Neural Net-Based MPC Control Law for Battery Pack Balancing

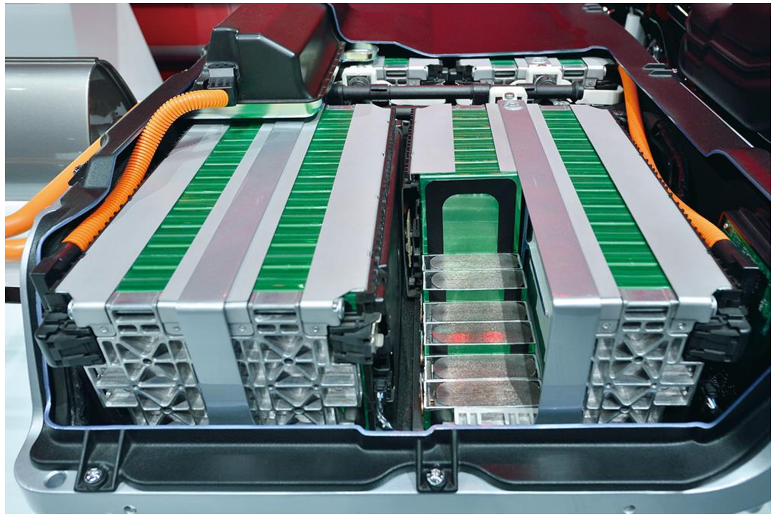
ECE 5772: High Performance Embedded Programming

Presented by: Luke Nuculaj & Michael Muller

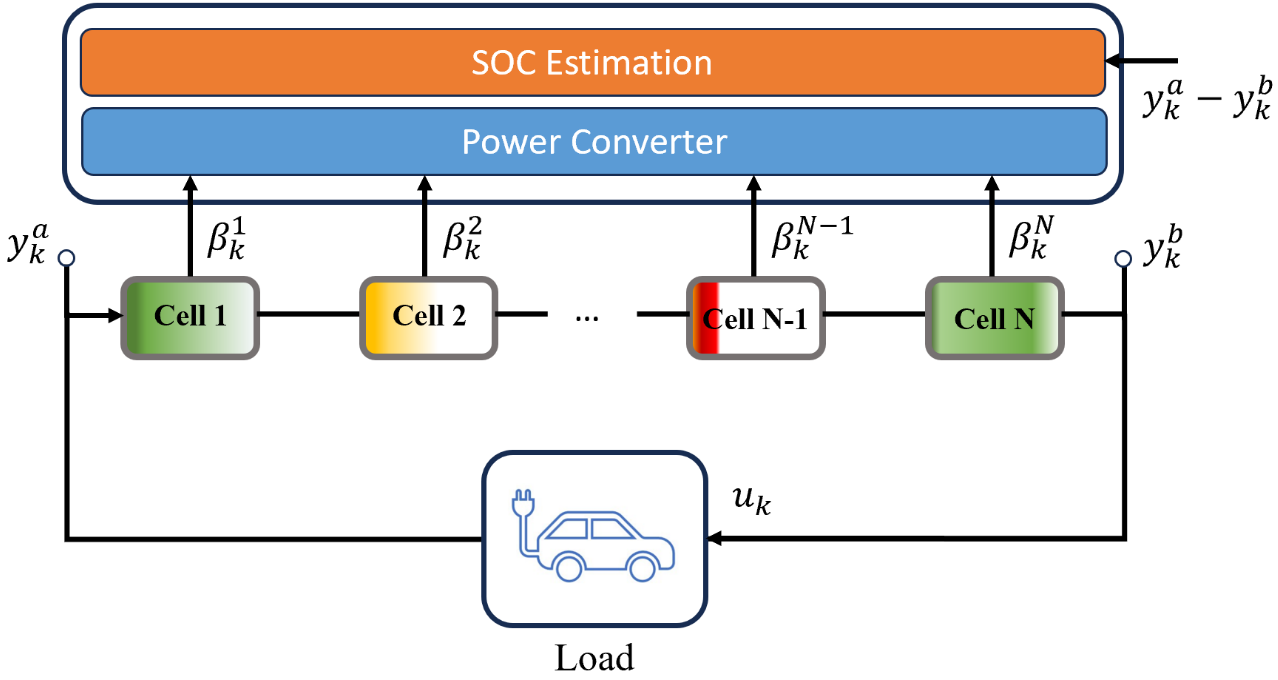
December 14th, 2023

I. Problem Formulation

- The electric vehicles (EVs) of today are powered by massive battery packs containing strings of (usually) serially-connected cells. Due to manufacturing variations, some cells discharge quicker than others. In a series-connected architecture, this means the pack performs only as well as its weakest link (dubbed "the barrel effect").



