Appendix

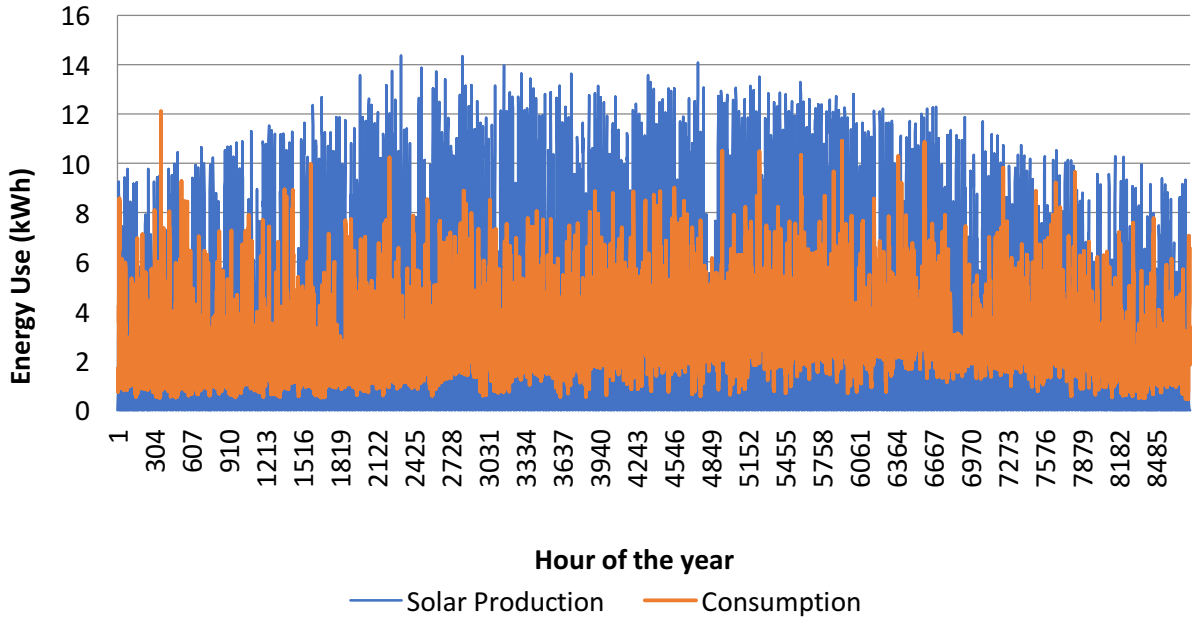

Figure A1: Hourly load and production profiles for Hawaii

Exhibit A1 – Code for all Schedule R Programs

```
%% GET RESIDENTIAL LOAD PROFILE
[num, ~, raw] = xlsread('Updated HI Residential Data.xlsx', 'Raw Data');
[~,~,raw1] = xlsread('Residential_Honolulu_Base.xlsx');
TwoM_vec  =  [num(70081:105120,6), num(70081:105120,1)];
%ThreeL_vec = [num(1:35040,6), num(1:35040,1)];

res_load   = spreadsheet_modify_fcn(TwoM_vec);
date_str   = cell(length(raw1)-1,1);

%Extracting data on residential loads
for iter = 2:length(raw1)
    res_load(iter,1) = datenum(raw1{iter,1}); %recording timestamp for load data
    date_str{iter}     = datestr(res_load(iter,1));
end
res_load(1,:) = [];
date_str(1,:) = [];

%% Output File Name
file_name = 'All Results Flat Rate.xlsx';
```

```
output_month =
{'January';'February';'March';'April';'May';'June';'July';'August';'September';'Octo
ber';'November';'December'};

%% TESLA POWERWALL 2 SPECIFICATIONS & COSTS
steady_power_draw = 5; %in kW
max_power_draw    = 7; %in kW, lasts 10 seconds
batt_eff      = 0.89;
storage_capacity = 13.2; %in kWh
depth_of_discharge = 1; %can discharge 100% of capacity
total_batt_cost = 7600; %in dollars, includes battery, install costs and
supporting hardware

%% FFR + NSAR SPECIFICATIONS
%Assumptions -  FFR 2 (responding within 30 cycles) and NSAR; x

%Residential Incentives - FFR
capab_incentive_ffr = 8 * steady_power_draw; %in $/month

%Residential Incentives - NSAR
capab_incentive_nsar = 6 * steady_power_draw; %in $/month

%Constraints - FFR
min_ffr_avail = 0.5 * steady_power_draw; %in kWhr, 30-min minimum
availability for FFR events at nominated capability

%Assumptions - FFR
avg_ffr_duration = 30/60; %in hr, assuming the average FFR event lasts 10
seconds

%Constraints - NSAR
min_nsar_avail = 1.0 * steady_power_draw; %in kWhr, 1 hour minimum
availability for NSAR events at nominated capability
max_nsar_time = 100; %hours/year that NSAR can be deployed

%Hard-coding FFR events
num_events_ffr = 60; %in one year
ffr_events = zeros(num_events_ffr,2);
ffr_events(:,1) =
[736707.603072136;736708.555595945;736709.016906427;736712.8294713
51;736731.404533136;736731.553476493;736739.374653551;736742.28669
3069;736747.717568787;736753.449966447;736755.271975596;736758.397
545726;736764.115062197;736777.579449354;736796.611137371;736796.9
38424813;736797.512597554;736811.582756812;736820.070593430;736835
.078053194;736838.966744081;736849.731982636;736855.922314017;7368
58.416165002;736872.919418342;736874.519116596;736877.653691179;73
```

6895.338307421;736909.330088458;736926.494937930;736931.581084705;
736934.783214866;736934.921683383;736935.017477253;736943.39895609
0;736943.751617846;736949.265025033;736953.353790911;736954.563472
637;736966.871776030;736971.083286848;736972.196268826;736975.0308
60420;736984.759562455;736985.850354761;736987.702221647;736992.96
6774261;736996.150368895;737005.507622161;737026.161150580;737028.
925488991;737029.785588333;737036.440527997;737042.355925804;73704
4.887341556;737045.011230434;737045.734978315;737045.826661771;737
047.701859912;737049.781057718];
ffr_events(:,2) =
[0.0775828933215146;0.0337388207514148;0.0664722381326890;0.066231
1871026733;0.0403087393732887;0.0533209472725848;0.01915654788632
39;0.0176354249074033;0.0507498300700675;0.0679977243126397;0.0787
507419603599;0.0229101533662313;0.0533905320050134;0.046485461184
5976;0.0147154214931418;0.0373001836388112;0.0251515491800863;0.06
90475375474935;0.0355010445864448;0.0505925788545981;0.0253922728
819292;0.0556931903751137;0.0321507836486211;0.0593110485053321;0.
0617510071625005;0.0658438606127576;0.0451764998960068;0.01970981
79164536;0.0297900672720013;0.0773150945487271;0.0244706957617516;
0.0712372901034408;0.0512737802263929;0.0830649108768670;0.0193177
450523044;0.0446304354010727;0.0212953312625406;0.0806873667260454
;0.0142107100093102;0.0677021156049655;0.0706460569898217;0.07421
49100946882;0.0197524892715910;0.0416515728540900;0.0319354446424
065;0.0694492000155769;0.0438481824627462;0.0771283051687169;0.026
5171547432536;0.0322085358695826;0.0239957625267165;0.02333809435
47683;0.0742564033083395;0.0541461519003868;0.0520736251275231;0.0
239551943210921;0.0731271609529093;0.0570871619086851;0.038260582
0064077;0.0495312180463232];

%Hard-coding NSAR events
num_events_nsar = 24; %in one year
nsar_events = zeros(num_events_nsar,2);
nsar_events(:,1) =
[736746.528607807;736750.417663298;736767.658966507;736784.7648482
82;736787.520063326;736788.685851292;736788.982166632;736789.86175
5648;736823.569078651;736824.179886254;736868.513779947;736880.421
339476;736895.460054338;736896.374248860;736909.328754451;736920.5
48277325;736950.813457317;736969.836834422;736992.806809677;736998
.837025252;737002.441430075;737020.734225238;737034.716579896;7370
45.661713384];
nsar_events(:,2) =
[2.50753347396642;1.14986121537851;1.83298397314024;1.347162228479
86;1.59961578451007;1.83313552360376;2.57194612118487;1.0402267930
2393;4.59465045798034;4.76994662384019;2.87860038528367;2.87188599
333341;2.24049754092241;4.58355769340693;2.37186158800090;1.296678
14705745;4.08438361800474;2.45724515400522;1.84038035797430;2.5163

```
0060661714;1.23522718820162;1.38322205252640;4.75854412823119;4.81
722725095751];

%Check for max 24-hour deployment in FFR & NSAR event vectors
%For FFR
if length(ffr_events) > 2
    bad_ffr_event = dr_24hr_check(ffr_events);
    if ~isempty(bad_ffr_event)
        ffr_events(bad_ffr_event,:) = []; %deleting the bad FFR events
    end
end

%For NSAR
if length(nsar_events) > 2
    bad_nsar_event = dr_24hr_check(nsar_events);
    if ~isempty(bad_nsar_event)
        nsar_events(bad_nsar_event,:) = []; %deleting the bad NSAR events
    end
end


%% GETTING SOLAR DATA
%Assumptions
panel_eff = 0.162;
panel_size = 1.61; %in m2
num_panels = 20; %For a 2.5 kW System

[~,~,raw_solar] = xlsread('Honolulu_Solar.xlsx');
solar_vec = zeros(length(raw_solar)-2,2);

%extracting solar data into a workable form
for iter_solar = 3:length(raw_solar)
    temp_date = datevec(raw_solar{iter_solar,1});
    temp_date(1) = 2017;
    if raw_solar{iter_solar,3} == 24
        temp_date(4) = 0;
    else
        temp_date(4) = raw_solar{iter_solar,3};
    end
    temp_ghi = raw_solar{iter_solar,6}; %in Wh/m2, total horizontal radiation
(diffuse + direct)
    panel_output = (temp_ghi*panel_size*panel_eff*num_panels)/1000; %in
kWh, electricity produced by average solar panel in that hour of the year

    solar_vec(iter_solar-2,1) = datenum(temp_date);
    solar_vec(iter_solar-2,2) = panel_output;
```

```
end

solar_vec(end,1) = datenum([2018 1 1 0 0 0]);

%% SOLAR PANEL COST

install_cost = 10441; %For a 2.5 kW system

%% FLAT RATE; NO DEMAND SERVICES; HONOLULU
%Assumptions - Residential Rate R, Single Phase, Month of January
considered, No DR Services involved,
%Base Load Profile Considered, Honolulu

%Sheet Name
sheet_name = 'Conventional Chris';

[total_resi_bill, month_resi_bill, month_resi_consum] = schedule_r(res_load);

%Writing Residential Load Consumption
xlswrite(file_name,{'Timestamp'},sheet_name,'A1');
xlswrite(file_name,date_str, sheet_name,'A2');
xlswrite(file_name,{'Consumption(kWh)'},sheet_name,'B1');
xlswrite(file_name,res_load(:,2),sheet_name,'B2');

%Writing Monthly Residential Load & Bill Consumption
xlswrite(file_name,{'Month'},sheet_name,'D2');
xlswrite(file_name,output_month,sheet_name,'D3');
xlswrite(file_name,{'Monthly Consumption (kWh)'},sheet_name,'E2');
xlswrite(file_name,month_resi_consum', sheet_name,'E3');
xlswrite(file_name,{'Monthly Bill ($/mo)'},sheet_name,'F2');
xlswrite(file_name, month_resi_bill, sheet_name,'F3');

%OUTPUTS TO EXCEL - MONTH_RESI_BILL, MONTH_RESI_CONSUM

%% FLAT RATE, FFR + NSAR, HONOLULU
%Assumptions - FFR & NSAR can be used; Stacked on Residential Rate R;
%Powerwall 2;  Honolulu. Battery ONLY used for DR Services (No
%arbitrage)

%Initialization
battery_level = zeros(length(res_load),1); %How many kWh are in the battery at
every hour
ffr_tracker    = zeros(length(res_load),1); %When do FFR events take place
ffr_tracker_amt = zeros(length(res_load),1); %Magnitude of FFR event
nsar_tracker   = zeros(length(res_load),1); %When do NSAR events take place
nsar_tracker_amt = zeros(length(res_load),1); %Magmitude of NSAR event
```

```
battery_level(1) = storage_capacity; %in kWh,
ffr_count = 1;
nsar_count = 1;

%Output sheet
sheet_name = 'DR Dave';

%The system running through the whole year; hourly basis
for iter_hour = 1:length(res_load)-1 %iterating through each hour in January

    current_timestamp = res_load(iter_hour,1);
    next_timestamp = res_load(iter_hour+1,1);
    current_battery_level = battery_level(iter_hour);

    if current_battery_level <  min_nsar_avail %if the battery level in any hour is
lower than the minimum NSAR requirement (which is larger than the FFR one)
        break
    end

    %Testing if FFR event occured in the interim
    if current_timestamp < ffr_events(ffr_count,1) && next_timestamp >
ffr_events(ffr_count,1)
        ffr_tracker(iter_hour) = true;
        ffr_tracker_amt(iter_hour) = (ffr_events(ffr_count,2)/batt_eff);
        battery_level(iter_hour+1) = battery_level(iter_hour) -
ffr_tracker_amt(iter_hour); %Assuming that the average FFR event lasts 10
seconds
        if ffr_count < length(ffr_events)
            ffr_count = ffr_count + 1;
        end

        %Testing if NSAR event occured in the interim
    elseif  current_timestamp < nsar_events(nsar_count,1) && next_timestamp >
nsar_events(nsar_count,1)
        nsar_tracker(iter_hour) = true;
        nsar_tracker_amt(iter_hour) = ((nsar_events(nsar_count,2))/batt_eff);

        battery_level(iter_hour+1) = battery_level(iter_hour) -
nsar_tracker_amt(iter_hour); %Subtracting the NSAR requirement from the
battery
        if nsar_count < length(nsar_events)
            nsar_count = nsar_count+1;
        end

    %If no FFR or NSAR event has taken place in that time-stamp
    else
```

```
        if battery_level(iter_hour) < storage_capacity %if the battery is not full
            %if deficit in battery is more than which can be charged in 1 hr
            if (storage_capacity - battery_level(iter_hour)) > steady_power_draw;
                battery_level(iter_hour+1) = battery_level(iter_hour) +
steady_power_draw;
            %if deficit is less than the steady power draw of the battery
            else
                battery_level(iter_hour+1) = storage_capacity;
            end
        else
            battery_level(iter_hour+1) = storage_capacity;
        end
    end
end

%Calculating Payback Period
total_monthly_incentive = capab_incentive_ffr + capab_incentive_nsar; %FFR
and NSAR monthly incentives gained together
simple_payback_ffr_nsar = total_batt_cost/(total_monthly_incentive * 12);
%Simple Payback of this rate schedule/DR combination

%Writing Residential Load Consumption
xlswrite(file_name,{'Timestamp'},sheet_name,'A1');
xlswrite(file_name,date_str, sheet_name,'A2');
xlswrite(file_name,{'Consumption(kWh)'},sheet_name,'B1');
xlswrite(file_name,res_load(:,2),sheet_name,'B2');

%Writing Residential State of Charge
xlswrite(file_name,{'State of Charge'},sheet_name,'C1');
xlswrite(file_name,battery_level,sheet_name,'C2');

%Writing DR Events
xlswrite(file_name,{'FFR Event Magnitude (kWh)'},sheet_name,'D1');
xlswrite(file_name, ffr_tracker_amt, sheet_name, 'D2');
xlswrite(file_name,{'NSAR Event Magnitude (kWh)'}, sheet_name, 'E1');
xlswrite(file_name, nsar_tracker_amt, sheet_name, 'E2');

%Writing Monhtly Bill & Consumption
xlswrite(file_name,{'Month'},sheet_name, 'F1');
xlswrite(file_name, output_month, sheet_name, 'F2');
xlswrite(file_name,{'Monthly Consumption (kWh/mo)'},sheet_name,'G1');
xlswrite(file_name, month_resi_consum', sheet_name, 'G2');
xlswrite(file_name, {'Bill ($/mo)'}, sheet_name, 'H1');
xlswrite(file_name, (month_resi_bill - total_monthly_incentive), sheet_name,
