Exascale Grid Optimization Toolkit (ExaGO$^{\text{TM}}$) User Manual

****—Draft—****

March 15, 2021

Contents

Chapter 1

Introduction

Exascale Grid Optimization (`ExaGO` TM) is an open source package for solving large-scale power grid optimization problems on parallel and distributed architectures, particularly targeted for exascale machines with heteregenous architectures (GPU). All `ExaGO` TMapplications use a nonlinear formulation based on full AC optimal power flow. Figure 1.1 shows the different applications available with `ExaGO` TMspanning the dimensions of security (contingencies), stochasticity, and time.

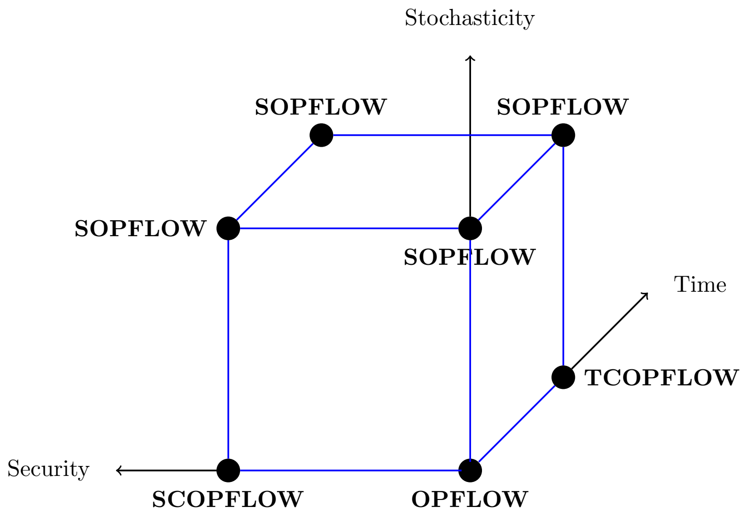

Figure 1.1: `ExaGO` provides applications along the dimensions of security (contingencies), time, and stochasticity. The label vertices denote different `ExaGO` applications available.

The different applications available with `ExaGO` are listed in Table 1.1

ExaGOTMis written in C/C++ and makes heavy use of the functionality provided by the PETSc[1] library. It uses RAJA [3]and Umpire [4]libraries for execution on the GPU. It makes use of several optimization solvers - Ipopt [7], HiOp [6, 5], and TAO [1].

While `ExaGO` is targeted for making use of distributed computing environments and GPUs, its full support is still under development. TCOPFLOW can only run on a single CPU process. SCOPFLOW and SOPFLOW can run in parallel on multiple CPU processes, but they can only execute independent ACOPFs for contingencies and scenarios. Solving the full SCOPF and Stochastic OPF problem in a parallel setting is under development. OPFLOW execution on the GPU has only been tested with NVIDIA GPUs.

Table 1.1: ExaGO applications

Application	**Description**	**Notes**
OPFLOW	AC optimal power flow	
SCOPFLOW	Security-constrained AC optimal power flow	Uses TCOPFLOW for multi-period contingencies
TCOPFLOW	Multi-period AC optimal power flow	
SOPFLOW	Stochastic AC optimal power flow	Uses SCOPFLOW for multi-contingency scenarios

The support for solving applications on different architectures in given in Table 1.2

Table 1.2: ExaGO application execution on different hardware

Application	**CPU (serial)**	**CPU (parallel)**	**GPU**
OPFLOW	Y	x	Y
SCOPFLOW	Y	Y (embarrasingly parallel)	N
SOPFLOW	Y	Y (embarrasingly parallel)	N
TCOPFLOW	Y	N	N

This document describes ExaGO implementation details, formulation descriptions, and a guide for executing ExaGO applications.

Chapter 2

Getting Started

2.1 System requirements

ExaGO is currently only built on 64b OSX and Linux machines, compiled with GCC >= 7.3. We build ExaGO on Intel and IBM Power9 architectures.

2.2 Prerequisites

This section assumes that you already have the ExaGO source code, and that the environment variable EXAGODIR is the directory of the ExaGO source code. ExaGO may be acquired via the PNNL git repository linked here, like so:

```
> git clone https://gitlab.pnnl.gov/exasgd/frameworks/exago.git exago
> export EXAGODIR=$PWD/exago
```

Paths to installations of third party software in examples are abbreviated with placeholder paths. For example, /path/to/cuda is a placeholder for a path to a valid CUDA Toolkit installation.

2.3 Dependencies

ExaGO has dependencies in table 2.1.

Table 2.1: Dependency Table

Dependency	**Version Constraints**	**Mandatory**	**Notes**
PETSc [1]	>= 3.13.0	✓	Only needed for the setup stage
CMake	>= 3.10	✓	Only a build dependency
MPI	>= 3.1.3		Only tested with openmpi and spectrummpi
Ipopt [7]	>= 3.12		
HiOp [6, 5]	>= 0.3.0		Prefer dynamically linked
RAJA [3]	>= 0.11.0		
Umpire [2]	>= 2.1.0		Only when RAJA is enabled
MAGMA	>= 2.5.2		Only when GPU acceleration is enabled
CUDA Toolkit	>= 10.2.89		Only when GPU acceleration is enabled

These may all be toggled via CMake which will be discussed in the section Building and installation.

2.3.1 Notes on environment modules

Many of the dependencies are available via environment modules on institutional clusters. To get additional information on your institution's clusters, please ask your institution's system administrators. Some end-to-end examples in this document will use system-specific modules and are not expected to expected to run on other clusters.

For example, the modules needed to build and run ExaGO on Newell, an IBM Power9 PNNL cluster, are as follows:

```
> module load gcc/7.4.0
> module load openmpi/3.1.5
> module load cuda/10.2
> module load magma/2.5.2_cuda10.2
> module load metis/5.1.0
> module load cmake/3.16.4
```

2.3.2 Additional Notes on GPU Accelerators

As of January 2021, CUDA is the only GPU accelerator platform ExaGO fully supports. We have preliminary support for HIP, however this is not in any of our main development branches yet. Full HIP support should arrive in early 2021.

2.3.3 Additional Notes on Umpire

Umpire is an implicit dependency of RAJA. If a user enables RAJA, they must also provide a valid installation of Umpire. Additionally, if a user would like to run ExaGO with RAJA and without CUDA, they must provide a CPU-only build of Umpire since an Umpire build with CUDA enabled will link against CUDA.

2.4 Building and installation

2.4.1 Default Build

ExaGO may be built with a standard CMake workflow:

Listing 2.1: Example CMake workflow

```
> cd $EXAGODIR
> export BUILDDIR=$PWD/build INSTALLDIR=$PWD/install
> mkdir $BUILDDIR $INSTALLDIR
> cd $BUILDDIR
> cmake .. -DCMAKE_INSTALL_PREFIX=$INSTALLDIR
> make install
```

Following sections will assume the user is following the basic workflow outlined above.

Note: For changes to the CMake configuration to take effect, the code wil have to be rebuilt.

