4CBLA30 Energy Storage and Transport

Self-Study Assignment

Group 46

SSA No.	Description
4	Simulink Model
SSA Owner	
Pranav Joshi	

Introduction

From the previous meeting, the Simulink Model needs to be further sharpened, especially the battery storage vs. time graph. Following that, the model needs to incorporate the system parameters and equations in different levels of complexity to obtain the desired results.

Goal

- To fix the storage vs. time graph in the model
- Work towards the first stage of incorporating the system parameters after understanding the modeling objective.

Collaboration

- Pranav: Upload Simulink Model files on GitLab for ease of collaborative access and fix the Simulink model to function as predicted by the graphs of Koen and Simon in SSA 1
- Ruben: Incorporate the system parameters as defined by Thomas in SSA 4 in a basic level of complexity, and work towards higher levels of complexity if time permits.

Problems

Valuable time was lost putting the Simulink files on GitLab, which proved to be a bit counterproductive to the collaboration. This reduced the time Ruben had to work on his part of the model and in future this needs to be avoided from my side.

Conclusion

- The Simulink model has been fixed and functions properly (as predicted by Simon in SSA 1)
- The initial system parameters have been integrated by Ruben

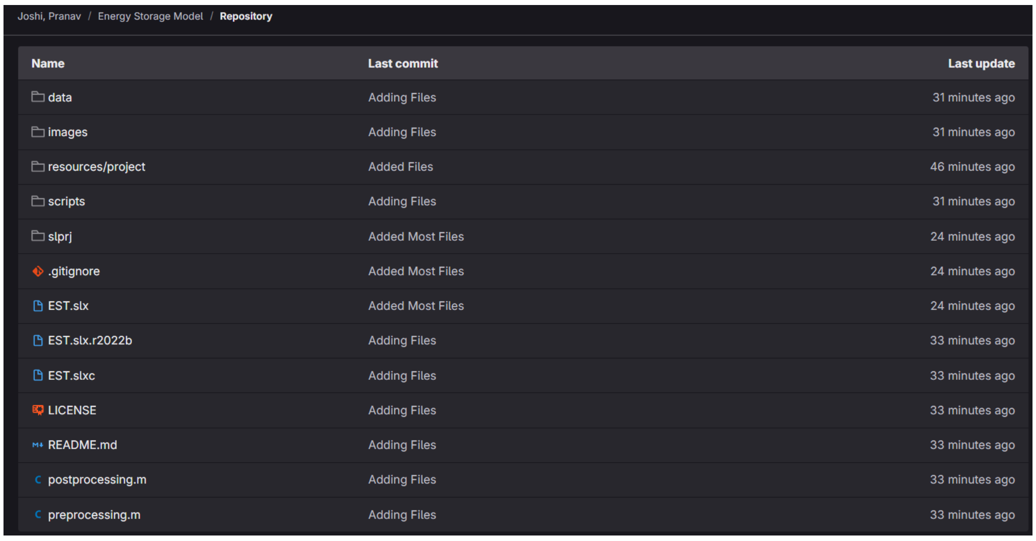
Recommendations

- The system parameters unique to the PHS system need to be integrated in the chronological order of complexity

1 Elaboration

1.1 Uploading git files on GitLab

Initially the files that were edited from the provided Simulink model were sent over WhatsApp to Ruben. Further, all the files were uploaded to a gitlab repository. This allows any group member to access and edit files with more convenience.



Joshi, Pranav / Energy Storage Model / Repository

Name	Last commit	Last update
data	Adding Files	31 minutes ago
images	Adding Files	31 minutes ago
resources/project	Added Files	46 minutes ago
scripts	Adding Files	31 minutes ago
slprj	Added Most Files	24 minutes ago
.gitignore	Added Most Files	24 minutes ago
EST.slx	Added Most Files	24 minutes ago
EST.slx.r2022b	Adding Files	33 minutes ago
EST.slxc	Adding Files	33 minutes ago
LICENSE	Adding Files	33 minutes ago
README.md	Adding Files	33 minutes ago
postprocessing.m	Adding Files	33 minutes ago
preprocessing.m	Adding Files	33 minutes ago

Figure 1: Gitlab repository

1.2 Investigating Storage Graph

The storage graph seems to disagree with the pie charts, it shows that the battery does not charge during the supply surplus (as shown in fig. 2). It reads a loss of nearly 0.12*10^{10} Joules of energy.