'H2');
```

```
%% FLAT RATE, W/ SOLAR PANEL, CSS, NO DR
%Assumptions - Residential (Base Case), Schedule Rate R, No export
%compensation, Solar Panel present, Battery Present, January only

battery_solar_level = zeros(length(res_load),1);
battery_solar_level(1) = storage_capacity; %assuming that the battery starts the
year fully charged
solar_prod_level = zeros(length(res_load),1);
solar_waste_level = zeros(length(res_load),1);

%Output Sheet
sheet_name = 'Self Supply Sam';

for iter_hour_solar = 1:length(res_load)-1 %iterating through all of the year's
load points, per hour

    current_load = res_load(iter_hour_solar,2); %the load in current iteration
    current_prod = solar_vec(iter_hour_solar,2); %the production in current
iteration
    excess_prod = current_prod - current_load; %How much did the solar panel
produce in excess of the load
    battery_space = storage_capacity - battery_solar_level(iter_hour_solar); %how
much free space is in the battery

    %If solar production is greater than load at that moment:
    if excess_prod > 0
        solar_prod_level(iter_hour_solar) = current_load;
        %Is the excess amount of production more than the maximum battery
absorption and how much can the battery actually absorb at that time?
        if excess_prod > battery_space
            min_value = min(battery_space, steady_power_draw);
            battery_solar_level(iter_hour_solar+1) =
battery_solar_level(iter_hour_solar) + min_value;
            solar_prod_level(iter_hour_solar) = solar_prod_level(iter_hour_solar) +
min_value;
            solar_waste_level(iter_hour_solar) = excess_prod - min_value;
        else %excess production can fit in battery
            if excess_prod > steady_power_draw
                battery_solar_level(iter_hour_solar+1) =
battery_solar_level(iter_hour_solar) + steady_power_draw;
                solar_prod_level(iter_hour_solar) = solar_prod_level(iter_hour_solar) +
steady_power_draw;
                solar_waste_level(iter_hour_solar) = excess_prod -
steady_power_draw;
            else %This is when the battery space is less than the maximum of what
can be absorbed
```

```
            battery_solar_level(iter_hour_solar+1) =
battery_solar_level(iter_hour_solar) + excess_prod;
            solar_prod_level(iter_hour_solar) = solar_prod_level(iter_hour_solar) +
excess_prod;
        end
    end

  else %either panel is not producing anything or it is producing less than
required load
    if current_prod ~= 0 %if the panel is producing something
        solar_prod_level(iter_hour_solar) = current_prod;
        current_load = current_load - current_prod; %this is the net load that
still needs to be satisfied after solar production
    end
    if battery_solar_level(iter_hour_solar) >= (current_load/batt_eff) %if battery
has enough to meet load
        if current_load >= steady_power_draw %if the load is more than
maximum possible discharge by battery
            battery_solar_level(iter_hour_solar+1) =
battery_solar_level(iter_hour_solar) - (steady_power_draw/batt_eff);
        else %if the load is less than maximum possible discharge from battery
            battery_solar_level(iter_hour_solar+1) =
battery_solar_level(iter_hour_solar) - (current_load/batt_eff);
        end
    else %if battery does not have enough to meet load
        if battery_solar_level(iter_hour_solar) >= (steady_power_draw/batt_eff)
            battery_solar_level(iter_hour_solar+1) =
battery_solar_level(iter_hour_solar) - (steady_power_draw/batt_eff);
        else %if the battery cannot meet load and has less energy than max
possible discharge
            battery_solar_level(iter_hour_solar+1) = 0;
        end
    end
  end
end

%In the above case, the self-supply nature is emphasized. Battery charging
%from grid is restricted and only happens from solar panel.

total_solar_prod = sum(solar_prod_level);
total_solar_waste = sum(solar_waste_level);

net_load_vec(:,1) = res_load(:,1);
net_load_vec(:,2) = res_load(:,2) - solar_prod_level;

percent_wasted   = (total_solar_waste/sum(solar_vec(:,2)))*100;
```

```
[total_solar_bill, month_solar_bill, month_solar_consum] =
css_schedule_r(net_load_vec);

solar_savings = total_resi_bill - total_solar_bill;
savings_percent = (solar_savings/total_resi_bill)*100;

simple_payback_css_base = (install_cost+  total_batt_cost)/(solar_savings); %in
years, costs include 5 kW solar system and Tesla Powerwall

%Writing Residential Load Consumption
xlswrite(file_name,{'Timestamp'},sheet_name,'A1');
xlswrite(file_name,date_str, sheet_name,'A2');
xlswrite(file_name,{'Consumption(kWh)'},sheet_name,'B1');
xlswrite(file_name,res_load(:,2),sheet_name,'B2');

%Writing Solar Production & Waste
xlswrite(file_name, {'State of Charge (kWh)'},sheet_name, 'C1');
xlswrite(file_name, battery_solar_level, sheet_name, 'C2');
xlswrite(file_name, {'Solar Energy Used (kWh)'}, sheet_name, 'D1');
xlswrite(file_name, solar_prod_level, sheet_name, 'D2');
xlswrite(file_name, {'Solar Energy Wasted (kWh)'}, sheet_name, 'E1');
xlswrite(file_name, solar_waste_level, sheet_name, 'E2');

%Writing Monthly Consumption & Bill
xlswrite(file_name,{'Month'},sheet_name, 'G1');
xlswrite(file_name, output_month, sheet_name, 'G2');
xlswrite(file_name, {'Monthly Consumption (kWh)'}, sheet_name, 'H1');
xlswrite(file_name, month_solar_consum', sheet_name, 'H2');
xlswrite(file_name, {'Monthly Bill ($/mo)'}, sheet_name, 'I1');
xlswrite(file_name, month_solar_bill, sheet_name, 'I2');


%% FLAT RATE W/ SOLAR PANEL, CSS, DR
%Assumptions - FFR + NSAR, Each cannot be deployed more than twice in 24
%hours, No Elec. Export except during DR events, Max NSAR availability of
%100 hours/year. After DR event, available again in 6 hours. 5 kW solar
%panel present, battery present, Schedule Rate R, January onls

%Intialization
css_dr_tracker    = zeros(length(res_load),1);
battery_css_dr = zeros(length(res_load),1);
battery_css_dr(1) = storage_capacity; %assuming that the battery starts the
year fully charged
solar_prod_css_dr = zeros(length(res_load),1);
solar_waste_css_dr = zeros(length(res_load),1);
battery_grid_consum = zeros(length(res_load),1);
```

```
ffr_count_css_dr = 1;
nsar_count_css_dr = 1;

%DR Tracking
ffr_tracker = zeros(length(res_load),1);
ffr_tracker_amt = zeros(length(res_load),1);
nsar_tracker = zeros(length(res_load),1);
nsar_tracker_amt = zeros(length(res_load),1);

%Charges
gif_fee = 1.27; %in $/mo
minimum_charge_css_dr = 25; %in dollars

%Output Sheet
sheet_name = 'Sam w DR';

for iter_hour_css_dr = 1:length(res_load)-1

    dr_min_satisfied = false;
    dr_min_not_satisfied = false;

    current_timestamp = res_load(iter_hour_css_dr,1);
    next_timestamp = res_load(iter_hour_css_dr+1,1);
    current_battery_level = battery_css_dr(iter_hour_css_dr);

    current_load = res_load(iter_hour_css_dr,2); %the load in current iteration
    current_prod = solar_vec(iter_hour_css_dr,2); %the production in current
iteration
    excess_prod = current_prod - current_load; %How much did the solar panel
produce in excess of the load

    %Checking if FFR event has taken place
    if current_timestamp < ffr_events(ffr_count_css_dr,1) && next_timestamp >
ffr_events(ffr_count_css_dr,1)
        ffr_tracker(iter_hour_css_dr) = true;
        ffr_tracker_amt(iter_hour_css_dr) =
(ffr_events(ffr_count_css_dr,2)/batt_eff);
        battery_css_dr(iter_hour_css_dr+1) = battery_css_dr(iter_hour_css_dr) -
ffr_tracker_amt(iter_hour_css_dr); %Assuming that the average FFR event lasts
10 seconds

        if battery_css_dr(iter_hour_css_dr+1) > min_nsar_avail
            dr_min_satisfied = true;
        else
            dr_min_not_satisfied = true;
        end
```

```
        if ffr_count_css_dr < length(ffr_events)
            ffr_count_css_dr = ffr_count_css_dr + 1;
        end

    %Check if NSAR event has taken place
    elseif current_timestamp < nsar_events(nsar_count_css_dr,1) &&
next_timestamp > nsar_events(nsar_count_css_dr,1)
        nsar_tracker(iter_hour_css_dr) = true;
        nsar_tracker_amt(iter_hour_css_dr) =
(nsar_events(nsar_count_css_dr,2)/batt_eff);
        battery_css_dr(iter_hour_css_dr+1) = battery_css_dr(iter_hour_css_dr) -
nsar_tracker_amt(iter_hour_css_dr); %Subtracting the NSAR requirement from
the battery

        if battery_css_dr(iter_hour_css_dr+1) > min_nsar_avail
            dr_min_satisfied = true;
        else
            dr_min_not_satisfied = true;
        end

        if nsar_count_css_dr < length(nsar_events)
            nsar_count_css_dr = nsar_count_css_dr + 1;
        end
    end


    %if there is excess solar production vs. load
    if excess_prod >= 0
        %If DR event happens and battery level is above minimum required

        if dr_min_satisfied
            solar_prod_css_dr(iter_hour_css_dr) = current_load; %tracking how
much solar is used and for what
            battery_space_dr = storage_capacity -
battery_css_dr(iter_hour_css_dr+1); %Total capacity - battery level in this hour;
            [solar_prod_css_dr, solar_waste_css_dr, battery_css_dr] =
excess_prod_eval_dr(battery_css_dr, battery_space_dr, steady_power_draw,
excess_prod, solar_prod_css_dr, solar_waste_css_dr, iter_hour_css_dr);

        %If DR event happens and Battery Level falls below minimum required
        elseif dr_min_not_satisfied
            deficit_battery_css_dr = min_nsar_avail -
battery_css_dr(iter_hour_css_dr+1); %how much is lacking for battery to meet
minimum requirement
```

```
            if current_prod > deficit_battery_css_dr %if solar production can meet
deficit in minimum battery requirement
                solar_prod_css_dr(iter_hour_css_dr) = deficit_battery_css_dr; %solar
panel's energy used to let battery reach minimum
                battery_css_dr(iter_hour_css_dr+1)  = min_nsar_avail; %battery is
now at minimum level for next hour
                leftover_solar_css_dr               = current_prod -
deficit_battery_css_dr;
            else
                battery_css_dr(iter_hour_css_dr+1)  = min_nsar_avail; %the battery
will import energy from the grid to meet the minimum requirement
                battery_grid_consum(iter_hour_css_dr) = deficit_battery_css_dr;
%recording consumption from grid
                leftover_solar_css_dr               = current_prod;
            end
            %now that minimum battery level is fulfilled, do the rest
            if leftover_solar_css_dr > current_load
                solar_prod_css_dr(iter_hour_css_dr) =
solar_prod_css_dr(iter_hour_css_dr) + current_load;
                leftover_solar_css_dr = leftover_solar_css_dr - current_load;
%updating the excess production left
                battery_space_dr = storage_capacity -
battery_css_dr(iter_hour_css_dr+1); %Total capacity - battery level in this hour;
                [solar_prod_css_dr, solar_waste_css_dr, battery_css_dr] =
excess_prod_eval_dr(battery_css_dr, battery_space_dr, steady_power_draw,
leftover_solar_css_dr, solar_prod_css_dr, solar_waste_css_dr, iter_hour_css_dr);
            else
                solar_prod_css_dr(iter_hour_css_dr) =
solar_prod_css_dr(iter_hour_css_dr) + leftover_solar_css_dr; %if the left-over
solar is not enough to meet current load
            end
        %if there is no DR event at all
        else
            battery_space_dr = storage_capacity - battery_css_dr(iter_hour_css_dr);
            solar_prod_css_dr(iter_hour_css_dr) = current_load;
            [solar_prod_css_dr, solar_waste_css_dr, battery_css_dr] =
excess_prod_eval(battery_css_dr, battery_space_dr, steady_power_draw,
excess_prod, solar_prod_css_dr, solar_waste_css_dr, iter_hour_css_dr);
        end

    %If the panel produces less energy than the load needs
    elseif excess_prod < 0 && current_prod > 0
        %If DR event happens and battery level is above minimum required
        if dr_min_satisfied
            solar_prod_css_dr(iter_hour_css_dr) = current_prod; %if production <