2.4.2 Additional Options

To enable additional options, CMake variables may be defined via CMake command line arguments, ccmake, or cmake-gui. CMake options specific to ExaGO have an EXAGO_ prefix. For example, the following shell commands will build ExaGO with MPI:

```
> cmake .. -DCMAKE_INSTALL_PREFIX=$INSTALLDIR -DEXAGO_ENABLE_MPI=ON
```

ExaGO's CMake configuration will search the usual system locations for an MPI installation.

For dependencies not installed to a system-wide location, users may also directly specify the location of a dependency. For example, this will build ExaGO with IPOPT enabled and installed to a user directory:

```
> cmake .. \
    -DCMAKE_INSTALL_PREFIX=$INSTALLDIR \
    -DEXAGO_ENABLE_IPOPT=ON \
    -DIPOPT_DIR=/path/to/ipopt
```

Notice that the CMake variable IPOPT_DIR does not have an EXAGO_ prefix. This is because the variables specifying locations often belong to external CMake modules. CMake variables indicating installation directories do not have an EXAGO_ prefix.

Some CMake options effect others. This is especially common when the user enables ExaGO's GPU options. For example, if the user enables EXAGO_ENABLE_GPU and EXAGO_ENABLE_RAJA, the user must provide a GPU-enabled RAJA installation. Umpire is also an implicit dependency of RAJA, so if the user enables EXAGO_ENABLE_GPU they must **also** provide a GPU-enabled Umpire installation.

Below is a complete shell session on PNNL's cluster Newell in which a more complicated ExaGO configuration is built, where each dependency installation is explicitly passed to CMake. Environment modules specific to Newell are provided to make the example thorough, even though they are not likely to work on another machine.

Listing 2.2: ExaGO build with all options enabled

```
> module load gcc/7.4.0
> module load cmake/3.16.4
> module load openmpi/3.1.5
> module load magma/2.5.2_cuda10.2
> module load metis/5.1.0
> module load cuda/10.2
> git clone https://gitlab.pnnl.gov/exasgd/frameworks/exago.git exago
> export EXAGODIR=$PWD/exago
> cd $EXAGODIR
> export BUILDDIR=$PWD/build INSTALLDIR=$PWD/install
> mkdir $BUILDDIR $INSTALLDIR
> cd $BUILDDIR
> cmake .. \
  -DCMAKE_INSTALL_PREFIX=$INSTALLDIR \
  -DCMAKE_BUILD_TYPE=Debug \
  -DEXAGO_ENABLE_GPU=ON \
  -DEXAGO_ENABLE_HIOP=ON \
  -DEXAGO_ENABLE_IPOPT=ON \
  -DEXAGO_ENABLE_MPI=ON \
  -DEXAGO_ENABLE_PETSC=ON \
  -DEXAGO_RUN_TESTS=ON \
  -DEXAGO_ENABLE_RAJA=ON \
  -DEXAGO_ENABLE_IPOPT=ON \
  -DIPOPT_DIR=/path/to/ipopt \
  -DRAJA_DIR=/path/to/raja \
  -Dumpire_DIR=/path/to/umpire \
  -DHIOP_DIR=/path/to/hiop \
  -DMAGMA_DIR=/path/to/magma \
  -DPETSC_DIR=/path/to/petsc
> make -j 8 install
> # For the following commands, a job scheduler command may be needed.
> # Run test suite
> make test
> # Run an ExaGO application:
> $INSTALLDIR/bin/opflow
```

2.5 Usage

Each `ExaGO` application has the following format for execution

```
  ./app <app_options>
```

Here, `app_options` are the command line options for the application. Each application has many options through which the input files and the control options can be set for the application. All application options have the form `-app_option_name` followed by the `app_option_value`. For instance,

```
  ./opflow -netfile case9mod.m -opflow_model POWER_BALANCE_POLAR -
     ↪ opflow_solver IPOPT
```

will execute OPFLOW application using `case9mod.m` input file with the model `POWER_BALANCE_POLAR` ↪ and Ipopt [7]solver.

Options can also be passed to each application through `-options_file`, or through a combination of the command line and options file. The configuration specified last on the command line overrides any previous options. For example, if `-options_file opflowoptions` specified `-netfile case9mod.m` within it's settings:

```
  ./opflow -netfile case118.m -options_file opflowoptions # Uses case9mod.
     ↪ m
  ./opflow -options_file opflowoptoins -netfile case118.m # Uses case118.m
```

If no options file is specified through the command line, ExaGO applications will attempt to locate the default options file for a given application in `.,./options,<install_dir>/options`.

Chapter 3

Optimal power flow (OPFLOW)

OPFLOW solves the full AC optimal power flow problem and provides various flexible features that can be toggled via run-time options. It has interfaces to different optimization solvers and includes different representations of the underlying equations (power-balance-polar and power-balance-cartesian) that can be used. By selecting the appropriate solver, OPFLOW can be executed on CPU (in serial or parallel) or on GPUs.

3.1 Formulation

Optimal power flow is a general nonlinear programming problem with the following form

$$\begin{aligned}
\text{min. } & f(x) && (3.1)\\
& \text{s.t.} && \\
& g(x) = 0 && (3.2)\\
& h(x) \leq 0 && (3.3)\\
& x^{\text{min}} \leq x \leq x^{\text{max}} && (3.4)
\end{aligned}$$

Here, x are the decision variables with lower and upper bounds x^{min} and x^{max}, respectively, $f(x)$ is the objective function, $g(x)$ and $h(x)$ are the equality and inequality constraints, respectively. In the following sections we describe what constitutes these different terms as used by OPFLOW.

3.1.1 Variables and bounds

The different variables used in OPFLOW formulation are described in Table 3.1.

Power imbalance variables are non-physical (slack) variables that measure the violation of power balance at buses. Having these variables (may) help in making the optimization problem easier to solve since they always ensure feasibility of the bus power balance constraints.

3.1.2 Objective Function

The objective function for OPFLOW is given in (3.5)

$$\text{min. } C(p^{\text{g}}) + C(\Delta p^{\text{g}}) + C(\Delta p^{\text{l}}, \Delta q^{\text{l}}) + C(\Delta p, \Delta q) \tag{3.5}$$

