Author: *Xingpeng Li*, initially documented in 2016, last updated in April 2023.
Website: https://rpglab.github.io/people/Xingpeng-Li/
Email: xplipower@gmail.com

SCED Manual for Developers

In today's power system operations, security-constrained economic dispatch (SCED) is one of the key functions in the energy management system (EMS). In general, SCED is an optimization process that aims to provide the least cost generation that meets all the operation and reliability constraints.

Typically, SCED is a linear programming (LP) problem since all generators' status are considered as fixed input and DC power flow model is used. Though AC power flow model is more accurate, it is not used due to its computational complexity. The non-linearity and non-convexity may cause divergence of the SCED problem and hence AC power flow model based SCED is not stable. On the contrary, DC power flow model based SCED is a LP problem, which is much simpler and the optimal solution can be guaranteed if it is feasible.

In terms of time-frame, the two types of SCED applications are day-ahead (DA) SCED and real-time (RT) SCED. DA-SCED considers multiple periods while RT-SCED only consider one-period. In this manual, SCED is for RT-SCED unless indicated specifically.

RT-SCED calculates the desired MW output level for all on-line units repeatedly with updated information. For instance, RT SCED runs every 5 minutes for PJM. The results from RT SCED are the actual dispatch signals that will be sent to each generator.

In terms of input data type, there are deterministic SCED and stochastic SCED. For deterministic SCED, all inputs, including load profile and generation of renewable energy resources, are assumed to be accurate without any forecasting error. However, for stochastic SCED, some degrees of uncertainty can be taken into account. As shown in Fig. 1, it is observed that the most possible value of the wind power is 100 MW, which is the only input for deterministic SCED; however, for stochastic SCED, the input can be a set of several discrete points (80 MW, 90 MW, 100 MW, 110 MW, and 120 MW) with weights assigned to them, or a range from 80 MW to 120 MW. Available mathematical methods include robust programming, stochastic programming, and chance programming. This set of codes is created for deterministic SCED only.

The SCED is expected to use the same model and function the same way with the PJM RT SCED. However, due to data unavailability and unknown tricks insider the PJM's commercial SCED, it is impossible to have exactly the same results. The input to this SCED tool includes the system network topology, generator status, load forecast, and forecasting generations of renewable energy resource. Its objective is to minimize the total cost, subject to physical constraints and other requirements such as reserve requirements. The output would be the dispatch point of each generator.

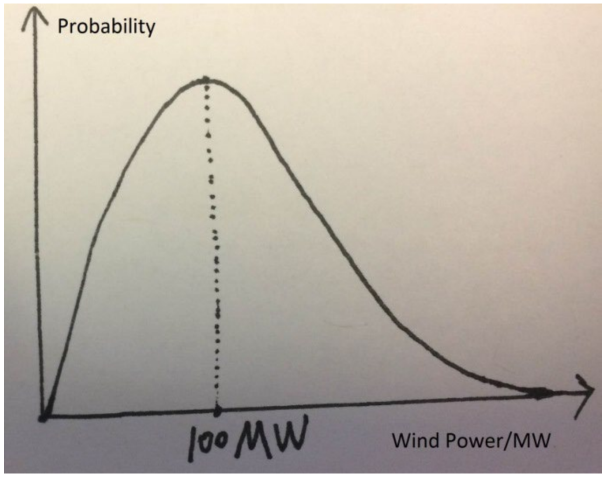


Fig. 1 Probability density function curve

The SCED can handle large-scale systems and serves as a simulator for off-line study. It is also able to load different input data formats. There are two sets of data for the PJM system. One is referred to as real-case and the other is referred to as generic-case. Real-case is a set of data files containing EMS information and market information. Generic-case is a simplified version of real-case, which contains much less data and hidden all sensitive information. Therefore, the SCED for generic-case is referred to as G-SCED while the SCED for real-case is referred to as R-SCED.

1 Pyomo set up

Pyomo is an open-source optimization library based on python. Before starting to install pyomo, users should make sure python is installed in their computers. Some available *Scientific Python distributions* can be downloaded from the following link:

http://www.scipy.org/install.html

Install pyomo with Anaconda (one of the scientific Python distributions):

Open "Anaconda Prompt", type "pip install pyomo"

The following operations may not be needed if you have a scientific Python distributions installed in your computer.

In "Anaconda Prompt", type "pip install numpy"

In "Anaconda Prompt", type "pip install scipy"

In "Anaconda Prompt", type "pip install pyomo.extras"

Reference: http://www.pyomo.org/installation/

How to test pyomo is installed successfully in your computer?

Open cmd, if it is not in C:\ disk, type "C:", press "Enter". Then, type "pyomo -h"; you will see some related information if pyomo is installed correctly.

2 Third party solver set up

Two open-source free solvers that have been tested are: glpk and cbc. Note that glpk 4.60 does not work while glpk 4.55 works. The latest cbc 2.7.5 can work. To call these two solvers, users can add their directory in the windows system environment variable PATH. This may require administrator rights. Note that after the system variable PATH is set for solvers, users may have to restart the computers to make the solver work.

For glpk: the directory containing the *glpsol executable* (glpsol.exe) must be in the list of paths defined by the PATH environment variable. GLPK-4.55 for windows can be downloaded from the below link. You need to unzip it. https://sourceforge.net/projects/winglpk/files/winglpk/

For cbc: the directory containing the *cbc executable* (cbc.exe)must be in the list of paths defined by the PATH environment variable. Cbc-2.7.5 for windows can be downloaded from the below link. You need to unzip it. http://www.coin-or.org/download/binary/Cbc/

There could be newer version of both glpk and cbc available. However, test them before trust them. At least, glpk-4.60 does not work very well with Pyomo-4.3.11328.

3 Configure file

All basic input parameters are specified in the file with a name of "configure.txt". In this file, any contents after "//" or "#" in the same line will be ignored.

//---------- Solver settings ------------//

@solverName: name of the solver.