load and battery minimum is maintained, current solar production is used
```

```
            current_load = current_load - current_prod; %if battery minimum is
met, solar used to displace load

            battery_css_dr = less_prod_eval_dr(current_load, steady_power_draw,
battery_css_dr,iter_hour_css_dr,min_nsar_avail,batt_eff);

        %If DR event happens and battery level below minimum required
        elseif dr_min_not_satisfied

            deficit_battery_css_dr = min_nsar_avail -
battery_css_dr(iter_hour_css_dr+1); %how much is lacking for battery to meet
minimum requirement
            if current_prod > deficit_battery_css_dr %if solar production can meet
deficit in minimum battery requirement
                solar_prod_css_dr(iter_hour_css_dr) = current_prod; %solar panel's
energy used to let battery reach minimum
                battery_css_dr(iter_hour_css_dr+1)  =
battery_css_dr(iter_hour_css_dr) + current_prod; %battery is now at minimum
level for next hour
                %leftover_solar_css_dr               = current_prod -
deficit_battery_css_dr;
            else
                deficit_battery_css_dr              = deficit_battery_css_dr -
current_prod;
                battery_grid_consum(iter_hour_css_dr) = deficit_battery_css_dr;
                battery_css_dr(iter_hour_css_dr+1)  =
battery_css_dr(iter_hour_css_dr) + deficit_battery_css_dr + current_prod; %the
battery will import energy from the grid to meet the minimum requirement
                %leftover_solar_cgs_dr               = 0;

            end

        %If DR event has NOT taken place BUT production < load
        else
            solar_prod_css_dr(iter_hour_css_dr) = current_prod; %if production <
load and battery minimum is maintained, current solar production is used
            current_load = current_load - current_prod; %if battery minimum is
met, solar used to displace load

            battery_css_dr = less_prod_eval_dr(current_load, steady_power_draw,
battery_css_dr,iter_hour_css_dr,min_nsar_avail,batt_eff);
        end

    %When there is no production from the solar panel, only load
    else
```

```
        if dr_min_satisfied %if there is a minimum in the battery after the DR
event
            battery_css_dr = less_prod_eval_dr(current_load, steady_power_draw,
battery_css_dr,iter_hour_css_dr,min_nsar_avail, batt_eff);
        elseif dr_min_not_satisfied %if the battery falls below a minimum after the
DR event
            %will be importing electricity from grid here
            deficit_battery_css_dr = min_nsar_avail -
battery_css_dr(iter_hour_css_dr+1);
            if deficit_battery_css_dr <= steady_power_draw
                battery_css_dr(iter_hour_css_dr+1) = min_nsar_avail;
                battery_grid_consum(iter_hour_css_dr) = deficit_battery_css_dr;
            else
                battery_css_dr(iter_hour_css_dr+1) = steady_power_draw;
                battery_grid_consum(iter_hour_css_dr) = steady_power_draw;
            end
        %there is no production from solar panel and there is no DR event
        else
            battery_css_dr = less_prod_eval_dr(current_load, steady_power_draw,
battery_css_dr,iter_hour_css_dr,min_nsar_avail, batt_eff);
        end
    end
end

%Calculating simple payback
%Savings
total_solar_prod_css_dr = sum(solar_prod_css_dr); %total amount of solar used
total_solar_waste_css_dr = sum(solar_waste_css_dr); %total amount of solar
wasted
percent_wasted_css_dr = (total_solar_waste_css_dr/(sum(solar_vec(:,2))))*100;

net_load_vec_dr(:,1) = res_load(:,1);
net_load_vec_dr(:,2) = res_load(:,2) - solar_prod_css_dr + battery_grid_consum;

[~, month_solar_bill_dr, month_solar_prod_dr] =
css_schedule_r(net_load_vec_dr);

%FFR and NSAR Incentives
total_monthly_incentive = capab_incentive_ffr + capab_incentive_nsar; %money
received from participating in FFR and NSAR activities

for month = 1:length(month_solar_bill_dr)
    temp_month_bill = month_solar_bill_dr(month) - total_monthly_incentive;
    if temp_month_bill < minimum_charge_css_dr
        month_solar_bill_dr(month) = minimum_charge_css_dr + gif_fee;
    else
```

```
        month_solar_bill_dr(month) = temp_month_bill;
    end
end
total_solar_bill_dr = sum(month_solar_bill_dr);
solar_savings_dr = total_resi_bill - total_solar_bill_dr; %savings per month

%Simple Payback
simple_payback_css_dr = (install_cost + total_batt_cost)/(solar_savings_dr);

%Writing Residential Load Consumption
xlswrite(file_name,{'Timestamp'},sheet_name,'A1');
xlswrite(file_name,date_str, sheet_name,'A2');
xlswrite(file_name,{'Consumption(kWh)'},sheet_name,'B1');
xlswrite(file_name,res_load(:,2),sheet_name,'B2');

%Writing Solar Production & Waste
xlswrite(file_name, {'State of Charge (kWh)'},sheet_name, 'C1');
xlswrite(file_name, battery_css_dr, sheet_name, 'C2');
xlswrite(file_name, {'Solar Energy Used (kWh)'}, sheet_name, 'D1');
xlswrite(file_name, solar_prod_css_dr, sheet_name, 'D2');
xlswrite(file_name, {'Solar Energy Wasted (kWh)'}, sheet_name, 'E1');
xlswrite(file_name, solar_waste_css_dr, sheet_name, 'E2');

%Writing DR Events
xlswrite(file_name,{'FFR Event Magnitude (kWh)'}, sheet_name, 'F1');
xlswrite(file_name, ffr_tracker_amt, sheet_name, 'F2');
xlswrite(file_name, {'NSAR Event Magnitude (kWh)'}, sheet_name, 'G1');
xlswrite(file_name, nsar_tracker_amt, sheet_name, 'G2');

%Writing Monthly Consumption & Bill
xlswrite(file_name,{'Month'},sheet_name, 'I1');
xlswrite(file_name, output_month, sheet_name, 'I2');
xlswrite(file_name, {'Monthly Consumption (kWh)'}, sheet_name, 'J1');
xlswrite(file_name, month_solar_prod_dr', sheet_name, 'J2');
xlswrite(file_name, {'Monthly Bill ($/mo)'}, sheet_name, 'K1');
xlswrite(file_name, month_solar_bill_dr, sheet_name, 'K2');

%% FLAT RATE, W/ SOLAR PANEL, CGS, NO DR
cgs_credit = 15.067/100; %in dollars per kWh
minimum_charge = 25; %in dollars
gif_charge = 1.27; %in dollars/month

%Output Sheet
sheet_name = 'CGS Only';
```

```
[total_cgs_credit, cgs_monthly_export, cgs_monthly_credit, net_load_vec,
net_total_resi_bill, net_month_resi_bill, net_month_resi_consum,
cgs_monthly_bill] = cgs_monthly(solar_vec, res_load, cgs_credit,
minimum_charge, gif_charge);

cgs_total_bill   = sum(cgs_monthly_bill);
cgs_savings = total_resi_bill - cgs_total_bill;
simple_payback_cgs = install_cost/(cgs_savings);

%Writing Residential Load Consumption
xlswrite(file_name,{'Timestamp'},sheet_name,'A1');
xlswrite(file_name,date_str, sheet_name,'A2');
xlswrite(file_name,{'Consumption(kWh)'},sheet_name,'B1');
xlswrite(file_name,res_load(:,2),sheet_name,'B2');

%Writing Monthly Consumption & Bill
xlswrite(file_name,{'Month'},sheet_name, 'D1');
xlswrite(file_name, output_month, sheet_name, 'D2');
xlswrite(file_name, {'Solar Export (kWh)'}, sheet_name, 'E1');
xlswrite(file_name, cgs_monthly_export', sheet_name, 'E2');
xlswrite(file_name, {'Net Consumption (kWh/mo)'}, sheet_name, 'F1');
xlswrite(file_name, net_month_resi_consum', sheet_name, 'F2');
xlswrite(file_name, {'Monthly Credit ($/mo)'}, sheet_name, 'G1');
xlswrite(file_name, cgs_monthly_credit, sheet_name, 'G2');
xlswrite(file_name, {'Net Bill After Credit ($/mo)'},sheet_name, 'H1');
xlswrite(file_name, cgs_monthly_bill, sheet_name, 'H2');


%% FLAT RATE, W SOLAR PANEL, CGS, DR

%OPERATIONALLY, THIS IS THE SAME BATTERY LEVELS AND INTERACTION
AS DR DAVE
%OPERATIONALLY, FOR THE SOLAR PANEL, SAME AS THE ABOVE CASE

total_savings_cgs_dr =  cgs_savings+ (total_monthly_incentive *12);
simple_payback_cgs_dr = (total_batt_cost + install_cost)/total_savings_cgs_dr;

%% FLAT RATE, W SOLAR PANEL, CGS, DR , INTEGRATED
% Solar panel is first used to offset immediate load, then fill battery and
% then export

%Intialization
cgs_dr_tracker    = zeros(length(res_load),1);
battery_cgs_dr = zeros(length(res_load),1);
battery_cgs_dr(1) = storage_capacity; %assuming that the battery starts the
year fully charged
```

```
solar_prod_cgs_dr = zeros(length(res_load),1);
solar_export_cgs_dr = zeros(length(res_load),1);
battery_grid_consum_cgs = zeros(length(res_load),1);
ffr_count_cgs_dr = 1;
nsar_count_cgs_dr = 1;

%DR Tracking
ffr_tracker = zeros(length(res_load),1);
ffr_tracker_amt = zeros(length(res_load),1);
nsar_tracker = zeros(length(res_load),1);
nsar_tracker_amt = zeros(length(res_load),1);

%Charges
minimum_charge_cgs_dr = 25; %in $/month

%Output sheet
sheet_name = 'CGS + DR Integrated';

for iter_hour_cgs_dr = 1:length(res_load)-1

    dr_min_satisfied = false;
    dr_min_not_satisfied = false;

    current_timestamp = res_load(iter_hour_cgs_dr,1);
    next_timestamp = res_load(iter_hour_cgs_dr+1,1);
    current_battery_level = battery_cgs_dr(iter_hour_cgs_dr);

    current_load = res_load(iter_hour_cgs_dr,2); %the load in current iteration
    current_prod = solar_vec(iter_hour_cgs_dr,2); %the production in current
iteration
    excess_prod = current_prod - current_load; %How much did the solar panel
produce in excess of the load

    %Checking if FFR event has taken place
    if current_timestamp < ffr_events(ffr_count_cgs_dr,1) && next_timestamp >
ffr_events(ffr_count_cgs_dr,1)
        ffr_tracker(iter_hour_cgs_dr) = true;
        ffr_tracker_amt(iter_hour_cgs_dr) =
(ffr_events(ffr_count_cgs_dr,2)/batt_eff);
        battery_cgs_dr(iter_hour_cgs_dr+1) = battery_cgs_dr(iter_hour_cgs_dr) -
ffr_tracker_amt(iter_hour_cgs_dr); %Assuming that the average FFR event lasts
10 seconds

        if battery_cgs_dr(iter_hour_cgs_dr+1) > min_nsar_avail
            dr_min_satisfied = true;
        else
```

```
            dr_min_not_satisfied = true;
        end

        if ffr_count_cgs_dr < length(ffr_events)
            ffr_count_cgs_dr = ffr_count_cgs_dr + 1;
        end

    %Check if NSAR event has taken place
    elseif current_timestamp < nsar_events(nsar_count_cgs_dr,1) &&
next_timestamp > nsar_events(nsar_count_cgs_dr,1)
        nsar_tracker(iter_hour_cgs_dr) = true;
        nsar_tracker_amt(iter_hour_cgs_dr) =
(nsar_events(nsar_count_cgs_dr,2)/batt_eff);
        battery_cgs_dr(iter_hour_cgs_dr+1) = battery_cgs_dr(iter_hour_cgs_dr) -
nsar_tracker_amt(iter_hour_cgs_dr); %Subtracting the NSAR requirement from
the battery

        if battery_cgs_dr(iter_hour_cgs_dr+1) > min_nsar_avail
            dr_min_satisfied = true;
        else
            dr_min_not_satisfied = true;
        end

        if nsar_count_cgs_dr < length(nsar_events)
            nsar_count_cgs_dr = nsar_count_cgs_dr + 1;
        end
    end


    %if there is excess solar production vs. load
    if excess_prod >= 0
        %If DR event happens and battery level is above minimum required

        if dr_min_satisfied %there is enough in the battery to meet DR response
            solar_prod_cgs_dr(iter_hour_cgs_dr) =
solar_prod_cgs_dr(iter_hour_cgs_dr) + current_load; %tracking how much solar
is used and for what
            battery_space_cgs_dr = storage_capacity -
battery_cgs_dr(iter_hour_cgs_dr+1); %Total capacity - battery level in this hour;
            [solar_prod_cgs_dr, solar_export_cgs_dr, battery_cgs_dr] =
excess_prod_cgs_dr(battery_cgs_dr,battery_space_cgs_dr, steady_power_draw,
excess_prod, solar_prod_cgs_dr, solar_export_cgs_dr, iter_hour_cgs_dr);

        %If DR event happens and Battery Level falls below minimum required
        elseif dr_min_not_satisfied
```

```
            deficit_battery_cgs_dr = min_nsar_avail -
battery_cgs_dr(iter_hour_cgs_dr+1); %how much is lacking for battery to meet
minimum requirement

            if current_prod > deficit_battery_cgs_dr %if solar production can meet
deficit in minimum battery requirement
                solar_prod_cgs_dr(iter_hour_cgs_dr) = deficit_battery_cgs_dr; %solar
panel's energy used to let battery reach minimum
                battery_cgs_dr(iter_hour_cgs_dr+1)  = min_nsar_avail; %battery is
now at minimum level for next hour
                leftover_solar_cgs_dr               = current_prod -
deficit_battery_cgs_dr;