Table 3.1: Optimal power flow (OPFLOW) variables

Variable	**Symbol**	**Bounds**	**Notes**
Generator real power dispatch	p_j^{g}	$p_j^{\mathrm{gmin}} \le p_j^{\mathrm{g}} \le p_j^{\mathrm{gmax}}$	
Generator reactive power dispatch	q_j^{g}	$q_j^{\mathrm{gmin}} \le q_j^{\mathrm{g}} \le q_j^{\mathrm{gmax}}$	
Generator real power deviation	Δp_j^{g}	$-p_j^{\mathrm{r}} \le \Delta p_j^{\mathrm{g}} \le p_j^{\mathrm{r}}$	• Only used when `-opflow_has_gensetpoint` or `-opflow_use_agc` option is active • Δp_j^{g} is the deviation from real power generation setpoint p_j^{gset}.
Generator real power setpoint	p_j^{gset}	$p_j^{\mathrm{gmin}} \le p_j^{\mathrm{gset}} \le p_j^{\mathrm{gmax}}$	• Only used when `-opflow_has_gensetpoint` or `-opflow_use_agc` option is active.
System power excess/deficit	ΔP	Unbounded	Only used when `-opflow_use_agc` is active
Bus voltage angle	θ_i	$-\pi \le \theta_i \le \pi$	• Used with power balance polar model (`-opflow_model POWER_BALANCE_CARTESIA` • θ_i is unbounded, except reference bus angle θ_i^{ref} which is fixed to 0
Bus voltage magnitude	v_i	$v_i^{\mathrm{min}} \le v_i \le v_i^{\mathrm{max}}$	• Used with power balance polar model (`-opflow_model POWER_BALANCE_POLAR`) • $v_i^{\mathrm{min}} = v_i^{\mathrm{max}} = v_i^{\mathrm{set}}$ if fixed generator set point voltage option is active (`-opflow_has_gensetpoint`)
Bus voltage real part	e_i	$-v_i^{\mathrm{max}} \le e_i \le v_i^{\mathrm{max}}$	Used with power balance cartesian model (`-opflow_model POWER_BALANCE_CARTESIAN`)
Bus voltage imaginary part	f_i	$-v_i^{\mathrm{max}} \le f_i \le v_i^{\mathrm{max}}$	Used with power balance cartesian model (`-opflow_model POWER_BALANCE_CARTESIAN`)
Bus real power mismatch	Δp_i	Unbounded	Used when power mismatch variable option is active (`-opflow_include_powerimbalance_variables1`
Bus reactive power mismatch	Δq_i	Unbounded	Used when power mismatch variable option is active (`-opflow_include_powerimbalance_variables1`
Real power load loss	Δp_j^{l}	$0 \le \Delta p_j^{\mathrm{l}} \le p_j^{\mathrm{l}}$	Used when load loss variable option is active (`-opflow_include_loadloss_variables1`)
Reactive power load loss	Δq_j^{l}	$0 \le \Delta q_j^{\mathrm{l}} \le q_j^{\mathrm{l}}$	Used when load loss variable option is active (`-opflow_include_loadloss_variables1`)

Total generation cost $C(p^{\mathrm{g}})$

Needs `-opflow_objective MIN_GEN_COST`

$$C(p^{\mathrm{g}}) = \sum_{j \in J^{\mathrm{gen}}} C^{\mathrm{g}}_j(p^{\mathrm{g}}_j) \tag{3.6}$$

Here, C^{g}_j is a quadratic function of the form $C^{\mathrm{g}}_j = a^{\mathrm{g}}_j {p^{\mathrm{g}}_j}^2 + b^{\mathrm{g}}_j p^{\mathrm{g}}_j + c^{\mathrm{g}}_j$.

Total generation setpoint deviation $C(\Delta p^{\mathrm{g}})$

Needs `-opflow_objective MIN_GENSETPOINT_DEVIATION` option

$$C(\Delta p^{\mathrm{g}}) = \sum_{j \in J^{\mathrm{gen}}} (\Delta {p^{\mathrm{gu}}_j}^2) \tag{3.7}$$

Load loss $C(\Delta p^{\mathrm{l}}, \Delta q^{\mathrm{l}})$

This term gets added to the objective when `-opflow_include_loadloss_variables` option is active.

$$C(\Delta p^{\mathrm{l}}, \Delta q^{\mathrm{l}}) = \sum_{j \in J^{\mathrm{ld}}} \sigma^{\mathrm{l}}_j (\Delta {p^{\mathrm{l}}_j}^2 + \Delta {q^{\mathrm{l}}_j}^2) \tag{3.8}$$

The load loss penalty σ^{l}_j can be set via the option `-opflow_loadloss_penalty`. The default is \$100/MW for all loads.

Power imbalance $C(\Delta p, \Delta q)$

This term gets added to the objective when `-opflow_include_powerimbalance_variables` option is active.

$$C(\Delta p, \Delta q) = \sum_{i \in J^{bus}} \sigma_i (\Delta {p_i}^2 + \Delta {q_i}^2) \tag{3.9}$$

The power imbalance cost σ_i can be set via the option `-opflow_powerimbalance_penalty`. The default is \$1000/MW for all buses.

3.1.3 Equality constraints

Nodal power balance

$$\sum_{\substack{j \in J^{\mathrm{gen}} \\ A^{\mathrm{g}}_{ij} \neq 0}} p^{\mathrm{g}}_j = p^{\mathrm{sh}}_i + \sum_{\substack{j \in J^{\mathrm{ld}} \\ A^{\mathrm{l}}_{ij} \neq 0}} p^{\mathrm{l}}_j + \sum_{\substack{j \in J^{\mathrm{br}} \\ A^{\mathrm{br}}_{oi} \neq 0}} p^{\mathrm{br}}_{j_{od}} + \sum_{\substack{j \in J^{\mathrm{br}} \\ A^{\mathrm{br}}_{id} \neq 0}} p^{\mathrm{br}}_{j_{do}} \tag{3.10}$$

$$\sum_{\substack{j \in J^{\mathrm{gen}} \\ A^{\mathrm{g}}_{ij} \neq 0}} q^{\mathrm{g}}_j = q^{\mathrm{sh}}_i + \sum_{\substack{j \in J^{\mathrm{ld}} \\ A^{\mathrm{l}}_{ij} \neq 0}} q^{\mathrm{l}}_j + \sum_{\substack{j \in J^{\mathrm{br}} \\ A^{\mathrm{br}}_{oi} \neq 0}} q^{\mathrm{br}}_{j_{od}} + \sum_{\substack{j \in J^{\mathrm{br}} \\ A^{\mathrm{br}}_{id} \neq 0}} q^{\mathrm{br}}_{j_{do}} \tag{3.11}$$

$$\tag{3.12}$$

where, the real and reactive power shunt consumption is given by (3.19) and (3.20)

The real and reactive power flow $p^{\mathrm{br}}_{j_{od}}$, $q^{\mathrm{br}}_{j_{od}}$ for line j from the origin bus o to bus d is given by (3.21) – (3.22) and from destination bus d to origin bus o is given by (3.23) – (3.24)

Generator real power output

When using `-opflow_has_gensetpoint`, two extra variables p_j^{gset} and Δp_j^{g} are added for each generator. The generator real power output p_j^{g} is related to the power deviation Δp_j^{g} by the following relations

$$p_j^{\text{gset}} + \Delta p_j^{\text{g}} - p_j^{\text{g}} = 0 \tag{3.13}$$

$$p_j^{\text{gset}} - p_j^{\text{g}^*} = 0 \tag{3.14}$$

Here, $p_j^{\text{g}^*}$ is the set-point for the generator real power output, which can be thought of as an operator set or contractual agreement set-point.

