Homework 5

Complex capacity expansion planning

In this final homework assignment, we will explore the workings on a complex electricity system capacity expansion planning model that includes chronologically sequential operational decisions with time coupling constraints and transport flow constraints to represent power transmission limits between multiple geospatial regions.

The 'core' model and data we will use are provided in Notebook 07. This core model includes economic dispatch decisions with ramp constraints for thermal generators and storage constraints (but no thermal unit commitment constraints/decisions). It uses a simplified capacitated transport flow constraint to represent inter-regional electricity transmission between three regions (rather than DC OPF constraints), with data representing the Electricity Reliability Corporation of Texas (ERCOT) region.

Part 1: Time domain reduction methods

To reduce dimensionality and keep runtimes manageable for a tutorial assignment running on a desktop or laptop computer, we provide a set of *N* representative time periods of *Y* consecutive hourly periods (here either 24 hour days or 168 hour weeks) selected via a clustering method adapted from Mallapragada et al. (2018), "Impact of model resolution on scenario outcomes for electricity sector system expansion" Energy 163).

This method creates clusters of representative periods by trying to minimize variation between time series for wind, solar, and demand in each of the within-cluster periods (days or weeks). It also always includes the period containing the peak demand hour, if this period was not already selected as a cluster centroid.

To represent a full year of hourly operations, hours within each time period are weighted by a multiplier equal to the total number of periods (days or weeks) within the cluster represented by each representative period. In the objective function of the model, variable costs incurred in each hour are thus multiplied by this hourly weight to represent the cost of a full year of operation, as shown in the following two expressions, which make up the variable cost related part of the model objective function:

```
@expression(Expansion_Model, eVariableCosts,
     # Variable costs for generation, weighted by hourly sample
weight
    sum(sample_weight[t]*generators.Var_Cost[g]*vGEN[t,g] for t
in T, g in G)
)
@expression(Expansion_Model, eNSECosts,
     # Non-served energy costs
    sum(sample_weight[t]*nse.NSE_Cost[s]*vNSE[t,s,z] for t in T,
s in S, z in Z)
)
```

For example, if a representative day represents a cluster with 8 total days, variable costs incurred in each hour in that representative period are multiplied by a `sample_weight` of 8, to represent repeated similar operational patterns in the clustered days.

While time sampling methods like this can significantly reduce computational time, they also introduce abstraction errors that can bias model results. In this part of the assignment, we will explore these tradeoffs.

Question 1(a) - Using an IJulia notebook or an integrated development environment (IDE) such as VSCode (see links for tutorials on IJulia Notebooks and VSCode), copy and paste the entire content of the model from Notebook 07.

Feel free to create functions to contain portions of the code (such as the section that reads inputs, the section that sets up the JuMP model, the section that extracts outputs, and the section that writes to files), but this is optional.

Run the model using 10 sample days, by reading from the input data at `/Notebooks/complex_expansion_data/10_days/` and record your outputs to .csv files to a directory labeled `/10_days_Your_Name` (where 'Your_Name' is replaced with your first and last name).

Also ***record the solution time*** for the optimize model step, by reading the output of the `@time` macro included at this line:

`@time optimize!(Expansion_Model)`

The `@time` macro in Julia will output the total number of seconds used to complete the subequent function call (in this case, solving your optimization model). Save this number to a spreadsheet or text file for later reference.

Question 1(b) - Now let's test how the time sampling method works here. Successively run the model using the inputs for 4 weeks, 8 weeks, and 16 weeks provided in the following directories:

`/Notebooks/complex_expansion_data/4_weeks/`

`/Notebooks/complex_expansion_data/8_weeks/`

`/Notebooks/complex_expansion_data/16_weeks/`

Each time you run the model, (a) record the solution time, and (b) save your outputs to a different folder named `/4_weeks_Your_Name`, `8_weeks_Your_Name`, and `16_weeks_Your_Name`.