            else
                battery_cgs_dr(iter_hour_cgs_dr+1)  = min_nsar_avail; %the battery
will import energy from the grid to meet the minimum requirement
                battery_grid_consum_cgs(iter_hour_cgs_dr) = deficit_battery_cgs_dr;
%recording consumption from grid
                leftover_solar_cgs_dr               = current_prod;
            end
            %now that minimum battery level is fulfilled, do the rest
            if leftover_solar_cgs_dr > current_load
                solar_prod_cgs_dr(iter_hour_cgs_dr) =
solar_prod_cgs_dr(iter_hour_cgs_dr) + current_load;
                leftover_solar_cgs_dr = leftover_solar_cgs_dr - current_load;
%updating how much solar is left over
                battery_space_cgs_dr = storage_capacity -
battery_cgs_dr(iter_hour_cgs_dr+1); %Total capacity - battery level in this hour;
                [solar_prod_cgs_dr, solar_export_cgs_dr, battery_cgs_dr] =
excess_prod_cgs_dr(battery_cgs_dr, battery_space_cgs_dr, steady_power_draw,
leftover_solar_cgs_dr, solar_prod_cgs_dr, solar_export_cgs_dr,
iter_hour_cgs_dr);
            else
                solar_prod_cgs_dr(iter_hour_cgs_dr) =
solar_prod_cgs_dr(iter_hour_cgs_dr) + leftover_solar_cgs_dr; %if the left-over
solar is not enough to meet current load
            end
        %if there is no DR event at all
        else
            battery_space_cgs_dr = storage_capacity -
battery_cgs_dr(iter_hour_cgs_dr);
            solar_prod_cgs_dr(iter_hour_cgs_dr) = current_load;
            [solar_prod_cgs_dr, solar_export_cgs_dr, battery_cgs_dr] =
excess_prod_eval(battery_cgs_dr, battery_space_cgs_dr, steady_power_draw,
excess_prod, solar_prod_cgs_dr, solar_export_cgs_dr, iter_hour_cgs_dr);
        end
```

```
    %If the panel produces less energy than the load needs
    elseif excess_prod < 0 && current_prod > 0
        %If DR event happens and battery level is above minimum required
        if dr_min_satisfied
            solar_prod_cgs_dr(iter_hour_cgs_dr) = current_prod; %if production <
load and battery minimum is maintained, current solar production is used
            current_load = current_load - current_prod; %if battery minimum is
met, solar used to displace load

            battery_cgs_dr = less_prod_eval_dr(current_load, steady_power_draw,
battery_cgs_dr,iter_hour_cgs_dr,min_nsar_avail,batt_eff);

        %If DR event happens and battery level below minimum required
        elseif dr_min_not_satisfied
            deficit_battery_cgs_dr = min_nsar_avail -
battery_cgs_dr(iter_hour_cgs_dr+1); %how much is lacking for battery to meet
minimum requirement
            if current_prod > deficit_battery_cgs_dr %if solar production can meet
deficit in minimum battery requirement
                solar_prod_cgs_dr(iter_hour_cgs_dr) = current_prod; %solar panel's
energy used to let battery reach minimum
                battery_cgs_dr(iter_hour_cgs_dr+1)  =
battery_cgs_dr(iter_hour_cgs_dr) + current_prod; %battery is now at minimum
level for next hour
                %leftover_solar_cgs_dr               = current_prod -
deficit_battery_cgs_dr;
            else
                deficit_battery_cgs_dr              = deficit_battery_cgs_dr -
current_prod;
                battery_grid_consum_cgs(iter_hour_cgs_dr) = deficit_battery_cgs_dr;
                battery_cgs_dr(iter_hour_cgs_dr+1)  =
battery_cgs_dr(iter_hour_cgs_dr) + deficit_battery_cgs_dr + current_prod; %the
battery will import energy from the grid to meet the minimum requirement
                %leftover_solar_cgs_dr               = 0;
            end
        %If DR event has NOT taken place BUT production < load
        else
            solar_prod_cgs_dr(iter_hour_cgs_dr) = current_prod; %if production <
load and battery minimum is maintained, current solar production is used
            current_load = current_load - current_prod; %if battery minimum is
met, solar used to displace load

            battery_cgs_dr = less_prod_eval_dr(current_load, steady_power_draw,
battery_cgs_dr,iter_hour_cgs_dr,min_nsar_avail,batt_eff);
        end
```

```
    %When there is no production from the solar panel, only load
    else
        if dr_min_satisfied %if there is a minimum in the battery after the DR
event
            battery_cgs_dr = less_prod_eval_dr(current_load, steady_power_draw,
battery_cgs_dr,iter_hour_cgs_dr,min_nsar_avail, batt_eff);
        elseif dr_min_not_satisfied %if the battery falls below a minimum after the
DR event
            %will be importing electricity from grid here
            deficit_battery_cgs_dr = min_nsar_avail -
battery_cgs_dr(iter_hour_cgs_dr+1);
            if deficit_battery_cgs_dr <= steady_power_draw
                battery_cgs_dr(iter_hour_cgs_dr+1) = min_nsar_avail;
                battery_grid_consum_cgs(iter_hour_cgs_dr) = deficit_battery_cgs_dr;
            else
                battery_cgs_dr(iter_hour_cgs_dr+1) = steady_power_draw;
                battery_grid_consum_cgs(iter_hour_cgs_dr) = steady_power_draw;
            end
        %there is no production from solar panel and there is no DR event
        else
            battery_cgs_dr = less_prod_eval_dr(current_load, steady_power_draw,
battery_cgs_dr,iter_hour_cgs_dr,min_nsar_avail, batt_eff);
        end
    end
end

%Calculating simple payback
%Savings
total_solar_prod_cgs_dr = sum(solar_prod_cgs_dr); %total amount of solar used
total_solar_export_cgs_dr = sum(solar_export_cgs_dr); %total amount of solar
wasted
net_load_vec_dr(:,1) = res_load(:,1);
net_load_vec_dr(:,2) = res_load(:,2) - solar_prod_cgs_dr +
battery_grid_consum_cgs;
%net_load_dr = res_load_total - total_solar_prod_cgs_dr;

[~, month_cgs_bill_dr, month_cgs_dr_consum, month_cgs_dr_export] =
cgs_schedule_r(net_load_vec_dr, solar_export_cgs_dr);

%FFR and NSAR Incentives
total_monthly_incentive = capab_incentive_ffr + capab_incentive_nsar; %money
received from participating in FFR and NSAR activities

for month_cgs = 1:length(month_cgs_bill_dr)
    temp_month_bill = month_cgs_bill_dr(month_cgs) - total_monthly_incentive;
    if temp_month_bill < minimum_charge_cgs_dr
```

```
        month_cgs_bill_dr(month_cgs) = minimum_charge_cgs_dr + gif_fee;
    else
        month_cgs_bill_dr(month_cgs) = temp_month_bill;
    end
end

total_cgs_bill_dr = sum(month_cgs_bill_dr);
cgs_savings_dr = total_resi_bill - total_cgs_bill_dr; %savings per month

%Simple Payback
simple_payback_cgs_dr = (install_cost + total_batt_cost)/(cgs_savings_dr);

%Writing Residential Load Consumption
xlswrite(file_name,{'Timestamp'},sheet_name,'A1');
xlswrite(file_name,date_str, sheet_name,'A2');
xlswrite(file_name,{'Consumption(kWh)'},sheet_name,'B1');
xlswrite(file_name,res_load(:,2),sheet_name,'B2');

%Writing Monthly Consumption & Bill
xlswrite(file_name,{'Month'},sheet_name, 'D1');
xlswrite(file_name, output_month, sheet_name, 'D2');
xlswrite(file_name, {'Solar Export (kWh)'}, sheet_name, 'E1');
xlswrite(file_name, month_cgs_dr_export', sheet_name, 'E2');
xlswrite(file_name, {'Consumption (kWh)'}, sheet_name, 'F1');
xlswrite(file_name, month_cgs_dr_consum', sheet_name, 'F2');
xlswrite(file_name, {'Net Bill ($/mo)'}, sheet_name, 'G1');
xlswrite(file_name, month_cgs_bill_dr, sheet_name, 'G2');
```

Exhibit A2 – Code for all Schedule TOU Programs

```
%% GET RESIDENTIAL LOAD PROFILE
[num, ~, raw] = xlsread('Updated HI Residential Data.xlsx', 'Raw Data');
[~,~,raw1] = xlsread('Residential_Honolulu_Base.xlsx');
TwoM_vec  =  [num(70081:105120,6), num(70081:105120,1)];
%ThreeL_vec = [num(1:35040,6), num(1:35040,1)];

res_load   = spreadsheet_modify_fcn(TwoM_vec);
date_str   = cell(length(raw1)-1,1);

%Extracting data on residential loads
for iter = 2:length(raw1)
    res_load(iter,1) = datenum(raw1{iter,1}); %recording timestamp for load data
    date_str   = cell(length(raw1)-1,1);
end
```

```
res_load(1,:) = [];
date_str(1,:) = [];

%% Output file name
file_name = 'All Results TOU.xlsx';
output_month =
        {'January';'February';'March';'April';'May';'June';'July';'August';'Sept
        ember';'October';'November';'December'};

%% TESLA POWERWALL 2 SPECIFICATIONS & COSTS
steady_power_draw = 5; %in kW
max_power_draw    = 7; %in kW, lasts 10 seconds
batt_eff      = 0.89;
storage_capacity = 13.2; %in kWh
depth_of_discharge = 1; %can discharge 100% of capacity
total_batt_cost = 7600; %in dollars, includes battery, install costs and
        supporting hardware

%% FFR + NSAR SPECIFICATIONS
%Assumptions -  FFR 2 (responding within 30 cycles) and NSAR; x

%Residential Incentives - FFR
capab_incentive_ffr = 8 * steady_power_draw; %in $/month

%Residential Incentives - NSAR
capab_incentive_nsar = 6 * steady_power_draw; %in $/month

%Constraints - FFR
min_ffr_avail = 0.5 * steady_power_draw; %in kWhr, 30-min minimum
        availability for FFR events at nominated capability

%Assumptions - FFR
avg_ffr_duration = 30/60; %in hr, assuming the average FFR event lasts 10
        seconds

%Constraints - NSAR
min_nsar_avail = 1.0 * steady_power_draw; %in kWhr, 1 hour minimum
        availability for NSAR events at nominated capability
max_nsar_time = 100; %hours/year that NSAR can be deployed

%Hard-coding FFR events
num_events_ffr = 60; %in one year
ffr_events = zeros(num_events_ffr,2);
ffr_events(:,1) =
        [736707.603072136;736708.555595945;736709.016906427;7367
        12.829471351;736731.404533136;736731.553476493;736739.37
```

```
4653551;736742.286693069;736747.717568787;736753.449966447;736755.271975596;736758.397545726;736764.115062197;736777.579449354;736796.611137371;736796.938424813;736797.512597554;736811.582756812;736820.070593430;736835.078053194;736838.966744081;736849.731982636;736855.922314017;736858.416165002;736872.919418342;736874.519116596;736877.653691179;736895.338307421;736909.330088458;736926.494937930;736931.581084705;736934.783214866;736934.921683383;736935.017477253;736943.398956090;736943.751617846;736949.265025033;736953.353790911;736954.563472637;736966.871776030;736971.083286848;736972.196268826;736975.030860420;736984.759562455;736985.850354761;736987.702221647;736992.966774261;736996.150368895;737005.507622161;737026.161150580;737028.925488991;737029.785588333;737036.440527997;737042.355925804;737044.887341556;737045.011230434;737045.734978315;737045.826661771;737047.701859912;737049.781057718];
ffr_events(:,2) =
[0.0775828933215146;0.033738820751414 8;0.0664722381326890;0.0662311871026733;0.0403087393732887;0.0533209472725848;0.0191565478863239;0.0176354249074033;0.0507498300700675;0.0679977243126397;0.0787507419603599;0.0229101533662313;0.0533905320050134;0.0464854611845976;0.0147154214931418;0.0373001836388112;0.0251515491800863;0.0690475375474935;0.0355010445864448;0.0505925788545981;0.0253922728819292;0.0556931903751137;0.0321507836486211;0.0593110485053321;0.0617510071625005;0.0658438606127576;0.0451764998960068;0.0197098179164536;0.0297900672720013;0.0773150945487271;0.0244706957617516;0.0712372901034408;0.0512737802263929;0.0830649108768670;0.0193177450523044;0.0446304354010727;0.0212953312625406;0.0806873667260454;0.0142107100093102;0.0677021156049655;0.0706460569898217;0.0742149100946882;0.0197524892715910;0.0416515728540900;0.0319354446424065;0.0694492000155769;0.0438481824627462;0.0771283051687169;0.0265171547432536;0.0322085358695826;0.0239957625267165;0.0233380943547683;0.0742564033083395;0.0541461519003868;0.0520736251275231;0.0239551943210921;0.0731271609529093;0.0570871619086851;0.0382605820064077;0.0495312180463232];

%Hard-coding NSAR events
num_events_nsar = 24; %in one year
nsar_events = zeros(num_events_nsar,2);
nsar_events(:,1) =
[736746.528607807;736750.417663298;736767.658966507;736784.764848282;736787.520063326;736788.685851292;736788.98
```

```
                2166632;736789.861755648;736823.569078651;736824.1798862
                54;736868.513779947;736880.421339476;736895.460054338;73
                6896.374248860;736909.328754451;736920.548277325;736950.