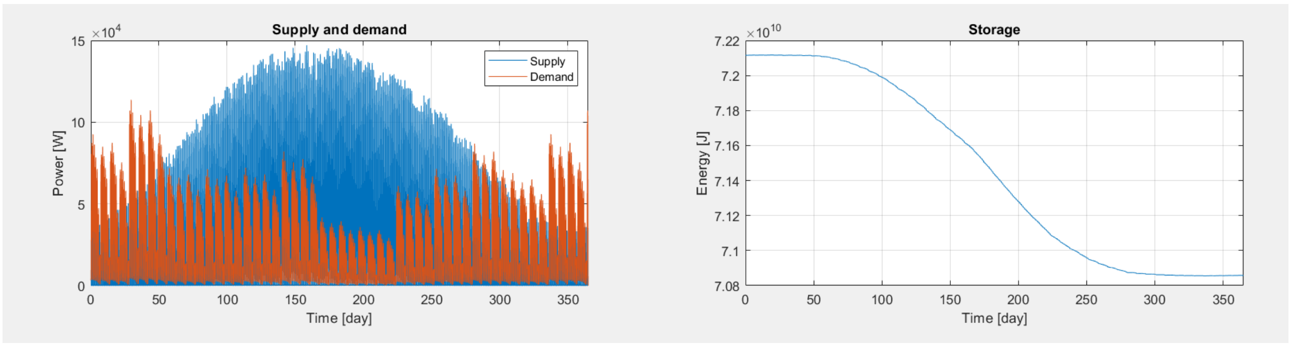


Figure 2: Storage Graph

The loss coefficient (bstorage) that defines energy losses due to storing energy has been set to 0 in the preprocessing file. Logically, no energy should be drawn from the battery since the supply is around 47 kW and demands is only 21 kW on average. This suggests that the battery has no reason to lose any power.

This reveals a potential possibility for this strange graph. The logic of power injection could be flawed. Instead of supplying power directly to demand in a surplus, perhaps all the power (47 kW) is being redirected to the battery and then the demand energy (21 kW) is being drawn. This logical error is searched for in the storage and control block of the model (as shown in fig. 3)