Create and save a scatter plot that shows the solution time in seconds (y axis) and the number of hours included in the model for the 10 days (240 hours), 4 weeks (672 hours), 8 weeks (1344 hours), and 16 weeks (2688) iterations. What do you notice about the increase in solution time? How does the computational performance of the model appear to scale as the number of time steps in the model (hours) increases?

Question 1(c) - Now let's see how the results of the model compare. Compile a spreadsheet that compares (a) the total cost results, (b) total final capacity (MW) results by resource, and (c) the total generation (GWh) results for all four iterations of the model.

What are the largest differences in results in each category? What do you think accounts for these differences? How would you measure or assess the accuracy of this model? How might this change, depending on the type of question are you are considering?

Question 1(d) The above experiment includes very little new wind or solar capacity additions. Let's try another case, which includes a carbon tax.

Save a new copy of your Julia file and then modify the following lines of code in the read inputs portion of yor model to incorporate a carbon price of $50 per ton of CO2 content in the fuel used by each resource. To do so, add an additional element to the total Variable Cost and Start-up Cost that includes 50 times the CO2 content of the fuel (tCO2/MMBtu) times the total fuel consumed by each resource (MMBtu).

```
for g in G
    # Variable cost ($/MWh) = variable O&M ($/MWh) + fuel cost
($/MMBtu) * heat rate (MMBtu/MWh)
    generators.Var_Cost[g] = generators.Var_OM_cost_per_MWh[g] +
        fuels[fuels.Fuel.==generators.Fuel[g],:Cost_per_MMBtu]
[1]*generators.Heat_rate_MMBTU_per_MWh[g]
    # CO2 emissions rate (tCO2/MWh) = fuel CO2 content
(tCO2/MMBtu) * heat rate (MMBtu/MWh)
    generators.CO2_Rate[g] =
fuels[fuels.Fuel.==generators.Fuel[g],:CO2_content_tons_per_MMBtu
][1]*generators.Heat_rate_MMBTU_per_MWh[g]
    # Start-up cost ($/start/MW) = start up O&M cost ($/start/MW)
+ fuel cost ($/MMBtu) * start up fuel use (MMBtu/start/MW)
    generators.Start_Cost[g] = generators.Start_cost_per_MW[g] +
        fuels[fuels.Fuel.==generators.Fuel[g],:Cost_per_MMBtu]
[1]*generators.Start_fuel_MMBTU_per_MW[g]
    # Start-up CO2 emissions (tCO2/start/MW) = fuel CO2 content
(tCO2/MMBtu) * start up fuel use (MMBtu/start/MW)
    generators.CO2_Per_Start[g] =
fuels[fuels.Fuel.==generators.Fuel[g],:CO2_content_tons_per_MMBtu
][1]*generators.Start_fuel_MMBTU_per_MW[g]
end
```

Now repeat the experiment in Question 1(b-c) above, recording solution time and results for 10 days, 4 weeks, 8 weeks, and 16 weeks time series inputs. Answer the following questions:

How have the capacity and energy results changed overall with addition of the carbon price to fuel costs (relative to the original cases?

How does the variation in the cost, capacity, and energy ouputs change now as you consider different number/duration of sample periods?

What does the overall experiment in Question 1 tell you about the generalizability of time sampling methods?

Bonus - As a bonus question: you can run the full 52 week time series and compare to the sample time series for Question 1(b-c) and/or 1(d) above. This case may take 1-2 hours to solve depending on your CPU. How do the full year results differ from your reduced time sample cases? Anything surprising? What might running this tell you about extrapolating the performance of time sampling methods based on a reduced time series rather than a full year (e.g. comparing 10 days or 4 weeks to 16 weeks vs 52 weeks)?

This is not a required question and is not worth additional credit, but do this if you're motivated/curious...

Question 2: Unit Commitment Constraints

Next, we will explore the implications of the operational resolution of your model. The current implementation includes simplified economic dispatch constraints that ignore the cycling costs and engineering constraints on thermal unit commitment.