@solverTimLimit: maximum computational time allowed for solver, in seconds.

@solverOptGap: optimization relative gap; if it is reached, solver will stop and exit.

//---------- SCED settings ------------//

@isRunSCED: if true, SCED simulator will run; if false, no SCED will be run.

@isPyomoDataFilesAvailable:

This parameter will be used ONLY when @isRunSCED is set to TRUE.

If true, SCED will directly load the input data file as specified by the parameters

@pathRealCase + @pyomoDataFormatInputFileRC for real-case, or

@pathGenericCase+ @pyomoDataFormatInputFileGCfor generic-case.

If false, the program will first generate those files from initial/original real-case or generic-case data files.

@generatePyomoDataFiles:

This parameter will be used ONLY when @isRunSCED is set to FALSE.

True means the program will produce pyomo-format based data files while 'false' will generate normal format based files.

@needHeading:

This parameter will matter only when the code is generating regular (non-pyomo) format data files. If true, first line is heading.

@runSCEDTimeFrame: SCED will be run every 'runSCEDTimeFrame' (most likely 5) minutes. This parameter is only used for determining the SCED period with the input time.

@blockPrice: Its unit is $, this parameter will matter ONLY when the code is generating pyomo input data file. It is used to linearize the slope cost curve.

@isPositivePgPmaxPminNeeded: if true, pg, pgmax, pgmin will be set to zero if they are negative.

@handle_CostCurveSegment_Pgmin: if true, the program will automatically fix the conflict between cost_curve_based_Pgmax and pgmin (cost_curve_based_Pgmax < pgmin).

@costCurveSegment_Pgmin_option: this parameter will matter only when CostCurveSegment is set to true. If it is 1, pgmin will be changed to cost_curve_based_Pgmax.

//---------- Date and Time -----------//

@Year, @Month, @Day, @Hour, @Minute, @Second:

These six parameters are used for the real-case study. Since some original data files for real-case are used for an entire day, while the pyomo-format case input file corresponds to a specific time slot, so this date and time information are needed for updated information search and match.

//---------- Real case input data -----------//

@pyomoDataFormatInputFileRC: input real-case that will be sent to pyomo. This parameter has to be in the same directory with the program codes.

@pathRealCase: directory of the original files for a real-case

@bidFileNameRC: daily bid data

@costCurveRampFileNameRC: costcurve ramping rate

@hourlyDataFileNameRC: hourly updated unit data

@interfaceFileNameRC: daily interface/constraint data

@reserveReqFileNameRC: reserve requirement

@aolFileNameRC: AOL file containing the dFAX (PTDF) data

@rawFileNameRC: PSS/E power flow raw file.

@rtUnitStatusFileNameRC: updated unit status data

@unitFileNameRC: This csv file is used to match unit name and unit ID; one line heading.

//---------- Generic case input data -----------//

@pyomoDataFormatInputFileGC: input generic-case that will be sent to pyomo. This parameter has to be in the same directory with the program codes.

@pathGenericCase: directory of the original files for a generic-case.

@costCurveOutputFileNameGC: unit cost curve data.

@costCurveMultiRampFileNameGC: unit energy ramping rate data.

@costCurveSpinRampFileNameGC: unit spinning ramping rate data.

@rawFileNameGC: PSS/E power flow raw file.

4 Main programs

There are two main programs:

- RunSCEDGenericCaseModel.py
- RunSCEDRealCaseModel.py.

Assume the programs are in the directory of C:/Users/obama/sced/. To run SCED simulation for generic case, open "cmd", change its current working directory to C: disk by typing "C:" and then press "Enter" key. Then, use command "cd" to enter each directory until the inside of sced folder; for instance, in "C:\", type "cd Users" will move the working directory to "C:\User".

When you are in the directory of C:/Users/obama/sced/, type "python RunSCEDGenericCaseModel", press "Enter", then, the G-SCED starts to run. If you need to all the information that shown in the screen saved to a file, use this command "python RunSCEDGenericCaseModel > filename.txt". So, if there is already a file that has the same name of "filename.txt", the program will first delete that file, and creates a new file with the same name. If you want to keep the old file, and still need to save those information to the same file, use command "python RunSCEDGenericCaseModel >> filename.txt"; then, all output will be appended to that file.

You can use the same procedure of G-SCED to run R-SCED. Just remember to change the name from "RunSCEDGenericCaseModel" to "RunSCEDRealCaseModel".

Another way to run the simulation is to use *Spyder's GUI run function*.

4.1 Python file explanations

- Diary.py: used for logging the process of simulation.
- GeneralClasses.py: used for dealing with date and time.
- GeneralFunctions.py: some general-purpose auxiliary methods.
- GeneratePyomoDataFiles.py: used to generate pyomo-format based data file.
- LoadInitFiles.py: used to read the initial data files for both real-case and generic-case.
- ParamManager.py: used to read "configure.txt" file and store all parameters' value.
- RawEMSMarketModel.py: classes for handling and storing real-case data.
- RawGenericModel.py: classes for handling and storing generic-case data.

- RunSCEDGenericCaseModel.py: main program to run SCED for generic-case.
- RunSCEDRealCaseModel.py: main program to run SCED for real-case.
- SCEDGenericCaseModel.py: Pyomo-based model for generic-case.
- SCEDRealCaseModel.py: Pyomo-based model for real-case.
- WriteResults.py: used for dumping results to files for both types of case.

