Homework 4

Spencer Egan

March 28, 2023

1 Problem Formulation

1.1 Sets

We choose to formulate this problem with three distinct sets. The first set $\mathcal{G}$ is the set of technologies we can invest in and install to meet the needs of our microgrid, which in our case is just {solar, storage, inverter}. The second set are the power sources and sinks behind the meter, $\mathcal{B}$. In order to handle solar curtailment and the charge and discharge of the battery all seperately, we will define $\mathcal{B}$ with the elements {solar, curtailed solar, battery discharge, battery charge}. The third set is the set of discrete time snapshots $\mathcal{T}$. We define the cardinality of each set with the variables $n = |\mathcal{G}|$, $m = |\mathcal{B}|$, and $s = |\mathcal{T}|$.

1.2 Decision Variables

We have several types of decision variables in this problem. The first is a vector $\gamma \in \mathbb{R}^n$ of the investment variables - how much DC capacity of solar and storage and how much AC inverter capacity we choose to install. Note that we are assuming a symmetric inverter capacity i.e. the the maximum power allowed to flow in either direction across the inverter is the same and equal to the single "capacity" variable as is standard in lithium-ion battery storage inverters. All three variables are positive, and in units of kW. We will follow the ordering of the set listed above and set γ_1 = solar capacity, γ_2 = storage capacity, and γ_3 = inverter capacity.

The second type is our matrix of power sources and loads behind the inverter $P \in \mathbb{R}^{s \times m}$. The rows of this matrix represent each time step $t \in \mathcal{T}$, while the columns represent (average) power of the the sources and sinks of $\mathcal{B}$ in each timestep in units of kW. Note that we are calling the curtailment of the solar a decision variable, but this is really a slack variable for an equality constraint we will define later. For ease of implementation, we choose to define all entries of P as positive. The first two columns of generated and curtailed solar will be less than the solar capacity γ_1. The thrid and fourth columns representing the battery discharge and charging power will be positive and less than the DC battery nameplate power γ_2.

The third decision variable is the slack vector $\xi \in \mathbb{R}^s$ representing the (positive) power from the grid into the microgrid at each timestep in $\mathcal{T}$ in units of kW. We will assume zero-export of the solar generation, constraining this variable to be only positive.

1.3 Parameters

We will define the annualized investment cost $\chi \in \mathbb{R}^n$ of each technology in $\mathcal{G}$ in units of \$/kW/yr. Using the information provided in the problem, $\chi = \begin{bmatrix}190 & 150 & 14\end{bmatrix}^\top$.

We define the inverter efficiency vector $\eta \in \mathbb{R}^m$ to map power flows on the DC side of the inverter to power flows on the AC side of the inverter. For our problem, $\eta = \begin{bmatrix}0.96 & 0 & 0.96 & \frac{-1}{0.96}\end{bmatrix}^\top$. Note that since we are treating the curtailed solar as a slack variable with non-physical power flows, none of the curtiled solar flows through the inverter. We are also choosing to define all power flows on the DC side as positive, hence the negative in this vector.

The battery roundtrip efficiency will be used in our state of charge constraint, and to make it consistant with our DC output matrix P, we will definie the vector $\upsilon \in \mathbb{R}^m$ such that $P\upsilon$ describes the net power flow out of (negative) and into (positive) the battery cells. Per the direction of Professor Hodge in email correspondance, we will take the stated 85% efficiency to include AC losses. One-way inverter efficiency is given as 96% so assuming symmetrical DC losses for the battery, the one way DC efficiency e is given by $e = \sqrt{\frac{0.85}{0.96^2}} = 0.9604$. We can then define the elements of υ as $\upsilon = \begin{bmatrix}0 & 0 & \frac{-1}{e} & e\end{bmatrix}^\top$.

The problem presents an hourly timeseries, but to be pedantic we should insist on including an hours-per-timestep scalar $h \in \mathbb{R}$ such that the units of $hP\upsilon$ are in kWh. Here, $h = 1$.

We will define the scalar d as the duration of battery in hours. In this problem we take $d = 4$.

We will define the scalar c to be the volumetric cost of energy from the grid in units of \$/kWh. For this problem we take $c = 0.2$.

We will define the demand $\delta \in \mathbb{R}^s$ with units of kW.

