サンプルサロゲートモデル
解説書

作成日時：2023 年 11 月 28 日

作成者：JMAAB PMWG

更新履歴

NO	内容	日付	変更者
1	初版発行	2023/11/28	PMWG
2			
3			
4			

目次

1. はじめに

1.1 背景・目的

MATLAB Deep Learning Toolbox では深層学習を実現するブロックが用意されている。しかし、このブロックを用いたサロゲートモデル作成の事例はまだ少ない。そこで今回 PWMG では、プラントモデルのサロゲート化プロセスを体験可能なサンプルモデルを用意した。

1.2 内容

MATLAB Deep Learning Toolbox を用いたプラントモデルのサロゲートモデル化プロセスの具体例の提示と、そのサロゲートモデルの説明。

1.3 対象ユーザー

・MATLAB を用いてプラントモデルを作成した経験があるモデル開発者
・深層学習を用いたモデル作成の初学者

1.4 記載範囲

以下の内容について、一例を記載する。
・Deep Learning Toolbox の Stateful Predict ブロックを用いてプラントモデルの一部をサロゲートモデル化するための手順

1.5 バージョン＆ツールボックス

・MATLAB R2022b
- MATLAB
- Simulink
- Simscape
- Deep Learning Toolbox

1.6 作業フォルダ

【注意】
・サンプルモデルを含むフォルダのパスに、全角文字を含まないこと。

1.7 参照ドキュメント

[1] Simscape による プラントモデリング入門　JMAAB プラントモデルワークショップ作成 （2017 年 6 月）https://jmaab.jp/

1.8 定義（用語、略語）

用語・略語	正式表記	意味
サロゲート	Surrogate	代理モデル。数値シミュレーションの代わりにニューラルネットワークなどの機械学習を活用して現象を計算・予測する手法
LSTM	Long Short Term Memory	長・短期記憶ネットワーク。 RNN（再帰型 ニューラル ネットワーク） の一種
SOC	State of Charge	充電率
OCV	Open Circuit Voltage	開回路電圧
推論	—	学習済みの「推論モデル」に新しいデータを入力し計算結果を得ること
RMSE	Root Mean Square Error	2 乗平均平方根誤差 $\hat{y}_i$:計算値(i=1,…,n) y_i:測定値(i=1,…,n) n:データ数 $RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$

2. サロゲートモデルサンプル

2.1 概要

2017 年に JMAAB にて公開した HEV（Hybrid Electric Vehicle）モデルを題材に、Deep Learning Toolbox の Stateful Predict ブロックを使用して、Battery モデルのサロゲートモデル化を試みた。本章では、サロゲートモデル化の題材とした HEV モデルおよびサロゲートモデル構築のために作成したサンプルモデルを示す。