4.2 Pyomo Instance Creation Acceleration tip

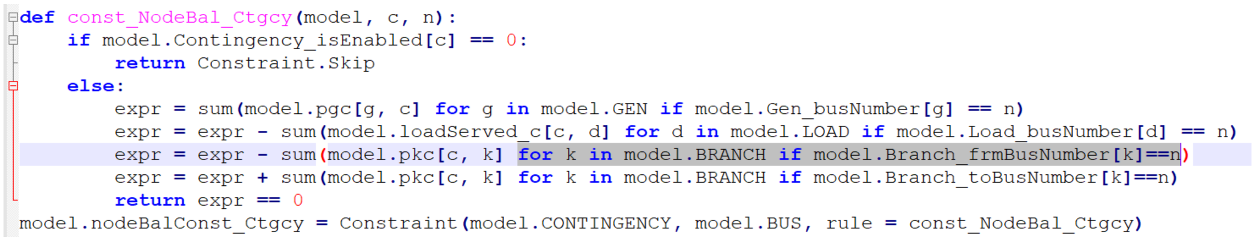
When the test system is a large-scale network with thousands buses, the following command create_instance() will take a lot of computing time:

instanceSCED = SCEDModel.create_instance(DatafileED)

## from SCEDGenericCaseModel import model as SCEDModel

## where model SCEDGenericCaseModel.py file is a pyomo AbstractModel.

For a system with over 10,000 buses, the entire program may take 15 minutes while 10 minutes are used to create an instance using create_instance(). To address this issue, we can avoid using for-loops in the model/formulation definition class. For example, we can define a busbranch class / data-structure to store the *Nb* lists of lines for each bus (*Nb* denotes the total number of buses); then, when we establish nodal power balance constraint, we do not have to loop through all branches (could be over 20,000) for every single bus like the following highlighted part (a loop and an if-condition check per iteration) that would take unnecessarily long computing time.



```
def const_NodeBal_Ctgcy(model, c, n):
    if model.Contingency_isEnabled[c] == 0:
        return Constraint.Skip
    else:
        expr = sum(model.pgc[g, c] for g in model.GEN if model.Gen_busNumber[g] == n)
        expr = expr - sum(model.loadServed_c[c, d] for d in model.LOAD if model.Load_busNumber[d] == n)
        expr = expr - sum(model.pkc[c, k] for k in model.BRANCH if model.Branch_frmBusNumber[k]==n)
        expr = expr + sum(model.pkc[c, k] for k in model.BRANCH if model.Branch_toBusNumber[k]==n)
        return expr == 0
model.nodeBalConst_Ctgcy = Constraint(model.CONTINGENCY, model.BUS, rule = const_NodeBal_Ctgcy)
```

5 Log file

Each of the two main programs will automatically generate a log file with date and time in its name. From this log file, the user can check how much time each specific part takes and, then, determine which block of code is the performance bottleneck.

For SCED application on a large-scale system like the PJM system, the bottleneck would be pyomo itself rather than the solver or the developer's code.

6 G-SCED

6.1 Introduction

G-SCED is a SCED for the generic case. The generic case for SCED consists of 4 data files. They are a power flow (PSS/E raw) file, an energy ramping rate file, a spinning ramping rate file, and a unit cost curve file.

There is no unit commitment in G-SCED model, which means that it will only not change the status of any unit. G-SCED only re-dispatch the generators that have a cost curve available, and fixes the outputs of all other generators. Energy ramping rate is used to restrict the change in generator output between the starting time and ending time of the same period. Spinning ramping rate is used to restrict the generator output change for the situations when a contingency occurs. *N*-1 is not represented explicitly; the largest generator contingency is used to as the spinning reserve requirement. Specific contingency constraints can be specified in the input data files for G-SCED. Slack variables are used to make sure the problem is feasible.

6.2 Procedure to generate pyomo-format based input data file

a) Load configure file.
b) Load original power flow file (fixed format of PSS/E version 26 raw file).
c) Load original generator energy ramping rate data file (first 4 heading lines skipped).
d) Load original generator spinning ramping rate data file (first 4 heading lines skipped).
e) Load original generator cost curve data file (first 3 heading lines skipped).
f) Use the related parameters (like "@needHeading") from configure file.
g) Then, write needed bus data.

h) Write needed load data.

i) Write needed generator cost curve data.

j) Write needed other generator data.

k) Write needed branch data.

Note that the order of loading data files may not matter.

The program does not generate the contingency-related data or the interface limit related data due to data availability at this point of time. However, if those data are available, user can append them with proper format to the end of the generated file. If not, the G-SCED simulator will automatically ignore the associated constraints from its model.

6.3 Mathematical model

Sets

N:	Buses.
I:	Interface.
K:	Branches.
K(n-):	Branches for which the from-bus is *n*.
K(n+):	Branches for which the to-bus is *n*.
K(i):	Branches that form the interface *i*.
G:	Generators.
GC:	Generators that have cost curve available.
G(n):	Generators at buses *n*.
D:	Loads.
D(n):	Loads at buses *n*.
C:	Branches that are in contingency list.

Parameters

P_d:	Total load in MW for load *d*.
P_{g0}:	Total initial output of generator *g*.
$P_{g,max}$:	Maximum output for generator *g*.
$P_{g,min}$:	Minimum output for generator *g*.
MRR_g:	Energy ramping rate (associated with the generation level of P_{g0}) for generator *g*, in MW/min.
SRR_g:	Spinning ramping rate (associated with the generation level of P_{g0}) for generator *g*.
NS_g:	Number of cost segments for generator *g*.

SRC:	Spinning reserve cost/price ($/MW).
$PT_{g,i}$:	Breadth of segment *i* for generator *g*.
$C_{g,i}$:	Cost of segment *i* for generator *g* ($/MW).
α_k:	Phase angle of branch *k*. 0 if branch *k* is not a phase shifter (phase angle regulator).
x_k:	Reactance of branch *k*.
$RateA_k$:	Branch limit for normal situation.
$RateC_k$:	Branch limit for contingency situation.
$Rate_i$:	Total flow limit for interface *i*.
St_d:	Status for load *d*. 1 denotes in service, while 0 means out of service.
St_g:	Status for generator *g*. 1 denotes in service, while 0 means out of service.
St_k:	Status for branch *k*. 1 denotes in service, while 0 means out of service.
T_{ED}:	Look-ahead time for SCED (15 minutes in this manual).
T_{SR}:	Timeframe for spinning reserve to react to a contingency (10 minutes in this manual).