We will define the solar capacity factor with $\nu \in \mathbb{R}^s$ which is unitless, and provided in the timeseries csv file.

1.4 Power Balance constraint

The power balance constraint for the system is given by

$$P\eta + \xi = \delta.$$

1.5 State-of-Charge Constraint

Assuming the battery starts with no stored energy, we define the state of charge constraint with the (elementwise) inequalities

$$0_s \leq h\Gamma P\upsilon \leq d\gamma_2 1_s$$

where $0_s \in \mathbb{R}^s$ is the vector of all zeros, $\Gamma \in \mathbb{R}^{s\times s}$ is the matrix with ones on the main and lower diagonals and zeros on the upper diagonals, and $1_s \in \mathbb{R}^s$ is the vector of all ones. Note that by enforcing the first inequality elementwise, we are requiring the battery to start at

zero state of charge since it must always have consumed more energy charging than it has discharging.

1.6 Inverter Capacity Constraint

We can ensure that the inverter's nameplate power capacity is respected with the inequality constraints

$$-\gamma_3 1_s \leq P\eta \leq \gamma_3 1_s$$

where again $1_s \in \mathbb{R}^s$ and $1_m \in \mathbb{R}^m$ are both vectors of all ones.

1.7 Curtailment Slack Variable Constraint

Since we have defined a slack variable to keep track of curtailment, we need to define an equality constraint that requires the DC solar power produced and the (DC) solar power curtailed to equal the available capacity of the solar array. We will do that will the equation

$$P\sigma = \gamma_1 \nu$$

where $\sigma = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}^\top$.

1.8 Objective Function

For our objective function, we want to minimize the cost of using electricity from the grid, along with the annualized captial costs of our battery system and PV array. The units of the objective function should therefore be dollars per year. We can write this function in vector-matrix form as

$$\chi^\top \gamma + c * h * 1_s^\top \xi$$

We will implement an optimization program to minimize this objective function.

2 Implementation

We choose to implement our program in Julia using the JuMP package. The code can be run directly in this markdown document, but a separate Julia script will also be provided.

```
using CSV, LinearAlgebra, DataFrames, PrettyTables, Dates
using JuMP, Gurobi

function microgrid_model()
    root = dirname(@__FILE__)
    tspath = joinpath(root, "timeseries.csv")
    timeseries = CSV.read(tspath, DataFrame)

    G = ["solar", "storage", "inverter"]
    n = length(G)
```

```
    B = ["solar_gen", "solar_curt", "batt_discharge", "batt_charge"]
    m = length(B)
    s = nrow(timeseries)
    h = 1.0
    χ = h*(s/8760.0)*[190.0, 150.0, 14.0]
    η = [0.96, 0.0, 0.96, -1.0/0.96]
    e = sqrt(0.85/(0.96^2))
    υ = [0.0, 0.0, -1.0/e, e]
    d = 4.0
    c = 0.2
    δ = timeseries.load_kW
    ν = timeseries.solar_cf
    Γ = [1*(i≥j) for i in 1:s, j in 1:s]
    σ = [1,1,0,0]

    params = (G, n, B, m, s, h, χ, η, e, υ, d, c, δ, ν, Γ, σ, timeseries)

    microgrid = Model(Gurobi.Optimizer)

    @variable(microgrid, γ[1:n] ≥ 0)
    @variable(microgrid, P[1:s, 1:m] ≥ 0)
    @variable(microgrid, ξ[1:s] ≥ 0)

    @constraint(microgrid, P .≤ ones(s)*[γ[1] γ[1] γ[2] γ[2]])
    @constraint(microgrid, P*η + ξ .== δ)
    @constraint(microgrid, zeros(s) .≤ h*Γ*P*υ)
    @constraint(microgrid, h*Γ*P*υ .≤ d*γ[2]*ones(s))
    @constraint(microgrid, P*η .≤ γ[3]*ones(s))
    @constraint(microgrid, P*η .≥ -γ[3]*ones(s))
    @constraint(microgrid, P*σ .== γ[1]*ν)
    return microgrid, params
end

function minimize_costs()
    microgrid, params = microgrid_model()
    G, n, B, m, s, h, χ, η, e, υ, d, c, δ, ν, Γ, σ, timeseries = params
    γ = microgrid[:γ]
    P = microgrid[:P]
    ξ = microgrid[:ξ]

    set_optimizer_attribute(microgrid, "Method", 0)
    set_optimizer_attribute(microgrid, "Presolve", 0)

    @objective(microgrid, Min, χ'*γ + c*h*ones(s)'*ξ)

    optimize!(microgrid)

    optimal_system = DataFrame(tech = G, capacity_kW = value.(γ))

    dc_power = value.(P)
    ac_power = dc_power*Diagonal(η)
    solar_ac = ac_power[:, 1]
    solar_curt = dc_power[:, 2]
    battery_ac = ac_power[:, 3] + ac_power[:, 4]
    grid = value.(ξ)
    SoC = h*Γ*dc_power*υ / (value(γ[2])*d)
    optimal_dispatch = DataFrame(
        t = timeseries.t,
        solar_cf = ν,
```

```
        demand = δ,
        solar = solar_ac,
        battery = battery_ac,
        grid = grid,
        solar_curt = solar_curt,
        SoC = SoC
        )
    start = DateTime(2023, 1, 1)
    transform!(optimal_dispatch, :t => ByRow(t -> start + Dates.Hour(t)) => :datetime)

    optimal_cost = objective_value(microgrid)

    return optimal_system, optimal_dispatch, optimal_cost
end

optimal_system, optimal_dispatch, optimal_cost = minimize_costs()
```