2.2 HEV モデル構成

HEV モデルの全体構成と、今回サロゲートモデル化対象とした Battery モデルを示す。

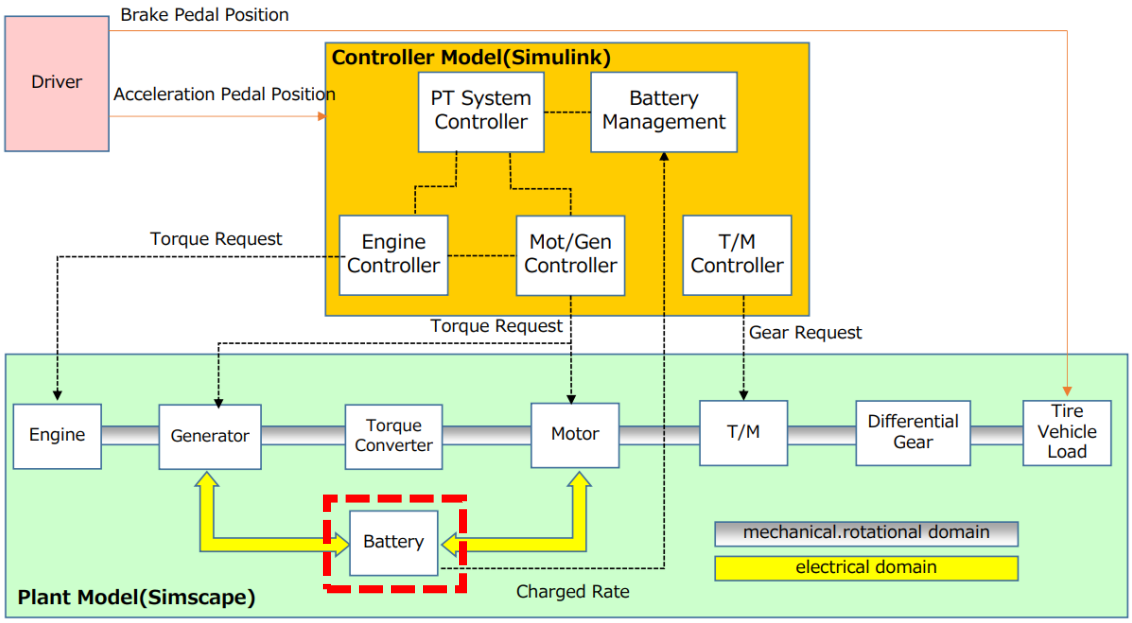


Fig. 2-1 HEV モデルの全体構成図。赤破線がサロゲートモデル化対象の Battery モデル

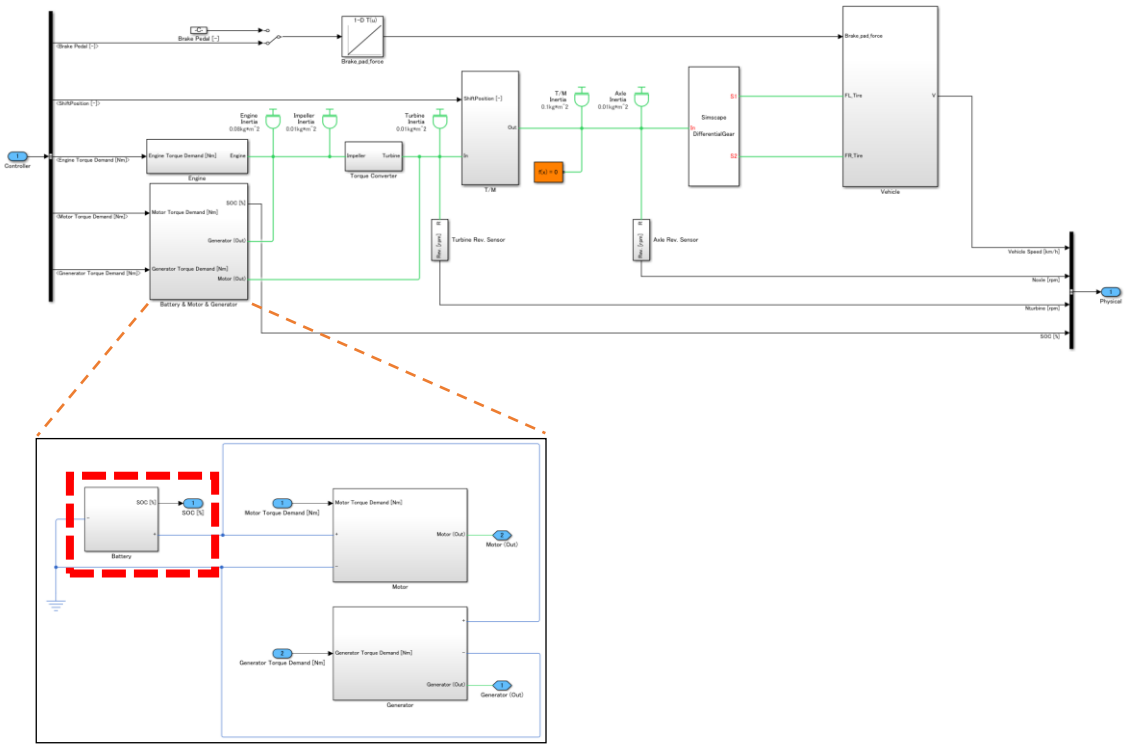


Fig. 2-2 Simscape で作成された HEV モデルと Battery モデル

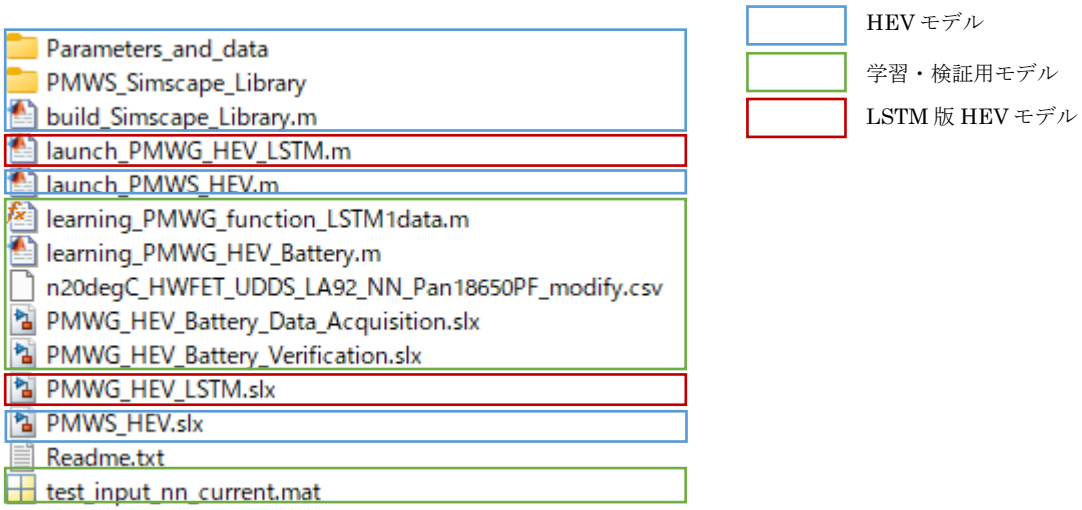
2.3 サンプルモデル構成

Battery モデルをサロゲートモデル化した HEV モデル（以降、LSTM 版 HEV モデル）を構築するために、以下の 3 つのモデルを作成した。

1. 学習用モデル
2. 検証用モデル
3. LSTM 版 HEV モデル

サンプルモデルのフォルダ構成を示す。サンプルモデルフォルダには、学習・検証用モデル、LSTM 版 HEV モデル、およびスクリプトが含まれている。また、従来の HEV モデルは R2016a 版であったため、今回 R2022b 版へ更新[1]し、サンプルモデルのフォルダに同梱した。

