazione indoor.
- Supercondensatore (SC): dispositivo di accumulo scelto per la sua capacità di immagazzinare energia rapidamente e restituirla con elevata efficienza, consentendo di stabilizzare l'alimentazione del sensore.

Il modello, sviluppato in Matlab/Simulink, è stato progettato in modo da riflettere con la maggiore fedeltà possibile le caratteristiche fisiche dei componenti reali.

3 Teoria

3.1 La cella fotovoltaica

Le celle fotovoltaiche (PV) costituiscono l'elemento fondamentale di un sistema di conversione solare, poiché trasformano direttamente la radiazione luminosa in energia elettrica. Diversi modelli matematici sono stati proposti in letteratura per descrivere il comportamento di tali dispositivi. In questo lavoro si fa riferimento in particolare ai contributi di Rahmantian [1], Warepam [2] e Tayeb [3], che hanno affrontato il problema della modellizzazione delle celle solari e delle celle sensibilizzate a colorante (DSSC) attraverso approcci circuitali e termoelettrici.

3.1.1 Modello a diodo singolo

Il modello più comunemente utilizzato è quello a diodo singolo, in cui la cella è rappresentata da una sorgente di corrente fotogenerata I_{ph}, un diodo, una resistenza serie R_s e una resistenza di shunt R_{sh}.
Il generatore di corrente rappresenta la corrente fotogenerata dai fotoni che colpiscono la cella. Il diodo modella il comportamento del giunto p–n della cella e le ricombinazioni al suo interno. La resistenza serie rappresenta tutte le perdite ohmiche al passaggio della corrente. La resistenza di shunt modella le correnti di perdita attraverso percorsi parassiti.

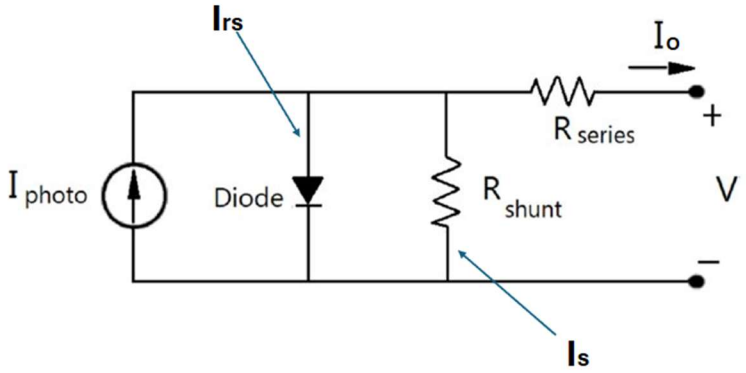

Figura 1: Modello a diodo singolo secondo Tayeb. [3]

L'equazione generale della corrente è:

$$I_o = I_{ph} - I_{rs}\left(e^{\frac{q(V_o+I_oR_s)}{nkT_{op}}} - 1\right) - \frac{V_o + I_oR_s}{R_{sh}}$$

dove:

- I_o è la corrente di uscita (A),
- I_{ph} è la corrente fotogenerata (A),
- I_{rs} è la corrente di saturazione inversa del diodo (A),
- q è la carica elementare (1.6×10^{-19}C),
- V_o è la tensione ai capi della cella (V),
- n è il fattore di idealità del diodo,
- k è la costante di Boltzmann (1.38×10^{-23} J/K),
- T_{op} è la temperatura assoluta della cella (K),
- R_s è la resistenza serie (Ω),
- R_{sh} è la resistenza di shunt (Ω).

3.1.2 Equazioni del modello di Tayeb

Tayeb [3] propone un insieme di equazioni per la determinazione dei parametri fondamentali della cella, implementabili in ambiente Matlab/Simulink:

1. Corrente fotogenerata

$$\mathrm{I}_{ph} = \left[I_{sc} + \mathrm{K}_i\left(T_{op} - T_{ref}\right)\right] \cdot \frac{\mathrm{G}}{G_{ref}}$$

2. Corrente di saturazione inversa (condizioni di riferimento)

$$I_{rs} = \frac{I_{sc}}{e^{\frac{qVoc}{N_s knT_{op}}} - 1}$$

3. Corrente di saturazione corretta per la temperatura

$$I_s = I_{rs}\left(\frac{T_{op}}{T_{ref}}\right)^3 \cdot e^{\left[\frac{qE_g\left(\frac{1}{T_{ref}} - \frac{1}{T_{op}}\right)}{Kn}\right]}$$

4. Equazione della corrente di uscita

$$I_o = N_p I_{ph} - I_{rs}\left(e^{\frac{q(V_o + I_o R_s)}{N_s nkT_{op}}} - 1\right) - \frac{V_o + I_o R_s}{R_{sh}}$$

dove I_{sc} è la corrente di corto circuito, V_{oc} la tensione a vuoto, E_g l'energia di gap del semiconduttore, K_i il coefficiente termico della corrente di corto circuito, N_s e N_p il numero di celle in serie e in parallelo. Considerato che i dati sperimentali si riferiscono ad una sola cella, entrambi questi valori sono unitari. Ciononostante, si sono voluti conservare nel modello per un futuro utilizzo flessibile.

3.1.3 Equazioni del modello a due diodi di Warepam

Warepam propone un equivalente più ricco del classico singolo-diodo, per riprodurre sia la curva I–V statica sia le dinamiche (transitorio/frequenza) osservate via EIS (spettroscopia di impedenza elettrochimica).

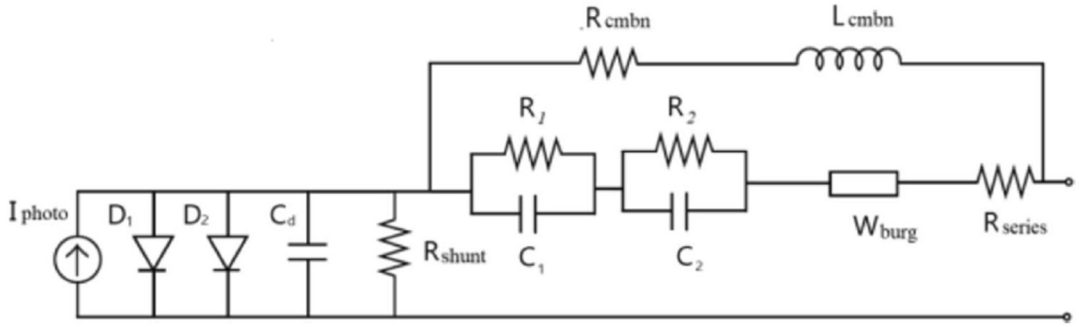

Figura 2: Modello a due diodi secondo Warepam. [2]

Gli elementi principali sono:

- Sorgente di fotocorrente I_{photo}.
- Due diodi: D_1(ricombinazione "principale", idealità n) e D_2 (ricombinazioni addizionali, idealità m). Sono attraversati da una impedenza serie equivalente Z che cattura gli effetti resistivi, induttivi e capacitivi.
- Capacità interfaccia C_d in parallelo a D_1.
- Ramo "serie" fisico del percorso di conduzione: R_{cmbn} + $j\omega L_{cmbn}$ (resistenza di percorso + induttanza che rende conto della risposta induttiva osservata in EIS).
- Rete "shunt" che modella perdite e cinetiche ai contatti: due rami RC (R_1,C_1), (R_2,C_2) un termine di diffusione di W_{burg} e una R_{series} (contributo ohmico addizionale).

Corrente d'uscita (modello a due diodi con impedenza serie Z):

$$I = I_{\text{ph}} - I_i\left(\exp\left(\frac{q\,(V + I\,Z)}{n\,k\,T}\right) - 1\right) - I_r\left(\exp\left(\frac{q\,(V + I\,Z)}{m\,k\,T}\right) - 1\right) - \frac{V + I\,Z}{R_{\text{sh}}}$$

È l'equazione operativa usata da Warepam per la I–V, dove Z è l'impedenza effettiva vista in serie ai diodi. I_i e I_r sono le correnti di saturazione (rettificante e di ricombinazione). Fotocorrente proporzionale all'illuminanza, K_{short} coefficiente termico di I_{sc}:

$$I_{\text{ph}} = \left[I_{\text{short}} + K_{\text{short}}\left(T_{\text{cell}} - T_f\right)\right]\lambda_{\text{inso}}$$

Impedenze "serie" e "shunt":

$$Z = \left(\frac{1}{R_{\text{cmbn}} + j\,\omega\,L_{\text{cmbn}}} + \frac{1}{Z_{\text{sh}}}\right)^{-1}$$

$$Z_{\text{sh}} = \frac{1}{j\,\omega\,C_1} + \frac{1}{R_1} + \frac{1}{j\,\omega\,C_2} + \frac{1}{R_2} + W_{\text{burg}} + R_{\text{series}}$$

Queste espressioni formalizzano due costanti di tempo RC (due "archi" EIS), un termine Warburg di diffusione e la via induttiva del percorso conduttivo (L_{cmbn}).
Dipendenze termiche delle correnti di saturazione:

$$I_i = I_{\text{rev.sat}}\left(\frac{T_{\text{cell}}}{T_f}\right)^3 \exp\left[\frac{q\,E_g}{n\,k}\left(\frac{1}{T_f} - \frac{1}{T_{\text{cell}}}\right)\right]$$

$$I_r = \sqrt{I_{\text{rev.sat}}}\left(\frac{T_{\text{cell}}}{T_f}\right)^{3/2} \exp\left[\frac{q\,E_g}{2\,n\,k}\left(\frac{1}{T_f} - \frac{1}{T_{\text{cell}}}\right)\right]$$

Con:

$$I_{\text{rev.sat}} = \frac{I_{\text{short}}}{\exp\left(\frac{q\,V_{\text{OC}}}{n\,k\,T_{\text{cell}}}\right) - 1}$$

Queste relazioni fissano il legame fra I_i, I_r, temperatura e bandgap (usate per aggiornare i parametri con T_{cell}).

3.1.4 Limiti del modello a un diodo

Nel presente lavoro si è scelto di utilizzare il modello a diodo singolo riproducendo per buona parte il modello Simulink proposto da Tayeb [3], che rappresenta un buon compromesso tra semplicità e accuratezza. Tuttavia, è noto che per applicazioni che richiedono una maggiore aderenza ai dati sperimentali, in particolare nella regione a bassa tensione della curva I-V, è preferibile adottare un modello a due diodi. Quest'ultimo permette di tenere conto in maniera più dettagliata dei fenomeni di ricombinazione e delle perdite parassite (dark current), migliorando l'accordo con le curve misurate. L'estensione verso il modello a due diodi rappresenta pertanto un possibile sviluppo futuro del lavoro di tesi.

3.2 Il super condensatore

I supercondensatori (SC) sono dispositivi di accumulo ad alta densità di potenza e lunghissima vita ciclica. Ai fini di modellazione circuito-equivalente, Cabrane[4] propone e confronta tre livelli di complessità: RC elementare, due-rami (Zubieta–Bonert) e multi-ramo (tipo linea di trasmissione semplificata con rami complementari). Il confronto è svolto in MATLAB/Simulink e validato contro un riferimento sperimentale (Belhachemi) su un Maxwell PC7223 da 2700 F.

3.2.1 Modello RC elementare

Il modello più semplice rappresenta l'SC con: resistenza serie Rsc, capacità Csc e resistenza di perdita Rf (leakage). Fornisce stime rapide di Rsc e Csc da prove a corrente costante, ma non cattura la dinamica di ridistribuzione interna della carica né la dipendenza della capacità dalla tensione. È il più veloce in simulazione, ma il meno accurato nel seguire la tensione misurata durante profili di carica/scarica.

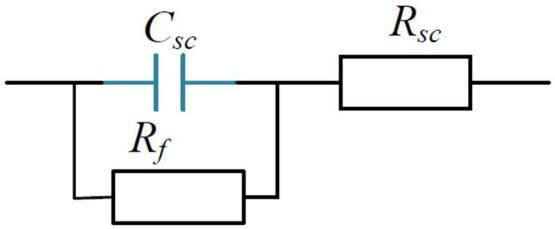

Figura 3: Modello RC del supercapacitore secondo Cabrane. [4]

3.2.2 Modello a due rami (Zubieta–Bonert)

Per riprodurre le due principali costanti di tempo osservate nei supercondensatori reali, il modello a due rami decompone la risposta in un ramo veloce e un ramo lento, oltre a un ramo di perdita R_f.

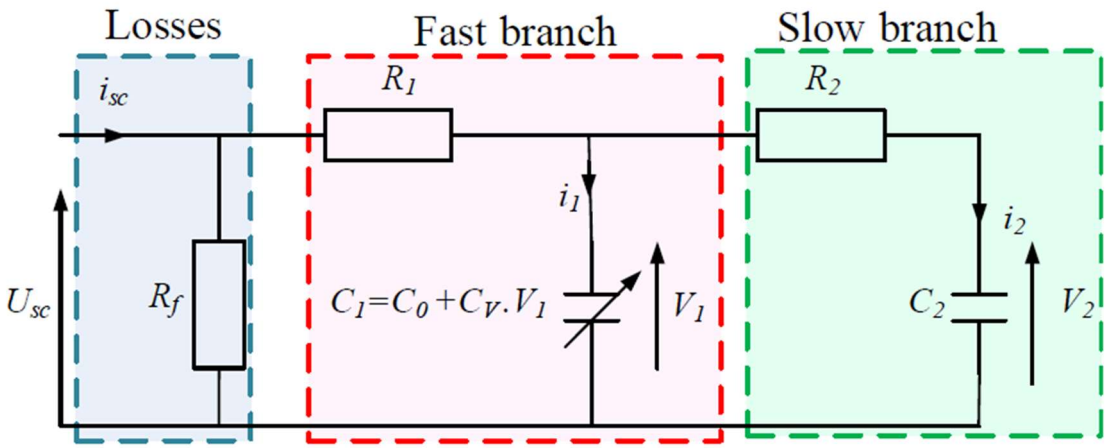

Figura 4: Modello a due rami del supercondensatore secondo Cabrane. [4]

Il ramo veloce introduce una capacità non lineare C_1 in serie a R_1, con:

$$C_1(v_1) = C_0 + C_v v_1,\ i_1 = (C_0 + C_v v_1) \cdot \frac{dv_1}{dt}$$

Il ramo lento R2–C2 modella la ridistribuzione della carica (costante di tempo maggiore), con:

$$v_2 = \frac{1}{C_2} \int i_2\, dt = C \frac{1}{C_2} \int \frac{v_1 - v_2}{R_2} dt$$

La carica del ramo veloce è $Q_1 = C_0 v_1 + \frac{1}{2} C_v v_1^2$, da cui si ricava

$$v_1 = \frac{-C_0 + \sqrt{C_0^2 + 2C_v Q_1}}{C_v}$$

Trascurando R_f (spesso lecito su orizzonti brevi), la tensione all'SC con N_s in serie e N_p in parallelo si esprime come

$$U_{SC} = N_s \left(v_1 + R_1 \frac{I_{SC}}{N_p} \right)$$

Questo modello riproduce bene sia l'immediato salto resistivo sia la fase successiva di rilassamento, mantenendo pochi parametri fisicamente interpretabili (R_1, C_0, C_v, R_2, C_2).

3.2.3 Modello multi-ramo

Per catturare anche la propagazione di carica a tempi brevissimi e un ventaglio più ampio di costanti di tempo, il modello estende il due-rami con:

I. una linea di trasmissione non lineare di n celle in parallelo, a resistenza totale R e capacità C dipendente dalla tensione

II. rami complementari R_m–C_m per i tempi lunghi. In pratica, C, C_2, C_3 sono funzioni a tratti della tensione; valori numerici e leggi C(v) sono forniti per il PC7223, con buoni risultati usando n=15 rami. L'accuratezza è la migliore, ma il costo computazionale cresce sensibilmente.

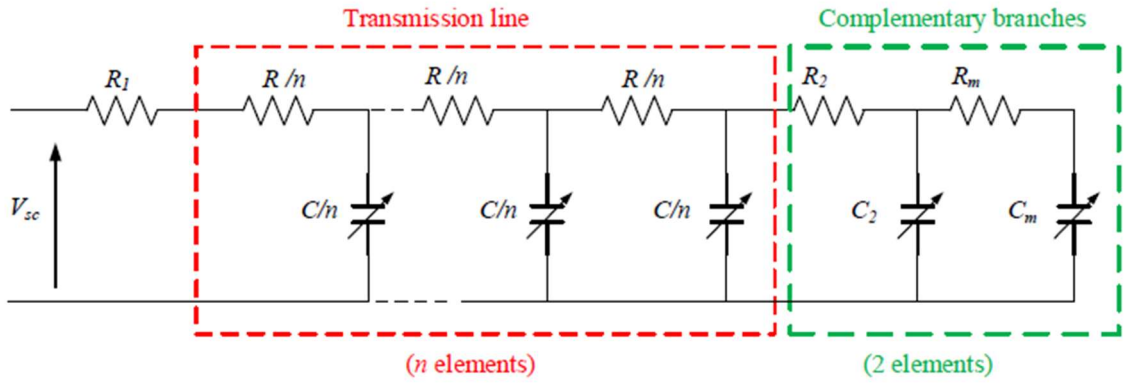

Figura 5: Modello multi-ramo del supercondensatore secondo Cabrane. [4]

3.2.4 Confronto e scelta del modello

Nel confronto su profili a ±100 A, la tensione simulata dal multi-ramo è quella più aderente al riferimento sperimentale (Belhachemi). L'errore assoluto massimo indicativo è ~0,08 V per il multi-ramo, ~0,09 V per il due-rami e ~0,125 V per l'RC elementare. Tuttavia, il multi-ramo richiede tempi di simulazione significativamente maggiori; il due-rami risulta molto più leggero e già sufficientemente accurato per molte applicazioni. Alla luce di questo trade-off, in questo lavoro adottiamo il modello a due rami come buon compromesso tra semplicità di parametrizzazione e implementazione e qualità dei risultati.

4 Algoritmo di fitting

4.1 L'algoritmo di fitting

Per riprodurre in modo accurato il comportamento elettrico della cella fotovoltaica è stato sviluppato un insieme di funzioni MATLAB che automatizzano l'intero processo di stima dei parametri, dall'importazione dei dati sperimentali fino al confronto finale con le curve simulate.
Queste funzioni non si limitano a leggere e rappresentare i dati, ma costituiscono un vero e proprio strumento di modellazione e validazione, poiché:

- estraggono in modo sistematico i parametri fondamentali dai file sperimentali;
- impongono vincoli fisici coerenti con le misure (ad esempio i valori di Isc e Voc);
- implementano una procedura iterativa di ottimizzazione basata sul modello a diodo singolo;
- permettono di confrontare i risultati simulati con quelli reali, visualizzando il grado di accordo con grafici comparativi.

L'intero workflow può essere descritto come una pipeline in cinque fasi principali.

4.1.1 Importazione e organizzazione dei dati sperimentali

Funzione chiave: `importaDatiSperimentali(folderPath)`
Questa funzione analizza automaticamente la cartella contenente i file sperimentali (*.txt*), individuando i dati di ogni curva e ricostruendo una struttura MATLAB dedicata.
Ogni file viene interpretato come composto da due sezioni:

- header dei parametri, contenente valori come area attiva, Isc, Voc, Pmax e FF;
- corpo dati, che riporta le coppie di punti tensione–densità di corrente.

I dati vengono convertiti in unità del Sistema Internazionale (SI) e memorizzati in una struttura (struct) che associa a ciascun dataset i vettori Voltage_V [V], CurrentDensity_mAcm2 [mA/cm²] e Current_A [A].
Contestualmente, la funzione genera una tabella riassuntiva con i valori principali (G, Voc, Isc, FF, ecc.), utile per una verifica preliminare dei dati.

4.1.2 Definizione dell'ottimizzazione

Il fitting viene eseguito per ciascun dataset sperimentale, stimando i tre parametri principali del modello a diodo singolo:

- n
- Rs
- Rsh

L'algoritmo di ottimizzazione, basato sulla funzione `lsqnonlin`, minimizza la somma dei quadrati dei residui tra la corrente simulata e quella misurata:

$$\min_{n,R_s,R_{sh}} \Sigma_k \left[I_{sim(V_k;\, n,R_s,R_{sh})} - I_{\exp(V_k)} \right]^2$$

In questa formulazione, la corrente simulata Isim è ottenuta dal modello Simulink, mentre Iexp rappresenta la corrente sperimentale misurata ai corrispondenti valori di tensione Vk.

I parametri vengono ottimizzati entro limiti fisici plausibili:

$$1.0 \leq n \leq 3.0 \qquad 0 \leq Rs \leq 50\ \Omega \qquad 10^3 \leq Rsh \leq 10^{10}\ \Omega$$

4.1.3 Calcolo del residuo e risoluzione iterativa

La funzione `residual_IV_for_fit_percurve` calcola i residui di corrente confrontando, punto per punto, i valori simulati e sperimentali.
Per ogni curva, viene eseguita una simulazione del modello `PV_SC_Model_IV.slx` tramite la funzione `runSimGetVI`, che restituisce i vettori Vsim e Isim.
La corrente simulata viene poi interpolata sui punti sperimentali per garantire una forma continua e fisicamente coerente. Il processo è iterativo e utilizza guess multipli (definiti in `x0_list`), con fallback automatico nel caso in cui la prima ottimizzazione non converga.

4.1.4 Estrazione e salvataggio dei risultati

Per ogni curva di irraggiamento, vengono salvati i parametri ottimali n,Rs,Rsh, insieme ai valori sperimentali Isc e Voc.
Il risultato finale è una struttura Fitted_Parameters che contiene, per ciascun dataset:

- i parametri stimati,
- il numero di iterazioni del solver,
- l'errore medio quadratico (RMSE),
- i vettori completi Vexp,Iexp,Vsim,IsimV,
- i valori del Fill Factor stimato e sperimentale.

La struttura viene esportata nel workspace e convertita in tabella (Fitted_Parameters_Table) per l'analisi o la visualizzazione in Simulink.

```
>> Caratteristica_IV
✓ Dati sperimentali letti e caricati correttamente.
✓ G = 0      W/m^2 | iterazioni = 16  | Resnorm = 6.563e-06 | RMSE = 3.084e-04 | Stop: Local minimum found.
✓ G = 0.3    W/m^2 | iterazioni = 4   | Resnorm = 2.463e-07 | RMSE = 6.632e-05 | Stop: Local minimum found.
✓ G = 1.5    W/m^2 | iterazioni = 3   | Resnorm = 3.305e-07 | RMSE = 7.186e-05 | Stop: Local minimum found.
✓ G = 3      W/m^2 | iterazioni = 6   | Resnorm = 2.549e-06 | RMSE = 1.951e-04 | Stop: Local minimum found.
✓ I parametri del fitting sono stati salvati in "Fitted_Parameters".
```

Figura 6: Report dell'algoritmo di Fitting.

G [W/m^2]	n [-]	Rs [Ω]	Rsh [MΩ]	Voc [V]	Isc [µA]	FF Simulato	FF Sperimentale
0.0	1.53	1.00	2.24	0.70	0.12	-	-
0.3	2.64	1.83	9.53	0.57	7.40	0.656	0.654
1.5	2.65	7.66	99.70	0.65	38.96	0.684	0.682
3.0	3.00	14.67	3.66	0.69	88.07	0.668	0.665

Figura 7: Risultati finali dell'algoritmo di fitting.

Il codice produce inoltre un grafico comparativo tra curve simulate e sperimentali per ciascun valore di G, evidenziando l'accordo globale tra modello e dati reali.

Plot comparativo: per ciascun dataset, la funzione genera una figura "I–V <nome>", sovrapponendo la curva simulata (linea) e i punti sperimentali (marker). Titoli, assi e legenda sono coerenti con l'unità scelta. Questo grafico è il riscontro visivo dell'aderenza tra modello e realtà: una buona sovrapposizione lungo la regione dal corto (vicino a I_{sc}) al circuito aperto (vicino a V_{oc}) indica che n, R_s, R_{sh} stimati — sotto i vincoli I_{sc} *e* V_{oc} misurati catturano correttamente la fisica del dispositivo.
Di seguito sono riportate i grafici per quattro punti di irraggiamento G = [0, 0.3, 1.5, 3] W/m^2.
Dai grafici riportati si osserva che il modello riproduce in modo accurato il comportamento reale del dispositivo. Il valore del *Fill Factor*, indice della qualità della curva I–V di una cella fotovoltaica, conferma tale coerenza, difatti il valore simulato differisce da quello reale solo di pochi millesimi.