Question 2(a) Save a new copy of your Julia file, and then modify your model by implementing the integer clustering approach to Unit Commitment introduced in Palmintier and Webster (2014), "Heterogeneous Unit Clustering for Efficient Operational Flexibility Modeling," *IEEE Transactions on Power Systems* 29(3) and Palmintier and Webster (2016), "Impact of Operational Flexibility on Electricity Generation Planning With Renewable and Carbon Targets," *IEEE Transactions on Sustainable Energy* 7(2).

For all generators in the subset UC, you must make the following modifications:

(0) Change the solver used to `Cbc` as this will be a mixed integer linear program now with discrete decisions. (Use `Pkg.add("Cbc")` if necessary.

(1) Change the `vRET_CAP`, and `vNEW_CAP` capacity investment decisions $\forall g \in UC$ to integer decisions in the domain $>= 0$. (Leave `vCAP` as a continuous decision.)

(2) Create a new set of operational decision variables `vSTART`, `vSHUT`, and `vCOMMIT` $\forall t \in T$ and $\forall g \in UC$, all of which are integer decision variables in the domain $>= 0$. `vSTART` and `vSHUT` indicate the number of units within each cluster that start up or shut down, respectively, in a given time period t. `vCOMMIT` is a commitment state variable that denotes the number of units currently committed (operating) in time period t.

(3) Remove generators in the set UC from Constraint 2 (`cMaxPower`).

(4) Add a new set of constraints to limit the max power of generators in the set UC:

$$vGEN_{t,g} \leq CapSize_g \times vCOMMIT_{t,g} \quad \forall t \in T, \forall g \in UC$$

Note that $CapSize_g$ parameter corresponds to the values in the `generators` data frame in column `:Cap_size`.

(5) Add a new set of constraints to limit the min power of generators in the set UC:

$$vGEN_{t,g} \geq MinPower_g \times CapSize_g \times vCOMMIT_{t,g} \quad \forall t \in T, \forall g \in UC$$

Note that $MinPower_g$ parameter corresponds to the values in the `generators` data frame in column `:Min_power`.

(6) Add the following upper bound constraints on the start-up, shut-down and commitment state variables:

$$vCOMMIT_{t,g} \leq vCAP_g / CapSize_g \quad \forall t \in T, \forall g \in UC$$

$$vSTART_{t,g} \leq vCAP_g / CapSize_g \quad \forall t \in T, \forall g \in UC$$

$$vSHUT_{t,g} \leq vCAP_g / CapSize_g \quad \forall t \in T, \forall g \in UC$$

(7) Add a new constraint to define the `vCOMMIT` state variable:

$$vCOMMIT_{t,g} = vCOMMIT_{t-1,g} + vSTART_{t,g} - vSHUT_{t,g} \quad \forall t \in T, \forall g \in UC$$

(8) Remove generators $ginUC$ from the Constraints (10), (10b), (11), and (11b) limiting ramp rates.

(9) Create new modified ramp rate constraints that account for unit commitment decisions:

$$vGEN_{t,g} - vGEN_{t-1,g} \leq RampUpPercentage_g \times CapSize_g \times (vCOMMIT_{t,g} - vSTART_{t,g}) + maximum(MinPower_g, RampUpPercenta$$

$$vGEN_{t-1,g} - vGEN_{t,g} \leq RampDnPercentage_g \times CapSize_g \times (vCOMMIT_{t,g} - vSTART_g) + maximum(MinPower_g, RampDnPercentag$$

Where $maximum(a,b)$ is the maximum (greater of) the two parameters and $RampUpPercentage_g$ and $RampDnPercentage_g$ are the maximum ramp up and down rates in the `:Ramp_Up_percentage` and `:Ramp_Dn_percentage` columns of the `generators` data frame.