3.1.4 Inequality constraints

MVA flow on branches

MVA flow limits at origin and destination buses for each line.

$$p_{j_{od}}^{\text{br}\,2} + q_{j_{od}}^{\text{br}\,2} \leq s_j^{\text{rateA}\,2}, \quad j \in J^{\text{br}} \tag{3.15}$$

$$p_{j_{do}}^{\text{br}\,2} + q_{j_{do}}^{\text{br}\,2} \leq s_j^{\text{rateA}\,2}, \quad j \in J^{\text{br}} \tag{3.16}$$

To reduce the size of inequality constraints, only lines that are in service and having MVA A rating s_j^{rateA} less than 10000 MVA are considered.

Automatic generation control (AGC)

With `-opflow_use_agc`, two additional constraints are added for each participating generator to enforce the proportional generator redispatch participation as done in automatic generation control (AGC). These two equations are

$$\begin{aligned} \left(\alpha_j^{\text{g}}\Delta P - \Delta p_j^{\text{g}})\right)\left(p_j^{\text{g}} - p_j^{\text{gmax}}\right) \geq 0 \\ \left((\Delta p_j^{\text{g}} - \alpha_j \Delta P\right)\left(p_j^{\text{gmin}} - p_j^{\text{g}}\right) \geq 0 \end{aligned} \tag{3.17}$$

Eq. 3.17 enforces the generator set-point deviation to be equal to the generation participation when the generator has head-room available $p_j^{\text{gmin}} \leq p_j^{\text{g}} \leq p_j^{\text{gmax}}$. Here, α_j^{g} is the generator participation factor which is the proportion of the power deficit/excess ΔP that the generator provides.

Generator bus voltage control

When the option `-opflow_genbusvoltageFIXED_WITHIN_QBOUNDS` is used, the generator bus voltage is fixed when the total reactive power generation available at the bus is within bounds. When it reaches its bounds, the voltage varies with the generator reactive power fixed at its bound. To implement this behavior, two inequality constraints are added for each generator bus

$$\begin{aligned} (v_i^{\text{set}} - v_i)(q_i - q^{\max_i}) \geq 0 \\ (v_i - v_i^{\text{set}})(q^{\min_i} - q_i) \geq 0 \end{aligned} \tag{3.18}$$

Here, q_i, $q^{\max_i}$, and $q^{\min_i}$ are the generated, maximum, and minimum reactive power at the bus, respectively.

3.2 Models

Table 3.2: OPFLOW models

Model type	OPFLOW option	Compatible solvers	CPU-GPU
Power balance with polar coordinates	`-opflow_model POWER_BALANCE_POLAR`	IPOPT	CPU
Power balance with cartesian coordinates	`-opflow_model POWER_BALANCE_CARTESIAN`	IPOPT, TAO	CPU
Power balance with polar coordinates used with HIOP	`-opflow_model POWER_BALANCE_HIOP`	HiOp [6, 5]	CPU
Power balance with polar coordinates used with HIOP on GPU	`-opflow_model PBPOLRAJAHIOP`	IPOPT, TAO	GPU

3.2.1 Power balance polar

Shunt power

$$p_i^{\text{sh}} = g_i^{\text{sh}} {v_i}^2 \tag{3.19}$$
$$q_i^{\text{sh}} = -b_i^{\text{sh}} {v_i}^2 \tag{3.20}$$

Branch flows

The real and reactive power flow p_{jod}^{br}, q_{jod}^{br} from bus o to bus d on line j is given by (3.21) – (3.22)

$$p_{jod}^{\text{br}} = g_{oo}v_o^2 + v_o v_d(g_{od}\cos(\theta_o - \theta_d) + b_{od}\sin(\theta_o - \theta_d)) \tag{3.21}$$
$$q_{jod}^{\text{br}} = -b_{oo}v_o^2 + v_o v_d(-b_{od}\cos(\theta_o - \theta_d) + g_{od}\sin(\theta_o - \theta_d)) \tag{3.22}$$

and from bus i to bus d is given by (3.23) – (3.24)

$$p_{jdo}^{\text{br}} = g_{dd}v_d^2 + v_d v_o(g_{do}\cos(\theta_d - \theta_o) + b_{do}\sin(\theta_d - \theta_o)) \tag{3.23}$$
$$q_{jdo}^{\text{br}} = -b_{dd}v_d^2 + v_d v_o(-b_{do}\cos(\theta_d - \theta_o) + g_{do}\sin(\theta_d - \theta_o)) \tag{3.24}$$

3.2.2 Power balance with HiOp

Power balance formulation with polar coordinates used with HiOp [6, 5][6] library. This model runs on CPU only, though the HiOp [6, 5]solver can run on GPU.

3.2.3 Power balance with HiOp,RAJA,and Umpire

Power balance formulation with polar coordinates used with HiOp [6, 5]and RAJA [3]. This model uses RAJA [3] [3] and Umpire [4] libraries to run OPFLOW calculations (objective, constraints, etc.) on the GPU.

Table 3.3: OPFLOW Model-solver compatibility

Model	Ipopt [7]	HiOp [6, 5]	TAO [1]
POWER_BALANCE_POLAR	✓		
POWER_BALANCE_CARTESIAN	✓		✓
POWER_BALANCE_HIOP		✓	
PBPOLRAJAHIOP		✓	

3.3 Solvers

OPFLOW can be used with a few different solvers. All the three solvers solve the optimization problem via a nonlinear interior-point algorithm.

1. Ipopt [7] is a popular open-source package for solving nonlinear optimization problems. It is the most robust of the solvers implemented for solving OPFLOW. However, it can be run only on a single process and does not have GPU support. Option:
   `-opflow_solver IPOPT`

2. The Toolkit for Applied Optimization (TAO [1]) is a toolkit available through PETSc [1] for solving optimization problems. TAO includes a interior-point algorithm that is used by OPFLOW. The TAO algorithm is fully parallel, but is less robust compared to IPOPT. Currently, the TAO solver can only used with the power balance cartesian model. Option:
   `-opflow_solver TAO` POWER_BALANCE_CARTESIAN

3. HiOp [6, 5] is a high-performance optimization library that implements an interior-point algorithm for solving nonlinear optimization problems. The library supports execution both on the CPU and the GPU. Options:
   CPU: `-opflow_solver HIOP` -opflow_model POWER_BALANCE_HIOP
   GPU: `-opflow_solver HIOP` -opflow_model PBPOLRAJAHIOP

3.4 Input and Output

The current `ExaGO` version only supports reading network file in MATPOWER format and can (optionally) write the output back in MATPOWER data file format.