Variables

δ_n:	Angle of bus *n*.
p_d:	Actual load served for load *d*.
p_{sd}:	The amount of shedded load for load *d*. Non-negative value.
p_k:	Flow on branch *k*.
p_g:	Total output of generator *g*.
$p_{g,i}$:	Output on segment *i* for generator *g*.
sr_g:	Spinning reserve that generator *g* provides.
p_i:	Total flow for interface *i*.
$p_{g,c}$:	Total output of generator *g* under contingency *c*.
$p_{k,c}$:	Flow on branch *k* under contingency *c*.
$p_{d,c}$:	Actual load served for load *d* under contingency *c*.
$p_{sd,c}$:	The amount of shedded load for load *d* under contingency *c*.
$p_{i,c}$:	Total flow for interface *i* under contingency *c*.

Slack variables (non-negative)

sv_sr_g:	Slack variable for ramping rate limit associated to spinning reserve.
sv_RATE_k:	Slack variable for the flow limit of branch *k*.
$sv\_P_{g,max}$:	Slack variable for parameter $P_{g,max}$.
$sv\_P_{g,min}$:	Slack variable for parameter $P_{g,min}$.

$sv\_MRR_{g,u}$:	Slack variable for parameter MRR_g - ramping up rate relaxation.
$sv\_MRR_{g,d}$:	Slack variable for parameter MRR_g - ramping down rate relaxation.
sv_SRR_g:	Slack variable associated with the largest generator contingency reserve requirement.
sv_RATE_i:	Slack variable for the total flow limit of interface *i*.

Penalty factors

PF_P_{max}:	A fix penalty factor for variable $sv\_P_{g,max}$.
PF_P_{min}:	A fix penalty factor for variable $sv\_P_{g,min}$.
PF_MRR_u:	A fix penalty factor for variable $sv\_MRR_{g,u}$.
PF_MRR_d:	A fix penalty factor for variable $sv\_MRR_{g,d}$.
PF_SRR:	A fix penalty factor for variable sv_SRR_g.
PF_sr:	A fix penalty factor for variable sv_sr_g.
PF_RATEK:	A fix penalty factor for variable sv_RATE_k.
PF_RATEI:	A fix penalty factor for variable sv_RATE_i.
PF_PD:	A fix penalty factor for variable p_{sd}.
PF_PDC:	A fix penalty factor for variable $p_{sd,c}$.

Formulation

Objective function

$$\begin{aligned} min \ \sum_{g\in G}\sum_{i=1}^{NS_g} P_{g,i}C_{g,i} + SRC\sum_{g\in G} sr_g + PF\_sr\sum_{g\in G} sv\_sr_g + PF\_RATEK\sum_{k\in K} sv\_RATE_k + PF\_PD\sum_{d\in D} p_{sd} \\ + PF\_P_{max}\sum_{g\in G} sv\_P_{g,max} + PF\_P_{min}\sum_{g\in G} sv\_P_{g,min} + PF\_MRR_u\sum_{g\in G} sv\_MRR_{g,u} \\ + PF\_MRR_d\sum_{g\in G} sv\_MRR_{g,d} + PF\_SRR\sum_{g\in G} sv\_SRR_g + PF\_RATEI\sum_{i\in I} sv\_RATE_i \\ + PF\_PDC\sum_{c\in C}\sum_{d\in D} p_{sd,c} \end{aligned}$$

Base case constraints

1) Bus power balance equation

$$\sum_{g\in G(n)} p_g + \sum_{k\in K(n+)} p_k - \sum_{k\in K(n-)} p_k = \sum_{d\in D(n)} p_d$$

$$p_d + p_{sd} = P_d \cdot St_d$$

$$p_{sd} = 0 \cdot if \ \ St_d == 0 \ \cdot for \ \ d$$

$$p_{sd} = 0 \cdot \ if\ P_d < 0\ \cdot\ for\ \ d$$
$$p_{sd} \geq 0 \cdot \ if\ St_d \cdot P_d \geq 0\ \cdot\ for\ \ d$$

2) B-theta power flow equation

If $St_k == 1$, then

$$p_k = (\delta_{n(k-)} - \delta_{n(k+)} + \alpha_k)/x_k$$

$\delta_{n(k-)}$ denotes the angle of bus that is the from-bus of branch k.

$\delta_{n(k+)}$ denotes the angle of bus that is the to-bus of branch k.

If $St_k == 0$, then

$$p_k = 0$$

3) Branch flow limit only for *branch k in KA*

$$P_k \leq RateA_k + sv\_RATE_k$$
$$P_k \geq -RateA_k - sv\_RATE_k$$

4) Generator total output equation

$$0 \leq P_{g,i} \leq PT_{g,i}$$

and,

If $St_g == 0$, then $p_g = 0$ and $p_{g,i} = 0$,

else if $P_{g0} > 0$ and $g \in GC$, then: $p_g = \sum_{i=1}^{NS_g} p_{g,i}$,

else if $P_{g0} \leq 0$, then $p_{g,i} = 0$.

and,

If $g \in G - GC$, then $p_g = P_{g0} \cdot St_g$.