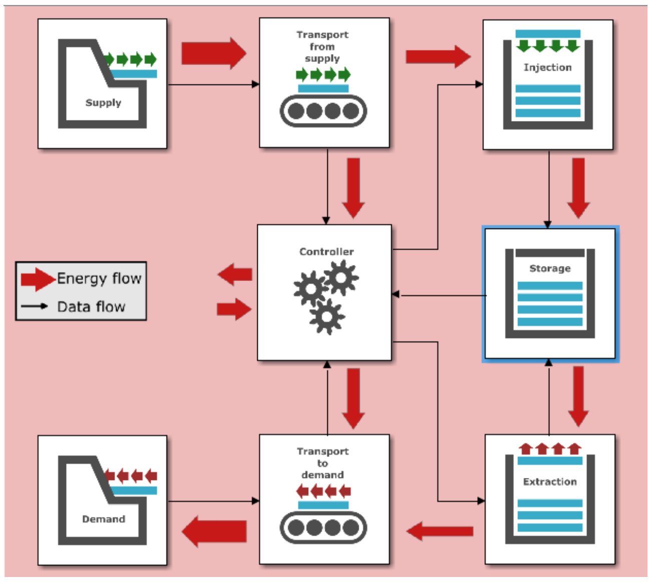


Figure 3: Complete EST Model

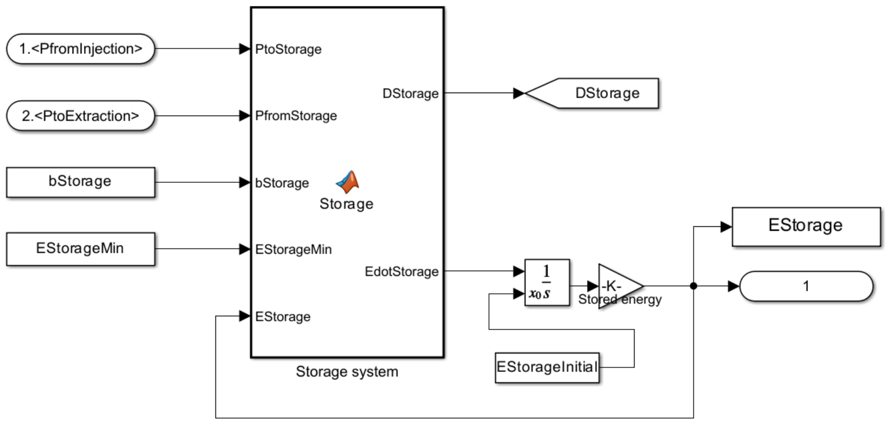


Figure 4: Storage Block

Looking at the storage block (as shown in fig. 4, the storage function is defined as following

```
function [DStorage, EdotStorage]= Storage(PtoStorage, PfromStorage, bStorage, EStorageMin, ...
    EStorage)
    DStorage = bStorage * (EStorage - EStorageMin);
    % EStorageMin is always 0 and bStorage is also 0
    EdotStorage = PtoStorage - PfromStorage - DStorage;
```

So far, no indications in the code that agree with the error hypothesis.

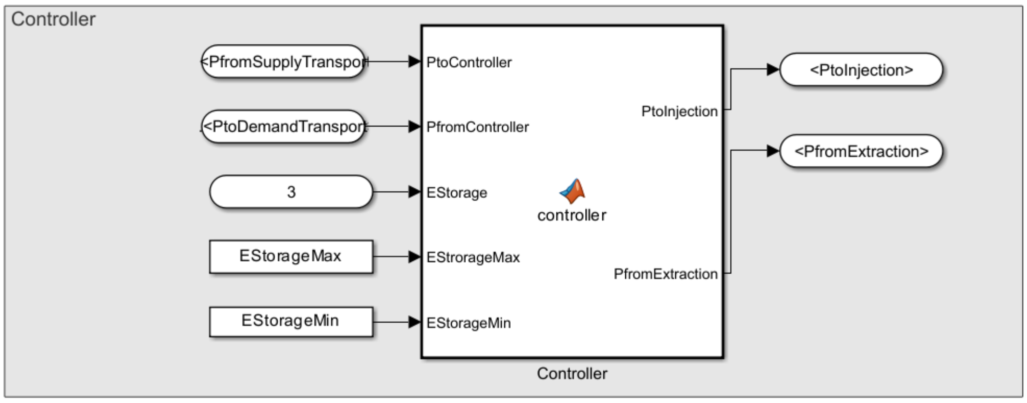


Figure 5: Controller

Looking into the control block (as shown in fig. 5), the controller function is defined as following;

```matlab
function [PtoInjection, PfromExtraction] = controller(PtoController, PfromController, ...
    EStorage, EStrorageMax, EStorageMin)
    %% Power surplus or deficit
    PtoInjection     = 0;
    PfromExtraction  = 0;

    %Supply - Demand
    ΔP = PtoController - PfromController;

    % Power surplus
    if ΔP ≥ 0
        disp("Injection initiation")
        %% Storage injection
        % Storage system is not full
        disp(EStrorageMax)
        disp(EStorage)
        if EStorage < EStrorageMax
            disp("Injection")
            PtoInjection=ΔP;
        end
    % Power deficit
    else
        %% Storage extraction

        % Storage system is not empty
        if EStorage > EStorageMin
            PfromExtraction = -ΔP;
        else
            PfromExtraction=0;
        end
    end
```

Both functions seem logically sound. The signal connections are appropriate as well. The EStorage data is being properly communicated to the Controller (Node 1 in Storage Block and Node 3 in the Controller Block). The controller block is getting the supply power (PtoController) and demand power (PfromController) properly as well upon double checking. The problem doesn't lie here.