MIT News: "Designing better batteries for electric vehicles"



I. Problem Formulation

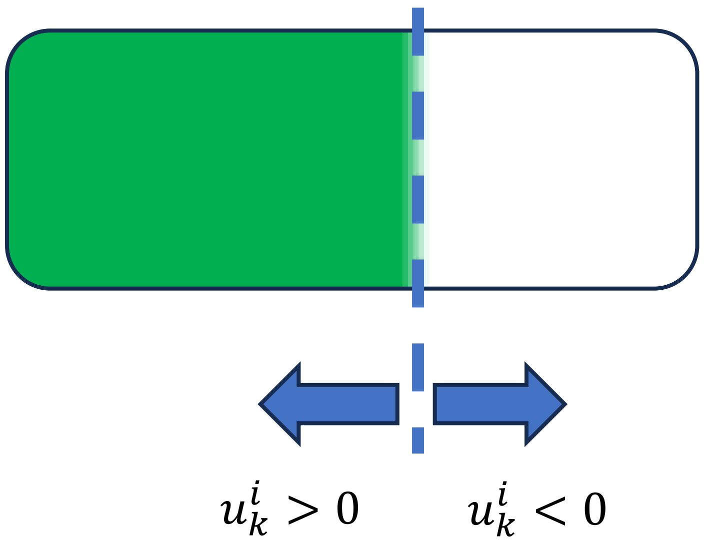
- To address concerns of the state-of-charge (SOC) imbalance between cells in a pack, active cell-balancing methods have been adopted by both literature and industry alike that move charge between different cells **with the use of balancing currents**.
- Let's consider, for example, the Coulomb counting model of a cell. Manipulating its current, we can introduce or remove charge from a given cell.

<u>Coulomb Counting Model</u>

$$s_{k+1}^i = s_k^i - \frac{\eta^i T_s}{3600 C^i} u_k^i$$

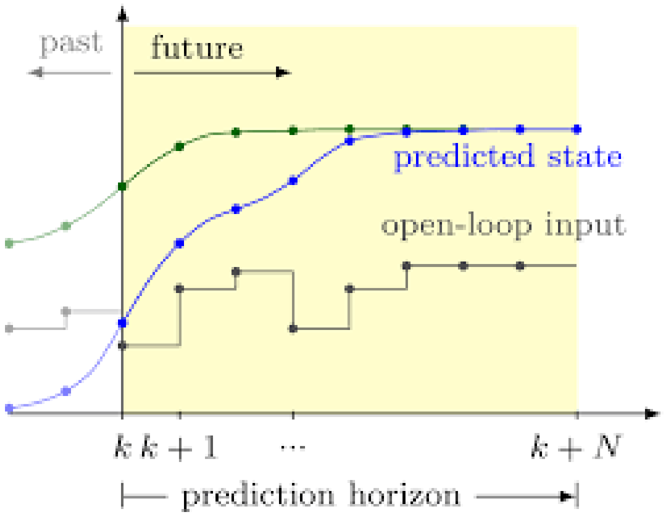
Balancing current

<u>Battery Cell</u>



I. Problem Formulation

- Applying said Coulomb counting framework to each cell, the decoupled system dynamics can be expressed in a sparse format and fed to a model-predictive control (MPC) algorithm, a receding horizon method of control which solves a QP problem at each time step.
- Computationally, this is an incredibly expensive algorithm. The approach outlined here seeks to use deep neural networks to instead learn the MPC control law. What's more, deep neural networks are easily parallelizable structures, and the techniques learned in class can be readily applied to further expedite the computation.



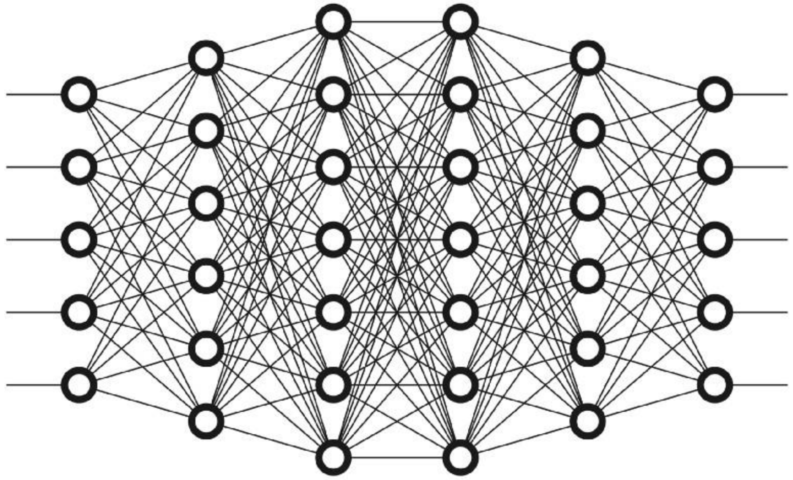
$$\min_{u_0,\dots,u_{N-1}} \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k$$

