High-performance interior point methods

Application to power grid problems

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Juraj Kardoš

under the supervision of
Olaf Schenk

March 2020

Dissertation Committee

Illia Horenko	Università della Svizzera italiana, Switzerland
Igor Pivkin	Università della Svizzera italiana, Switzerland
Petr Korba	Zurich University of Applied Sciences, Switzerland
Tomáš Kozubek	Technical University of Ostrava, Czech Republic
Andreas Wächter	Northwestern University, USA

Dissertation accepted on 26 March 2020

Research Advisor	PhD Program Director
Olaf Schenk	**Walter Binder and Silvia Santini**

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Juraj Kardoš
Lugano, 26 March 2020

Enlightenment is man's emergence from his self-incurred immaturity. Immaturity is the inability to use one's own understanding without the guidance of another. This immaturity is self-incurred if its cause is not lack of understanding, but lack of resolution and courage to use it without the guidance of another. The motto of enlightenment is therefore: Sapere aude! Have courage to use your own understanding!

Immanuel Kant

Abstract

A software library for the solution of large-scale structured nonconvex optimization problems is presented in this work, with the purpose of accelerating the solution on single-core, multicore, or massively parallel high-performance distributed memory computing infrastructures. A large class of industrial and engineering problems possesses a particular structure, motivating the development of structure exploiting interior point methods. Interior point methods are among the most popular techniques for large-scale nonlinear optimization and their efficiency has attracted a lot of attention in recent years. Since the overall performance of interior point methods relies heavily on scalable sparse linear algebra solvers, this work thoroughly analyzes cutting-edge research based on the sparse linear algebra and structure exploiting methods presented over recent years, and further advances the performance by inspecting the structure of the underlying linear systems, resulting in an additional computational time and memory savings.

The primal-dual interior point framework is applied for the solution of optimal power flow problems, a class of optimization problems attracting increasing attention in power system research, operations, and planning. Optimal power flow involves large-scale nonconvex optimization problems with a number of variables and constraints ranging up to hundreds of millions depending on the grid resolution and specific problem formulation. The robustness and reliability of interior point methods is investigated for different optimal power flow formulations for a wide range of realistic power grid networks. Furthermore, the object-oriented parallel and distributed scalable solver is implemented and applied to large-scale problems solved on a daily basis for the secure transmission and distribution of electricity in modern power grids. Similarly, an efficient algorithm is investigated for optimal power flow spanning long time horizons. Using computational studies from security constrained and multiperiod optimal power flow problems, the robustness and scalability of the structure exploiting approach is demonstrated.

Acknowledgements

It is my pleasure to thank all the people I have met on my academic journey, whether in a professional or friendly manner. The professional encounters enriched my knowledge of the filed and provided me valuable insights into many solution techniques, while the friends supported me in strenuous times, helped me to find courage for difficult decisions, and kept me going toward my final destination. Many thanks to all of you! It is only because of you I endured and I will do my best to share my knowledge and try to help others as much as you did!

This project was carried out within the frame of the Swiss Centre for Competence in Energy Research on the Future Swiss Electrical Infrastructure (SCCER-FURIES) with the financial support of the Swiss Innovation Agency (Innosuisse-SCCER program).

Contents

Nomenclature

$N_{\mathrm{B}}, N_{\mathrm{L}}$	Number of buses and lines (branches)
$N_{\mathrm{G}}, N_{\mathrm{S}}$	Number of generators and energy storage devices
$\mathcal{B}_{\mathrm{PV}}$	Set of PV buses
$\mathcal{B}_{\mathrm{slack}}$	Slack (reference) bus
$C_{\mathrm{B}}, C_{\mathrm{L}}, C_{\mathrm{G}}$	Bus, branch and generator connectivity matrices, respectively
C_f, C_t	Branch connectivity matrix for its both ends
C_{S}	Energy storage connectivity matrix
$\mathbf{v}, \boldsymbol{\theta}$	Bus voltage magnitude and angle
$\boldsymbol{u}, \boldsymbol{w}$	Real and imaginary components of the complex bus voltage
$\underline{\mathbf{v}}$	Complex bus voltage vector $\underline{\mathbf{v}} = \mathbf{v}e^{j\boldsymbol{\theta}}$
$\underline{\mathbf{v}}_f, \underline{\mathbf{v}}_t$	Complex branch voltage vector for 'from' and 'to' ends
$\mathbf{p}^{\mathrm{D}}, \mathbf{q}^{\mathrm{D}}$	Active and reactive power demands
$\underline{\mathbf{s}}^{D}$	Complex vector of power demands
$\mathbf{p}^{\mathrm{G}}, \mathbf{q}^{\mathrm{G}}$	Vectors of active and reactive power generation
$\underline{\mathbf{s}}^{G}$	Complex vector of generator power injections, $\underline{\mathbf{s}}^{G} = \mathbf{p}^{\mathrm{G}} + j\mathbf{q}^{\mathrm{G}}$
$\mathbf{p}^{\mathrm{G,min}}, \mathbf{p}^{\mathrm{G,max}}$	Active generation power box bounds
$\mathbf{q}^{\mathrm{G,min}}, \mathbf{q}^{\mathrm{G,max}}$	Reactive power generation box bounds
$\mathbf{p}^{\mathrm{B}}, \mathbf{q}^{\mathrm{B}}$	Active and reactive power flow bus injections
$\underline{\mathbf{s}}^{\mathrm{B}}$	Complex power flow bus injections
$\underline{\mathbf{s}}^{f}, \underline{\mathbf{s}}^{t}$	Complex vectors of branch power injections at 'from' and 'to' ends
$\mathbf{p}^{\mathrm{Sd}}, \mathbf{p}^{\mathrm{Sc}}$	Discharging and charging active power of storage devices
$\mathbf{p}^{\mathrm{S}}$	Composite vector of the storage power injections $\mathbf{p}^{\mathrm{S}} = \left[\mathbf{p}^{\mathrm{Sd}}, \mathbf{p}^{\mathrm{Sc}}\right]$
$\mathbf{p}^{\mathrm{S,min}}, \mathbf{p}^{\mathrm{S,max}}$	Minimum and maximum active storage power output
$\underline{\mathbf{s}}^{\mathrm{S}}$	Complex power output of storage devices
$\boldsymbol{s}_{\mathrm{L}}^{\mathrm{max}}$	Branch power flow limits
$\underline{\mathbf{i}}^{\mathrm{B}}$	Complex current injections vector
$\underline{\mathbf{i}}_f, \underline{\mathbf{i}}_t$	Complex current injections, from and to branch ends
$\boldsymbol{G}, \boldsymbol{B}$	Bus conductance and susceptance matrices
$\underline{\mathbf{Y}}^{\mathrm{B}}$	Complex bus admittance matrix $\underline{\mathbf{Y}}^{\mathrm{B}} = \boldsymbol{G} + j\boldsymbol{B}$
$\underline{\mathbf{Y}}^{\mathrm{L}}$	Complex branch admittance matrix

$\boldsymbol{\epsilon}_{\mathrm{S}}^{\max}$	Capacity of the storage devices
$\boldsymbol{\epsilon}_0$	Initial state of charge of the storage devices
$\eta_{\mathrm{d}}, \eta_{\mathrm{c}}$	Discharging and charging efficiency
$\boldsymbol{u}_t^{Sd,i}, \boldsymbol{d}_t^{Sd,i}$	Up and down discharging flexibility
$\boldsymbol{u}_t^{Sc,i}, \boldsymbol{d}_t^{Sc,i}$	Up and down charging flexibility
$\boldsymbol{u}_t^f, \boldsymbol{d}_t^f$	Up and down flexibility requirements
$\boldsymbol{r}_s$	Resistance of branches
$\boldsymbol{x}_s$	Reactance of branches
$\underline{\mathbf{z}}_s$	Impedance of branches, $\underline{\mathbf{z}}_s = \boldsymbol{r}_s + j\boldsymbol{x}_s$
$\boldsymbol{b}_s$	Susceptance of branches
$\mathbf{g}_s$	Conductance of branches
$\underline{\boldsymbol{y}}_s$	Admittance of branches, $\underline{\boldsymbol{y}}_s = \mathbf{g}_s + j\boldsymbol{b}_s$
$\boldsymbol{b}_c$	Branch charging susceptances
τ	Transformer per unit tap ratios
$\boldsymbol{\theta}_{\text{shift}}$	Transformer phase shift angles
$\mathcal{T}$	Ideal phase shifting transformers $\mathcal{T} = \tau e^{j\boldsymbol{\theta}_{\text{shift}}}$
$\boldsymbol{b}_{sh}, \mathbf{g}_{sh}$	Shunt susceptance and conductance
$\underline{\boldsymbol{y}}_{sh}$	Shunt admittance from branch to neutral $\boldsymbol{y}_{sh} = \mathbf{g}_{sh} + j\boldsymbol{b}_{sh}$
$\mathcal{C}$	Set of contingency scenarios
N_c	Size of the contingency set $\lvert\mathcal{C}\rvert$
N	Number of time periods
δt	Time period length
$\boldsymbol{x}$	Optimization vector
$\boldsymbol{x}_{\min}, \boldsymbol{x}_{\max}$	Optimization vector bounds
f	Objective function
$\boldsymbol{c}_\varepsilon, \boldsymbol{c}_I$	Equality and inequality constraints
$\boldsymbol{s}$	Slack variables
$\boldsymbol{\lambda}_\varepsilon, \boldsymbol{\lambda}_I$	Lagrange multipliers for the constraints
$\boldsymbol{\lambda}_x, \boldsymbol{\lambda}_s$	Lagrange multipliers for the variables
$\mathcal{L}$	Lagrangian
$\boldsymbol{z}, \boldsymbol{y}$	Dual variables for $\boldsymbol{x}$ and $\boldsymbol{s}$
μ	Barrier parameter
$\boldsymbol{J}_\varepsilon, \boldsymbol{J}_I$	Jacobian of the equality and inequality constraints
$\boldsymbol{H}$	Hessian of the Lagrangian $\nabla^2_{xx}\mathcal{L}$
$\odot$	Operator denoting element-wise product of two vectors
$[\cdot]$	Operator representing concatenation of column vectors $[x_1, x_2] = (x_1^\intercal, x_2^\intercal)^\intercal$

List of abbreviations

AC	Alternating current
DC	Direct current
PF	Power flow
OPF	Optimal power flow
SCOPF	Security constrained optimal power flow
MPOPF	Multiperiod optimal power flow
DSO	Distribution system operator
TSO	Transmission system operator
ES	Energy storage
SOC	State of charge
RE	Renewable energy
DER	Distributed energy resources
MO	Market operator
LP	Linear programming
QP	Quadratic programming
NLP	Nonlinear programming
SQP	Sequential quadratic programming
MIP	Mixed integer programming
IP	Interior point
BSP	Barrier subproblem
KKT	Karush–Kuhn–Tucker
SC	Schur complement
NR	Newton-Raphson
HPC	High-performance computing
GPU	Graphics processing unit
CPU	Central processing unit
MPI	Message passing interface

Chapter 1

Introduction

Due to global energy transition based on both economic growth and decarbonization, in conjunction with the liberalization of the energy market, the importance of the computational tools in the power grid analysis has emerged more pronounced than ever before. At the same time the proportion of renewable energies (REs) is called upon to increase in all major electricity markets. Currently, hydroelectricity is by far the largest contributor among the REs while the proportion of RE that comes from variable sources such as wind and solar is still relatively limited. The reason is that variability of wind and solar REs creates distinct challenges for integration into the larger power system. Namely, RE poses requirements on generation, transmission, and operation technology in terms of flexibility while maintaining system security and reliability. Power grid operators, market players, or balancing authorities daily decision-making could benefit significantly from computational software tools able to accurately model power grid operations and solve optimization problems within short time frames.

We briefly discuss the most common optimal control problems supporting transmission system operators daily decisions and further introduce a class of optimization methods that has been successful for the solution of power grid problems.

1.1 Optimal power flow problems

A set of optimization problems in electric power systems known in the literature as optimal power flow (OPF), is one of the most studied subfields of constrained nonlinear optimization. The OPF problem was introduced by Carpentier Carpentier [1962], who extended the economic dispatch (ED) in power systems by inclusion of the electric power flow equations. The presence of the power flow

equations in the set of equality constraints remains the defining characteristic of the problem. OPF is a large-scale, nonlinear, nonconvex optimization problem aiming to minimize the generation cost of electricity (or similar metric, e.g. transmission losses, interface reactive power cost, social welfare, etc.), while at the same time satisfying the physical constraints of the power grid among other engineering and operational constraints further classifying OPF problems.

Power-grid optimization problems also include unit commitment (UC), which is day ahead planning of generators and ED, which is used to balance supply and demand. These problems need to be solved in restrictive time limits, less than one hour for UC and even smaller time window of several minutes is available for ED. Sub-hourly generation scheduling allows system operators to deal with intermittent RE sources by reducing the period of uncertainty around wind generation schedules and allows wind plant owners to adjust schedules more frequently and thus helps to support system flexibility requirements. Consequently, a high-end, distributed memory, supercomputing solution is required in order to meet these time limits and find the solution of large stochastic optimization problems.

For the operation of power systems with high penetration of large-capacity RE generation, RE power forecasting is critical for grid operators to carry out operational planning studies and ensure that adequate resources are available for managing the variability of RE output. The forecast, however, provides only an estimate of the expected power generating capacities. One possible approach on how to deal with error in forecasting is to generate multiple forecasts and corresponding future scenarios. This is the general idea of the sample average approximation (SAA) technique that is used in stochastic optimization to evaluate the stochastic objective function. The resulting optimization problem is then solved by deterministic methods, providing operators the optimal grid setup with respect to certain criteria, such as operational costs or environmental impact.

The security constrained OPF (SCOPF) is an extension of the OPF problem that additionally guarantees that the network operation will remain secure at the even of a set of postulated contingencies. The SCOPF has become an essential tool for many transmission system operators for the planning, operational planning, and real time operation of the power system. Increasing the number of considered contingencies requires the introduction of additional variables and constraints, which in turn results in a significant growth of the problem size, rendering the solution computationally intractable for standard general purpose optimization tools. The structure of the SCOPF problems however, is appropriate for parallel structure-exploiting IP methods, where each contingency corresponds to a separate partition on the linear level.

1.2 Interior point methods

Interior point (IP) methods have become a successful and ubiquitous tool for the solution of constrained optimization problems. The IP “revolution”, a term coined by M. Wright Wright [2005], can be traced back to 1984 when Karmarkar Karmarkar [1984] announced a polynomial time linear program (LP) that was considerably faster than the most popular simplex method to date. Since then, IP methods have continued to transform both the theory and practice of constrained optimization.

The basic principle of the IP family of methods is to move through the interior of the feasible region towards the (local) optimal solution by approximately solving a sequence of "simpler" subproblems. IP methods are easily applicable to problems involving large numbers of equality and inequality constraints. They are therefore the method of choice for large-scale optimal control problems, and they allow customized direct or iterative solution methods for the underlying linear systems solved at each iteration. Since different linear system solvers can be plugged in with ease, large-scale structured problems can benefit by factorization methods that can exploit their structure as well as parallel computing infrastructures. The main steps in every IP method consist of (i) transforming the inequality constraints or variable bounds into logarithmic barrier terms, (ii) defining a Lagrangian function, (iii) stating the first-order optimality conditions known as Karush–Kuhn–Tucker (KKT) conditions, (iv) applying Newton’s method to solve the KKT system, and finally, (v) updating the iterates and the barrier parameter. The steps are repeated until the solution is found and the optimality conditions are satisfied.

The step involving solution of the so-called KKT linear system represents the major computational bottleneck in state-of-the-art IP packages. In the case of large-scale problems, the cost of factorizations may be prohibitive in terms of memory and time, thus limiting the effective use of optimization codes. Thus, highly optimized linear algebra algorithms and software components is of paramount importance for an efficient and scalable IP solver.

Structure exploiting IP methods

Structure exploiting IP methods were first attempted in OOQP Gertz and Wright [2003], where a software package for solving convex QPs based on primal-dual IP algorithm was developed to allow users to solve structured problems by supplying linear algebra customized kernels to the particular block structure. This attempt provided became the cornerstone for more advanced implementations

that followed in OOPS Gondzio and Grothey [2009] and PIPS Petra [2014]. The crucial algorithmic improvement allowing for the efficient linear decomposition was introduced in Petra, Schenk, Lubin and Gärtner [2014]. It adopted the evaluation of local Schur complement contributions by performing an incomplete factorization of the augmented matrices. The approach was first used in PIPS-IPM Petra, Schenk and Anitescu [2014] for solution of stochastic LPs and convex QPs. PIPS-NLP, a software library for the solution of large-scale structured nonconvex optimization problems on high-performance computers, was introduced in Chiang et al. [2014]. It also exploited the structured linear algebra to achieve high computational efficiency. The linear decomposition of the nonconvex problems was applied also recently in Schanen et al. [2018] aiming at a petascale computational strategy on a system equipped with Intel manycore processors. The decomposition algorithm can be very efficient also on single core computations, as demonstrated in Kourounis et al. [2018]. Further advancement of the linear decomposition was explored in Kardoš et al. [2020], analyzing additional elimination of the problem size on the linear level, thus reducing the memory footprint of the decomposition algorithm, which is critical for the truly large-scale problems. The decomposition algorithm is described in detail in part III of this document.

1.3 Contribution

This work focuses on the interplay between the IP optimization techniques and numerical linear algebra kernels involved in direct linear solvers. Specifically, large-scale problems with millions of decision variables and constraints are considered. Linear systems resulting from many engineering and scientific domains are not only sparse but also intrinsically structured. Truly large-scale problems are by necessity generated by some repeated process. An example of such repeated processes are scenarios of real life operations — considering possible outages of the operational equipment and production resources, scenarios introduced by the discretization of the underlying probability space or evolution of the process in the discretized time horizon D'Apuzzo et al. [2010]. The individual instances of the process are usually intercoupled, representing the relations between the specific scenarios, or intertemporal coupling constraints.

It is a fair assumption that details related to the problem structure can be an additional input to the solver for accelerating the solution process. Appropriate factorization algorithms from sparse linear algebra are then adopted that exploit the structure. The KKT systems of such problems are commonly reordered to

an arrowhead block structure. Since linear algebra operations that exploit block structure lend themselves to parallelization, the block structure can be efficiently solved by the Schur complement decomposition techniques, where the individual blocks are distributed to available processes. Scalable solvers based on the Schur complement decomposition are applied for the solution of OPF and SCOPF problems. The linear solvers are designed for distributed memory architectures due to the large-scale character of such problems. The proposed method also addresses shared memory parallelism, taking into account modern computing architectures, and utilizes advanced linear algebra techniques to further reduce the problem size, improving memory requirements and time to solution. The computational experiments involve a variety of power grids of increasing complexity, ranging from one up to several thousand buses resulting to KKT systems with up to millions of variables constraints.

This thesis is divided into three parts. Part I, *Power grid optimization problems*, introduces the power grid infrastructure and formulation of the associated optimization problems, including OPF, SCOPF, and multiperiod OPF (MPOPF). Each problem is briefly discussed from the perspective of possible solution approaches proposed in the literature. In Part II, *Interior point methods*, the state-of-the-art IP algorithm is introduced and various building block and heuristics are discussed. Possible solution methods of the saddle point systems, associated with the IP algorithm, are also briefly summarized. Finally, some of the state-of-the-art IP nonlinear programming solvers and power grid simulation software packages are discussed in part III – *High-performance IP algorithms and software for power grid problems*. The discussion is centered around the structure exploiting algorithms designed for the power grid optimization problems. A parallel HPC algorithmic framework is introduced. This part is concluded with a set of extensive performance tests, comparing various commercial and academic IP packages and OPF problem formulations. The new linear algebra treatment is analyzed in an HPC environment and the performance improvement of the new library, both with respect to memory consumption and running time, is assessed. The document is concluded by a short summary and offers additional research directions that could further advance the concepts introduced in this document.

The main contribution of this work to the existing literature are listed here (not using any particular order).

- The design and implementation of a parallel, scalable, structure-exploiting KKT linear system solver suitable for the solution of large-scale structured optimization problems, based on the distributed Schur complement decomposition, taking into account both modern multicore CPUs and GPU accelerated architectures.

- An additional sparsity-maintaining elimination of the variables is employed during the solution of the KKT systems. This additional level of Schur complement further reduces the solution time and memory requirements of the KKT systems in each iteration of the IP method even on single-core execution.
- The study of the parallel performance of the scalable framework is performed through weak-scaling and strong-scaling tests on nodes of a Cray system using large-scale SCOPF power grid problems.
- An interface to the parallel and distributed linear solver was implemented and added to state-of-the-art primal-dual IP library IPOPT.
- An exhaustive comparative study of the IP packages on a set of OPF benchmarks, using multiple OPF formulations and considering additional factors contributing to the overall performance of the optimization software.
- Detailed evaluation of solution algorithms for the OPF, SCOPF, and MPOPF problems is performed on a set of power grid benchmarks with an increasing complexity.
- An extension of the energy storage model is introduced and embedded in the AC MPOPF framework, providing the flexibility to the (distribution) system operator during the power grid operation and planning.
- C++ power grid simulator was implemented, supporting various applications, including power flow, OPF, SCOPF, and MPOPF problems with support of various optimizers such as IPOPT (including also the parallel linear solver). Similar OPF extensions were also implemented in MATPOWER.

Part I

Power grid optimization problems

Chapter 2

Architecture of the power grid

An electrical grid, or power grid, is an interconnected network for delivering electricity from producers to consumers. It consists of generating stations that produce electrical power, high voltage transmission lines that carry power from distant sources to demand centers, and distribution lines that connect individual customers. A scheme of a typical electric power grid is illustrated in Figure 2.1. Extrahigh-voltage electricity (380 kV and 220 kV) reaches the transmission grid from power plants as well as imports from abroad. The voltage must be as high as possible so that as much energy as possible can be transported over great distances with minimal losses. Depending on the target customer (industrial center or a typical family house) the voltage level is stepped down across multiple stages and different grid levels into medium- and low-voltage distribution grids.