The best way to debug a seemingly invisible logical flaw is by meticulously tracking the signal and energy flow. A variety of print/display statements are setup in the controller, injection and storage block to really understand what is going on.

Controller Block:

```matlab
function [PtoInjection, PfromExtraction] = controller(PtoController, PfromController, ...
    EStorage, EStrorageMax, EStorageMin)
    %% Power surplus or deficit
    PtoInjection     = 0;
    PfromExtraction  = 0;

    %Supply - Demand
    ΔP = PtoController - PfromController;

    % Power surplus
    if ΔP ≥ 0
        %% Storage injection
        disp("Injection initiated")
        % Storage system is not full
        if EStorage < EStrorageMax
            PtoInjection=ΔP;
        end
    % Power deficit
    else
        %% Storage extraction

        % Storage system is not empty
        if EStorage > EStorageMin
            PfromExtraction = -ΔP;
        else
            PfromExtraction=0;
        end
    end
```

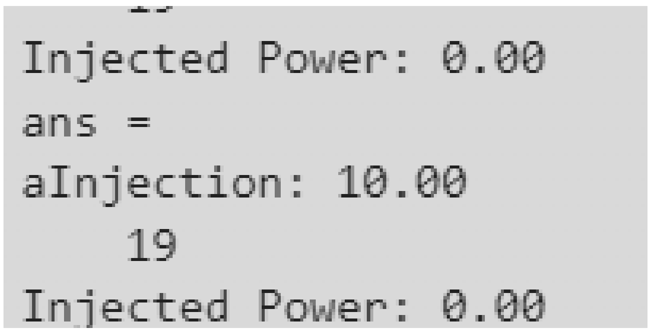
Injection Block:

```
function [PfromInjection, DInjection] = injection(PtoInjection, aInjection)
    DInjection = aInjection * PtoInjection;
    fprintf('aInjection: %.2f \n', aInjection)
    PfromInjection = PtoInjection - DInjection;
    fprintf('Injected Power: %.2f \n', PfromInjection);
```

Storage Block:

```
function [DStorage, EdotStorage]= Storage(PtoStorage, PfromStorage, bStorage, EStorageMin, ...
    EStorage)
    DStorage = bStorage * (EStorage - EStorageMin);
    % EStorageMin is always 0 and bStorage is really small
    EdotStorage = PfromStorage - PtoStorage - DStorage;
    if PfromStorage>PtoStorage
        fprintf("Injection was succesful, Injected Power: %2f \n",EdotStorage)
    end
```

The diagnostics report of the model (console debugging as shown in fig. 6) revealed the hole in logic, aInjection wasn't treated as a percentage as described in the comments of the preprocessing file made by the EST team.



```
Injected Power: 0.00
ans =
aInjection: 10.00
     19
Injected Power: 0.00
```

Figure 6: Diagnostics showing aInjection = 10.00

To fix this, in the preprocessing file the aInjection and aExtraction was changed from integers (10 and 5) to fractions (10/100 and 5/100). Currently these values are not in agreement with the actual system parameters, these values have been used to test the proper functioning of the system.

```
% injection system
aInjection = 10/100; % Dissipation coefficient

% extraction system
```

```
aExtraction = 5/100; % Dissipation coefficient
```

In retrospect, perhaps this was a personal misunderstanding. To write the comment "Dissipation coeffecient" the percentage sign needs to be used. This was misinterpreted to be "percentage value of dissipation coefficient"

As expected, the pie charts (as shown in fig. 7) and graphs (as shown in fig. 8) are in complete agreement with Simon's graphs from SSA 1 (as shown in fig. 9)