                813457317;736969.836834422;736992.806809677;736998.83702
                5252;737002.441430075;737020.734225238;737034.716579896;
                737045.661713384];
nsar_events(:,2) =
                [2.50753347396642;1.14986121537851;1.83298397314024;1.347
                16222847986;1.59961578451007;1.83313552360376;2.57194612
                118487;1.04022679302393;4.59465045798034;4.7699466238401
                9;2.87860038528367;2.87188599333341;2.24049754092241;4.58
                355769340693;2.37186158800090;1.29667814705745;4.0843836
                1800474;2.45724515400522;1.84038035797430;2.516300606617
                14;1.23522718820162;1.38322205252640;4.75854412823119;4.8
                1722725095751];

%Check for max 24-hour deployment in FFR & NSAR event vectors
%For FFR
if length(ffr_events) > 2
    bad_ffr_event = dr_24hr_check(ffr_events);
    if ~isempty(bad_ffr_event)
        ffr_events(bad_ffr_event,:) = []; %deleting the bad FFR events
    end
end

%For NSAR
if length(nsar_events) > 2
    bad_nsar_event = dr_24hr_check(nsar_events);
    if ~isempty(bad_nsar_event)
        nsar_events(bad_nsar_event,:) = []; %deleting the bad NSAR events
    end
end

%% GETTING SOLAR DATA
%Assumptions
panel_eff = 0.162;
panel_size = 1.61; %in m2
num_panels = 20; %For a 5 kW system

[~,~,raw_solar] = xlsread('Honolulu_Solar.xlsx');
solar_vec = zeros(length(raw_solar)-2,2);

%extracting solar data into a workable form
for iter_solar = 3:length(raw_solar)
    temp_date = datevec(raw_solar{iter_solar,1});
    temp_date(1) = 2017;
```

```
    if raw_solar{iter_solar,3} == 24
        temp_date(4) = 0;
    else
        temp_date(4) = raw_solar{iter_solar,3};
    end
    temp_ghi = raw_solar{iter_solar,6}; %in Wh/m2, total horizontal radiation
                (diffuse + direct)
    panel_output = (temp_ghi*panel_size*panel_eff*num_panels)/1000; %in
                kWh, electricity produced by average solar panel in that hour of
                the year

    solar_vec(iter_solar-2,1) = datenum(temp_date);
    solar_vec(iter_solar-2,2) = panel_output;
end

solar_vec(end,1) = datenum([2018 1 1 0 0 0]);

%% SOLAR COST

%install_cost = 7006; %For 2.5 kW system
install_cost = 10441; %For 5 kW system

%% CONVENTIONAL CHRIS BILL
conventional_chris_bill = 1960.4; %in $

%% TOU, NO DEMAND SERVICES, NO BATTERY
%Assumptions - No arbitrage, No Demand Services,
%Schedule TOU-RI, Honolulu, Single Phase, Only January
[tou_monthly_bill, on_peak_base, off_peak_base, midday_peak_base] =
                schedule_tou(res_load); %Calculating the TOU bill for no
tou_bill = sum(tou_monthly_bill(:,4)); %this is the total Annual TOU Charge

%Output Sheet
sheet_name = 'TOU Tim';

%Writing Residential Load Consumption, by three periods
xlswrite(file_name,{'Timestamp'},sheet_name,'A1');
xlswrite(file_name,date_str, sheet_name,'A2');
xlswrite(file_name, {'On Peak Load(kWh)'},sheet_name,'B1');
xlswrite(file_name, on_peak_base, sheet_name, 'B2');
xlswrite(file_name, {'Midday Peak Load (kWh)'}, sheet_name, 'C1');
xlswrite(file_name, midday_peak_base, sheet_name, 'C2');
xlswrite(file_name, {'Off Peak Load (kWh)'}, sheet_name, 'D1');
xlswrite(file_name, off_peak_base, sheet_name, 'D2');

%Writing Monthly Bills
```

```
xlswrite(file_name, {'Month'}, sheet_name, 'E1');
xlswrite(file_name, output_month, sheet_name, 'E2');
xlswrite(file_name, {'On Peak Bill'}, sheet_name, 'F1');
xlswrite(file_name, tou_monthly_bill(:,1), sheet_name, 'F2');
xlswrite(file_name, {'Midday Peak Bill'}, sheet_name, 'G1');
xlswrite(file_name, tou_monthly_bill(:,2), sheet_name, 'G2');
xlswrite(file_name, {'Off Peak Bill'}, sheet_name, 'H1');
xlswrite(file_name, tou_monthly_bill(:,3), sheet_name, 'H2');
xlswrite(file_name, {'Total Monthly Bill'}, sheet_name, 'I1');
xlswrite(file_name, tou_monthly_bill(:,4), sheet_name, 'I2');

%% TOU, ARBITRAGE W/ BATTERY, NO DEMAND SERVICES
% Assumptions - COMPLETE ASSUMPTIONS HERE
customer_charge = 9.0; %in $/month, assumed single-phase
minimum_charge  = 17.0; %in $/month, assumed single-phase
surcharge_fee = 4.27/100; %in $/kWh
gif_fee       = 1.27; %in $/mo

%Time-of-Use Charges
on_peak_charge       = 35.1826/100; %in $/kWhr
mid_peak_charge      = 12.807/100; %in $/kWhr
off_peak_charge      = 21.581/100; %in $/kWhr

%Usage Charges
first_tier_charge = 1.1535/100; %in $,for between 350 and 1200 kWh/month
second_tier_charge = 3.0309/100; %in $,anything over 1200 kWh/month

%Initialization
battery_state = zeros(length(res_load),1);
battery_state(1) = storage_capacity;

%The three numbers below will track amount of consumption (from grid) in
%each pricing period
%Vector Initialization
on_peak_vec = zeros(length(res_load),1);
on_peak_vec_charge = zeros(length(res_load),1);
midday_peak_vec   = zeros(length(res_load),1);
midday_peak_vec_charge = zeros(length(res_load),1);
off_peak_vec      = zeros(length(res_load),1);
off_peak_vec_charge = zeros(length(res_load),1);

%Output Sheet
sheet_name = 'Arbitrage Amy';

%Iterating through the load profile
for i = 1:length(res_load)-1
```

```
    current_time = res_load(i,1); %time of current iteration, in datenum
    current_time_vec = datevec(current_time);
    current_load = res_load(i,2); %load of current iteration, in kWh

[on_peak_vec, on_peak_vec_charge, midday_peak_vec,midday_peak_vec_charge,
                off_peak_vec, off_peak_vec_charge, battery_state] =
                tou_arbitrage(false, i,current_load, current_time_vec,
                battery_state,storage_capacity, steady_power_draw,
                min_nsar_avail, batt_eff,on_peak_vec, on_peak_vec_charge,
                midday_peak_vec,midday_peak_vec_charge, off_peak_vec,
                off_peak_vec_charge);
end

[tou_monthly_bill_arbitrage] = tou_bill_aggregation(res_load,
                on_peak_vec_charge, midday_peak_vec_charge,
                off_peak_vec_charge, gif_fee, customer_charge, minimum_charge);
tou_bill_batt = sum(tou_monthly_bill_arbitrage(:,4));
tou_savings = conventional_chris_bill - tou_bill_batt;

%Simple Payback Calcualtion
simple_payback = total_batt_cost/tou_savings;

%Writing Residential Load Consumption, by three periods
xlswrite(file_name,{'Timestamp'},sheet_name,'A1');
xlswrite(file_name,date_str, sheet_name,'A2');
xlswrite(file_name, {'On Peak Load(kWh)'},sheet_name,'B1');
xlswrite(file_name, on_peak_vec, sheet_name, 'B2');
xlswrite(file_name, {'Midday Peak Load (kWh)'}, sheet_name, 'C1');
xlswrite(file_name, midday_peak_vec, sheet_name, 'C2');
xlswrite(file_name, {'Off Peak Load (kWh)'}, sheet_name, 'D1');
xlswrite(file_name, off_peak_vec, sheet_name, 'D2');
xlswrite(file_name, {'State of Charge (kWh)'},sheet_name, 'E1');
xlswrite(file_name, battery_state, sheet_name, 'E2');

%Writing Monthly Numbers
xlswrite(file_name, {'Month'}, sheet_name, 'F1');
xlswrite(file_name, output_month, sheet_name, 'F2');
xlswrite(file_name, {'On Peak Bill'}, sheet_name, 'G1');
xlswrite(file_name, tou_monthly_bill_arbitrage(:,1), sheet_name, 'G2');
xlswrite(file_name, {'Midday Peak Bill'}, sheet_name, 'H1');
xlswrite(file_name, tou_monthly_bill_arbitrage(:,2), sheet_name, 'H2');
xlswrite(file_name, {'Off Peak Bill'}, sheet_name, 'I1');
xlswrite(file_name, tou_monthly_bill_arbitrage(:,3), sheet_name, 'I2');
xlswrite(file_name, {'Total Monthly Bill'}, sheet_name, 'J1');
xlswrite(file_name, tou_monthly_bill_arbitrage(:,4), sheet_name, 'J2');
```

```
%% TOU, ARBITRAGE W/ BATTERY, DEMAND SERVICES
% Assumptions -

%Other Charges
customer_charge = 9.0; %in $/month, assumed single-phase
minimum_charge_tou_dr  = 17.0; %in $/month, assumed single-phase
surcharge_fee   = 4.27/100; %in $/kWh
gif_fee         = 1.27; %in $/mo

%Time-of-Use Charges
on_peak_charge       = 35.1826/100; %in $/kWhr
mid_peak_charge      = 12.807/100; %in $/kWhr
off_peak_charge      = 21.581/100; %in $/kWhr

%Initialization
ffr_count_tou_dr = 1;
nsar_count_tou_dr = 1;
battery_tou_dr = zeros(length(res_load),1);
battery_tou_dr(1) = storage_capacity;
on_peak_vec_dr = zeros(length(res_load),1);
on_peak_vec_charge_dr = zeros(length(res_load),1);
off_peak_vec_dr = zeros(length(res_load),1);
off_peak_vec_charge_dr = zeros(length(res_load),1);
midday_peak_vec_dr = zeros(length(res_load),1);
midday_peak_vec_charge_dr = zeros(length(res_load),1);

%DR Tracking
ffr_tracker = zeros(length(res_load),1);
ffr_tracker_amt = zeros(length(res_load),1);
nsar_tracker = zeros(length(res_load),1);
nsar_tracker_amt = zeros(length(res_load),1);

%Output Sheet
sheet_name = 'Amy w DR';

%Iterating through each hour in the load profile
for iter_hour_tou_dr = 1:length(res_load)-1

    dr_min_satisfied = false; %initializing
    dr_min_not_satisfied = false; %initializing

    current_timestamp = res_load(iter_hour_tou_dr,1); %current time of load
                profile
    current_time_vec = datevec(current_timestamp);
    next_timestamp = res_load(iter_hour_tou_dr+1,1);
```

```
current_battery_level = battery_tou_dr(iter_hour_tou_dr); %battery level at
        current time of load profile

current_load = res_load(iter_hour_tou_dr,2); %the load in current iteration

%Checking if FFR event has taken place
if current_timestamp < ffr_events(ffr_count_tou_dr,1) && next_timestamp >
        ffr_events(ffr_count_tou_dr,1)
   ffr_tracker(iter_hour_tou_dr) = true;
   ffr_tracker_amt(iter_hour_tou_dr) =
        (ffr_events(ffr_count_tou_dr,2)/batt_eff);
   battery_tou_dr(iter_hour_tou_dr+1) = battery_tou_dr(iter_hour_tou_dr) -
        ffr_tracker_amt(iter_hour_tou_dr); %Assuming that the average
        FFR event lasts 10 seconds


   if battery_tou_dr(iter_hour_tou_dr+1) > min_nsar_avail
      dr_min_satisfied = true;
   else
      dr_min_not_satisfied = true;
   end

   if ffr_count_tou_dr < length(ffr_events)
      ffr_count_tou_dr = ffr_count_tou_dr + 1;
   end

%Check if NSAR event has taken place
elseif current_timestamp < nsar_events(nsar_count_tou_dr,1) &&
        next_timestamp > nsar_events(nsar_count_tou_dr,1)
   nsar_tracker(iter_hour_tou_dr) = true;
   nsar_tracker_amt(iter_hour_tou_dr) =
        (nsar_events(nsar_count_tou_dr,2)/batt_eff);
   battery_tou_dr(iter_hour_tou_dr+1) = battery_tou_dr(iter_hour_tou_dr) -
        nsar_tracker_amt(iter_hour_tou_dr); %Subtracting the NSAR
        requirement from the battery

   if battery_tou_dr(iter_hour_tou_dr+1) > min_nsar_avail
      dr_min_satisfied = true;
   else
      dr_min_not_satisfied = true;
   end

   if nsar_count_tou_dr < length(nsar_events)