Recent developments in modern power grids involve widespread deployment of intermittent renewable generation, embrace installation of a wide variety of energy storage devices, as well as an increasing and widespread usage of electric vehicles. On the other hand, conventional energy sources are continually discontinued, such as coal or nuclear power Swiss Federal Office of Energy [2018]. These developments motivate fundamental changes in methods and tools for the optimal daily operation and planning of modern power grids. Operational decisions taken by power system operators on a daily basis are commonly assisted by repeatedly solving complex optimization problems, aiming to determine optimal operating levels for electric power plants, so that the overall electricity generation cost is minimized, while at the same time it satisfies load demands imposed throughout the transmission grid and meets safe operating limits. However, exploitation of renewable energy sources and their grid integration poses many new challenges for grid operations due to their intermittent nature and high variability. New strategies for the operation and management of the electricity grid have

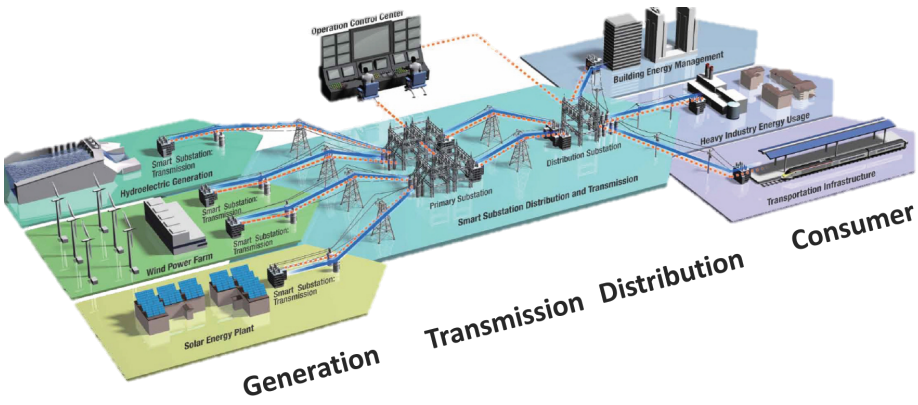


Figure 2.1. Structure of an electric power system.

to be developed, in order to maintain power-supply reliability and resolve the imbalances in the grid caused by large forecast errors. Energy storage is considered to be an effective and flexible approach for addressing future operational challenges associated with renewables. Storing the energy generated during periods of low demand and reusing it during periods of high demand does not only help overcome the problem of supply from renewable energy sources, but it allows electricity grids to operate more efficiently and cost effectively. The modeling of storage devices, however, introduces an intertemporal coupling of the related optimization problems corresponding to each time period. The resulting very large optimization problems become intractable for general optimization methods and call for new computational tools assisting the grid operations.

Substantial increase in electricity demand had neither been foreseen nor adequately supported by necessary upgrades of the electricity generation and transmission systems. As a consequence, power networks nowadays have to operate in stressed conditions close to their physical and engineering limits. These cost-efficient operational conditions are usually uncertain because of the widespread integration of renewable energy sources and the introduction of intraday electricity markets Capitanescu et al. [2011]. Additionally, the renewable energy is also placing greater stress on the power grid equipment and shifting their operational conditions towards their limits. As a result, failures of any network component, such as a transmission line or power generator, can be critical to the overall grid operation. Consequently, strict security measures have to be honored by power system operators, which also heavily rely on available computational tools.

2.1 Mathematical model

Consider a power grid with N_{B} buses, N_{G} generators, and N_{L} transmission lines. The bus voltage vector $\underline{\mathbf{v}} \in \mathbb{C}^{N_{\mathrm{B}}}$ is defined in polar notation as $\underline{\mathbf{v}} = \mathbf{v}e^{j\boldsymbol{\theta}}$, where $\mathbf{v}, \boldsymbol{\theta} \in \mathbb{R}^{N_{\mathrm{B}}}$ specify the magnitude and phase of the complex voltage. The complex voltages $\underline{\mathbf{v}}$ determine the entire power flow (PF) in the grid that can be computed using the Kirchhoff equations and the grid configuration, such as the transmission line parameters, transformer tap ratios, and shunt elements. The current injections $\underline{\mathbf{i}}^{\mathrm{B}} \in \mathbb{C}^{N_{\mathrm{B}}}$ into the buses are defined as $\underline{\mathbf{i}}^{\mathrm{B}} = \underline{\mathbf{Y}}^{\mathrm{B}}\underline{\mathbf{v}}$, where $\underline{\mathbf{Y}}^{\mathrm{B}} \in \mathbb{C}^{N_{\mathrm{B}} \times N_{\mathrm{B}}}$ is the bus admittance matrix. The complex power at each bus of the network $\underline{\mathbf{s}}^{\mathrm{B}} = \underline{\mathbf{v}}\underline{\mathbf{i}}^{\mathrm{B}*}$, $\underline{\mathbf{s}}^{\mathrm{B}} \in \mathbb{C}^{N_{\mathrm{B}}}$ is to be balanced by the net power injections from the generators $\underline{\mathbf{s}}^{G} \in \mathbb{C}^{N_{\mathrm{G}}}$, storage devices $\underline{\mathbf{s}}^{\mathrm{S}} \in \mathbb{C}^{N_{\mathrm{S}}}$ and demand centers' power consumption $\underline{\mathbf{s}}^{D} \in \mathbb{C}^{N_{\mathrm{B}}}$. Thus, the alternating current (AC) nodal PF balance equations, also known as the mismatch equations, are expressed as a function of the complex bus voltages and generator injections as $\underline{\mathbf{s}}^{\mathrm{B}} + \underline{\mathbf{s}}^{D} - C_{\mathrm{G}}\underline{\mathbf{s}}^{G} - C_{\mathrm{S}}\underline{\mathbf{s}}^{\mathrm{S}} = \mathbf{0}$, where $C_{\mathrm{G}} \in \mathbb{R}^{N_{\mathrm{B}} \times N_{\mathrm{G}}}$ is the generator connectivity matrix and $C_{\mathrm{S}} \in \mathbb{R}^{N_{\mathrm{B}} \times N_{\mathrm{S}}}$ is the storage connectivity matrix. They specify location of the individual generators and storage devices in the power grid.

The real and reactive power units are megawatts (MW) and megavolt-ampere (MVAr), respectively. The complex apparent power is measured in megavolt amperes (MVA). For the analysis of electrical machines or electrical machine system, the per unit (p.u.) system is usually used. The per unit value of any quantity is defined as the ratio of actual value in any unit and the base or reference value in the same unit. Any quantity is converted into per unit measure by dividing the numeral value by the chosen base value of the same dimension. The per unit value are dimensionless.

The basic problem in the PF analysis is that current flow cannot be directed along any particular branch in the network, but is determined by Kirchhoff's laws and the relative impedances of the various branches. Power flow analysis is concerned with describing the operating state of an entire power system, that is, a network of generators, transmission lines, and loads that could span an area as large as several states. Given certain known quantities, such as the amount of power generated and consumed at different locations, PF analysis allows one to determine the remaining unknown quantities. The most important of these quantities are the voltages at locations throughout the transmission system, which, for alternating current, consist of both a magnitude and a time element or phase angle. Once the voltages are known, the currents flowing through every transmission link can be easily calculated.

However, the realistic power systems cannot be solved analytically since there

is no closed-form solution. We can only get at a numerical answer through a process of successive approximation or iteration. In order to find out what the voltage or current at any given point will be, we must in effect simulate the entire system. The so-called AC PF equations are a set of nonlinear and non-convex equations. The common way to solve the equations is to use iterative methods based on Newton's method, including algorithms such as Gauss-Seidel, Newton-Raphson (NR), Dishonest Newton-Raphson, Decoupled Load Flow, Fast Decoupled Load Flow or DC Power-Flow. An overview to the methods can be found in literature, Frank and Rebennack [2012]; von Meier [2006]; Venkatasubramanian and Tomsovic [2005]; Costa, VM ; Martins, Nelson; Pereira [1999] to mention a few. The most general iterative PF method is NR, with its quadratic rate of convergence. The idea of the NR method is to solve a linearization of the PF equations at the current iterate to find an update for the next iterate. If the power network is connected, the linearization is square and non-singular for fixed control variables. The solution of the linear system is the most expensive operation in the NR algorithm, thus other methods address this by various approximations and simplifications of the problem. For example, Dishonest Newton-Raphson algorithm updates the linearization of the PF equations only every couple of iterations, thus the costly factorization of the linear system are avoided. Decoupled Load Flow algorithm assumes weak coupling between active power – voltage magnitude, and reactive power – voltage phase pairs, resulting in two significantly smaller independent linear systems, compared to the NR algorithm. Additional simplifications, such as assumption that the voltage phase difference is negligible or neglecting the reactive power flows lead to methods such as Fast Decoupled Load Flow or DC Power Flow, which becomes linear problem. The disadvantage of the simplifying assumptions are that the methods become less robust, and the solution might not be valid for the stressed networks where the simplifying assumptions do not hold.

The iterative algorithms highly depend on the starting points. The NR has theoretical quadratic convergence rate, but this holds only for the points sufficiently close to the optimal solution. Therefore a bad choice can lead to a non-operable solution or even no solution in case of divergence. Recently, there has been introduced a non-iterative algorithm, the Holomorphic Embedding Load Flow Method (HELM) Trias [2015]. It is based on concepts of complex analysis and as such does not rely on any initial values or simplifying assumptions. However, it was shown that for obtaining the desired precision, the computation of HELM proved to be significantly slower compared to the NR method Sauter et al. [2017]. Therefore, hybrid approaches are recommended, such that HELM is used to calculate initial values with low precision and an iterative method is

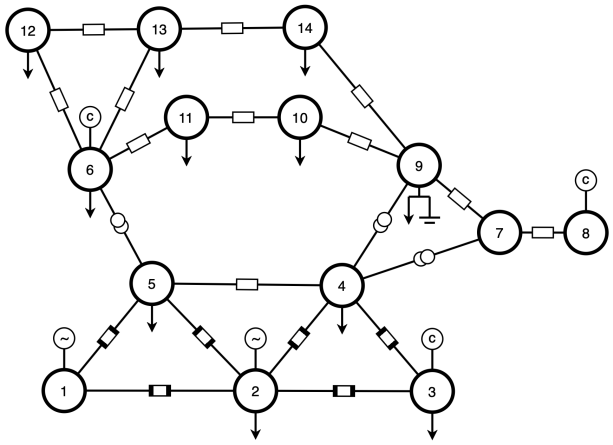


Figure 2.2. One line diagram for power grid with 14 nodes.

used to refine this initial solution.

2.2 Power grid components

The architecture of the power grid is usually illustrated using one-line diagram, using a simplified notation for representing a three-phase AC power system. Instead of representing each of three phases with a separate line or terminal, only one conductor is represented. A typical one-line diagram for a power grid is shown in Figure 2.2. Common electrical elements, including bus bars, conductors, transformers, circuit breakers and capacitors are shown by standardized schematic symbols. The individual components are described in the following paragraphs.

2.2.1 Electric generator

A generator is a device used to convert energy (e.g. mechanical, geothermal, solar) into electric power. A typical energy sources nowadays include steam, gas or water turbines. The proportion of renewable energies (REs) is called upon to increase in all major electricity markets. Currently, hydroelectricity is by far the largest contributor among the REs while the proportion of RE that comes from variable sources such as wind and solar is still relatively limited. The reason is that variability of wind and solar REs creates distinct challenges for integration into the larger power system. RE sources are fundamentally different from traditional generation methods because their energy output cannot be easily reg-

ulated since it depends on weather conditions. Therefore, energy from the RE sources, in particular, wind and solar, is intermittent, highly variable, and difficult to predict. These properties, combined with other unexpected power system failures, pose many challenges to power-grid operations that need to be overcome before high levels of RE penetration can be attained without losing the grid reliability and economic efficiency. One particularly difficult challenge is that large forecast errors in available RE cause large power imbalances in the grid. These imbalances need to be resolved by making fast power adjustments that typically have a high cost, both economical and environmental. This cost may be associated with load curtailments or with the operational and environmental costs of conventional fast-ramping generators Tinoco De Rubira and Hug [2016].

The general principle of a classical generator is that an external force spins the magnet placed inside a conductive wire, resulting in a magnetic flux that changes over time. This changing magnetic field induces an alternating voltage and current in a loop of wire surrounding the space. The AC in turn produces its own magnetic field, which acts to retard the motion of the spinning magnet. In this way, the interaction of the magnetic fields mediates the transfer of energy from mechanical movement into electricity. A standard type of generator used in utility power systems is a synchronous generator. The induction (asynchronous) generator is used in some specific applications and has some distinct and important properties. It operates without an independent source for its rotor field current, but the rotor field current appears by electromagnetic induction from the field of the armature current. Their one important application in power systems is in association with wind turbines. In this case, induction generators offer an advantage because they can readily absorb the erratic fluctuations of mechanical power delivered by the wind resource.

Generator complex power injections $\underline{\mathbf{s}}^G = \mathbf{p}^{\mathrm{G}} + j\mathbf{q}^{\mathrm{G}}$ are expressed in terms of active and reactive power components $\mathbf{p}^{\mathrm{G}}, \mathbf{q}^{\mathrm{G}} \in \mathbb{R}^{N_{\mathrm{G}}}$, respectively. The output of the generator is limited by the lower and upper bounds $\mathbf{p}^{\mathrm{G,min}} \leq \mathbf{p}^{\mathrm{G}} \leq \mathbf{p}^{\mathrm{G,max}}$, $\mathbf{q}^{\mathrm{G,min}} \leq \mathbf{q}^{\mathrm{G}} \leq \mathbf{q}^{\mathrm{G,max}}$. Such limits define a simple rectangular box bounds, while more detailed generator models involve generator capability P-Q curves which define a boundary within which the machine can operate safely in terms of excitation limits and flows of the active and reactive power. In practice, it is usually not possible to produce the maximum real output and the maximum reactive output simultaneously. Additionally, inner dynamics of power generator does not always allow to change the power production level instantaneously and so a maximum rate of change of produced power must be respected, resulting in the generator ramp limit constraints. A sparse $N_{\mathrm{B}} \times N_{\mathrm{G}}$ generator connection matrix C_{G} can be defined such that its (i, j)th element is one if generator j is located at

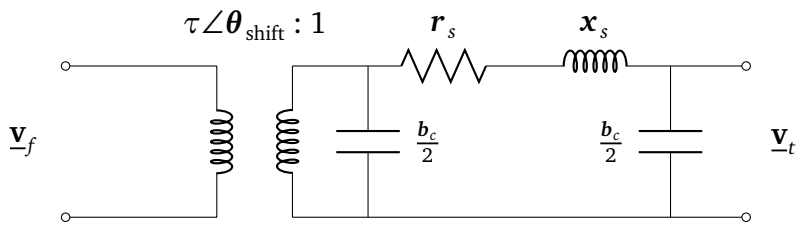


Figure 2.3. Branch π model.

bus i and zero otherwise. The $N_{\text{B}} \times 1$ vector of all bus injections from generators can then be expressed as $C_{\text{G}}\underline{\mathbf{s}}^G$.

2.2.2 Transmission line

The key component in the electric power transmission is the bulk movement of electrical energy from a generating site to an electrical substation via the high voltage transmission lines, as illustrated in Figure 2.1. Low resistance is generally desirable for power lines to minimize energy losses, but also because heating limits the conductor's ability to carry current. Conductors of overhead transmission and distribution lines typically consist of aluminum, which is lightweight and relatively inexpensive, while copper is the material of choice for underground cables because, while it is more expensive, it has a lower resistance than aluminum. In the latter case, the heat dissipation is more of an issue, whereas weight is not.

The transmission lines are modeled with a common π branch model, with a series impedance $\underline{\mathbf{z}}_s = \boldsymbol{r}_s + j\boldsymbol{x}_s$, $\underline{\mathbf{z}}_s \in \mathbb{C}^{N_{\text{L}}}, \boldsymbol{r}_s, \boldsymbol{x}_s \in \mathbb{R}^{N_{\text{L}}}$. In practice, it is convenient to use inverse of impedance, known as series admittance $\underline{\boldsymbol{y}}_s = \mathbf{g}_s + j\boldsymbol{b}_s$, $\underline{\boldsymbol{y}}_s \in \mathbb{C}^{N_{\text{L}}}$. The model also includes a total charging susceptance $\boldsymbol{b}_c \in \mathbb{R}^{N_{\text{L}}}$, which is usually divided into two equal parts and added to the both ends of the each branch. The branch model also considers an ideal phase shifting transformer $\mathcal{T} = \tau e^{j\boldsymbol{\theta}_{\text{shift}}}$ with the per unit tap ratio τ and the transformer shift angle $\boldsymbol{\theta}_{\text{shift}}$. The model is illustrated in Figure 2.3. The complex current injections $\underline{\mathbf{i}}_f$ and $\underline{\mathbf{i}}_t \in \mathbb{C}^{N_{\text{L}}}$, are determined using the branch admittance matrix $\underline{\mathbf{Y}}^{\text{L}}$ and voltage of the adjacent "from" and "to" buses, denoted as $\underline{\mathbf{v}}_f$ and $\underline{\mathbf{v}}_t \in \mathbb{C}^{N_{\text{L}}}$, respectively, connected to the corresponding ends of the line i,

$$\begin{pmatrix} \underline{\mathbf{i}}_f \\ \underline{\mathbf{i}}_t \end{pmatrix}_i = \underline{\mathbf{Y}}_i^{\text{L}} \begin{pmatrix} \underline{\mathbf{v}}_f \\ \underline{\mathbf{v}}_t \end{pmatrix}_i, \tag{2.1}$$

where $\underline{\mathbf{Y}}_i^{\mathrm{L}}$ is the branch admittance matrix

$$\underline{\mathbf{Y}}_i^{\mathrm{L}} = \begin{pmatrix} \left(\underline{\boldsymbol{y}}_s + j\frac{\boldsymbol{b}_c}{2}\right)\frac{1}{\tau^2} & -\underline{\boldsymbol{y}}_s \frac{1}{\tau e^{-j\boldsymbol{\theta}_{\text{shift}}}} \\ -\underline{\boldsymbol{y}}_s \frac{1}{\tau e^{j\boldsymbol{\theta}_{\text{shift}}}} & \underline{\boldsymbol{y}}_s + j\frac{\boldsymbol{b}_c}{2} \end{pmatrix}_i . \tag{2.2}$$

The sparse connectivity matrices $C_f, C_t \in \mathbb{Z}^{N_{\mathrm{L}} \times N_{\mathrm{B}}}$ are defined such that (i,j)th element of C_f and (i,k)th element of C_t are equal to 1 for each branch i, where branch i connects from bus j to bus k. All other elements are zero. The apparent power flow in the transmission lines, $\underline{\mathbf{s}}^f \in \mathbb{C}^{N_{\mathrm{L}}}$ and $\underline{\mathbf{s}}^t \in \mathbb{C}^{N_{\mathrm{L}}}$ can be determined using the bus voltages $\underline{\mathbf{v}}$ as $\underline{\mathbf{s}}^f = (C_f \underline{\mathbf{v}})\underline{\mathbf{i}}_f^*$ and $\underline{\mathbf{s}}^t = (C_t \underline{\mathbf{v}})\underline{\mathbf{i}}_t^*$.

Real-world transmission lines are limited by the instantaneous amount of power that can flow through the lines due to the thermal limits. The apparent power flow in the transmission lines are therefore limited by the power injections at both ends of the lines, which cannot exceed a prescribed upper bound $\boldsymbol{s}_{\mathrm{L}}^{\max}$. Squared values of the apparent power magnitude are usually used in practice, such that $\underline{\mathbf{s}}^f (\underline{\mathbf{s}}^f)^* \leq (\boldsymbol{s}_{\mathrm{L}}^{\max})^2$.

The branch model also consists of transformer with a tap ratio with magnitude τ and phase shifting angle $\boldsymbol{\theta}_{\text{shift}}$ and is assumed to be located at the "from" end of the branch.

The properties of the branch admittance matrix $\underline{\mathbf{Y}}^{\mathrm{L}}$ are used to build the nodal admittance matrix $\underline{\mathbf{Y}}^{\mathrm{B}} = \boldsymbol{G} + j\boldsymbol{B} \in \mathbb{C}^{N_{\mathrm{B}} \times N_{\mathrm{B}}}$, which represents the nodal admittance of the buses in a power system. $\boldsymbol{G}$ is the conductance and $\boldsymbol{B}$ the susceptance. The nodal admittance matrix is used in the power system analysis to determine the power flow in the network. In realistic systems, the $\underline{\mathbf{Y}}^{\mathrm{B}}$ matrix is quite sparse since each bus in a real power system is usually connected to only a few other buses through the transmission lines. The nodal power $\underline{\mathbf{s}}^{\mathrm{B}} = \mathbf{p}^{\mathrm{B}} + j\mathbf{q}^{\mathrm{B}}, \underline{\mathbf{s}}^{\mathrm{B}} \in \mathbb{C}^{N_{\mathrm{B}}}$ is accounting for all the power flow that is entering or leaving a particular node i via the transmission lines. Assume bus i that is connected to other n_k nodes with indices $\mathcal{B}(i)$, we can write the power flow equation for the bus i as

$$\underline{\mathbf{s}}_i^{\mathrm{B}} = \underline{\mathbf{v}}_i \left(\sum_{k \in \mathcal{B}(i)} (\boldsymbol{G}_{ik} + j\boldsymbol{B}_{ik}) \underline{\mathbf{v}}_k \right)^* . \tag{2.3}$$

Written in the matrix form, the nodal power flow is defined by $\underline{\mathbf{s}}^{\mathrm{B}} = \underline{\mathbf{v}}(\underline{\mathbf{Y}}^{\mathrm{B}}\underline{\mathbf{v}})^*$.

2.2.3 Transformer

High voltages are crucial for power transmission over long distances. Closer to the end users of electricity, however, safety prohibits the use of equipment at

excessively high voltages. In the design of power delivery systems, the greater energy efficiency of high voltage and low current must therefore be weighed against safety and capital cost. Rather than having to settle for some intermediate voltage as a compromise, the use of transformers makes it possible to operate the different parts.

A transformer is a device for changing the voltage in an AC circuit and is used to provide the effective interface between the high- and low-voltage parts of the system. It consists of two conductor coils, as illustrated in Figure 2.3, that are connected not electrically but through magnetic flux. As a result of electromagnetic induction, an alternating current in one coil will set up an alternating current in the other. In the primary coil, the AC supplied by a generator produces a magnetic field, or flux, inside the core of the coil. Like the conductors inside a generator, transformer coils are wound around a core of magnetically susceptible material, generally some type of iron, to enhance the magnetic field. The magnetic flux resulting from the current is proportional to both the current's magnitude and the number of turns in the coil.

In a real transformer, some power is dissipated in the form of heat. A portion of these power losses occur in the conductor windings due to electrical resistance and are referred to as copper losses. Even a small percentage of losses in a large transformer corresponds to a significant amount of heat that must be dealt with. Large transformers like those at substations or power plants require the heat to be removed from the core and windings by active cooling, generally through circulating oil that simultaneously functions as an electrical insulator.

2.2.4 Load

Load refers to any device in which power is being dissipated (i.e., consumed). From the circuit perspective, a load is defined by its impedance. In the larger context of power systems, loads are usually modeled in an aggregated way: rather than considering an individual appliance, load may refer to an entire household, a city block, or all the customers within a certain region. In the language of electric utilities, the term load therefore has attributes beyond impedance that relate to aggregate behavior, such as the timing of demand. Serving the instantaneous demand under diverse circumstances is the central challenge in designing and operating power systems, and the one that calls for the majority of investment and effort.

Load is an externally given quantity, a variable beyond control, that is in any given time to be met by supply at any costs. This assumption is codified in the social contract between utilities – or even just as transmission and distribution

service companies – and the public, which insists upon the utilities' fundamental obligation to serve the load, since the electricity cannot be stored in sufficient quantities on large scale. From the standpoint of economics as well as logistics, a relatively flat load duration curve with a high load factor is clearly desirable for utilities. This is because the cost of providing service consists in large part of investments related to peak capacity, whereas revenues are generally related to total energy consumed (i.e., average demand). A pronounced peak indicates a considerable effort that the service provider must undertake to meet demand on just a few occasions, although the assets required to accomplish this will tend not to be utilized much during the remainder of the year.