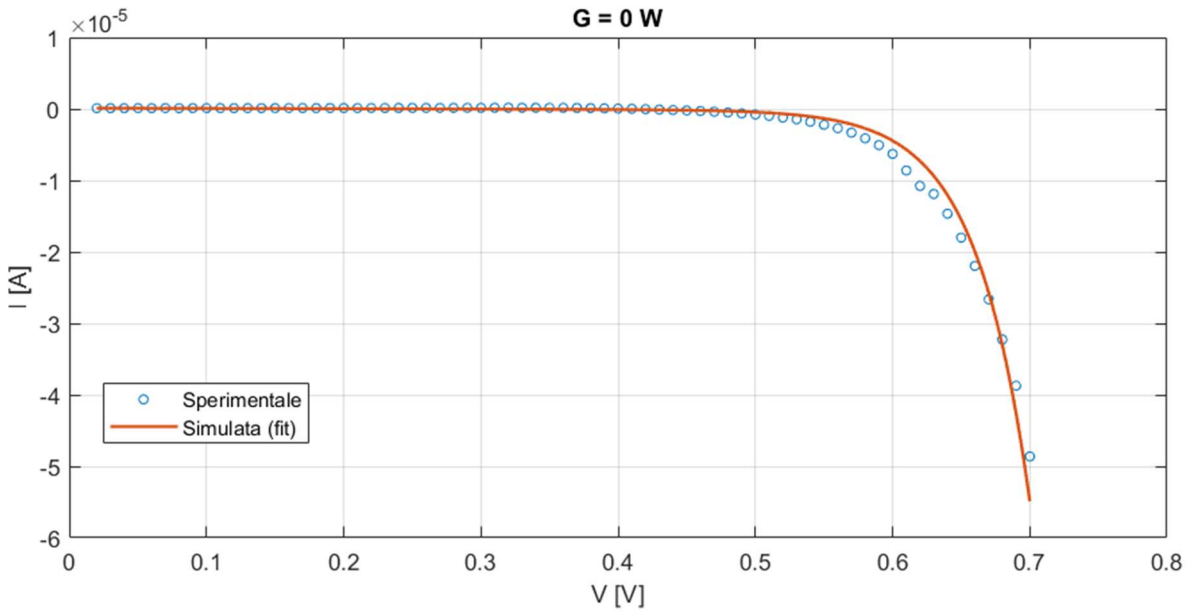

Figura 8: Caratteristica IV con G=0 W/m2

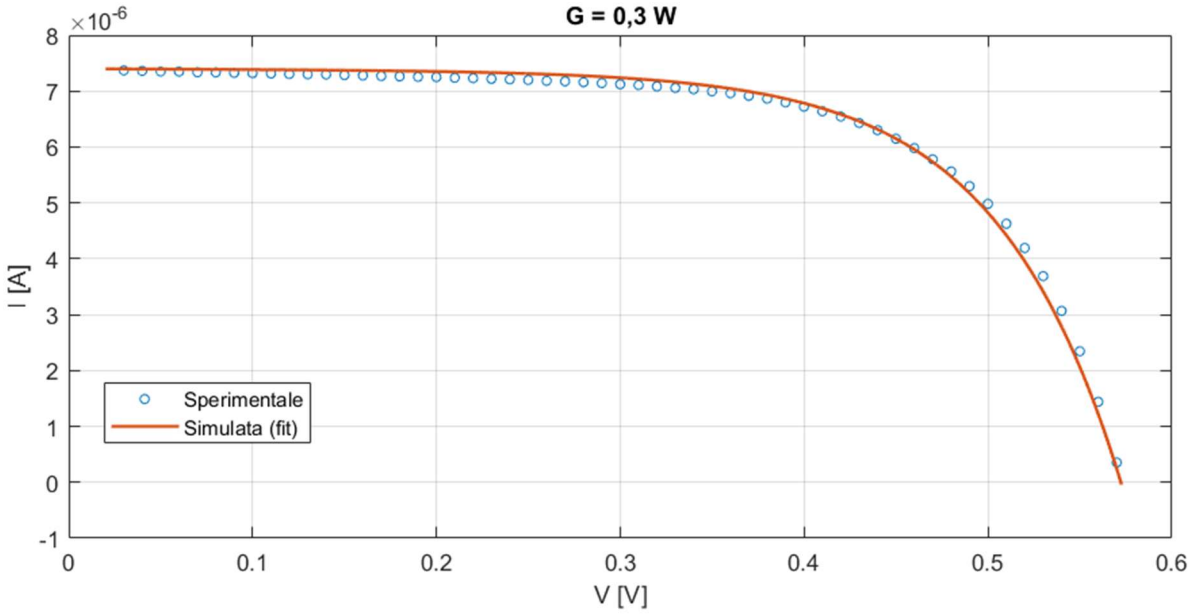

Figura 9: Caratteristica IV con G=0.3 W/m2

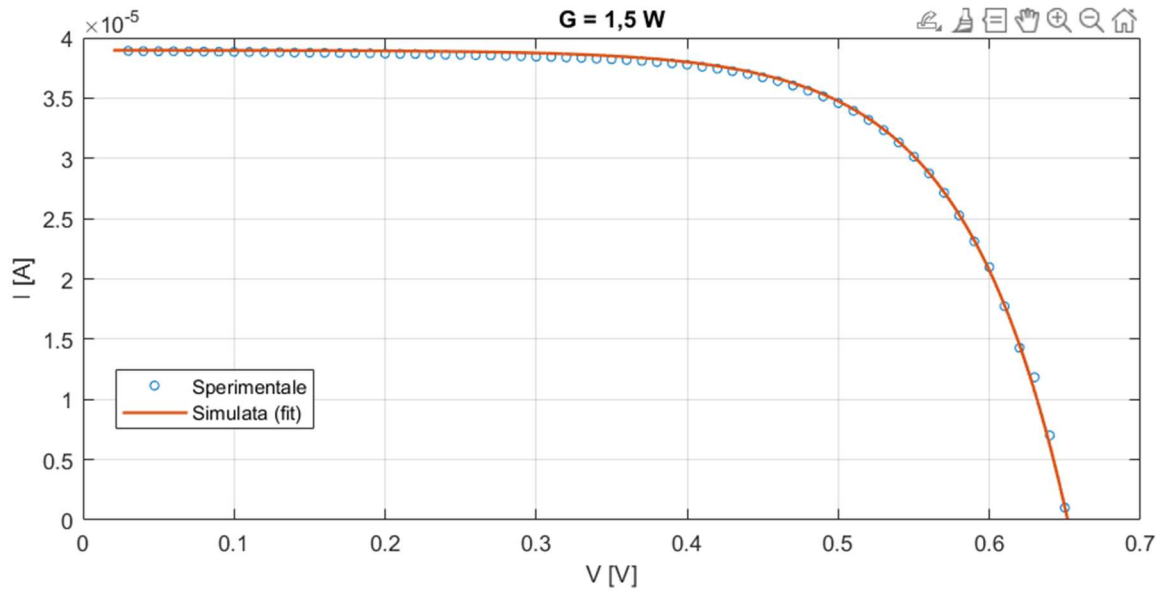

Figura 10: Caratteristica IV con G=1.5 W/m2

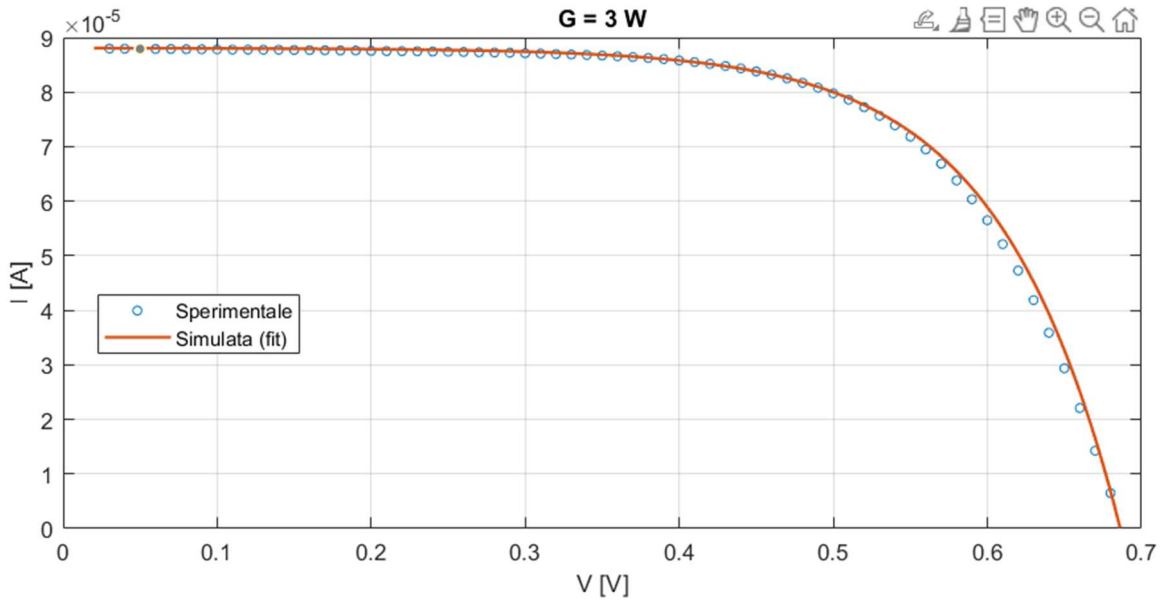

Figura 11: Caratteristica IV con G=3 W/m2

I risultati ottenuti ci confermano che il modello a un diodo, scelto per la cella solare, nonostante non perfettamente accurato, ci permetta di avere risultati sufficientemente validi per simulare il comportamento reale di questo dispositivo e questo conferma nuovamente la nostra scelta di questo modello.

4.2 Il modello dinamico in Simulink

Per comprendere e riprodurre in maniera fedele il comportamento del sistema sperimentale, è stato realizzato un modello numerico capace di integrare le principali componenti fisiche in gioco e di riprodurne le interazioni dinamiche. L'obiettivo è ottenere uno strumento versatile, non solo per interpretare i dati sperimentali, ma anche per prevedere l'evoluzione del sistema in condizioni operative differenti, analizzandone le prestazioni e le criticità.
Il modello si compone di tre elementi fondamentali, ciascuno dei quali è stato descritto, semplificato e parametrizzato in modo da rappresentare i meccanismi fisici principali senza introdurre complessità eccessive.

- Il pannello fotovoltaico, che costituisce la sorgente primaria di energia e che viene descritto attraverso un modello elettrico equivalente basato su equazioni non lineari. La sua funzione è trasformare la radiazione luminosa incidente in energia elettrica, e per questo è necessario caratterizzarne accuratamente le curve I–V e i parametri di funzionamento.
- Il supercondensatore, che svolge il ruolo di accumulatore temporaneo di energia. La sua rappresentazione nel modello tiene conto della capacità effettiva, delle resistenze parassite e dei meccanismi di carica e scarica, cruciali per comprendere la dinamica di accumulo energetico.

- La break out box, che funge da interfaccia di misura e controllo, consentendo di monitorare, condizionare e rendere disponibili i segnali provenienti dalle altre componenti.

Questi tre blocchi sono stati collegati in un'unica architettura che permette di simulare con continuità il comportamento complessivo del sistema. Una particolare attenzione è stata inoltre dedicata all'introduzione di switch di controllo, che assumono un duplice ruolo:

- da un lato permettono di includere o escludere il modello termico, ossia la descrizione della dipendenza della cella fotovoltaica dalla temperatura;
- dall'altro lato rendono possibile gestire l'apertura o la chiusura dell'interfaccia tra pannello e supercondensatore, consentendo così di analizzare scenari differenti a prescindere dalle condizioni operative dell'altro componente.

Grazie a questa struttura modulare e flessibile, il modello costituisce uno strumento utile per l'analisi del sistema e al tempo stesso per la validazione sperimentale, in quanto consente di confrontare in modo diretto i dati simulati con quelli reali.

5 Modello per caratteristica I-V

5.1 Introduzione

La curva I–V è la rappresentazione più immediata delle prestazioni di un dispositivo fotovoltaico: essa permette di individuare parametri chiave come la corrente di corto circuito (Isc), la tensione a vuoto (Voc) e il Fill Factor (FF).

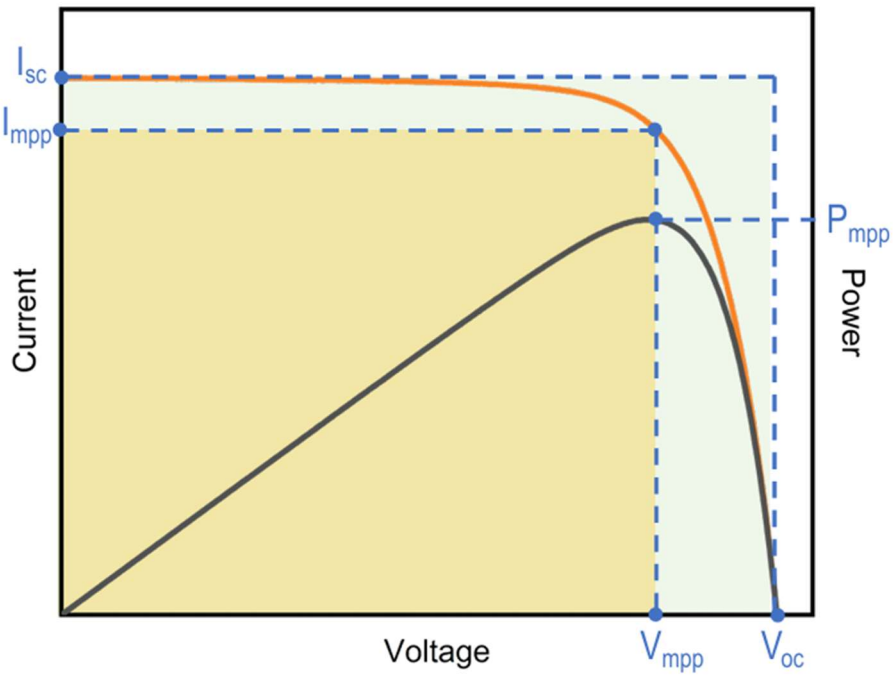

Figura 12: Curva caratteristica I(V) tipica di una cella fotovoltaico, con particolare riferimento ai rettangoli presi in considerazione per il calcolo del Fill Factor [6]

Il Fill Factor (FF) è un parametro fondamentale per la caratterizzazione delle celle fotovoltaiche e misura l'efficienza con cui la cella converte l'energia fotonica in energia elettrica utile.
Esso rappresenta il rapporto tra la potenza massima effettivamente erogata dalla cella e la potenza teorica massima data dal prodotto tra la corrente di corto circuito (I_{sc}) e la tensione a circuito aperto (V_{oc}):

$$FF = \frac{Vmpp \times Impp}{Voc \times Isc}$$

dove:

- V_{mp} = tensione al punto di massima potenza
- I_{mpp} = corrente al punto di massima potenza
- V_{oc} = tensione a circuito aperto
- I_{sc} = corrente di corto circuito

Tuttavia, mentre la misura sperimentale fornisce un'istantanea del comportamento della cella nelle condizioni di test, il modello matematico consente di estendere tale descrizione a scenari diversi, prevedere l'andamento sotto condizioni variabili e integrare il dispositivo in simulazioni più ampie.
Per soddisfare queste esigenze si è fatto riferimento ai modelli circuitali della letteratura. L'implementazione del modello è stata realizzata in ambiente Matlab/Simulink, così da garantire un collegamento diretto con i dati sperimentali.
Questa fase di modellazione rappresenta la base su cui poggia tutto il lavoro successivo: senza una corretta caratterizzazione della cella fotovoltaica, infatti, non sarebbe possibile prevedere in maniera affidabile il comportamento del sistema integrato con il supercondensatore.
Inoltre, il modello I–V non ha soltanto un ruolo di validazione sperimentale, ma funge anche da interfaccia tra il dominio fisico e quello numerico: da un lato conserva il legame con le misure reali, dall'altro fornisce ai blocchi di Simulink un set di parametri coerenti e utilizzabili nelle simulazioni dinamiche.

5.2 Architettura del modello per la caratterizzazione I-V

L'architettura del modello per la caratterizzazione I–V della cella fotovoltaica è stata progettata in ambiente Matlab/Simulink con l'obiettivo di riprodurre in modo fedele i meccanismi fisici principali del dispositivo, mantenendo al contempo una struttura sufficientemente modulare e flessibile da consentire analisi e modifiche successive.
Il modello si compone di tre blocchi funzionali fondamentali:

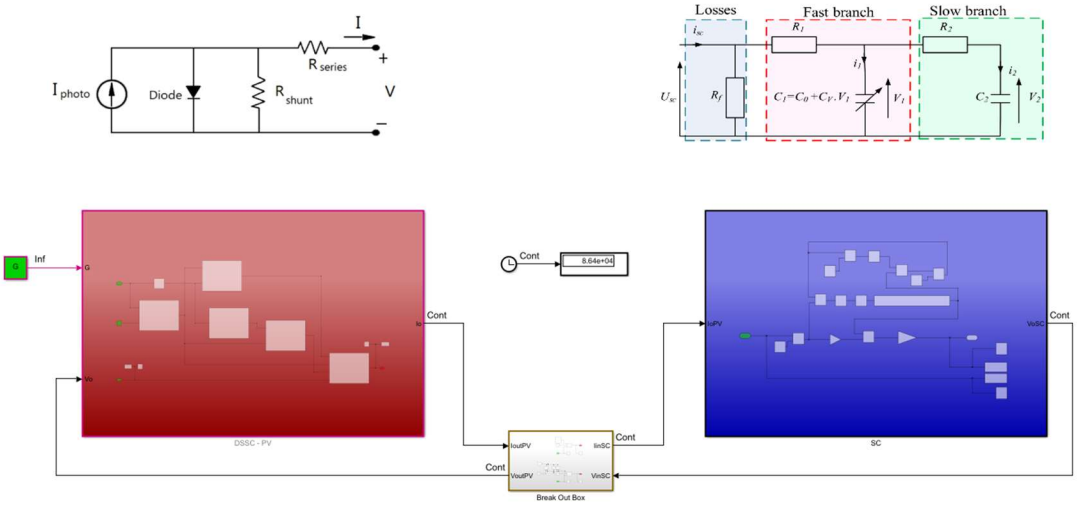

Figura 33: Schema a blocchi del modello I–V in Simulink. A sinistra il sottosistema PV, al centro la break out box che impone la tensione, a destra il sottosistema SC.

5.2.1 Il sottosistema fotovoltaico (PV)

In questo blocco, la cella è rappresentata come una sorgente di corrente fotogenerata in parallelo a un diodo ideale, cui si aggiungono la resistenza serie (*Rs*) e la resistenza di shunt (*Rsh*).

- La corrente fotogenerata (*Iph*) viene calcolata a partire dalla corrente di corto circuito (*Isc*) e dal livello di irraggiamento incidente (*G*).
- Il diodo introduce il termine esponenziale tipico della legge di Shockley, permettendo di riprodurre la dipendenza della corrente dalla tensione ai capi della cella.
- Le resistenze parassite modellano le perdite interne e le dispersioni, elementi fondamentali per avvicinare la simulazione al comportamento reale.

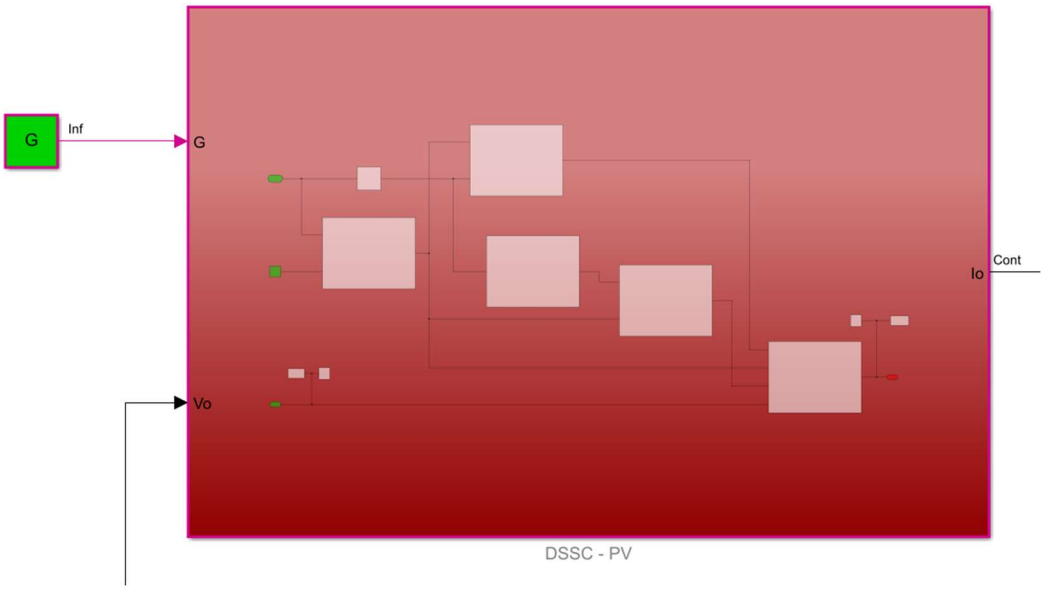

Figura 14: Panoramica esterna del sottosistema della cella fotovoltaica.

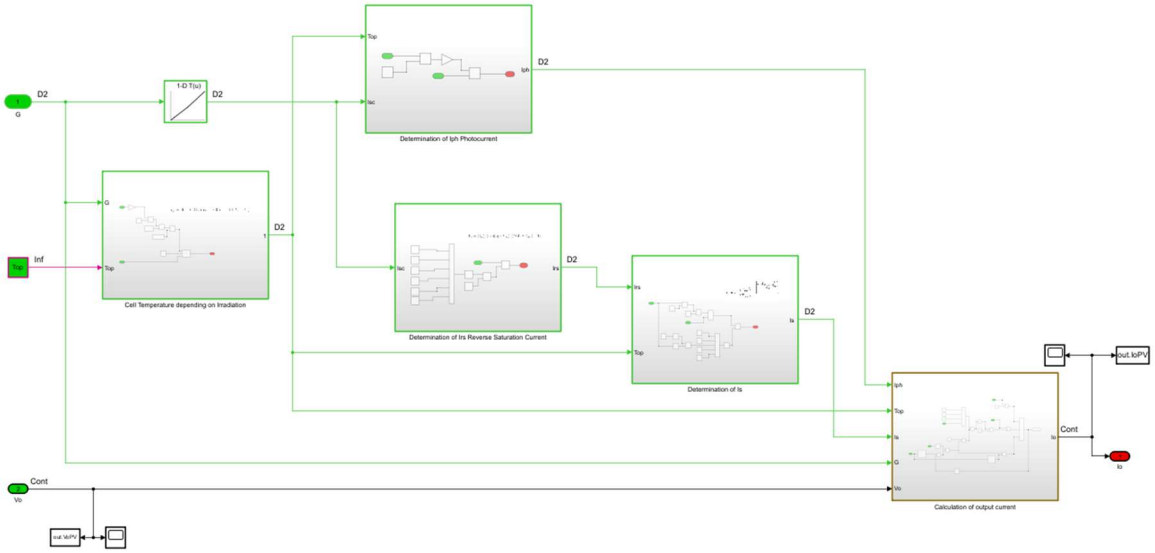

Figura 15: Panoramica del sottosistema della cella fotovoltaica.

Il sottosistema è organizzato in più blocchi funzionali:

- Cell Temperature depending on Irradiation: permette di introdurre la dipendenza della temperatura dalla radiazione solare, modulando così le prestazioni del pannello. Qui è presente uno switch che consente di abilitare o disabilitare il modello termico, così da confrontare simulazioni con e senza effetti di temperatura. La formula empirica è proposta da Kalogirou [5] nel suo manuale di ingegneria di sistemi fotvoltaici e citata per la prima volta da Lasnier nel 1990. Data l'applicazione indoor, il modulo non viene utilizzato in quanto la radiazione considerata produce cambiamenti di temperatura della cella solare trascurabili.

$$T_C = 30 + 0.0175(G_t - 300) + 1.14(T_a - 25)$$

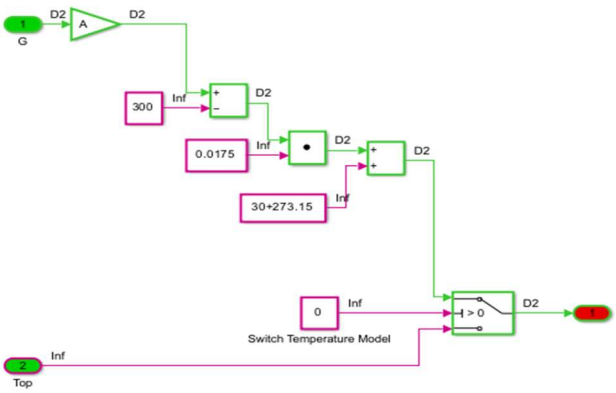

Figura 16: Calcolo della temperatura operativa in funzione dell'irradiazione luminosa.

- Determination of Photocurrent (Iph): calcola la corrente fotogenerata a partire da Isc (corrente di corto circuito) e dall'irradianza G. Questo passaggio è fondamentale perché garantisce che il modello sia ancorato ai valori misurati sperimentalmente, fornendo realismo alle simulazioni. Nel caso di applicazioni indoor come quella trattata in questa tesi, l'influsso della differenza di temperatura con l'ambiente è nullo. Per completezza nel caso di utilizzo del modello in applicazioni future, si è comunque voluto lasciare il sottosistema.

  La corrente Isc non viene scalata linearmente come suggerisce la formula proposta da Tayeb, ma viene interpolata attraverso una lookup table prima di entrare nel sottosistema. Questo è necessario in quanto Isc viene poi utilizzata nei sottosistemi successivi.

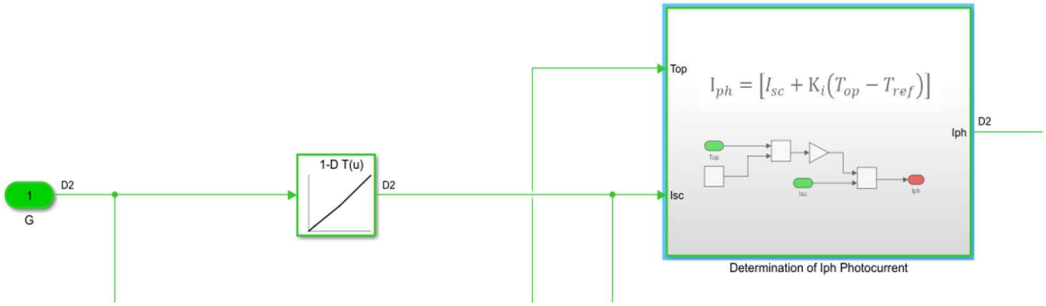

Figura 17: Lookup table da cui viene calcolata Isc in dipendenza di G.

$$I_{ph} = [I_{sc} + K_i(T_{op} - T_{ref})]$$

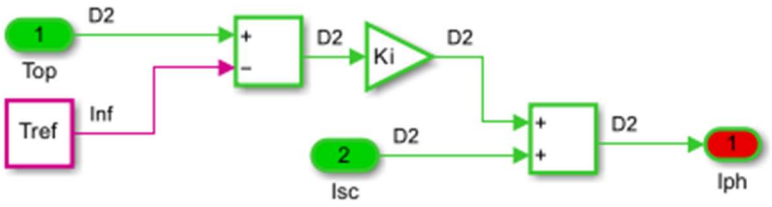

Figura 4: Calcolo della corrente fotogenerata.

- Determination of Reverse Saturation Current (Irs): implementa la stima della corrente di saturazione inversa del diodo, dipendente dai parametri termodinamici (costante di Boltzmann, carica dell'elettrone, energia di gap) e dal fattore di idealità n.

$$I_{rs} = \frac{I_{sc}}{e^{\frac{qVoc}{N_s k n T_{op}}} - 1}$$

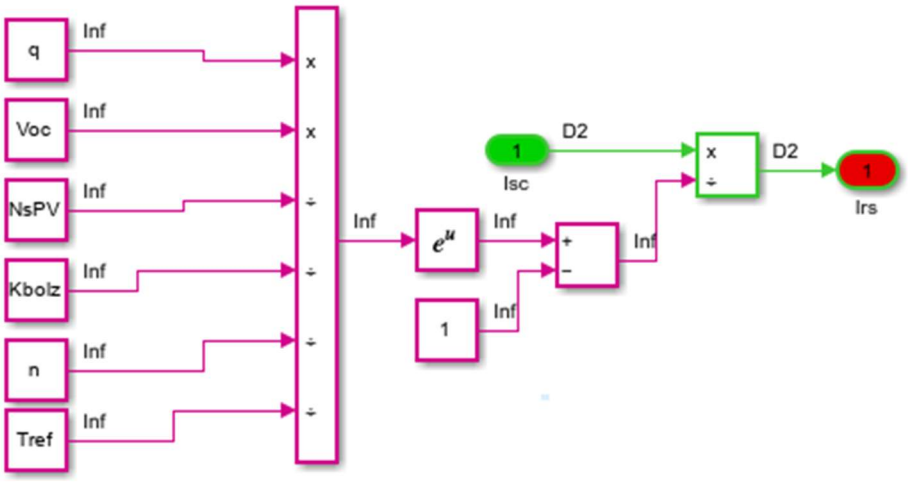

Figura 19: Calcolo della corrente di saturazione inversa.

- Determination of Is: da Irs ricava la corrente di saturazione effettiva, utilizzata nel calcolo della corrente totale del pannello.

$$I_s = I_{rs}\left(\frac{T_{op}}{T_{ref}}\right)^3 \cdot e^{\left[\frac{qE_g\left(\frac{1}{T_{ref}}-\frac{1}{T_{op}}\right)}{Kn}\right]}$$

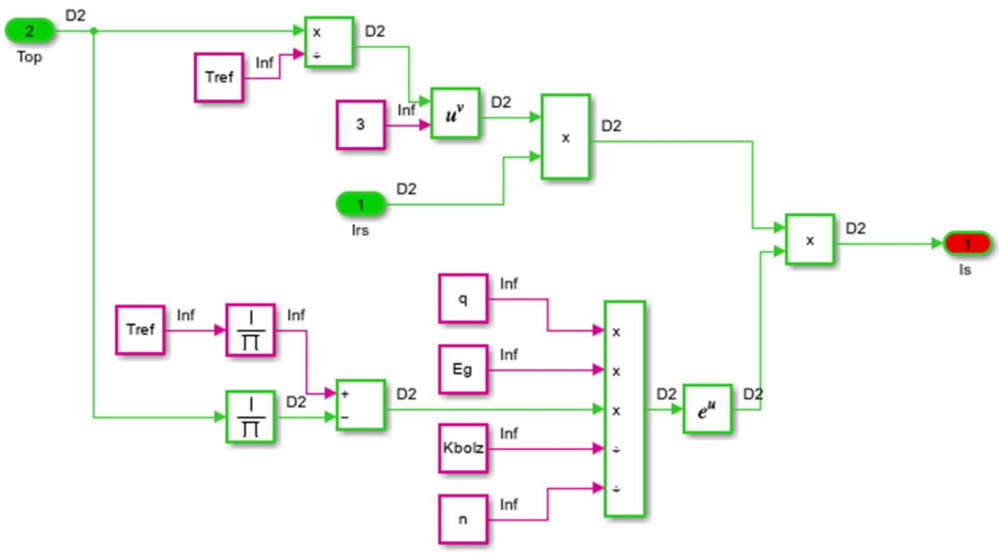

Figura 5: Calcolo della corrente di saturazione effettiva.