[1] MATLAB バージョンアップ（R2016a ⇒ R2022b）に伴う変更に対応し、ソルバを可変ステップから固定ステップへ修正した。



なお上記サンプルモデルはステップサイズを 1ms としているが、Battery サブシステムを LSTM ブロックで置き換えたモデルは、処理負荷の影響で推定の実行に実時間以上の時間が必要となっている。

以下、順に説明する。

2.3.1 学習用モデル

Battery モデルの入出力関係を学習するためのデータ取得を目的に、Battery モデル単体のモデルを用意した。

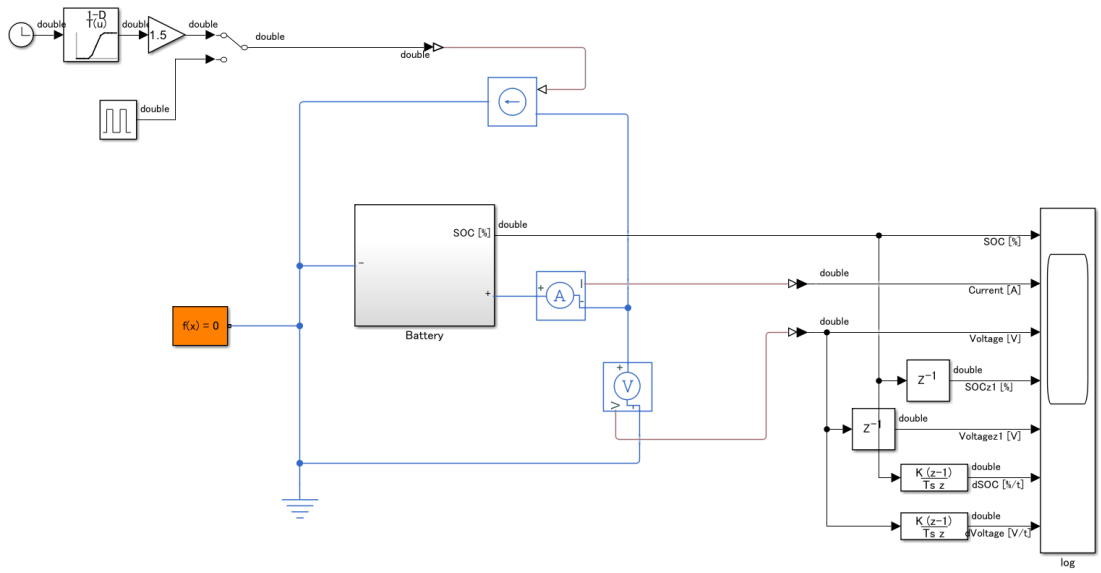


Fig. 2-3　学習データ取得用モデル（PMWG_HEV_Battery_Data_Acquisition.slx）

電流のソースとして、Kollmeyer, Phillip (2018) [2]の”Panasonic 18650PF Li-ion Battery Data” を採用した。このデータは https://data.mendeley.com/datasets/wykht8y7tg/1 で公開されている。なお、Battery モデルへの入力は Controlled Current Source を用い、理想的な電流が流れると仮定した。

Battery モデルの出力は、SOC、Current、Voltage、1 step 前の SOC、1 step 前の Voltage、SOC の微分値、電圧の微分値の 7 種類とした。これら出力変数は、Battery モデルの物理式（入力が電流、出力が電圧と SOC であり、SOC は SOC 前回値に電流積分で得た SOC 増減分を加算して計算）に基づいて決定した。なお、ソルバの設定は HEV モデルと同一としている。

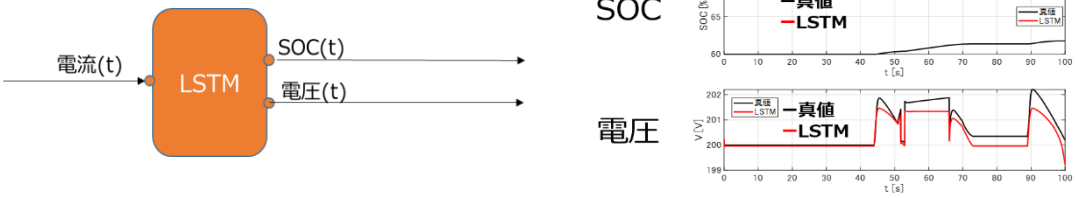
実際の学習は m スクリプトで実施しており（3.1 節参照）、学習に使用したデータは電流、1step 前の SOC、SOC の微分値、電圧の 4 データとした。

[2] Kollmeyer, Phillip (2018), “Panasonic 18650PF Li-ion Battery Data”, Mendeley Data, V1, doi: 10.17632/wykht8y7tg.1

2.3.1.1 サロゲートモデル入出力信号の選定過程

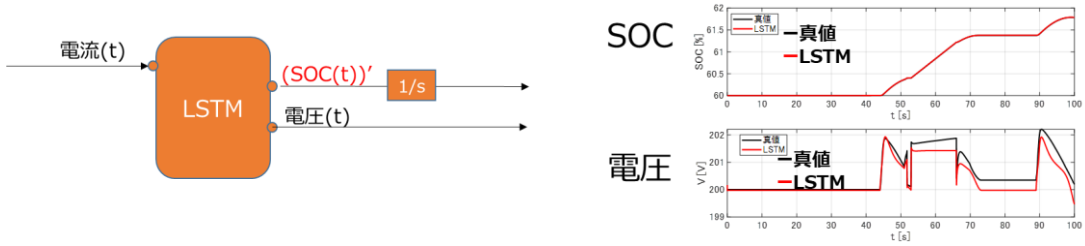
Battery モデルをサロゲートモデル化するために、入力を電流と 1step 前の SOC、出力を SOC の微分値と電圧とした。これらの信号を決定した過程を述べる。