Each bus has an associated complex power demand $\underline{\mathbf{s}}^D = \mathbf{p}^{\mathrm{D}} + j\mathbf{q}^{\mathrm{D}}$, which is assumed to be known at all of the buses and is modeled by a static polynomial (ZIP) model Zimmerman and Murillo-Sanchez [2016]. If there are no loads connected to the bus i then $\{\underline{\mathbf{s}}^D\}_i = \mathbf{0}$.

2.2.5 Energy storage device

Energy storage devices are cruicial components in the grid when it comes to a high penetration of variable RE sources. Traditionally, pumped-storage hydroelectricity is used for load balancing which allows energy from intermittent sources (such as solar, wind) and other renewables, or excess electricity from continuous base-load sources (such as coal or nuclear) to be saved for periods of higher demand. Other storage technology include lithium-ion batteries, flywheels or compressed air storage. Each device varies in its application and properties, with varying time scale of operation or storage capacity.

Nowadays, lithium-ion (Li-ion) battery energy storage system represent alternative solution in order to guarantee grid stability in distribution areas with large PV penetration, where the increasing share of PV generation at regional and national scales brings technical and economic challenges related to the variability and uncertainty associated with PV generation Sevilla et al. [2018]. At the moment, Li-ion batteries are considered as the most relevant technology for distribution grids given their maturity level, modular design and capability for both short-term and mid-term applications. Additionally, Li-ion batteries are advantageous due to its capability for charging and discharging efficiently at high power rates even with limited battery capacity.

Each storage unit in the network is modeled by two network power injections. A positive active power injection $\mathbf{p}^{\mathrm{Sd},i} \in \mathbb{R}$, $\mathbf{p}^{\mathrm{Sd},i} \geq 0$ models the discharging of storage unit i. A negative active power injection $\mathbf{p}^{\mathrm{Sc},i} \in \mathbb{R}$, $\mathbf{p}^{\mathrm{Sc},i} \leq 0$ models the charging of storage unit i. The vector of active storage power injections $\mathbf{p}^{\mathrm{S}} \in \mathbb{R}^{2N_{\mathrm{S}}}$

is defined as

$$\mathbf{p}^{\mathrm{S}} = \left[\mathbf{p}^{\mathrm{Sd},1}, \cdots, \mathbf{p}^{\mathrm{Sd},N_{\mathrm{S}}}, \mathbf{p}^{\mathrm{Sc},1}, \cdots, \mathbf{p}^{\mathrm{Sc},N_{\mathrm{S}}}\right] \tag{2.4}$$

and bounded by $\mathbf{p}^{\mathrm{S,min}} \leq \mathbf{p}^{\mathrm{S}} \leq \mathbf{p}^{\mathrm{S,max}}$. Identical definitions apply for the reactive storage power injections $\mathbf{q}^{\mathrm{Sd},i}$, $\mathbf{q}^{\mathrm{Sc},i}$, $\mathbf{q}^{\mathrm{S}}$ with bounds $\mathbf{q}^{\mathrm{S,min}}$ and $\mathbf{q}^{\mathrm{S,max}}$. Together, they yield the complex storage power injections $\underline{\mathbf{s}}^{\mathrm{S}} = \mathbf{p}^{\mathrm{S}} + j\mathbf{q}^{\mathrm{S}}$.

The storage connectivity matrix $C_{\mathrm{S}} \in \mathbb{R}^{N_{\mathrm{B}} \times 2N_{\mathrm{S}}}$ maps the $2N_{\mathrm{S}}$ storage injections $\underline{\mathbf{s}}^{\mathrm{S}}$ to the N_{B} buses. The connectivity matrix has the entries (i,k) and $(i,N_{\mathrm{S}}+k)$ equal to one if the injection $\underline{\mathbf{s}}^{\mathrm{S}}_k$ of the storage device k is into bus i and zero otherwise.

In our model, the storage size $\boldsymbol{\epsilon}^{\mathrm{max}}_{\mathrm{S}}$ is chosen to contain up to two hours of the nominal active power demand $\mathbf{p}^{\mathrm{D}}_k$ of the bus k. If the storage device i is located at bus k, then $\boldsymbol{\epsilon}^{\mathrm{max}}_{\mathrm{S},i} = 2\,\mathbf{p}^{\mathrm{D}}_k$ and $\boldsymbol{\epsilon}^{\mathrm{min}}_{\mathrm{S},i} = 0$. The initial state of charge is 70%, which represents $\boldsymbol{\epsilon}_0 = 0.7\boldsymbol{\epsilon}^{\mathrm{max}}_{\mathrm{S}}$. The storage device power ratings are limited to allow a complete discharging and charging within three hours and two hours, respectively. Therefore, $\mathbf{p}^{\mathrm{Sd},i\,\mathrm{max}} = \frac{1}{3}\boldsymbol{\epsilon}^{\mathrm{max}}_{\mathrm{S}}$ and $\mathbf{p}^{\mathrm{Sc},i\,\mathrm{min}} = -\frac{1}{2}\boldsymbol{\epsilon}^{\mathrm{max}}_{\mathrm{S}}$. All storage device discharging and charging efficiencies are chosen as $\eta_{\mathrm{d}} = 0.97$ and $\eta_{\mathrm{c}} = 0.95$.

Other components

Elements such as a capacitor or inductor, referred to as shunt elements, are modeled as fixed impedance to ground at a given bus. The admittance of the shunt element at the buses are given as $\underline{\boldsymbol{y}}_{sh} = \mathbf{g}_{sh} + j\boldsymbol{b}_{sh}$, where $\underline{\boldsymbol{y}}_{sh} \in \mathbb{C}^{N_{\mathrm{B}}}, \mathbf{g}_{sh}, \boldsymbol{b}_{sh} \in \mathbb{R}^{N_{\mathrm{B}}}$.

Other crucial components of the grid are circuit breakers and switches. Breakers serve as protective devices that open automatically in the event of a fault, that is, when a protective relay indicates excessive current due to some abnormal condition. Switches are control devices that can be opened or closed deliberately to establish or break a connection. An important difference between circuit breakers and switches is that breakers are designed to interrupt abnormally high currents (as they occur only in those very situations for which circuit protection is needed), whereas regular switches are designed to be operable under normal currents. Reasons for switching include contingencies, work clearances, service restoration following an outage, managing overloads, and enhancing system efficiency. For example, a contingency might be a fault on one line that has to be isolated from the system, and other connections need to be rearranged so as to redistribute the load of the lost line among them. Further details regarding the grid infrastructure can be found in Momoh [2000]; von Meier [2006]; Zimmerman and Murillo-Sanchez [2016]; Frank and Rebennack [2012].

2.3 Power grid models

During the development process of robust and fast power grid analysis methods, it is crucial to have a set of realistic problems, which can be used to evaluate various algorithms. However, the power grid infrastructure represents the classified information of national interest, which might be easily targeted and misused to threaten the national security. Thus the realistic data are handled very carefully and are not available to public. However, many synthetic or simplified models with properties closely representing the actual system are available.

For example Pegase Josz et al. [2016]; Fliscounakis et al. [2013a] data accurately represent the size and complexity of part of the European high voltage transmission network. The network contains 1,354 buses, 260 generators, and 1,991 branches and it operates at 380 and 220 kV. The data are fictitious and do not correspond to real world data. They can thus be used to validate methods and tools but should not be used for operation and planning of the European grid. Representations of the power grid with different granularity exist, from 1,354 up to 13,659 buses. Similarly, the dynamic model of continental Europe is available from European Network of Transmission System Operators (ENTSO-E), which is designed for performing transient analysis Sevilla et al. [2017]; Semerow et al. [2015].

ACTIVS is a collection of entirely synthetic cases consisting from 200 up to 10,000 buses. The grid is geographically situated in the eastern US. The case is designed with a 500, 230, 161, and 115 kV transmission network to serve a load that roughly mimics the actual population of its geographic footprint. The synthetic transmission system was designed by algorithms described in Birchfield et al. [2017] to be statistically similar to actual transmission system models but without modeling any actual lines.

The data for the Polish 400, 220, and 110 kV networks during different operating conditions, such as summer 2008 morning peak or during winter 2007–08 evening peak conditions are available Zimmerman et al. [2011] and include some equivalents of the German, Czech, and Slovak networks.

The size and complexity of French high and very high voltage transmission networks are also available Josz et al. [2016]. These data are snapshots of French very high voltage and high voltage grids in 2013. These data can be used to validate mathematical methods and tools but should not be used for operation nor planning of the French or European grids.

Various characteristics for selected benchmark cases[1] with increasing com-

[1]For abbreviation purposes the prefix “case” was removed from all benchmark names.

plexity are listed in Table 2.1. In addition to the standard benchmark cases distributed with MATPOWER package, there are four larger cases, namely case21k–case193k, built from the Polish system winter 2007–08 evening peak power flow data (case3012wp), considering the largest generator outage and line contingencies. The table lists number of buses, generators and lines for each benchmark case, together with some properties of the optimization problem described in Chapter 3.

The power grid data are distributed with the MATPOWER Zimmerman et al. [2011] package in the *MATPOWER case* format Zimmerman and Murillo-Sanchez [2016] or are available as PGLib-OPF Babaeinejadsarookolaee et al. [2019], an open-access benchmark library, where all the network data are provided under a creative commons license. The latter project is the official IEEE Power and Energy Society OPF benchmark library, which has replaced archives such as NESTA Coffrin et al. [2014].

Additionally, CIGRE-Networks Rudion et al. [2006]; CIGRE Task Force [2014] were developed by the CIGRE Task Force to facilitate the analysis and validation of new methods and techniques that aim to enable the economic, robust, and environmentally responsible integration of distributed energy resources (DER). CIGRE-Networks are a set of comprehensive reference systems that allow the analysis of DER integration at high voltage, medium voltage, and low voltage at the desired degree of detail.

Table 2.1. Benchmark cases statistics including the number of buses N_{B}, generators N_{G}, transmission lines N_{L} and nonlinear equality and inequality constraints (considering OPF formulation with polar voltage coordinates).

Benchmark	N_{B}	N_{G}	N_{L}	$\lvert x\rvert$	$\lvert c_E(x)\rvert$	$\lvert c_I(x)\rvert$
1951rte	1,951	391	2,596	4,634	3,902	4,198
2383wp	2,383	327	2,896	5,420	4,766	5,792
2868rte	2,868	599	3,808	6,858	5,736	4,562
ACTIVSg2000	2,000	544	3,206	4,864	4,000	6,412
2869pegase	2,869	510	4,582	6,758	5,738	5,486
2737sop	2,737	399	3,506	5,912	5,474	6,538
2736sp	2,736	420	3,504	6,012	5,472	6,538
2746wop	2,746	514	3,514	6,354	5,492	6,614
2746wp	2,746	520	3,514	6,404	5,492	6,558
3012wp	3,012	502	3,572	6,794	6,024	7,144
3120sp	3,120	505	3,693	6,836	6,240	7,386
3375wp	3,374	596	4,161	7,706	6,748	8,322
6468rte	6,468	1,295	9,000	13,734	12,936	4,626
6470rte	6,470	1,330	9,005	14,462	12,940	6,220
6495rte	6,495	1,372	9,019	14,350	12,990	6,218
6515rte	6,515	1,388	9,037	14,398	13,030	6,262
9241pegase	9,241	1,445	16,049	21,372	18,482	12,590
13659pegase	13,659	4,092	20,467	35,502	27,318	0
ACTIVSg10k	10,000	2,485	12,706	23,874	20,000	20,488
Large-scale benchmarks						
ACTIVSg25k	25,000	4,834	32,230	57,558	50,000	46,660
ACTIVSg70k	70,000	10,390	88,207	156,214	140,000	137,234
case21k	21,084	2,692	25,001	54,091	42,168	50,002
case42k	42,168	5,384	50,001	107,027	84,336	100,002
case99k	99,396	12,689	117,860	250,703	198,792	235,720
case193k	192,768	24,611	228,574	485,135	385,536	457,148

Chapter 3

Optimization problems in the power grid

Historically, when the first optimal power flow (OPF) models were formulated Carpentier [1962], the usual setting was that of a monopolistic energy producer. The responsibility of such a producer, typically operating in a national setting, spanned from the electrical production, transmission, and distribution in one given area, comprising also the regulation of exchanges with neighboring regions. In the liberalized markets that are nowadays prevalent, the decision chain is instead decentralized and significantly more complex, as shown in the simplified scheme of Figure 3.1. In a typical setting, companies owning generation assets have to bid their generation capacity over one of the market operators (MO). Alternatively, or in addition, they can stipulate bilateral contracts with final users (representing major loads such as industrial centers) or with wholesales and traders. Once they have received the bids and offers, the MO clears the energy market and defines clearing prices. A transmission system operator (TSO), in possession of the transmission infrastructure, then has the duty to ensure safe delivery of the energy, which in turns means different duties such as real-time frequency-power balancing, spinning reserve satisfaction, voltage profile stability, and enforcing real-time network capacity constraints Tahanan et al. [2015].

From the perspective of the energy trading and wholesale market, efficient energy trading relies on high-fidelity price prediction systems with short response times. However, the consideration of realistic engineering constraints encountered in daily power grid operations results in high computational complexity of pricing models with unacceptable response times. Furthermore, the liberalized wholesale power markets, which consist of energy, ancillary service, and

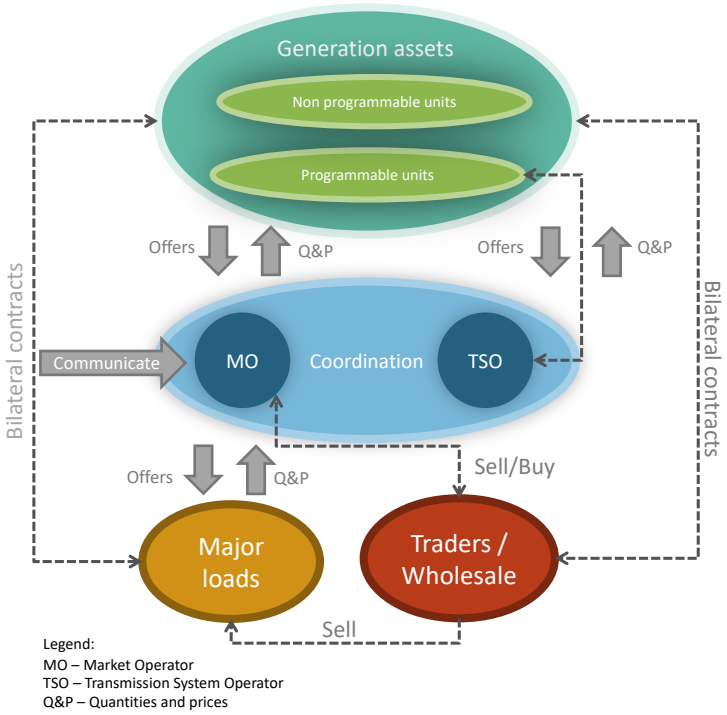


Figure 3.1. General structure of electrical systems, presenting the different entities solving OPF problems and their interactions.

capacity markets, include a growing number of different types of DER. Real-time responses are essential in modern trading environments and are not currently provided by energy trading software. The research in this area focuses on developing novel computational tools for the market pricing mechanism that meet real-time responses required by modern trading environments Helman [2019]. The goal of the trading entities is to have available high-fidelity price forecasting tools in order to gain several advantages, such as better identifying important events along the forecasting horizon, understanding when and if the market might be panicking from particular events, facilitating the price discovery, and enhancing the understanding of physical behavior of power plants. Generally speaking, representing the potential price outcome over a given time horizon and for a given country is essential for both trading and risk management purposes. OPF is a fundamental building block of such computations.

On the other hand, TSO operates the extrahigh voltage transmission grid and coordinates energy exchanges and operational activities with neighboring utilities and is responsible for secure and reliable grid planning and operation. Extensive operational planning is performed on different time frames to ensure the

secure operation of the transmission system. Scheduled maintenance of the major power plants and grid elements (lines, transformers, substation equipment, etc.) are coordinated in order to avoid insecure situations. Analyses are performed on yearly, monthly, weekly and daily basis and the core computational tool is based on an OPF engine López et al. [2015].

3.1 Optimal power flow

Since the formulation of OPF by Carpentier Carpentier [1962] as a continuous nonlinear programming (NLP) problem, OPF has become one of the most important and widely studied constrained nonlinear optimal control problems, since it is a fundamental building block in power system research, operation and it is widely used in planning problems. As the power industry has become a deregulated environment, operation is strongly influenced by a competitive market. The security and economical issues of power systems are coordinated more tightly than before. Thus, the need for fast and robust optimization tools that consider both security and economy is more demanding than before to support the system operation and control.

OPF is concerned with the optimization of the operation of an electric power network subject to physical constraints imposed by electrical laws and engineering limits. The objective is to identify the operating configuration that best meets a particular set of evaluation criteria. These criteria may include the most economic generator dispatch which minimizes hourly fuel cost, transmission line losses, and various requirements concerning the system's security, or resilience with respect to disturbances. The constraints are imposed by the power flow equations or may be related to various limits, including voltage, thermal line flow, transformer tap or generator limits.

3.1.1 Problem formulation

The OPF problem is defined in terms of the conventional economic dispatch problem, aiming at determining the optimal settings for optimization variables. The standard formulation of the OPF problem takes the form of a general NLP prob-

lem, with the following form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) && \text{(3.1a)} \\ \text{subject to} \quad & c_E(x) = 0, && \text{(3.1b)} \\ & c_I(x) \leq 0, && \text{(3.1c)} \\ & x_{\min} \leq x \leq x_{\max}. && \text{(3.1d)} \end{aligned}$$

The objective function $f(x)$ consists of polynomial costs of generator injections, the equality constraints $c_E(x)$ are the nodal balance equations, the inequality constraints $c_I(x)$ are the branch flow limits, and the $x_{\min}$ and $x_{\max}$ bounds include slack bus angles, voltage magnitudes, and active and reactive generator power injections. The optimization vector x for the standard AC OPF problem consists of the $N_{\mathrm{B}} \times 1$ vectors of voltage angles $\boldsymbol{\theta}$ and magnitudes $\mathbf{v}$ and the $N_{\mathrm{G}} \times 1$ vectors of generator real and reactive power injections $\mathbf{p}^{\mathrm{G}}$ and $\mathbf{q}^{\mathrm{G}}$, defined in Chapter 2:

$$x = \begin{bmatrix} \boldsymbol{\theta} & \mathbf{v} & \mathbf{p}^{\mathrm{G}} & \mathbf{q}^{\mathrm{G}} \end{bmatrix}. \tag{3.2}$$

The complex voltage in polar coordinates is defined as $\underline{\mathbf{v}} = \mathbf{v}e^{j\boldsymbol{\theta}}$ and the complex power in rectangular form is expressed as $\underline{\mathbf{s}}^G = \mathbf{p}^{\mathrm{G}} + j\mathbf{q}^{\mathrm{G}}$. The objective function $f(x)$ in (3.1a) is simply a summation of individual polynomial cost functions f_P^i of active power injections for each generator:

$$f(\mathbf{p}^{\mathrm{G}}) = \sum_{i=1}^{N_{\mathrm{G}}} f_P^i(\mathbf{p}^{\mathrm{G}i}). \tag{3.3}$$

The equality constraints in (3.1b) are the complex power balance equations

$$c_E^S(\underline{\mathbf{v}}, \underline{\mathbf{s}}^G) = \underline{\mathbf{s}}^{\mathrm{B}} + \underline{\mathbf{s}}^D - C_{\mathrm{G}}\underline{\mathbf{s}}^G = 0, \tag{3.4}$$

or when expressed in terms of the full set of $2N_{\mathrm{B}}$ real and reactive components, the constraints become nonlinear and nonconvex

$$c_E^P(\boldsymbol{\theta}, \mathbf{v}, \mathbf{p}^{\mathrm{G}}) = \mathbf{p}^{\mathrm{B}} + \mathbf{p}^{\mathrm{D}} - C_{\mathrm{G}}\mathbf{p}^{\mathrm{G}} = 0, \tag{3.5}$$

$$c_E^Q(\boldsymbol{\theta}, \mathbf{v}, \mathbf{q}^{\mathrm{G}}) = \mathbf{q}^{\mathrm{B}} + \mathbf{q}^{\mathrm{D}} - C_{\mathrm{G}}\mathbf{q}^{\mathrm{G}} = 0. \tag{3.6}$$

The nodal power vector is defined as $\underline{\mathbf{s}}^{\mathrm{B}} = \mathbf{p}^{\mathrm{B}} + j\mathbf{q}^{\mathrm{B}}$, the nodal power demand vector is expressed as $\underline{\mathbf{s}}^D = \mathbf{p}^{\mathrm{D}} + j\mathbf{q}^{\mathrm{D}}$. The inequality constraints (3.1c) consist of two sets of N_{L} branch flow limits expressed as nonlinear functions of the bus voltage angles and magnitudes, one for the each end of the branch:

$$c_I^P(\boldsymbol{\theta}, \mathbf{v}) = \left|\underline{\mathbf{s}}^f(\boldsymbol{\theta}, \mathbf{v})\right| - \boldsymbol{s}_{\mathrm{L}}^{\max} \leq 0, \tag{3.7}$$

$$c_I^Q(\boldsymbol{\theta}, \mathbf{v}) = \left|\underline{\mathbf{s}}^t(\boldsymbol{\theta}, \mathbf{v})\right| - \boldsymbol{s}_{\mathrm{L}}^{\max} \leq 0. \tag{3.8}$$

The flows $\underline{\mathbf{s}}^f$ and $\underline{\mathbf{s}}^t$ are the apparent power flows expressed in megavolt ampere (MVA), though real power or current can also be used.

The variables bounds (3.1d) include an equality constraint for the voltage angle at a slack bus $\mathcal{I}_{\text{slack}}$ (equivalently expressed as $x_{\min}^{\mathcal{I}_{\text{slack}}} = x_{\max}^{\mathcal{I}_{\text{slack}}} = \boldsymbol{\theta}^{\text{slack}}$). Since the value is known, the variable is usually removed internally in the optimizer, which is the case for IPOPT. The upper and lower bounds are imposed on all bus voltage magnitudes and real and reactive generator injections:

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}^{\text{slack}}, \qquad i \in \mathcal{I}_{\text{slack}}, \tag{3.9}$$

$$\mathbf{v}^{i,\min} \leq \mathbf{v}^i \leq \mathbf{v}^{i,\max}, \qquad i = 1, 2, \ldots N_{\text{B}}, \tag{3.10}$$

$$\mathbf{p}^{\text{G}i,\min} \leq \mathbf{p}^{\text{G}i} \leq \mathbf{p}^{\text{G}i,\max}, \qquad i = 1, 2, \ldots N_{\text{G}}, \tag{3.11}$$

$$\mathbf{q}^{\text{G}i,\min} \leq \mathbf{q}^{\text{G}i} \leq \mathbf{q}^{\text{G}i,\max}, \qquad i = 1, 2, \ldots N_{\text{G}}. \tag{3.12}$$

The table 2.1 lists the number of decision variables together with a number of nonlinear equality and inequality constraints (considering OPF formulation with polar and voltage coordinates).