- Calculation of output current (Io): calcola la corrente in uscita (Io) combinando la fotocorrente generata (Iph), la corrente di saturazione inversa (Is), la tensione ai capi del dispositivo (Vo) e le resistenze di serie e di shunt. L'equazione del diodo viene implementata in forma discretizzata, con blocchi di somma, prodotto e lookup table, così da consentire sia il calcolo diretto sia eventuali linearizzazioni numeriche per Rs e Rsh. La presenza di Transport Delay preferito dopo varie simulazioni ai blocchi Memory e Unit Delay serve a spezzare loop algebrici che altrimenti renderebbero instabile la simulazione.

$$I_o = N_p I_{ph} - I_s \left(e^{\frac{q(V_o + I_o R_s)}{N_s n k T_{op}}} - 1 \right) - \frac{V_o + I_o R_s}{R_{sh}}$$

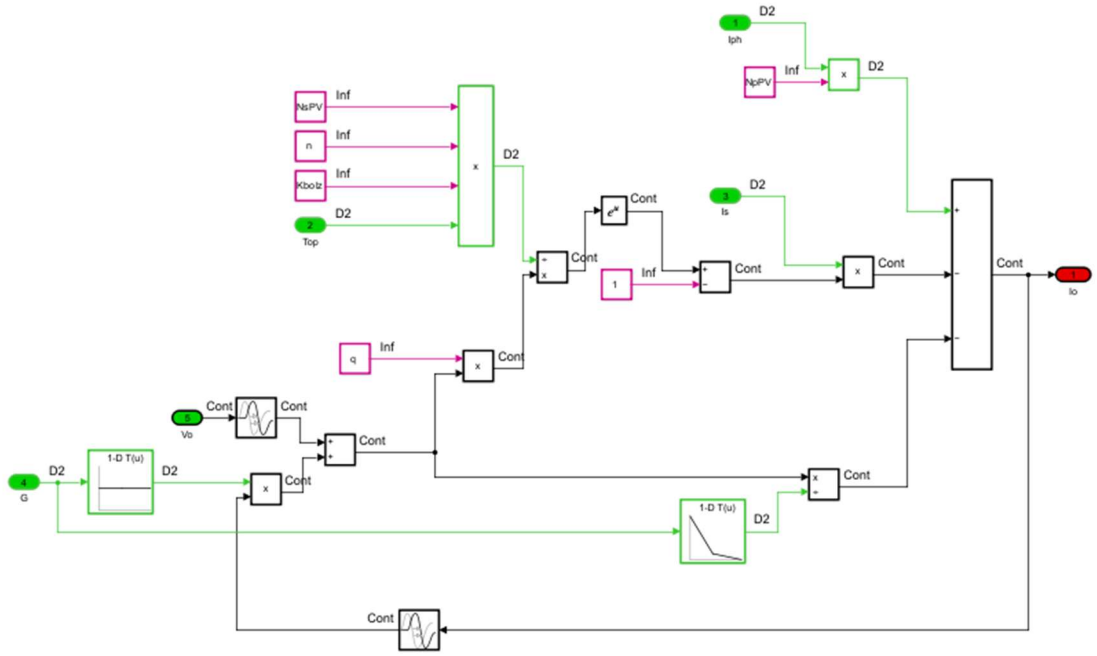

Figura 21: Calcolo della corrente in uscita.

In sintesi, il sottosistema DSSC – PV fornisce al modello la caratteristica I–V del pannello, vincolata a dati sperimentali di Isc e Voc e arricchita da un livello di dettaglio che consente sia simulazioni semplificate sia scenari più complessi con variazioni termiche.

5.2.2 La break out box

Questo sottosistema funge da interfaccia di misura e controllo. Il suo scopo principale è quello di consentire il tracciamento della caratteristica I–V imponendo dall'esterno un range di tensione noto e misurando la corrispondente corrente erogata dalla cella. Attraverso una sorgente di tensione controllata, la break out box forza il pannello a percorrere l'intera gamma operativa, dalla condizione di corto circuito (V ≈ 0) fino a quella di circuito aperto (I ≈ 0).
In questo modo è possibile confrontare direttamente la curva simulata con i dati sperimentali, validando l'accuratezza del modello.
Inoltre, la break out box permette di isolare la cella fotovoltaica dal supercondensatore, semplificando la fase di caratterizzazione e riducendo le variabili in gioco.

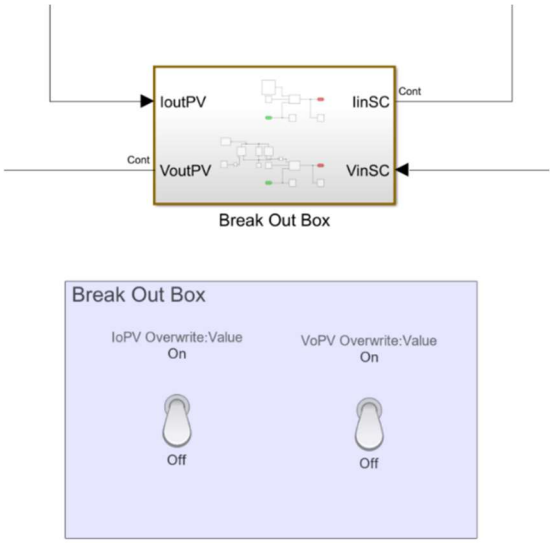

Figura 22: Pannello di controllo della break out box

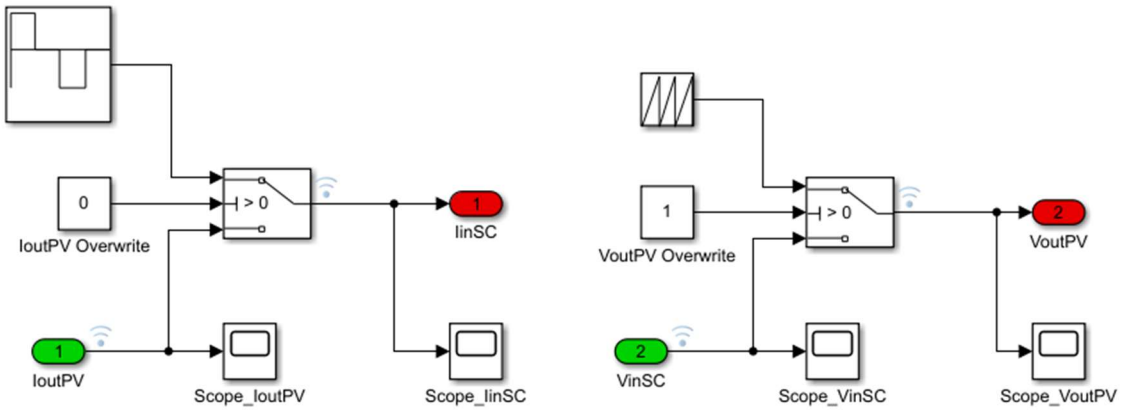

Figura 23: Panoramica del sottosistema break out box.

5.2.3 Il sottosistema supercondensatore (SC)

Il supercondensatore è stato incluso nel modello non tanto per influenzare direttamente la curva I–V, quanto per predisporre l'architettura del sistema completo. In questa fase, infatti, la sua presenza ha un ruolo marginale, ma è importante che la struttura generale lo preveda, così da consentire in un secondo momento il passaggio al modello dinamico integrato.
Il sottosistema SC è stato rappresentato inizialmente tramite un modello semplificato a due rami, in cui:

- Il ramo veloce riproduce la capacità principale e le perdite resistive a breve termine.
- Il ramo lento rappresenta i fenomeni di rilassamento e ridistribuzione della carica.

Questa rappresentazione, ispirata al modello di Cabrane, permette di simulare con un buon livello di realismo la risposta del supercondensatore, pur mantenendo una struttura leggera dal punto di vista computazionale.
Nella fase di caratterizzazione I–V, il sottosistema SC non interagisce direttamente con la cella fotovoltaica, in quanto la break out box spezza il collegamento e impone dall'esterno la tensione operativa. Tuttavia, la sua inclusione nel modello è funzionale alla modularità complessiva: in questo modo, una volta completata la validazione della curva I–V, sarà possibile utilizzare la stessa architettura per integrare la dinamica di accumulo e testare l'alimentazione di un carico reale.

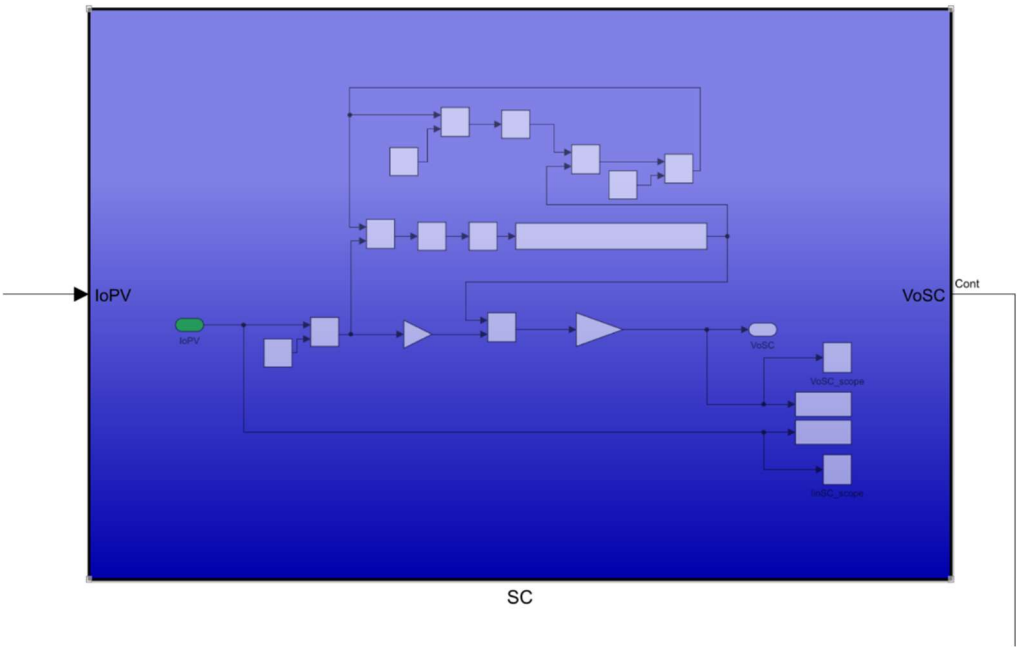

Figura 6: Panoramica esterna del sottosistema supercondensatore.

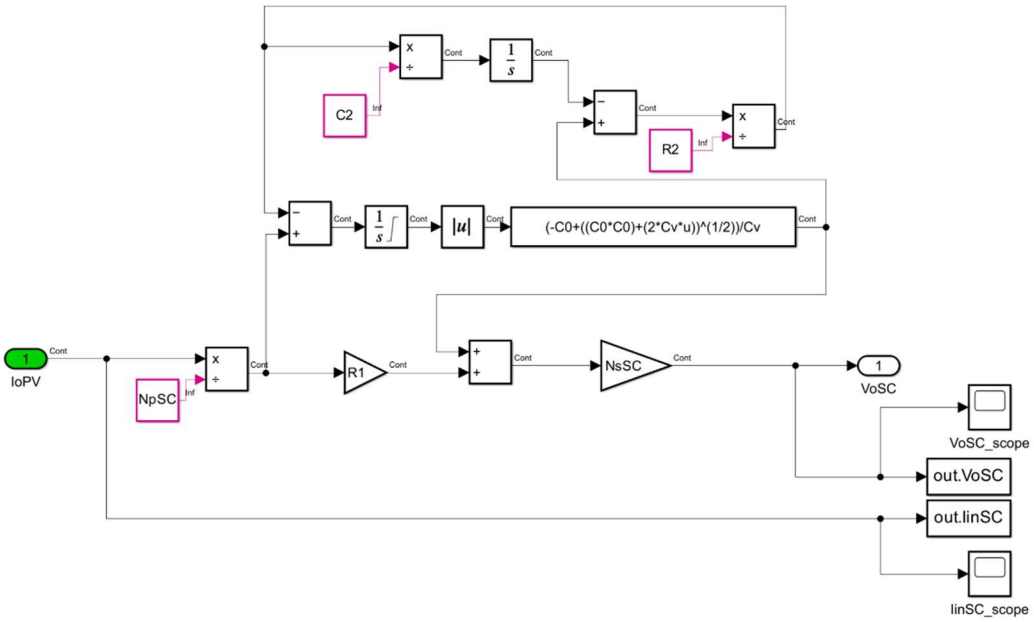

Figura 75: Panoramica del sottosistema supercondensatore.

La modularità di questa architettura si è rivelata particolarmente utile per l'analisi. Infatti, la possibilità di separare i sottosistemi (PV e Supercap) consente di testare singolarmente ciascun componente e di sostituire facilmente una parte del modello qualora si voglia aumentare il livello di dettaglio (ad esempio, passando da un modello a un diodo a uno a due diodi, o introducendo effetti termici).
Un aspetto importante è la gestione dei parametri in ingresso. Voc e Isc non vengono stimati liberamente, ma imposti direttamente dai dati sperimentali. Questo vincolo assicura che la simulazione sia sempre ancorata a grandezze fisicamente misurate, mentre i parametri *n*, *Rs* e *Rsh* sono stimati numericamente tramite fitting.
In sintesi, l'architettura del modello unisce semplicità concettuale e robustezza numerica, fornendo una base solida sia per la validazione contro dati sperimentali, sia per le simulazioni successive in cui il pannello sarà collegato al supercondensatore e a un carico reale.

5.3 Confronto con i dati sperimentali

Una volta determinati i parametri del modello mediante l'algoritmo di fitting, è stato necessario verificare la loro coerenza confrontando i risultati simulati con i dati sperimentali disponibili. Questa fase di validazione è cruciale: senza un confronto diretto, infatti, il modello rischierebbe di rimanere una semplice costruzione numerica priva di legame con la realtà fisica.

5.3.1 Procedura di confronto

Il confronto è stato effettuato in più passaggi successivi:

1. Pubblicazione dei parametri nel workspace
   I valori di Voc, Isc, n, Rs e Rsh ottenuti dal fitting vengono caricati nel workspace di

Matlab e resi disponibili al modello Simulink. Questo permette di riprodurre, in maniera fedele, la curva caratteristica associata alle condizioni di irraggiamento considerate.

2. Simulazione della curva I–V
   Attraverso la *break out box* del modello, viene forzata una tensione ai capi della cella variabile da zero (condizione di corto circuito) fino a un valore superiore a Voc (condizione di circuito aperto). In questo modo si ottiene la curva simulata dell'uscita in corrente.
3. Allineamento delle unità di misura
   Per evitare discrepanze dovute a scelte diverse nelle grandezze, i dati sperimentali e simulati vengono riportati in unità omogenee. Due opzioni principali sono previste:
   - corrente assoluta [A],
   - densità di corrente [mA/cm²].
     In entrambi i casi, i dati sono normalizzati rispetto all'area attiva della cella, così da garantire un confronto diretto.
4. Pulizia e filtraggio
   Durante la simulazione, possono comparire punti non significativi, come correnti negative dovute a oscillazioni numeriche o valori non finiti (NaN). Questi dati vengono eliminati e la curva viene ordinata in funzione della tensione, così da ottenere un andamento regolare e monotono.
5. Generazione dei grafici comparativi
   Infine, i risultati sono presentati sotto forma di grafici sovrapposti:
   - la curva sperimentale è rappresentata tramite punti discreti,
   - la curva simulata è mostrata come linea continua.

Questa scelta grafica consente di valutare immediatamente il grado di accordo tra modello e realtà.

5.3.2 Considerazioni finali

Il confronto con i dati sperimentali ha quindi permesso di:

- Validare il modello numerico
- Quantificare i suoi limiti di accuratezza
- Individuare i margini di miglioramento (ad esempio, l'adozione futura di un modello a due diodi).

Questa fase non è solo un passaggio obbligato di verifica, ma rappresenta anche un momento di riflessione: l'analisi dei punti di accordo e disaccordo tra modello e realtà fornisce indicazioni preziose su quali aspetti fisici siano ben catturati e quali invece richiedano un livello di descrizione più complesso.

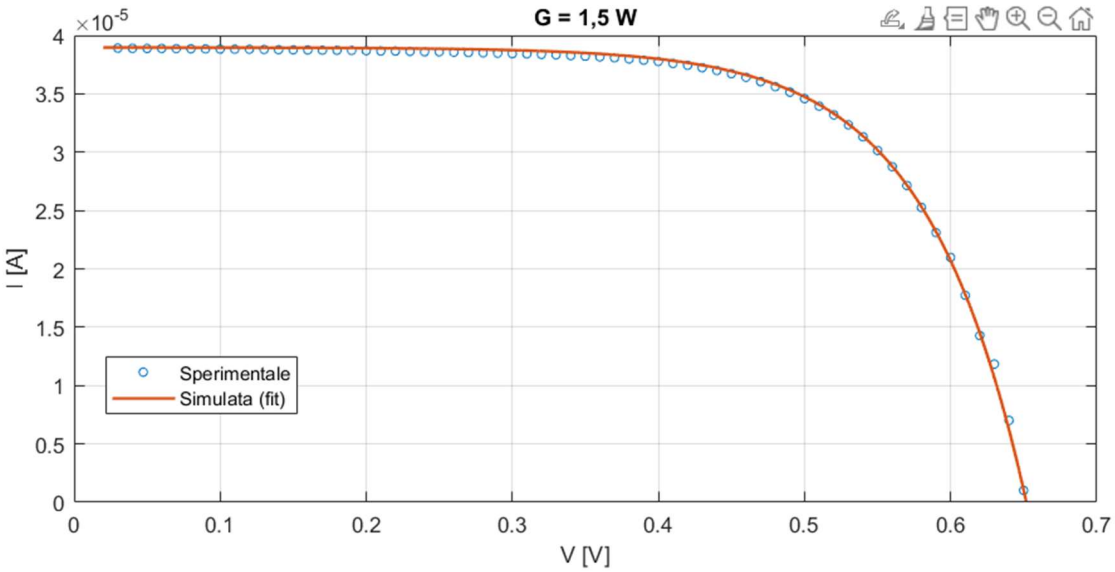

Figura 26: Curva sperimentale vs curva simulata nel caso di G = 1.5 W/m².

5.4 Risultati

L'applicazione dell'algoritmo di fitting e il confronto con i dati sperimentali hanno permesso di valutare l'efficacia del modello a diodo singolo nella riproduzione delle caratteristiche elettriche della cella fotovoltaica. I risultati ottenuti, pur con alcune limitazioni, dimostrano la validità dell'approccio e costituiscono la base per le simulazioni successive con l'intero sistema PV–SC.

5.4.1 Accuratezza della stima dei parametri

Per ciascun livello di irraggiamento sperimentale considerato, l'algoritmo ha restituito un set coerente di parametri. In particolare:

- i valori di n (fattore di idealità) si collocano in un intervallo realistico (1–3), tipico delle celle fotovoltaiche sensibilizzate a colorante;
- la resistenza serie Rs è risultata molto contenuta, come atteso per un dispositivo di piccole dimensioni e buona qualità di contatto;
- la resistenza di shunt Rsh assume valori elevati (10^5–10^7 Ω), indicando perdite limitate e confermando la bontà della fabbricazione.

Questi risultati suggeriscono che il modello è in grado di catturare i fenomeni principali che determinano il comportamento della cella.

5.4.2 Aderenza del modello alle curve I–V

Dalle simulazioni condotte è emerso che il modello a diodo singolo, pur nella sua semplicità, riesce a descrivere in maniera soddisfacente la maggior parte della curva I–V, soprattutto nella regione centrale.

- In prossimità di Isc (V ≈ 0): l'accordo è generalmente buono, poiché il vincolo sul valore di corto circuito costringe la curva simulata a coincidere con quella reale.
- Nella regione intermedia: il fitting è in grado di riprodurre correttamente l'andamento non lineare della corrente, garantendo un errore ridotto.

- Vicino a Voc (I ≈ 0): possono comparire discrepanze più evidenti, dovute al fatto che il modello a un solo diodo non sempre cattura adeguatamente i fenomeni di ricombinazione parassita che caratterizzano questa zona.

Nonostante queste differenze, l'errore complessivo risulta contenuto e la forma globale della curva simulata ricalca con buona fedeltà quella sperimentale. Ciò conferma l'efficacia dell'approccio scelto per l'obiettivo della tesi, ossia disporre di un modello sufficientemente accurato da essere impiegato nelle simulazioni successive.

5.4.3 Discussione dei limiti

Il principale limite osservato riguarda la mancanza di precisione nella regione ad alta tensione. Tale limite è intrinseco nel modello a diodo singolo, che non contempla fenomeni come:

- correnti di ricombinazione addizionali,
- contributi capacitivi o induttivi distribuiti,
- dipendenze più complesse dalla temperatura.

Questi aspetti sono invece trattati in modelli più avanzati, come quello a due diodi o il modello multifisico proposto da Warepam. Tuttavia, tali approcci comportano una maggiore complessità e un costo computazionale più elevato, difficilmente compatibile con le simulazioni dinamiche di lungo periodo richieste da questo lavoro di tesi.

5.4.4 Conclusioni sui risultati

Nel complesso, i risultati ottenuti dimostrano che:

- il modello a diodo singolo, se opportunamente calibrato, fornisce una descrizione realistica del comportamento della cella nelle condizioni operative di interesse;
- l'accuratezza raggiunta è sufficiente per impiegare il modello come base nelle simulazioni con il supercondensatore e il carico associato;
- i margini di miglioramento individuati aprono la strada a futuri sviluppi, in particolare verso l'adozione di modelli più complessi laddove sia richiesto un livello di fedeltà superiore.

In sintesi, questa fase di modellazione e validazione ha permesso di costruire un ponte tra sperimentazione e simulazione, garantendo che le successive analisi dinamiche si fondino su un modello fisicamente consistente e verificato.

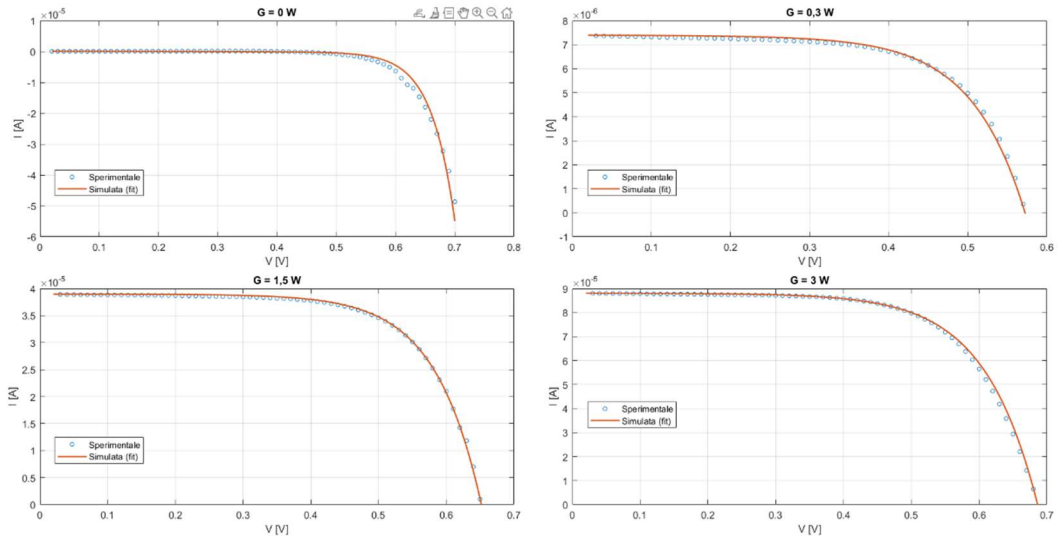

Figura 27: Risultati dell'algoritmo di Fitting.

6 Modello per simulazioni

6.1 Introduzione

Dopo aver sviluppato e validato il modello statico per la caratterizzazione I–V della cella fotovoltaica, il passo successivo è stato quello di estendere l'analisi a uno scenario più realistico, in cui il dispositivo non opera come elemento isolato, ma come parte integrante di un sistema completo. In particolare, l'obiettivo di questa fase è stato lo sviluppo di un modello dinamico in Matlab/Simulink che permetta di simulare l'interazione tra la cella fotovoltaica, il supercondensatore e un carico rappresentativo (ad esempio un sensore a basso consumo).
Mentre la curva I–V fornisce una fotografia del comportamento della cella in condizioni stazionarie e controllate, il modello dinamico consente di studiare l'evoluzione del sistema in funzione del tempo, tenendo conto delle variazioni di illuminazione, delle caratteristiche del carico e non solo. Questa prospettiva è essenziale quando si vuole verificare la possibilità di alimentare in maniera autonoma un dispositivo elettronico in condizioni indoor, dove la radiazione luminosa è tipicamente ridotta, variabile e intermittente.
L'architettura del modello per le simulazioni presenta alcune differenze sostanziali rispetto a quella utilizzata per il tracciamento della curva I–V:

- la *break out box*, che in precedenza serviva esclusivamente a forzare la tensione e registrare la corrente, è stata modificata per emulare il funzionamento del carico;
- il supercondensatore non è più un elemento passivo di confronto, ma viene modellato nella sua interezza come sistema di accumulo, calibrato sui valori ottenuti sperimentalmente e dimensionato per garantire una determinata autonomia di funzionamento;
- il profilo di irraggiamento (*G(t)*) non è costante, ma viene definito come una funzione variabile nel tempo, in grado di riprodurre condizioni luminose realistiche e cicli giorno/notte.

In questo contesto, il modello Simulink *PV_SC_Model_SIM.slx* diventa lo strumento principale per le analisi: attraverso di esso è possibile testare diversi scenari di irraggiamento, configurazioni di collegamento (serie/parallelo) e dimensionamenti dei componenti, così da verificare la fattibilità di un sistema di alimentazione autonomo.
La fase di simulazione non rappresenta soltanto un banco di prova numerico, ma anche una guida progettuale: essa consente di rispondere a domande pratiche come:

- qual è la capacità minima del supercondensatore necessaria per alimentare il sensore durante le ore di buio?
- quale combinazione di celle fotovoltaiche (serie/parallelo) garantisce la tensione e la corrente richieste dal carico?
- quanto influisce il profilo di illuminazione indoor sulle prestazioni complessive del sistema?

In sintesi, lo sviluppo del modello per simulazioni segna il passaggio dalla fase di validazione dei singoli componenti a quella di analisi del sistema integrato, aprendo la strada alla verifica delle reali potenzialità del sistema PV–SC per applicazioni di sensoristica autonoma.

6.2 Architettura del modello per le simulazioni

Il modello per le simulazioni dinamiche è stato sviluppato in Matlab/Simulink con l'obiettivo di rappresentare il comportamento integrato del sistema cella fotovoltaica - supercondensatore - carico. Rispetto al modello utilizzato per la caratterizzazione I–V, questa nuova architettura è stata progettata per analizzare fenomeni tempo-dipendenti, come l'accumulo e il rilascio di energia, nonché per valutare l'autonomia e l'affidabilità del sistema in condizioni operative realistiche.