      nsar_count_tou_dr = nsar_count_tou_dr + 1;
   end
end
```

```
if dr_min_not_satisfied %if it did go below the minimum
    deficit_min_req = min_nsar_avail - battery_tou_dr(iter_hour_tou_dr+1);
else
    deficit_min_req = min_nsar_avail - battery_tou_dr(iter_hour_tou_dr);
end

%DR event took place but minimum battery threshold is maintained
if dr_min_satisfied
    [on_peak_vec_dr, ~, midday_peak_vec_dr,~, off_peak_vec_dr,~,
            battery_tou_dr] = tou_arbitrage(true,
            iter_hour_tou_dr,current_load, current_time_vec,
            battery_tou_dr,storage_capacity, steady_power_draw,
            min_nsar_avail, batt_eff,on_peak_vec_dr, on_peak_vec_charge_dr,
            midday_peak_vec_dr,midday_peak_vec_charge_dr, off_peak_vec_dr,
            off_peak_vec_charge_dr);

%DR event took place but minimum battery threshold NOT maintain
elseif dr_min_not_satisfied
    %you want to see which  period you're in
    %Checking for On-Peak Pricing
    if any(current_time_vec(4) == 17:21)
        if deficit_min_req > steady_power_draw
            battery_tou_dr(iter_hour_tou_dr+1) =
              battery_tou_dr(iter_hour_tou_dr+1) + steady_power_draw;
            on_peak_vec_dr(iter_hour_tou_dr) =
              on_peak_vec_dr(iter_hour_tou_dr) + steady_power_draw;
        else
            battery_tou_dr(iter_hour_tou_dr+1) =
              battery_tou_dr(iter_hour_tou_dr+1) + deficit_min_req;
            on_peak_vec_dr(iter_hour_tou_dr) =
              on_peak_vec_dr(iter_hour_tou_dr) + deficit_min_req;
        end
    %Checking for Mid-Day pricing
    elseif any(current_time_vec(4) == 9:16)
        if deficit_min_req > steady_power_draw
            battery_tou_dr(iter_hour_tou_dr+1) =
              battery_tou_dr(iter_hour_tou_dr+1) + steady_power_draw;
            midday_peak_vec_dr(iter_hour_tou_dr) =
              midday_peak_vec_dr(iter_hour_tou_dr) + steady_power_draw;
        else
            battery_tou_dr(iter_hour_tou_dr+1) =
              battery_tou_dr(iter_hour_tou_dr+1) + deficit_min_req;
            midday_peak_vec_dr(iter_hour_tou_dr) =
              midday_peak_vec_dr(iter_hour_tou_dr) + deficit_min_req;
        end
```

```
        %Checking for Off-Peak Pricing
        else
            if deficit_min_req > steady_power_draw
                battery_tou_dr(iter_hour_tou_dr+1) =
                  battery_tou_dr(iter_hour_tou_dr+1) + steady_power_draw;
                off_peak_vec_dr(iter_hour_tou_dr) =
                  off_peak_vec_dr(iter_hour_tou_dr) + steady_power_draw;
            else
                battery_tou_dr(iter_hour_tou_dr+1) =
                  battery_tou_dr(iter_hour_tou_dr+1) + deficit_min_req;
                off_peak_vec_dr(iter_hour_tou_dr) =
                  off_peak_vec_dr(iter_hour_tou_dr) + deficit_min_req;
            end
        end

    %DR event did NOT take place
    else
        [on_peak_vec_dr, ~, midday_peak_vec_dr,~, off_peak_vec_dr,~,
                battery_tou_dr] = tou_arbitrage(true,
                iter_hour_tou_dr,current_load, current_time_vec,
                battery_tou_dr,storage_capacity, steady_power_draw,
                min_nsar_avail, batt_eff,on_peak_vec_dr, on_peak_vec_charge_dr,
                midday_peak_vec_dr,midday_peak_vec_charge_dr, off_peak_vec_dr,
                off_peak_vec_charge_dr);

    end
end

for iter = 1:length(on_peak_vec_dr)
     on_peak_vec_charge_dr(iter) = on_peak_vec_dr(iter) * on_peak_charge +
                on_peak_vec_dr(iter) * surcharge_fee;
    midday_peak_vec_charge_dr(iter) = midday_peak_vec_dr(iter) *
                mid_peak_charge + midday_peak_vec_dr(iter) * surcharge_fee;
    off_peak_vec_charge_dr(iter) = off_peak_vec_dr(iter) * off_peak_charge +
                off_peak_vec_dr(iter) * surcharge_fee;
end

[tou_monthly_bill_dr] = tou_bill_aggregation(res_load, on_peak_vec_charge_dr,
                midday_peak_vec_charge_dr, off_peak_vec_charge_dr, gif_fee,
                customer_charge, minimum_charge_tou_dr);
total_dr_incentive = (capab_incentive_nsar + capab_incentive_ffr); %in $/yr
total_tou_bill_dr = tou_monthly_bill_dr(:,4);

for month = 1:length(tou_monthly_bill_dr)
    temp_month_bill = total_tou_bill_dr(month) - total_dr_incentive;
    if temp_month_bill < minimum_charge_tou_dr
```

```
        total_tou_bill_dr(month) = minimum_charge_tou_dr + gif_fee;
    else
        total_tou_bill_dr(month) = temp_month_bill;
    end
end

tou_bill_batt_dr = sum(total_tou_bill_dr);
tou_savings = conventional_chris_bill - tou_bill_batt_dr;

% Simple Payback
simple_payback_tou_dr = total_batt_cost/(tou_savings);

%Writing Hourly Data to Excel Sheet
xlswrite(file_name,{'Timestamp'},sheet_name,'A1');
xlswrite(file_name,date_str, sheet_name,'A2');
xlswrite(file_name, {'On Peak Load(kWh)'},sheet_name,'B1');
xlswrite(file_name, on_peak_vec_dr, sheet_name, 'B2');
xlswrite(file_name, {'Midday Peak Load (kWh)'}, sheet_name, 'C1');
xlswrite(file_name, midday_peak_vec_dr, sheet_name, 'C2');
xlswrite(file_name, {'Off Peak Load (kWh)'}, sheet_name, 'D1');
xlswrite(file_name, off_peak_vec_dr, sheet_name, 'D2');
xlswrite(file_name, {'State of Charge (kWh)'},sheet_name, 'E1');
xlswrite(file_name, battery_state, sheet_name, 'E2');
xlswrite(file_name, {'FFR Magnitude Event (kWh)'}, sheet_name, 'F1');
xlswrite(file_name, ffr_tracker_amt, sheet_name, 'F2');
xlswrite(file_name, {'NSAR Mangitude Event (kWh)'}, sheet_name, 'G1');
xlswrite(file_name, nsar_tracker_amt, sheet_name, 'G2');

%Writing Monthly Numbers
xlswrite(file_name, {'Month'}, sheet_name, 'H1');
xlswrite(file_name, output_month, sheet_name, 'H2');
xlswrite(file_name, {'On Peak Bill'}, sheet_name, 'I1');
xlswrite(file_name, tou_monthly_bill_dr(:,1), sheet_name, 'I2');
xlswrite(file_name, {'Midday Peak Bill'}, sheet_name, 'J1');
xlswrite(file_name, tou_monthly_bill_dr(:,2), sheet_name, 'J2');
xlswrite(file_name, {'Off Peak Bill'}, sheet_name, 'K1');
xlswrite(file_name, tou_monthly_bill_dr(:,3), sheet_name, 'K2');
xlswrite(file_name, {'Total Monthly Bill'}, sheet_name, 'L1');
xlswrite(file_name, total_tou_bill_dr, sheet_name, 'L2');


%% CSS + DR + TOU
% %Assumptions -
%Other Charges
customer_charge = 9.0; %in $/month, assumed single-phase
surcharge_fee   = 4.27/100; %in $/kWh
```

```
gif_fee         = 1.27; %in $/mo
minimum_charge_tou_css_dr  = 25.0; %in $/month, assumed single-phase

%Time-of-Use Charges
on_peak_charge       = 35.1826/100; %in $/kWhr
mid_peak_charge      = 12.807/100; %in $/kWhr
off_peak_charge      = 21.581/100; %in $/kWhr

%Intiialization
ffr_count_tou_css_dr = 1;
nsar_count_tou_css_dr = 1;
battery_tou_css_dr = zeros(length(res_load),1);
battery_tou_css_dr(1) = storage_capacity;

on_peak_vec_css_dr = zeros(length(res_load),1);
on_peak_css_charge_dr = zeros(length(res_load),1);
off_peak_vec_css_dr = zeros(length(res_load),1);
off_peak_css_charge_dr = zeros(length(res_load),1);
midday_peak_vec_css_dr = zeros(length(res_load),1);
midday_peak_css_charge_dr = zeros(length(res_load),1);

%Solar Panel Intialization
solar_prod_tou_css_dr = zeros(length(res_load),1);
solar_waste_tou_css_dr = zeros(length(res_load),1);

%DR Tracking
ffr_tracker = zeros(length(res_load),1);
ffr_tracker_amt = zeros(length(res_load),1);
nsar_tracker = zeros(length(res_load),1);
nsar_tracker_amt = zeros(length(res_load),1);

%Output Sheet
sheet_name = 'Integrated Isaac';

for iter_hour_tou_css_dr = 1:length(res_load)-1

    dr_min_satisfied = false; %initializing
    dr_min_not_satisfied = false; %initializing

    current_timestamp = res_load(iter_hour_tou_css_dr,1); %current time of load
            profile
    current_time_vec = datevec(current_timestamp);
    next_timestamp = res_load(iter_hour_tou_css_dr+1,1);
    current_battery_level = battery_tou_css_dr(iter_hour_tou_css_dr); %battery
            level at current time of load profile
```

```
current_load = res_load(iter_hour_tou_css_dr,2); %the load in current
        iteration
current_prod = solar_vec(iter_hour_tou_css_dr,2); %the production in
        current iteration
excess_prod = current_prod - current_load; %How much did the solar panel
        produce in excess of the load

%Checking if FFR event has taken place
if current_timestamp < ffr_events(ffr_count_tou_css_dr,1) &&
        next_timestamp > ffr_events(ffr_count_tou_css_dr,1)
   ffr_tracker(iter_hour_tou_css_dr) = true;
   ffr_tracker_amt(iter_hour_tou_css_dr) =
        (ffr_events(ffr_count_tou_css_dr,2)/batt_eff);
   battery_tou_css_dr(iter_hour_tou_css_dr+1) =
        battery_tou_css_dr(iter_hour_tou_css_dr) -
        ffr_tracker_amt(iter_hour_tou_css_dr); %Assuming that the average
        FFR event lasts 10 seconds


   if battery_tou_css_dr(iter_hour_tou_css_dr+1) > min_nsar_avail
      dr_min_satisfied = true;
   else
      dr_min_not_satisfied = true;
   end

   if ffr_count_tou_css_dr < length(ffr_events)
      ffr_count_tou_css_dr = ffr_count_tou_css_dr + 1;
   end

%Check if NSAR event has taken place
elseif current_timestamp < nsar_events(nsar_count_tou_css_dr,1) &&
        next_timestamp > nsar_events(nsar_count_tou_css_dr,1)
   nsar_tracker(iter_hour_tou_css_dr) = true;
   nsar_tracker_amt(iter_hour_tou_css_dr) =
        (nsar_events(nsar_count_tou_css_dr,2)/batt_eff);
   battery_tou_css_dr(iter_hour_tou_css_dr+1) =
        battery_tou_css_dr(iter_hour_tou_css_dr) -
        nsar_tracker_amt(iter_hour_tou_css_dr); %Subtracting the NSAR
        requirement from the battery

   if battery_tou_css_dr(iter_hour_tou_css_dr+1) > min_nsar_avail
      dr_min_satisfied = true;
   else
      dr_min_not_satisfied = true;
   end
```

```
    if nsar_count_tou_css_dr < length(nsar_events)
        nsar_count_tou_css_dr = nsar_count_tou_css_dr + 1;
    end
end


if dr_min_not_satisfied %if it did go below the minimum
    deficit_min_req = min_nsar_avail -
            battery_tou_css_dr(iter_hour_tou_css_dr+1);
else
    deficit_min_req = min_nsar_avail -
            battery_tou_css_dr(iter_hour_tou_css_dr);
end

%   DR event took place but minimum battery threshold is maintained
if dr_min_satisfied
    [on_peak_vec_css_dr, midday_peak_vec_css_dr, off_peak_vec_css_dr,
            battery_tou_css_dr,solar_prod_tou_css_dr, solar_waste_tou_css_dr]
            = tou_css_dr(iter_hour_tou_css_dr, current_load, current_prod,
            current_time_vec, battery_tou_css_dr,storage_capacity,
            steady_power_draw, on_peak_vec_css_dr, midday_peak_vec_css_dr,
            off_peak_vec_css_dr, min_nsar_avail, batt_eff,