2.1 Optimal Capacity

Printing our optimal capacity γ_{opt} we have

```
pretty_table(optimal_system, nosubheader = true)
```

```
     tech   capacity_kW

    solar       178.911
  storage       22.6939
 inverter        136.76
```

This shows that the optimal system has a solar DC:AC ratio of about 1.3 which is right in line with current common PV design best practices and so is not suprising. It is somewhat suprising that the optimal battery size is quite small relative to the PV size, but the costs of the battery (about $600/kWh) are fairly high and without further incentives a larger battery has rapidly diminishing returns. If the volumetric cost of power from the grid would be higher than twenty cents (as it currently is in many parts of the country right now with high natural gas prices), we would expect to see a larger battery be built to charge off of the curtailed solar power.

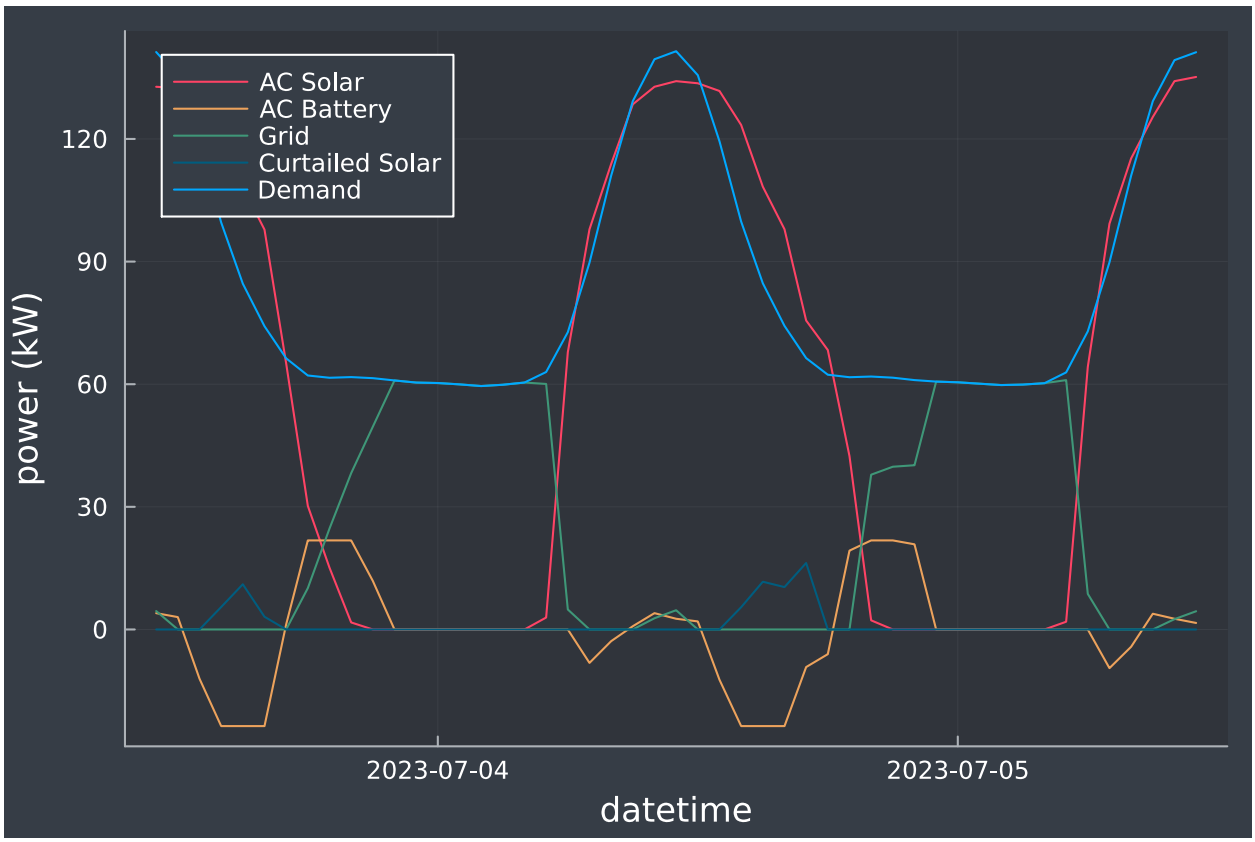
2.2 Dispatch Visualization

To get a sense of how the system is operating, we will look at the 48-hour period around the hour of maximum demand.

```
maxdemand = maximum(optimal_dispatch.demand)
maxhour = optimal_dispatch[optimal_dispatch.demand .== maxdemand, :t]
peakdispatch = subset(optimal_dispatch, :t => t -> abs.(t .- maxhour).≤24)

using Plots, PlotThemes
theme(:dark)
plot(
    peakdispatch.datetime,
    [peakdispatch.solar peakdispatch.battery peakdispatch.grid peakdispatch.solar_curt
peakdispatch.demand],
```

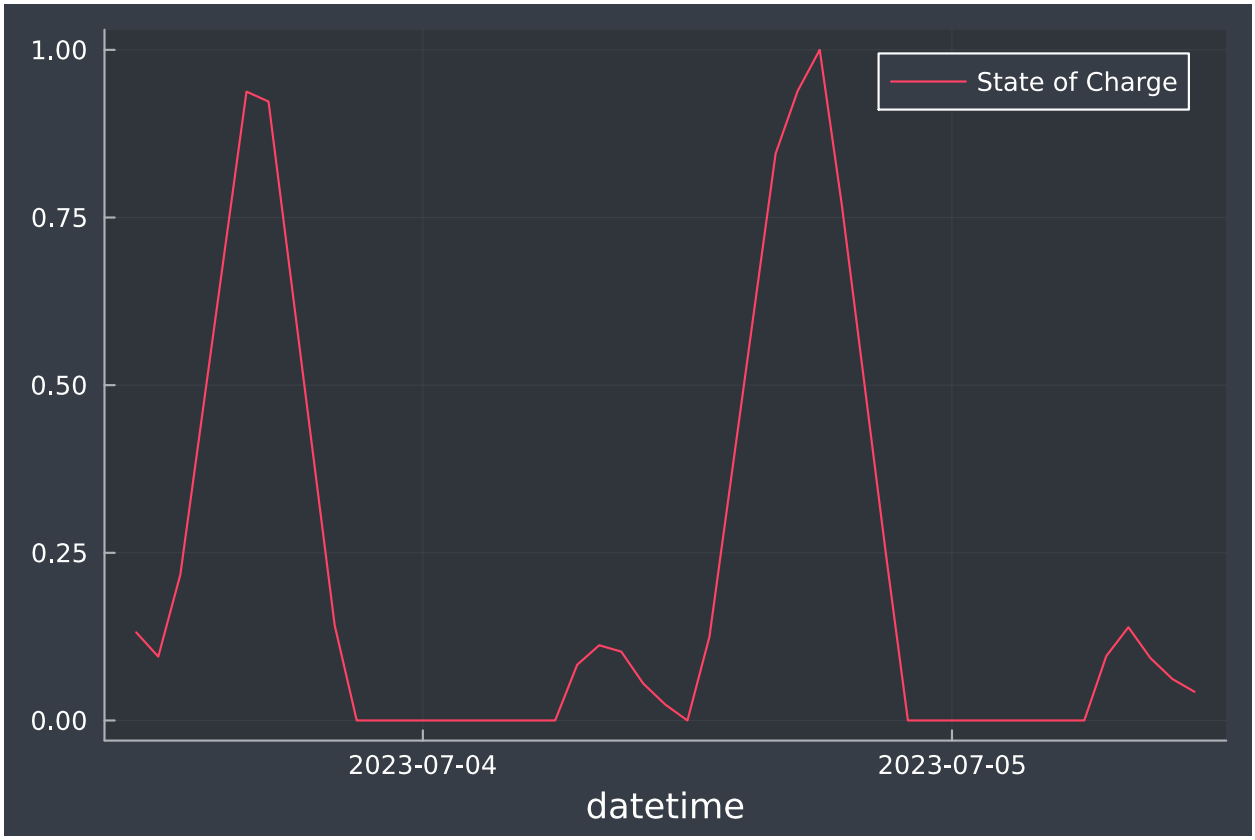
```
    label = ["AC Solar" "AC Battery" "Grid" "Curtailed Solar" "Demand"],
    xlabel = "datetime",
    ylabel = "power (kW)"
)
```