$$s.t. \quad x_{k+1} = Ax_k + Bu_k$$

$$0 \le x_k \le 1$$

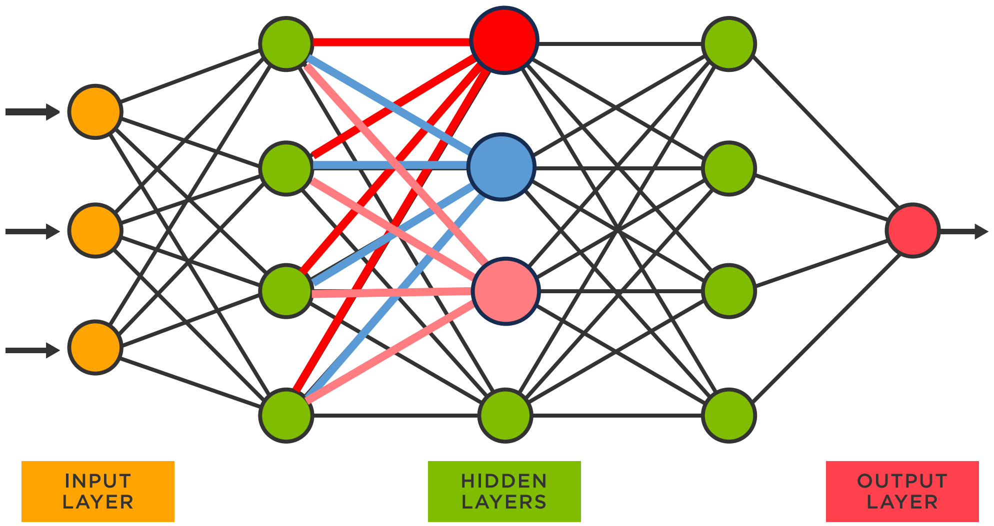
$$-0.3 \le u_k \le 0.3$$

$$\sum u_k = 0$$

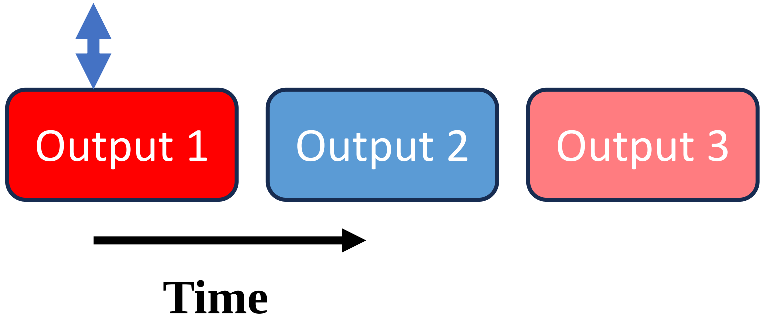


II. The Approach

- We developed neural networks for the cases of 3, 5, 10, and 20 cells. Naturally, the sizes of the neural networks grow for more cells to capture the nonlinearity of the MPC control law.
  - 3 cells: 3-48-48-48-48-3
  - 5 cells: 5-64-64-64-64-5
  - 10 cells: 10-96-128-128-96-10
  - 20 cells: 20-256-256-256-256-256-20
- The standard sequential implementation computes each neuron’s dot product with the inputs and weights going into it, passes it through the ReLU activation function, and sends it to the next layer.