            solar_prod_tou_css_dr, solar_waste_tou_css_dr);

%DR event took place but minimum battery threshold NOT maintain
elseif dr_min_not_satisfied
    if current_prod > 0 %if the panel is producing energy when the DR event
            happens
        if current_prod > deficit_min_req %if the panel can cover the deficit in
            the battery
            if deficit_min_req > steady_power_draw %if battery deficit is more
            than how much it can take in at once
             battery_tou_css_dr(iter_hour_tou_css_dr+1) =
            battery_tou_css_dr(iter_hour_tou_css_dr+1) + steady_power_draw;
             leftover_solar_css_dr = current_prod - steady_power_draw; %how
            much excess solar there is
             solar_prod_tou_css_dr(iter_hour_tou_css_dr) =
            solar_prod_tou_css_dr(iter_hour_tou_css_dr) + steady_power_draw;
            else %if the battery deficit is within the permissible limit
             battery_tou_css_dr(iter_hour_tou_css_dr+1) =
            battery_tou_css_dr(iter_hour_tou_css_dr+1) + deficit_min_req;
            %battery level is restored to minimum level in next hour iteration
             leftover_solar_css_dr = current_prod - deficit_min_req;
             solar_prod_tou_css_dr(iter_hour_tou_css_dr) =
            solar_prod_tou_css_dr(iter_hour_tou_css_dr) + deficit_min_req;
            end
```

```
    [on_peak_vec_css_dr, midday_peak_vec_css_dr, off_peak_vec_css_dr,
      battery_tou_css_dr,solar_prod_tou_css_dr, solar_waste_tou_css_dr]
      = tou_css_dr(iter_hour_tou_css_dr, current_load,
      leftover_solar_css_dr, current_time_vec,
      battery_tou_css_dr,storage_capacity, steady_power_draw,
      on_peak_vec_css_dr, midday_peak_vec_css_dr,
      off_peak_vec_css_dr, min_nsar_avail, batt_eff,
      solar_prod_tou_css_dr, solar_waste_tou_css_dr);

else %if the panel cannot cover the deficit in the battery
    new_deficit_req = deficit_min_req - current_prod; %how much still
      needs to be covered after solar panel production
    solar_prod_tou_css_dr(iter_hour_tou_css_dr) =
      solar_prod_tou_css_dr(iter_hour_tou_css_dr) + current_prod;
      %recording how much the solar panel was used

    %Filling the rest of the deficit from the grid
    %you want to see which  period you're in
    %Checking for On-Peak Pricing
    if any(current_time_vec(4) == 17:21)
        if deficit_min_req > steady_power_draw
            battery_tou_css_dr(iter_hour_tou_css_dr+1) =
      battery_tou_css_dr(iter_hour_tou_css_dr+1) + steady_power_draw;
            on_peak_vec_css_dr(iter_hour_tou_css_dr) =
      on_peak_vec_css_dr(iter_hour_tou_css_dr) + steady_power_draw;
        else
            battery_tou_css_dr(iter_hour_tou_css_dr+1) =
      battery_tou_css_dr(iter_hour_tou_css_dr+1) + deficit_min_req;
            on_peak_vec_css_dr(iter_hour_tou_css_dr) =
      on_peak_vec_css_dr(iter_hour_tou_css_dr) + deficit_min_req;
        end
    %Checking for Mid-Day pricing
    elseif any(current_time_vec(4) == 9:16)
        if deficit_min_req > steady_power_draw
            battery_tou_css_dr(iter_hour_tou_css_dr+1) =
      battery_tou_css_dr(iter_hour_tou_css_dr+1) + steady_power_draw;
            midday_peak_vec_css_dr(iter_hour_tou_css_dr) =
      midday_peak_vec_css_dr(iter_hour_tou_css_dr) +
      steady_power_draw;
        else
            battery_tou_css_dr(iter_hour_tou_css_dr+1) =
      battery_tou_css_dr(iter_hour_tou_css_dr+1) + deficit_min_req;
            midday_peak_vec_css_dr(iter_hour_tou_css_dr) =
      midday_peak_vec_css_dr(iter_hour_tou_css_dr) + deficit_min_req;
        end
```

```
            %Checking for Off-Peak Pricing
            else
                if deficit_min_req > steady_power_draw
                    battery_tou_css_dr(iter_hour_tou_css_dr+1) =
                battery_tou_css_dr(iter_hour_tou_css_dr+1) + steady_power_draw;
                    off_peak_vec_css_dr(iter_hour_tou_css_dr) =
                off_peak_vec_css_dr(iter_hour_tou_css_dr) + steady_power_draw;
                else
                    battery_tou_css_dr(iter_hour_tou_css_dr+1) =
                battery_tou_css_dr(iter_hour_tou_css_dr+1) + deficit_min_req;
                    off_peak_vec_css_dr(iter_hour_tou_css_dr) =
                off_peak_vec_css_dr(iter_hour_tou_css_dr) + deficit_min_req;
                end
            end
        end
    end

    %DR event did NOT take place
    else
    [on_peak_vec_css_dr, midday_peak_vec_css_dr, off_peak_vec_css_dr,
            battery_tou_css_dr,solar_prod_tou_css_dr, solar_waste_tou_css_dr]
            = tou_css_dr(iter_hour_tou_css_dr, current_load, current_prod,
            current_time_vec, battery_tou_css_dr,storage_capacity,
            steady_power_draw, on_peak_vec_css_dr, midday_peak_vec_css_dr,
            off_peak_vec_css_dr, min_nsar_avail, batt_eff,
            solar_prod_tou_css_dr, solar_waste_tou_css_dr);

  end
end

for iter = 1:length(on_peak_vec_css_dr)
  on_peak_css_charge_dr(iter) = on_peak_vec_css_dr(iter) * on_peak_charge +
            on_peak_vec_css_dr(iter) * surcharge_fee;
  midday_peak_css_charge_dr(iter) = midday_peak_vec_css_dr(iter) *
            mid_peak_charge + midday_peak_vec_css_dr(iter) * surcharge_fee;
  off_peak_css_charge_dr(iter) = off_peak_vec_css_dr(iter) * off_peak_charge +
            off_peak_vec_css_dr(iter) * surcharge_fee;
end

total_solar_waste_tou_css_dr = sum(solar_waste_tou_css_dr);
percent_solar_waste_tou_css_dr =
            (total_solar_waste_tou_css_dr/(sum(solar_vec(:,2))))*100;

[tou_monthly_bill_css_dr] = tou_bill_aggregation(res_load,
            on_peak_css_charge_dr, midday_peak_css_charge_dr,
```

```
                off_peak_css_charge_dr, gif_fee, customer_charge,
                minimum_charge_tou_css_dr);
total_dr_incentive = (capab_incentive_nsar + capab_incentive_ffr); %in $/yr
total_bill_css_dr = tou_monthly_bill_css_dr(:,4);

for month_css_dr = 1:length(tou_monthly_bill_css_dr)

   temp_month_bill = total_bill_css_dr(month_css_dr) - total_dr_incentive;
   if temp_month_bill < minimum_charge_tou_css_dr
      total_bill_css_dr(month_css_dr) = minimum_charge_tou_css_dr + gif_fee;
   else
      total_bill_css_dr(month_css_dr) = temp_month_bill;
   end
end

tou_css_bill_batt_dr = sum(total_bill_css_dr);
tou_css_savings_dr = conventional_chris_bill - tou_css_bill_batt_dr;
%Simple Payback
simple_payback_tou_dr = (total_batt_cost + install_cost)/(tou_css_savings_dr);

%Writing Hourly Data to Excel Sheet
xlswrite(file_name,{'Timestamp'},sheet_name,'A1');
xlswrite(file_name,date_str, sheet_name,'A2');
xlswrite(file_name, {'On Peak Load(kWh)'},sheet_name,'B1');
xlswrite(file_name, on_peak_vec_css_dr, sheet_name, 'B2');
xlswrite(file_name, {'Midday Peak Load (kWh)'}, sheet_name, 'C1');
xlswrite(file_name, midday_peak_vec_css_dr, sheet_name, 'C2');
xlswrite(file_name, {'Off Peak Load (kWh)'}, sheet_name, 'D1');
xlswrite(file_name, off_peak_vec_css_dr, sheet_name, 'D2');
xlswrite(file_name, {'State of Charge (kWh)'},sheet_name, 'E1');
xlswrite(file_name, battery_tou_css_dr, sheet_name, 'E2');
xlswrite(file_name, {'Solar Energy Used (kWh)'}, sheet_name, 'F1');
xlswrite(file_name, solar_prod_tou_css_dr, sheet_name, 'F2');
xlswrite(file_name, {'Solar Energy Wasted (kWh)'}, sheet_name, 'G1');
xlswrite(file_name, solar_waste_tou_css_dr, sheet_name, 'G2');
xlswrite(file_name, {'FFR Magnitude Event (kWh)'}, sheet_name, 'H1');
xlswrite(file_name, ffr_tracker_amt, sheet_name, 'H2');
xlswrite(file_name, {'NSAR Magnitude Event (kWh)'}, sheet_name, 'I1');
xlswrite(file_name, nsar_tracker_amt, sheet_name, 'I2');

%Writing Monthly Numbers
xlswrite(file_name, {'Month'}, sheet_name, 'J1');
xlswrite(file_name, output_month, sheet_name, 'J2');
xlswrite(file_name, {'On Peak Bill'}, sheet_name, 'K1');
xlswrite(file_name, tou_monthly_bill_css_dr(:,1), sheet_name, 'K2');
xlswrite(file_name, {'Midday Peak Bill'}, sheet_name, 'L1');
```

```
xlswrite(file_name, tou_monthly_bill_css_dr(:,2), sheet_name, 'L2');
xlswrite(file_name, {'Off Peak Bill'}, sheet_name, 'M1');
xlswrite(file_name, tou_monthly_bill_css_dr(:,3), sheet_name, 'M2');
xlswrite(file_name, {'Total Monthly Bill'}, sheet_name, 'N1');
xlswrite(file_name, total_bill_css_dr, sheet_name, 'N2');

%% CGS + DR + TOU
cgs_credit = 15.067/100; %in dollars per kWh

%Other Charges
customer_charge = 9.0; %in $/mont, assumed single-phase
minimum_charge_cgs_dr  = 25.0; %in $/month, assumed single-phase
surcharge_fee = 4.27/100; %in $/kWh
gif_fee       = 1.27; %in $/mo

%Time-of-Use Charges
on_peak_charge       = 35.1826/100; %in $/kWhr
mid_peak_charge      = 12.807/100; %in $/kWhr
off_peak_charge      = 21.581/100; %in $/kWhr

%Intiialization
ffr_count_tou_cgs_dr = 1;
nsar_count_tou_cgs_dr = 1;
battery_tou_cgs_dr = zeros(length(res_load),1);
battery_tou_cgs_dr(1) = storage_capacity;
on_peak_vec_cgs_dr = zeros(length(res_load),1);
off_peak_vec_cgs_dr = zeros(length(res_load),1);
midday_peak_vec_cgs_dr = zeros(length(res_load),1);

%Solar Panel Intialization
solar_prod_tou_cgs_dr = zeros(length(res_load),1);
solar_export_tou_cgs_dr = zeros(length(res_load),1);

%DR Tracking
ffr_tracker = zeros(length(res_load),1);
ffr_tracker_amt = zeros(length(res_load),1);
nsar_tracker = zeros(length(res_load),1);
nsar_tracker_amt = zeros(length(res_load),1);

%Output Sheet
sheet_name = 'CGS+DR+TOU';

for iter_hour_tou_cgs_dr = 1:length(res_load)-1

    dr_min_satisfied = false; %initializing
    dr_min_not_satisfied = false; %initializing
```

```
current_timestamp = res_load(iter_hour_tou_cgs_dr,1); %current time of load
        profile
current_time_vec = datevec(current_timestamp);
next_timestamp = res_load(iter_hour_tou_cgs_dr+1,1);
current_battery_level = battery_tou_cgs_dr(iter_hour_tou_cgs_dr); %battery
        level at current time of load profile

current_load = res_load(iter_hour_tou_cgs_dr,2); %the load in current
        iteration
current_prod = solar_vec(iter_hour_tou_cgs_dr,2); %the production in
        current iteration
excess_prod = current_prod - current_load; %How much did the solar panel
        produce in excess of the load

%Checking if FFR event has taken place
if current_timestamp < ffr_events(ffr_count_tou_cgs_dr,1) &&
        next_timestamp > ffr_events(ffr_count_tou_cgs_dr,1)
   ffr_tracker(iter_hour_tou_cgs_dr) = true;
   ffr_tracker_amt(iter_hour_tou_cgs_dr) =
        (ffr_events(ffr_count_tou_cgs_dr,2)/batt_eff);
   battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) =
        battery_tou_cgs_dr(iter_hour_tou_cgs_dr) -
        ffr_tracker_amt(iter_hour_tou_cgs_dr); %Assuming that the average
        FFR event lasts 10 seconds


   if battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) > min_nsar_avail
      dr_min_satisfied = true;
   else
      dr_min_not_satisfied = true;
   end

   if ffr_count_tou_cgs_dr < length(ffr_events)
      ffr_count_tou_cgs_dr = ffr_count_tou_cgs_dr + 1;
   end

%Check if NSAR event has taken place
elseif current_timestamp < nsar_events(nsar_count_tou_cgs_dr,1) &&
        next_timestamp > nsar_events(nsar_count_tou_cgs_dr,1)
   nsar_tracker(iter_hour_tou_cgs_dr) = true;
   nsar_tracker_amt(iter_hour_tou_cgs_dr) =
        (nsar_events(nsar_count_tou_cgs_dr,2)/batt_eff);
   battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) =
        battery_tou_cgs_dr(iter_hour_tou_cgs_dr) -
```

```
            nsar_tracker_amt(iter_hour_tou_cgs_dr); %Subtracting the NSAR
            requirement from the battery

        if battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) > min_nsar_avail
            dr_min_satisfied = true;
        else
            dr_min_not_satisfied = true;
        end

        if nsar_count_tou_cgs_dr < length(nsar_events)
            nsar_count_tou_cgs_dr = nsar_count_tou_cgs_dr + 1;
        end
    end


    if dr_min_not_satisfied %if it did go below the minimum
        deficit_min_req = min_nsar_avail -
                battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1);
    else
        deficit_min_req = min_nsar_avail -
                battery_tou_cgs_dr(iter_hour_tou_cgs_dr);
    end

%   DR event took place but minimum battery threshold is maintained
    if dr_min_satisfied
        [on_peak_vec_cgs_dr, midday_peak_vec_cgs_dr, off_peak_vec_cgs_dr,
                battery_tou_cgs_dr,solar_prod_tou_cgs_dr,
                solar_export_tou_cgs_dr] = tou_cgs_dr(iter_hour_tou_cgs_dr,
                current_load, current_prod, current_time_vec,
                battery_tou_cgs_dr,storage_capacity, steady_power_draw,
                on_peak_vec_cgs_dr, midday_peak_vec_cgs_dr,
                off_peak_vec_cgs_dr, min_nsar_avail, batt_eff,
                solar_prod_tou_cgs_dr, solar_export_tou_cgs_dr);

    %DR event took place but minimum battery threshold NOT maintain
    %FIX BELOW W/ NEW LOGIC
    elseif dr_min_not_satisfied
        if current_prod > 0 %if the panel is producing energy when the DR event
                happens
            if current_prod > deficit_min_req %if the panel can cover the deficit in
                the battery
                if deficit_min_req > steady_power_draw %if battery deficit is more
                than how much it can take in at once
                 battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) =
                battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) + steady_power_draw;
```

```
        leftover_solar_cgs_dr = current_prod - steady_power_draw; %how
        much excess solar there is
        solar_prod_tou_cgs_dr(iter_hour_tou_cgs_dr) =
        solar_prod_tou_cgs_dr(iter_hour_tou_cgs_dr) + steady_power_draw;
    else %if the battery deficit is within the permissible limit
        battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) =
        battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) + deficit_min_req;
        %battery level is restored to minimum level in next hour iteration
        leftover_solar_cgs_dr = current_prod - deficit_min_req;
        solar_prod_tou_cgs_dr(iter_hour_tou_cgs_dr) =
        solar_prod_tou_cgs_dr(iter_hour_tou_cgs_dr) + deficit_min_req;
    end

    [on_peak_vec_cgs_dr, midday_peak_vec_cgs_dr, off_peak_vec_cgs_dr,
        battery_tou_cgs_dr,solar_prod_tou_cgs_dr,
        solar_export_tou_cgs_dr] = tou_cgs_dr(iter_hour_tou_cgs_dr,
        current_load, leftover_solar_cgs_dr, current_time_vec,
        battery_tou_cgs_dr,storage_capacity, steady_power_draw,
        on_peak_vec_cgs_dr, midday_peak_vec_cgs_dr,
        off_peak_vec_cgs_dr, min_nsar_avail, batt_eff,
        solar_prod_tou_cgs_dr, solar_export_tou_cgs_dr);

else %if the panel cannot cover the deficit in the battery
    new_deficit_req = deficit_min_req - current_prod; %how much still
        needs to be covered after solar panel production
    solar_prod_tou_cgs_dr(iter_hour_tou_cgs_dr) =
        solar_prod_tou_cgs_dr(iter_hour_tou_cgs_dr) + current_prod;
        %recording how much the solar panel was used

    %Filling the rest of the deficit from the grid
    %you want to see which  period you're in
    %Checking for On-Peak Pricing
    if any(current_time_vec(4) == 17:21)
        if deficit_min_req > steady_power_draw
            battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) =
        battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) + steady_power_draw;
            on_peak_vec_cgs_dr(iter_hour_tou_cgs_dr) =
        on_peak_vec_cgs_dr(iter_hour_tou_cgs_dr) + steady_power_draw;
        else
            battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) =
        battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) + deficit_min_req;
            on_peak_vec_cgs_dr(iter_hour_tou_cgs_dr) =
        on_peak_vec_cgs_dr(iter_hour_tou_cgs_dr) + deficit_min_req;
        end
    %Checking for Mid-Day pricing
    elseif any(current_time_vec(4) == 9:16)
```

```
                if deficit_min_req > steady_power_draw
                    battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) =
                battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) + steady_power_draw;
                    midday_peak_vec_cgs_dr(iter_hour_tou_cgs_dr) =
                midday_peak_vec_cgs_dr(iter_hour_tou_cgs_dr) +
                steady_power_draw;
                 else
                    battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) =
                battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) + deficit_min_req;
                    midday_peak_vec_cgs_dr(iter_hour_tou_cgs_dr) =
                midday_peak_vec_cgs_dr(iter_hour_tou_cgs_dr) + deficit_min_req;
                 end
            %Checking for Off-Peak Pricing
            else
                if deficit_min_req > steady_power_draw
                    battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) =
                battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) + steady_power_draw;
                    off_peak_vec_cgs_dr(iter_hour_tou_cgs_dr) =
                off_peak_vec_cgs_dr(iter_hour_tou_cgs_dr) + steady_power_draw;
                 else
                    battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) =
                battery_tou_cgs_dr(iter_hour_tou_cgs_dr+1) + deficit_min_req;
                    off_peak_vec_cgs_dr(iter_hour_tou_cgs_dr) =
                off_peak_vec_cgs_dr(iter_hour_tou_cgs_dr) + deficit_min_req;
                 end
            end
        end
    end

    %DR event did NOT take place
  else
    [on_peak_vec_cgs_dr, midday_peak_vec_cgs_dr, off_peak_vec_cgs_dr,
            battery_tou_cgs_dr,solar_prod_tou_cgs_dr,
            solar_export_tou_cgs_dr] = tou_cgs_dr(iter_hour_tou_cgs_dr,
            current_load, current_prod, current_time_vec,
            battery_tou_cgs_dr,storage_capacity, steady_power_draw,
            on_peak_vec_cgs_dr, midday_peak_vec_cgs_dr,
            off_peak_vec_cgs_dr, min_nsar_avail, batt_eff,
            solar_prod_tou_cgs_dr, solar_export_tou_cgs_dr);
  end
end

[net_cgs_monthly_tou_dr_bill,gross_cgs_monthly_tou_dr_bill,
            cgs_monthly_credit_tou_dr] = cgs_monthly_tou_dr(res_load,
            solar_export_tou_cgs_dr, cgs_credit, minimum_charge_cgs_dr,
            gif_fee,on_peak_vec_cgs_dr, off_peak_vec_cgs_dr,
```

```
                midday_peak_vec_cgs_dr, on_peak_charge, off_peak_charge,
                mid_peak_charge);
%the above is how much the monthly bill is given everything

total_dr_incentive = (capab_incentive_nsar + capab_incentive_ffr); %in $/yr

for month = 1:length(net_cgs_monthly_tou_dr_bill)
   temp_month_bill = net_cgs_monthly_tou_dr_bill(month) - total_dr_incentive;
   if temp_month_bill < minimum_charge_cgs_dr
      net_cgs_monthly_tou_dr_bill(month) = minimum_charge_cgs_dr + gif_fee;
   else
      net_cgs_monthly_tou_dr_bill(month) = temp_month_bill;
   end
end

tou_cgs_bill_dr_final = sum(net_cgs_monthly_tou_dr_bill);
tou_cgs_savings_dr = conventional_chris_bill - tou_cgs_bill_dr_final;

%Simple Payback
simple_payback_cgs_tou_dr = (total_batt_cost +
                install_cost)/(tou_cgs_savings_dr);

%Writing Hourly Data to Excel Sheet
xlswrite(file_name,{'Timestamp'},sheet_name,'A1');
xlswrite(file_name,date_str, sheet_name,'A2');
xlswrite(file_name, {'On Peak Load(kWh)'},sheet_name,'B1');
xlswrite(file_name, on_peak_vec_cgs_dr, sheet_name, 'B2');
xlswrite(file_name, {'Midday Peak Load (kWh)'}, sheet_name, 'C1');
xlswrite(file_name, midday_peak_vec_cgs_dr, sheet_name, 'C2');
xlswrite(file_name, {'Off Peak Load (kWh)'}, sheet_name, 'D1');
xlswrite(file_name, off_peak_vec_cgs_dr, sheet_name, 'D2');
xlswrite(file_name, {'State of Charge (kWh)'},sheet_name, 'E1');
xlswrite(file_name, battery_tou_cgs_dr, sheet_name, 'E2');
xlswrite(file_name, {'Solar Energy Used (kWh)'}, sheet_name, 'F1');
xlswrite(file_name, solar_prod_tou_cgs_dr, sheet_name, 'F2');
xlswrite(file_name, {'Solar Energy Exported (kWh)'}, sheet_name, 'G1');
xlswrite(file_name, solar_export_tou_cgs_dr, sheet_name, 'G2');
xlswrite(file_name, {'FFR Magnitude Event (kWh)'}, sheet_name, 'H1');
xlswrite(file_name, ffr_tracker_amt, sheet_name, 'H2');
xlswrite(file_name, {'NSAR Magnitude Event (kWh)'}, sheet_name, 'I1');
xlswrite(file_name, nsar_tracker_amt, sheet_name, 'I2');

%Writing Monthly Numbers
xlswrite(file_name, {'Month'}, sheet_name, 'J1');
xlswrite(file_name, output_month, sheet_name, 'J2');
xlswrite(file_name, {'On Peak Bill'}, sheet_name, 'K1');
```

```
xlswrite(file_name, gross_cgs_monthly_tou_dr_bill(:,1), sheet_name, 'K2');
xlswrite(file_name, {'Midday Peak Bill'}, sheet_name, 'L1');
xlswrite(file_name, gross_cgs_monthly_tou_dr_bill(:,2), sheet_name, 'L2');
xlswrite(file_name, {'Off Peak Bill'}, sheet_name, 'M1');
xlswrite(file_name, gross_cgs_monthly_tou_dr_bill(:,3), sheet_name, 'M2');
xlswrite(file_name, {'Total Monthly Bill'}, sheet_name, 'N1');
xlswrite(file_name, net_cgs_monthly_tou_dr_bill, sheet_name, 'N2');
```