5) Ramping rate limit

If $St_g == 0$, then

$sv\_MRR_{g,u} = 0$

$sv\_MRR_{g,d} = 0$,

$sv\_sr_g = 0$

$sv\_P_{g,max} = 0$

$sv\_P_{g,min} = 0$

$sr_g = 0$

$sv\_SRR_g = 0$

If $St_g == 1$, then:

$$p_g - P_{g0} \leq MRR_g \cdot T_{ED} + sv\_MRR_{g,u}$$
$$p_g - P_{g0} \geq -MRR_g \cdot T_{ED} - sv\_MRR_{g,d}$$
$$sr_g \leq SRR_g \cdot T_{SR} + sv\_sr_g$$
$$p_g \leq P_{g,max} + sv\_P_{g,max}$$
$$p_g \geq P_{g,min} - sv\_P_{g,min}$$
$$p_g + sr_g \leq P_{g,max} + sv\_P_{g,max}$$
$$\sum_{m \in G} sr_m \geq sr_g + p_g - sv\_SRR_g$$

6) Interface limit

$$\sum_{k \in K(i)} p_k \leq Rate_i$$

Contingency case constraints

1) Bus power balance equation

$$\sum_{g \in G(n)} p_{g,c} + \sum_{k \in K(n+)} p_{k,c} - \sum_{k \in K(n-)} p_{k,c} = \sum_{d \in D(n)} p_{d,c}$$
$$p_{d,c} + p_{sd,c} = P_d \cdot St_d$$
$$p_{sd,c} = 0 \cdot if \ \ St_d == 0 \ \cdot for \ \ d$$
$$p_{sd,c} = 0 \cdot \ if \ P_d < 0 \ \cdot \ for \ \ d$$
$$p_{sd,c} \geq 0 \cdot \ if \ St_d \cdot P_d \geq 0 \ \cdot \ for \ \ d$$

2) B-theta power flow equation

If $St_k == 0$, then

$$p_{k,c} = 0$$

If $k = c$, then,

$$p_{c,c} = 0$$

If $k \neq c$, then,

$$p_{k,c} = (\delta_{n(k-),c} - \delta_{n(k+),c} + \alpha_k)/x_k$$

3) Branch flow limit

$$-RateC_k - sv\_RATE_k \leq p_{k,c} \leq RateC_k + sv\_RATE_k$$

4) Generator limits

If $St_g == 0$, then

$$p_{g,c} = 0$$

If $St_g == 1$, then

$$p_{g,c} \leq P_{g,max} + sv\_P_{g,max}$$
$$p_{g,c} \geq P_{g,min} - sv\_P_{g,min}$$
$$p_{g,c} - p_g \leq SRR_g \cdot T_{SR} + sv\_SRR_g$$
$$p_{g,c} - p_g \geq -SRR_g \cdot T_{SR} - sv\_SRR_g$$

5) Interface limits

$$\sum_{k \in K(i)} p_{k,c} \leq Rate_i$$

7 R-SCED

7.1 Introduction

R-SCED is a SCED for the real-case. The real case for SCED consists of 9 data files. They are listed below.

- power flow (PSS/E raw) file
- bid data (unit bid data, operation related data as well as market related data)
- cost curve and ramping rate file (contain information of energy ramping rate, spinning ramping rate, and cost curve)
- Hourly unit data: updated unit data for each operational hour. If contradiction (e.g. different unit economic maximum output) exists between bid data file and this file, use this one.
- Interface limit (thermal constraints)
- local_as_service (reserve requirements)
- AOL file (contains the dFAX data)

- Rt unit status fix (contain information of the unit schedule status). "U" means that the associated schedule is not available.
- Units.csv (converted from units.xls file). It is used to match unit name and unit ID.

Hourly unit data file will provide unit regulation qualification status and spinning reserve qualification status (not all units are qualified for providing regulation reserve or spinning reserve service), regulation offer MW, spinning reserve offer MW, hourly updated economic maximum output and economic minimum output.

Similar to G-SCED, there is no unit commitment in R-SCED model, which means that it will only not change the status of any unit. R-SCED only re-dispatch the generators that have a cost curve available, and fixes the outputs of all other generators. Energy ramping rate is used to restrict the change in generator output between the starting time and ending time of the same period. Spinning ramping rate is used to restrict the generator output change for the situations when a contingency occurs. Some *N*-1 contingency constraints are represented explicitly. Slack variables are used to make sure the problem is feasible.

7.2 Procedure to generate pyomo-format based input data file

a) Load configure file.
b) Load original power flow file (fixed format of PSS/E version 26 raw file).
c) Load original interface file.
d) Load original AOL file.
e) Load original reserve requirements file.
f) Load original bid data file.
g) Load original hourly data file.
h) Load original Rt unit status file.
i) Load original cost curve and ramping rate file.
j) Load original units file.
k) Use the related parameters (like "@needHeading") from configure file.
l) Then, write needed bus data.
m) Write needed load data.
n) Write needed generator data.

o) Write needed generator cost curve data.

p) Write needed branch data.

q) Write needed reserve requirements data.

r) Write needed branch thermal limit constraints data.

Note that the order of loading data files may not matter.

7.3 Mathematical model

Sets

A:	Areas.
D:	Loads.
N:	Buses.
K:	Branches.
K(i):	Branches that form the interface *i*.
KM:	Branches under monitor for base case.
KM(c):	Branches for monitor under contingency c.
G:	Generators.
G(a):	Units in area a.
GC:	Dispatchable units that have available cost curve data.
GR:	Units qualified for regulation reserve.
GS:	Units qualified for spinning reserve.

Parameters

T_{ED}:	Look-ahead time for one period RT SCED.
T_{RR}:	Time for regulation reserve requirements.
T_{SR}:	Time for spinning reserve requirements.
T_{PR}:	Time for primary reserve requirements.
$T_{PR,g}$:	Time required for unit *g* to start providing reserve.
$F_{PR,g}$:	1 indicates unit *g* is an off-line fast start unit.
P_d:	Active load of load *d*.
St_g:	Status of unit *g*.
St_d:	Status of load *d*.
P_{g0}:	Total initial output of unit *g*.
NS_g:	Number of cost segments for unit *g*.

$BS_{g,i}$:	Breadth of segment i for unit g.
MRR_g:	Energy ramp rate for unit g.
SRR_g:	Spinning ramp rate for unit g.
RR_a:	Regulation reserve requirement for area a.
SR_a:	Spinning reserve requirement for area a.
PR_a:	Primary reserve requirement for area a.
$RR_{g,o}$:	Regulation reserve amount that unit g offer.
$SR_{g,o}$:	Spinning reserve amount that unit g offer.
$C_{g,i}$:	Cost for segment i of unit g.
CRR_g:	Regulation reserve price that unit g offer.
CSR_g:	Spinning reserve price that unit g offer.
$RateA_k$:	Normal flow limit of branch k.
$RateC_k$:	Emergency flow limit of branch k.
$P_{g,max}$:	Maximum output for unit g.
$P_{g,min}$:	Minimum output for unit g.