Rectangular vs. polar coordinates for voltage

The AC OPF problem takes different forms based on the different representations of the complex bus voltages, which can be represented either in rectangular or polar coordinates. The standard formulation uses polar voltage representation $\underline{\mathbf{v}} = \mathbf{v}e^{j\boldsymbol{\theta}}$. The optimization vector x, considering alternative rectangular coordinates, includes the real and imaginary parts of the complex voltage, denoted, respectively, by $\boldsymbol{u}$ and $\boldsymbol{w}$, where $\underline{\mathbf{v}} = \boldsymbol{u} + j\boldsymbol{w}$:

$$x = \begin{bmatrix} \boldsymbol{u} & \boldsymbol{w} & \mathbf{p}^{\text{G}} & \mathbf{q}^{\text{G}} \end{bmatrix}. \tag{3.13}$$

The objective function remains unchanged, but in the case of rectangular voltage representation, the nodal power balance constraints (3.5) and (3.6) and branch flow constraints (3.7) and (3.8) are implemented as quadratic functions of $\boldsymbol{u}$ and $\boldsymbol{w}$:

$$c_E^P(\boldsymbol{u}, \boldsymbol{w}, \mathbf{p}^{\text{G}}) = \mathbf{p}^{\text{B}}(\boldsymbol{u}, \boldsymbol{w}) + \mathbf{p}^{\text{D}} - C_{\text{G}}\mathbf{p}^{\text{G}} = 0, \tag{3.14}$$

$$c_E^Q(\boldsymbol{u}, \boldsymbol{w}, \mathbf{q}^{\text{G}}) = \mathbf{q}^{\text{B}}(\boldsymbol{u}, \boldsymbol{w}) + \mathbf{q}^{\text{D}} - C_{\text{G}}\mathbf{q}^{\text{G}} = 0, \tag{3.15}$$

$$c_I^P(\boldsymbol{u}, \boldsymbol{w}) = \left|\underline{\mathbf{s}}^f(\boldsymbol{u}, \boldsymbol{w})\right| - \boldsymbol{s}_{\text{L}}^{\max} \leq 0, \tag{3.16}$$

$$c_I^Q(\boldsymbol{u}, \boldsymbol{w}) = \left|\underline{\mathbf{s}}^t(\boldsymbol{u}, \boldsymbol{w})\right| - \boldsymbol{s}_{\text{L}}^{\max} \leq 0. \tag{3.17}$$

In the case of rectangular formulation, the voltage angle constraint at the slack bus (3.9) and voltage magnitude limits (3.10) cannot be simply applied as bounds on optimization variables. These constrained quantities become nonlinear trigonometric functions of $\boldsymbol{u}$ and $\boldsymbol{w}$:

$$\boldsymbol{\theta}_i(\boldsymbol{u}_i, \boldsymbol{w}_i) = \boldsymbol{\theta}^{\text{slack}}, \qquad i \in \mathcal{I}_{\text{slack}}, \tag{3.18}$$

$$\mathbf{v}^{i,\min} \leq \mathbf{v}^i(\boldsymbol{u}_i, \boldsymbol{w}_i) \leq \mathbf{v}^{i,\max}, \qquad i = 1, 2, \ldots N_{\text{B}}. \tag{3.19}$$

Current vs. power for nodal balance constraints

Another variation of the standard AC OPF problem uses current balance constraints in place of the power balance constraints. Complex power balance constraints are expressed as (3.4), where (3.5)–(3.6) or (3.14)–(3.15) are their real and imaginary components, respectively. The complex current balance constraints are expressed as

$$c_E(\underline{\mathbf{v}}, \underline{\mathbf{s}}^G) = \underline{\mathbf{i}}^{\text{B}} + [\underline{\mathbf{v}}^*]_D^{-1}(\underline{\mathbf{s}}^D - C_{\text{G}}\underline{\mathbf{s}}^G)^* = 0, \tag{3.20}$$

where $[\underline{\mathbf{v}}^*]_D^{-1}$ is a diagonal matrix whose ith diagonal entry is $1/\mathbf{v}_i^*$, that is, $e^{j\boldsymbol{\theta}_i}/\mathbf{v}^i$ or $1/(\boldsymbol{u}_i - j\boldsymbol{w}_i)$. For further details, comprehensive explanation of the equations and discussion regarding the modelling aspects, see Zimmerman and Murillo-Sanchez [2016].

3.1.2 Solution approaches

The AC OPF problem is a non-convex optimization problem that is difficult to solve quickly and reliably for large-scale grids. One important advantage of NLP for OPF is that it naturally captures nonlinearities in stressed power system behavior rendering the NLP solution technique into an excellent tool for modeling and simulation of modern power system operations. However, the performance of NLP algorithms need to be extensively investigated for a variety of real-life power networks and their applicability need to be assessed for real-time power systems operations.

An important complexity source of OPF is its highly non-convex nature, which makes solving this optimisation problem computationally hard. This is why the operators who are dealing with large power systems usually use highly simplified OPF models. To manage the complexity to a reasonable level, research has focused on solution techniques based on (i) problem relaxation or approximation (ii) power grid decomposition, and (ii) high-performance nonlinear large-scale optimization approaches.

An efficient algorithm for constructing accurate linear approximations of line flow constraints was presented in Shchetinin et al. [2019]. These approximations reduce the complexity of the optimization problem while ensuring that the solution is physically meaningful and has a high quality. The algorithm is based on an in-depth analysis of the feasible set of the line flow constraints. Linear and quadratic programming (LP/QP) based OPF problems were formulated in Fortenbacher and Demiray [2019] that can be used for power system planning and operation. They consider a linear power flow and branch flow approximation and derive a power loss approximation in the form of absolute value functions that are suitable to cover a broader operating range. The proposed formulation is also an approximation and there are loss regions that are underestimated above the certain operating values, which results in an underestimation of voltage angles and magnitudes. Hence, there is no guarantee that the approximated OPF solutions will lie inside the original feasible solution space.

One of the aspects of the current research trends for the future smart grid is the possibility of devising distributed algorithms for solving a global problem, which decompose the power grid and the associated optimization problem into smaller tasks. This corresponds to the idea of a decentralized access to generation/storage resources, as well as to the much more challenging task of decentralized control. The nonlinear decomposition of the grid into zones with the aid of additional user information was performed e.g. in Hug-Glanzmann and Andersson [2009]. The idea is to partition the network in multiple regions was also used in Erseghe [2015], where a distributed algorithm based upon alternating direction method of multipliers (ADMM) was proposed. ADMM Boyd et al. [2011] is an algorithm that solves convex optimization problems by breaking them into smaller pieces, each of which are then easier to handle. However, the non-convexity of the OPF introduces a lot of difficulties for the ADMM algorithm. The improved convergence speed, proposed in Engelmann et al. [2019], comes at the cost of increasing the communication effort per iteration. Therefore, inexact Hessians are used to reduce the communication volume.

Over the last five decades, almost every mathematical programming approach that can be applied to OPF has been attempted. Sequential linear programming has been presented by Stott and Hobson Stott and Hobson [1978] and Alsac et al. Alsac et al. [1990]. Despite its good performance, the linear model introduces inaccuracies and might lead to infeasible solutions. Sequential quadratic programming (SQP) was applied by Burchett Burchett et al. [1982] delivering accurate and feasible solutions but the computational times are relatively high, rendering the entire solution approach inappropriate for large-scale power systems. Linear and nonlinear interior point methods were applied by Vargas et

al. Vargas et al. [1993]; Granville [1994]; Torres and Quintana [1998], and semidefinite programming by Low Lavaei and Low [2012]; Low [2014]. Several extensions of these algorithmic approaches are reported in recent review papers; see, e.g., Huneault and Galiana [1991]; Momoh et al. [1999]; Frank et al. [2012]. Modern trends in power grid operations and modeling, however, render approximation-based optimization techniques less attractive for coping with stressed operating conditions. As a result, there is a great demand for new algorithms and software tools able to address strong nonlinearities in system behavior, in order to guarantee reliable and economic system operation.

The high-performance nonlinear large-scale optimization approaches focus on problems such as security constrained OPF, important from the perspective of the TSO, or the multiperiod OPF that are necessary to predict the evolution of the energy markets. The formulation of such problems and their solution approaches are discussed in the following sections.

3.2 Security of the power grid

The security constrained OPF (SCOPF) Monticelli et al. [1987], is an extension of the OPF problem, which finds an optimal operational state but at the same time takes into account a set of security constraints arising from the operation of the system under a set of postulated contingencies. It guarantees that the whole power system can work under the nominal long-term cost-efficient operation plan, but can also remain in the operational state when some of the predetermined contingencies occur, such as failures or outages of equipment in the power system. The SCOPF has become an essential tool for many transmission system operators for the operational planning, and real time operation of their system. Some of the SCOPF models may allow limited corrective actions after the accident occurs Phan and Sun [2015], while the preventive SCOPF formulation keeps the control variables fixed such that the system remains in a feasible state if any of the contingencies occur. Both of these models are similar from the point of view of the computational complexity, which is the primary focus of this work. Each additional contingency corresponds to an extra set of constraints in the OPF problem specified for the associated power grid. The nominal scenario and all contingency states are coupled, rendering the whole problem computationally intractable for realistic size grids.

3.2.1 Problem formulation

A variety of SCOPF formulations have been proposed in the literature, considering different types of contingencies or security modes (the preventive mode being more prevalent). The SCOPF optimization problem considered in this work is the preventive SCOPF, although the same algorithmic improvements would apply also to the corrective variant. SCOPF problem is formulated as:

$$\underset{\boldsymbol{\theta}_c, \mathbf{v}_c, \mathbf{p}^{\mathrm{G}}{}_c, \mathbf{q}^{\mathrm{G}}{}_c}{\text{minimize}} \sum_{i=1}^{N_{\mathrm{G}}} f_i(\mathbf{p}_0^{G_i}) \tag{3.21a}$$

$$\text{subject to } \forall c \in \{c_0, c_1, \ldots, c_{N_c}\},$$

$$\boldsymbol{c}_{\varepsilon}^{c}(\boldsymbol{\theta}_c, \mathbf{v}_c, \mathbf{p}^{\mathrm{G}}{}_c, \mathbf{q}^{\mathrm{G}}{}_c) = \mathbf{0}, \tag{3.21b}$$

$$\boldsymbol{c}_{I}^{c}(\boldsymbol{\theta}_c, \mathbf{v}_c) \leq \boldsymbol{s}_{\mathrm{L}}^{\max}, \tag{3.21c}$$

$$\mathbf{v}^{\min} \leq \mathbf{v}_c \leq \mathbf{v}^{\max}, \tag{3.21d}$$

$$\boldsymbol{\theta}_c^{slack} = 0, \tag{3.21e}$$

$$\mathbf{p}^{\mathrm{G,min}} \leq \mathbf{p}^{\mathrm{G}}{}_c \leq \mathbf{p}^{\mathrm{G,max}}, \tag{3.21f}$$

$$\mathbf{q}^{\mathrm{G,min}} \leq \mathbf{q}^{\mathrm{G}}{}_c \leq \mathbf{q}^{\mathrm{G,max}}, \tag{3.21g}$$

$$\forall b \in \mathcal{B}_{\mathrm{PV}} : \mathbf{v}_c = \mathbf{v}_{c_0}, \tag{3.21h}$$

$$\forall g \in \mathcal{B}_{\mathrm{PV}} : \mathbf{p}^{\mathrm{G}}{}_c = \mathbf{p}^{\mathrm{G}}{}_{c_0}. \tag{3.21i}$$

Note that the SCOPF problem replicates the OPF constraints and variables for each contingency scenario c. It supplements the standard OPF problem with constraints for the nodal power flow balance (3.21b), the branch flow limits (3.21c), and other operational limits (3.21d), (3.21f), which have to be honored not only for the nominal case c_0, but also for every contingency event $c \in \mathcal{C}$, $N_c = |\mathcal{C}|$. The values of the non-automatic control variables are the same in all system scenarios, as expressed by the two non-anticipatory constraints (3.21h) and (3.21i). These declare that the voltage magnitude and the active power generation at the PV buses $\mathcal{B}_{\mathrm{PV}}$ (also known as generator buses) should remain the same as in the nominal scenario c_0, regardless which contingency they are associated with. The only generator that is allowed to change its active power output is the generator at the slack bus $\mathcal{B}_{\mathrm{slack}}$ (also known as the reference or swing bus) as its active power generation can be modified to refill the power transmission losses occurring in each contingency c. This implies that part of the optimization vector $\boldsymbol{x}$ will be shared between the scenarios and part of it will be local to each contingency. Therefore, the vector of variables can be partitioned into local components $\boldsymbol{x}_c$ for each contingency $\forall c \in \{c_0, c_1, \ldots, c_{N_c}\}$ and the global

(shared) part $\boldsymbol{x}_g$:

$$\boldsymbol{x}_c = \left[\boldsymbol{\theta},\ \mathbf{v}_i,\ \mathbf{q}^{\mathrm{G}},\ \mathbf{p}^{\mathrm{G}}{}_j\right],\ i \notin \mathcal{B}_{\mathrm{PV}}, j \in \mathcal{B}_{\mathrm{slack}}, \tag{3.22}$$

$$\boldsymbol{x}_g = \left[\mathbf{v}_i,\ \mathbf{p}^{\mathrm{G}}{}_j\right],\ i \in \mathcal{B}_{\mathrm{PV}}, j \notin \mathcal{B}_{\mathrm{slack}}, \tag{3.23}$$

$$\boldsymbol{x} = \left[\boldsymbol{x}_0,\ \boldsymbol{x}_1,\ \ldots,\ \boldsymbol{x}_{N_c},\ \boldsymbol{x}_g\right]. \tag{3.24}$$

This ordering of the variables allow implementation of the efficient structure exploiting algorithms which represent the main focus of this work and is discussed in part III of this document.

3.2.2 Credible contingencies selection

An important question arises in the SCOPF problem studies, which is how to select a reduced number of equivalent credible (so called umbrella) contingencies and associated variables that the SCOPF problem has to consider in order to obtain the same or nearly the same solution as with the full set of contingencies. In other words, how to identify contingencies that do not pose any security risks and at the same time do not restrict the optimal solution, therefore do not have to be considered in the SCOPF problem. The set of umbrella contingencies for a given SCOPF is strongly dependent on the operating conditions of the power grid (loading, grid configuration, etc.). Hence, as the parameters of the problem change, the membership in the set of umbrella contingencies also varies. The expertise and detailed knowledge of the power system by the system engineers might be leveraged or an analysis ranking the severity of the events in power systems Rüeger et al. [2019]; Kaplunovich and Turitsyn [2016] might be performed. Alternatively, machine learning techniques are usually used for this purpose, including three classes of learning methods, namely machine learning, artificial neural networks and statistical pattern recognition Du et al. [2019]; R et al. [2013]. However, the resulting set might be still very large and sophisticated optimization algorithms are needed. For the purpose of this study, such contingencies are selected that result in non-empty feasible region of the SCOPF problem (3.21). This restricts the selection of the transmission lines that can experience failure and do not compromise operation of the power grid. Such lines are characterized by the following properties: (i) no islands and isolated buses appear in the grid after the line failure, (ii) the reduced grid remains feasible in the PF sense, and (iii) only limited reactive power generation violations are allowed after the contingency occurrence. These considerations are usually made in the planning stage of the transmission grid design, but since mainly synthetic power grid networks are used to evaluate the solution algorithms, these considerations might not be satisfied for all conceivable contingencies.

3.2.3 Solution approaches

Several techniques have been proposed in the literature aiming to reduce the computational complexity of the SCOPF problem. Table 3.1 summarizes application of various methodologies for solution of the SCOPF problems in the recent years. Techniques such as DC grid approximations Fliscounakis et al. [2013b], combination of AC and DC SCOPF Marano-Marcolini et al. [2012], filtering of contingencies using prior knowledge about the grid, Capitanescu et al. [2007], Capitanescu and Wehenkel [2008], identification of the binding contingencies Platbrood et al. [2014], or nonlinear decomposition of the grid into zones with the aid of additional user information Kargarian et al. [2015]; Mohammadi et al. [2018] are possible approaches. Other works focus on efficient handling of the discrete variables in large-scale grids Macfie et al. [2010], compared to the usual practice of their continuous relaxation or mixed integer programming (MIP). Recently, in addition to classic CPU-based computing, graphics processing units (GPUs) have also been exploited for security analysis in Chen et al. [2017], Zhou et al. [2017]. Other algorithms adopt the dual decomposition or the alternating direction method of multipliers, e.g., applied to the DC OPF problems in Chakrabarti et al. [2014], Kargarian et al. [2018].

A variety of the SCOPF decomposition methods have been proposed. Parallel solution techniques are commonly employed to accelerate the solution of SCOPF problems since the advent of the multicore hardware. Initially, parallel algorithms were introduced for identifying the active constraints in a linear programming (LP) formulation where the simplex method used to solve the $1,663$ bus case with $1,555$ contingencies Rodrigues et al. [1994]. Later, the parallel algorithms were also extended to solve the NLP problems. In the meantime, IP methods had been widely accepted as the most robust and successful tools for large-scale nonconvex optimization, especially due to the fact that both the total number of iterations and the overall execution time grow at a slower rate than in the simplex method Lu and Unum [1993]. Furthermore, IP methods can easily handle problems with many inequality constraints and allow a wide variety of sparse linear solvers, potentially exploiting the particular problem structure Gondzio and Grothey [2009].

A decade later after the parallel simplex method, the nonlinear IP method was used to solve the $3,493$ busbar case with 79 contingencies Qiu et al. [2005]. The problem decomposition on the linear level explored the Schur complement (SC) decomposition, using preconditioned iterative solution methods for computing the local SC contributions; however, good preconditioners are not always available. Alternative parallelization schemes were also explored, such as Benders

Table 3.1. Overview of the SCOPF solution methods with parallelism focusing on compute nodes and cores (nodes×cores).

Year	Method	Buses	Cont.	Parallel
1994	LP Simplex Rodrigues et al. [1994]	1,663	1,555	64×1
2005	NLP IP + SC Qiu et al. [2005]	3,493	79	16×1
2007	NLP IP + Benders Borges and Alves [2007]	2,330	700	12×2
2010	Iterative MIP technique Macfie et al. [2010]	3,551	40	1×1
2012	NLP IP + LP Simplex Marano-Marcolini et al. [2012]	2,746	2,468	1×1
2013	NLP IP Capitanescu and Wehenkel [2013]	8,387	12	1×1
2014	NLP IP + SC Jiang and Xu [2014]	543	556	1×8
2014	NLP IP + cont. filter Platbrood et al. [2014]	9,241	12,000	8×8
2014	NLP IP + SC Chiang et al. [2014]	300	271	160×1
2018	NLP IP + SC Schanen et al. [2018]	118	12,228	1,536×64
2019	NLP IP + SC Kardoš et al. [2020]	9,241	512	64×16

decomposition, which decomposes the SCOPF problem into a master problem, corresponding to normal operation, and multiple subproblems each corresponding to a contingency case. It was applied to DC Corrective SCOPF Mohammadi et al. [2013] or static security control Borges and Alves [2007]. Trivial parallelization of the contingency assessment during the security analysis, based on the combination of a contingency filtering scheme, used to identify the binding contingencies at the optimum, and a network compression method, used to reduce the complexity of the post-contingency model, was explored in Platbrood et al. [2014].

A decade later after the SC decomposition was used, the crucial algorithmic improvement allowing for the efficient linear decomposition was introduced. It adopted the evaluation of local SC contributions by performing an incomplete factorization of the augmented matrices Petra, Schenk, Lubin and Gärtner [2014] used in PIPS-IPM Petra, Schenk and Anitescu [2014] for solution of stochastic LPs and convex QPs. PIPS-NLP, a software library for the solution of large-scale structured nonconvex optimization problems on high-performance computers was introduced in Chiang [2014]. It exploits the structured linear algebra to achieve high computational efficiency to solve the SCOPF problems, as demonstrated on 300 bus case with 271 contingency scenarios Chiang et al. [2014]. The linear decomposition of the nonconvex SCOPF was applied also recently in Schanen et al. [2018] to solve the SCOPF problems with up to $12,228$ (synthetic) contingencies, using thousands of CPU cores, although solving only small scale grids, such as the 118 busbar case.

The solution of the SCOPF problems using large-scale grids was addressed in Kardoš et al. [2020], exploring further elimination of the problem size on the linear level, reducing the memory footprint of the SC algorithm, which is critical for the truly large-scale grids. The algorithm is described in detail in part III.

3.3 Multiperiod problems

Time-coupled formulations, such as storage scheduling Sperstad and Korpas [2019], sizing Park et al. [2015] or storage placement Joubert et al. [2018], are collectively known as multiperiod OPF (MPOPF) problems. The MPOPF problem consists of multiple OPF problems linked together by constraints relating the variables from OPF problems in various time instances. Depending on the application, MPOPF problems have to be solved at different time scales, ranging from long-term planning decisions to real-time intraday operating decisions.