(10) Implement minimum up and down time constraints that limit how frequently thermal units can cycle:

$$vCOMMIT_{t,g} \geq \sum_{i=t-MinUp_g}^{t} vSTART_{i,g} \quad \forall t \in T, \forall g \in UC$$

$$vCAP_g/CapSize_g - vCOMMIT_{t,g} \geq \sum_{i=t-MinDn_g}^{t} vSHUT_{i,g} \quad \forall t \in T, \forall g \in UC$$

Where $MinUp_g$ and $MinDn_g$ are the minimum up time and minimum down time parameters and stored in the `:Up_time` and `:Down_time` columns of `generators` data frame.

(11) Remove $g \in UC$ from the total capacity constraints 7(a) (`cCapOld`) and 7(b) (`cCapNew`) and add two new constraints for these units:

$$vCAP = ExistingCap_g/CapSize_g - vRETCAP_g \quad \forall g \in UC \& \in OLD$$

$$vCAP = CapSize_g \times vNEWCAP_g \quad \forall g \in UC \& \in NEW$$

Where $ExistingCap_g$ is the existing capacity parameters in `:Existing_Cap_MW` and $vRETCAP_g$ and $vNEWCAP_g$ are the decision variables `vRET_CAP` and `vNEW_CAP`.

(12) Finally, add a new expression to the objective function that calculates start-up costs incurred by thermal units:

$$eStartCost = \sum_{t \in T} \sum_{g \in UC} Weight_t \times StartCost_g \times vSTART_{t,g}$$

Where $Weight_t$ is the `sample_weight` for time `t`, and $StartCost_g$ is the start-up cost parameter calculated in this line:

```
    # Start-up cost ($/start/MW) = start up O&M cost ($/start/MW)
+ fuel cost ($/MMBtu) * start up fuel use (MMBtu/start/MW)
    generators.Start_Cost[g] = generators.Start_cost_per_MW[g] +
        fuels[fuels.Fuel.==generators.Fuel[g],:Cost_per_MMBtu]
[1]*generators.Start_fuel_MMBTU_per_MW[g]
```

Question 2(b) - Run the version of your model with unit commitment constraints and with 8 weeks of sample data. Compare your results to the 8 weeks results without unit commitment constraints, in terms of (a) costs, (b) capacity results, (c) energy results, (d) non-served energy results.

What do you notice? What are the major differences? What do you think accounts for these differences? When (for what type of research question or study context) do you think it might be appropriate to ignore unit commitment constraints and use economic dispatch constraints only?

Question 2(c) - Now we will try a 'linear relaxation' of the unit commitment implementation. Leave all constraints as they are. Simply change the definition of all of the integer decisions defined above (`vRET_CAP`, `vNEW_CAP`, `vSTART`, `vSHUT`, and `vCOMMIT`) $\forall g \in UC$. Then re-run your model using the Clp solver as an LP problem.

How much faster does this implementation solve than the integer unit commitment constraints? What are the major differences between results for the relaxed unit commitment problem? What do you think accounts for these differences? When (for what type of research question or study context) do you think it might be appropriate to use this implementation rather than the integer unit commitment implementation?

Extra credit question: cap and trade or clean electricity standard policy

As a bonus question (***worth extra credit up to 2 percentage points of total course grade***), implement a CO_2 cap and trade program or clean electricity standard policy constraint in your model.

Note that CO_2 emissions from generation and thermal unit start-ups are already calculated in the Notebook 07 model code and recorded in the parameters `generators.CO2_Rate[g]` and `generators.CO2_Per_Start[g]`.

Eligibility for a clean electricity standard is also recorded in the `generators.CES[g]` parameter.

Take care in your implementation to consider the role of time weights when using a reduced time series as we are here (not all hours are created equal!).

Run the model with 8 or 16 weeks of data and under increasing stringency (e.g. from 20% to 40%, to 60%, to 80%, to 100% emissions reduction or renewable energy requirement), and record and discuss results of each case.

What do you notice about solution time and results as you change the stringency of the policy? How much does the policy increase costs at each level of stringency, relative to no policy? How much does it reduce CO_2 emissions? (You will need to add a calculation of CO_2 emissions to the model outputs recorded)