Variables

p_k:	Flow on branch k.
p_g:	Total output of generator g.
$p_{g,i}$:	Output on segment i for generator g.
rr_g:	Regulation reserve that generator g provides.
sr_g:	Spinning reserve that generator g provides.
pr_g:	Primary reserve that generator g provides.
p_i:	Total flow for interface i.
$p_{g,c}$:	Total output of generator g under contingency c.
$p_{k,c}$:	Flow on branch k under contingency c.
$p_{i,c}$:	Total flow for interface i under contingency c.

Slack variables (non-negative)

sv_rr_g:	Slack variable for ramping rate limit associated to regulation reserve.
sv_sr_g:	Slack variable for ramping rate limit associated to spinning reserve.
sv_pr_g:	Slack variable for ramping rate limit associated to primary reserve.
sv_RATE_k:	Slack variable for the flow limit of branch k.
$sv\_P_{g,max}$:	Slack variable for parameter $P_{g,max}$.

$sv\_P_{g,min}$:	Slack variable for parameter $P_{g,min}$.
$sv\_MRR_{g,u}$:	Slack variable for parameter MRR_g - ramping up rate relaxation.
$sv\_MRR_{g,d}$:	Slack variable for parameter MRR_g - ramping down rate relaxation.
$sv\_SRR_{g,u}$:	Slack variable for parameter SRR_g - ramping up rate relaxation.
$sv\_SRR_{g,d}$:	Slack variable for parameter SRR_g - ramping down rate relaxation.
sv_RATE_i:	Slack variable for the total flow limit of interface *i*.
sv_rrReq_a:	Slack variable for regulation reserve requirement of area a.
sv_srReq_a:	Slack variable for spinning reserve requirement of area a.
sv_prReq_a:	Slack variable for primary reserve requirement of area a.
$sv\_rrOfferMW_g$:	Slack variable for regulation reserve MW offer limit.
$sv\_srOfferMW_g$:	Slack variable for spinning reserve MW offer limit.

Penalty factors

PF_P_{max}:	A fix penalty factor for variable $sv\_P_{g,max}$.
PF_P_{min}:	A fix penalty factor for variable $sv\_P_{g,min}$.
PF_rr:	A fix penalty factor for variable sv_rr_g.
PF_sr:	A fix penalty factor for variable sv_sr_g.
PF_pr:	A fix penalty factor for variable sv_pr_g.
$PF\_rrOfferMW$:	A fix penalty factor for variable $sv\_rrOfferMW_g$.
$PF\_srOfferMW$:	A fix penalty factor for variable $sv\_srOfferMW_g$.
PF_MRR_u:	A fix penalty factor for variable $sv\_MRR_{g,u}$.
PF_MRR_d:	A fix penalty factor for variable $sv\_MRR_{g,d}$.
PF_SRR_u:	A fix penalty factor for variable $sv\_SRR_{g,u}$.
PF_SRR_d:	A fix penalty factor for variable $sv\_SRR_{g,d}$.
PF_RATEK:	A fix penalty factor for variable sv_RATE_k.
PF_RATEI:	A fix penalty factor for variable sv_RATE_i.
PF_sv_rrReq:	A fix penalty price for variable sv_rrReq_a.
PF_sv_srReq:	A fix penalty price for variable sv_srReq_a.
PF_sv_prReq:	A fix penalty price for variable sv_prReq_a

Formulation

Objective function

$$\begin{aligned} min \ & \sum_{g\in G}\sum_{i=1}^{NS_g} P_{g,i}\cdot C_{g,i} + \sum_{g\in G} RR_g\cdot CRR_g + \sum_{g\in G} SR_g\cdot CSR_g + PF\_P_{max}\sum_{g\in G} sv\_P_{g,max} + PF\_P_{min}\sum_{g\in G} sv\_P_{g,min} \\ & + PF\_rr\sum_{g\in G} sv\_rr_g + PF\_sr\sum_{g\in G} sv\_sr_g + PF\_pr\sum_{g\in G} sv\_pr_g + PF\_sv\_rrReq\sum_{a\in A} sv\_rrReq_a \\ & + PF\_sv\_srReq\sum_{a\in A} sv\_srReq_a + PF\_sv\_prReq\sum_{a\in A} sv\_prReq_a \\ & + PF\_rrOfferMW\sum_{g\in G} sv\_rrOfferMW_a + PF\_srOfferMW\sum_{g\in G} sv\_srOfferMW_a \\ & + PF\_MRR_u\sum_{g\in G} sv\_MRR_{g,u} + PF\_MRR_d\sum_{g\in G} sv\_MRR_{g,d} + PF\_SRR_u\sum_{g\in G} sv\_SRR_{g,u} \\ & + PF\_SRR_d\sum_{g\in G} sv\_SRR_{g,d} + PF\_RATEK\sum_{k\in KM+KM(c)} sv\_RATE_k + PF\_RATEI\sum_{i\in I} sv\_RATE_i \end{aligned}$$

Constraints

1) Power balance equation

Base case

$$\sum_{g\in G} p_g = \sum_{d\in D} P_d St_d$$

Contingency case

$$\sum_{g\in G} p_{g,c} = \sum_{d\in D} P_d St_d$$

2) Branch flow limit

Base case

$$-RateA_k - sv\_RATE_k \le p_k \le RateA_k + sv\_RATE_k\,, \qquad k\in KM$$

Contingency case

$$-RateC_k - sv\_RATE_k \le p_{k,c} \le RateC_k + sv\_RATE_k \quad k\in \mathrm{KM(c)}$$

3) Branch flow calculation

Base case

$$p_k = P_{k0} + \sum_{g\in G} DFAX_{g,k}\cdot \Delta p_g \quad ,k\in KM$$

$$\Delta p_g = p_g - P_{g0}.$$

Contingency case

$$p_{k,c} = P_{k0} + \sum_{g\in G} DFAX_{g,k,c}\cdot \Delta p_{g,c}\,, \qquad k\in KM(c)$$

$$\Delta p_{g,c} = p_{g,c} - P_{g0}.$$

4) Unit generation equation

If $St_g == 0$, then, $p_g = 0$.