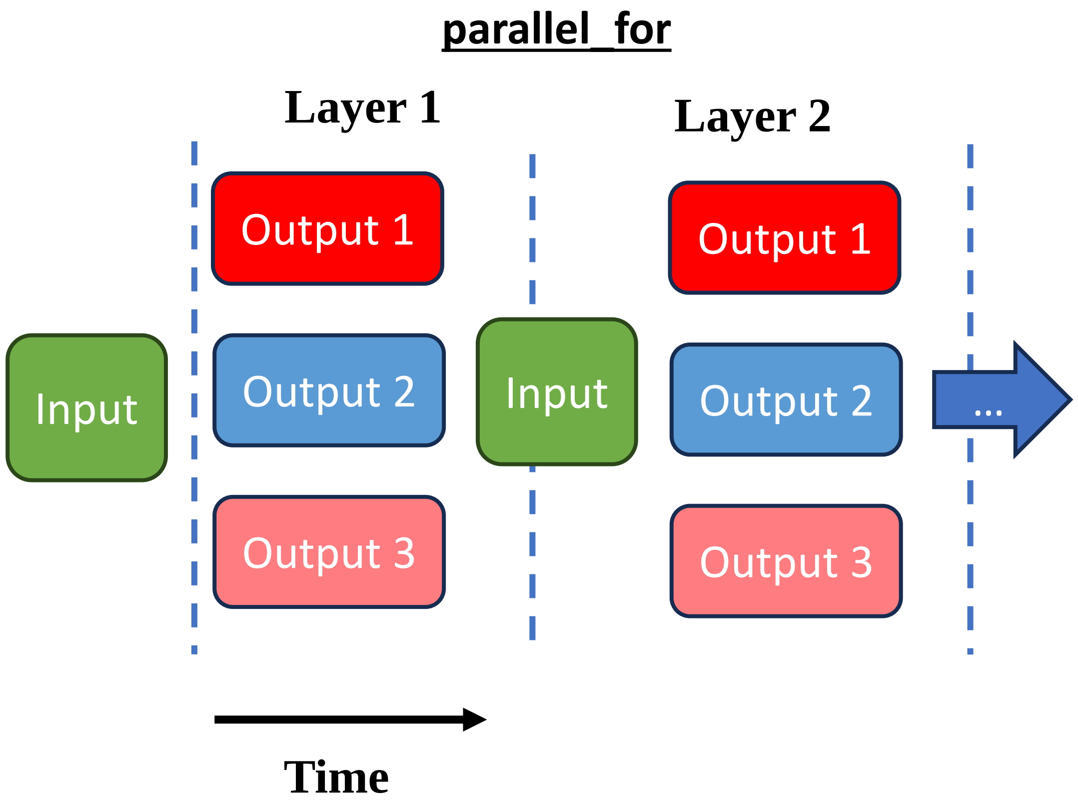


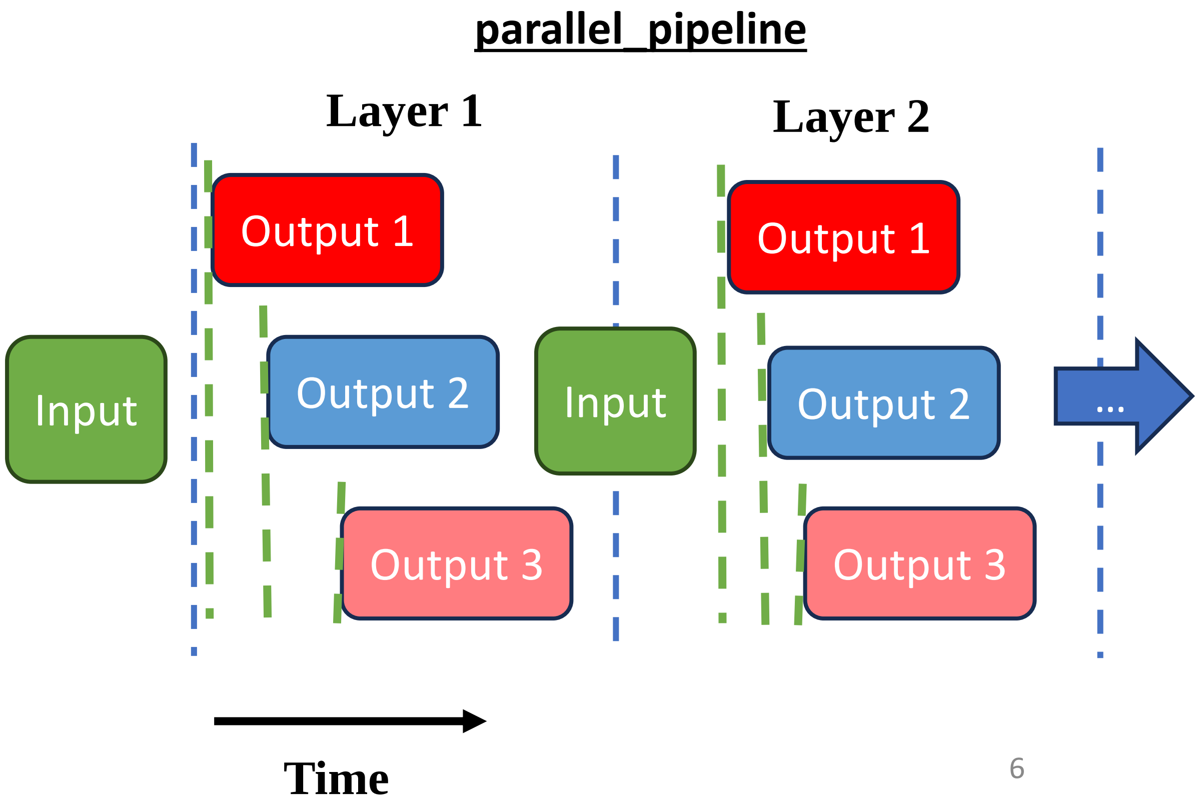
$$\mathbf{o_1} = \boldsymbol{\sigma}\left(\sum \mathbf{w_i x_i} + \mathbf{b_1}\right)$$