6.2.1 Componenti principali dell'architettura

L'architettura si compone di tre sottosistemi principali, connessi tra loro tramite la *break out box*:

1. Sottosistema PV (cella fotovoltaica)
    - Mantiene la struttura del modello a diodo singolo, già validato nella fase precedente.
    - Riceve in ingresso un profilo temporale di irraggiamento G(t), definito dall'utente in base allo scenario di interesse (ad esempio un ciclo giorno/notte o un pattern tipico di illuminazione indoor).
    - Restituisce in uscita la corrente generata, che costituisce la sorgente primaria di energia del sistema.
    - Sono previsti parametri configurabili come il numero di celle in serie/parallelo, per adattare la tensione e la corrente del generatore fotovoltaico.
2. Sottosistema SC (supercondensatore)
    - Implementa il modello a due rami (Zubieta–Bonert), scelto come compromesso tra accuratezza e complessità computazionale.
    - Il ramo veloce riproduce la capacità non lineare e la resistenza serie, responsabili della risposta immediata del dispositivo.
    - Il ramo lento descrive i processi di rilassamento e ridistribuzione della carica, che influenzano l'autonomia su tempi più lunghi.
    - Sarebbe dovuto essere presente anche un ramo di perdita (*Rf*), che simula le correnti di leakage, ma queste ultime sono così piccole rispetto agli ordini di grandezza del sistema da non fare praticamente differenza, per cui è stato scelto di non inserirlo.
    - I parametri del supercondensatore non sono arbitrari: essi vengono derivati dalla fase di calibrazione (Capitolo 6.3), che garantisce coerenza tra modello e specifiche di progetto.
3. Sottosistema Break Out Box
    - In questa versione del modello assume una funzione differente rispetto al caso della curva I–V.
    - Oltre a collegare pannello e supercondensatore, la break out box integra la logica di funzionamento del carico.
    - Il carico è rappresentato come un sensore a basso consumo, attivo solo entro un determinato intervallo di tensione (tipicamente 1.8–3.3 V).

La scelta di un carico a basso consumo, attivo solo entro un intervallo di tensione compreso tra 1.8 V e 3.3 V, è motivata dal fatto che tale tipologia di dispositivo rappresenta in modo realistico le applicazioni tipiche dei fotocapacitori, come sensori autonomi o nodi IoT alimentati da energia solare. Questo intervallo di tensione corrisponde alla finestra di funzionamento di molti circuiti a bassa potenza (microcontrollori, moduli di comunicazione, sensori digitali), e consente di valutare il comportamento del sistema in condizioni operative concrete, evitando zone in cui il dispositivo non sarebbe in grado di operare correttamente.
    - Al di sotto della tensione minima, il carico risulta spento, mentre al di sopra della massima viene protetto dal sistema per evitare danni.
    - Questa configurazione consente di valutare non solo il comportamento dei componenti, ma anche la capacità del sistema di alimentare concretamente un dispositivo reale.

6.2.2 Gestione dei parametri di simulazione

Un aspetto centrale dell'architettura è la possibilità di definire e variare facilmente i parametri di simulazione:

- Profilo di irraggiamento G(t): può essere impostato tramite vettori *G_vals* e *G_time*, che permettono di simulare diverse condizioni di luce (costanti, cicliche, casuali).
- Configurazione del generatore PV: numero di celle in serie/parallelo.
- Configurazione del supercondensatore: numero di unità in serie/parallelo, parametri capacitivi e resistivi calibrati.
- Carico: corrente assorbita e intervallo di tensione operativo.

Questa flessibilità rende il modello adatto a testare diversi scenari di progetto e a condurre analisi comparative tra soluzioni alternative.

6.2.3 Vantaggi dell'architettura

- Modularità: ogni sottosistema è indipendente e può essere sostituito o modificato senza compromettere l'intera struttura.
- Realismo: l'introduzione del carico e del profilo temporale di irraggiamento avvicina il modello a uno scenario operativo concreto.
- Efficienza: il compromesso tra accuratezza e complessità (ad esempio la scelta del modello a due rami per il supercondensatore) permette di ottenere simulazioni significative senza eccessivi tempi di calcolo.

In sintesi, l'architettura del modello per simulazioni rappresenta un passo decisivo rispetto al semplice tracciamento della curva I–V: essa consente di passare da un'analisi statica dei singoli componenti a una valutazione dinamica del sistema integrato, condizione necessaria per studiare la fattibilità di un'alimentazione autonoma per dispositivi elettronici a basso consumo.

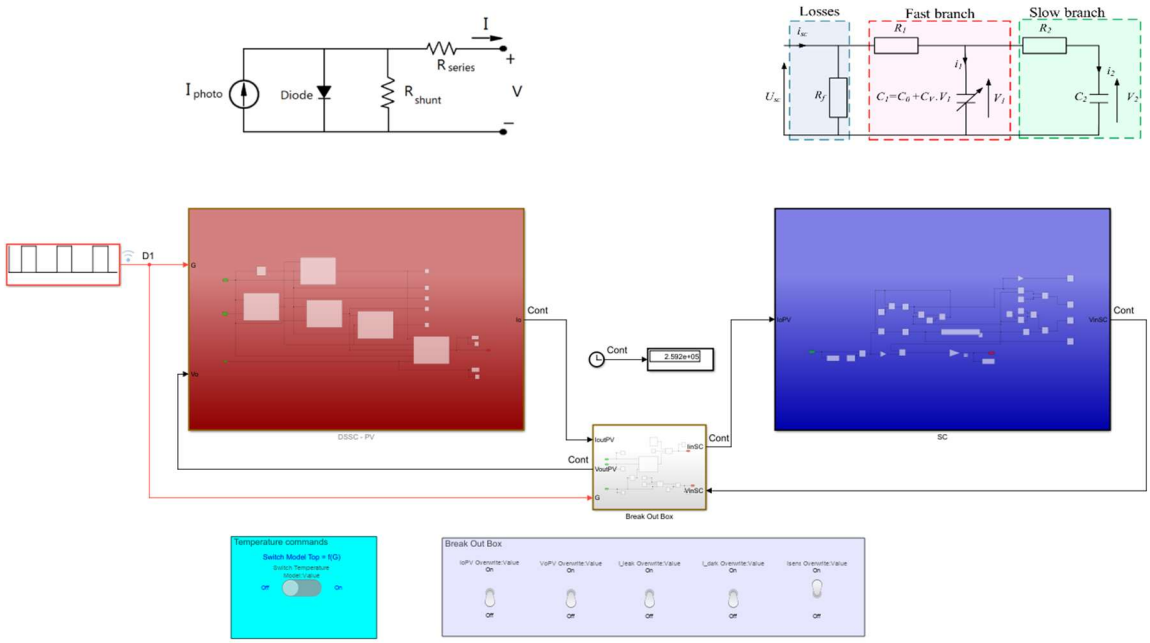

Figura 28: Schema a blocchi del modello per simulazioni in Simulink. A sinistra il sottosistema PV, a destra il supercondensatore, al centro la break out box con il pannello di controllo.

6.3 Calibrazione del supercapacitore

Il supercondensatore è il cuore del sistema di accumulo: la sua capacità di immagazzinare e rilasciare energia determina in larga misura l'autonomia del dispositivo alimentato. Per poter rappresentare correttamente il suo comportamento all'interno del modello Simulink è stato necessario sviluppare una fase di calibrazione, mirata a ridimensionare i parametri del modello circuito-equivalente in modo coerente con i requisiti progettuali. Mentre nel caso della calibrazione della cella fotovoltaica si è scelto di usare dei dati sperimentali, nel caso del supercapacitore si è scelto di utilizzare i dati proposti dal paper di Cabrane [4], in quanto si è reputato ci fosse una funzionale relazione tra le componenti e le costanti di tempo. Per questo la calibrazione del supercapacitore consiste nello scalare di un fattore k queste componenti in modo da mantenere costante la relazione tra le componenti e lasciare invariate le costanti di tempo.

6.3.1 Obiettivo della calibrazione

L'obiettivo principale è stato dimensionare il supercondensatore affinché fosse in grado di:

- Garantire un intervallo operativo di tensione compreso tra 1.8 V e 3.3 V, corrispondente ai limiti di funzionamento tipici di un sensore a basso consumo.
- Erogare una corrente costante al carico.
- Mantenere l'alimentazione per una durata prefissata (Autonomy), per esempio pari a circa 17 ore, in assenza di radiazione luminosa.

A partire da queste specifiche, è stato possibile calcolare la carica minima necessaria e, di conseguenza, la capacità equivalente richiesta.

6.3.2 Procedura di calibrazione

La calibrazione è stata realizzata attraverso uno script Matlab dedicato C*alibrazione_supercap.m*, lo script si occupa di:

1. Calcolo della carica richiesta

$$Q_{target}[C] = I_{load}[A] \cdot Autonomy\ [s]$$

dove nel caso standard Iload è la corrente media necessaria per alimentare il carico ed Autonomy è il tempo di autonomia desiderato in secondi senza considerare le perdite dovute al leakage o alla dark current.

2. Determinazione della capacità necessaria
   Poiché il supercondensatore deve garantire un'escursione di tensione da:

$$V_{max} = 3.3\,V \;\; a\; V_{min} = 1.8V$$

   La capacità minima è data da:

$$C_{need} = Q_{target} \cdot (V_{max} - V_{min})$$

3. Scalamento dei parametri del modello
   Il modello scelto (a due rami, secondo Zubieta–Bonert) rappresenta il supercondensatore tramite:
   - una capacità principale C0,
   - una capacità non lineare Cv,
   - una seconda capacità C2,
   - resistenze R1 e R2 associate ai rami veloce e lento,
   - una resistenza di perdita Rf.

I valori iniziali, tratti dal lavoro di Cabrane, sono stati scalati con un fattore k calcolato come rapporto tra la carica richiesta e quella fornita dal modello di partenza.

$$k = \frac{Q_{max}}{Q_{old}}$$

Tutte le capacità sono state moltiplicate per k, mentre le resistenze sono state divise per lo stesso fattore, in modo da mantenere inalterate le costanti di tempo caratteristiche del sistema.

4. Ricalcolo delle costanti di tempo
   Le nuove costanti di tempo associate ai rami veloce e lento sono state determinate come:

$$\tau_1 = R_1 \cdot (C_0 + C_v \cdot V_{max})$$

$$\tau_2 = R_2 \cdot C_2$$

Questi parametri consentono di verificare che la dinamica di carica e scarica rimanga coerente con quella di un supercondensatore reale.

6.3.3 Risultati della calibrazione

L'applicazione della procedura ha portato a un insieme di parametri "scalati" che rispondono alle specifiche del progetto. Tra i valori principali si riportano:

- Capacità equivalente richiesta
- Resistenze ridimensionate: R1 e R2
- Capacità distribuite: C0, Cv, C2 con valori compatibili con la risposta di un supercondensatore commerciale

Questi parametri sono stati successivamente trasferiti nello script di simulazione (*Simulatore.m*), nella sezione dedicata ai "parametri calibrati del supercapacitore".

6.3.4 Importanza della calibrazione

La calibrazione non è un mero esercizio numerico, ma rappresenta un passaggio fondamentale per garantire la coerenza fisica del modello. Senza questo adattamento, il comportamento simulato del supercondensatore risulterebbe distante dalle condizioni reali di utilizzo, compromettendo l'affidabilità delle analisi successive.
Grazie a questa procedura, il modello finale è in grado di:

- riprodurre correttamente il processo di carica/scarica,
- rispettare i limiti operativi di tensione del sensore,
- fornire previsioni realistiche sull'autonomia del sistema.

```
%%% ! SCEGLIERE QUI I PARAMETRI TARGET DEL SUPERCAPACITORE ! %%%

Vmax = 3.3;           % Tensione massima di operatività del carico [V]
Vmin = 1.8;           % Tensione minima di operatività del carico [V]
Iload = 10e-6;        % Corrente richiesta dal carico [A]
Autonomy = 17*3600;   % Autonomia desiderata nell'alimentazione del carico in un caso ideale senza perdite [s]

Q_target = Iload * Autonomy;   % Carica minima necessaria del supercap per soddisfare l'autonomia richiesta [C]

C_need = Q_target/(Vmax-Vmin); % Capacità necessaria per mantenere l'autonomia nella dinamica del carico
```

Figura 29: Definizione dei parametri target del supercapacitore.

	Iload[I]	Autonomy[ore]	Vmax[V]	Q_target[C]	Qmax[C]	C_need[F]
chosen parameters	1e-05	17	3.3	0.612	1.3464	0.408

```
Copiare e incollare questi parametri su Simulazioni.m
nella sezione: --- Parametri calibrati del supercap (SI) ---
-------------------------------------
C_need = 0.408;
R1 = 62.3;
R2 = 7789.2;
C0 = 0.279;
Cv = 0.067;
C2 = 0.019;
-------------------------------------
```

Figura 30: Risultato della calibrazione.

6.4 Scelta dei parametri di simulazione

Una volta calibrato il supercondensatore, è stato necessario definire con precisione l'insieme dei parametri utilizzati per le simulazioni dinamiche. Questa fase ha un ruolo centrale, perché determina le condizioni operative del sistema e consente di valutare, attraverso scenari differenti, la capacità della configurazione PV–SC di alimentare in modo affidabile un carico a basso consumo.

6.4.1 Parametri di ingresso

1. Profilo di irraggiamento G(t)
    - È uno degli elementi più critici del modello, poiché la disponibilità di energia fotovoltaica dipende direttamente dall'intensità luminosa.
    - In ambiente indoor, l'irraggiamento non è costante e varia in base alla sorgente artificiale, all'orario di utilizzo e alla posizione del dispositivo.
    - Nei file di simulazione (*Simulatore.m*), G(t) è definito tramite due vettori:
        - *G_vals*: valori discreti di irraggiamento in W/m².
        - *G_time*: istanti temporali (in secondi) corrispondenti ai cambiamenti di livello.
    - Questa formulazione permette di rappresentare scenari realistici come cicli giorno/notte, variazioni rapide di luce o periodi di oscurità prolungata.
2. Configurazione del generatore fotovoltaico (PV)
    - Il modello consente di variare il numero di celle in serie (NsPV) e in parallelo (NpPV).
    - Le celle in serie aumentano la tensione disponibile, mentre quelle in parallelo incrementano la corrente erogabile.
    - Nel caso specifico, la configurazione di base adottata è stata di almeno cinque celle in serie, scelta per ottenere un livello di tensione compatibile con l'intervallo operativo del carico.

3. Configurazione del supercondensatore (SC)
    - Anche per il supercondensatore è possibile definire il numero di unità in serie (NsSC) e in parallelo (NpSC).
    - Le unità in serie innalzano la tensione massima sopportabile, a scapito della capacità, mentre quelle in parallelo aumentano la capacità mantenendo invariata la tensione operativa.
    - Per le simulazioni di riferimento è stata mantenuta una configurazione elementare (uno in serie, uno in parallelo), così da lavorare direttamente con i valori calibrati nella fase precedente.
4. Parametri del carico
    - Il carico è stato modellato come un sensore elettronico a basso consumo.
    - La sua richiesta è stata fissata costante con vincoli di tensione operativa compresi tra 1.8 V e 3.3 V.
    - Al di fuori di questo intervallo, il carico risulta spento o in protezione.

6.4.2 Importanza della scelta dei parametri

La possibilità di configurare liberamente questi parametri rende il modello estremamente flessibile e adattabile a scenari diversi. È infatti possibile:

- testare il sistema sotto condizioni ottimali (irraggiamento costante e abbondante),
- simulare ambienti più critici (irraggiamento ridotto o variabile),
- esplorare diverse configurazioni di PV e SC per trovare il miglior compromesso tra prestazioni, dimensioni e stabilità dell'alimentazione.

In questo modo, la fase di scelta dei parametri non è solo preparatoria alla simulazione, ma costituisce un vero e proprio strumento di progettazione, che permette di adattare il sistema alle esigenze applicative specifiche.
Nella figura seguente è visibile il profilo di irraggiamento G(t) che verrà utilizzato per la maggior parte delle successive simulazioni.

- G = 3 dalle 8.00 alle 18.00 (per tre giorni consecutivi)
- G = 0 altrimenti.

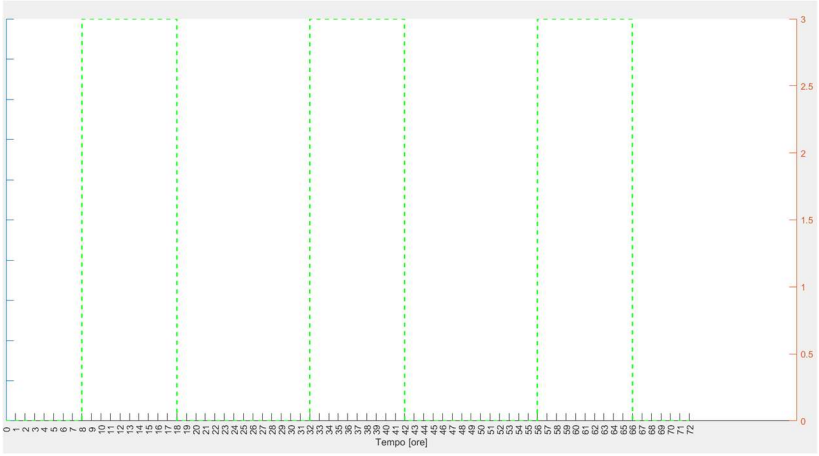

Figura 31: Esempio di profilo di irraggiamento G(t) utilizzato per le simulazioni. Sono riportati i livelli di G in W/m² e i corrispondenti istanti temporali.

6.5 Simulazione e risultati

Una volta definiti e calibrati i parametri, il modello integrato PV-SC-carico è stato utilizzato per eseguire diverse simulazioni numeriche in ambiente Matlab/Simulink. L'obiettivo era verificare se, nelle condizioni di progetto, il sistema fosse in grado di garantire un'alimentazione stabile e continua al sensore, anche in presenza di variazioni dell'irraggiamento.

6.5.1 Esecuzione della simulazione

Le simulazioni sono state condotte su un periodo maggiore di 48 ore, in modo da coprire più cicli di carica e scarica.
L'output del modello ha restituito l'andamento nel tempo delle principali variabili:

- Profilo di irraggiamento (G).
- Corrente in ingresso al supercondensatore (IinSC).
- Tensione ai capi del supercondensatore (VinSC).
- Stato operativo del carico (acceso/spento).

Elaborazione e rappresentazione dei risultati
I risultati sono stati visualizzati tramite grafici che mostrano, in un'unica figura, l'andamento di G(t), IinSC e VinSC. Questo consente di osservare in maniera immediata la correlazione tra illuminazione, accumulo di energia e alimentazione del carico.

6.5.2 Risultati principali

- Comportamento del supercondensatore:
  Durante le fasi di irraggiamento elevato, il supercondensatore si carica rapidamente. Nelle fasi di buio, la tensione decresce gradualmente a causa della corrente assorbita dal sensore, dalla corrente di leakage e dalla dark current. L'obbiettivo di base è di calibrarlo in modo che possa mantenere una tensione sufficiente ad alimentare il sensore fino alla prossima fase di irraggiamento.
- Stabilità dell'alimentazione:
  Il sensore rimane operativo per gran parte della simulazione. Solo in condizioni particolarmente critiche (assenza di luce per periodi molto prolungati) la tensione scende al di sotto della soglia minima, causando lo spegnimento temporaneo del carico. Questo comportamento è atteso e rappresenta il limite fisico del sistema.
- Influenza della configurazione PV e SC:
  È stato osservato che aumentando il numero di celle fotovoltaiche o la capacità equivalente del supercondensatore si ottiene un incremento dell'autonomia, ma a fronte di un maggiore ingombro e complessità del sistema. La configurazione scelta rappresenta quindi un buon compromesso tra prestazioni e semplicità.

6.5.3 Discussione

I risultati ottenuti confermano la validità del modello sviluppato:

- il comportamento dinamico riproduce fedelmente l'alternanza di fasi di carica e scarica osservabili in un sistema reale;

- il supercondensatore, se dimensionato correttamente, è in grado di garantire un'alimentazione stabile al sensore per un'intera giornata operativa indoor;
- il modello permette di esplorare in anticipo scenari diversi, fornendo indicazioni utili per il design di sistemi autonomi destinati all'Internet of Things, al monitoraggio ambientale o ad applicazioni biomedicali portatili.

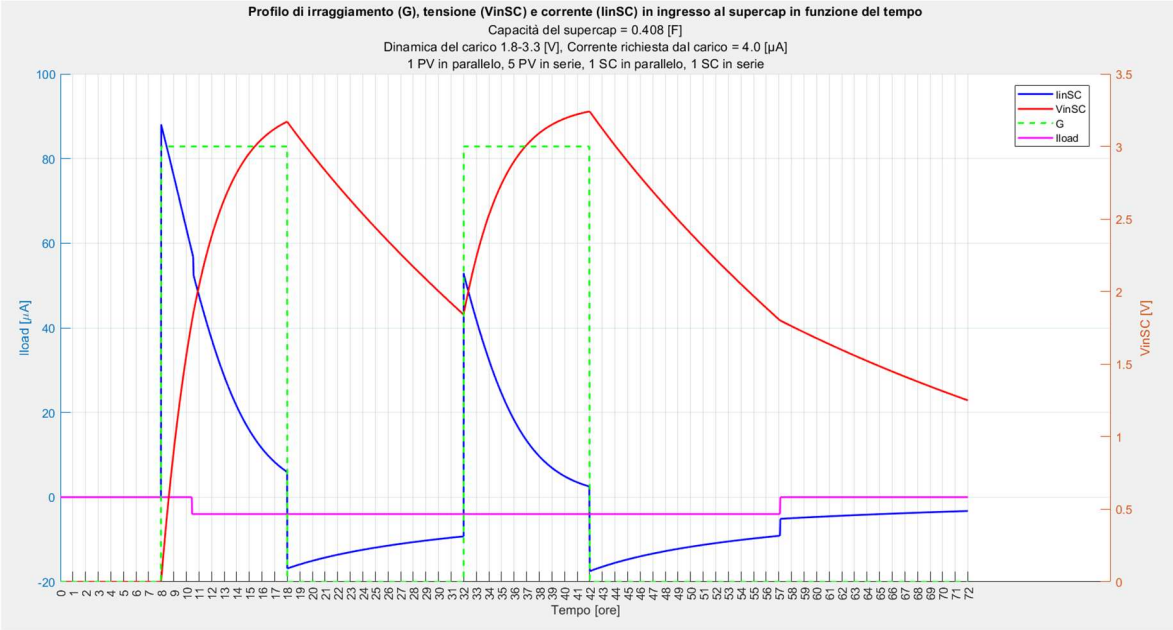

Figura 32: Risultati della simulazione per un ciclo di 72 ore. Sono mostrati l'andamento dell'irraggiamento G(t), la corrente in ingresso al supercondensatore (IinSC), la tensione ai suoi capi (VinSC) e le soglie operative del carico.

7 Risultati

In questo capitolo vengono presentati e analizzati i risultati delle simulazioni effettuate, con l'obiettivo di valutare l'accuratezza e la coerenza fisica del modello sviluppato.

In particolare, si vuole mostrare come modificando i parametri di input il simulatore risponda in modo coerente e fisicamente realistico.

Per analizzare l'effetto della variazione dei parametri di input, è stata definita una configurazione di base, utilizzata come riferimento per i successivi confronti.
In tale configurazione di riferimento sono stati impostati i seguenti parametri di input:

PAMETRI STANDARD DI SIMULAZIONE:

- Tempo di simulazione: T_sim = 72 ore = 259200 s
- Profilo di irraggiamento: (profilo orario d'ufficio) G = 3 W/m^2 dalle 8.00 alle 18.00, G = 0 W/m^2 altrimenti.
- Numero di celle in serie: NsPV = 5
- Numero di celle in parallelo: NpPV = 1
- Numero di supercondensatori in serie: NsSC = 1
- Numero di supercondensatori in parallelo: NpSC = 1
- Dinamica richiesta dal carico: VsensMin = 1.8 V - VsensMax = 3.3 V
- Corrente media richiesta dal carico: Isens = 4µA
- Capacità del supercondensatore: C_need = 0.408 F
- Massima corrente di leakage: I_leak_max = 6.6uA

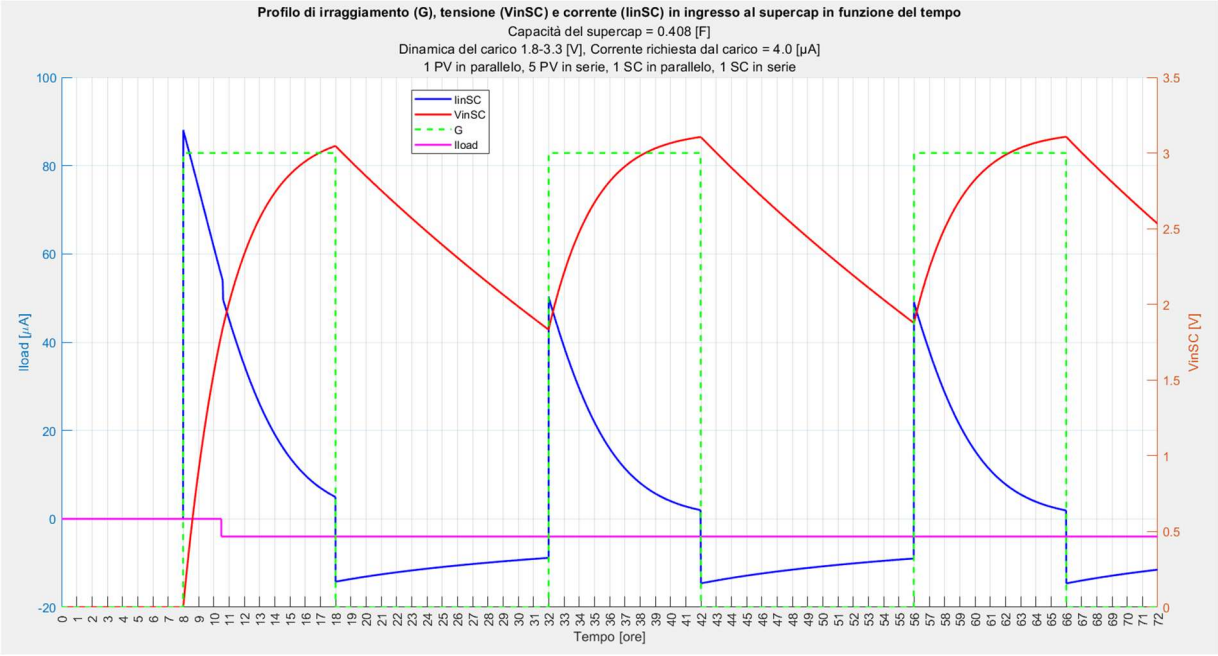

Figura 33: Caso di simulazione con i parametri standard scelti

Come si può vedere dall'immagine sovrastante non appena la cella fotovoltaica viene illuminata si ottiene un elevato apporto di corrente nel supercapacitore che diminuisce regolarmente all'aumentare di tensione nello stesso, in linea con quanto visto essere l'andamento della caratteristica della cella solare. Non appena entrati nella dinamica del carico questo viene alimentato, sottraendo così alla corrente in ingresso al supercapacitore la corrente richiesta dal carico, oltre alle già presenti perdite dovute alla corrente di leakage. Nelle fasi di buio sono visibili gli effetti dovuti alla dark current che si aggiungono alle correnti di scarica del supercapacitore appena nominate. Per tutto il tempo senza illuminazione il supercapacitore dimensionato come definito sopra riesce a mantenere alimentato il carico, in seguito quando la cella verrà lluminata nuovamente il secondo giorno di simulazione, questo riprenderà a caricarsi più lentamente rispetto al primo giorno di simulazione dato che la fase di carica non inizia più con una tensione pari a 0 V, la carica che riuscirà ad accumulare come vediamo sarà comunque sufficiente a mantenere alimentato il carico anche nei giorni successivi.