まず初めに Battery モデルの物理式をもとに、入力を電流、出力を SOC と電圧としてサロゲートモデルを作成した。結果を以下に示す。この結果から、SOC は定性的にも定量的にも振る舞いを推定することができていないが、電圧は定性的な振る舞いはある程度推定することはできる、という結果が得られた。

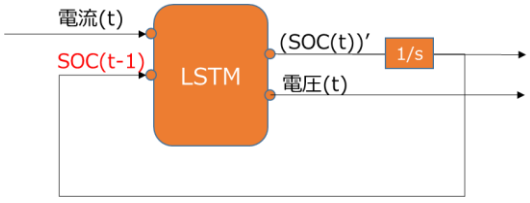


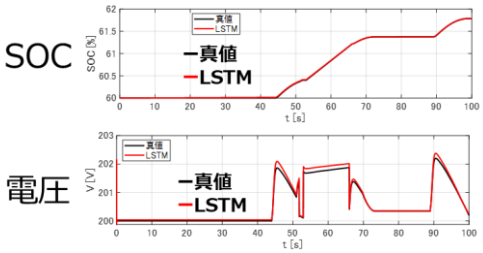
上記結果を踏まえ、学習データの入力と出力の相関関係を調べたところ、電流と SOC は相関がなかったが、電流と SOC の微分値には相関があることが分かった。

そこで次に、入力を電流、出力を SOC の微分値と電圧として、サロゲートモデルを作成した。結果を以下に示す。この結果から、SOC は定性的にも定量的にも振る舞いをよく推定できることができ、電圧は定性的な振る舞いをある程度推定することができるという結果が得られた。



上記結果から、電圧の推定を改善する方法を考えた。電圧は OCV と電流による電圧の変化で決まり、さらに OCV は SOC で決まるので、電圧の推定には電流と SOC が必要となる。そこで、SOC を入力に追加することを考えた。結果を以下に示す。この結果から、入力を電流と SOC の前回値、出力を SOC の微分値と電圧とすると SOC と電圧を推定可能なサロゲートモデルが作成可能であると判断した。


電流(t)
SOC(t-1)
LSTM
(SOC(t))'
1/s
電圧(t)


SOC
真値
LSTM
SOC [%]
t [s]
電圧
V [V]

