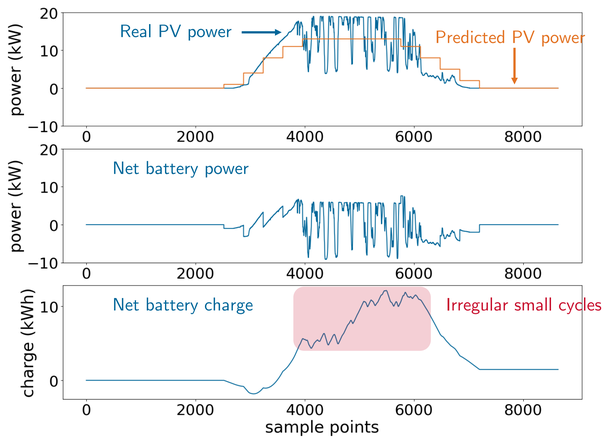
A Real-Time Degradation Estimation Approach for Batteries in PV and Battery Hybrid Plant Operation

Shumeng Wang[1], Mahshid Amirabadi[2], Brad Lehman[3]

[1,2,3]Northeastern University, United States

① Introduction

- Battery storage compensates for the **discrepancies** between forecasted and actual PV power, ensuring capacity firming.
- PV generation is weather-sensitive → Complex battery cycles → Impractical real-time battery degradation estimation.
- **Example:** City of Tallahassee solar farm on Sep 18th, 2020.



Contribution:

- Proposed a new algorithm based on memory stacking and linked list to efficiently compute incremental degradation.
- Accurate real-time battery health monitoring under complex situation is achieved, enabling real-time system optimization.

② Conventional Method

Canonical three-step form (from M. Sandelic & G. Angenendt)

$$deg_{0\to t} = f\left(\mathbb{O}(\mathbf{A}_t)\right) = \sum_{i=1}^{n} a_{cyc}\left(N_{eq}(\Delta SOC_i)\right)^{b_{cyc}}$$

- $\mathbf{A}_t$ is the sequence of SOC samples for period $0 \to t$.
- $\mathbb{O}$ is the rainflow cycle counting algorithm operator.
- f is the degradation function applied to rainflow results.

Limitation: post-processing, not real-time.

Real-time incremental degradation

$$deg_{inc,t\to t+1} = f\left(\mathbb{O}(\mathbf{A}_{t+1})\right) - f\left(\mathbb{O}(\mathbf{A}_t)\right)$$

- $deg_{inc,t\to t+1}$ is the incremental degradation from $t \to t+1$
- Computation complexity with n total samples: $\boldsymbol{\mathcal{O}(n^3)}$

Limitation: significant burden, not suitable for real-time.

This research was supported by Department of Energy, SETO: DE-EE0009340, and also by the Massachusetts Clean Energy Technology.

③ Proposed Real-Time Method

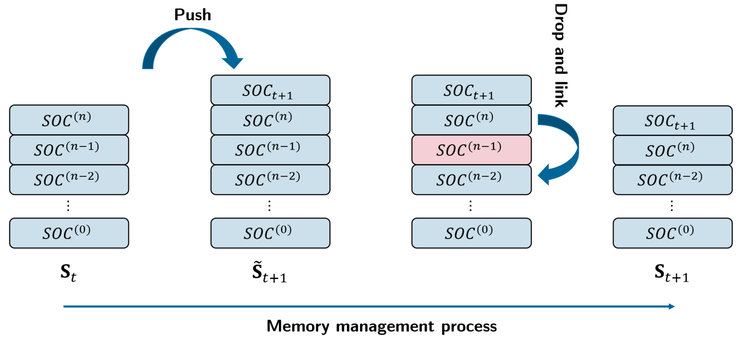
Recurrent structure with memory $\mathbf{S}_t$

$$deg_{inc,t\to t+1} = f\left(\mathbb{O}(\tilde{\mathbf{S}}_{t+1})\right) - f\left(\mathbb{O}(\mathbf{S}_t)\right)$$

- $\tilde{\mathbf{S}}_{t+1} := [\mathbf{S}_t | SOC_{t+1}]$ is the appended memory

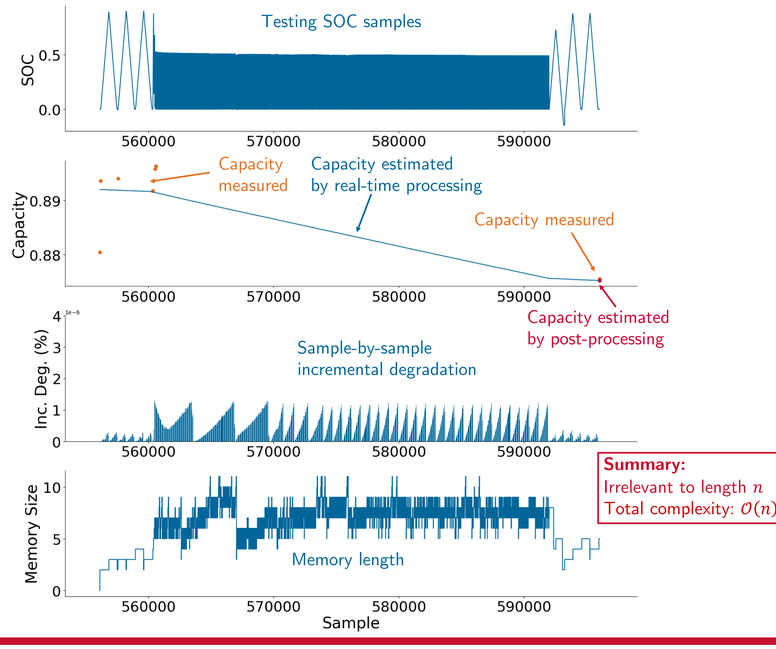
Memory management

- Memory stores critical history SOC data, to accurately determine the marginal degradation of each new sample.
- **"Push" (memory stack):** append on top.
- **"Drop" (linked list):** drops arbitrary samples, re-link the rest stack components, criteria applied.