7.1 Parametri variabili

I parametri che saranno modificati nelle successive simulazioni per mettere in evidenza il confronto con la configurazione di base sono i seguenti:

I. Tempo di simulazione:

T_sim = 72 ore = 259200 s
T_sim = 168 ore = 604800 s con profilo di irraggiamento costante
T_sim = 168 ore = 604800 s con profilo di irraggiamento non costante

II. Profilo di irraggiamento:

G = 3 $\frac{W}{m^2}$dalle 8.00 alle 18.00, (profilo orario d'ufficio) G = 0 $\frac{W}{m^2}$ altrimenti.	vs	G = 3 $\frac{W}{m^2}$ dalle 20.00 alle 0.00, (profilo indoor casalingo) G = 0 $\frac{W}{m^2}$ altrimenti.

III. Numero di celle in serie:

NsPV = 5 vs NsPV = 3
NsPV = 5 vs NsPV = 7

IV. Numero di celle in parallelo:

NpPV = 1 vs NpPV = 2

V. Numero di supercondensatori in serie:

NsSC = 1 vs NsSC = 2

VI. Numero di supercondensatori in parallelo:

NpSC = 1 vs NpSC = 2

VII. Dinamica di funzionamento del carico:

VsensMin = 1.8 V - VsensMax = 3.3 V
vs
VsensMin = 1.0 V - VsensMax = 4.0 V

VIII. Corrente media richiesta dal carico:

Isens = 4μA vs Isens = 10μA

IX. Capacità del supercapacitore:

C_need = 0.408 F vs C_need = 0.163 F
C_need = 0.408 F vs C_need = 0.816 F

X. Impatto della corrente di leakage:

I_leak_max = 6.6uA vs I_leak_max = 33uA
I_leak_max = 6.6uA vs I_leak_max = 3.3uA

7.2 Variazione dei parametri di simulazione

7.2.1 Confronto tempo di simulazione

Casi a confronto:
Caso standard: T_sim = 72 ore = 259200 s
Caso alterato: T_sim = 168 ore = 604800 s con profilo di G costante
Caso alterato: T_sim = 168 ore = 604800 s con profilo di G non costante

Caso standard: T_sim = 72 ore = 259200 s

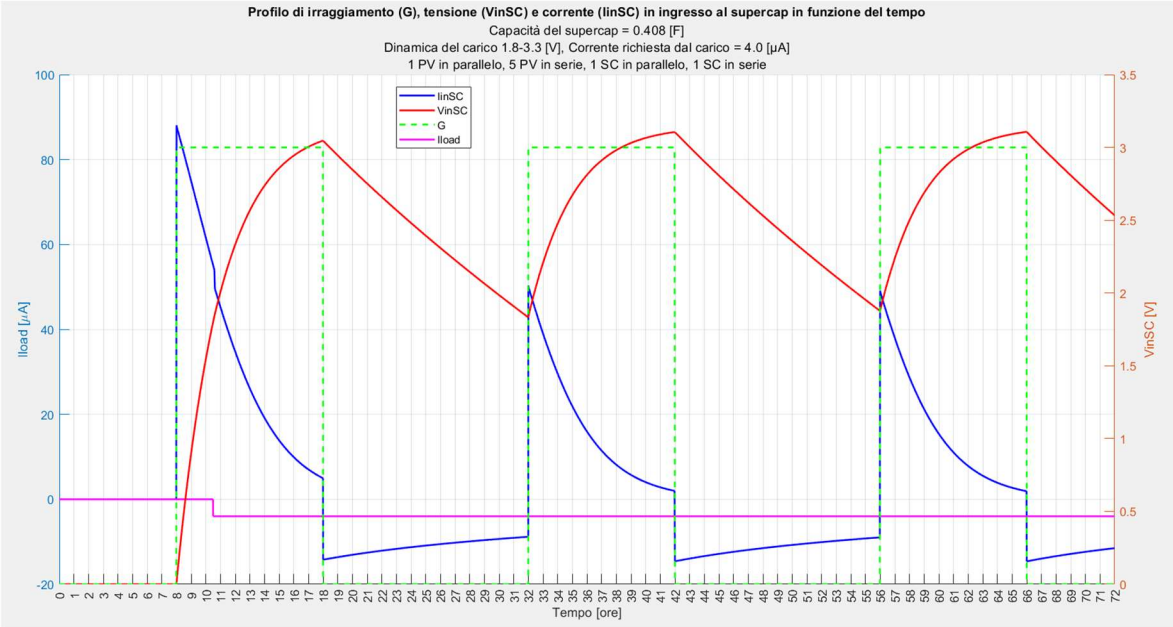

Figura 84: Simulazione standard di 72 ore (tre giorni)

Caso: T_sim = 168 ore = 604800 s con profilo di G costante

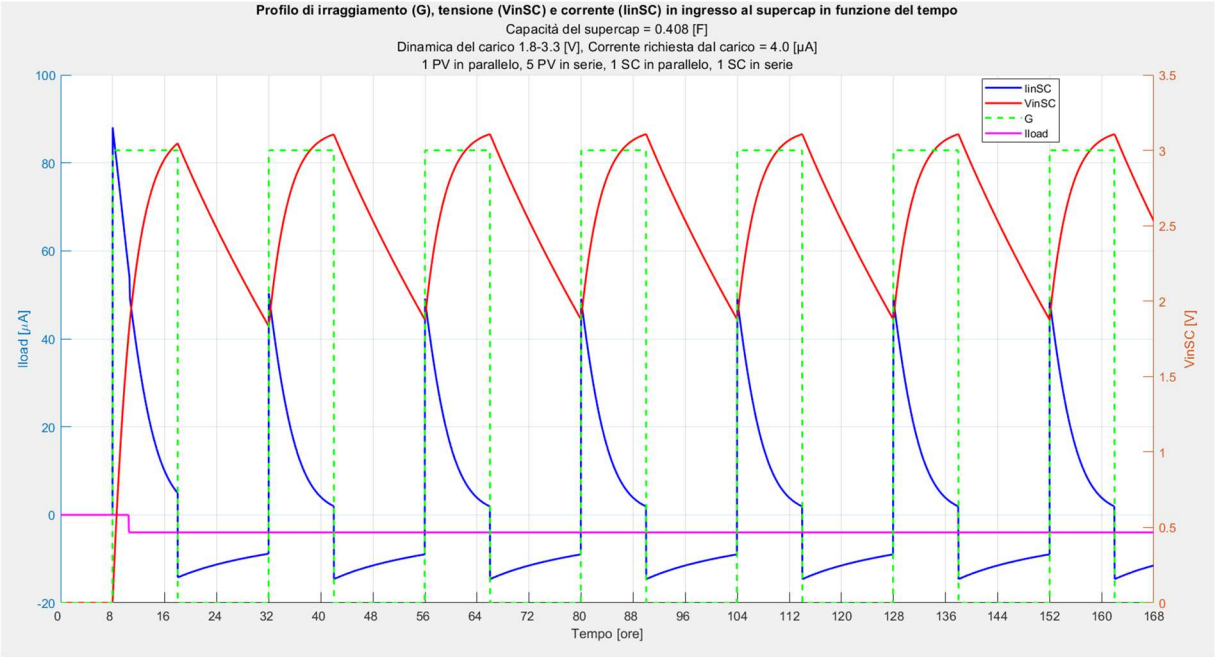

Figura 95: Simulazione di 168 ore (una settimana) con profilo di irraggiamento costante

Caso: T_sim = 168 ore = 604800 s con profilo di G non costante

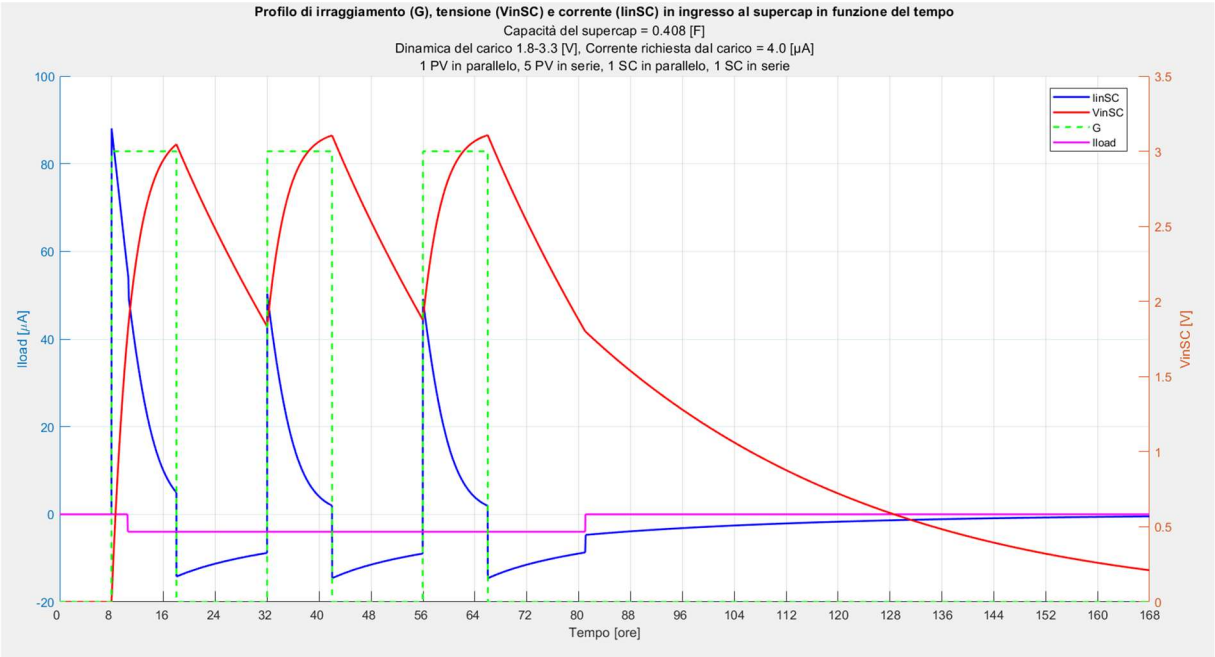

Figura 106: Simulazione di 168 ore (una settimana) con profilo di irraggiamento non costante

7.2.2 Confronto profilo di irraggiamento

Casi a confronto:

Caso standard: Profilo d'ufficio, G = 3 $\frac{W}{m^2}$ dalle 8.00 alle 18.00,

G = 0 $\frac{W}{m^2}$ altrimenti.

Caso tratteggiato: Profilo casalingo, G = 3 $\frac{W}{m^2}$ dalle 20.00 alle 0.00,

G = 0 $\frac{W}{m^2}$ altrimenti.

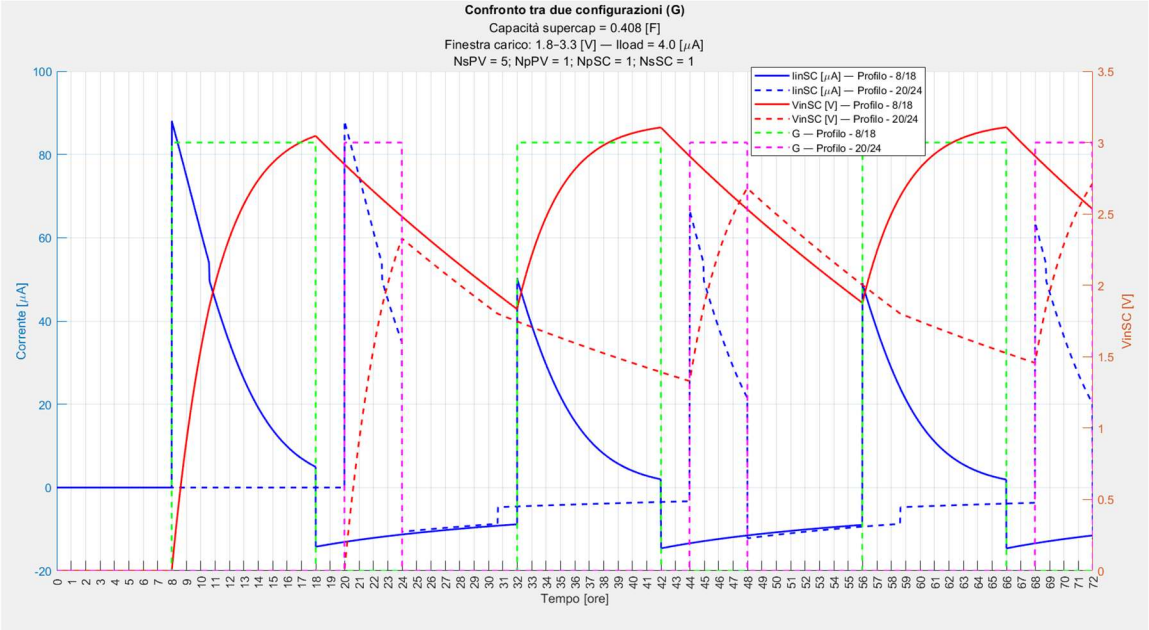

Figura 117: Simulazioni con diversi profili di irraggiamento a confronto. (ufficio vs casa)

Dalla precedente immagine si può notare come cambiando il profilo di irraggiamento il simulatore restituisca due grafici in linea con i profili di irraggiamento impostati che mostrano un diverso impiego del dispositivo, permettendo così di valutarne la funzionalità e l'efficienza in diverse situazioni.

7.2.3 Confronto celle in serie

Casi a confronto:
Caso standard: NsPV = 5
Caso tratteggiato: NsPV = 3

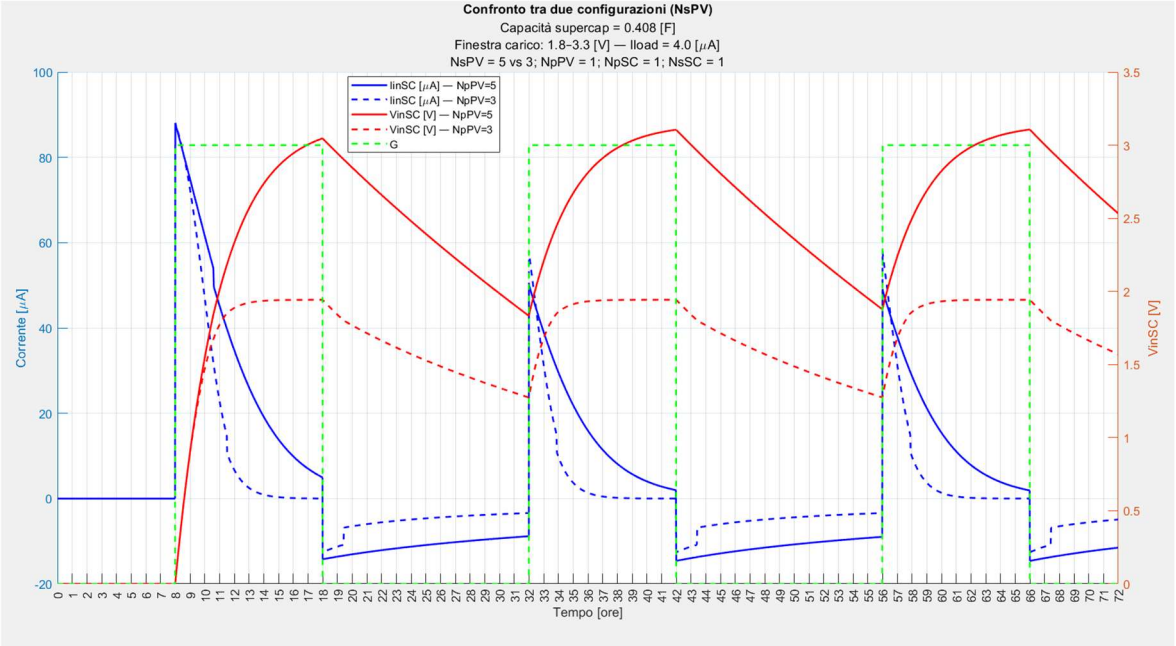

Figura 128: Simulazione con diverso numero di celle in serie. (5 vs 3)

Dall'immagine precedente è possibile notare come la variazione del numero di celle in serie infici sulla tensione massima raggiungibile dal supercapacitore, minore nel caso tratteggiato in quanto è inferiore il numero di celle in serie. Difatti il supercapacitore ha una tensione pari alla somma della tensione delle celle, che singolarmente hanno come valore di tensione massimo Voc, valore che si aggira intorno ai 0.65 V.

Casi a confronto:
Caso standard: NsPV = 5
Caso tratteggiato: NsPV = 7

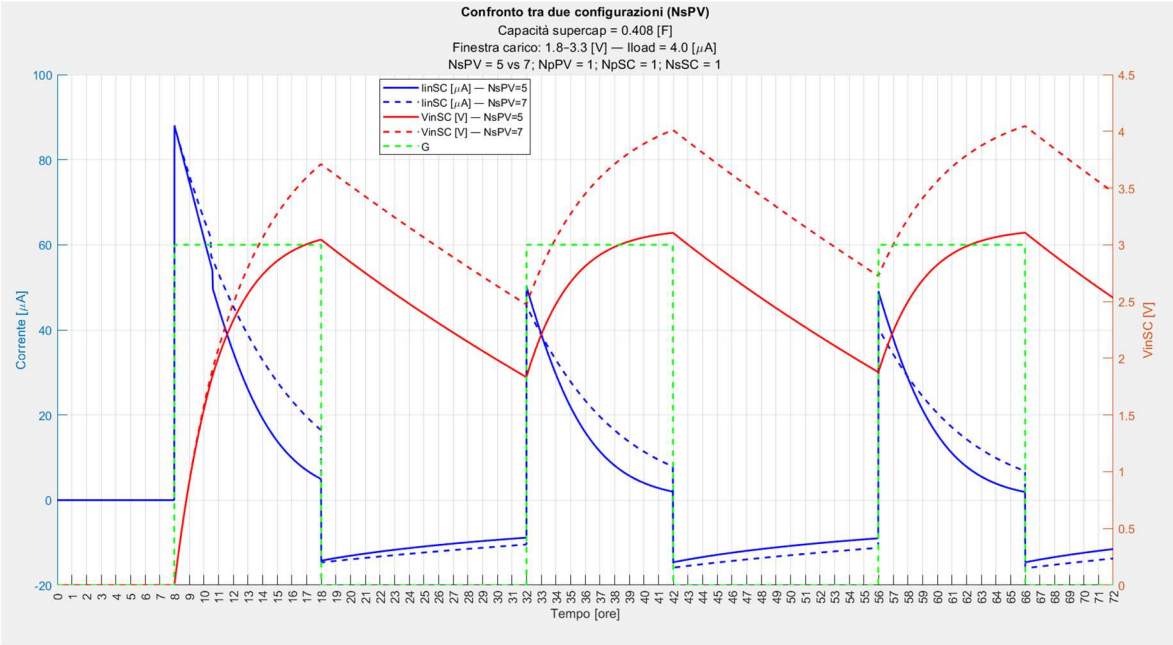

Figura 139: Simulazione con diverso numero di celle in serie. (5 vs 7)

Dall'immagine precedente è possibile notare come la variazione del numero di celle in serie infici sulla tensione massima raggiungibile dal supercapacitore, maggiore nel caso tratteggiato in quanto è superiore il numero di celle in serie.

7.2.4 Confronto celle in parallelo

Casi a confronto:
Caso standard: NpPV = 1
Caso tratteggiato: NpPV = 2

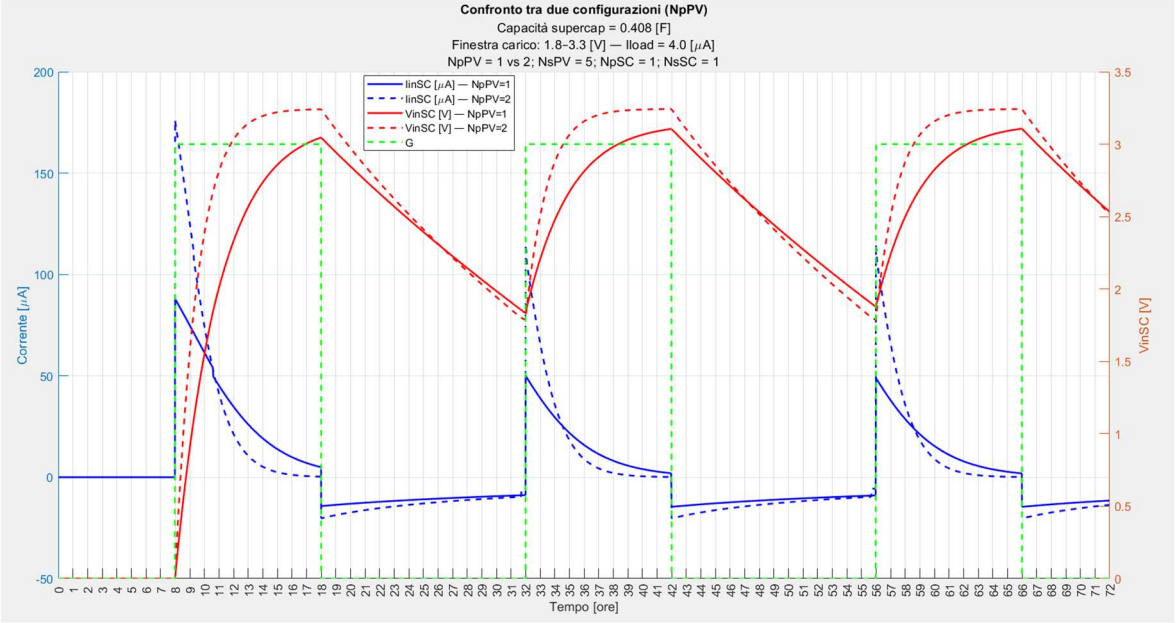

Figura 40: Simulazione con diverso numero di celle in parallelo. (1 vs 2)

Dall'immagine precedente è possibile notare come la variazione del numero di celle in parallelo infici sulla corrente in ingresso al supercapacitore, questa è maggiore nel caso tratteggiato in quanto è superiore il numero di celle in parallelo. Di conseguenza la carica del supercondensatore avviene più velocemente nel caso tratteggiato ed anche la scarica sarà più veloce in quanto sarà amplificato anche il fenomeno della dark current.

7.2.5 Confronto supercondensatori in serie

Casi a confronto:
Caso standard: NsSC = 1
Caso tratteggiato: NsSC = 2

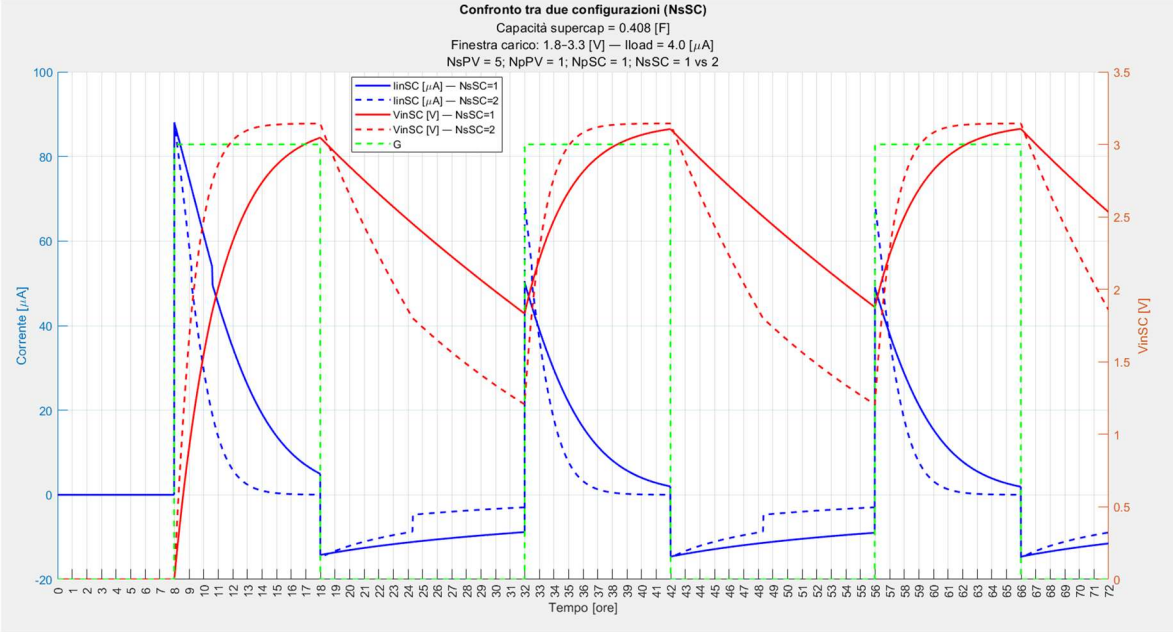

Figura 41: Simulazione con diverso numero di supercondensatori in serie. (1 vs 2)

Dall'immagine precedente è possibile notare come la variazione del numero di condensatori in serie infici sui tempi di carica e scarica del supercondensatore. Difatti con l'aumento del numero di questi in serie, la tensione e la carica massima aumentano ma la capacità totale viene divisa, riducendo così i tempi di carica e di scarica.

7.2.6 Confronto supercondensatori in parallelo

Casi a confronto:
Caso standard: NpSC = 1
Caso tratteggiato: NpSC = 2

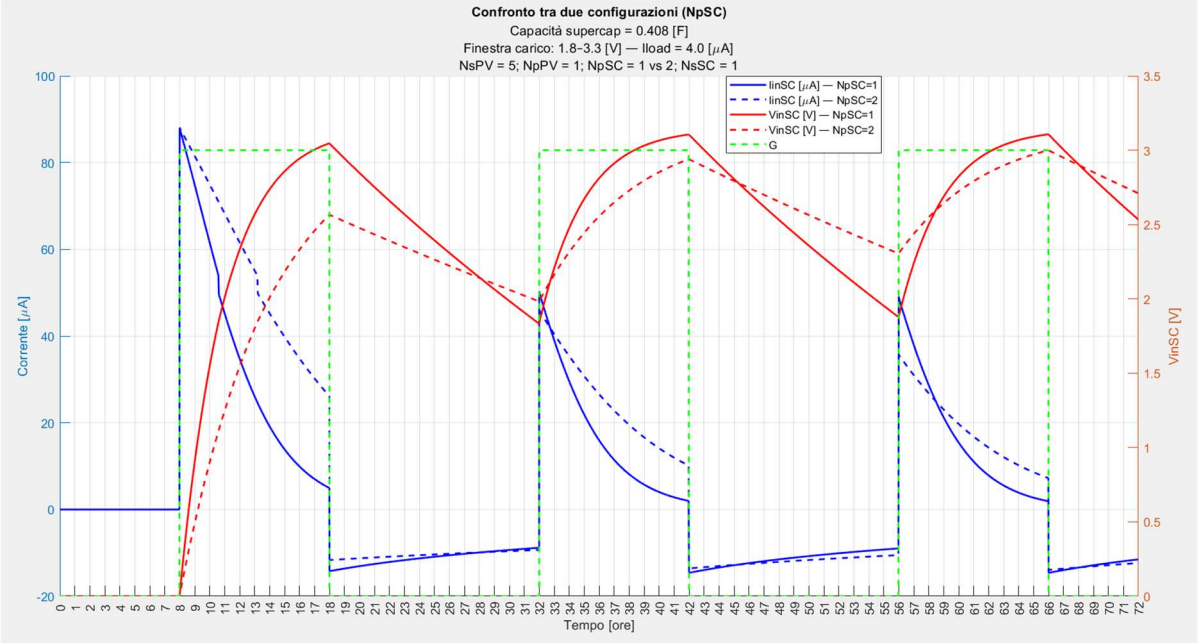

Figura 42: Simulazione con diverso numero di supercondensatori in parallelo. (1 vs 2)

Dall'immagine precedente è possibile notare come la variazione del numero di condensatori in parallelo infici sui tempi di carica e scarica del supercondensatore. Difatti con l'aumento del numero di questi in parallelo, la tensione e la carica massima rimangono invariate ma la capacità totale viene moltiplicata, aumentando così sia i tempi di carica che di scarica.