3.3.1 Problem formulation

The MPOPF is formulated as

$$\underset{\boldsymbol{\theta}_n,\ \mathbf{v}_n,\ \mathbf{p}_n^{\mathrm{G}},\ \mathbf{q}_n^{\mathrm{G}}}{\text{minimize}} \sum_{n=1}^{N}\sum_{i=1}^{N_{\mathrm{G}}} f_i(\mathbf{p}_n^{G_i}) \tag{3.25a}$$

$$\text{subject to } \forall n \in \{1,2,\ldots,N\},$$

$$\boldsymbol{c}_{\varepsilon}^{n}(\boldsymbol{\theta}_n, \mathbf{v}_n, \mathbf{p}_n^{\mathrm{G}}, \mathbf{q}_n^{\mathrm{G}}) = \mathbf{0}, \tag{3.25b}$$

$$\boldsymbol{c}_{I}^{n}(\boldsymbol{\theta}_n, \mathbf{v}_n) \leq \boldsymbol{s}_{\mathrm{L}}^{\max}, \tag{3.25c}$$

$$\mathbf{v}^{\min} \leq \mathbf{v}_n \leq \mathbf{v}^{\max}, \tag{3.25d}$$

$$\boldsymbol{\theta}_n^{slack} = 0, \tag{3.25e}$$

$$\mathbf{p}^{\mathrm{G,min}} \leq \mathbf{p}_n^{\mathrm{G}} \leq \mathbf{p}^{\mathrm{G,max}}, \tag{3.25f}$$

$$\mathbf{q}^{\mathrm{G,min}} \leq \mathbf{q}_n^{\mathrm{G}} \leq \mathbf{q}^{\mathrm{G,max}}, \tag{3.25g}$$

$$\boldsymbol{\epsilon}_{\mathrm{S}}^{\min} \leq \boldsymbol{\epsilon}_n \leq \boldsymbol{\epsilon}_{\mathrm{S}}^{\max}. \tag{3.25h}$$

The OPF constraints (3.25b)–(3.25g) must hold in each time period $n \in 1,2,\ldots,N$, while the inter-temporal coupling is introduced by energy storage devices (3.25h) (and possibly generator ramp limits). For convenience, the $\mathbf{p}_n^{\mathrm{G}}$ above consists not only of the conventional generator injections (denoted as $\mathbf{p}^{\mathrm{G}'}, \mathbf{q}^{\mathrm{G}'}$) but also includes the injections incurred by the storage devices $\mathbf{p}^{\mathrm{S}}$. In a practical MPOPF application, consider N_{S} energy storage units, where the vector of the storage power injections consists of discharging and charging injections,

$$\mathbf{p}^{\mathrm{S}} = \left[\mathbf{p}^{\mathrm{Sd},1}, \cdots, \mathbf{p}^{\mathrm{Sd},N_{\mathrm{S}}}, \mathbf{p}^{\mathrm{Sc},1}, \cdots, \mathbf{p}^{\mathrm{Sc},N_{\mathrm{S}}}\right]. \tag{3.26}$$

Similar ordering applies for the vector of reactive powers $\mathbf{q}^{\mathrm{S}}$. The composite vector considering also the generator power injections is thus $\mathbf{p}^{\mathrm{G}} = \left[\mathbf{p}^{\mathrm{G}'}, \mathbf{p}^{\mathrm{S}}\right]$ and $\mathbf{q}^{\mathrm{G}} = \left[\mathbf{q}^{\mathrm{G}'}, \mathbf{q}^{\mathrm{S}}\right]$. At each network bus, the external power injections must equal

the injections from the connected generators, storages and load components, resulting in the constraint (3.25b), which, written in complex power notation, takes form

$$\boldsymbol{c}_{\varepsilon}^{n} := C_{\mathrm{G}}\underline{\mathbf{s}}_{n}^{\mathrm{G}'} + C_{\mathrm{S}}\underline{\mathbf{s}}_{n}^{\mathrm{S}} - \underline{\mathbf{s}}_{n}^{\mathrm{D}} = \underline{\mathbf{s}}_{n}^{\mathrm{B}}. \tag{3.27}$$

The evolution of the vector of storage levels $\boldsymbol{\epsilon}_n \in \mathbb{R}^{N_{\mathrm{S}}}$ follows the update equation

$$\boldsymbol{\epsilon}_n = \boldsymbol{\epsilon}_{n-1} + \mathbf{B}^{\mathrm{S}}\,\mathbf{p}_n^{\mathrm{S}} \quad n = 1, \ldots, N, \tag{3.28}$$

and introduces a coupling between the individual time periods. The energy level in each period needs to honor the storage capacity, as expressed by the constraint (3.25h). The initial storage level is denoted $\boldsymbol{\epsilon}_0$ and the constant matrix $\mathbf{B}^{\mathrm{S}} \in \mathbb{R}^{N_{\mathrm{S}} \times 2N_{\mathrm{S}}}$ models discharging and charging efficiencies of the storage devices

$$\mathbf{B}^{\mathrm{S}} = -\delta t \begin{bmatrix} \eta_{\mathrm{d},1}^{-1} & & & \eta_{\mathrm{c},1} & & \\ & \ddots & & & \ddots & \\ & & \eta_{\mathrm{d},N_{\mathrm{S}}}^{-1} & & & \eta_{\mathrm{c},N_{\mathrm{S}}} \end{bmatrix} \tag{3.29}$$

with the discharging and charging efficiencies $\eta_{\mathrm{d},i}$ and $\eta_{\mathrm{c},i}$, $i = 1, 2, \ldots, N_{\mathrm{S}}$. The linear inequality constraints introduced by storage devices can be written in matrix form involving powers from all storage devices, generators, and time periods as

$$\underbrace{\begin{bmatrix} \boldsymbol{\epsilon}_{\mathrm{S}}^{\min} \\ \boldsymbol{\epsilon}_{\mathrm{S}}^{\min} \\ \vdots \\ \boldsymbol{\epsilon}_0 \end{bmatrix}}_{\boldsymbol{\epsilon}^{\min}} \leq \underbrace{\begin{bmatrix} \boldsymbol{\epsilon}_0 \\ \boldsymbol{\epsilon}_0 \\ \vdots \\ \boldsymbol{\epsilon}_0 \end{bmatrix}}_{\mathbf{E}^0} + \underbrace{\begin{bmatrix} \mathbf{B}^{\mathrm{S}} & & & \\ \mathbf{B}^{\mathrm{S}} & \mathbf{B}^{\mathrm{S}} & & \\ \vdots & \vdots & \ddots & \\ \mathbf{B}^{\mathrm{S}} & \mathbf{B}^{\mathrm{S}} & \cdots & \mathbf{B}^{\mathrm{S}} \end{bmatrix}}_{\mathbf{E}} \underbrace{\begin{bmatrix} \mathbf{p}_1^{\mathrm{S}} \\ \mathbf{p}_2^{\mathrm{S}} \\ \vdots \\ \mathbf{p}_N^{\mathrm{S}} \end{bmatrix}}_{\mathbf{p}^{\mathrm{S}}} \leq \underbrace{\begin{bmatrix} \boldsymbol{\epsilon}_{\mathrm{S}}^{\max} \\ \boldsymbol{\epsilon}_{\mathrm{S}}^{\max} \\ \vdots \\ \boldsymbol{\epsilon}_{\mathrm{S}}^{\max} \end{bmatrix}}_{\boldsymbol{\epsilon}^{\max}}. \tag{3.30}$$

Storage models with known self-discharge rate can also be included through a modification of $\mathbf{E}^0$. Furthermore, a storage degradation cost, modeled as affine or quadratic function of the storage powers $\mathbf{p}_n^{\mathrm{S}}$ and the state of charge $\boldsymbol{\epsilon}_n$, can be incorporated through the problem's objective function.

The variables of the MPOPF problem $\boldsymbol{x}$ can be split into local components $\boldsymbol{x}_n$ according to a time period $n = 1 \ldots N$,

$$\boldsymbol{x}_n = \left[\boldsymbol{\theta}_n,\ \mathbf{v}_n,\ \mathbf{p}_n^{\mathrm{G}},\ \mathbf{q}_n^{\mathrm{G}}\right], \tag{3.31}$$

$$\boldsymbol{x} = \left[\boldsymbol{x}_1,\ \ldots,\ \boldsymbol{x}_N\right]. \tag{3.32}$$

This ordering of the variables allow implementation of the efficient structure exploiting algorithms which represent the main focus of this work.

3.3.2 Distribution system flexibility

This subsection proposes an extension of the MPOPF model providing the flexibility reserves, a pilot project performed in collaboration with ETH Power System Laboratory[1] within the SCCER-FURIES[2] project. The MPOPF problem can be formulated at the distribution level for scheduling the storage devices, where the flexibility reserves which might be used in certain situations, e.g. absorbing the intermittent renewable resources Mohler and Sowder [2014]. Alternatively, in the day-ahead market clearing, the outcomes may not be necessarily feasible for the distribution system operator (DSO) because of network constraints. Hence, the DSO may consider an adjustment market to exploit the potential flexibility of DER, such as energy storage (ES). ES can provide flexibility provision by adjusting its charging and discharging cycles. This may provide a feasible operating solution to the DSO in the day-ahead. Additionally, the DSO may need flexibility to balance their schedule at the transmission-distribution interface and to avoid network violations in real-time. This will become necessary with the increased installation of renewable energy resources.

For this purpose, the ES model needs to be revised such that the ESs are modeled carefully as a switch to consider their charging and discharging efficiencies. In this manner, the flexibility of ES can be fully exploited because it can act both as a generator and a demand. The modeling of the ES needs to consider the state-of-charge (SOC) of ES, since up-flexibility (increased discharging rate of the device if it is in a discharging state, or reduced charging rate if the device is charging) can be only provided if there is enough energy stored and to provide down-flexibility (additional charging, or reduced discharging rate) the SOC needs to be sufficiently depleted. In cases where the ES is partially charged, both up and down-flexibility can be provided up to a certain limit. The model includes two more features. First, it considers the smooth transition between charging and discharging of ES. As ES can swing between generation and demand while providing up or down-flexibility, this becomes important to include the effect of different charging and discharging efficiencies for an ES. Second, it maintains the state-of-charge for the actual time of delivery. This is specifically important for the cases when the SOC is zero or at the maximum capacity. If an ES is scheduled for providing for up or down-flexibility then it must have sufficient SOC to provide the required flexibility. The model can be summarized

[1] `https://psl.ee.ethz.ch`

[2] `https://www.epfl.ch/research/domains/sccer-furies`

as

$$\forall i \in N_S, t \in N: \qquad 0 \le \mathbf{p}_t^{\mathrm{Sd},i} + \boldsymbol{u}_t^{Sd,i} \le \mathbf{p}_d^{\mathrm{S,max}}, \tag{3.33}$$

$$\forall i \in N_S, t \in N: \qquad 0 \le \mathbf{p}_t^{\mathrm{Sd},i} + \boldsymbol{d}_t^{Sd,i} \le \mathbf{p}_d^{\mathrm{S,max}}, \tag{3.34}$$

$$\tag{3.35}$$

$$\forall i \in N_S, t \in N: \qquad \mathbf{p}_c^{\mathrm{S,max}} \le \mathbf{p}_t^{\mathrm{Sc},i} + \boldsymbol{u}_t^{Sc,i} \le 0, \tag{3.36}$$

$$\forall i \in N_S, t \in N: \qquad \mathbf{p}_c^{\mathrm{S,max}} \le \mathbf{p}_t^{\mathrm{Sc},i} + \boldsymbol{d}_t^{Sc,i} \le 0. \tag{3.37}$$

Here, $\mathbf{p}_t^{\mathrm{Sc},i}$ and $\mathbf{p}_t^{\mathrm{Sd},i}$ represent charging and discharging power of the ES with the maximum charging and discharging rate $\mathbf{p}_c^{\mathrm{S,max}}, \mathbf{p}_d^{\mathrm{S,max}}$, respectively. The charging is modeled as negative power injection. The up/down charging/discharging flexibility is modeled by $\boldsymbol{u}_t^{Sd,i}, \boldsymbol{u}_t^{Sc,i}, \boldsymbol{d}_t^{Sd,i}, \boldsymbol{d}_t^{Sc,i}$. The total up and down flexibility required by DSO are ensured in the constraint given below, The minimum up and down flexibility required by the DSO is enforced by requiring

$$\forall t \in N: \qquad \boldsymbol{u}_t^f \le \sum_{i \in N_S} (\boldsymbol{u}_t^{Sc,i} + \boldsymbol{u}_t^{Sd,i}), \tag{3.38}$$

$$\forall t \in N: \qquad \sum_{i \in N_S} (\boldsymbol{d}_t^{Sc,i} + \boldsymbol{d}_t^{Sd,i}) \le \boldsymbol{d}_t^f. \tag{3.39}$$

Furthermore, several operational constraints need to be satisfied by the model, which are imposed by the so-called complementarity constraints, e.g., the ES cannot be simultaneously charging and discharging

$$\forall i \in N_S, t \in N: \qquad \mathbf{p}_t^{\mathrm{Sc},i} \mathbf{p}_t^{\mathrm{Sd},i} = 0, \tag{3.40}$$

and it is enforced to provide either up or down-regulation and not both at the same time interval. This is enforced by

$$\forall i \in N_S, t \in N: \qquad \boldsymbol{u}_t^{Sd,i} \boldsymbol{d}_t^{Sd,i} = 0, \quad \boldsymbol{u}_t^{Sc,i} \boldsymbol{d}_t^{Sc,i} = 0. \tag{3.41}$$

In order to prevent usage of charging flexibility when the storage device is discharging and vice versa, it has to satisfy also the following constraint

$$\forall i \in N_S, t \in N: \qquad (\mathbf{p}_t^{\mathrm{Sd},i} + \boldsymbol{u}_t^{Sd,i} - \boldsymbol{d}_t^{Sd,i})(-\mathbf{p}_t^{\mathrm{Sc},i} + \boldsymbol{u}_t^{Sc,i} - \boldsymbol{d}_t^{Sc,i}) = 0. \tag{3.42}$$

The NLP problem (3.25) can be easily extended by additional 'virtual' generators modeling the flexibility variables, and including the additional constraints (3.33) –(3.42). The equality constraints with zero term at the right-hand side,

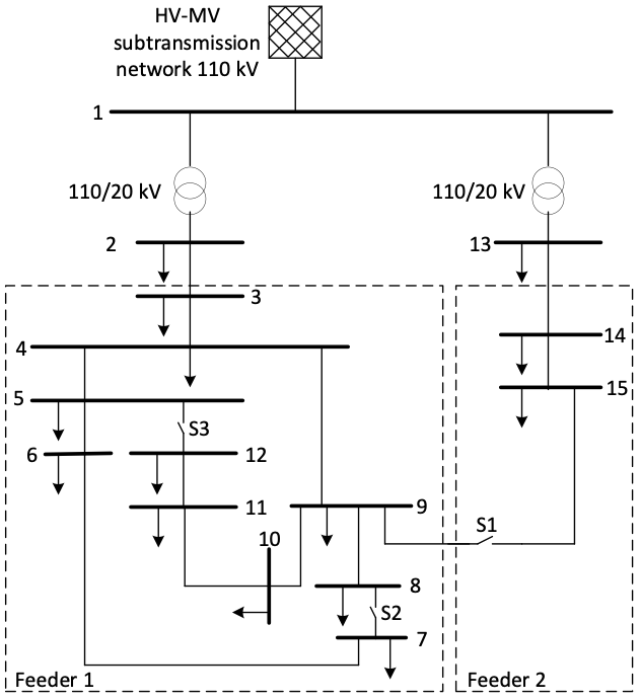


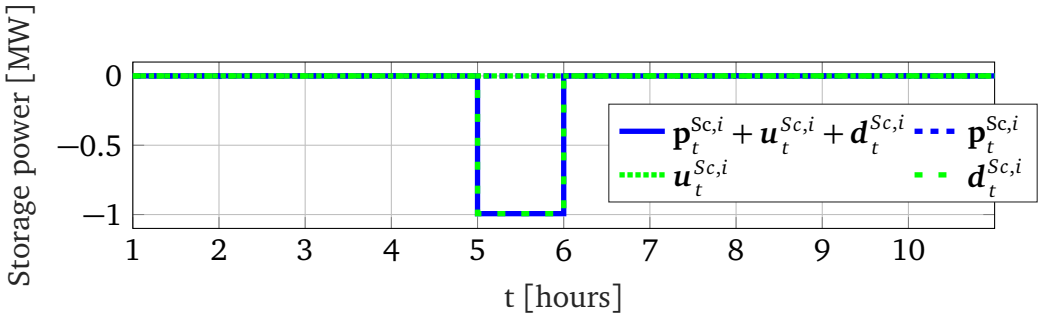
Figure 3.2. CIGRE medium voltage distribution system.

representing the complementarity constraints, might be relaxed to avoid problems with the numerical instability. The problems with the complementarity constraints usually suffer from ill-posedness of the feasible set of the smooth NLP and well-developed nonlinear programming theory and numerical methods are not readily applicable for solving this form of problems Jiang and Ralph [2000]; Fletcher and Leyffer [2004]. The resulting relaxed and smooth continuous NLP can be solved by using any general-purpose optimizer, or as suggested later, using a specialized solver adapted for the particular structure of the problem.

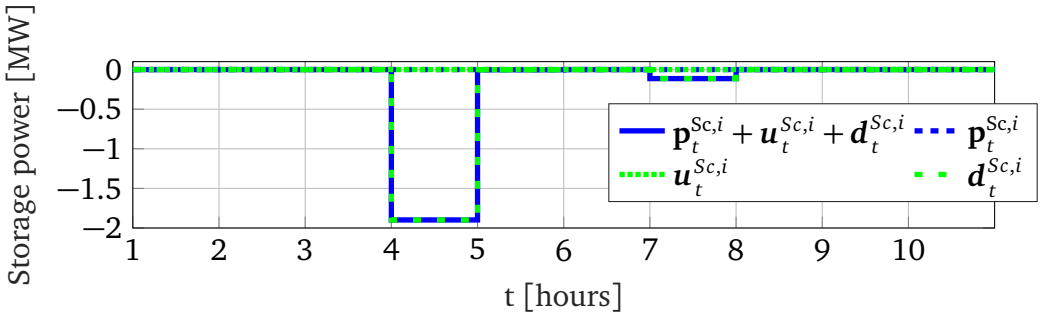
Example

The flexibility of the storage model can be demonstrated on a 15 bus Cigre medium voltage distribution network Rudion et al. [2006], which can have either radial or meshed topology, depending on the combination of the switches, as observed in Figure 3.2. The ES unit is located at the bus 2 and 13. The storage size is chosen to contain up to two hours of the nominal active power demand of the connected bus, that is 39.6 MWh and 40 MWh. The storages are initially at 70% charge level. The storage power ratings are limited to allow a complete dis-

charging and charging within three hours and two hours, respectively. All storage discharging and charging efficiencies are chosen as $\eta_d = 0.97$ and $\eta_c = 0.95$.



(a) Charging flexibility (green) of the storage device 1.



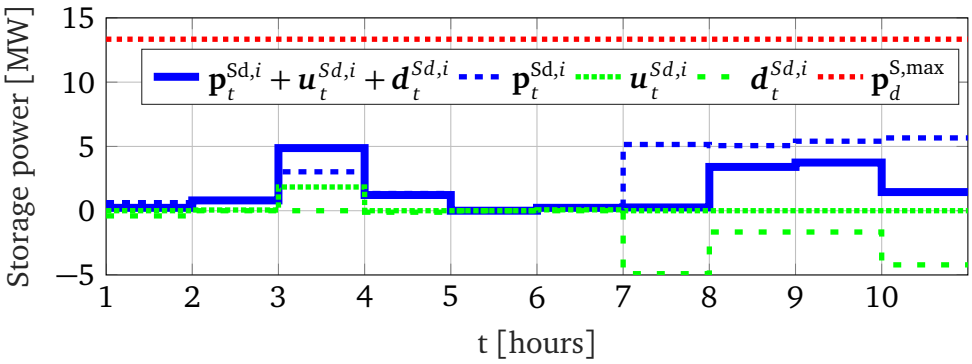
(b) Charging flexibility (green) of the storage device 2.

Figure 3.3. Flexibility provisions by the ES charging ($\mathbf{p}_c^{S,\max} = -20\,\text{MW}$).

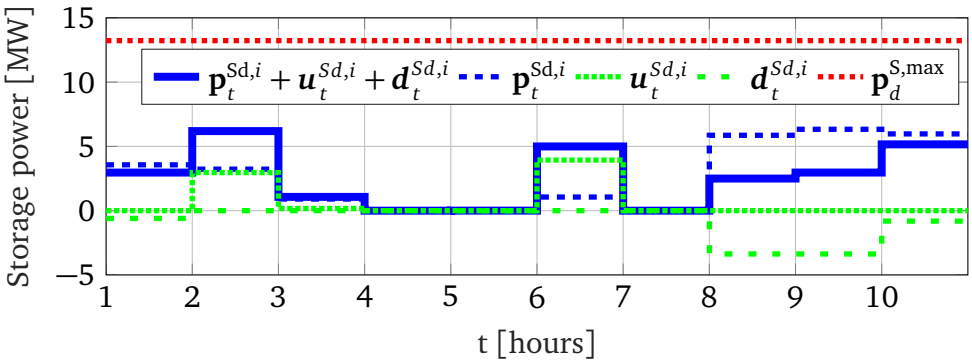
The DSO specifies the grid requirements and parameters, including the required flexibility, summarized in Table 3.2, and runs the proposed optimization problem to minimize the operation cost and identifies the flexibility providers (ES in this case) at each time interval. For simplicity, zero reactive loads are assumed in the illustrative example. The results of flexibility provision from the ES is given in Figures 3.3 and 3.4, while Figure 3.5 demonstrates that the flexibility provision required by the DSO is satisfied.

Two observations are highlighted. First, the ES devices are not actively charging in the current example, as can be observed Figure 3.3, where the charging power $\mathbf{p}_t^{Sc,i}$ is zero across the whole dispatch horizon. However, a portion of the charging down flexibility is provided during 5th and 4th time interval by the ES device 1 and 2, respectively, meaning that the charging rate of the devices can be increased during these time periods. Note that the corresponding discharging power and flexibility, illustrated in Figure 3.4, are zero during these time periods. Second, down flexibility of 2 MW is requested by the DSO in the 4th time period,

as specified in Table 3.2. The ES device 2 provides 1.9 MW of charging down flexibility while 0.1 MW of discharging down flexibility is provided by the ES 1. The contribution of both devices can be observed also in Figure 3.5, where the contribution of both charging and discharging up/down flexibilities are shown.



(a) Discharging flexibility (green) of the storage device 1.

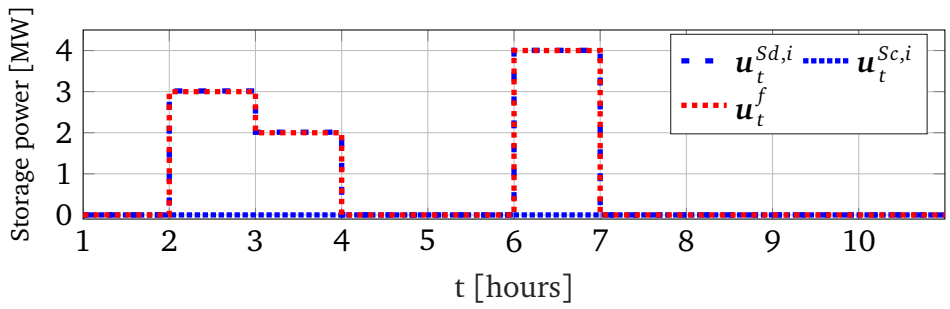


(b) Discharging flexibility (green) of the storage device 2.

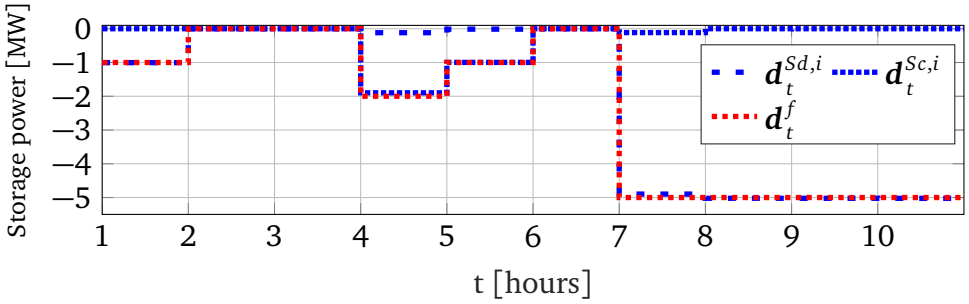
Figure 3.4. Flexibility provisions by the ES discharging.

Table 3.2. DSO flexibility requirements over the 10 hour scheduling horizon.

Time period	1	2	3	4	5	6	7	8	9	10
Up flexibility (MW)	0	3	2	0	0	4	0	0	0	0
Down flexibility (MW)	1	0	0	2	1	0	5	5	5	5



(a) Cumulative Storage up flexibility (all storage devices).