3.5 Usage

```
./opflow -netfile <netfilename>  <opflowoptions>
```

3.6 Options

See table 3.4

3.7 Examples

****—To be completed—****

Table 3.4: OPFLOW options

Option	**Meaning**	**Values (Default value)**
-netfile	Network file name	string (case9mod.m)
-print_output	Print output to screen	0 or 1 (0)
-save_output	Save output to file	0 or 1 (0)
-opflow_model	Representation of network balance equations and bus voltages	See Table 3.2 (POWER_BALANCE_POLAR)
-opflow_solver	Optimization solver	See section 3.3
-opflow_initialization	Type of initialization	See Table 3.5 (MIDPOINT)
-opflow_has_gensetpoint	Uses generation set point and activates ramping variables	0 or 1 (0)
-opflow_use_agc -opflow_objective	Uses AGC formulation in OPF type of objective function	0 or 1 (0) See table **??** (MIN_GEN_COST)
-opflow_genbusvoltage	Type of generator bus voltage control	See Table 3.6 (FIXED_WITHIN_QBOUNDS)
-opflow_ignore_lineflow_constraints	Ignore line flow constraints	0 or 1 (0)
-opflow_include_loadloss_variables	Include load loss	0 or 1 (0)
-opflow_include_powerimbalance_variables	Include power imbalance	0 or 1 (0)
-opflow_loadloss_penality	$ penalty for load loss	real (1000)
-opflow_powerimbalance_penalty	$ penalty for power imbalance	real (10000)
-opflow_tolerance	Optimization solver tolerance	real (1e-6)

Table 3.5: OPFLOW initializations

Initialization type	**Meaning**
MIDPOINT	Use mid-point of bounds
FROMFILE	Use values from network file
ACPF	Run AC power flow for initialization
FLATSTART	Flat-start

Table 3.6: OPFLOW generator bus voltage control modes

Voltage control type	**Meaning**
FIXED_AT_SETPOINT	Fixed at given set-point. Reactive power limits are ignored
FIXED_WITHIN_QBOUNDS	Fixed within reactive power bounds
VARIABLE_WITHIN_BOUNDS	Variable within voltage bounds

Table 3.7: OPFLOW objective function types

Objective function	**Meaning**
MIN_GEN_COST	Minimize generation cost
MIN_GENSETPOINT_DEVIATION	Minimize deviation (ramp up-down) from generataor set-point
NO_OBJ	No objecive function

Chapter 4

Multi-period optimal power flow (TCOPFLOW)

TCOPFLOW solves a full AC multi-period optimal power flow problem with the objective of minimizing the total cost over the given time horizon while adhering to intra-time and inter-temporal constraints.

4.1 Formulation

The multi-period optimal power flow problem is a series of optimal power flow problems coupled via temporal constraints. The generator real power deviation ($p^{\mathrm{g}}_{jt} - p^{\mathrm{g}}_{jt-\Delta t}$) constrained within the ramp limits form the temporal constraints. An illustration of the temporal constraints is shown in Fig. 4.1 with four time steps. Each time-step t is coupled with its preceding time $t - \Delta t$, where Δt is the time-step where the objective is to find a least cost dispatch for the given time horizon.

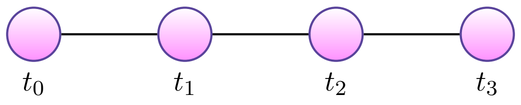

Figure 4.1: Multi-period optimal power flow example with four time-steps. The lines connecting the different time-periods denote the coupling between them.

In general form, the equations for multi-period optimal power flow are given by (4.1) – (4.5). TCOPFLOW solves to minimize the total generation cost $\sum_{t=0}^{N_t-1} f(x_t)$ over the time horizon, where N_t is the number of time-steps. At each time-step, the equality constraints ($g(x_t)$), inequality $h(x_t)$, and the lower/upper limit (x^-,x^+) constraints need to be satisfied. Equation (4.5) represents the coupling between the consecutive time-steps. It is a most common form of coupling that limits the deviation of the real power generation at time t from its preceding time-step $t - \Delta t$ to within its ramping capability $\delta_t x$.

$$
\begin{aligned}
\min \sum_{t=0}^{N_t-1} & f(x_t) && && (4.1)\\
\text{s.t.} & && && \\
g(x_t) &= 0, && t \in [0, N_t-1] && (4.2)\\
h(x_t) &\le 0, && t \in [0, N_t-1] && (4.3)\\
x^- \le x_t &\le x^+, && t \in [0, N_t-1] && (4.4)\\
-\delta_t x \le x_t - x_{t-\Delta t} &\le \delta_t x, && t \in [1, N_t-1] && (4.5)
\end{aligned}
$$

4.2 Solvers

TCOPFLOW solves the coupled multi-period using the solver Ipopt [7]. Currently, `ExaGO` only supports solving TCOPFLOW using Ipopt [7] , which is also the default solver. Solving TCOPFLOW is supported only on single process.

4.3 Input and Output

- **Network file:** The network file describing the network details. Only MATPOWER format files are currently supported.
- **Load data:** One file for load real power and one fo reactive power. The files need to be in CSV format. An example of the format for the 9-bus case is here.
- **Wind generation:** The wind generation time-series described in CSV format. See an example of the format here.

If the load data and/or wind generation profiles are not set then a flat profile is assumed, i.e., the load and wind generation for all hours is constant.

The TCOPFLOW output is saved to a directory named *tcopflowout*. This directory contains N_t files, one for each time-step, in MATPOWER data file format.

4.4 Usage

****—To be completed—****

4.5 Options

See table 4.1 In addition, all OPFLOW options given in Table 3.4 can be used.

4.6 Examples

****—To be completed—****

Table 4.1: TCOPFLOW options

Option	**Meaning**	**Values (Default value)**
-netfile	Network file name	string (case9mod˙gen3˙wind.m)
-save_output	Save output to file	0 or 1 (0)
-tcopflow_pload_profile	Real power load profile	string (load˙P.csv)
-tcopflow_qload_profile	Reactive power load profile	string (load˙Q.csv)
-tcopflow_windgen_profile	Wind generation profile	string (scenarios.csv)
-tcopflow_dT	Length of time-step (minutes)	double (5.0)
-tcopflow_duration	Total duration (hours)	double (0.5)

Chapter 5

Security-constrained optimal power flow (SCOPFLOW)

SCOPFLOW solves a contingency-constrained optimal power flow problem. The problem is set up as a two-stage optimization problem where the first-stage (base-case) represents the normal operation of the grid and the second-stage comprises of N_c contingency scenarios. Each contingency scenario can be single or multi-period.

5.1 Formulation

5.1.1 Single-period

The contingency-constrained optimal power flow (popularly termed as security-constrained optimal power flow (SCOPF) in power system parlance) attempts to find a least cost dispatch for the base case (or no contingency) while ensuring that if any of contingencies do occur then the system will be secure. This is illustrated in Fig. 5.1 for a SCOPF with a base-case c_0 and three contingencies.