III. Parallelization via TBB

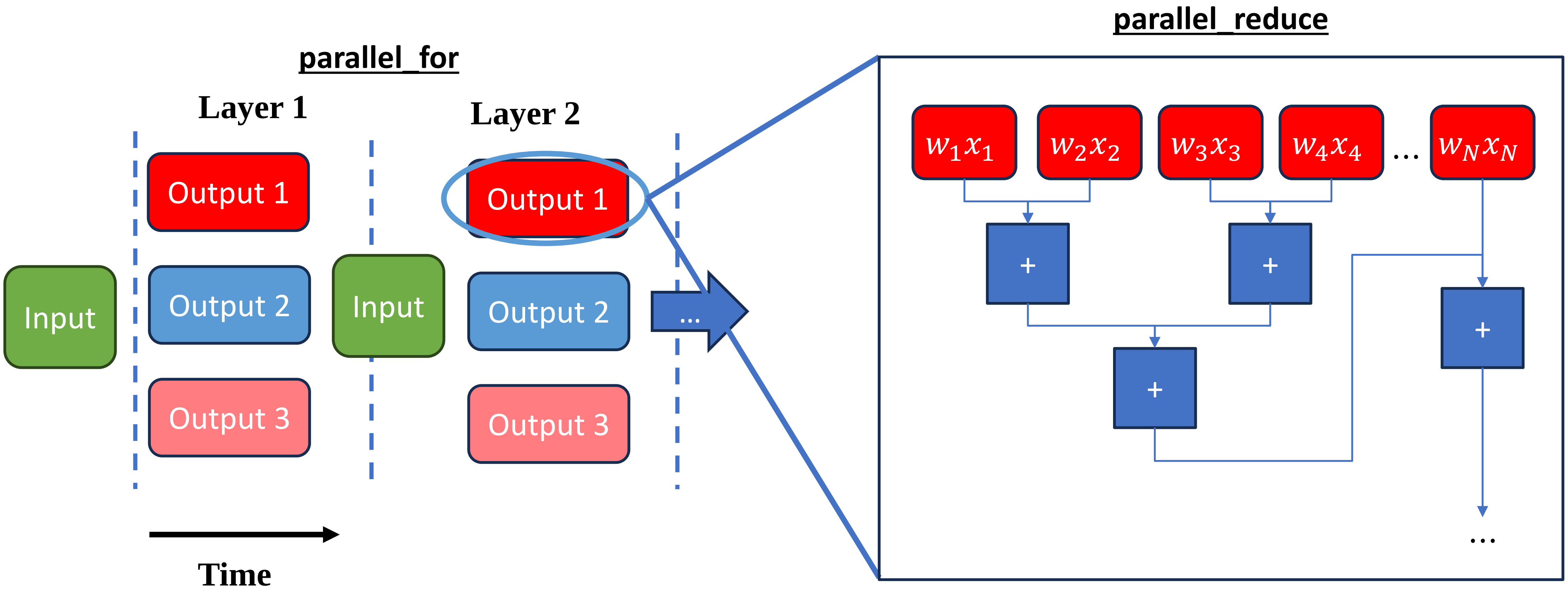
- Because these tasks can be computed independently of one another, this creates a fruitful opportunity for parallelization. In our case, we exploited three methods using the Intel TBB library:
  - parallel_for
  - parallel_pipeline
  - parallel_for + parallel_reduce