7.2.7 Confronto dinamica di funzionamento del carico

Casi a confronto:
Caso standard: VsensMin = 1.8 V - VsensMax = 3.3 V
Caso tratteggiato: VsensMin = 1.0 V - VsensMax = 4.0 V

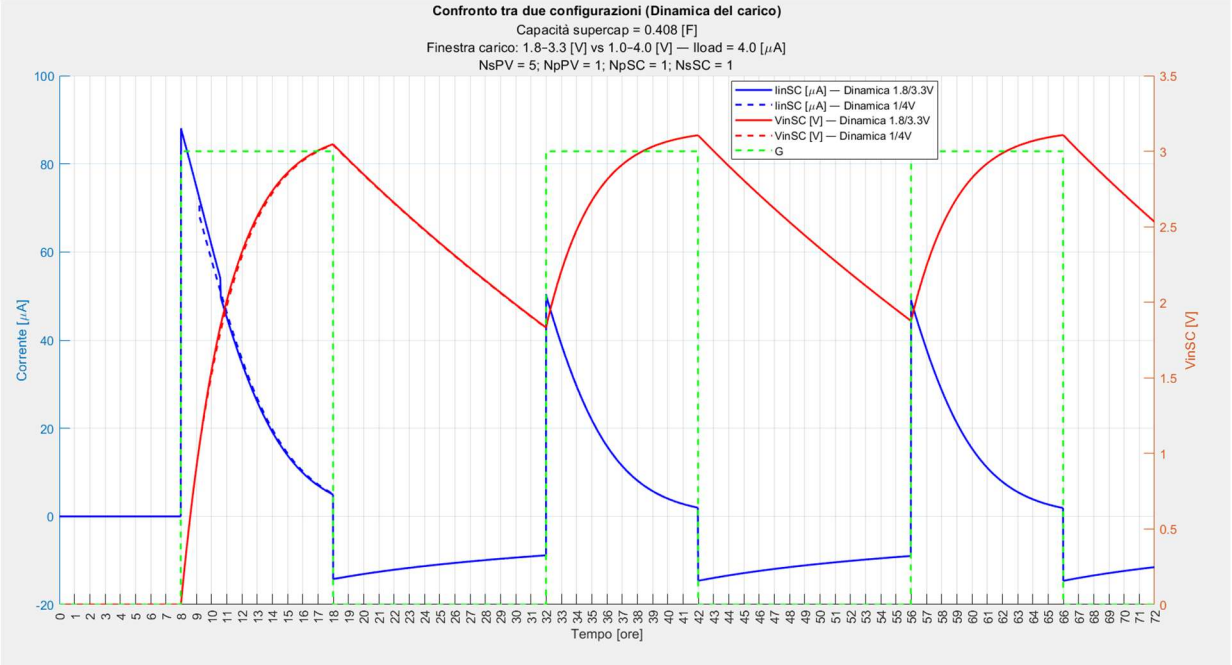

Figura 143: Simulazione con diversa dinamica del carico. (1.8 - 3.3V vs 1.0 - 4.0V)

Dall'immagine precedente è possibile notare come la variazione della dinamica di funzionamento del carico infici sulla richiesta di corrente al supercapacitore. Difatti la corrente nel caso tratteggiato verrà fornita al carico prima rispetto al caso standard.

7.2.8 Confronto corrente richiesta dal carico

Casi a confronto:
Caso standard: Isens = $4\mu A$
Caso tratteggiato: Isens = $10\mu A$

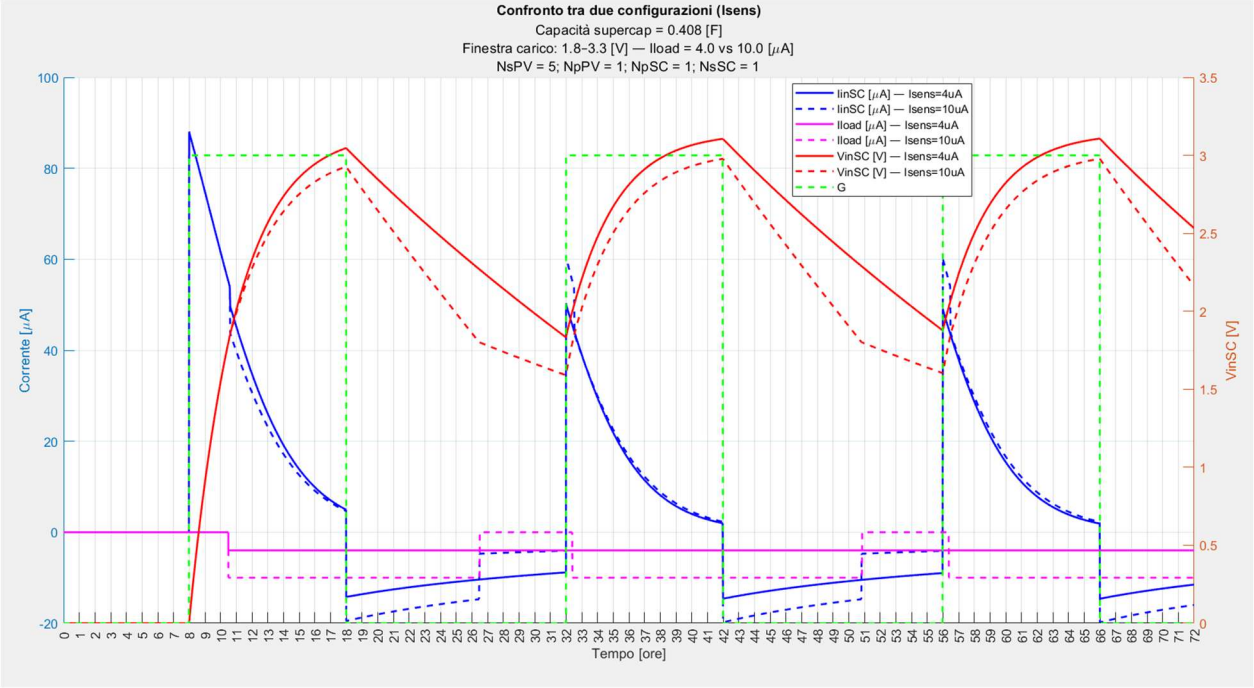

Figura 154: Simulazione con diversa corrente richiesta dal carico. (4uA vs 10uA)

Dall'immagine precedente è possibile notare come la variazione della corrente richiesta dal carico infici sulla quantità di corrente richiesta al supercapacitore. Nel caso tratteggiato questa è troppo alta affinché il supercondensatore abbia carica sufficiente per garantirne ininterrottamente l'alimentazione fino al successivo rifornimento di corrente tramite la cella solare.

7.2.9 Confronto capacità del supercondensatore

Casi a confronto:
Caso standard: C_need = 0.408 F
Caso tratteggiato: C_need = 0.163 F

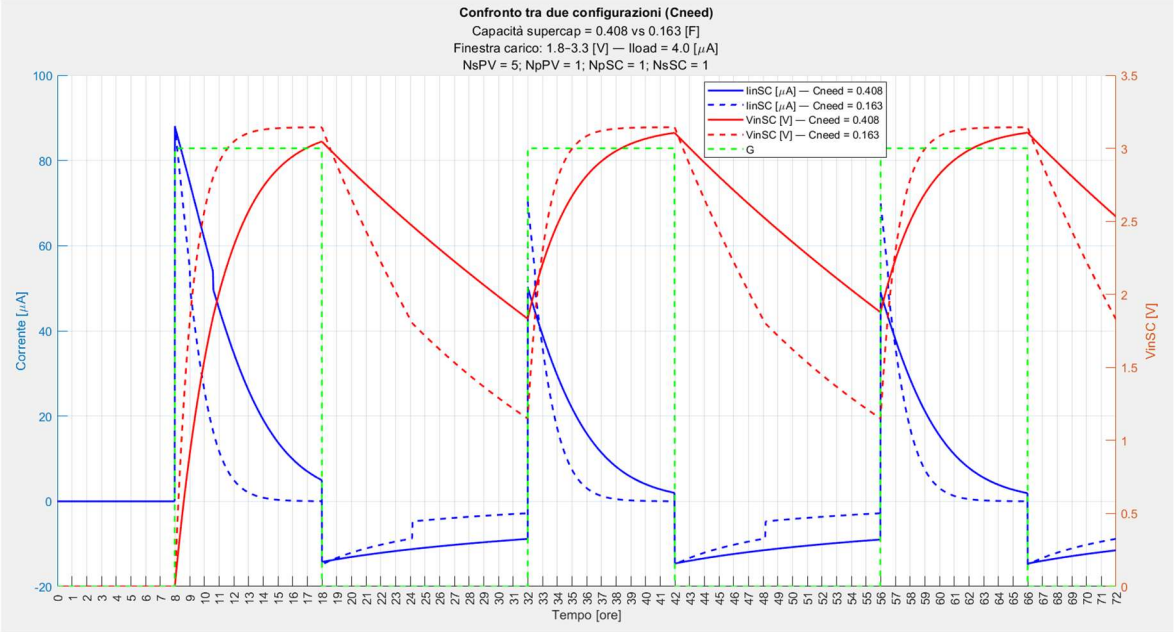

Figura 165: Simulazione con diversa capacità del supercondensatore. (0.408 F vs 0.163 F)

Dall'immagine precedente è possibile notare come la variazione della capacità del carico infici sul tempo necessario a caricarlo e scaricarlo. Difatti il caso tratteggiato con una capacità inferiore si carica e scarica più velocemente rispetto al caso standard.

Casi a confronto:
Caso standard: C_need = 0.408 F
Caso tratteggiato: C_need = 0.816 F

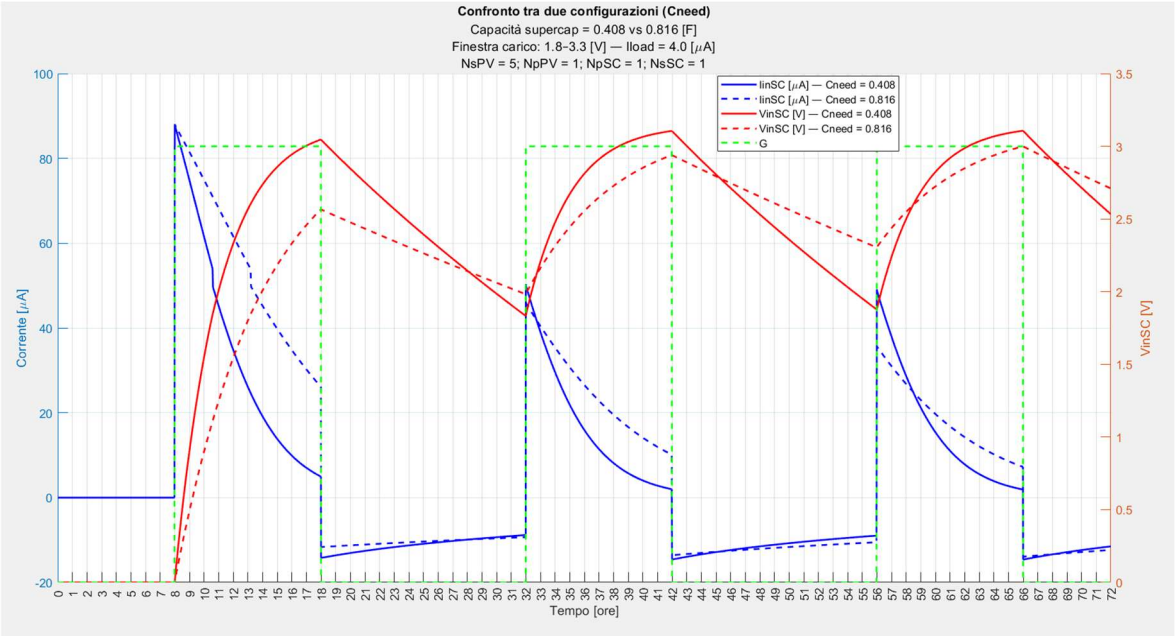

Figura 17: Simulazione con diversa capacità del supercondensatore. (0.408 F vs 0.816 F)

Dall'immagine precedente è possibile notare come la variazione della capacità del carico infici sul tempo necessario a caricarlo e scaricarlo. Difatti il caso tratteggiato con una capacità superiore si carica e scarica più lentamente rispetto al caso standard.

7.2.10 Confronto corrente di leakage

Casi a confronto:
Caso standard: I_leak_max = 6.6uA
Caso tratteggiato: I_leak_max = 33uA

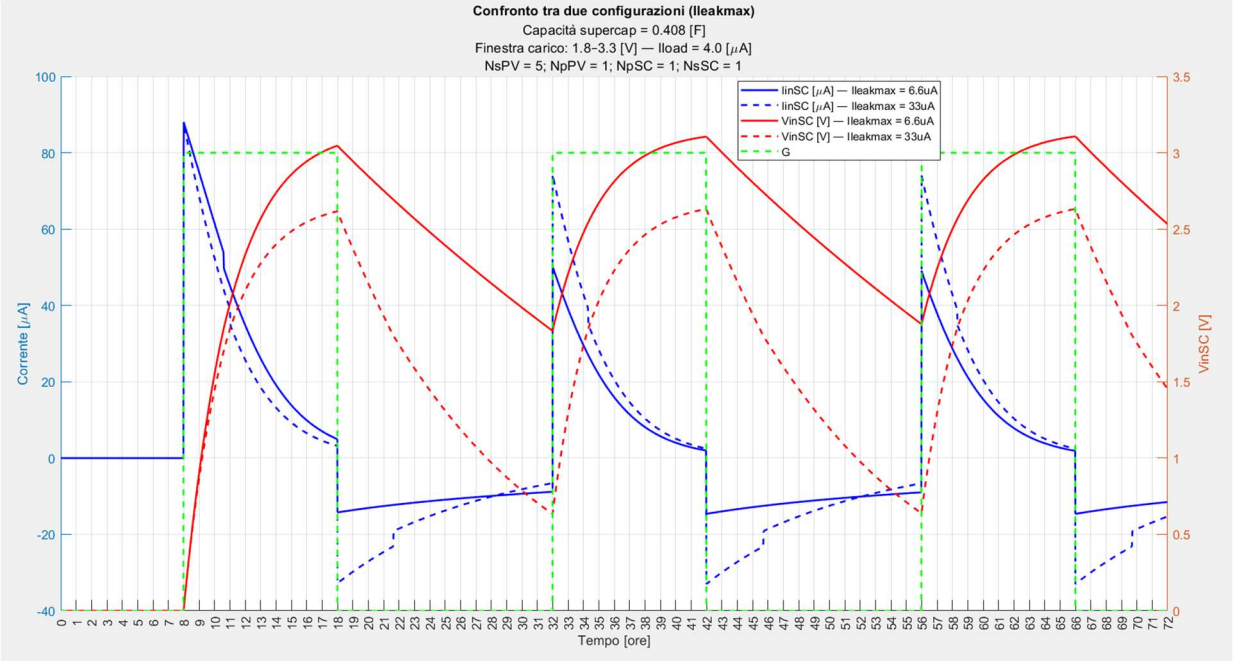

Figura 18: Simulazione con variazione della massima corrente di leakage. (6.6uA vs 33uA)

Dall'immagine precedente è possibile notare come la variazione della corrente di leakage in funzione della tensione infici sulla carica e scarica del supercondensatore. Nel caso tratteggiato con una corrente di leakage più alta il dispositivo si carica più lentamente e si scarica molto più velocemente.

Casi a confronto:
Caso standard: I_leak_max = 6.6uA
Caso tratteggiato: I_leak_max = 3.3uA

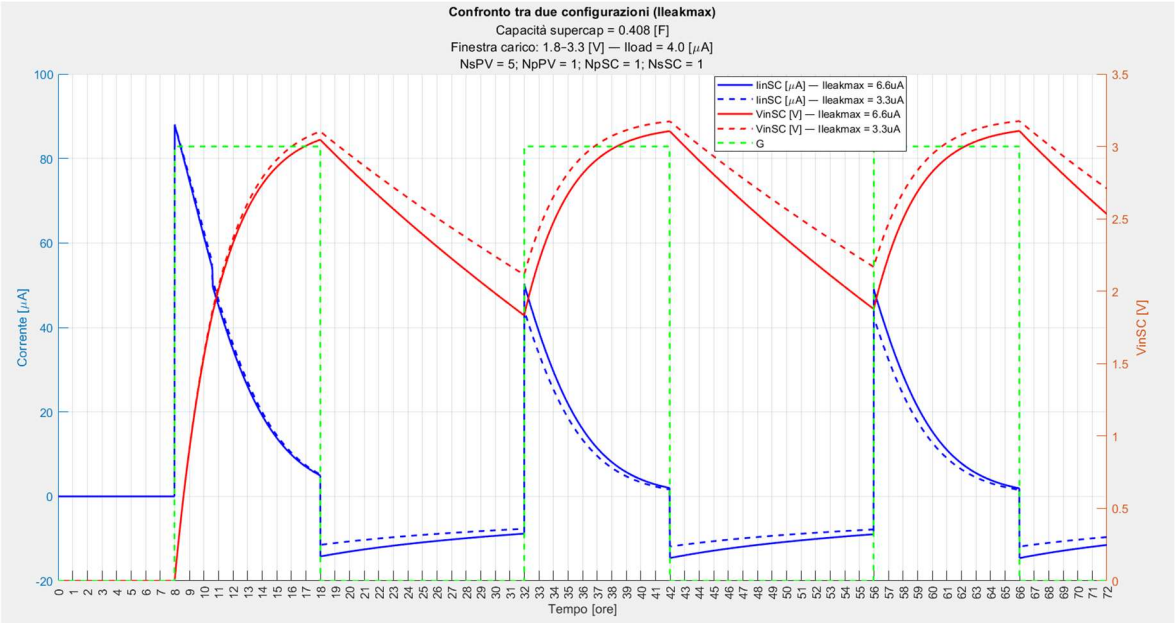

Figura 198: Simulazione con variazione della massima corrente di leakage. (6.6uA vs 3.3uA)

Dall'immagine precedente è possibile notare come la variazione della corrente di leakage in funzione della tensione infici sulla carica e scarica del supercondensatore. Nel caso tratteggiato con una corrente di leakage più bassa il dispositivo si carica più velocemente e si scarica più lentamente.

8 Conclusione

Il lavoro svolto in questa tesi ha avuto come obiettivo principale lo sviluppo e la validazione di un modello numerico completo capace di descrivere il comportamento di un sistema di alimentazione autonomo basato sull'integrazione tra una cella fotovoltaica e un supercondensatore, destinato ad applicazioni di sensoristica a basso consumo in ambiente indoor.
L'intero percorso ha permesso di passare da una fase di modellazione fisica dei singoli componenti a una rappresentazione dinamica dell'intero sistema, validata attraverso simulazioni realistiche e confronti con dati sperimentali.

8.1 Perché questo progetto di tesi?

La crescente diffusione di dispositivi connessi – dai sensori IoT ai sistemi di monitoraggio ambientale – richiede soluzioni di alimentazione autonome, affidabili e sostenibili, che riducano la necessità di sostituzione delle batterie e gli impatti ambientali correlati.
In questo contesto, questo lavoro tesi fornisce un contributo concreto allo sviluppo di strumenti di modellazione e analisi per sistemi energetici di piccola scala, dimostrando come l'integrazione tra celle fotovoltaiche e supercondensatori possa rappresentare un'alternativa efficace alle soluzioni tradizionali a batteria.
Il modello realizzato costituisce un riferimento utile sia per la fase di progettazione che per l'ottimizzazione di sistemi autonomi di alimentazione indoor, grazie alla sua capacità di prevedere con buona accuratezza il comportamento del sistema al variare delle condizioni di illuminazione e dei parametri costruttivi.

8.2 Cosa si è fatto?

Con questo lavoro si è voluto dimostrare che, tramite una modellazione accurata ma computazionalmente sostenibile, è possibile rappresentare in modo realistico l'interazione tra la generazione fotovoltaica e lo stoccaggio energetico in un supercondensatore, valutando la fattibilità di un'alimentazione autonoma per sensori di piccola potenza.
Questo lavoro ha inoltre mostrato come, partendo da dati sperimentali limitati, sia possibile estrarre parametri fisicamente coerenti attraverso un processo di fitting automatico, e successivamente utilizzare tali parametri per alimentare simulazioni dinamiche del sistema reale.
Il valore aggiunto di questo approccio sta nella creazione di un modello modulare e scalabile, che può essere facilmente adattato a diverse configurazioni di celle fotovoltaiche, capacità del supercondensatore o profili di carico, diventando uno strumento di previsione e ottimizzazione.

8.3 Fasi di sviluppo

Il percorso di lavoro si è articolato in più fasi coerenti e integrate tra loro:

1. Analisi teorica e selezione dei modelli:
   Sono stati studiati diversi modelli di letteratura per la rappresentazione delle celle fotovoltaiche (Rahmatian, Tayeb, Warepam) e dei supercondensatori (Cabrane),

valutandone limiti e potenzialità. È stato infine scelto il modello a diodo singolo per la cella e quello a due rami (Zubieta–Bonert) per il supercondensatore, in quanto rappresentano un buon compromesso tra accuratezza e semplicità.

2. Sviluppo dell'algoritmo di fitting:
   In ambiente MATLAB è stato implementato un algoritmo iterativo per la stima automatica dei parametri della cella fotovoltaica, basato su un metodo ai minimi quadrati non lineari. L'algoritmo consente di determinare i parametri n, Rs e Rsh a partire dai dati sperimentali, garantendo la coerenza fisica con i valori misurati di Isc e Voc.

3. Validazione del modello fotovoltaico:
   I risultati del fitting sono stati confrontati con le curve I–V sperimentali, mostrando una buona aderenza in tutta la regione operativa, con errori ridotti nella zona centrale della curva e discrepanze limitate solo vicino alla tensione di circuito aperto. Ciò ha confermato la validità del modello per le condizioni di illuminazione indoor considerate.

4. Implementazione del modello dinamico:
   A partire dai parametri ottenuti, è stato sviluppato un modello in Simulink che integra la cella fotovoltaica, il supercondensatore e il carico rappresentato da un sensore. La struttura modulare del modello consente di simulare il comportamento del sistema nel tempo, considerando variazioni di irraggiamento e di stato di carica.

5. Calibrazione del supercondensatore e definizione dei parametri di simulazione:
   È stata introdotta una procedura di calibrazione automatica che, partendo dai requisiti del carico (tensione minima, corrente richiesta e autonomia desiderata nel caso ideale senza perdite), dimensiona la capacità e le resistenze equivalenti del supercondensatore. Questo ha permesso di riprodurre realisticamente le dinamiche di carica e scarica e di verificare la coerenza dei risultati.

6. Simulazione e analisi dei risultati:
   Le simulazioni condotte in ambiente MATLAB/Simulink hanno permesso di osservare l'evoluzione nel tempo della tensione ai capi del supercondensatore e della corrente in ingresso, in funzione del profilo di irraggiamento definito. Le variabili di progetto (numero di celle PV, capacità del supercondensatore, corrente di carico, profilo luminoso) sono state modificate sistematicamente per valutare la risposta del sistema in diversi scenari operativi.

8.4 Risultati

Le simulazioni hanno evidenziato che il sistema PV–SC è in grado di garantire un'alimentazione stabile per sensori a basso consumo in condizioni di illuminazione indoor realistica.
In particolare:

- La tensione del supercondensatore può essere definita per non superare l'intervallo operativo del sensore, dimostrando la capacità del sistema di mantenere la continuità di alimentazione senza rischi per il carico.
- Il comportamento dinamico simulato riproduce fedelmente le fasi di carica e scarica, confermando la correttezza delle costanti di tempo calcolate e della calibrazione effettuata.
- L'influenza dei parametri costruttivi (numero di celle fotovoltaiche, capacità, corrente di leakage, profilo di G) è stata analizzata in dettaglio nel Capitolo 7, mostrando come ogni fattore contribuisca in modo prevedibile alle prestazioni complessive del sistema.

Nel complesso, i risultati confermano la bontà dell'approccio proposto e la sua applicabilità a sistemi reali. Il modello sviluppato non si limita a descrivere il comportamento elettrico dei singoli componenti, ma fornisce una piattaforma di simulazione completa e riutilizzabile per futuri sviluppi.

8.5 Considerazioni finali

Il lavoro di questa tesi ha dimostrato che, attraverso una modellazione integrata e una validazione numerico-sperimentale accurata, è possibile progettare sistemi fotovoltaici con supercondensatori capaci di alimentare dispositivi autonomi in ambienti a bassa illuminazione.
Il valore principale di questo lavoro risiede nella metodologia: il modello è stato costruito in modo modulare, verificabile e facilmente estendibile, permettendo di analizzare diversi scenari applicativi senza dover riprogettare da zero l'intero sistema.

8.5.1 Sviluppi Futuri

Le prospettive future includono:

- l'estensione verso modelli fotovoltaici a due diodi per migliorare la precisione nella regione ad alta tensione;
- l'inclusione di modelli di leakage dipendenti dalla temperatura e dalla tensione;
- l'integrazione con algoritmi di gestione dell'energia (energy management) per massimizzare l'autonomia del sistema.
-

In conclusione, il lavoro svolto fornisce una base metodologica solida per la progettazione e la simulazione di sistemi energetici ibridi destinati a sensori autonomi, ponendo le fondamenta per lo sviluppo di soluzioni sostenibili e affidabili nel campo della microelettronica e dell'Internet of Things.