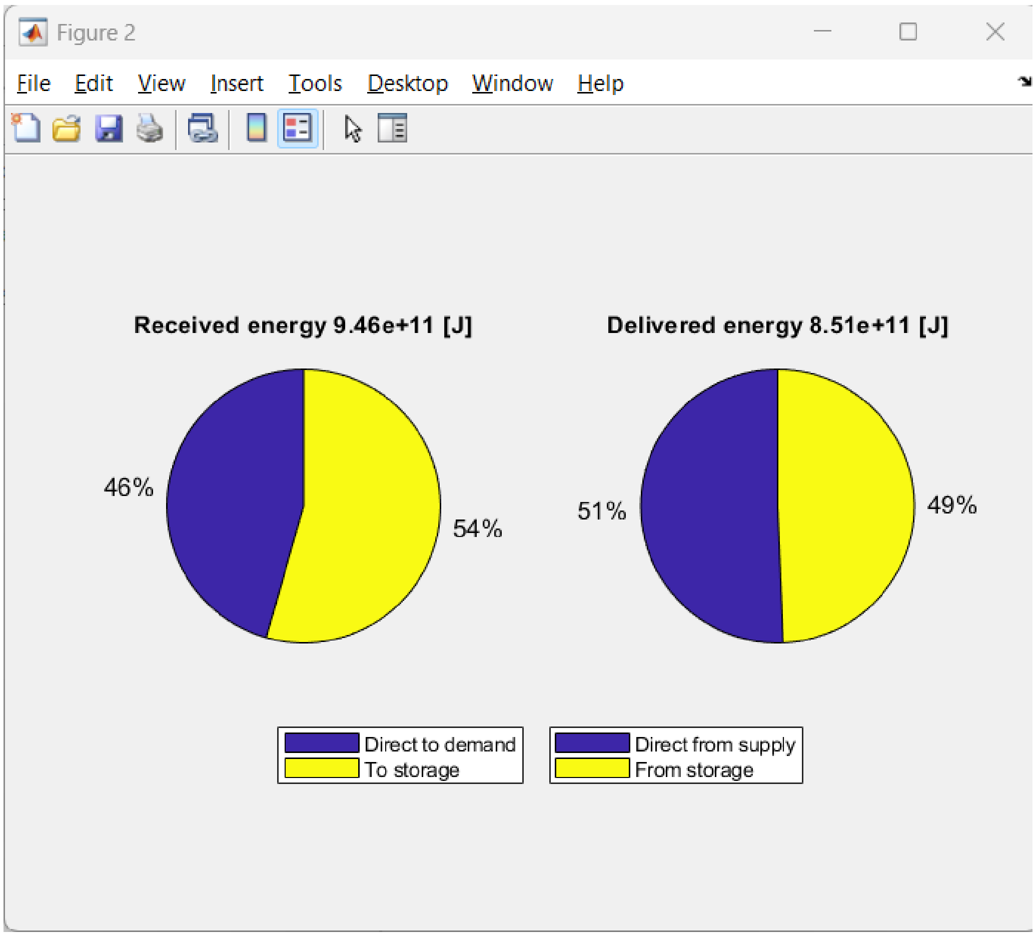


Figure 2
File Edit View Insert Tools Desktop Window Help
Received energy 9.46e+11 [J]
46%
54%
Delivered energy 8.51e+11 [J]
51%
49%
Direct to demand
To storage
Direct from supply
From storage