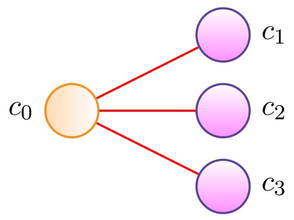

Figure 5.1: Contingency constrained optimal power flow example with three contingencies. c_0 represents the base case (or no contingency case). c_1, c_2, c_3 are the three contingency cases. Each of the contingency states is coupled with the base-case through ramping constraints (denoted by red lines)

In general form, the equations for contingency-constrained optimal power flow are given by (5.1) – (5.5). This is a two-stage stochastic optimization problem where the first stage is the base case c_0 and each of the contingency states $c_i, i \in [1, N_c - 1]$ are second-stage subproblems. SCOPFLOW aims to minimize the objective $\sum_{c=0}^{N_c-1} f(x_c)$, while adhering to the equality $(g(x_c))$, inequality $h(x_c)$, and the lower/upper bound (x^-,x^+) constraints. Equation (5.5) represents the coupling between the base-case and each of the contingency states c_i. Equation (5.5) is the most typical form of coupling that limits the deviation of the contingency variables x_c from the base x_0 to within $\delta_c x$. An example of this constraint could be the allowed real power output deviation for the generators constrained by their ramp limit.

$$\min \sum_{c=0}^{N_c-1} f(x_c) \tag{5.1}$$

s.t.

$$g(x_c) = 0, \quad c \in [0, N_c - 1] \tag{5.2}$$
$$h(x_c) \leq 0, \quad c \in [0, N_c - 1] \tag{5.3}$$
$$x^- \leq x_c \leq x^+, \quad c \in [0, N_c - 1] \tag{5.4}$$
$$-\delta_c x \leq x_c - x_0 \leq \delta_c x, \quad c \in [1, N_c] \tag{5.5}$$

5.1.2 Multiperiod

In the multi-period version,each contingency comprises of multiple time-periods. The multiple periods have variables and constraints as described in chapter 4. An example of multi-contingency multi-period optimal power flow is illustrated in Fig. 5.2 with two contingencies c_0 and c_1. Here, c_0 is the case with no contingencies, i.e., the base-case. Each contingency is multi-period with four time-periods. Each time-step is coupled with its adjacent one through ramping constraints. We assume that the contingency is incident at the first time-step, i.e. at t_0. This results in the coupling between the contingency cases $c_i, i \in [1, N_c - 1]$ and the base-case c_0 only at time-step t_0 as shown in Fig. 5.2.

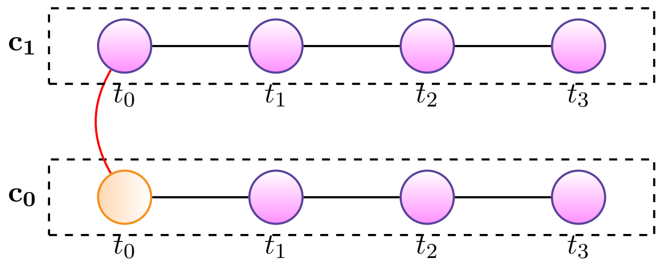

Figure 5.2: Multi-period contingency constrained optimal power flow example with two contingencies c_0 and c_1, each with four time-periods t_0, t_1, t_2, t_3. State c_0, t_0 represent the base case (no contingency) case. We assume that any contingency is incident at the first time-step, i.e., at t_0. The contingency states c_1, t_0 is coupled with the no-contingency state c_0, t_0 at time t_0. The red line denotes the coupling between the contingency.

The overall objective of this contingency-constrained multi-period optimal power flow is to find a secure dispatch for base-case c_0 while adhering to contingency and temporal constraints. Its general formulation is given in Eqs. (5.6) – (5.12).

$$\min \sum_{c=0}^{N_c-1} \sum_{t=0}^{N_t-1} f(x_{c,t}) \tag{5.6}$$

s.t.

$$g(x_{c,t}) = 0, \quad c \in [0, N_c - 1], t \in [0, N_t - 1] \tag{5.7}$$

$$h(x_{c,t}) \leq 0, \quad c \in [0, N_c - 1], t \in [0, N_t - 1] \tag{5.8}$$

$$x^- \leq x_{c,t} \leq x^+, \quad c \in [0, N_c - 1], t \in [0, N_t - 1] \tag{5.9}$$

$$-\delta_t x \leq x_{c,t} - x_{c,t-\Delta t} \leq \delta_t x, \quad c \in [0, N_c - 1], t \in [1, N_t - 1] \tag{5.10}$$

$$-\delta_c x \leq x_{c,0} - x_{0,0} \leq \delta_c x, \quad c \in [1, N_c - 1] \tag{5.11}$$

$$\tag{5.12}$$

In this formulation, the objective is to reduce the cost for the base-case time horizon, where $f(x_{0,t})$ is the objective cost for contingency c_0 at time t. Equation (5.12) represents the coupling between the base case c_0 and each contingency c_i at time-step t_0. We use a simple box constraint $\delta_c x$ to restrict the deviation of decision variables $x_{c,0}$ from the base-case $x_{0,0}$. The bound $\delta_c x$ could represent here, for example, the allowable reserve for each generator.

5.2 Solvers

SCOPFLOW can be solved with Ipopt [7]. If one wants to solve each contingency independently, i.e., without any coupling constraints then use *EMPAR* solver. *EMPAR* distributes the contingencies to different processes when executed in parallel.

5.3 Input and Output

To execute SCOPFLOW, the following files are required:

- **Network file:** The network file describing the network details. Only MATPOWER format files are currently supported.
- **Contingency file:** The file describing the contingencies. Contingencies can be single or multiple outages. The contingency file needs to be described in PTI format.

If the multi-period option is chosen, then additional files describing the load and wind generation can be (optionally) set.

- **Load data:** One file for load real power and one fo reactive power. The files need to be in CSV format. An example of the format for the 9-bus case is here.
- **Wind generation:** The wind generation time-series described in CSV format. See an example of the format here.

The SCOPFLOW output is saved to a directory named *scopflowout*. This directory contains N_c files to save the solution for each contingency in MATPOWER datafile format. Each file has the name *cont˙xx* where *xx* is the contingency number.

If multi-period option is chosen then N_c subdirectories are created (one for each contingency), and each subdirectory contains N_t output files, one for each time-period. The subdirectories have the naming convention *cont˙xx* and the output file are named as *t˙yy* where *yy* is the time-step number.

5.4 Usage

****—To be completed—****

```
./scopflow -netfile <netfilename>  -ctgcfile <ctgcfilename> <
    ↪ scopflowoptions>
```

5.5 Options

See table 5.1

Table 5.1: SCOPFLOW options

Option	**Meaning**	**Values (Default value)**
-netfile	Network file name	string (case9mod˙gen3˙wind.m)
-ctgcfile	Contingency file name	string (case9.cont)
-save_output	Save output to directory	0 or 1 (0)
-scopflow_Nc	Number of contingencies	int (0)
-scopflow_mode	Operation mode: Preventive or corrective	0 or 1 (0)
-scopflow_enable_multiperiod	Multi-period SCOPFLOW	TRUE or FALSE (FALSE)
-scopflow_pload_profile	Real power load profile	string (load˙P.csv)
-scopflow_qload_profile	Reactive power load profile	string (load˙Q.csv)
-scopflow_windgenprofile	Wind generation profile	string (scenarios.csv)
-scopflow_dT	Length of time-step (minutes)	double (5.0)
-scopflow_duration	Total duration (hours)	double (0.5)

In addition, all OPFLOW options given in Table 3.4 can be used to tune the individual time-period.