I. Appendice

Caratteristica_I-V.m

```
%% Caratteristica_IV.m – Fit per curva (n, Rs, Rsh) indipendenti per ogni G - stimare i parametri del diodo per ciascun irraggiamento
% Obiettivo: data una famiglia di curve I–V sperimentali (una per ogni irraggiamento G)
% stimare per ciascuna curva, i parametri del diodo equivalente [n, Rs, Rsh]
% Parametri da stimare per ogni curva minimizzando lo scarto tra corrente simulata e corrente misurata.
% Criterio di ottimizzazione (least-squares)
% Nota: i blocchi Simulink leggono le variabili dal base workspace (assegnate via assignin).
% Il modello attinge dal base workspace.
% Il modello atteso è 'PV_SC_Model_IV' con segnali VoutPV e IoutPV disponibili in out.


clear;                          % Pulisce le variabili nel workspace corrente
%clc;                           % (Opzionale) Pulisce la Command Window

% ================== Impostazioni base ==================
Cartella_con_dati = fullfile(pwd, 'Dati Sperimentali');
% cartella dove leggere i .txt (usa path relativo alla cartella corrente)
modelName = 'PV_SC_Model_IV';
% nome del modello Simulink da simulare

% Peso per avvicinare anche il Fill Factor [adim]
lambdaFF = 0.08;      % tipicamente 0.01–0.2; 0 disattiva la penalità sul FF
% Regolarizzazione su FF nel residuo

% Bounds e guess "ampi" (usati per curve in luce)
% x = [n, Rs[Ω], Rsh[Ω]]
lb         = [1.0,  0,     1e3];     % Limiti inferiori per n, Rs, Rsh nelle curve illuminate
ub         = [3.0,  50,    1e10];    % Limiti superiori per n, Rs, Rsh nelle curve illuminate
x0_default = [1.3,  0.5,   1e6];     % Guess di partenza di default per le curve illuminate

% Bounds più "conduttivi" SOLO per G=0 (dark IV), per evitare Rsh→∞ e Rs→0
lb_dark = [1.2,  5e-3,  1e4];        % Limiti inferiori per il caso al buio
ub_dark = [3.0,    1,   1e7];        % Limiti superiori per il caso al buio
x0      = [1.8,  0.2,   1e6];        % Guess iniziale (non usata direttamente più sotto, tenuta per riferimento)

% Guess specifici (una riga per dataset; se i dataset sono di più, si riusa l'ultima riga)
x0_list = [                          % Matrice di guess per curva (k-esima riga -> k-esimo dataset)
    1.10   0.20   1e5                % Guess per dataset 1
    1.50   1.00   1e7                % Guess per dataset 2
    2.00   5.00   1e8                % Guess per dataset 3
    1.80   0.10   5e6                % Guess per dataset 4
                                      (riusata per i successivi se >4 file)
```

```
];

% Opzioni ottimizzatore lsqnonlin:
% - Display 'iter-detailed' => utile in debug; usa 'final' per output più
pulito
% - Tolleranze strette ma non estreme per evitare iterazioni inutili
% - Forward diff stabile; pchip in interp1 later evita oscillazioni
opts = optimoptions('lsqnonlin', ...           % Crea struttura opzioni per
lsqnonlin
    'Display','final', ...                     % Mostra solo il riepilogo
finale
    'MaxFunctionEvaluations', 1e4, ...         % Numero massimo di valutazioni
della funzione
    'MaxIterations',            200, ...       % Numero massimo di iterazioni
    'FunctionTolerance',       1e-9, ...       % Tolleranza sul valore della
funzione obiettivo
    'StepTolerance',           1e-8, ...       % Tolleranza sul passo
dell'algoritmo
    'FiniteDifferenceType','forward');         % Differenze finite in avanti
per il gradiente numerico

% ================== Import dati & parametri iniziali ==================
% Lettura file .txt, costruzione struttura dati e tabella riassuntiva
[Dati_sperimentali_da_file, Tabella_riassuntiva] =
importaDatiSperimentali(Cartella_con_dati);  % Legge e parsifica tutti i .txt

% --- PREPARA LUT per il modello (lette da una 1-D Lookup Table nel modello)
% La LUT mappa G -> (Isc, Voc) per consolidare coerenza tra fit e Simulink
G_all   = [Dati_sperimentali_da_file.G_value]';   % Vettore di tutti i valori G
dai file
Isc_all = [Dati_sperimentali_da_file.Isc_A]';     % Vettore Isc associati ai
dataset
Voc_all = [Dati_sperimentali_da_file.Voc_V]';     % Vettore Voc associati ai
dataset

% Rimuovi duplicati su G e garantisci ordine (qui 'stable' preserva l'ordine di
apparizione)
[ G_bp, idx ] = unique(G_all, 'stable');          % Breakpoints G univoci per
la LUT e indici corrispondenti
Isc_tab = Isc_all(idx);                           % Tabella Isc allineata ai
breakpoints univoci
Voc_tab = Voc_all(idx);                           % Tabella Voc allineata ai
breakpoints univoci

% Esponi LUT nel base workspace (il modello le legge da qui)
assignin('base','G_bp',    G_bp);                 % Mette G_bp nel base
workspace
assignin('base','Isc_tab', Isc_tab);              % Mette Isc_tab nel base
workspace
assignin('base','Voc_tab', Voc_tab);              % Mette Voc_tab nel base
workspace

disp('✓ Dati sperimentali letti e caricati correttamente.');  % Messaggio di
stato

% Inizializza parametri di default in base (usati da blocchi/maschere se
necessario)
```

```
setInitialParameters();                          % Popola variabili base
(Tref, q, ecc.) per il modello

% Numero di dataset (curve) disponibili
nSets = numel(Dati_sperimentali_da_file);
% Conta quante curve sono state trovate
if nSets == 0
% Se nessun dataset è presente
    error('Nessun dataset trovato in "%s".', Cartella_con_dati);  % Interrompe
con errore
end

% ================== Fit per curva ==================
% Prealloc struttura risultati per efficienza e ordine
fitTemplate = struct( ...
% Template con tutti i campi risultati
    'G_label',"", 'G_value',NaN, ...
    'n',NaN, 'Rs',NaN, 'Rsh',NaN, ...
    'Voc',NaN, 'Isc',NaN, ...
    'FF_fit',NaN, 'FF_exp',NaN, ...
    'exitflag',NaN, 'rmse',NaN, ...
    'Vexp',[], 'Iexp',[], 'Vsim',[], 'Isim',[]);
Fitted_Parameters = repmat(fitTemplate, nSets, 1);
% Prealloca array di struct dei risultati

% Costante grafica e filtro stabilità (riusate più volte)
VMIN_PLOT = 20e-3;   % Evita V~0 in grafico (per chiarezza visiva)
VMIN_FIT  = 20e-3;   % Evita V~0 nel fit (stabilizza interpolazioni/pesi)

for k = 1:nSets                                  % Cicla su ciascun
dataset/curva
    ds = Dati_sperimentali_da_file(k);           % Estrae la k-esima curva

    % --- Dati sperimentali (usa solo V>=0 e valori finiti)
    Vexp = ds.Voltage_V(:);                      % Vettore tensione
sperimentale
    Iexp = ds.Current_A(:);                      % Vettore corrente
sperimentale
    mexp = isfinite(Vexp) & isfinite(Iexp) & Vexp >= 0;  % Maschera validi:
numeri finiti e V>=0
    Vexp = Vexp(mexp);                           % Applica maschera a V
    Iexp = Iexp(mexp);                           % Applica maschera a I

    if numel(Vexp) < 5                           % Controllo minimo numero
di punti utili
        warning('Dataset %d: punti utili insufficienti (<5). Salto curva.', k);
% Avvisa e salta
        continue;                                % Passa al dataset
successivo
    end

    % --- Seleziona guess per questa curva: riga k di x0_list, altrimenti
ultima riga
    x0 = x0_list(min(k, size(x0_list,1)), :);    % Prende la riga
corrispondente o l'ultima se k eccede
    if any(~isfinite(x0)), x0 = x0_default; end  % Se la riga contiene non-
finiti, fallback al default
```

```
    % --- Distinzione G=0 (dark) vs luce: pesi e bounds diversi
    isDark = (ds.G_value <= 0);                     % Flag: curva al buio se
G<=0
    if isDark
        lambdaFF_local = 0;                         % Nessuna penalità FF in
dark
        Isc_eff        = max(1e-12, 1e-6*max(Isc_tab));  % Piccolo Isc
effettivo per scalare FF/parametri
        %Isc_eff        = max(abs(ds.Isc_A), 1e-9); % Alternativa: usa Isc dal
file con epsilon
        x0             = [1.8, 1.0, 1e6];          % Guess più robusto per il
dark
        lb_use         = lb_dark;                   % Limiti inferiori per dark
        ub_use         = ub_dark;                   % Limiti superiori per dark
    else
        lambdaFF_local = lambdaFF;                  % In luce, usa penalità FF
        Isc_eff        = ds.Isc_A;                  % Isc reale per la curva in
luce
        lb_use         = lb;                        % Limiti inferiori per luce
        ub_use         = ub;                        % Limiti superiori per luce
    end

    % --- Residuo di fit: scarto sui punti + eventuale penalità FF
    resfun = @(x) residual_IV_for_fit_percurve( ... % Handle funzione residuo
per lsqnonlin
                    x, modelName, Vexp, Iexp, ...
                    ds.G_value, ds.Voc_V, Isc_eff, lambdaFF_local, VMIN_FIT);

    % --- Ottimizzazione con lsqnonlin
    % Se fallisce con x0, riprova con x0_default (robustezza)
    try
        evalc('[xh, resnorm, r, exitflag, output] = lsqnonlin(resfun, x0,
lb_use, ub_use, opts);'); % Esegue lsqnonlin sopprimendo output a video
    catch
        evalc('[xh, resnorm, r, exitflag, output] = lsqnonlin(resfun,
x0_default, lb_use, ub_use, opts);'); % Retry con guess default
    end

    iters = NaN;                                    % Inizializza contatore
iterazioni a NaN
    if exist('output','var') && isfield(output,'iterations') % Se 'output' ha
il campo iterations
        iters = output.iterations;                  % Leggi numero di
iterazioni eseguite
    end

    % --- Metriche
    n_fit  = xh(1);  Rs_fit = xh(2);  Rsh_fit = xh(3);   % Parametri stimati
dal fit
    RMSE   = sqrt(mean(r.^2, 'omitnan'));               % Radice errore
quadratico medio sui residui

    % --- Simulazione finale con parametri stimati (stesso Isc_eff usato nel
fit)
    [Vsim, Isim] = runSimGetVI(modelName, ds.G_value, ds.Voc_V, Isc_eff, n_fit,
Rs_fit, Rsh_fit); % Simula con parametri stimati
    [Vsim, Isim] = uniqueXYavg(Vsim, Isim);                % Ordina e media
duplicati per stabilità
```

```
    % --- Fill Factor: da curva simulata vs da valori sperimentali (se
disponibili)
    if isDark
        %FF_fit = NaN;                                      % Opzione: non
calcolare FF in dark (senza significato fisico)
        FF_fit = ff_from_curve(Vsim, Isim, ds.Voc_V, max(abs(ds.Isc_A), 1e-
12)); % Calcolo FF anche in dark (solo indicativo)
    else
        FF_fit = ff_from_curve(Vsim, Isim, ds.Voc_V, ds.Isc_A);   % FF dalla
curva simulata in luce
    end

    FF_exp = NaN;                                           % Inizializza FF
sperimentale
    if isfinite(ds.Vmax_V) && isfinite(ds.Imax_A) && isfinite(ds.Voc_V) &&
isfinite(ds.Isc_A) ...
            && ds.Voc_V > 0 && abs(ds.Isc_A) > 1e-12     % Se sono disponibili
Voc/Isc e punto di massima potenza
        FF_exp = abs(ds.Vmax_V * ds.Imax_A) / (ds.Voc_V * abs(ds.Isc_A)); % FF
sperimentale da Vmax*Imax
    elseif isfinite(ds.Pmax_W) && isfinite(ds.Voc_V) && isfinite(ds.Isc_A) ...
            && ds.Voc_V > 0 && abs(ds.Isc_A) > 1e-12     % In alternativa, se
c'è Pmax
        FF_exp = abs(ds.Pmax_W) / (ds.Voc_V * abs(ds.Isc_A));              %
FF sperimentale da Pmax
    end

    % --- Salvataggio risultati
    Fitted_Parameters(k).G_label  = string(ds.G_label);  % Etichetta G testuale
    Fitted_Parameters(k).G_value  = ds.G_value;          % Valore G numerico
    Fitted_Parameters(k).n        = n_fit;               % Parametro n stimato
    Fitted_Parameters(k).Rs       = Rs_fit;              % Parametro Rs stimato
    Fitted_Parameters(k).Rsh      = Rsh_fit;             % Parametro Rsh
stimato
    Fitted_Parameters(k).Voc      = ds.Voc_V;            % Voc dal dataset
    Fitted_Parameters(k).Isc      = ds.Isc_A;            % Isc dal dataset
    Fitted_Parameters(k).exitflag = exitflag;           % Codice di uscita
dell'ottimizzatore
    Fitted_Parameters(k).rmse     = RMSE;               % RMSE sui residui
    Fitted_Parameters(k).iterations = iters;            % Iterazioni eseguite
    Fitted_Parameters(k).Vexp     = Vexp;               % Vettore V
sperimentale (filtrato)
    Fitted_Parameters(k).Iexp     = Iexp;               % Vettore I
sperimentale (filtrato)
    Fitted_Parameters(k).Vsim     = Vsim;               % Vettore V simulato
    Fitted_Parameters(k).Isim     = Isim;               % Vettore I simulato
    Fitted_Parameters(k).FF_fit   = FF_fit;             % FF da curva simulata
    Fitted_Parameters(k).FF_exp   = FF_exp;             % FF da dati
sperimentali

    % --- Riepilogo sintetico a video
    motivo_stop = '';                                   % Inizializza stringa
motivo stop
    if exist('output','var') && isfield(output,'message') &&
~isempty(output.message) % Se è presente un messaggio di stop
        msg_lines = regexp(strtrim(output.message), '\r?\n', 'split'); % Spezza
su righe
```

```
        motivo_stop = msg_lines{1};                     % Prendi la prima riga
(riassunto)
    end

    fprintf('✓ G = %-6g W/m^2 | iterazioni = %-3d | Resnorm = %.3e | RMSE =
%.3e | Stop: %s\n', ...
            ds.G_value, iters, resnorm, RMSE, motivo_stop);  % Stampa riga di
riepilogo per la curva k

end

disp('✓ I parametri del fitting sono stati salvati in "Fitted_Parameters".');
% Conferma salvataggio risultati
fprintf(  '\n');
% Riga vuota per leggibilità

% --- Metti nel workspace + tabella per analisi esterna
Fitted_Parameters_Table = struct2table(Fitted_Parameters);
% Converte struct in tabella
assignin('base','Fitted_Parameters',       Fitted_Parameters);
% Esporta struct nel base workspace
assignin('base','Fitted_Parameters_Table', Fitted_Parameters_Table);
% Esporta tabella nel base workspace

% ================== Grafici: Confronto I–V (Sperimentale vs Simulata)
==================
figure('Name','Confronto I–V (Sperimentale vs Simulata - fit)','Color','w');  %
Crea figura dedicata
tiledlayout('flow','Padding','compact','TileSpacing','compact');
% Layout a tiles con spazi compatti
for k = 1:nSets
% Cicla su tutte le curve fitted
    fp = Fitted_Parameters(k);
% Accorcia riferimento
    if isempty(fp.Vsim), continue; end
% Salta se la curva simulata è vuota

    nexttile; hold on; grid on; box on;
% Nuovo riquadro, abilita griglia e riquadro

    mexpP = isfinite(fp.Vexp) & isfinite(fp.Iexp) & fp.Vexp >= VMIN_PLOT;
% Maschera dati sperimentali da plottare
    msimP = isfinite(fp.Vsim) & isfinite(fp.Isim) & fp.Vsim >= VMIN_PLOT;
% Maschera dati simulati da plottare

    plot(fp.Vexp(mexpP), fp.Iexp(mexpP), 'o', 'MarkerSize',4,
'DisplayName','Sperimentale'); % Plot punti sperimentali
    plot(fp.Vsim(msimP), fp.Isim(msimP), '-', 'LineWidth',1.4,
'DisplayName','Simulata (fit)'); % Plot curva simulata

    xlabel('V [V]'); ylabel('I [A]');
% Etichette assi
    title(sprintf('G = %s', fp.G_label));
% Titolo tile con etichetta G
    legend('Location','best');
% Legenda posizionata automaticamente
end
```

```
% ================== Stampa tabella riassuntiva in Command Window
==================
fprintf('\nCaso  G [W/m^2]   n       Rs [Ω]      Rsh [Ω]     Voc [V]    Isc [A]
FF (fit)    Iter\n'); % Intestazione tabella
fprintf(  '-------------------------------------------------------------------
----------------------\n'); % Riga separatrice
for k = 1:nSets
% Cicla sui risultati
    fp = Fitted_Parameters(k);
% Alias
    if ~isfinite(fp.n), continue; end
% Salta se fit non valido

    % FF: vuota se G<=0 (dark)
    if isfinite(fp.G_value) && fp.G_value <= 0
% Se dark
        ffStr = '';
% Non stampare FF
    else
        ffStr = '–';
% Placeholder
        if isfinite(fp.FF_fit), ffStr = sprintf('%.4g', fp.FF_fit); end
% Sostituisci con FF numerico se presente
    end

    iterStr = '–';
% Placeholder iterazioni
    if isfield(fp,'iterations') && isfinite(fp.iterations)
% Se disponibile
        iterStr = sprintf('%d', fp.iterations);
% Stampa numero iterazioni
    end
    fprintf('%-4d  %-10.1f %-7.3f %-11.4e %-11.4e %-9.4f %-9.3e %-10s %5s', ...
% Riga formattata con valori principali
        k, fp.G_value, fp.n, fp.Rs, fp.Rsh, fp.Voc, fp.Isc, ffStr, iterStr);
    if isfinite(fp.FF_exp) && ~(isfinite(fp.G_value) && fp.G_value<=0)
% Se FF exp valido e non dark
        fprintf('  (FF exp: %.4f)', fp.FF_exp);
% Aggiungi FF sperimentale
    end
    fprintf('\n');
% A capo
end
fprintf('-------------------------------------------------------------------
---------------------\n'); % Riga chiusura tabella

open_system(fullfile(pwd,'PV_SC_Model_IV.slx'));
% Apre il modello Simulink a fine script

%% ================== FUNZIONI LOCALI ==================
function setInitialParameters()
% setInitialParameters — Popola nel base workspace i parametri default
richiesti dal modello. % Descrizione funzione helper
% Motivazione: molte maschere/From Workspace leggono direttamente dal base;
avere default sensati evita errori quando il modello viene lanciato standalone.
```

```
    P.Tref  = 298.15;      P.Kbolz = 1.380649e-23;  P.q = 1.602176634e-19;   %
Costanti fisiche base
    P.Ki    = 1.3e-3;      P.Eg    = 1.12;          P.Rs  = 1e-4;            %
Parametri PV generici
    P.Rsh   = 1e3;         P.NsPV  = 1;             P.NpPV = 1;               %
Configurazione stringa PV
    P.n     = 1.2;         P.G     = 1;
% Fattore idealità e G default
    P.A_cm2 = 1.44;        P.A     = P.A_cm2*1e-4;
% Area in cm^2 e in m^2
    P.Isc   = 5.711;       P.Voc   = 0.577;         P.Top  = P.Tref;
% Isc, Voc, temperatura operativa
    P.Irs0 = 1e-12;
% Corrente di saturazione di riferimento

    % Parametri supercap (se il modello li richiede; altrimenti restano
innocui)
    P.R1=16.9; P.C0=0.04; P.Cv=0.0061; P.R2=1000; P.C2=0.31; P.NpSC=1;
P.NsSC=1; % Parametri SC

    % Pubblica tutto nel base workspace
    fn = fieldnames(P);
% Lista dei nomi campo
    for i = 1:numel(fn)
% Cicla su ogni campo
        assignin('base', fn{i}, P.(fn{i}));
% Assegna nel base workspace
    end
end

function [data, summary] = importaDatiSperimentali(folderPath)
% importaDatiSperimentali — Legge tutti i .txt nella cartella e costruisce una
struct per curva. % Descrizione
% Formato atteso:
%   - header con righe tipo "Area (cm^2): <val>", "Isc (A): <val>", "Voc (V):
<val>", ecc.  % Spiega l'header atteso
%   - tabella con colonne "Voltage (V)" e "Current density (mA/cm^2)" (titoli
presenti su una riga) % Colonne dati attese
% Ritorna:
%   data   — struct array con campi utili alla simulazione/fit
% Output 1
%   summary— tabella riassuntiva senza le colonne lunghe (serie V/J/I)
% Output 2

    if nargin<1 || isempty(folderPath), folderPath = fullfile(pwd,'Dati
Sperimentali'); end % Default path se non fornito
    if ~isfolder(folderPath), error('Cartella "%s" non trovata.', folderPath);
end          % Validazione cartella

    files = dir(fullfile(folderPath,'*.txt'));
% Elenco dei .txt nella cartella
    if isempty(files), data=struct([]); summary=table(); return; end
% Se non ci sono file, ritorna vuoto

    template =
struct('FileName',"",'G_label',"",'G_value',NaN,'Area_cm2',NaN,'Isc_A',NaN, ...
% Template per ogni file
```

```
'Voc_V',NaN,'Pmax_W',NaN,'Imax_A',NaN,'Vmax_V',NaN,'FF',NaN,'Efficiency_pct',Na
N, ...

'Jsc_mAcm2',NaN,'Voltage_V',[],'CurrentDensity_mAcm2',[],'Current_A',[]);

    nFiles = numel(files);             % Numero file trovati
    data   = repmat(template, nFiles, 1);  % Preallocazione array struct
    w = 0;                             % Contatore reali caricati (per
eventuali skip)

    headerMap = {                      % Mappa “testo header file” -> “nome
campo nella struct”
        'Area (cm^2)','Area_cm2'
        'Isc (A)','Isc_A'
        'Voc (V)','Voc_V'
        'Pmax (W)','Pmax_W'
        'Imax (A)','Imax_A'
        'Vmax (V)','Vmax_V'
        'FF','FF'
        'Efficiency (%)','Efficiency_pct'
        'Jsc (mA/cm^2)','Jsc_mAcm2'};

    for k = 1:numel(files)             % Cicla su ogni file .txt
        fname  = files(k).name;        % Nome del file
        fpath  = fullfile(files(k).folder, fname); % Percorso completo del file
        Glabel = extractGlabelFromFileName(fname); % Estrae etichetta G dal
nome file
        Gvalue = labelToDouble(Glabel); % Converte etichetta G in valore
numerico

        lines = strip(readlines(fpath));   % legge tutto il file come vettore
di righe (stringhe) rimuovendo spazi ai bordi

        % --- Parse header chiave:valore
        H = struct();                  % Struct temporanea per header estratti
        for i = 1:size(headerMap,1)    % Cicla su ogni chiave attesa
            H.(headerMap{i,2}) = findHeaderValue(lines, headerMap{i,1}); %
Estrae il valore numerico associato
        end

        % --- Trova intestazione delle colonne dati (Voltage, Current density)
        hdrIdx = find(contains(lines,"Voltage (V)",'IgnoreCase',true) & ...
% Cerca riga con entrambe le parole chiave
                      contains(lines,"Current density",'IgnoreCase',true), 1,
'first');
        if isempty(hdrIdx)                % Se intestazione non trovata
            error('Header colonne non trovato in "%s". Attesi "Voltage (V)" e
"Current density".', fname); % Errore bloccante
        end

        % --- Estrai blocco dati e converti in numeric
        blk = lines(hdrIdx+1:end);         % Prende tutte le righe dopo
l’intestazione
        blk = blk(~(blk==""));             % Elimina righe vuote

        V = []; J = [];                    % Prepara vettori per Voltage e
Current density
```

```
        for r = 1:numel(blk)              % Cicla su ogni riga dati
            row    = strrep(char(blk(r)), ',', '.');       % Supporta decimale
con virgola -> punto
            tokens = regexp(row,'[-+0-9eE\.]+','match');   % Estrae tutte le
“parole” numeriche
            if numel(tokens) >= 2                           % Se ci sono almeno
due numeri
                v = str2double(tokens{1});                 % Primo numero =
tensione
                j = str2double(tokens{2});                 % Secondo numero =
densità di corrente
                if isfinite(v) && isfinite(j)              % Valori validi?
                    V(end+1,1) = v; %#ok<AGROW>            % Accoda a V
(colonna)
                    J(end+1,1) = j; %#ok<AGROW>            % Accoda a J
(colonna)
                end
            end
        end

        % --- Costruisci entry
        entry = template;                 % Parte dal template vuoto
        entry.FileName = fname;           % Salva nome file
        entry.G_label  = Glabel;          % Etichetta G
        entry.G_value  = Gvalue;          % Valore G numerico
        entry.Area_cm2 = H.Area_cm2;      % Area dal header
        entry.Isc_A    = H.Isc_A;         % Isc dal header
        entry.Voc_V    = H.Voc_V;         % Voc dal header
        entry.Pmax_W   = H.Pmax_W;        % Pmax dal header
        entry.Imax_A   = H.Imax_A;        % Imax dal header
        entry.Vmax_V   = H.Vmax_V;        % Vmax dal header
        entry.FF       = H.FF;            % FF dal header
        entry.Efficiency_pct = H.Efficiency_pct; % Efficienza dal header
        entry.Jsc_mAcm2      = H.Jsc_mAcm2;      % Jsc dal header
        entry.Voltage_V           = V;           % Colonna tensioni
        entry.CurrentDensity_mAcm2= J;           % Colonna densità corrente

        % Corrente assoluta I [A] = J [mA/cm^2] * Area[cm^2] * 1e-3
        if isfinite(entry.Area_cm2)                   % Se area valida
            entry.Current_A = entry.CurrentDensity_mAcm2 * (entry.Area_cm2 *
1e-3); % Converte J in I assoluta
        else
            entry.Current_A = nan(size(entry.CurrentDensity_mAcm2));
% Altrimenti NaN
        end

        w = w + 1;                        % Incrementa contatore effettivi
        data(w,1) = entry;                % Inserisce entry nell’array struct
    end

    if w < nFiles                          % Se alcuni file sono stati scartati
        data = data(1:w,1);                % Ridimensiona l’array ai soli
caricati
    end

    % --- Ordina per G crescente e costruisci summary
```

```
    summary =
struct2table(rmfield(data,{'Voltage_V','CurrentDensity_mAcm2','Current_A'})); %
Tabella senza serie lunghe
    [~,ord] = sort(summary.G_value);       % Ordina per valore G
    summary = summary(ord,:);              % Riordina tabella
    data    = data(ord);                   % Riordina struct coerentemente
end

function G_label = extractGlabelFromFileName(fname)
% Estrae stringa tipo "X,YYY W" dal nome file se presente; altrimenti "N/D". %
Descrizione
    tok = regexp(string(fname), "G\s*([0-9]+(?:[.,][0-9]+)?)\s*W", 'tokens',
'ignorecase'); % Cerca pattern "G <numero> W"
    if ~isempty(tok) && ~isempty(tok{1})    % Se trovato
        numTxt = string(tok{1}{1});         % Prende la parte numerica come
stringa
        if contains(numTxt,'.'), numTxt = replace(numTxt,'.',','); end %
Normalizza con virgola nell'etichetta
        G_label = numTxt + " W";            % Ritorna etichetta tipo "X,YYY W"
    else
        G_label = "N/D";                    % Non disponibile
    end
end

function val = labelToDouble(G_label)
% Converte "X,YYY W" -> double X.YYY (W/m^2 equivalenti in questa etichetta) %
Descrizione
    s = upper(strrep(G_label,' ', ''));     % Rimuove spazi e porta a maiuscolo
    s = erase(s,"W");                       % Elimina la lettera W
    s = strrep(s, ',', '.');                % Sostituisce virgola con punto per
conversione numerica
    val = str2double(s);                    % Converte in double (NaN se
fallisce)
end

function v = findHeaderValue(lines, key)
% Cerca la riga che contiene 'key' e ne estrae l'ultimo numero (supporta , e .)
% Descrizione
    idx = find(contains(lines, key, 'IgnoreCase', true), 1, 'first'); % Trova
la prima riga che contiene la chiave
    if isempty(idx), v = NaN; return; end     % Se non trovata, ritorna NaN
    row    = strrep(char(lines(idx)), ',', '.');      % Normalizza decimale
    tokens = regexp(row,'[-+0-9eE\.]+','match');      % Estrae tutti i numeri
nella riga
    if isempty(tokens), v = NaN; else, v = str2double(tokens{end}); if
~isfinite(v), v = NaN; end, end % Prende l'ultimo numero
end

function [Vsim, Isim] = runSimGetVI(modelName, Gval, Voc, Isc, n_id, Rs, Rsh)
% runSimGetVI — Esegue il modello Simulink con parametri fissati e ritorna i
vettori V,I. % Descrizione high-level
% Note importanti:
%   - Il modello deve esporre i segnali 'VoutPV' e 'IoutPV' negli Outputs
(out).          % Requisito segnali
%   - Le variabili G, Voc, Isc, n, Rs, Rsh sono lette nel modello (From
Workspace/Mask).  % Parametri passati via base
%   - La variabile 'G' viene passata come [t u] per compatibilità con From
Workspace.      % Formato segnale G
```

```
    % --- Controllo di presenza LUT nel base workspace (usate da Lookup nel
modello)
    needLUT = @(v) evalin('base', sprintf('exist(''%s'',''var'')', v)) ~= 1;
% Funzione anonima: verifica esistenza variabile
    if needLUT('G_bp') || needLUT('Isc_tab') || needLUT('Voc_tab')
% Se mancano le LUT necessarie
        error(['Mancano G_bp/Isc_tab/Voc_tab nel base workspace. ' ...
               'Creare le LUT prima di chiamare runSimGetVI (vedi blocco
PREPARA LUT).']); % Errore esplicativo
    end

    % --- Parametri per il modello
    assignin('base','G',   [0 Gval; 1 Gval]);  % Passa G come segnale costante
[tempo valore]
    assignin('base','Voc', Voc);               % Passa Voc al modello
    assignin('base','Isc', Isc);               % Passa Isc al modello
    assignin('base','n',   n_id);              % Passa n (idealità)
    assignin('base','Rs',  Rs);                % Passa Rs
    assignin('base','Rsh', Rsh);               % Passa Rsh

    % --- Sopprimi warning/issue dei From Workspace (no-op se il modello non ha
questi blocchi)
    setFromWorkspaceST0(modelName);            % Imposta Sample Time = 0 per i
From Workspace (se presenti)

    % --- Esecuzione simulazione (ritorna oggetto 'out' con segnali nominati)
    out = sim(modelName, 'ReturnWorkspaceOutputs','on'); % Esegue simulazione e
ritorna outputs in 'out'

    % --- Estrai i segnali V,I (usa helper che gestisce diversi tipi:
timeseries, numeric, timetable)
    Vsim = fetch_signal_vector('VoutPV', out); % Estrae vettore V simulato
    Isim = fetch_signal_vector('IoutPV', out); % Estrae vettore I simulato

    % --- Pulizia NaN/Inf
    Vsim = Vsim(:); Isim = Isim(:);            % Forza vettori colonna
    ok = isfinite(Vsim) & isfinite(Isim);      % Maschera di valori finiti
    Vsim = Vsim(ok); Isim = Isim(ok);          % Filtra non-finiti
end

function r = residual_IV_for_fit_percurve(x, modelName, Vexp, Iexp, Gval, Voc,
Isc_eff, lambdaFF, VMIN_FIT)
% residual_IV_for_fit_percurve — Restituisce il vettore residuo per lsqnonlin.
% Descrizione
% Componenti del residuo:
%   1) Scarto sui punti I(V) sperimentali (per V >= VMIN_FIT), pesato più dove
I è piccola; % Spiega la prima parte del residuo
%   2) (opzionale) Penalità su differenza di Fill Factor (FF_sim - FF_exp)
scalata da lambdaFF. % Spiega la componente FF

    % Simula con i parametri correnti x = [n, Rs, Rsh]
    [Vsim, Isim] = runSimGetVI(modelName, Gval, Voc, Isc_eff, x(1), x(2),
x(3)); % Simula il modello con parametri correnti
    [Vsim, Isim] = uniqueXYavg(Vsim, Isim);
% Ordina e media duplicati

    if numel(Vsim) < 3                         % Se troppo pochi punti simulati
```

```
        r = 1e3 * ones(size(Iexp));           % Restituisci residui grandi
(spinge l'ottimizzatore altrove)
        return;                                % Esci dalla funzione
    end

    % --- Seleziona sottoinsieme "stabile" per il fit (evita V ~ 0)
    mexp = isfinite(Vexp) & isfinite(Iexp) & (Vexp >= VMIN_FIT); % Maschera per
punti sperimentali sopra soglia V
    if ~any(mexp)                             % Se nessun punto valido
        r = 1e3 * ones(size(Iexp));           % Penalizza fortemente
        return;                               % Esci
    end
    Vq = Vexp(mexp);                          % Vettore tensioni dove interpolare
la simulazione

    % --- Interpola I_sim nei punti V sperimentali (pchip robusto, fallback
linear)
    try
        Isim_on_exp = interp1(Vsim, Isim, Vq, 'pchip', 'extrap'); % Interpola
con pchip per forma morbida
    catch
        Isim_on_exp = interp1(Vsim, Isim, Vq, 'linear', 'extrap'); % Fallback
lineare se pchip fallisce
    end

    % --- Pesi: enfatizza regioni a bassa corrente (tipicamente più
informative)
    w = 1 ./ max(1e-9, abs(Iexp(mexp)));      % Pesi inversamente proporzionali
alla corrente (evita divisione per zero)

    % --- Al buio: dai più peso alla regione vicino a Voc (comportamento diodo)
    if Gval <= 0 && isfinite(Voc) && Voc > 0  % Se dark e Voc valido
        alphaV = 3;                            % Esponente che accentua pesi
vicino a Voc
        w = w .* max( (Vq./Voc).^alphaV, 1e-2 ); % Scala i pesi con funzione di
V/Voc (min 1e-2 per non annullare)
    end

    % --- Residuo base (scarto sulla corrente normalizzato dai pesi)
    rI = (Isim_on_exp - Iexp(mexp)) .* sqrt(w); % Residuo pesato punto-a-punto

    % --- Penalità sul Fill Factor (solo in luce e se definibili)
    if lambdaFF > 0 && isfinite(Voc) && Voc > 0 && abs(Isc_eff) > 0        % Se
attiva e definibile
        FF_sim = ff_from_curve(Vsim, Isim, Voc, Isc_eff);                  %
Calcola FF dalla curva simulata
        FF_exp = ff_from_curve(Vexp(mexp), Iexp(mexp), Voc, Isc_eff);      %
Stima FF dalla curva sperimentale (stessi punti selezionati)
        if isfinite(FF_sim) && isfinite(FF_exp)                            % Se
entrambi finiti
            r = [rI; lambdaFF * (FF_sim - FF_exp)];                        %
Aggiunge componente FF al vettore residuo
            return                                                         %
Esce
        end
    end

    r = rI;  % solo componente sui punti se penalità FF non applicabile
```

```
end

function FF = ff_from_curve(V, I, Voc, Isc)
% ff_from_curve — Calcola il Fill Factor da una curva I–V discreta.          %
Descrizione
% Definizione: FF = Pmax / (Voc * |Isc|) con Pmax = max(V.*|I|).             %
Formula
    if ~(isfinite(Voc) && isfinite(Isc)) || Voc <= 0 || abs(Isc) <= 1e-9     %
Controlli di validità
        FF = NaN;
        return;
    end
    V = V(:); I = I(:);                          % Vettori colonna
    ok = isfinite(V) & isfinite(I);              % Maschera finiti
    V = V(ok); I = I(ok);                        % Filtra non-finiti
    if numel(V) < 2, FF = NaN; return; end       % Necessari almeno 2 punti

    [V, I] = uniqueXYavg(V, I);                  % Ordina e media duplicati
    P = V .* abs(I);                             % Potenza istantanea
    Pmax = max(P, [], 'omitnan');                % Massimo della potenza
    FF = Pmax / (Voc * abs(Isc));                % Normalizzazione per
definizione di FF
end

function [xU, yU] = uniqueXYavg(x, y)
% uniqueXYavg — Ordina per x crescente, raggruppa duplicati e ne fa la media su
y. % Descrizione
    x = x(:); y = y(:);                          % Vettori colonna
    ok = isfinite(x) & isfinite(y);              % Maschera finiti
    x = x(ok); y = y(ok);                        % Filtra
    [x, ord] = sort(x);                          % Ordina x crescente e ottieni
l'ordine
    y = y(ord);                                  % Riordina y coerentemente
    [xU, ~, ic] = unique(x, 'stable');           % Trova valori unici preservando
l'ordine
    yU = accumarray(ic, y, [], @mean);           % Media dei corrispondenti y per
duplicati
end

function vec = fetch_signal_vector(varName, out)
% fetch_signal_vector — Estrae un vettore numerico dal base workspace o da
'out'. % Descrizione
% Accetta:
%   - timeseries (usa .Data)
%   - numeric matrix [t y] (prende la 2a colonna), o vettore (prende così
com'è)
%   - timetable (prende la prima colonna)
% Ordine di ricerca: base workspace -> 'out' (WorkspaceOutputs della
simulazione).
    vec = [];                                    % Inizializza output
    try
        % 1) Prova dal base workspace (alcuni modelli lasciano variabili
globali)
        if evalin('base', sprintf('exist(''%s'',''var'')', varName)) == 1
% Se la variabile esiste nel base
            val = evalin('base', varName);
% Recupera la variabile
```

```
            vec = local_cast_vector(val);
% Converte in vettore numerico se possibile
            if ~isempty(vec), return; end
% Se ottenuto vettore, esci
        end
        % 2) Prova dall'oggetto 'out' ritornato da sim()
        if nargin >= 2 && ~isempty(out)
% Se 'out' è disponibile
            try
                ts = out.get(varName);
% Tenta di leggere il segnale per nome
                vec = local_cast_vector(ts);
% Converte in vettore numerico
            catch
                % ignorabile: variabile non presente in out                    %
Se non presente, prosegue
            end
        end
    catch
        vec = [];
% In caso di errore, ritorna vuoto
    end

    function v = local_cast_vector(val)
% Helper nested per normalizzare il tipo
        v = [];
        if isa(val,'timeseries')
% Se timeseries
            v = val.Data(:);
% Prendi i dati come vettore colonna
        elseif isnumeric(val)
% Se numerico
            if ismatrix(val) && size(val,2)>=2
% Se matrice [t y] o simile
                v = val(:,2);
% Prendi la seconda colonna come segnale
            else
                v = val(:);
% Altrimenti vettorializza
            end
        elseif istimetable(val)
% Se timetable
            v = val{:,1}; v = v(:);
% Prendi la prima colonna come vettore
        end
    end
end

function setFromWorkspaceST0(modelName)
% setFromWorkspaceST0 — Tenta di impostare i From Workspace su 'Sample time =
0' % Descrizione
% e disabilitare warning fastidiosi. Se Simulink non è disponibile o il modello
% non contiene tali blocchi, non fa nulla (fail-safe).
    try
        if ~bdIsLoaded(modelName), load_system(modelName); end  % Carica il
modello se non già caricato
        blks = find_system(modelName, 'BlockType','FromWorkspace'); % Trova
tutti i blocchi From Workspace
```

```
        for i = 1:numel(blks)                                       % Cicla sui
blocchi trovati
            try
                set_param(blks{i}, 'SampleTime', '0');              % Imposta
Sample Time a 0 (ereditato dal solver)
                set_param(blks{i}, 'Interpolate', 'on');            % Abilita
interpolazione lineare
                % Formato dati struttura con campi time/values opzionale:
                % qui lasciamo il default, usiamo [t u].             % Nota
operativa
            catch
                % ignora blocchi protetti/locked                    % Se blocco
non modificabile, passa oltre
            end
        end
    catch
        % Simulink non disponibile o modello non accessibile: nessuna azione
necessaria % Fail-safe: non solleva errore
    end
end

function s = exitflag_to_text(exitflag)
% exitflag_to_text – Converte l'exitflag numerico in un breve testo
esplicativo. % Descrizione
    switch exitflag
        case {1,2,3,4}
            s = 'Local minimum found.';                         % Minimo locale
trovato
        case 0
            s = 'Stopped: max iterations/func-count.';          % Stop per limiti
su iterazioni/valutazioni
        case -1
            s = 'Stopped by output/plot function.';             % Stop da
funzione esterna
        case -2
            s = 'No feasible point found.';                     % Nessun punto
ammissibile
        otherwise
            s = sprintf('Stopped (exitflag=%d).', exitflag);  % Caso generico
con codice
    end
end
```