Figure 7: Final Pie Charts

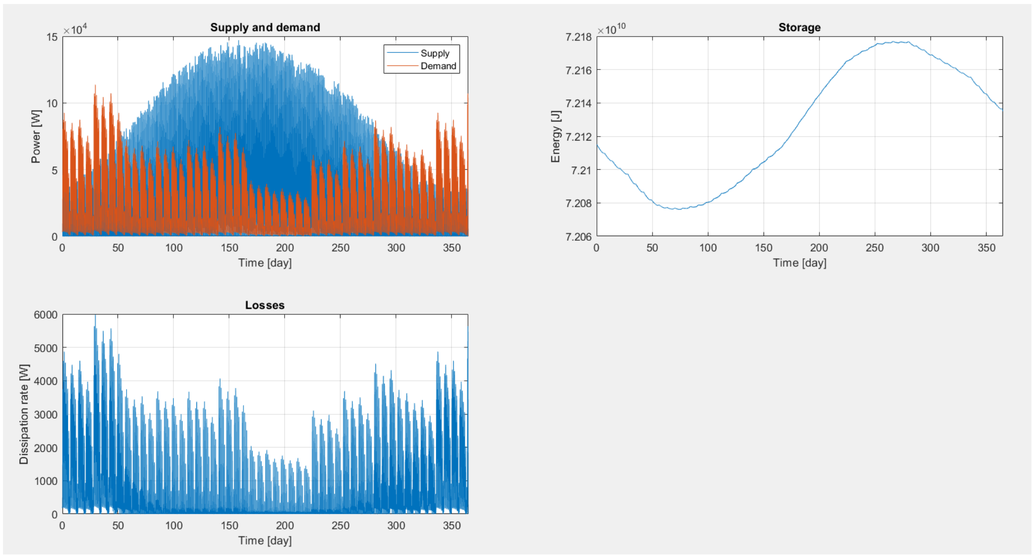


Figure 8: Fixed Graphs from Simulink Model

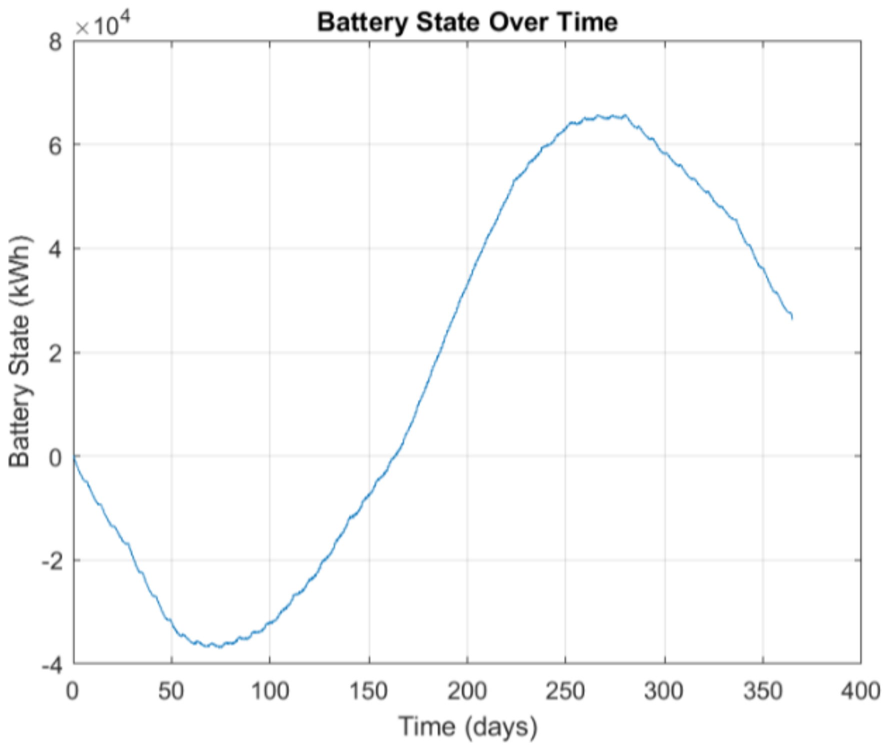


Figure 9: Simon's Graphs from SSA 1

Further, the detailing of the model ie; adding the system parameters that are specific to the chosen PHS system and so on is being worked on by Ruben. Hence, the graphs obtained currently have not been analyzed further, given their significance is limited only to their basic function, ie; their shape. This has been proven to be accurate as explained above.

Overleaf Link to this SSA

`https://www.overleaf.com/read/vbbwhynfkhmt#08ac9d`