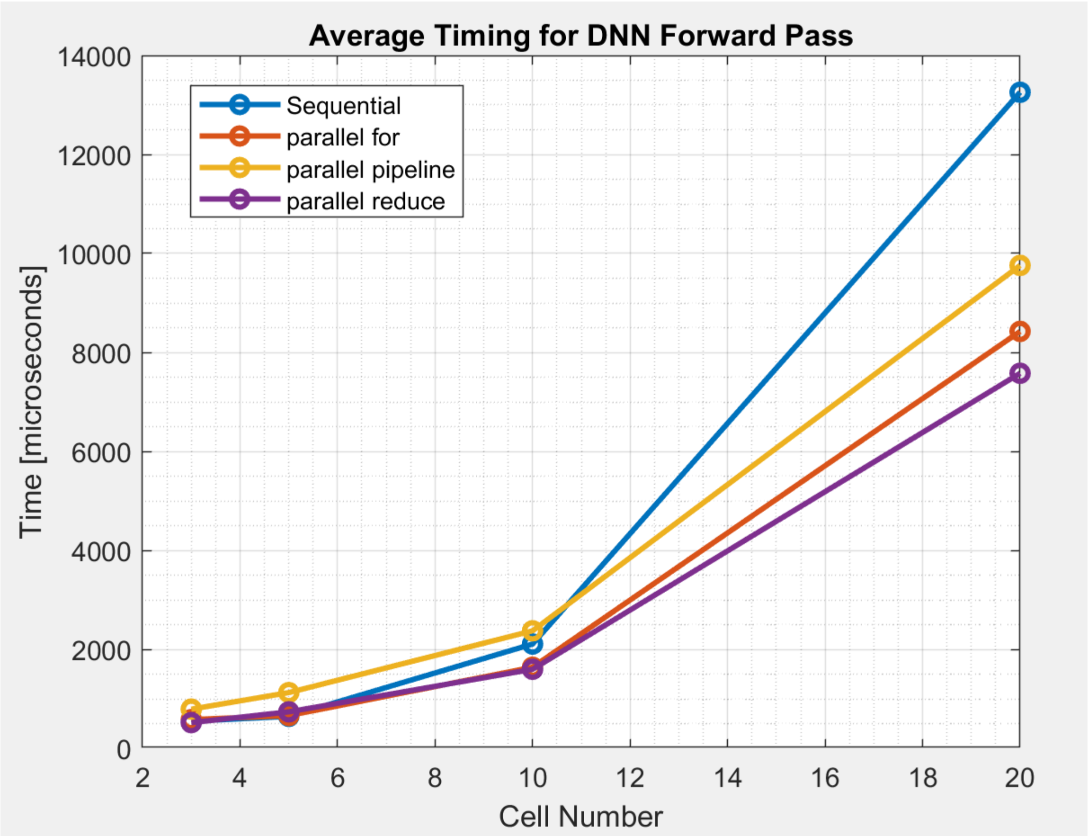
III. Parallelization via TBB

- Furthermore, the associative quality of the addition is the perfect place for implementation of the “parallel_reduce” function, which is combined with the “parallel_for” function implementation for the optimal parallelization of the deep neural network’s computation.



IV. Results

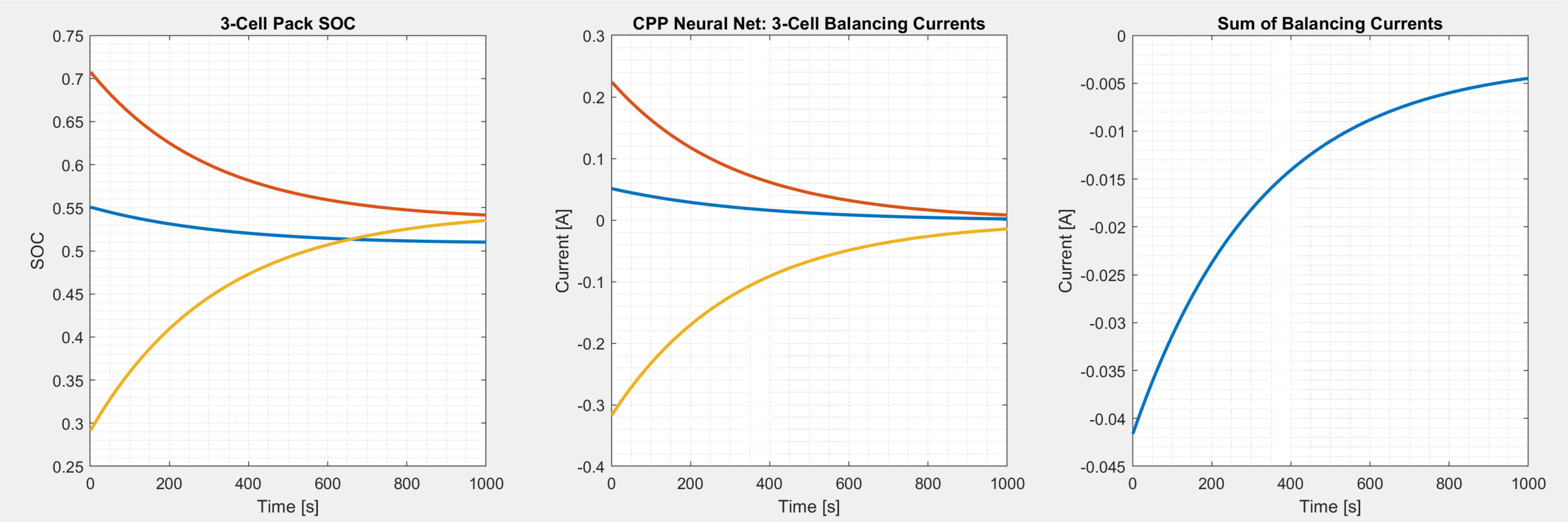
- Average timing results over 20 runs:



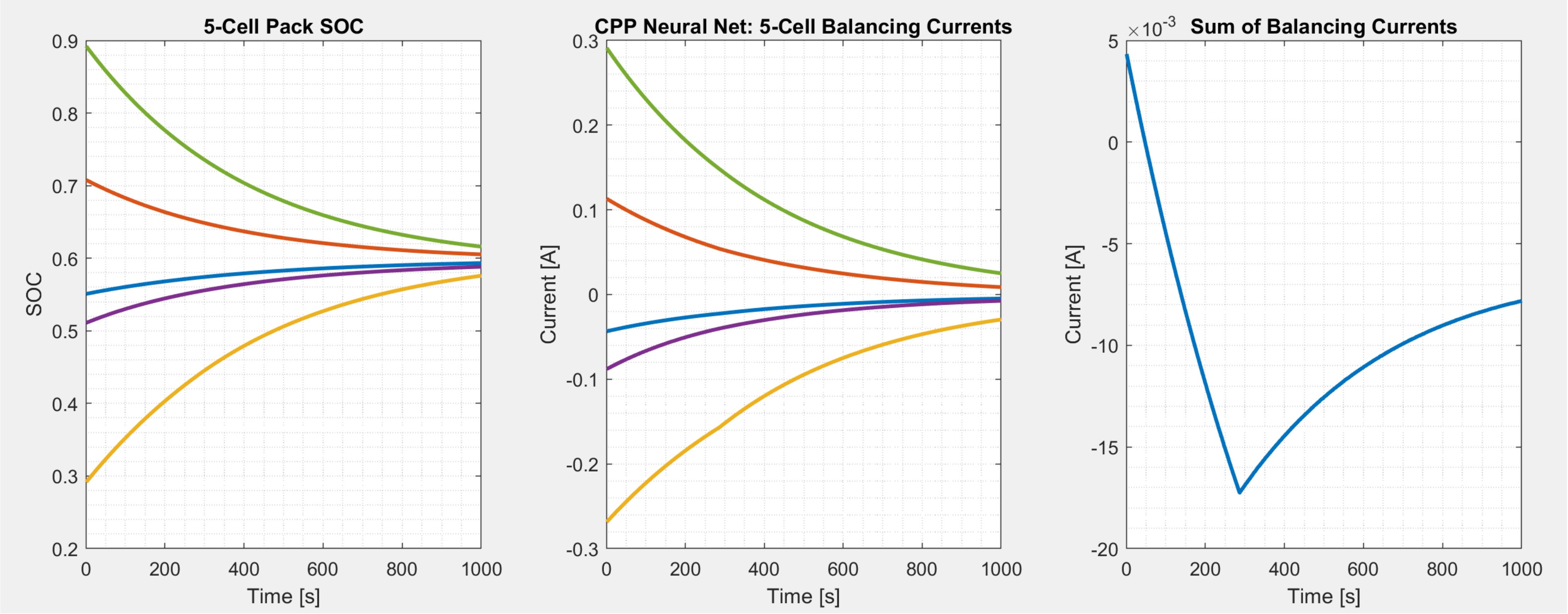
	3 cells	5 cells	10 cells	20 cells
Sequential	547.35	645.7	2107.05	13256.7
parallel_for	580.4	661.7	1638.5	8424.35
parallel_pipeline	791.65	1125.95	2374.5	9758.45
parallel_reduce	518.95	737.25	1596.95	7578.15

Timing (microseconds)