2.3.2 検証用モデル

オリジナルの Battery モデルと、サロゲートモデル化した LSTM 版 Battery モデルの性能比較を目的に、検証用モデルを用意した。

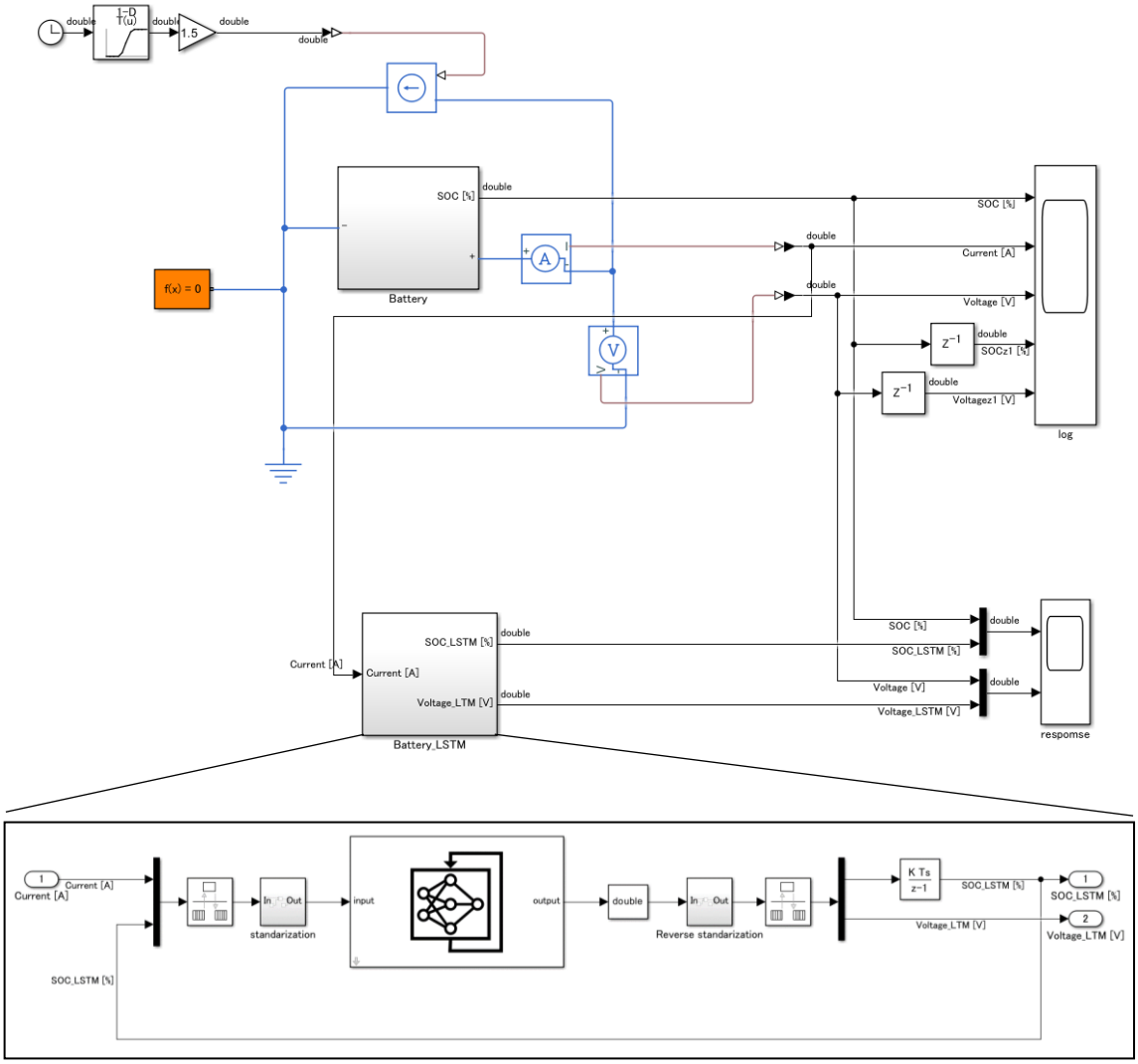


Fig. 2-4　検証用モデル（PMWG_HEV_Battery_Verification.slx）

上側のブロックは学習用に作成したモデルと同一である。

下側のブロックは、今回作成した LSTM 版 Battery モデルである。

LSTM 版 Battery モデルの入力は電流と 1step 前の SOC であり、出力は SOC の微分値と電圧である。SOC は、SOC 微分値を積分して推定している。Stateful Predict ブロックの前後では、信号の値を標準化、逆標準化した。理由はニューラルネットワーク学習時に信号を標準化したからである。