If $St_g == 1$ and $g \in (G - GC)$, then, $p_g = P_{g0}$.

If $St_g == 1$ and $g \in GC$ and $P_{g0} \leq 0$, then: $p_{g,i} = 0$

If $St_g == 1$ and $g \in GC$ and $P_{g0} > 0$, then:

$$p_g = \sum_{i=1}^{NS_g} p_{g,i} \ , \qquad g \in GD$$

$$0 \leq p_{g,i} \leq BS_{g,i} \ , \ g \in GD.$$

5) Ramping rate limit

If $St_g == 0$, then,

$$sv\_MRR_{g,u} = 0$$

$$sv\_MRR_{g,D} = 0$$

$$sv\_SRR_{g,u} = 0$$

$$sv\_SRR_{g,D} = 0$$

If $St_g == 1$, then,

$$p_g - P_{g0} \leq MRR_g \cdot T_{ED} + sv\_MRR_{g,u}$$

$$p_g - P_{g0} \geq -MRR_g \cdot T_{ED} - sv\_MRR_{g,d}$$

$$p_{g,c} - p_g \leq SRR_g \cdot T_{SR} + sv\_SRR_{g,u}$$

$$p_{g,c} - p_g \geq -SRR_g \cdot T_{SR} - sv\_SRR_{g,d}$$

6) Reserve limit

If $St_g == 0$, then,

$$sv\_rr_g = 0$$

$$sv\_sr_g = 0$$

$$sv\_pr_g = 0$$

$$rr_g = 0$$

$$sr_g = 0$$

$$pr_g = 0$$

If $St_g == 1$, then,

$$pr_g \geq sr_g$$

Author: *Xingpeng Li*, initially documented in 2016, last updated in April 2023.
Website: https://rpglab.github.io/people/Xingpeng-Li/
Email: xplipower@gmail.com

$$0 \le rr_g \le SRR_g \cdot T_{RR} \cdot St_g + sv\_rr_g$$

$$0 \le sr_g \le SRR_g \cdot T_{SR} \cdot St_g + sv\_sr_g$$

$$0 \le pr_g \le SRR_g \cdot (T_{PR} \cdot St_g + (T_{PR} - T_{PR,g}) \cdot F_{PR,g} \cdot (1 - St_g)) + sv\_pr_g$$

If $St_g == 1$ and $g \in (G - GC)$, then, $rr_g = 0$ and $sr_g = 0$.

If $St_g == 1$ and $g \in (G - GR)$, then, $rr_g = 0$.

If $St_g == 1$ and $g \in GR \; or \; g \in (G - GS)$, then, $sr_g = 0$.

If regulation reserve is not counted as primary reserve, then, we need the following constraint:

If $St_g == 1$ and $g \in GR$ and $g \in GC$, then, $pr_g = 0$.

7) Generation limits

If $St_g == 0$, then,

$$sv\_P_{g,max} = 0$$

$$sv\_P_{g,min} = 0$$

$$p_{g,c} = 0$$

If $St_g == 0$ and $P_{g,max} \ge 0$, then,

$$pr_g \le P_{g,max} + sv\_P_{g,max}$$

If $St_g == 0$ and $P_{g,max} < 0$, then,

$$pr_g \le sv\_P_{g,max}$$

If $St_g == 1$, then,

$$p_g \le P_{g,max} + sv\_P_{g,max}$$

$$p_g \ge P_{g,min} - sv\_P_{g,min}$$

$$p_g + rr_g \le P_{g,max} + sv\_P_{g,max}$$

$$p_g + sr_g \le P_{g,max} + sv\_P_{g,max}$$

$$p_g + pr_g \le P_{g,max} + sv\_P_{g,max}$$

$$p_{g,c} \le P_{g,max} + sv\_P_{g,max}$$

$$p_{g,c} \ge P_{g,min} - sv\_P_{g,min}$$

8) Reserve requirements

$$\sum_{g \in G(a)} rr_g \ge P_{RR,a} - sv\_rrReq_a \,, a \in A$$

$$\sum_{g \in G(a)} sr_g \ge P_{SR,a} - sv\_srReq_a \,, a \in A$$

$$\sum_{g \in G(a)} pr_g \ge P_{PR,a} - sv\_prReq_a \,, a \in A$$

9) Offer constraints

$$rr_g \leq RR_{g,o} - sv\_rrOfferMW_g$$
$$sr_g \leq SR_{g,o} - sv\_srOfferMW_g$$

10) Additional constraints

$$P_{gc} = P_g\,, \quad g \in (G - GC)$$
$$P_{gc} = P_g\,, \quad if\ "contingency"\ c\ is\ actually\ base\ case.$$

11) Interface limits

$$\sum_{k \in K(i)} p_k \leq Rate_i$$

This type of constraint can be used for both base case and contingency case, as long as DFAX data are available.

8 Specific data explanation

Flat cost curve (Fig. 2.a) v.s. slope cost curve (Fig. 2.b).

For generator with flat cost curve, the program assigns each block a variable $p_{g,i}$ indicating how much power the system requires from the associated block of that generator. For the same generator, the optimal solution will not select a block if any other block with a lower price is not entirely selected since the objective function is to minimize the total cost.