(b) Cumulative Storage down flexibility (all storage devices).

Figure 3.5. Flexibility requirements (red) and actual flexibility provided.

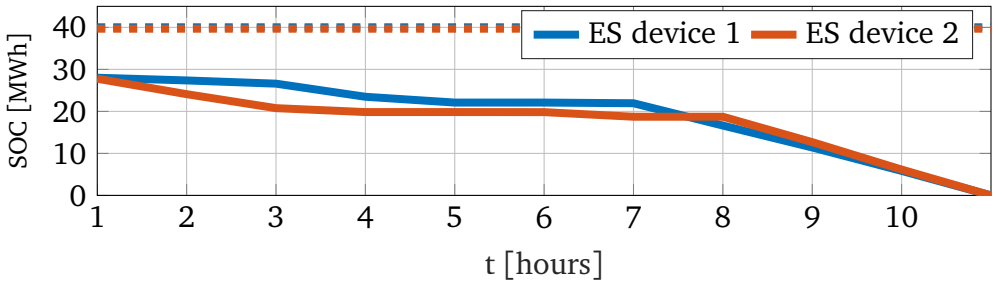


Figure 3.6. Individual ES SOC, trajectory (solid) and maximum (dashed).

3.3.3 Solution approaches

Standard solutions strategies for MPOPF problems (3.25) adopt general purpose NLP methods, usually employed for the solution of nonconvex optimization problems, e.g. time-coupled co-optimization that accounts for energy and reserve ramping across all time intervals Fuchs et al. [2017]. The limitation of such approach is that the problem complexity quickly grows with the increasing length of the time horizon and problem becomes computationally intractable. The com-

putational complexity can be mitigated e.g. by applying successive quadratic programming approach with a second-order cone programming relaxation of the OPF problem Marley et al. [2017]. This combined method was demonstrated on several test cases with up to 4259 nodes although with a time horizon of only eight time steps. A wide-spread solution heuristic to solve the MPOPF problems is to adopt a decomposition strategy based on splitting the long time horizon into smaller intervals Fortenbacher et al. [2018]; Joubert et al. [2018]. The solution thus involves repeatedly solving a constrained optimization problem, using predictions of future costs, disturbances, and constraints over a moving time horizon. Such solutions, however, may become suboptimal compared to the fully coupled solution.

A scaling study of a parallel nonlinear, nonconvex optimization approach applied to a MPOPF problem was presented in Schanen et al. [2018]. Relaxation of the time-binding constraints was used to yield a time-separable optimization problem, necessary for efficient parallelization of the numerical optimization on massively parallel HPC hardware. An algorithm for solving the fully coupled MPOPF problems was introduced in Kourounis et al. [2018]. The solution is based on exploiting the particular structure of the MPOPF problem and the coupling constraints introduced by the energy storage devices. Significant speedups up to several orders of magnitude can be achieved over general purpose NLP methods. The algorithm and the numerical results are elaborated in detail in part III of this document.

Part II

Interior point methods

Chapter 4

Interior point methods

Interior point (IP) methods have become a successful and ubiquitous tool for solving the constrained optimization problems. Various forms of barrier methods were used since the 1960s for programs with nonlinear constraints but gradually, the barrier methods were replaced by more efficient methods such as augmented Lagrangian and sequential quadratic programming (SQP). The IP "revolution", a term coined by M. Wright Wright [2005], can be traced back to 1984 when Karmarkar Karmarkar [1984] announced a polynomial time linear program (LP) that was considerably faster than the most popular simplex method to date. Since then, IP methods have continued to transform both the theory and practice of constrained optimization. IP variants are being extended to solve all kind of programs: from linear to nonlinear and from convex to nonconvex, and they are being applied to solve many practical problems, including scientific and engineering optimization problems.

There are two major directions for solving LP problems. Historically, it was the simplex method but the IP methods have also gained a lot of attention. In case of large problems, the search space might become difficult to explore for the simplex method, which follows a sequence of adjacent extreme points of the feasible region polytope to the optimal solution. However, sophisticated management techniques of the bases, including methods such as column generation Barnhart et al. [1998] or Bender's decomposition Rahmaniani et al. [2017], are very efficient and enable one to solve problems up to several million variables relatively fast. In recent years, however, the performance of the simplex method is being questioned by the rise of IP methods, especially for very large problems (more than several million variables), where the IP method is usually faster. Nevertheless, the simplex method is still very popular nowadays, especially in applications such as integer or stochastic LP. In these applications, a LP is slightly modified

and reoptimizing using the (dual) simplex method can be done very quickly, as it is possible to start with an extreme point that will be close to one of the new extreme optimal points (not necessary inside the new feasible regions). In comparison, the IP method cannot take information from the previous LP resolution to solve a slightly modified LP since its performance suffers in cases where the initial point is not sufficiently inside the interior of the feasible region and is near the boundary of the feasible region.

Amongst many possible approaches to solution of general nonlinear programming (NLP) problems, SQP and IP methods prevail nowadays. The methods are supported by numerous commercial or academic optimization packages, e.g., KNITRO, SNOPT, MATLAB, or ScyPy provide a variant of SQP method, while IPOPT, KNITRO, MIPS, MATLAB, PIPS-IPM or BELTISTOS are packages providing the IP method. Both approaches are based on the idea of moving toward a local solution by approximately solving a sequence of "simpler" problems. SQP methods try to guess which inequality constraints are binding and iteratively refine the active set approximation. Nonbinding constraints can be discarded at the current iteration since they do not restrict the feasible region. The solver then works on the smaller space of the remaining constraints. The SQP iterations proceed by building a quadratic model of the problem and use the solution to update the current iterate and the active set approximation. The identification of the active set is a NP-hard combinatorial problem; thus for problems where the number of inequality constraints is large and identification of the active set becomes difficult, the SQP solution times might become prohibitive. SQP allows efficient warm starting, infeasibility detection, and performs better on problems with a large number of equality constraints or pathological problems.

On the other hand, IP methods move through the interior of the feasible region towards the optimal solution. IP methods are easily applicable to problems with a large number of inequality constraints, which are conveniently handled by logarithmic barrier functions. The main advantages of the IP methods lie in their polynomial time asymptotic complexity and convergence properties. Another advantage of IP methods is that they are applicable to large-scale problems and allow for a variety of different direct sparse or iterative solution methods for the underlying linear systems solved at each iteration. Since different linear system solvers can be plugged in with ease, large-scale structured problems can be solved by exploiting parallel computing infrastructures. Despite the numerous advantages, there exist some intrinsic difficulties related to the design of an appropriate heuristic to decrease the barrier parameter which penalizes the inequality constraints approaching the boundary of the feasible region, selection of the globalization strategy for the nonconvex problems, and inherent

ill-conditioning of the underlying linear systems. It was also shown that the IP algorithms may lead to convergence problems on very hard optimization problems Capitanescu and Wehenkel [2013]. The main cause of convergence problems of the IP methods is that the iterations become stuck at a nonoptimal point if the feasibility boundary is approached too early, i.e., the slack variables prematurely go to zero. This is also a reason why it is, in general, difficult to warm start the IP methods.

The general conceptual IP framework defines the main algorithm, while various IP algorithms differ in details related to the treatment of the optimization problem, such as definition of the (dual) variables, assembly of the linear system, or update strategy of the variables. The IP algorithms can be classified in multiple ways, although a common practice suggests three main categories: projective methods, affine-scaling methods, and primal-dual methods. Among the different IP methods the primal-dual (including primal-dual algorithms that incorporate predictor and corrector) algorithms have gained a reputation for being the most efficient. Another way to categorize the IP methods is related to feasibility of the iterates, hence, we can consider feasible and infeasible IP methods. Yet another classification may be related to the solution of the KKT system during the IP iterations: standard IP algorithms usually use a direct sparse solver, however, inexact IP methods may be devised, using an iterative solver applied to the KKT system. Related to the KKT system are also quasi-Newton IP methods, which use an approximation of the second-order derivatives based on e.g., secant updates and low rank update scheme for the solution of the KKT system.

IP methods are well suited to large-scale optimization since they feature a consistently small number of iterations needed to reach the optimal solution of the problem as well as requiring fairly simple linear algebra. The number of iterations does not increase significantly even for problems with many millions of variables. The linear algebra requirements boil down to factorizations and solves with the augmented system matrix of the problem. In the case of the large-scale problems, the cost of the factorizations may be prohibitive in terms of memory and time, thus limiting the effective use of optimization codes, so a highly optimized linear algebra is paramount to the design of an efficient IP solver. Thus, the effective implementation of IP methods is highly dependent on the availability of effective linear algebra algorithms and software, that are able, in turn, to take into account specific needs of the optimization solvers, which is true especially for large-scale optimization problems.

4.1 Problem definition and optimality conditions

Definition 4.1 *A general NLP problem is formulated as a minimization problem*

$$\underset{\boldsymbol{x}}{\text{minimize}}\ f(\boldsymbol{x}) \tag{4.1a}$$

$$\text{subject to } \boldsymbol{c}_\varepsilon(\boldsymbol{x}) = \boldsymbol{0}, \tag{4.1b}$$

$$\boldsymbol{c}_I(\boldsymbol{x}) \geq \boldsymbol{0}, \tag{4.1c}$$

$$\boldsymbol{x} \geq \boldsymbol{0}, \tag{4.1d}$$

where $\boldsymbol{x} \in \mathbb{R}^{N_x}$, the objective function f is a mapping $f : \mathbb{R}^{N_x} \to \mathbb{R}$, the constraints $\boldsymbol{c}_\varepsilon : \mathbb{R}^{N_x} \to \mathbb{R}^{N_\varepsilon}$ and $\boldsymbol{c}_I : \mathbb{R}^{N_x} \to \mathbb{R}^{N_I}$ are assumed to be sufficiently smooth, with continuous second order derivatives, and $N_x > N_\varepsilon, N_I$, where N_ε, N_I are the numbers of equality and inequality constraints, respectively.

Definition 4.2 *The* feasible set Ω *is a set of points $\boldsymbol{x}$ that satisfy the constraints of the NLP problem* (4.1)*; that is*

$$\Omega = \{\boldsymbol{x} \in \mathbb{R}^{N_x} \mid \boldsymbol{c}_\varepsilon(\boldsymbol{x}) = \boldsymbol{0},\ \boldsymbol{c}_I(\boldsymbol{x}) \geq \boldsymbol{0},\ \boldsymbol{x} \geq \boldsymbol{0}\}. \tag{4.2}$$

Definition 4.3 *The* active set *at any feasible point $\boldsymbol{x}$ is a set of inequality constraints indices, for which the equality constraint holds; that is, $\mathcal{A}(\boldsymbol{x}) = \{i \mid \boldsymbol{c}_I^i(\boldsymbol{x}) = \boldsymbol{0}\}$.*

Definition 4.4 *Given the solution of the NLP problem $\boldsymbol{x}^*$ and the active set $\mathcal{A}(\boldsymbol{x}^*)$, the* linear independence constraint qualification *(LICQ) holds if the set of constraint gradients $\{\nabla \boldsymbol{c}_\varepsilon^i(\boldsymbol{x}^*),\ i = 1 \ldots N_\varepsilon;\quad \nabla \boldsymbol{c}_I^j(\boldsymbol{x}^*),\ j \in \mathcal{A}(\boldsymbol{x}^*)\}$, is linearly independent.*

The NLP problem (4.1) can be transformed into an equivalent problem formulation where the inequality constraints are converted to equality constraints by introducing the slack variables $\boldsymbol{s} \in \mathbb{R}^{N_I}$ with additional nonnegativity bounds $\boldsymbol{s} \geq \boldsymbol{0}$. The NLP problem can be written as

$$\underset{\boldsymbol{x}}{\text{minimize}}\ f(\boldsymbol{x}) \tag{4.3a}$$

$$\text{subject to } \boldsymbol{c}_\varepsilon(\boldsymbol{x}) = \boldsymbol{0}, \tag{4.3b}$$

$$\boldsymbol{c}_I(\boldsymbol{x}) - \boldsymbol{s} = \boldsymbol{0}, \tag{4.3c}$$

$$(\boldsymbol{x}, \boldsymbol{s}) \geq \boldsymbol{0}. \tag{4.3d}$$

Definition 4.5 *The* Lagrangian *for the NLP problem* (4.3) *is defined as*

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda}_\varepsilon, \boldsymbol{\lambda}_I, \boldsymbol{\lambda}_x, \boldsymbol{\lambda}_s) = f(\boldsymbol{x}) + \boldsymbol{\lambda}_\varepsilon^\intercal \boldsymbol{c}_\varepsilon(\boldsymbol{x}) + \boldsymbol{\lambda}_I^\intercal(\boldsymbol{c}_I(\boldsymbol{x}) - \boldsymbol{s}) - \boldsymbol{\lambda}_x^\intercal \boldsymbol{x} - \boldsymbol{\lambda}_s^\intercal \boldsymbol{s}. \tag{4.4}$$

The vectors $\boldsymbol{\lambda}_\varepsilon, \boldsymbol{\lambda}_I, \boldsymbol{\lambda}_x$, and $\boldsymbol{\lambda}_s$ are the Lagrange multipliers associated with the equality, original inequality, and the bound constraints on the primal and slack variables. This allows us to state the Karush–Kush–Tucker (KKT) first-order necessary conditions for the NLP problem (4.3) which characterize the solution.

Theorem 4.1 *Suppose that $\boldsymbol{x}^*$ is a local solution of the NLP problem* (4.3) *and that the LICQ holds at $\boldsymbol{x}^*$. Then there exist Lagrange multiplier vectors $\boldsymbol{\lambda}_\varepsilon^* \in \mathbb{R}^{N_\varepsilon}, \boldsymbol{\lambda}_I^* \in \mathbb{R}^{N_I}, \boldsymbol{\lambda}_x^* \in \mathbb{R}^n$ and $\boldsymbol{\lambda}_s^* \in \mathbb{R}^{N_I}$, $(\boldsymbol{\lambda}_x^*, \boldsymbol{\lambda}_s^*) \geq \mathbf{0}$, such that the following conditions are satisfied at $(\boldsymbol{x}^*, \boldsymbol{s}^*, \boldsymbol{\lambda}_\varepsilon^*, \boldsymbol{\lambda}_I^*, \boldsymbol{\lambda}_x^*, \boldsymbol{\lambda}_s^*)$:*

$$\nabla_x f(\boldsymbol{x}^*) + \nabla_x \boldsymbol{c}_\varepsilon(\boldsymbol{x}^*)^\intercal \boldsymbol{\lambda}_\varepsilon^* + \nabla_x \boldsymbol{c}_I(\boldsymbol{x}^*)^\intercal \boldsymbol{\lambda}_I^* - \boldsymbol{\lambda}_x^* = \mathbf{0}, \tag{4.5a}$$

$$-\boldsymbol{\lambda}_I^* - \boldsymbol{\lambda}_s^* = \mathbf{0}, \tag{4.5b}$$

$$\boldsymbol{c}_\varepsilon(\boldsymbol{x}^*) = \mathbf{0}, \tag{4.5c}$$

$$\boldsymbol{c}_I(\boldsymbol{x}^*) - \boldsymbol{s}^* = \mathbf{0}, \tag{4.5d}$$

$$\boldsymbol{\lambda}_x^* \boldsymbol{x}^* = \mathbf{0}, \tag{4.5e}$$

$$\boldsymbol{\lambda}_s^* \boldsymbol{s}^* = \mathbf{0}, \tag{4.5f}$$

$$(\boldsymbol{x}^*, \boldsymbol{s}^*) \geq \mathbf{0}. \tag{4.5g}$$

The conditions (4.5a) and (4.5b) are referred to as dual feasibility, (4.5c), (4.5d) as primal feasibility, and (4.5f), (4.5e) as complementarity conditions. The point $\boldsymbol{x}^*$ satisfying the KKT conditions is called a stationary, or critical, point. In order to ensure that any stationary point $\boldsymbol{x}^*$ is indeed an optimal (local) solution of the NLP problem (4.3), the second-order sufficient conditions are needed.

Theorem 4.2 *Let $\boldsymbol{x}^*$ be a point at which LICQ holds, the KKT conditions are satisfied, and strict complementarity holds for the active inequality constraints. Then, the point $\boldsymbol{x}^*$ satisfies the second-order sufficient conditions for the NLP problem* (4.3) *if the Hessian of the Lagrangian $\nabla^2_{xx}\mathcal{L}(\boldsymbol{x}^*, \boldsymbol{s}^*, \boldsymbol{\lambda}_\varepsilon^*, \boldsymbol{\lambda}_I^*, \boldsymbol{\lambda}_x^*, \boldsymbol{\lambda}_s^*)$ projected onto the null space of the active constraint Jacobian is positive definite.*

In practice, the second-order conditions are monitored using the inertia of the iteration matrix, which is further elaborated in section 4.4. Proofs of Theorems 4.1 and 4.2 can be found in classic optimization textbooks, e.g., Nocedal and Wright [2006]; Wright [1997]. If the active set at the solution of the NLP problem were known, we could apply a Newton-class method directly to the linearization of the KKT conditions. However, the identification of the active set is known to be an NP-hard combinatorial problem for which, in the worst case, the computation time increases exponentially with the size of the problem. Therefore, many solution strategies adopt an IP approach, introducing a barrier subproblem where the

nonnegativity bounds on the variables and slacks $(\boldsymbol{x}, \boldsymbol{s}) \geq \mathbf{0}$ are handled by the standard logarithmic barrier function. This is, in fact, a penalty term penalizing the iterates that approach the boundary of the feasible region.

Definition 4.6 *The* barrier subproblem *(BSP) reads:*

$$\underset{\boldsymbol{x},\boldsymbol{s}}{\text{minimize}}\ f(\boldsymbol{x}) - \mu \sum_{i=1}^{n} \log(x_i) - \mu \sum_{i=1}^{N_I} \log(s_i) \tag{4.6a}$$

$$\textit{subject to } \boldsymbol{c}_\varepsilon(\boldsymbol{x}) = \mathbf{0}, \tag{4.6b}$$

$$\boldsymbol{c}_I(\boldsymbol{x}) - \boldsymbol{s} = \mathbf{0}. \tag{4.6c}$$

Under certain conditions the solution $\boldsymbol{x}^*$ of the BSP (4.6) converges to the solution of the original NLP problem (4.1) as $\mu_j \downarrow 0$. Consequently, a strategy to solve the original NLP problem is to solve a sequence of the BSPs decreasing the barrier parameter μ_j. The solution of each iterate is not relevant for the solution of the original problem, so it can be relaxed to a certain accuracy and such an approximate solution is used as a starting point for the next BSP. The strategy for updating the μ parameter and thus switching to the next BSP is discussed later in section 4.5.

The solutions of the barrier problem (4.6) are critical points of the Lagrangian function

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda}_\varepsilon, \boldsymbol{\lambda}_I) = & f(\boldsymbol{x}) - \mu_j \sum_{i=1}^{N_x} \log(x_i) - \mu_j \sum_{i=1}^{N_I} \log(s_i) \\ & + \boldsymbol{\lambda}_\varepsilon^\intercal \boldsymbol{c}_\varepsilon(\boldsymbol{x}) + \boldsymbol{\lambda}_I^\intercal (\boldsymbol{c}_I(\boldsymbol{x}) - \boldsymbol{s}).\end{aligned} \tag{4.7}$$

Formulating and solving the optimality conditions of (4.7) directly would lead to singularities, since the derivatives of the barrier terms involve the fractions $\frac{\mu}{x_i}$ and $\frac{\mu}{s_i}$, which are not defined at the solution $\boldsymbol{x}^*, \boldsymbol{s}^*$ of the NLP problem (4.1) when active bounds $x_i^* = 0$ or $s_i^* = 0$ are attained. Primal-dual IP methods Conn et al. [2000]; Gould et al. [2001] define the dual variables $\boldsymbol{z}$ and $\boldsymbol{y}$ as

$$z_i = \frac{\mu}{x_i},\ i = 1, 2, \ldots, N_x, \tag{4.8a}$$

$$y_i = \frac{\mu}{s_i},\ i = 1, 2, \ldots, N_I. \tag{4.8b}$$

From the definition of the dual variables it follows that $z_i = \frac{\mu}{x_i} > 0$; therefore, $z_i x_i = \mu\ \forall i = 1, \ldots, N_x$. Similarly, $y_i s_i = \mu, y_i > 0\ \forall i = 1, \ldots, N_I$. The optimality

conditions of the BSP (4.6), considering also the dual variables (4.8), are written

$$\nabla_x f(\boldsymbol{x}^*) + \nabla_x \boldsymbol{c}_\varepsilon(\boldsymbol{x}^*)^\intercal \boldsymbol{\lambda}_\varepsilon^* + \nabla_x \boldsymbol{c}_I(\boldsymbol{x}^*)^\intercal \boldsymbol{\lambda}_I^* - \boldsymbol{z}^* = \boldsymbol{0}, \tag{4.9a}$$
$$-\boldsymbol{\lambda}_I^* - \boldsymbol{y}^* = \boldsymbol{0}, \tag{4.9b}$$
$$\boldsymbol{c}_\varepsilon(\boldsymbol{x}^*) = \boldsymbol{0}, \tag{4.9c}$$
$$\boldsymbol{c}_I(\boldsymbol{x}^*) - \boldsymbol{s}^* = \boldsymbol{0}, \tag{4.9d}$$
$$\boldsymbol{z}^* \boldsymbol{x}^* = \mu \boldsymbol{e}, \tag{4.9e}$$
$$\boldsymbol{y}^* \boldsymbol{s}^* = \mu \boldsymbol{e}, \tag{4.9f}$$
$$(\boldsymbol{x}^*, \boldsymbol{s}^*) \geq \boldsymbol{0}. \tag{4.9g}$$

Note that the dual variables $\boldsymbol{z}, \boldsymbol{y}$ correspond to the Lagrange multipliers $\boldsymbol{\lambda}_x$ and $\boldsymbol{\lambda}_s$ for the bound constraints. The KKT conditions of the BSP (4.9) are equivalent to the perturbed conditions (4.5) of the original NLP problem (4.3), except for the strict positivity of the dual variables $(\boldsymbol{z}, \boldsymbol{y}) > 0$. The primal-dual equations then become

$$\begin{aligned}
\boldsymbol{l}_a &:= \quad \nabla_x f(\boldsymbol{x}) + \boldsymbol{J}_\varepsilon^\intercal \boldsymbol{\lambda}_\varepsilon + \boldsymbol{J}_I^\intercal \boldsymbol{\lambda}_I - \boldsymbol{z} \;=\; \boldsymbol{0}, && (4.10a)\\
\boldsymbol{l}_b &:= \quad -\boldsymbol{\lambda}_I - \boldsymbol{y} \;=\; \boldsymbol{0}, && (4.10b)\\
\boldsymbol{l}_c &:= \quad \boldsymbol{c}_\varepsilon(\boldsymbol{x}) \;=\; \boldsymbol{0}, && (4.10c)\\
\boldsymbol{l}_d &:= \quad \boldsymbol{c}_I(\boldsymbol{x}) - \boldsymbol{s} \;=\; \boldsymbol{0}, && (4.10d)\\
\boldsymbol{l}_e &:= \quad Z\boldsymbol{x} - \mu \boldsymbol{e} \;=\; \boldsymbol{0}, && (4.10e)\\
\boldsymbol{l}_f &:= \quad Y\boldsymbol{s} - \mu \boldsymbol{e} \;=\; \boldsymbol{0}, && (4.10f)
\end{aligned}$$

where the Jacobian of constraints is written as $\boldsymbol{J}_\varepsilon = \nabla_x \boldsymbol{c}_\varepsilon(\boldsymbol{x})$ and $\boldsymbol{J}_I = \nabla_x \boldsymbol{c}_I(\boldsymbol{x})$. The diagonal matrices X, S, Z, Y are defined as $X = \operatorname{diag}(\boldsymbol{x})$, $S = \operatorname{diag}(\boldsymbol{s})$, $Z = \operatorname{diag}(\boldsymbol{z})$, and $Y = \operatorname{diag}(\boldsymbol{y})$.