2.3.3 LSTM 版 HEV モデル

HEV の Battery モデルを LSTM 版 Battery モデルに置き換えた LSTM 版 HEV モデルを用意した。

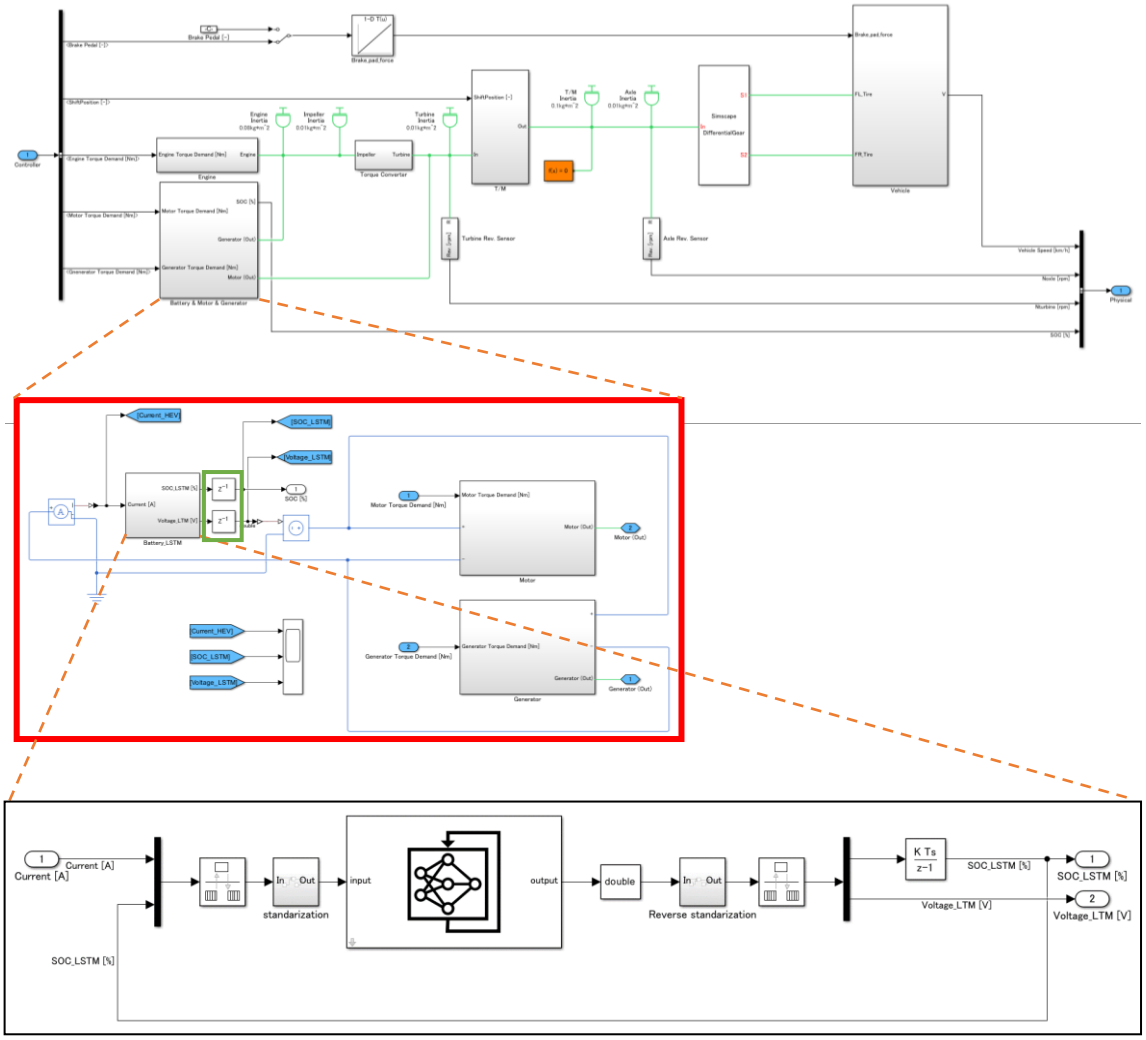


Fig. 2-5　サロゲートモデル版バッテリモデルを組み込んだ HEV モデル（PMWG_HEV_LSTM.slx）

LSTM 版 Battery モデルは Simulink で作成される。Simscape で構築されたプラントモデルに組込むために Current Sensor と Controlled Voltage Source を用いた。また代数ループを避けるため、一次遅れブロックを追加した（Fig 2.5 緑枠参照）。LSTM 版 Battery モデルは、検証用モデルで示した構成と同一である。

3. 学習・検証用モデル スクリプト解説

LSTM 版 HEV モデルを構築するにあたり、学習用モデルおよび検証用モデルを構築、実行するためのスクリプトを新たに用意した。本章ではこのスクリプトについて概要を説明する。

3.1 学習用データ生成、検証用モデル実行スクリプト

学習用データの取得および、検証用モデルを実行するスクリプトである learning_PMWG_HEV_Battery.m の中身について概要を解説する。

まず初めに、ワークスペースの初期化とパスの設定を行う（①)。次に、学習データ取得用モデルを用いて任意時間分のシミュレーション結果を取得し（②)、学習用データと検証用データに分割する（③)。現状では学習用データの範囲と検証用データの範囲はユーザーが手動で設定することとしている。その後、学習用スクリプト（learning_PMWG_function_LSTM1data.m）を用いてネットワークの学習を実施し（④)、学習に用いた Simscape で作成した Battery モデルと、学習済み再帰型ニューラルネットワークを用いた Battery モデルの性能を比較する（⑤）

```
%% 初期化
clear functions;
clear all;
close all;

%% start
tic;

%% パスの追加
addpath(genpath([pwd,'\PMWS_Simscape_Library']))

%% モデル実行
% シミュレーション時間を設定
end_time = 4500;

% パターン読み込み
load('test_input_nn_current.mat');

% モデルオープン
open_system('PMWG_HEV_Battery_Data_Acquisition');

% モデルの実行と結果保存
in = Simulink.SimulationInput('PMWG_HEV_Battery_Data_Acquisition');
out = sim(in,'ShowProgress','on');
save('PMWG_HEV_Battery_data_acquisition_simulation.mat',"out")

%% 事前準備
% 入力信号の調整
Current = squeeze(out.log.signals(2).values)';
SOCz1   = squeeze(out.log.signals(4).values)';
X_transient = [Current; SOCz1];

% 出力信号の調整
dSOC    = squeeze(out.log.signals(6).values)';
Voltage = squeeze(out.log.signals(3).values)';
Y_transient = [dSOC; Voltage];

% spliti Train Test
%【課題】index で分けているが、sim 時間変更時など手動入力が必要
Index_train = 2 : 4500001*1/2;
X_train = X_transient(:, Index_train);
Y_train = Y_transient(:, Index_train);
Index_test = 2 : 4500001*1/2;
X_test = X_transient(:, Index_test);
Y_test = Y_transient(:, Index_test);

%% DeepLearningToolbox の LSTM を用いた学習
% 学習の実行
[testY_Pred, rmse, net, Xmu, Xsig, Ymu, Ysig] =
learning_PMWG_function_LSTM1data(X_train,Y_train, X_test, Y_test);

% 保存
save('PMWG_HEV_Battery_net.mat','net')
save('PMWG_HEV_Battery_data_params.mat','X_transient', 'Y_transient',
'X_train','Y_train', 'X_test', 'Y_test', "Xmu","Xsig","Ymu","Ysig")

%% 検証
% モデルオープン
open_system('PMWG_HEV_Battery_Verification');

% 検証用モデルの実行
%【課題】リアルタイムより時間がかかる 60 秒の sim で約 300 秒必要(メモリ 32GB の Windows-PC)
in_ver = Simulink.SimulationInput('PMWG_HEV_Battery_Verification');
out_ver = sim(in_ver,'ShowProgress','on');

%% end
toc;
```