Depending on the *mode*, SCOPFLOW can either be *preventive* (mode = 0) or *corrective* (mode = 1). In the preventive mode, the PV and PQ generator real power is fixed to its corresponding base-case values. The generators at the referencce bus pick up any make-up power required for the contingency. The corrective mode allows deviation of the PV and PQ generator real power from the base-case dispatch constrained by its 30-min. ramp rate capability.

5.6 Examples

****—To be completed—****

Chapter 6

Stochastic optimal power flow (SOPFLOW)

SOPFLOW solves a stochastic security-constrained multi-period optimal power flow problem. The problem is set up as a two-stage optimization problem where the first-stage (base-case) represents the normal operation of the grid (or the most likely forecast) and the second-stage comprises of N_s scenarios of forecast deviation. Each scenario can have multiple contingencies and each contingency can be multi-period. Thus, depending on the options selected, each stochastic scenario can be

- Single-period, no contingency
- Single-period contingencies
- Multi-period contingencies

6.1 Formulation

An illustration of SOPFLOW in Fig. 6.1 for a case with two scenarios s_0 and s_1. Each scenario has two contingencies c_0, c_1, and each contingency has four time-periods.

The formulation for the stochastic security-constrained multi-period optimal power flow is given in (6.1) – (6.7). In this formulation, the objective is to reduce the expected cost, where $f(x_{s,0,0})$ is the cost for scenario s with no contingencies (hence 0 for the contingency index). ρ_s is the probability of scenario s.

$$\min \sum_{s=1}^{N_s-1} \pi_s \sum_{c=0}^{N_c-1} \sum_{t=0}^{N_t-1} f(x_{s,c,t}) \tag{6.1}$$

s.t.

$$g(x_{s,c,t}) = 0, \qquad s \in [1, N_s-1], c \in [0, N_c-1], t \in [0, N_t-1] \tag{6.2}$$

$$h(x_{s,c,t}) \leq 0, \qquad s \in [1, N_s-1], c \in [0, N_c-1], t \in [0, N_t-1] \tag{6.3}$$

$$x^- \leq x_{s,c,t} \leq x^+, \qquad s \in [1, N_s-1], c \in [0, N_c-1], t \in [0, N_t-1] \tag{6.4}$$

$$-\delta_t x \leq x_{s,c,t} - x_{s,c,t-\Delta t} \leq \delta_t x, \qquad s \in [1, N_s-1], c \in [0, N_c-1], t \in [1, N_t-1] \tag{6.5}$$

$$-\delta_c x \leq x_{s,c,0} - x_{s,0,0} \leq \delta_c x, \qquad s \in [1, N_s-1], c \in [1, N_c-1] \tag{6.6}$$

$$-\delta_s x \leq x_{s,0,0} - x_{0,0,0} \leq \delta_s x, \qquad s \in [1, N_s-1] \tag{6.7}$$

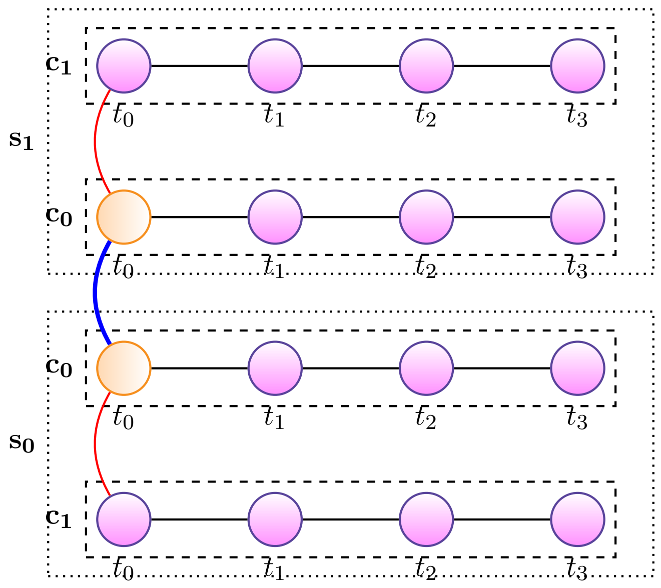

Figure 6.1: Stochastic multi-period contingency constrained example with two scenarios s_0 and s_1. Each scenario has two contingencies c_0,c_1 and each contingency consists of four time-periods t_0, t_1, t_2, t_3. State s_0, c_0, t_0 represent the base case (no contingency) case for the two scenarios. We assume that any contingency is incident at the first time-step, i.e., at t_0. Thus, the contingency states c_1, t_0 is coupled with the no-contingency state c_0, t_0 at time t_0 for both the scenarios. The red line denotes the coupling between the contingency and the no-contingency states.The blue line denotes the coupling between the scenarios

SOPFLOW uses all the modeling details used for modeling an optimal power flow problem, i.e., each of the circles shown in Fig. 6.1 has the modeling details of an optimal power flow problem (OPFLOW). Incorporating the probabilities π_s for each scenario is not implemented yet which leads to each scenario having an equal probability.

Currently, SOPFLOW uses wind power generation as the stochastic variables and each scenario is a realization of the power output from wind generators. A zero fuel cost is used for wind power generation to ensure wind generation would be the dispatched to the given target level (upper limit).

For contingencies, SOPFLOW supports generation and/or transmission outages. A contingency can have multiple outages, but, it should not cause any islanding. The coupling between the no-contingency and the contingency case for each scenario is also the difference in real power output ($p^{\mathrm{g}}_{jsct} - p^{\mathrm{g}}_{js0t},\ j \in J^{\mathrm{gen}}$) that must be within the 30 minute generator ramp rate.

For multi time-period, we use ramping constraints on the generator real power output between successive time steps.

SOPFLOW can be run in two modes: preventive and corrective. In the preventive mode, generator real power output is fixed to the base-case values for generators at PV bus(es). In this mode, the generators at the reference bus provide/absorb any deficit/surplus power. The corrective mode allows deviation of the PV and PQ generator real power from the base-case dispatch constrained by its 30-min. ramp rate capability. Note that the preventive/corrective mode is only applied at the first step t_0. In the successive time-steps, the generator dispatch is dictated by the previous step dispatch and the ramp limits.

6.2 Solvers

SOPFLOW can be solved with Ipopt [7]. If one wants to solve each scenario independently, i.e., without any coupling constraints then use *EMPAR* solver. *EMPAR* distributes the contingencies to different processes when executed in parallel.

6.3 Input and Output

The following files are needed for executing a SOPFLOW.

- **Network file:** The network file describing the network details. Only MATPOWER format files are currently supported.
- **Scenario file:** SOPFLOW only supports reading wind generation scenarios in a CSV format. An example of this format for the 9-bus case is here.
- **Contingency file:** Contingencies can be specified via PTI format file as described in chapter 5.The option `-sopflow_enable_multicontingency` should be set for multi-contingency problems.
- **Load data:** One file for load real power and one fo reactive power. The files need to be in CSV format. An example of the format for the 9-bus case is here.