Linearizing the primal-dual equations and solving them by applying Newton's method starting from an arbitrary value of the barrier parameter μ may result in slow convergence or poor conditioning of the associated KKT systems. Following the central path ensures that certain favorable conditions for the KKT systems and primal-dual variables are satisfied and descent directions can be obtained with reasonable accuracy.

Definition 4.7 *The* central path *$\mathcal{C}$ is an arc of strictly feasible points of the BSP problem* (4.6)*, $\mathcal{C} = \{(\boldsymbol{x}^\mu, \boldsymbol{s}^\mu, \boldsymbol{\lambda}_\varepsilon^\mu, \boldsymbol{\lambda}_I^\mu, \boldsymbol{z}^\mu, \boldsymbol{y}^\mu) \mid \mu > 0\}$, such that $(\boldsymbol{x}^\mu, \boldsymbol{s}^\mu, \boldsymbol{\lambda}_\varepsilon^\mu, \boldsymbol{\lambda}_I^\mu, \boldsymbol{z}^\mu, \boldsymbol{y}^\mu)$ is a solution of the BSP problem for every value of $\mu > 0$. Points on the central path are characterized by the first-order KKT conditions* (4.10).

Definition 4.8 *The* duality measure τ *is an average pairwise complementarity value* $x_i z_i$ *and* $s_i y_i$,

$$\tau = \frac{\boldsymbol{x}^\intercal \boldsymbol{z} + \boldsymbol{s}^\intercal \boldsymbol{y}}{N_x + N_I}. \tag{4.11}$$

The barrier parameter μ is usually chosen proportionally to the duality measure and the centering parameter $\sigma \in [0,1]$, such that $\mu = \tau\sigma$. By choosing $\sigma = 1$ the algorithm moves toward the central path $\mathcal{C}$. Such a step is biased toward the interior of the feasible region defined by the constraints $(\boldsymbol{z}, \boldsymbol{x}) > \boldsymbol{0}$, $(\boldsymbol{y}, \boldsymbol{s}) > \boldsymbol{0}$. At the other extreme, the value $\sigma = 0$ results in the standard Newton step aiming to satisy the KKT conditions (4.5). Many algorithms use intermediate values of σ from the open interval $(0,1)$ to trade off between the two objectives of reducing duality measure and improving centrality. A strategy for selecting the centering parameter is discussed later in section 4.5.

Remark 1 *The treatment for general box constraints* $\boldsymbol{x}^{min} \leq \boldsymbol{x} \leq \boldsymbol{x}^{max}$ *and general upper and lower bounds on the nonlinear constraints* $\boldsymbol{c}_I^{min} \leq \boldsymbol{s} \leq \boldsymbol{c}_I^{max}$ *requires the addition of modified logarithmic barrier terms*

$$f_\mu(\boldsymbol{x}) = \mu \sum_i \log(x_i - x_i^{min}) + \mu \sum_i \log(x_i^{max} - x_i), \tag{4.12a}$$

$$g_\mu(\boldsymbol{s}) = \mu \sum_i \log(s_i - \boldsymbol{c}_{Ii}^{min}) + \mu \sum_i \log(\boldsymbol{c}_{Ii}^{max} - s_i), \tag{4.12b}$$

The dual variables for $i = 1, 2, \ldots, N_x$ *are defined by*

$$z_i^L = \frac{\mu}{x_i - x_i^{min}}, \qquad z_i^U = \frac{\mu}{x_i^{max} - x_i}, \tag{4.13}$$

while for the constraints the dual variables are defined by

$$y_i^L = \frac{\mu}{s_i - \boldsymbol{c}_{Ii}^{min}}, \qquad y_i^U = \frac{\mu}{\boldsymbol{c}_{Ii}^{max} - s_i}. \tag{4.14}$$

4.2 Search direction computation

Since the solution of the barrier problem (4.6) satisfies the perturbed KKT conditions (4.10), Newton's method may be applied to solve the system of nonlinear equations. The search direction $(\Delta\boldsymbol{x}^k, \Delta\boldsymbol{s}^k, \Delta\boldsymbol{\lambda}_\varepsilon^k, \Delta\boldsymbol{\lambda}_I^k, \Delta\boldsymbol{z}^k, \Delta\boldsymbol{y}^k)$ at the kth iteration can be obtained from the linearization of (4.10) at the current iterate

$(\boldsymbol{x}^k, \boldsymbol{s}^k, \boldsymbol{\lambda}_\varepsilon^k, \boldsymbol{\lambda}_I^k, \boldsymbol{z}^k, \boldsymbol{y}^k)$, resulting in a system of linear equations

$$\begin{bmatrix} \boldsymbol{H} & \mathbf{0} & \boldsymbol{J}_\varepsilon^\intercal & \boldsymbol{J}_I^\intercal & -I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -I & \mathbf{0} & -I \\ \boldsymbol{J}_\varepsilon & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{J}_I & -I & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ Z & \mathbf{0} & \mathbf{0} & \mathbf{0} & X & \mathbf{0} \\ \mathbf{0} & Y & \mathbf{0} & \mathbf{0} & \mathbf{0} & S \end{bmatrix}^k \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{s} \\ \Delta\boldsymbol{\lambda}_\varepsilon \\ \Delta\boldsymbol{\lambda}_I \\ \Delta\boldsymbol{z} \\ \Delta\boldsymbol{y} \end{bmatrix}^k = - \begin{bmatrix} \boldsymbol{l}_a \\ \boldsymbol{l}_b \\ \boldsymbol{l}_c \\ \boldsymbol{l}_d \\ \boldsymbol{l}_e \\ \boldsymbol{l}_f \end{bmatrix}^k, \tag{4.15}$$

where $\boldsymbol{H} = \nabla^2_{xx}\mathcal{L}$. The system (4.15) is clearly unsymmetric. A symmetric system can be obtained after eliminating the last two block rows:

$$\begin{bmatrix} \tilde{\boldsymbol{H}} & \mathbf{0} & \boldsymbol{J}_\varepsilon^\intercal & \boldsymbol{J}_I^\intercal \\ \mathbf{0} & L_s & \mathbf{0} & -I \\ \boldsymbol{J}_\varepsilon & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{J}_I & -I & \mathbf{0} & \mathbf{0} \end{bmatrix}^k \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{s} \\ \Delta\boldsymbol{\lambda}_\varepsilon \\ \Delta\boldsymbol{\lambda}_I \end{bmatrix}^k = - \begin{bmatrix} \boldsymbol{l}_a + X^{-1}\boldsymbol{l}_e \\ \boldsymbol{l}_b + S^{-1}\boldsymbol{l}_f \\ \boldsymbol{l}_c \\ \boldsymbol{l}_d \end{bmatrix}^k, \tag{4.16}$$

where $\tilde{\boldsymbol{H}} = \boldsymbol{H} + X^{-1}Z$ and $L_s = S^{-1}Y$. The directions $\Delta\boldsymbol{z}^k$ and $\Delta\boldsymbol{y}^k$ can be recovered from the equations

$$\Delta\boldsymbol{z}^k = -X^{-1}(\boldsymbol{l}_e + Z\Delta\boldsymbol{x}^k), \tag{4.17}$$

$$\Delta\boldsymbol{y}^k = -S^{-1}(\boldsymbol{l}_f + Y\Delta\boldsymbol{s}^k). \tag{4.18}$$

For a robust algorithm it is crucial to obtain highly accurate search directions. Most of the burden is shifted to the sparse linear solver, where techniques such as fill-in minimization reordering, symmetric scaling vectors, matching, and pivoting can provide substantial improvement to the solution accuracy. Additional improvement can be achieved by performing iterative refinement using the unsymmetrical version KKT linear system of form (4.15). It is possible to further reduce the KKT system by eliminating the slack variables $\boldsymbol{s}$. The system (4.16) can be permuted to the structure with the diagonal block L_s in the lower right corner,

$$\begin{bmatrix} \tilde{\boldsymbol{H}} & \boldsymbol{J}_\varepsilon^\intercal & \boldsymbol{J}_I^\intercal & \mathbf{0} \\ \boldsymbol{J}_\varepsilon & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{J}_I & \mathbf{0} & \mathbf{0} & -I \\ \mathbf{0} & \mathbf{0} & -I & L_s \end{bmatrix}^k \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{\lambda}_\varepsilon \\ \Delta\boldsymbol{\lambda}_I \\ \Delta\boldsymbol{s} \end{bmatrix}^k = - \begin{bmatrix} \boldsymbol{l}_a + X^{-1}\boldsymbol{l}_e \\ \boldsymbol{l}_c \\ \boldsymbol{l}_d \\ \boldsymbol{l}_b + S^{-1}\boldsymbol{l}_f \end{bmatrix}^k. \tag{4.19}$$

Since the block L_s is a diagonal matrix, the reordered system (4.19) can be trivially reduced by computing the Schur complement with respect to the 3×3 block

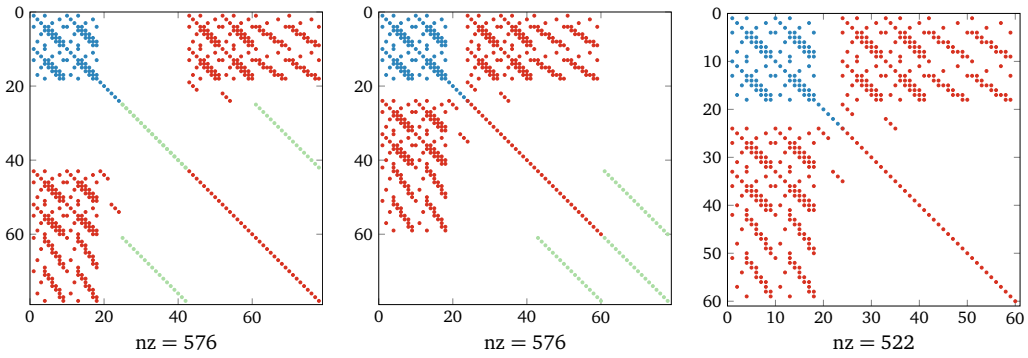


Figure 4.1. Structure of the KKT system (4.16), reordered according to (4.19), and the structure of the reduced KKT with the slacks removed (4.21).

in the upper left corner, as illustrated in Figure 4.1,

$$\begin{bmatrix} \tilde{\boldsymbol{H}} & \boldsymbol{J}_\varepsilon^\intercal & \boldsymbol{J}_I^\intercal \\ \boldsymbol{J}_\varepsilon & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{J}_I & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}^k - \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & -I \end{bmatrix}^\intercal (L_s^k)^{-1} \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & -I \end{bmatrix}. \tag{4.20}$$

The additional elimination, compared to the previous work presented in Petra, Schenk, Lubin and Gärtner [2014]; Petra, Schenk and Anitescu [2014], further reduces the memory requirements and computation time due to the smaller amount of factorization fill-in. Such an elimination, however, can be performed only for the nonzero elements of L_s^k sufficiently away from zero in order to avoid the ill-conditioning of the reduced system. The reduced linear system that needs to be solved now has the structure

$$\begin{bmatrix} \tilde{\boldsymbol{H}} & \boldsymbol{J}_\varepsilon^\intercal & \boldsymbol{J}_I^\intercal \\ \boldsymbol{J}_\varepsilon & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{J}_I & \boldsymbol{0} & -L_s^{-1} \end{bmatrix}^k \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{\lambda}_\varepsilon \\ \Delta \boldsymbol{\lambda}_I \end{bmatrix}^k = - \begin{bmatrix} \boldsymbol{l}_a + X^{-1} \boldsymbol{l}_e \\ \boldsymbol{l}_c \\ \boldsymbol{l}_d + L_s^{-1} (\boldsymbol{l}_b + S^{-1} \boldsymbol{l}_f) \end{bmatrix}^k \tag{4.21}$$

and the eliminated slack variables can be recovered by solving

$$L_s^k \Delta \boldsymbol{s}^k = -\boldsymbol{l}_b^k - S_k^{-1} \boldsymbol{l}_f^k + \Delta \boldsymbol{\lambda}_I^k. \tag{4.22}$$

4.3 Backtracking line-search filter method

After the successful computation of the search direction from (4.16) and (4.17) the step sizes $\alpha_k, \alpha_k^z \in (0, 1]$ need to be determined in order to obtain the next

iterate:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha_k \Delta \boldsymbol{x}^k, \tag{4.23}$$
$$\boldsymbol{s}^{k+1} = \boldsymbol{s}^k + \alpha_k \Delta \boldsymbol{s}^k, \tag{4.24}$$
$$\boldsymbol{\lambda}_\varepsilon^{k+1} = \boldsymbol{\lambda}_\varepsilon^k + \alpha_k \Delta \boldsymbol{\lambda}_\varepsilon^k, \tag{4.25}$$
$$\boldsymbol{\lambda}_I^{k+1} = \boldsymbol{\lambda}_I^k + \alpha_k \Delta \boldsymbol{\lambda}_I^k, \tag{4.26}$$
$$\boldsymbol{z}^{k+1} = \boldsymbol{z}^k + \alpha_k^z \Delta \boldsymbol{z}^k, \tag{4.27}$$
$$\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \alpha_k^z \Delta \boldsymbol{y}^k. \tag{4.28}$$

Different step sizes for the primal and dual variables is commonly employed to prevent unnecessarily small steps in either variables and delay the convergence to the optimal. A first candidate step length is chosen such that the strict positivity of $\boldsymbol{x}, \boldsymbol{s}$, and $\boldsymbol{z}$ is preserved, since it needs to hold both in the solution of the barrier problem (4.6) and also in every iteration, which is necessary in order to evaluate the barrier function. This is accomplished by the fraction-to-the-boundary rule, which identifies the maximum step size $\alpha_k, \alpha_k^z \in (0,1]$, such that

$$\alpha_k^{\max} = \max\left(\alpha \in (0,1] : \boldsymbol{x}^k + \alpha \Delta \boldsymbol{x}^k \geq (1-\tau)\boldsymbol{x}^k\right), \tag{4.29}$$
$$\alpha_k^z = \max\left(\alpha \in (0,1] : \boldsymbol{z}^k + \alpha \Delta \boldsymbol{z}^k \geq (1-\tau)\boldsymbol{z}^k\right), \tag{4.30}$$

where $\tau \in (0,1)$ is a function of the current barrier parameter μ_j. The step size for the dual variables α_k^z is used directly, but in order to ensure global convergence the step size $\alpha_k \in (0, \alpha_k^{\max})$ for the remaining variables is determined by a backtracking line-search procedure, exploring a decreasing sequence of trial step sizes $\alpha_k^i = 2^{-i}\alpha_k^{\max}$ for $i = 0, 1, 2, \ldots$.

The variant of the backtracking line-search filter method Fletcher and Leyffer [2002] used in IPOPT is based on the idea of a biobjective optimization problem with the two goals (i) minimizing the objective function

$$\varphi_{\mu_j}(\boldsymbol{x}, \boldsymbol{s}) := f(\boldsymbol{x}) - \mu_j \sum_{i=1}^{n} \log(x_i) - \mu_j \sum_{i=1}^{N_I} \log(s_i), \tag{4.31}$$

and (ii) minimizing the constraint violation

$$\theta(\boldsymbol{x}, \boldsymbol{s}) := \| \left(\boldsymbol{c}_\varepsilon(\boldsymbol{x}),\ \boldsymbol{c}_I(\boldsymbol{x}) - \boldsymbol{s}\right) \|_1. \tag{4.32}$$

A trial point $\boldsymbol{x}^k(\alpha_k^i) := \boldsymbol{x}^k + \alpha_k^i \Delta \boldsymbol{x}^k$ and $\boldsymbol{s}^k(\alpha_k^i) := \boldsymbol{s}^k + \alpha_k^i \Delta \boldsymbol{s}^k$ during the backtracking line search is considered to be acceptable, if it leads to sufficient progress toward either goal compared to the current iterate. The emphasis is put on the latter goal, until the constraint violations satisfy a certain threshold. Afterwards, the former goal is emphasized and reduction in the barrier function is required, accepting only iterates satisfying the Armijo condition.

Definition 4.9 *The filter $\mathcal{F}$ is a set of ordered pairs containing a constraint violation value θ and the objective function value φ, such that*

$$\mathcal{F} \subseteq \{(\theta, \varphi) \in \mathbb{R}^2 : \theta > 0\}. \tag{4.33}$$

The algorithm also maintains a filter $\mathcal{F}_j$ for each BSP j for which the μ_j is fixed. The filter $\mathcal{F}_j$ contains those combinations that are prohibited for a successful trial point in all iterations within the jth BSP. The filter is initialized so that the algorithm will never allow trial points to be accepted that have a constraint violation larger than $\theta^{\max}$. During the line search, a trial point $\boldsymbol{x}^k(\alpha_k^i),\ \boldsymbol{s}^k(\alpha_k^i)$ is rejected if $(\theta(\boldsymbol{x}_k(\alpha_k^i), \boldsymbol{s}^k(\alpha_k^i)),\ \varphi_{\mu_j}(\boldsymbol{x}^k(\alpha_k^i), \boldsymbol{s}^k(\alpha_k^i))) \in \mathcal{F}_j$. After every iteration, in which the accepted trial step size does not satisfy the two objectives of the backtracking linesearch, the filter is augmented, subject to a procedure described in Wächter and Biegler [2006]. This ensures that the iterates cannot return to the neighborhood of the unsatisfactory iterates. Overall, this procedure ensures that the algorithm cannot cycle, for example, between two points that alternate between decrease of the constraint violation and the barrier objective function.

In cases when it is not possible to identify a satisfactory trial step size, the algorithm reverts to a feasibility restoration phase. Here, the algorithm tries to find a new iterate which is acceptable to the current filter, by reducing the constraint violation with some iterative method. Note that the restoration phase algorithm might not be able to produce a new iterate for the filter line-search method, for example, when the problem is locally infeasible.

4.4 Inertia correction and curvature detection

Definition 4.10 *The inertia of a square matrix is defined as the ordered triplet $(n+, n-, n_0) \in \{\mathbb{N} \cup 0\}^3$, where the terms denote the number of positive, negative, and zero eigenvalues, respectively.*

In order to guarantee descent properties for the line-search procedure, it is necessary to ensure that the Hessian matrix projected on the null space of the constraint Jacobian is positive definite (see Theorem 4.2). Also, if the constraint Jacobian does not have full rank, the iteration matrix in (4.16) is singular, and the solution might not exist. These conditions are satisfied if the iteration matrix has the inertia $(N_x+N_I, N_\varepsilon+N_I, 0)$. The sizes correspond to the size of the Hessian block (with respect to both primal variables $\boldsymbol{x}$ and the slack variables $\boldsymbol{s}$) and the Jacobians of the equality and inequality constraints. If the inertia is not correct,

the iteration matrix needs to be modified. In the IPOPT implementation, the diagonal perturbations $\delta_w, \delta_c \geq 0$ are added to the Hessian (4.16), such that

$$\begin{bmatrix} \tilde{\boldsymbol{H}} + \delta_w I & \mathbf{0} & \boldsymbol{J}_\varepsilon^\intercal & \boldsymbol{J}_I^\intercal \\ \mathbf{0} & L_s + \delta_w I & \mathbf{0} & -I \\ \boldsymbol{J}_\varepsilon & \mathbf{0} & -\delta_c I & \mathbf{0} \\ \boldsymbol{J}_I & -I & \mathbf{0} & -\delta_c I \end{bmatrix}. \tag{4.34}$$

The system is refactorized with different trial values of δ_w, δ_c until the inertia is correct. The inertia of the iteration matrix is readily available from several sparse indefinite linear solvers, such as PARDISO Schenk et al. [2001]. In case the correct inertia cannot be achieved, the current search direction computation is aborted and the algorithm uses a different objective function that does try to solely minimize the feasibility violation (e.g., minimizing the constraints violation), ignoring the original objective function, in the hope that the matrix has better properties close to the feasible points.

The inertia detection strategy focuses on the properties of the augmented iteration matrix (4.16) alone and can thus discard search directions that are of descent but for which the inertia of the augmented matrix is not correct. Furthermore, the inertia detection strategy might require multiple factorizations of the iteration matrix and, because the factorization is the most expensive step in the algorithm, computational performance can be greatly affected. Furthermore, the inertia estimates might vary, depending on which linear solver is used or not be available at all. To bypass the need for the inertia information, several authors suggest using the curvature test, e.g., Chiang et al. [2014] Chiang and Zavala [2016]:

$$\begin{aligned} &\boldsymbol{d}_k^\intercal \boldsymbol{W}_k(\delta)\, \boldsymbol{d}_k \geq \kappa \boldsymbol{d}_k^\intercal \boldsymbol{d}_k, \quad \kappa > 0, \delta \geq 0, \\ &\boldsymbol{W}_k(\delta) = \begin{bmatrix} \tilde{\boldsymbol{H}} & \mathbf{0} \\ \mathbf{0} & L_s \end{bmatrix}^k + \delta I, \quad \boldsymbol{d}_k = \begin{bmatrix} \Delta \boldsymbol{x}_k, & \Delta \boldsymbol{s}_k \end{bmatrix}. \end{aligned} \tag{4.35}$$

If the test is satisfied, the search direction is accepted; if it is not satisfied, the regularization parameter δ is increased and a new search direction is computed using the new regularized matrix.

Remark 2 *While the curvature detection strategy usually requires more IP iterations until convergence compared with the inertia detection, it may require fewer extra factorizations. Overall, the solution time is less than that of the inertia detection because significantly fewer regularizations are needed.*

4.5 Barrier parameter update strategy

The strategy of the barrier parameter update is an important factor influencing the convergence properties, especially for difficult nonconvex problems. When solving nonlinear nonconvex programming problems, it is of great importance to prevent the iteration from failing. Different barrier parameter update strategies are discussed here, including the monotone Fiacco–McCormick strategy Byrd et al. [1998] and an adaptive strategy based on minimization of a quality function Nocedal et al. [2009].