IV. Results

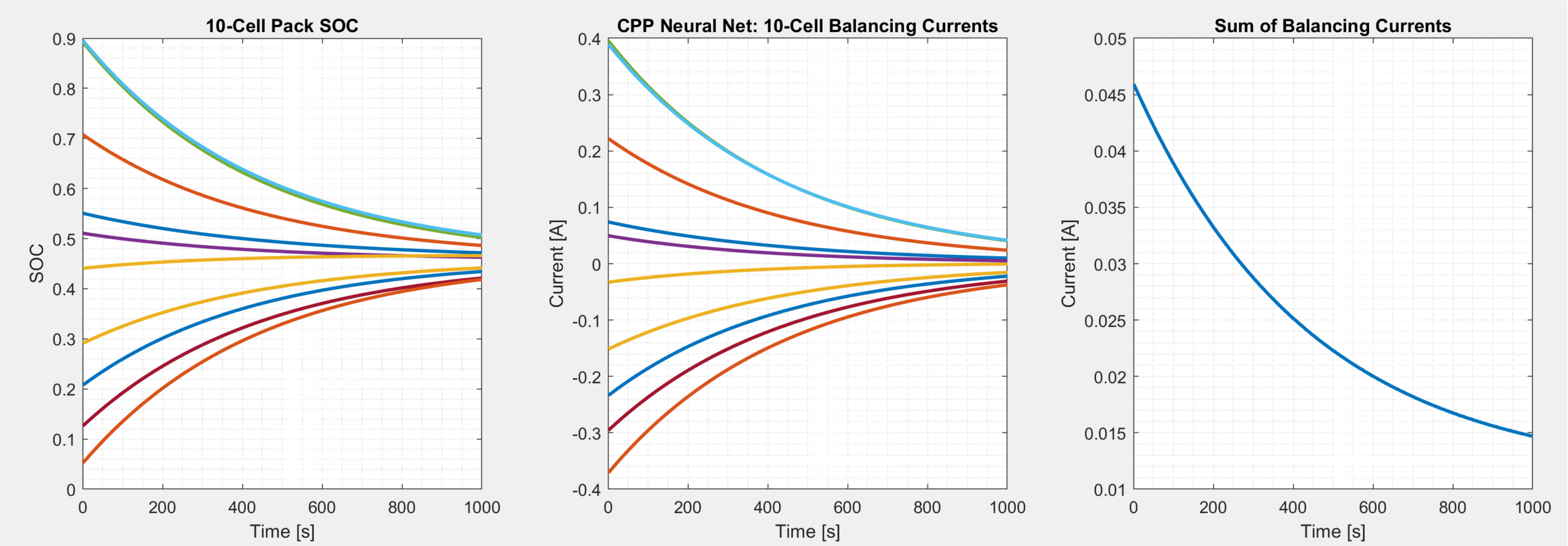

3-Cell Pack SOC
SOC
Time [s]
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0
200
400
600
800
1000
CPP Neural Net: 3-Cell Balancing Currents
Current [A]
Time [s]
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
Sum of Balancing Currents
Current [A]
Time [s]
0
-0.005
-0.01
-0.015
-0.02
-0.025
-0.03
-0.035
-0.04
-0.045

IV. Results

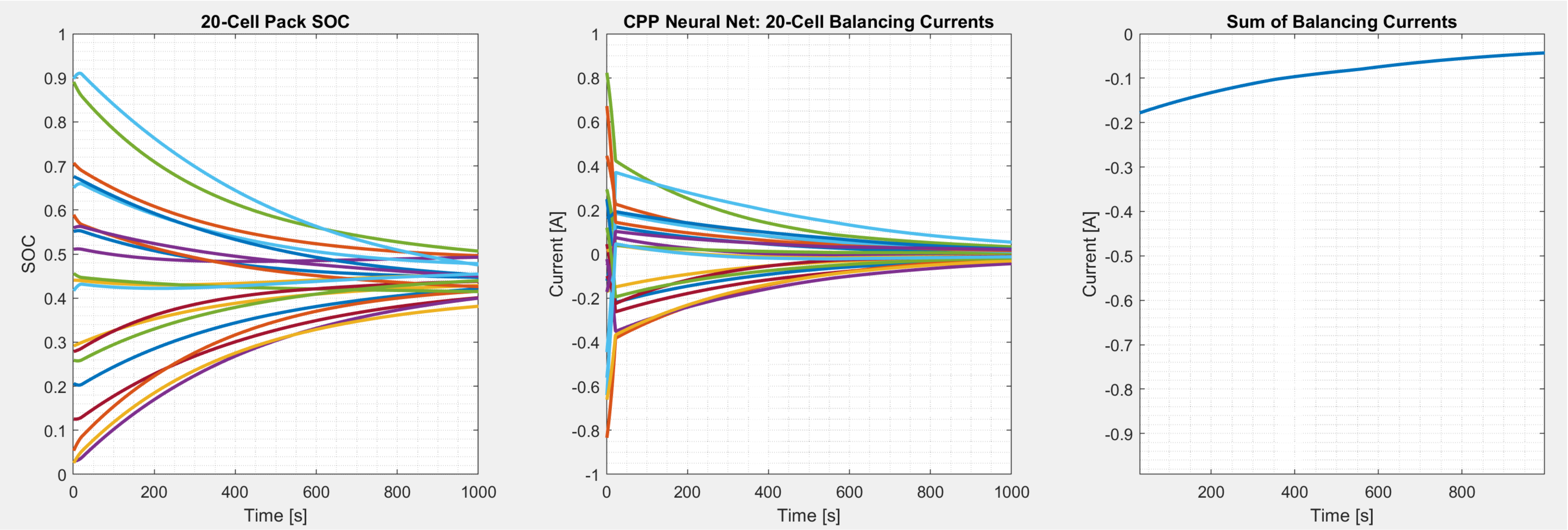


5-Cell Pack SOC
SOC
Time [s]
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
200
400
600
800
1000
CPP Neural Net: 5-Cell Balancing Currents
Current [A]
Time [s]
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
200
400
600
800
1000
Sum of Balancing Currents
×10-3
Current [A]
Time [s]
5
0
-5
-10
-15
-20
0
200
400
600
800
1000

IV. Results


10-Cell Pack SOC
SOC
Time [s]
CPP Neural Net: 10-Cell Balancing Currents
Current [A]
Time [s]
Sum of Balancing Currents
Current [A]
Time [s]

IV. Results


20-Cell Pack SOC
SOC
Time [s]
CPP Neural Net: 20-Cell Balancing Currents
Current [A]
Time [s]
Sum of Balancing Currents
Current [A]
Time [s]

V. Closing Remarks

- As we have demonstrated, deep neural networks can be deployed as a means of learning the MPC control law and expedited by way of Intel TBB’s library and core concepts learned in class.
- Originally, we set out to do an actor-critic reinforcement learning approach but crafting a reward structure proved exceptionally difficult. With the focus of the project being on the hardware, we instead shifted the project idea to training a DNN to learn the MPC control law, which we were able to do with great success.
- With further training, the DNN’s could have no doubt learned the MPC control law to a greater degree and stayed more tightly within the system constraints.
- **Questions?**