```
% HEMS Smart Battery Control Full Script

clc;
clear all;
close all;

%% === 1) Load & Interpolate Solar Data ===
solarData    = readtable('Corrected_House_1_Solar_Production.xlsx');
dailySolar   = solarData.Power;               % [365×1]
t_daily      = (0:length(dailySolar)-1)';     % days
t_hourlyDays = linspace(0, 364, 365*24)';     % fractional days

% Linear interpolation to hourly resolution
hourlySolar = interp1(t_daily, dailySolar, t_hourlyDays, 'linear');

% Create a realistic daylight bell curve (24-hour profile)
daylightProfile = sin((pi/24) * (0:23));  % 1×24
daylightProfile(daylightProfile < 0) = 0; % clamp negative

% Expand to match 365 days → 365×24
fullProfile = repmat(daylightProfile, 365, 1);

% Reshape interpolated data into 365×24
hourlyMatrix = reshape(hourlySolar, 24, [])';  % 365×24

% Apply daylight profile element-wise
hourlySolarRealistic = fullProfile .* hourlyMatrix;  % 365×24
```

```
% Flatten back to 8760×1 (hours)
hourlySolarRealistic = reshape(hourlySolarRealistic', [], 1);

% Create solar timeseries (time in hours)
ts_solar = timeseries(hourlySolarRealistic, (0:length(hourlySolarRealistic)-1)');
ts_solar.Name        = 'SolarHourly';
ts_solar.TimeInfo.Units = 'hours';
assignin('base', 'solarTS', ts_solar);

%% === 2) Load Consumption Data ===
consumption      = readtable('House_1_Consumption_Reshaped.xlsx');
consumptionPower = consumption.Power;          % [365×1]
t_consDays       = (0:length(consumptionPower)-1)';  % days

% Create consumption timeseries (assume 1-day steps; Simulink can interpolate)
ts_load = timeseries(consumptionPower, t_consDays);
ts_load.Name = 'LoadConsumption';
assignin('base', 'consumptionTS', ts_load);

%% === 3) Prepare & Train LSTM for Next-Day Consumption ===
data     = [solarData.Power, consumption.Power];   % daily [365×2]
minVals  = min(data);
maxVals  = max(data);
dataNorm = (data - minVals) ./ (maxVals - minVals);

sequenceLength = 30;
```

```
numSamples     = size(dataNorm,1) - sequenceLength;
XTrain         = cell(numSamples,1);
YTrain         = zeros(numSamples,1);

for i = 1:numSamples
    XTrain{i}    = dataNorm(i : i+sequenceLength-1, :)';  % [2×30]
    YTrain(i)    = dataNorm(i+sequenceLength, 2);        % next-day consumption
end

layers = [
    sequenceInputLayer(2)
    lstmLayer(50, 'OutputMode','last')
    fullyConnectedLayer(1)
    reluLayer
    regressionLayer
];

options = trainingOptions('adam', ...
    'MaxEpochs',150, ...
    'GradientThreshold',1, ...
    'InitialLearnRate',0.005, ...
    'Verbose',0, ...
    'Plots','training-progress');

net = trainNetwork(XTrain, YTrain, layers, options);

%% === 4) Predict & Assign Consumption Timeseries ===
```

```
lastSeq      = dataNorm(end-sequenceLength+1:end, :)';
predNorm     = predict(net, {lastSeq});
predictedKWh = predNorm * (maxVals(2) - minVals(2)) + minVals(2);
predictedKWh = max(0, predictedKWh);  % clamp ≥ 0

fprintf("? Predicted next day consumption (norm): %.4f\n", predNorm);
fprintf("? Predicted consumption (kWh): %.4f\n", predictedKWh);

% Create constant timeseries over 8760 hours
tsTime   = [0, 8760]';
tsValues = [predictedKWh, predictedKWh]';
ts_pred  = timeseries(tsValues, tsTime);
ts_pred.Name = 'PredictedConsumption';
assignin('base', 'predictedTS', ts_pred);

%% === 5) Run Simulink Model ===
simOut = sim('commpjt');

%% === 6) Plot Logged Signals ===
if isprop(simOut, 'logsout') && isa(simOut.logsout, 'Simulink.SimulationData.Dataset')
   sigNames = simOut.logsout.getElementNames;
   disp('Logged signals:');
   disp(sigNames);

   figure;

   % 6.1 Battery SOC
```

```
subplot(4,1,1);
try
    socStruct = simOut.logsout.getElement('SOC').Values;
    % Navigate struct to actual timeseries
    if isstruct(socStruct) && isfield(socStruct, 'SOC____')
        socTS = socStruct.SOC____;
    else
        socTS = socStruct;  % assume timeseries
    end
    plot(socTS.Time, socTS.Data, 'b');
    title('Battery SOC');
    ylabel('SOC');
    xlabel('Time (hours)');
catch
    warning('SOC plot failed');
end

% 6.2 Solar Current
subplot(4,1,2);
try
    sc = simOut.logsout.getElement('SolarCurrent').Values;
    plot(sc.Time, sc.Data, 'r');
    title('Solar Current');
    ylabel('Current (A)');
    xlabel('Time (hours)');
catch
    warning('SolarCurrent plot failed');
```

```
    end

    % 6.3 Battery Control Current
    subplot(4,1,3);
    try
        gc = simOut.logsout.getElement('BatteryControlCurrent').Values;
        plot(gc.Time, gc.Data, 'g');
        title('Battery Control Current');
        ylabel('Current (A)');
        xlabel('Time (hours)');
    catch
        warning('BatteryControlCurrent plot failed');
    end

    % 6.4 Load Current
    subplot(4,1,4);
    try
        lc = simOut.logsout.getElement('LoadCurrent').Values;
        plot(lc.Time, lc.Data, 'm');
        title('Load Current');
        ylabel('Current (A)');
        xlabel('Time (hours)');
    catch
        warning('LoadCurrent plot failed');
    end

else
```

```
    error('No logsout Dataset found. Enable signal logging in Simulink.');
end
```