The SOPFLOW output is saved to a directory named *sopflowout*. This directory contains N_s subdirectories to save the solution for each scenario. Each of these subdirectories contain N_c subdirectories, one for each contingency. Each contingency subdirectory has N_t MATPOWER format files to store the output for each time-period for the given contingency and scenario. The subdirectories have the directory name format *scen_x* where x is the scenario number, *cont_y* where y is the contingency number, and the output files have the file name format *t_z* where z is the time-step number.

6.4 Usage

```
./sopflow -netfile <netfilename>  -scenfile <scenfilename> <
    ↪ sopflowoptions>
```

****—To be completed—****

6.5 Options

See table 6.1

Table 6.1: SOPFLOW options

Option	**Meaning**	**Values (Default value)**
-netfile	Network file name	string (case9mod˙gen3˙wind.m)
-scenfile	Scenario file naame	string (scenarios.csv)
-sopflow_mode	Operation mode: Preventive or corrective	0 or 1 (0)
-sopflow_enable_multicontingency	Multi-contingency SOPFLOW	TRUE or FALSE (FALSE)

With multi-contingency SOPFLOW, all SCOPFLOW options given in Table 5.1 can be used to tune the contingencies.

Depending on the *mode*, SOPFLOW can either be *preventive* (mode = 0) or *corrective* (mode = 1). In the preventive mode, the PV and PQ generator real power is fixed to its corresponding base-case values. The generators at the reference bus pick up any make-up power required for the contingency. The corrective mode allows deviation of the PV and PQ generator real power from the base-case dispatch constrained by its 30-min. ramp rate capability.

6.6 Examples

****—To be completed—****

Appendices

Appendix A

Symbol reference

Units of measurement are given in Table (A.1), indices and index sets in Table (A.2), subsets in Table (A.3), special set elements in Table (A.4), and real-valued parameters in Table (A.5).

Table A.1: Units of measurement

Symbol	Description
1	dimensionless. Dimensionless real number quantities are indicated by a unit of 1.
USD	US dollar. Cost, penalty, and objective values are expressed in USD.
h	hour. Time is expressed in h.
pu	per unit. Voltage magnitude is expressed in a per unit system under given base values, and the unit is denoted by pu
rad	radian. Voltage angles are expressed in rad.
MW	megawatt. Real power is expressed in MW.
MVar	megavolt-ampere-reactive. Reactive power is expressed in MVar.
MVA	megavolt-ampere. Apparent power is expressed in MVA.
MW at 1 pu	megawatt at unit voltage. Conductance is expressed in MW at 1 pu, meaning the conductance is such as to yield a real power flow equal to the indicated amount when the voltage is equal to 1 pu
MVar at 1 pu	megavolt-ampere-reactive at unit voltage. Susceptance is expressed in MVar at 1 pu, meaning the susceptance is such as to yield a reactive power flow equal to the indicated amount when the voltage is equal to 1 pu

Table A.2: Index sets

Symbol	Description
$a \in A$	areas
$i \in I$	buses

Table A.2: Continued

Symbol	Description
$j \in J$	bus-connected grid components, i.e. loads, shunts, generators, stochastic resources, branches
$k \in K$	security contingencies, i.e. NERC $(n-1)$- or $(n-k)$-style contingencies, different from the severe event we are modeling
$s \in S$	stochastic scenarios
$t \in T$	time periods

Table A.3: Subsets

Symbol	Description
$J^{\text{gen}} \subset J$	generators
$J^{\text{ld}} \subset J$	loads
$J^{\text{br}} \subset J$	branches, i.e. lines, transformers
$J^{\text{sh}} \subset J$	shunts
$J_i^{\text{o}} \subset J$	branches with origin bus at bus i
$J_i^{\text{d}} \subset J$	branches with destination bus at bus i

Table A.4: Special set elements

Symbol	Description
$a_i \in A$	area of bus i
$i_j^{\text{d}} \in I$	destination bus of branch $j \in J$
$i_j^{\text{o}} \in I$	origin bus of branch $j \in J^{\text{br}}$

Table A.5: Real-valued parameters

Symbol	Description
$b_j^{\max}$	maximum susceptance for shunt $j \in J^{\text{sh}}$ (MVar at 1 pu)

Table A.5: Continued

Symbol	Description
$b_j^{\min}$	minimum susceptance for shunt $j \in J^{\text{sh}}$ (MVar at 1 pu)
b_j^{ch}	charging susceptance for branch $j \in J^{\text{br}}$ (MVar at 1 pu)
b_j^{s}	series susceptance for branch $j \in J^{\text{br}}$ (MVar at 1 pu)
p_j^{gset}	real power set point of generator $j \in J^{\text{gen}}$ (MW)
p_j^{gmax}	maximum real power output for generator $j \in J^{\text{gen}}$ (MW)
p_j^{gmin}	minimum real power output for generator $j \in J^{\text{gen}}$ (MW)
p_j^{r}	maximum ramp rate for generator $j \in J^{\text{gen}}$ (MW/h)
q_j^{gmax}	maximum reactive power output for generator $j \in J^{\text{gen}}$ (MVar)
q_j^{gmin}	minimum reactive power output for generator $j \in J^{\text{gen}}$ (MVar)
$v_i^{\max}$	maximum voltage magnitude for bus $i \in I$ (pu)
$v_i^{\min}$	minimum voltage magnitude for bus $i \in I$ (pu)
π_s	probability of scenario s (1)

Bibliography

[1] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dmitry Karpeyev, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.12, Argonne National Laboratory, 2019.

[2] D. Beckingsale, M. Mcfadden, J. Dahm, R. Pankajakshan, and R. Hornung. Umpire: Application-focused management and coordination of complex hierarchical memory. *IBM Journal of Research and Development*, 2019.

[3] David A Beckingsale, Jason Burmark, Rich Hornung, Holger Jones, William Killian, Adam J Kunen, Olga Pearce, Peter Robinson, Brian S Ryujin, and Thomas RW Scogland. Raja: Portable performance for large-scale scientific applications. In *2019 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, pages 71–81. IEEE, 2019.

[4] David A Beckingsale, Marty J Mcfadden, Johann PS Dahm, Ramesh Pankajakshan, and Richard D Hornung. Umpire: Application-focused management and coordination of complex hierarchical memory. *IBM Journal of Research and Development*, 64(3/4):00–1, 2019.

[5] C. G. Petra, I.Aravena Solis, N. Gawande, V. Amatya, A. Li, J. Li, S. Abhyankar, S. Peles, M. Schanen, K. Kim, A. Maldonado, M. Anitescu. ExaSGD - Optimization grid dynamics at Exascale, 2020.

[6] Cosmin Petra. HiOP User Guide version 0.3, 2017.

[7] Andreas Waechter and L. Biegler. On the implementation of an interior-point filter line-search algorithm for large scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.