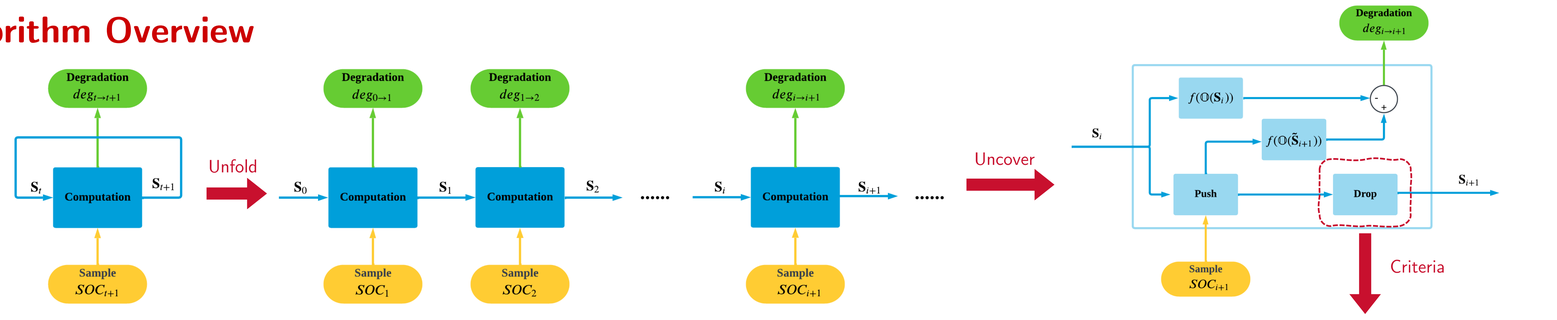
④ Validation Results

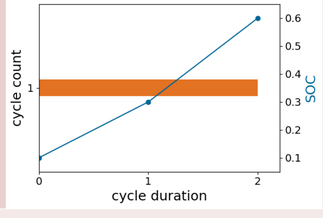
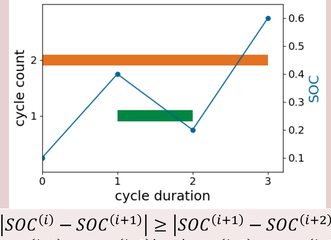
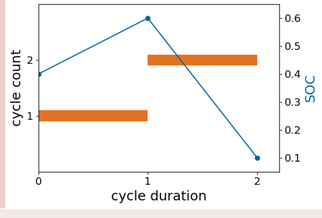
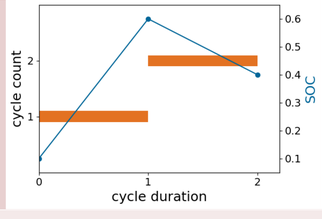
Using the subset of 39907 consecutive samples of LFP battery testing from *Sandia National Laboratories*.



Summary:
Irrelevant to length n
Total complexity: $\mathcal{O}(n)$

Algorithm Overview



Case	Half-cycle Extension	Full-cycle	Unextendible half-cycle	Extendible half-cycle
Example				
Expression	$(SOC^{(i)} - SOC^{(i+1)})(SOC^{(i+1)} - SOC^{(i+2)}) > 0$	$\lvert SOC^{(i)} - SOC^{(i+1)}\rvert \geq \lvert SOC^{(i+1)} - SOC^{(i+2)}\rvert$ $\lvert SOC^{(i+1)} - SOC^{(i+2)}\rvert \leq \lvert SOC^{(i+2)} - SOC^{(i+3)}\rvert$	$\lvert SOC^{(i)} - SOC^{(i+1)}\rvert \leq \lvert SOC^{(i+1)} - SOC^{(i+2)}\rvert$	$\lvert SOC^{(i)} - SOC^{(i+1)}\rvert \geq \lvert SOC^{(i+1)} - SOC^{(i+2)}\rvert$
Drop indicator	$\mathbb{I}(SOC^{(i+1)}) = 0$ $\mathbb{I}(SOC^{(i)}) = \mathbb{I}(SOC^{(i+2)}) = 1$	$\mathbb{I}(SOC^{(i+1)}) = \mathbb{I}(SOC^{(i+2)}) = 0$ $\mathbb{I}(SOC^{(i)}) = \mathbb{I}(SOC^{(i+3)}) = 1$	$\mathbb{I}(SOC^{(i)}) = 0$ $\mathbb{I}(SOC^{(i+1)}) = \mathbb{I}(SOC^{(i+2)}) = 1$	$\mathbb{I}(SOC^{(i)}) = \mathbb{I}(SOC^{(i+1)}) = \mathbb{I}(SOC^{(i+2)}) = 1$