①初期設定

②学習用データ取得

【注 1】
サンプルモデルでは簡略化のため、Train と Test で同じデータを使用

③学習用データ整形

【注 2】
学習範囲と検証範囲は各々のデータで標準化

④学習用スクリプト実行

⑤Battery モデル計算結果比較

3.2 学習用スクリプト

再帰型ニューラルネットワーク学習用スクリプト
learning_PMWG_function_LSTM1data.m について概要を説明する。
まず初めに、学習用データを用いた学習検証用データの標準化、再帰型ニューラルネットワークの定義、学習パラメータの設定を行う（①)。次に、①で定義した条件でニューラルネットワークの学習を行う（②)。学習したニューラルネットワークを用いて、学習データに対する推論と、検証データに対する推論を実行し、性能は RMSE で評価する（③)。最後に、実行結果を可視化する（④)

```
function [testY_Pred, rmse, net, Xmu, Xsig, Ymu,
Ysig]=learning_PMWG_function_LSTM1data(X_train,Y_train, X_test, Y_test)
% ネットワーク学習に関する設定
training = true;
params.plot = 'training-progress'; % 'training-progress' / 'none'
params.executionenv = 'cpu';

%% 過渡応答データの準備
% 平均･分散の計算
Xmu = mean(X_train,2);
Xsig = std(X_train,0,2);
Ymu = mean(Y_train,2);
Ysig = std(Y_train,0,2);
% 平均･分散を用いてデータを標準化
std_trainX = (X_train - Xmu) ./ Xsig;
std_trainY = (Y_train - Ymu) ./ Ysig;
std_testX  = (X_test - Xmu) ./ Xsig;
std_testY  = (Y_test - Ymu) ./ Ysig;

%% 過渡応答用ネットワークの作成
% 入出力の次元
numFeatures = size(X_train,1);
numResponses = size(Y_train,1);
% LSTM ノードの数
numHiddenUnits = 50;

% ネットワークの定義
layers = [ ...
    sequenceInputLayer(numFeatures,"Name","Sequenceinput")
    lstmLayer(numHiddenUnits,"Name","lstm","OutputMode","sequence")
    fullyConnectedLayer(numResponses,"Name","fc2")
    regressionLayer("Name","regressionoutput")];
lgraph = layerGraph(layers);

% 学習パラメータの設定
maxEpochs = 50;% 損失関数がほぼ収束するために必要な値。今回は 10 と仮定
miniBatchSize = 64; % 学習データセットをサブセットへ分割する際のデータ数。2 の n 乗の値と
することが多い。今回は 64 を採用
options = trainingOptions('adam', ...
    'MaxEpochs',maxEpochs, ...
    'MiniBatchSize',miniBatchSize, ...
    'InitialLearnRate',0.01, ...
    'ValidationFrequency',50,...
    'Shuffle','every-epoch', ...
    'Plots',params.plot,...
    'Verbose',0);
% ネットワークの学習
if training == true
    disp('train LSTM model...')
    % ネットワークの学習
    [net, info] = trainNetwork(std_trainX, std_trainY, lgraph, options);
else
    % training フラグが False であればネットワークの読み込み
    load('net.mat');
end
```

①学習条件の設定

②ネットワークの学習

```
%% 推論(学習したモデルを実行し数値を予測)と評価
% train は学習用。test は検証用。
[~, std_trainY_Pred] = predictAndUpdateState(net, std_trainX,
'MiniBatchSize',1);
[~, std_testY_Pred]  = predictAndUpdateState(net, std_testX,
'MiniBatchSize',1);
% 標準化して扱ってきたので、元のスケールに戻す。
trainY_Pred = (std_trainY_Pred .* Ysig) + Ymu;
testY_Pred  = (std_testY_Pred  .* Ysig) + Ymu;
% RMSE を計算
rmse = sqrt(mean((Y_test - testY_Pred)'.^2));

%% 結果をグラフで表示
modeltype = "LSTM";
% グラフ&評価結果(左側:学習用データ、右側:評価用データ)
figure('Name', '推論'),
for n = 1 : size(Y_train, 1)
    subplot(size(Y_train, 1), 2, 2*n-1)
    plot(Y_train(n, :),'DisplayName','standardizedYtrain2');
    hold on;
    plot(trainY_Pred(n, :),'DisplayName','YPred');
    legend('真値', '推論値')
    grid on
    hold off;
    if n == 1
        title(["【学習:SOC】データ番号:"+ num2str(n)]);
        xlabel('Index [-]')
        ylabel('SOC [%]')
    elseif n == 2
        title(["【学習:Voltage】データ番号:"+ num2str(n)]);
        xlabel('Index [-]')
        ylabel('Voltage [V]')
    end

    subplot(size(Y_train, 1), 2, 2*n)
    plot(Y_test(n, :),'DisplayName','standardizedYtrain2');
    hold on;
    plot(testY_Pred(n, :),'DisplayName','YPred');
    legend('真値', '推論値')
    grid on
    hold off;
    if n == 1
        title(["【検証:SOC】データ番号:"+ num2str(n)]);
        xlabel('Index [-]')
        ylabel('SOC [%]')
    elseif n == 2
        title(["【検証:Voltage】データ番号:"+ num2str(n)]);
        xlabel('Index [-]')
        ylabel('Voltage [V]')
    end
end
end
```