Calibrazione_supercap.m

```
% === Target ===
      %%% ! SCEGLIERE QUI IL TARGET DEL SUPERCAPACITORE ! %%%

Vmax = 3.3;           % Tensione massima di operatività del carico [V]
Vmin = 1.8;           % Tensione minima di operatività del carico [V]
Iload = 10e-6;        % Corrente richiesta dal carico [A]
Autonomy = 17*3600;   % Autonomia desiderata nell'alimentazione del carico in
un caso ideale senza perdite [s]

Q_target = Iload * Autonomy;   % Carica minima necessaria del supercap per
soddisfare l'autonomia richiesta [C]

C_need = Q_target/(Vmax-Vmin); % Capacità necessaria per mantenere l'autonomia
nella dinamica del carico

%C_need = ***** ;              % Impostazione diretta della capacità desiderata
C_need = round (C_need,3);

Qmin = C_need*Vmin;           % Carica necessaria a raggiungere Vmin
Qmax = Q_target + Qmin;       % Carica necessaria a raggiungere Vmax

      %%% ! FINE SCELTA DEL TARGET DEL SUPERCAPACITORE ! %%%

% === Valori iniziali (dal paper di Cabrane) ===
Rf = 15e6;
R1 = 0.008;
R2 = 1;
C0 = 2170;
Cv = 520;
C2 = 150;

% === Fattore di scalamento k ===
Q1_old = C0*Vmax + 0.5*Cv*Vmax^2;
Q2_old = C2*Vmax;
k = Qmax / (Q1_old + Q2_old);

% === Capacità scalate ===
C0_new = k*C0;
C0_new = round (C0_new,3);
Cv_new = k*Cv;
Cv_new = round (Cv_new,3);
C2_new = k*C2;
C2_new = round (C2_new,3);
C1_max_new = C0_new + Cv_new*Vmax;
C1_max_new = round (C1_max_new,3);

% === Cariche scalate ===
Q1_new = C0_new*Vmax + 0.5*Cv_new*Vmax^2;
Q2_new = C2_new*Vmax;

% === Resistenze scalate per mantenere le stesse tau ===
R1_new = R1 / k;
R1_new = round (R1_new,1);
R2_new = R2 / k;
R2_new = round (R2_new,1);
```

```
Rf_new = Rf / k;

% === Costanti di tempo tau scalate ===
Tau1_max = R1_new*C1_max_new;
Tau2 = R2_new*C2_new;

%-------------------------------------------------------------------------


fprintf('\n');
fprintf('Riassunto delle variazioni nei parametri del supercap:\n');
fprintf('\n');

disp(table([Rf Rf_new]', [R1 R1_new]', [R2 R2_new]', [C0 C0_new]', [Cv
Cv_new]', [C2 C2_new]', ...
    'VariableNames', {'Rf[Ω]', 'R1[Ω]','R2[Ω]','C0[F]','Cv[F]','C2[F]'},
'RowNames', {'old','new'}));

disp(table(Iload', Autonomy/3600', Vmax', Q_target', Qmax', C_need', ...
    'VariableNames', {'Iload[I]', 'Autonomy[ore]', 'Vmax[V]', 'Q_target[C]',
'Qmax[C]', 'C_need[F]'}, 'RowNames', {'chosen parameters'}));

fprintf('\n');
fprintf('Copiare e incollare questi parametri su Simulazioni.m \n'); % ---
Parametri calibrati del supercap (SI) ---
fprintf('nella sezione: --- Parametri calibrati del supercap (SI) --- \n');
fprintf('-------------------------------------\n');
fprintf('C_need = %.12g;\n',C_need);
%fprintf('Rf = %.3e;\n',Rf_new);
fprintf('R1 = %.12g;\n',R1_new);
fprintf('R2 = %.12g;\n',R2_new);
fprintf('C0 = %.12g;\n',C0_new);
fprintf('Cv = %.12g;\n',Cv_new);
fprintf('C2 = %.12g;\n',C2_new);
fprintf('-------------------------------------\n');
fprintf('\n');
```

Simulazioni.m

```
clear
clc
load("Fitted_Parameters.mat");

%% ----------------------------------------
            %%% DEFINIZIONE DEI PARAMETRI INIZIALI %%%
    Tref  = 298.15;
    Kbolz = 1.380649e-23;
    q = 1.602176634e-19;
    Ki    = 1.3e-3;
    Eg    = 1.12;
    A_cm2 = 1.44*10;
    A     = A_cm2*1e-4;
    Isc   = 1.183e-7;
    Voc   = 6.686e-1;
    Top  = Tref;

%% ----------------------------------------
             %%% DEFINIZIONE DEI PARAMETRI DI SIMULAZIONE %%%

% --- Profilo di G(t)
h = 3600;    % Assegnazione alla varibile 'h' dell'equivalente in secondi di
un'ora
T_sim = 24*7;  % Tempo di simulazione in ore

%
% Profilo di G standard (3 giorni)
G_vals = [0, 0, 0, 3, 3, 0, 0, 3, 3, 0, 0, 3, 3, 0, 0]';
% Valori assegnati all'irraggiamento in dei momenti precisi della simulazione
G_time = [0, 1, (8*h)-1, 8*h, (18*h)-1, 18*h, (32*h)-1, 32*h, 42*h, (42*h)+1,
(56*h)-1, 56*h, 66*h, (66*h)+1, T_sim*h]';
% Momenti della simulazione in cui viene cambiato il valore di irraggimento
%

x = 8;          % ora inizio
y = x+10;       % ora fine

% Profilo di G per Nd giorni: inizio alle ore x, fine alle ore y (ogni giorno)
h   = 3600;
x   = 8;          % ora inizio
y   = 18;         % ora fine
Nd  = 7;          % numero giorni
T_sim = 24*Nd;    % [ore]

G_time = [0; 1];  %inizializzazione
G_vals = [0; 0];

for d = 0:Nd-1
    t_rise_m1 = (24*d + x)*h - 1;
    t_rise    = (24*d + x)*h;
    t_set_m1  = (24*d + y)*h - 1;
    t_set     = (24*d + y)*h;

    G_time = [G_time; t_rise_m1; t_rise; t_set_m1; t_set];
    G_vals = [G_vals; 0; 3; 3; 0];
```

```
end

% tempo finale
G_time = [G_time; T_sim*h];
G_vals = [G_vals; 0];

% controllo coerenza
assert(numel(G_time) == numel(G_vals), 'G_time e G_vals devono avere la stessa
lunghezza');
%

% --- PV (SI)
NsPV  = 5;
% Numero di celle fotovoltaiche in serie, aumenta tensione stessa corrente
NpPV  = 1;
% Numero di celle fotovoltaiche in parallelo, aumenta corrente stessa tensione

% --- SuperCap (SI)
NpSC = 1;
% Numero di supercondensatori in parallelo, tensione e carica massima invariata
ma somma delle capacità
NsSC = 1;
% Numero di supercondensatori in serie, tensione e carica massima aumenta ma
capacità divisa

% --- Parametri calibrati del supercap (SI) ---
% ------------------------------------------------------------------------
-----------
%        ***** Copiare e incollare qui sotto i parametri calibrati del supercap
*****
%                      ***** ottenuti da: Calibrazione_supercap.m *****
                          % C_need [F] - R1,R2 [Ω] - C0,C1,C2 [C]
C_need = 0.408;
R1 = 62.3;
R2 = 7789.2;
C0 = 0.279;
Cv = 0.067;
C2 = 0.019;

% ------------------------------------------------------------------------
% --- Carico (SI)
Isens = 4e-6; % Corrente assorbita richiesta dal carico
VSensMax = 3.3; % Massimo valore di tensione accettato per il funzionamento del
carico
VSensMin = 1.8; % Minimo valore di tensione richiesto per il funzionamento del
carico

% --- Leakage (SI)
I_leak_max = 6.6e-6;
Rf = VSensMax / I_leak_max;

% --- Costanti utili per caso ideale senza perdite
I_leak_constant = 0;
Idark_constant =0;

%-------------------------------------------------------------------------

% Lettura dal file della corrente richiesta del carico
```

```
T = readtable('Current consumption.txt', ...
    'Delimiter', '\t', ...
    'ReadVariableNames', true, ...
    'VariableNamingRule', 'preserve');

t_bp = T.("Time(s)");
i_bp = (T.("Current(mA)") * 1e-3);

Tpat = t_bp(end);

if t_bp(end) > t_bp(1)
    t_bp = [t_bp; Tpat];
    i_bp = [i_bp; i_bp(1)];
end

% Rimuove eventuali duplicati sul bordo
[ t_bp, idx ] = unique(t_bp,'stable');
i_bp = i_bp(idx);

% Crea timeseries
Isens_ts = timeseries(i_bp, t_bp);
Isens_ts.Name = 'Isens';

%--------------------------------------------------------------------------

U0   = 0;                 % tensione iniziale desiderata
Q0   = C0*U0 + 0.5*Cv*U0^2;
Qmax = C_need * Voc *NsPV;

           %%% FINE DEFINIZIONE DEI PARAMETRI DI SIMULAZIONE %%%

Autonomy = (C_need*(VSensMax-VSensMin))/((Isens+I_leak_max_C)*h)*0.686;

%
% Calcolo media (aritmetica)
i_mean = mean(i_bp, 'omitnan');

% Conversione in microampere
i_micro = i_bp * 1e6;
i_mean_micro = i_mean * 1e6;

% Plot
figure;
plot(t_bp, i_micro, 'b', 'LineWidth', 1.5); hold on;
yline(i_mean_micro, 'r--', 'LineWidth', 2);

xlabel('Tempo [s]');
ylabel('Corrente [µA]');
title('Corrente e valore medio');
legend('Segnale originale', sprintf('Media = %.3f µA', i_mean_micro),
'Location', 'best');
grid on;

% Calcolo RMS
i_rms = sqrt(mean(i_bp.^2));

% Conversione in microampere
i_micro = i_bp * 1e6;
```

```
i_rms_micro = i_rms * 1e6;
%
% Plot
figure;
plot(t_bp, i_micro, 'b', 'LineWidth', 1.5); hold on;
yline(i_rms_micro, 'r--', 'LineWidth', 2);

xlabel('Tempo [s]');
ylabel('Corrente [µA]');
title('Corrente e valore quadratico medio (RMS)');
legend('Segnale originale', sprintf('RMS = %.3f µA', i_rms_micro), 'Location',
'best');
grid on;


% --- Simulazione del modello e plot dei risultati

open_system(fullfile(pwd,'PV_SC_Model_SIM.slx'));
out = sim('PV_SC_Model_SIM');

plot_IinSC_VinSC(out, G_time, G_vals, C_need, Autonomy, VSensMin, VSensMax,
Isens, NpSC, NsSC, NpPV, NsPV, T_sim);

function plot_IinSC_VinSC(out, G_time, G_vals, C_need, ~, VSensMin, VSensMax,
Isens, NpSC, NsSC, NpPV, NsPV, T_sim)


    % Estrai i timeseries
    tsI = out.IinSC;      % corrente
    tsV = out.VinSC;      % tensione
    tsS = out.Isens_scope;      % corrente

    % Tempo e dati
    t  = tsI.Time;
    yI = tsI.Data*1e6;    % µA
    yV = tsV.Data;        % V
    yS = tsS.Data*1e6;    % µA

    % Taglia i dati
    t_max = T_sim * 3600;
    idx_valid = t <= t_max;
    t = t(idx_valid);
    yI = yI(idx_valid);
    yV = yV(idx_valid);
    yS = yS(idx_valid);

    % Interpolazione diretta senza wrap
    G_cont = interp1(G_time, G_vals, t, 'previous', 'extrap');

    % Plot con doppio asse Y
    figure; hold on;
    %
    yyaxis left
    hI = plot(t, yI, 'b-', 'LineWidth', 1.5);
    ylabel('IinSC [\muA]');
    %
    yyaxis left
    %hS = plot(t-850, yS, 'm-', 'LineWidth', 1.5);
```

```
    %----
    offset = 0;
    tq = t + offset;
    % Clampa dentro [t(1), t(end)] per evitare extrapolazioni
    tq = min(max(tq, t(1)), t(end));
    yS_shift = interp1(t, yS, tq, 'previous');
    hS = plot(t, yS_shift, 'm-', 'LineWidth', 1.5);
    %---
    ylabel('Iload [\muA]');
    %
    yyaxis right
    hV = plot(t, yV, 'r-', 'LineWidth', 1.5);
    ylabel('VinSC [V]');
    %
    %T = G_time(end) - G_time(1);
    %t_wrap = mod(t - G_time(1), T) + G_time(1);
    %G_cont = interp1(G_time, G_vals, t_wrap, 'linear');
    %G_cont = interp1(G_time, G_vals, t, 'previous', 'extrap');

    yyaxis right
    hG = plot(t, G_cont, 'g--', 'LineWidth', 1.5, 'DisplayName','G');

    xlabel('Tempo [ore]');
    %xt = 0:1:T_sim;                % tick ogni ora
    %xt_sec = xt*3600;              % corrispondenti in secondi

    % Tick in ore dentro [0, T_sim]
    xt_hours = 0:1:T_sim;
    xt_sec   = xt_hours * 3600;
    set(gca, 'XTick', xt_sec(xt_sec <= t_max), 'XTickLabel', xt_hours(xt_sec <=
t_max));


    % Limita esplicitamente l'asse X all'intervallo di simulazione
    t_max = T_sim * 3600;          % (già calcolato sopra)
    xlim([0, t_max]);              % blocca i limiti
    set(gca, 'XLimMode','manual'); % evita che l'autoscale li cambi


    %set(gca, 'XTick', xt_sec, ...  % posizioni dei tick
     %    'XTickLabel', xt);        % etichette in ore

    grid on;
    legend([hI hV hG hS], {'IinSC','VinSC','G','Iload'}, 'Location','best');
    %legend([hI hV hG], {'IinSC','VinSC','G'}, 'Location','best');
    title('Profilo di irraggiamento (G), tensione (VinSC) e corrente (IinSC) in
ingresso al supercap in funzione del tempo');
    subtitle({
    %sprintf('Capacità del supercap = %.3f [F], Autonomia stimata = %.1f
[ore]', C_need, Autonomy)
    sprintf('Capacità del supercap = %.3f [F]', C_need)
    sprintf('Dinamica del carico %.1f-%.1f [V], Corrente richiesta dal carico =
%.1f [µA]', VSensMin, VSensMax, Isens*1e6)
    sprintf('%d PV in parallelo, %d PV in serie, %d SC in parallelo, %d SC in
serie', NpPV, NsPV, NpSC, NsSC)
    });

end
```

II. Tavola delle figure

III. Bibliografia

[1] M. Rahmatian, H. Sayyaadi, e M. Ameri, «A novel thermoelectrical model for dye-sensitized solar cells for consideration of the effects of solar irradiation, wavelength, and surface temperature», *Solar Energy*, vol. 266, dic. 2023, doi: 10.1016/j.solener.2023.112140.

[2] D. Warepam, R. S. Dhar, K. J. Singh, e A. Biswas, «Development and Analysis on Micro-Electrical Photovoltaic Model of Dye Sensitized Solar Cell for Optimized Performance», *Journal of The Institution of Engineers (India): Series D*, 2024, doi: 10.1007/s40033-024-00689-6.

[3] A. M. Tayeb, A. A. A. Solyman, M. Hassan, e T. M. Abu el-Ella, «Modeling and simulation of dye-sensitized solar cell: Model verification for different semiconductors and dyes», *Alexandria Engineering Journal*, vol. 61, n. 12, pagg. 9249–9260, dic. 2022, doi: 10.1016/j.aej.2022.02.057.

[4] Z. Cabrane e S. H. Lee, «Electrical and Mathematical Modeling of Supercapacitors: Comparison», *Energies (Basel)*, vol. 15, n. 3, feb. 2022, doi: 10.3390/en15030693.

[5] S. A. Kalogirou, «Solar Energy Engineering Processes and Systems Second Edition». [Online]. Disponibile su: http://store.elsevier.com/

[6] R. Speranza, *Integrated Energy Harvesting and Storage Systems for a Sustainable Future*, documento tecnico interno, 2024.