Here we see that the demand is quite well aligned with the solar production (which is perhaps why the optimal battery size was relatively small), but on the peak demand hour, there is not quite enough solar power to meet demand. The battery generally seems to charge in in the early morning and later afternoon when there is excess solar production, and discharge in the middle of the day when load is highest, and in the evenings when there is no available solar power.

If we plot the battery state of charge, we see that in this 48-hour period at least, the battery is cycling fully.

```
plot(
    peakdispatch.datetime,
    peakdispatch.SoC,
    label = "State of Charge",
    xlabel = "datetime"
)
```



2.3 Maximize Self-Consumption

To maximize self-consumption, we can keep the same problem formulation but set the objective function to minimize total energy consumed from the grid. We can write this total grid energy in vector-matrix form as $h1_s^\top \xi$.

To constrain the problem, we will impose a size limit on the PV system of 1MW, so $\gamma_1 \leq 1000$.

```
function minimize_gridenergy()
    microgrid, params = microgrid_model()
    G, n, B, m, s, h, χ, η, e, υ, d, c, δ, ν, Γ, σ, timeseries = params
    γ = microgrid[:γ]
    P = microgrid[:P]
    ξ = microgrid[:ξ]

    @constraint(microgrid, γ[1] ≤ 1000)
    @objective(microgrid, Min, h*ones(s)'*ξ)

    set_optimizer_attribute(microgrid, "Method", 3)
    set_optimizer_attribute(microgrid, "Presolve", -1)

    optimize!(microgrid)

    γ_opt = value.(γ)
    ξ_opt = value.(ξ)
    P_opt = value.(P)
    optimal_system = DataFrame(tech = G, capacity_kW = γ_opt)
    total_cost = χ'*γ_opt + c*h*ones(s)'*ξ_opt

    return optimal_system, total_cost
end
```

```
min_gridenergy_system, min_gridenergy_cost = minimize_gridenergy()

pretty_table(min_gridenergy_system, nosubheader = true)

println("Annual cost minimizing grid imports: \$", round(min_gridenergy_cost, digits =
2))
println("Minimum annual costs for optimal system: \$", round(optimal_cost, digits = 2))
```

```
     tech   capacity_kW

    solar        1000.0
  storage       2170.58
 inverter        141.47

Annual cost minimizing grid imports: $517665.43
Minimum annual costs for optimal system: $106312.11
```

We see that the the system built when trying to minimize grid imports is significantly larger, and significantly more expensive than the system built when trying to minimize total annual costs. As we might expect, the optimal system utilizes the maximum PV system allowed in the problem formulation. Furthermore, we note that the inverter size is only as big as the peak load, which makes sense since the battery can charge on the DC bus.

While we know that the system built when trying to minimize grid imports will be more expensive than the optimal system, we have no gaurantee that the system cost shown here is optimal at the level of grid imports acheived in our grid-import minimization problem. That is because there might be systems with smaller battery sizes that achieve this same level of grid imports (a larger battery would of course also be able to achieve this same level of grid imports). For this reason, comparing total costs between the two optimization problems is not too meaningful.