③学習済みネットワークの評価

④学習済みネットワークの評価結果可視化

4. サンプルモデル実行方法および実行結果

本章では、学習用モデルおよび検証用モデルと、HEV モデルの実行手順を示す。

【事前実施事項】

・初回モデル実行時、Simscape ファイル変更時は、Simscape ライブラリをビルドすること。ビルドする場合は、下記コマンドを実行する。

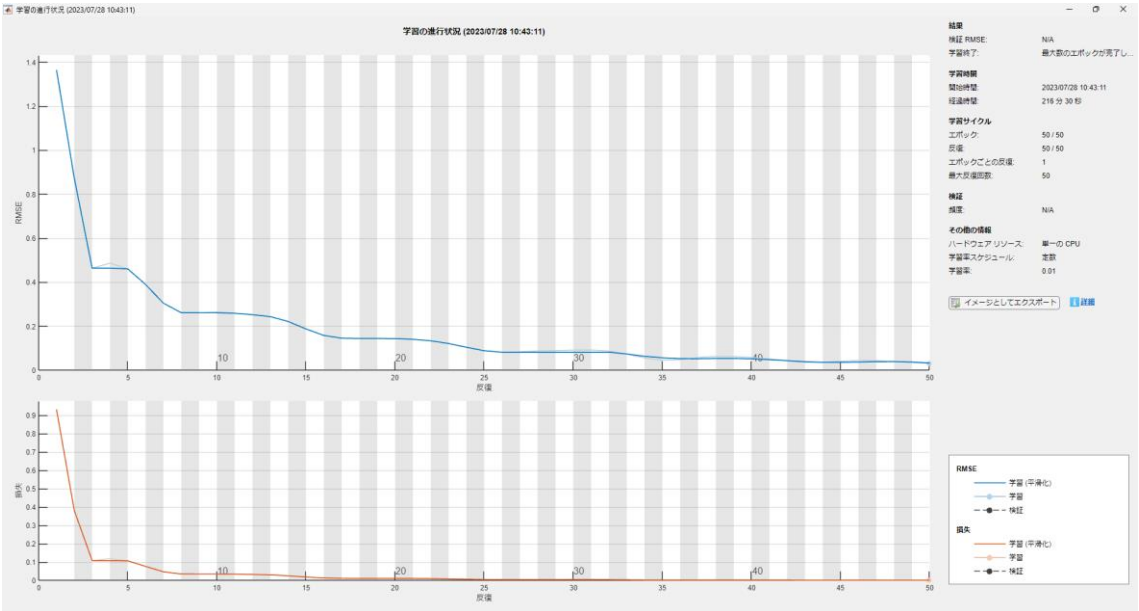
・build_Simscape_Library.m

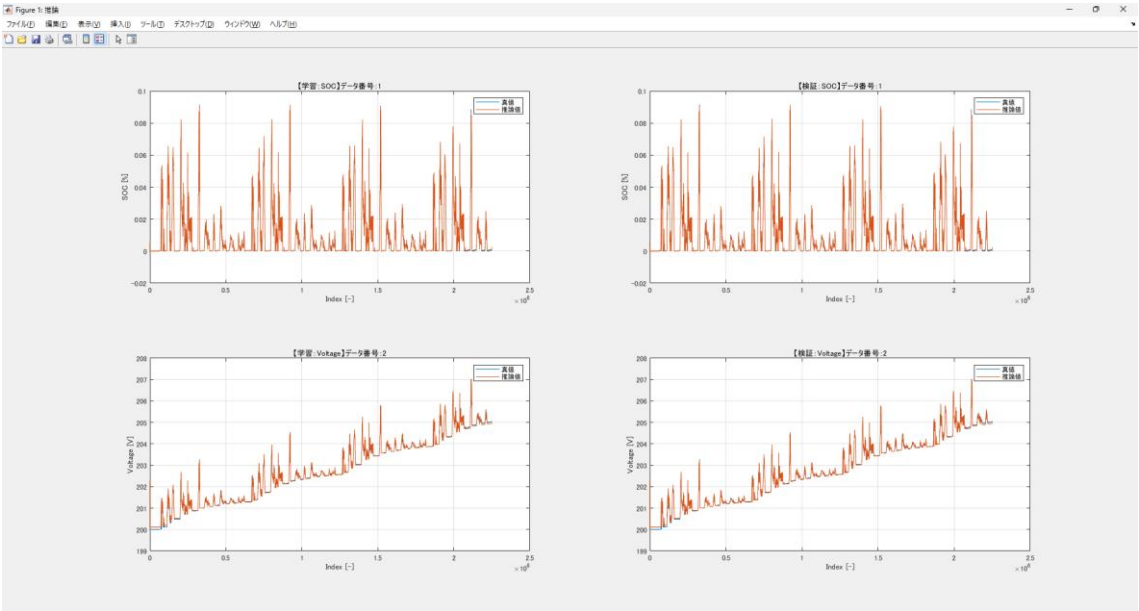
4.1 学習用モデル、検証用モデル

4.1.1 実行手順

1. learning_PMWG_HEV_Battery.m を実行
2. 終了

4.1.2 実行結果





4.2 LSTM 版 HEV モデル

4.2.1 実行手順

1. launch_PMWG_HEV_LSTM.m を実行
2. 起動したモデル（PMWG_HEV_LSTM.slx）を実行
3. 終了

4.2.2 実行結果

4.3 HEV モデル

4.3.1 実行手順

1. launch_PMWS_HEV.m を実行
2. 起動したモデル（PMWS_HEV.slx）を実行
3. 終了

4.3.2 実行結果

5. まとめ

JAMMB PMWG モデリングサブ WG では、近年注目されている機械学習/深層学習を用いたプラントモデルのサロゲートモデル化について、サンプルモデルを作成しながら、そのプロセスや手法を明確にすることを目的に活動を実施した。本書はこの活動内容を基に、サンプルサロゲートモデルの構成と実行方法を整理した。本書の内容が、読者の一助となれば幸いである。