For generator with slope cost curve, the objective of representing the total cost is no longer linear. Thus, to use linear programming technique and to make the problem simple and easy to solve, linearization is needed. In this manual, the linearization is made for each slope curve after the first flat segment.

How accurate can the linearization be depends on the parameter @blockPrice. Assume that the lowest price and highest price for one slope curve is P_1 and P_2, and the associated interval is between $MW_1$and MW_2. Then, the linearized cost curves can be obtained by the following procedure.

a) Calculate number of flat segments that will be generated, $n = ceil((P_2 - P_1)/deltaP)$
where, $deltaP$ is the same with input parameter @blockPrice, ceil(*numberA*) denotes the lowerest interger number is greater than or equal to *numberA*.

b) Each segment shares the same breadth: $bw = (MW_2 - MW_1)/n$

Author: *Xingpeng Li*, initially documented in 2016, last updated in April 2023.
Website: https://rpglab.github.io/people/Xingpeng-Li/
Email: xplipower@gmail.com

c) Calculate the initial slope: $slope = (P_2 - P_1)/(MW_2 - MW_1)$

d) The price for each segment is $Ps_i = P_1 + (i - 0.5) \cdot \frac{bw}{slope},\ where\ i = 1..n$

Thus, a sloped cost curve segment is converted to a flat cost curve with n segments. The precision of the linearization largely depends on the value of parameter @blockPrice.

The following example is used to explain the flat cost curve. As shown in Fig. 2 (a), there are three segments. The variable $p_{g,i}$ used in both G-SCED and R-SCED only indicates the power output that is from segment i. Assume the three segments in Fig. 2(a) have breadth of 10 MW, 10 MW, and 15 MW. Then, $0 \le p_{g,1} \le 10$, $0 \le p_{g,2} \le 10$, $0 \le p_{g,3} \le 15$, and the total output of generator g is $p_g = p_{g,1} + p_{g,2} + p_{g,3}$, and $p_{g,min} \le p_g \le p_{g,max}$ if generator g is online.

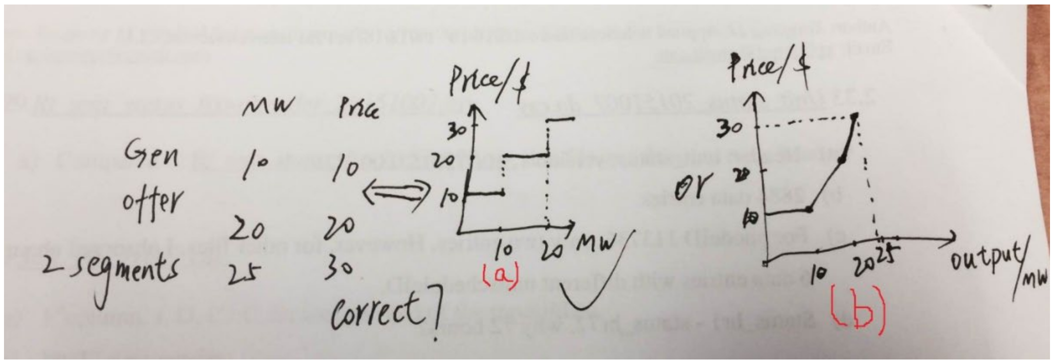


Fig. 2 Unit cost curve

The program does not generate interface limit data and contingency data for G-SCED, however, G-SCED has the capability to model those types of constraints if the associated data is available.

The program does not generate interface limit data for R-SCED, however, R-SCED has the capability to model that type of constraints if the associated data is available.

In real-case, set SCENARIO actually means contingency scenario (contingency-case); set CONTINGENCY works as the set CONTINGENCYLINE for generic-case. The set CONTINGENCY works in such a way so that the SCED can model a contingency that consists of multiple lines. Note that SCENARIO can be either the base case or a contingency case, which can be determined by the parameter @isBaseCase (1 denotes base case, 0 denotes contingency case).

In real-case, if parameter @Contingency_branchIdx has a value of -1, it means that the contingency line cannot be found or cannot be matched due to name inconsistency. Or, if parameter @Contingency_branchIdx has a value of -2, it means that the contingency line does not exist since it is not a contingency, which means "ACTUAL" congestion is happening.

When generate separate generator data without running SCED, the unitScheduleID will be generated and it is an integer number; some details are listed below:

# unitScheduleID: -1, the unitID is -1
# unitScheduleID: -2, the unitID cannot be found in the bid_data
# unitScheduleID: -3, the unitID is found in the bid_data, however, the schedule is not available per rt_unit_status_fix_20160409

9 Outputs

One data file generated by the "run" program is "*genData_Actually.txt*". It contains the generator data that are used by the solver.

The file "WriteResults.py" is a subroutine that is to dump desired results to files on disk. If there are files with the same name on disk; then, they will be deleted first and then be created by the code.

results_summary.txt shows the objective values, load shedding, and all non-zero slack variables.

results_load.txt lists all the load data, which is actually used by the solver. Load may be adjusted by a simple factor to represent the system loss.

10 Others

Note that, if you do not have a specific set of data, for instance, no interface limit, then, just do not have any data related to interface limit in the input data file; just like no such type of constraint modeled in the problem.

As shown below, a very simple trick is used to represent the system loss.

$$f = \frac{\sum_g P_{g0} \cdot St_g}{\sum_d P_d \cdot St_d}$$

$$P_{d\_adjusted} = P_{d\_initial}\,, \quad d \in D$$

11 References

[1] Xingpeng Li and Kory W. Hedman, "Enhanced Energy Management System with Corrective Transmission Switching Strategy— Part I: Methodology," *IEEE Transactions on Power Systems*, vol. 34, no. 6, pp. 4490-4502, Nov. 2019.

[2] Xingpeng Li and Kory W. Hedman, "Enhanced Energy Management System with Corrective Transmission Switching Strategy— Part II: Results and Discussion," *IEEE Transactions on Power Systems*, vol. 34, no. 6, pp. 4503-4513, Nov. 2019.