Using the default monotone Fiacco–McCormick strategy, an approximate solution to the barrier problem (4.6) for a fixed value of μ is computed, possibly iterating over multiple primal-dual steps. Subsequently, the barrier parameter is updated and the computation continues by solution of the next barrier problem, starting from the approximate solution of the previous one. The approximate solution for the barrier problem (4.6), for a given value of μ_j, is required to satisfy the tolerance

$$E_\mu(\boldsymbol{x}^{j+1}, \boldsymbol{s}^{j+1}, \boldsymbol{\lambda}_\varepsilon^{j+1}, \boldsymbol{\lambda}_I^{j+1}, \boldsymbol{z}^{j+1}, \boldsymbol{y}^{j+1}) < \kappa_\epsilon \mu_j \tag{4.36}$$

for a constant $\kappa_\epsilon > 0$ before the algorithm continues with the solution of the next barrier problem. The optimality error for the barrier problem is defined by considering the individual parts of the primal-dual equations (4.10), that is, the dual feasibility (optimality), primal feasibility (constraint violations), and the complementarity conditions,

$$E_\mu(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda}_\varepsilon, \boldsymbol{\lambda}_I, \boldsymbol{z}, \boldsymbol{y}) = \max\left(\|\boldsymbol{l}_a\|_\infty, \|\boldsymbol{l}_b\|_\infty, \|\boldsymbol{l}_c\|_\infty, \|\boldsymbol{l}_d\|_\infty, \|\boldsymbol{l}_e\|_\infty, \|\boldsymbol{l}_f\|_\infty\right). \tag{4.37}$$

In the monotone barrier update strategy, the new barrier parameter is obtained from

$$\mu_{j+1} = \max\left(\frac{\epsilon_{\text{tol}}}{10},\ \min\left(\kappa_\mu \mu_j,\ \mu_j^{\theta_\mu}\right)\right) \tag{4.38}$$

with constants $\kappa_\mu \in (0,1)$ and $\theta_\mu \in (1,2)$. In this way, the barrier parameter is eventually decreased at a superlinear rate. On the other hand, the update rule (4.38) does not allow μ to become smaller than necessary given the desired tolerance ϵ_{tol}, thus avoiding numerical difficulties at the end of the optimization procedure. The monotone Fiacco–McCormick strategy can be very sensitive to the choice of the initial point, the initial value of the barrier parameter, and the scaling of the problem. Furthermore, different problems might favor strategies for selecting the barrier parameter at every iteration of an IP method, that is, for every primal-dual step computation. Adaptive strategies commonly choose μ_{k+1}

proportionally to the duality measure for the kth iterate,

$$\mu_{k+1} = \sigma \tau_k, \tag{4.39}$$

where $\sigma > 0$ is a centering parameter and τ denotes the duality measure (4.11). The adaptive strategies vary in how the centering parameter is determined. Two adaptive strategies implemented in IPOPT are discussed next.

Mehrotra's proposed a predictor-corrector principle Mehrotra [1992] for computing the search direction. The centering parameter is computed as the ratio between the duality measure (4.11) in the current iterate and the iterate updated by the predictor step, considering the longest possible step sizes that retain the nonnegativity of the variables in the barrier problem. If good progress in the duality measure is made in the predictor step, the centering parameter obtained in this way is small, $\sigma < 1$; therefore, the μ will be small in the next iteration. In other cases σ may be chosen to be greater than 1. This heuristic is based on experimentation with linear programming problems, and has proved to be effective for convex quadratic programming.

The adaptive barrier update strategy based on the quality function, as suggested in Nocedal et al. [2009], is trying to determine the centering parameter by minimizing a linear approximation of the quality function. The quality function is a measure defined by the infeasibility norms in the current iterate updated by the probing search direction, which is expressed as a function of the sought parameter σ. The minimization problem is solved by a golden bisection procedure on the specified $(\sigma_{\min}, \sigma_{\max})$ interval with a maximum of 12 bisections. The evaluation of the barrier update strategies on both linear and nonlinear problems revealed superior performance of the adaptive methods over the monotone strategy, both in terms of CPU time and number of IP iterations. Although this superior performance is more pronounced for the linear benchmarks, significant improvements can be expected by using adaptive strategies, particularly in applications where the function evaluation has the dominant cost Nocedal et al. [2009]. Figure 4.2 shows the convergence behavior with different barrier parameter update strategies. The value of the barrier parameter μ over the iterations of the IP is shown for the two update strategies. Feasibility, optimality and the objective function are shown as well. The monotone strategy uses $\mu_0 = 100$. The adaptive strategy used was the latter of the two, based on the quality function minimization.

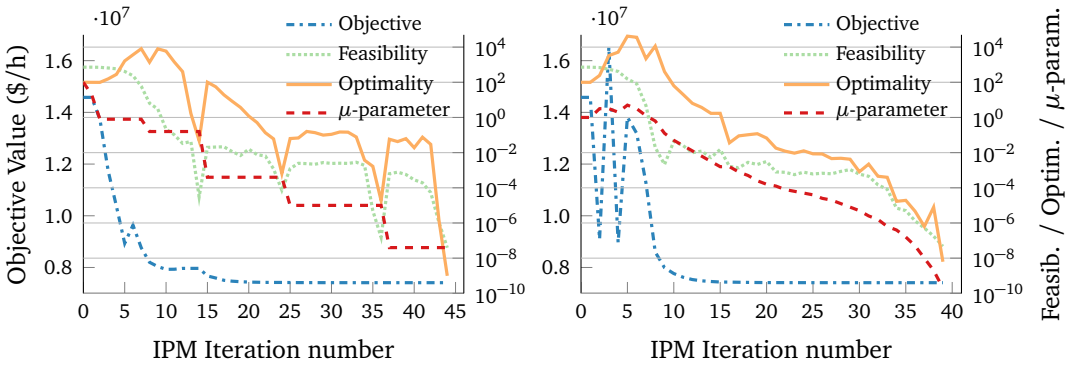


Figure 4.2. Barrier parameter update strategies (left: monotone; right: adaptive).

4.6 Problem scaling and convergence criteria

Optimal control of realistic industrial and engineering problems, such as modern power networks, multienergy carrier systems, the variables and constraints encountered, commonly involve different scales that usually differ by several orders of magnitude. Sophisticated scaling is necessary to remedy problems related to establishing accurate stopping criteria, improving convergence deteriorated by unbalanced direction vectors, and dealing with loss of accuracy of the descent direction computation due to poor conditioning of the associated KKT systems. In the ideal case, not only the variables but also the functions should be scaled so that changing a variable by a given amount has a comparable effect on any function which depends on these variables. In other words, so that the nonzero elements of the function gradients are of the same order of magnitude. For this purpose, gradient-based scaling is commonly employed so that at the starting point the gradients are scaled close to one. The scaling factors for the gradients are defined as

$$s_f = \min(1,\ g_{\max}/\|\nabla_x f(\boldsymbol{x}_0)\|_\infty), \tag{4.40}$$

$$s_g^{(j)} = \min(1,\ g_{\max}/\|\nabla_x \boldsymbol{c}_\varepsilon^{(j)}(\boldsymbol{x}_0)\|_\infty),\ j = 1\ldots N_\varepsilon, \tag{4.41}$$

$$s_h^{(j)} = \min(1,\ g_{\max}/\|\nabla_x \boldsymbol{c}_I^{(j)}(\boldsymbol{x}_0)\|_\infty),\ j = 1\ldots N_I, \tag{4.42}$$

for a given $g_{\max} > 0$. If the maximum gradient is above this value, then gradient-based scaling will be performed. Note that all gradient components in the scaled problem are at most of size $g_{\max}$ at the starting point. The scaling factors are computed only at the beginning of the optimization using the starting point and

kept constant throughout the whole optimization process.

Even if the original problem is well scaled, the multipliers $\boldsymbol{\lambda}_\varepsilon, \boldsymbol{\lambda}_I, \boldsymbol{z}$ might become very large, for example, when the gradients of the active constraints are (nearly) linearly dependent at a solution of (4.1). In this case, the algorithm might encounter numerical difficulties satisfying the unscaled primal-dual equations (4.16) to a tight tolerance. The convergence criteria in (4.37), therefore, need to be scaled accordingly. The scaled optimality error used to determine the convergence criteria is defined as

$$E_0(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda}_\varepsilon, \boldsymbol{\lambda}_I, \boldsymbol{z}) = \max\left(\frac{\|\boldsymbol{l}_a\|_\infty}{s_1}, \frac{\|\boldsymbol{l}_b\|_\infty}{s_1}, \|\boldsymbol{l}_c\|_\infty, \|\boldsymbol{l}_d\|_\infty, \frac{\|\boldsymbol{l}_e\|_\infty}{s_2}, \frac{\|\boldsymbol{l}_f\|_\infty}{s_2} \right), \tag{4.43}$$

where the scaling factors s_1, s_2 are defined as

$$s_1 = \frac{\max\left(s_{\max}, \frac{\|\boldsymbol{\lambda}_\varepsilon\|_1 + \|\boldsymbol{\lambda}_I\|_1 + \|\boldsymbol{z}\|_1 + \|\boldsymbol{y}\|_1}{N_\varepsilon + N_I + N_x + N_I}\right)}{s_{\max}}, \quad s_2 = \frac{\max\left(s_{\max}, \frac{\|\boldsymbol{z}\|_1 + \|\boldsymbol{y}\|_1}{N_x + N_I}\right)}{s_{\max}}. \tag{4.44}$$

The overall IPOPT algorithm terminates successfully, if the NLP error for the current iterate with $\mu = 0$ in (4.43),

$$E_0(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda}_\varepsilon, \boldsymbol{\lambda}_I, \boldsymbol{z}, \boldsymbol{y}) \leq \epsilon_{\text{tol}}, \tag{4.45}$$

becomes smaller than the user provided value $\epsilon_{\text{tol}} > 0$, and if the individual criteria according to dual, primal, and complementarity conditions in (4.43) are met. Each criterion uses a separate, user provided tolerance value.

4.7 Initial point selection and warm-start strategies

The simplex method moves from vertex to vertex of the polytope encompassing the feasible set. In the typical case, a reasonably small change in the objective will result in a new optimal solution that is only a few simplex pivots away. An optimal basis of an original problem usually serves as an excellent warm-start to resolve another closely related problem.

Unlike the simplex method, IPMs generate a sequence of interior-points that converge to an optimal solution in the limit. Since IPMs work with interior-points, they tend to generate much better search directions at points that are away from the boundary of the feasible region and close to the central path. When the new optimization problem is being solved, using an optimal solution of a near problem, it might take several iterations just to get back to the central

path. Therefore, an optimal or a near-optimal solution of the original problem is in general not a very good candidate to be used as a warm-start for the solution of a nearby problem. Approaches proposed in the existing literature on warm-start strategies and reoptimization in the context of IPMs relies on the set of starting points with desirable properties such as near-feasibility, near-optimality, and/or proximity to the central path John and Yıldırım [2008].

Typically, after perturbing the original problem, the previous solution fails to satisfy primal feasibility or dual feasibility or both. The existing methods propose different ways to handle this situation. Some strategies rely on computing an adjustment or several adjustments to the previously stored iterate to regain feasibility for the perturbed problem, which might reduce the number of iterations, but introduces additional computational cost required to compute the adjustments for the initial guess. Other methods modify the perturbed problem using judiciously chosen penalty or barrier parameters so that the stored iterate can be used as is. This however requires user experience in order to properly determine the appropriate barrier parameter values.

Chapter 5

KKT solution methods

The solution of smooth optimal control problems by IP methods using exact second-derivative information, entails the solution of a sparse indefinite linear system at each iteration. This linear system is commonly known as the KKT system, since it is derived from the linearizion of the optimality conditions (also known as the KKT conditions). This linear system is referred to as the KKT system in the following text, and KKT matrix refers to the associated sparse matrix. For large-scale optimization problems the KKT system is large and sparse, while its solution, obtained via the aid of direct sparse linear solvers, dominates the overall runtime performance of the optimization. For this reason, optimization and numerical linear algebra are strongly interlinked domains of scientific computing. Much progress in numerical linear algebra has been motivated by the need for solving linear systems with special features in the context of optimization, and many optimization codes have benefited, in terms of both efficiency and robustness, from advances in numerical linear algebra. The mutual feedback mechanism between the fields of linear algebra and optimization has been pointed out in multiple publications, e.g., D'Apuzzo et al. [2010]; Benzi et al. [2005]; Kourounis et al. [2018].

The importance of the KKT system solver in context of the IP methods is crucial for two main reasons. First, the solution of the KKT linear system provides the search direction, therefore, inaccurate solutions deteriorate the convergence enforcing very small steps especially if a line search method is used, and may also lead to divergence especially in the case of ill-conditioned KKT systems. Second, the computational complexity of the IP method depends heavily on the efficiency of the linear solver, since the main computational task at each iteration of the IP method is the computation of the search direction.

5.1 Basic properties of the KKT matrix

The KKT structure of the primal-dual IP was discussed in section 4.2. For a general NLP problem, the KKT system (4.15) is unsymmetric, sparse, and indefinite. As a common practice the linear system is usually reduced to a symmetric form resulting in more efficient factorization and backsubstitution as well as memory consumption by direct sparse solvers. The majority of the IP frameworks use the augmented system (4.16), thus we limit our discussion to the systems with the following structure:

$$\underbrace{\begin{pmatrix} H & J^T \\ J & 0 \end{pmatrix}}_{K} \begin{pmatrix} v \\ w \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix}, \tag{5.1}$$

where $H = \begin{pmatrix} \tilde{\boldsymbol{H}} & \mathbf{0} \\ \mathbf{0} & L_s \end{pmatrix}$, $J = \begin{pmatrix} \boldsymbol{J}_\varepsilon & \mathbf{0} \\ \boldsymbol{J}_I & -I \end{pmatrix}$, $v = \begin{pmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{s} \end{pmatrix}$, $w = \begin{pmatrix} \Delta \boldsymbol{\lambda}_\varepsilon \\ \Delta \boldsymbol{\lambda}_I \end{pmatrix}$, $c = -\begin{pmatrix} \boldsymbol{l}_a + X^{-1}\boldsymbol{l}_e \\ \boldsymbol{l}_b + S^{-1}\boldsymbol{l}_f \end{pmatrix}$, $d = -\begin{pmatrix} \boldsymbol{l}_c \\ \boldsymbol{l}_d \end{pmatrix}$, and $n = N_x + N_s$, $m = N_\varepsilon + N_I$, using the notation from section 4.2. The following theorem provides sufficient conditions for the nonsingularity of the KKT matrix K and hence the uniqueness of the solution.

Theorem 5.1 *Assume J has full rank and $Z \in \mathbb{R}^{n\times(n-m)}$ is a basis for $ker(J)$. If $Z^T H Z$ is positive definite, then K is nonsingular.*

The positive definiteness means that the quadratic problem related to the KKT matrix is strictly convex and hence the KKT system provides its solution. Proof for the theorem can be found in [Nocedal and Wright, 2006, Ch. 16]. In practical applications, computing a basis is a very expensive operation, thus the IP framework controls the inertia instead. The inertia of the KKT matrix reveals if the problem is locally convex at the current iterate. If we assume constrained optimization problems, where $m \geq 1$, the KKT matrix K is always indefinite, as suggested by the following theorem.

Theorem 5.2 *Let K be defined by* (5.1)*, and suppose that J has rank m. Then $inertia(K) = inertia(Z^T H Z) + (m, m, 0)$. Therefore, if $Z^T H Z$ is positive definite, $inertia(K) = (n, m, 0)$.*

The proof is presented in Forsgren et al. [2002]. Since $Z^T H Z$ are the reduced Hessian matrices of suitable local quadratic models of the optimization problem, the inertia of K reveals whether the problem is locally strictly convex at the current iterate or not. In this context, the inertia from Theorem 5.2 is called the

correct inertia. In the convex case, the KKT matrix always has the correct inertia, and in this situation an off-the shelf sparse factorization routine can be used, where the pivot selection is based on sparsity and numerical stability. In the nonconvex case, it is of interest to find out the inertia during the factorization process so that the matrix may be modified a posteriori. As mentioned before, this relation between convexity and inertia has motivated the interest for suitable modifications of both direct and iterative solvers, in order to detect if the KKT matrix has the correct inertia during the solution process.

5.2 Direct methods

Direct methods are widely used for solving the KKT systems in well-established optimization codes based on IP methods, capable of solving convex quadratic programing problems, such as LOQO Vanderbei [1999], OOQP Gertz and Wright [2003], or general NLP problems, IPOPT Wächter and Biegler [2005, 2006], KNITRO Byrd et al. [2006], OOPS Gondzio and Grothey [2009], PIPS-NLP Chiang [2014]. In this case, the KKT system is factorized into $K = LDL^T$ factorizations, where L is unit lower triangular and D is a (block) diagonal matrix. In the case of symmetric indefinite systems, factorization consists of two phases, a symbolic and a numeric one. In the symbolic phase, an initial fill-reducing ordering is computed based on the structure of K only. Suitable reordering strategies are exploited to deal with the fill-in problem, e.g., METIS Gupta et al. [1997] based on the multilevel recursive-bisection or multilevel k-way partitioning. As a consequence, the linear solver performs factorization of the permuted KKT matrix $PKP^T = LDL^T$, where P is a product of permutation matrices that holds the pivot order and is chosen to try to preserve sparsity and limit growth in the size of the factor entries. The pivoting strategies such as Bunch and Kaufman Bunch and Kaufman [1977], are generally used during the factorization to ensure the numerically stable factorization. In this case, LDL^T factorizations, where L is unit lower triangular and D is symmetric block diagonal with 1×1 or 2×2 blocks, are applied to the KKT system.

In the case of the IP methods, the condition number of the system may increase dramatically as optimality is approached. When using direct methods, the inherent ill-conditioning of the KKT matrix is not a severe problem and, under pretty general assumptions, these methods are able to compute a search direction accurately enough to advance toward the optimal solution Wright [1998]. However, in some instances it can be beneficial to scale the linear system before it is solved, thus also enabling the linear solver to more accurately report the iner-

tia of the matrix. For larger problems within an open source testing environment for optimization and linear algebra solvers, CUTEr set, it was observed that the use of scaling can offer worthwhile savings but this is highly problem dependent Hogg and Scott [2013]. Finally, for some factorizations, particularly those where a small pivot threshold was used, it may be necessary to use a refinement process to improve the quality of the computed solution. For example, the iterative refinement process used in PARDISO. Some codes (e.g. KNITRO) do, however, use preconditioned conjugate gradient as an alternative to iterative refinement for improving the accuracy, when the direct approach fails to produce a solution of sufficient accuracy Byrd et al. [1999].

When the problem is large-scale, the cost of the factorizations may be prohibitive in terms of memory and time, thus limiting the effective use of optimization codes. This has motivated in the last years an increasing research activity devoted to the development of suitable high-performance solution approaches which are discussed in Part III.

5.2.1 Selective elimination of the slack variables

In section 4.2 it was discussed how the KKT system can be reduced to a symmetric structure and how its size can be further reduced by elimination of the diagonal terms L_s, resulting in system (4.21). However, since in the neighborhood of the optimal solution some of the diagonal terms in L_s approach zero, the associated slacks, whose coefficients in L_s are close to machine epsilon, are not eliminated. This prevents the excessive ill-conditioning of the reduced system. The direct sparse solver PARDISO treats indefinite systems by a symmetric maximum weighted matching algorithm and 2×2 pivoting Schenk and Gärtner [2006a], which permutes and scales the system such that either the diagonal entry is 1 or the corresponding nearest off-diagonal element is 1. This method is particularly efficient for the highly indefinite matrices stemming from IP methods Gould et al. [2007].

The numerical values in L_s differ significantly for the active and inactive constraints. First, consider the active constraints (4.1c). The associated slack variables are approaching zero and the Lagrange multipliers attain large values, therefore the corresponding L_s coefficients will become large as well. These values can be eliminated without increasing of the condition number of the reduced system. On the other hand, the slack variables of the inactive inequality constraints are in the order of up to hundreds of thousands, depending on the power flow rating of the transmission lines, and the corresponding Lagrange multipliers $\boldsymbol{\lambda}_I$ approach zero. The corresponding coefficients L_s will thus also

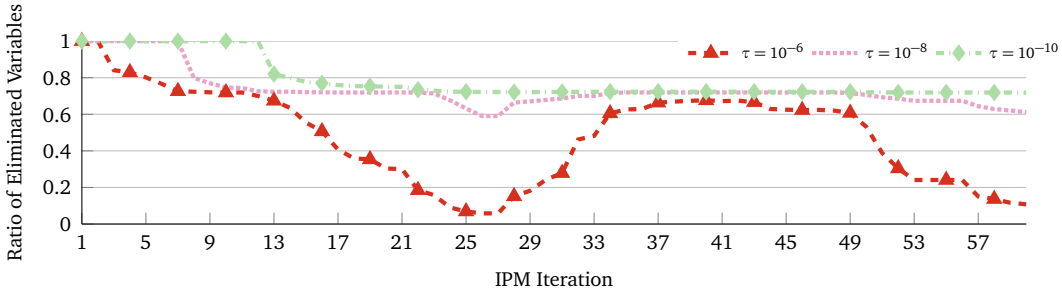


Figure 5.1. Number of eliminated variables relative to the overall number of the slack variables. The elimination is performed only if the corresponding L_s term is above the threshold τ.

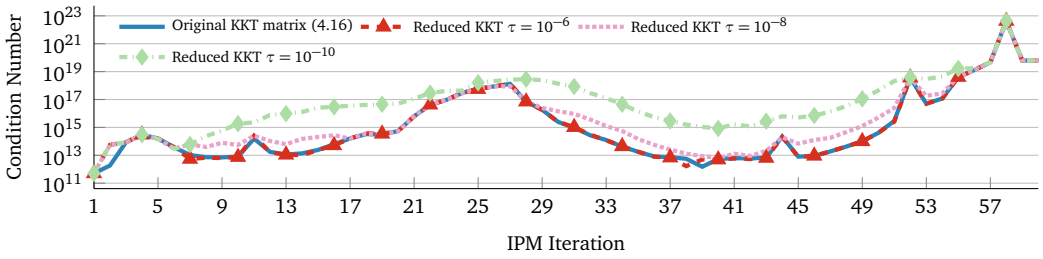


Figure 5.2. Condition number of the KKT system and its reduced versions with various elimination thresholds τ.

approach zero in the neighborhood of the optimal solution. Elimination of these variables would introduce large numerical error, since the elimination of the variables involves inversion. Figure 5.1 demonstrates the ratio of variables that can be eliminated throughout the IPM iterations with various elimination thresholds τ, such that $L_s > \tau$, and Figure 5.2 illustrates the condition number estimates (MATLAB's `condest`) of the reduced and the original KKT system (4.16). The results are demonstrated for the PEGASE1354 network with 20 arbitrarily chosen contingencies; other networks were observed to behave similarly. The appropriate threshold that does not increase the ill-conditioning, while still allowing a substantial number of variables to be eliminated, was chosen to be $\tau = 10^{-8}$.

5.3 Iterative methods

Sparse direct methods are the solvers of choice in various optimization codes for solution of the OPF problems. However, the optimization framework might resort to the iterative method when the direct approach fails to produce a solution

of sufficient accuracy, such as in the KNITRO case. Similarly, the IP algorithm in FMINCON applies a projected conjugate gradient method to solve the KKT system in an iterative fashion. It was observed that FMINCON is not competitive with other optimization frameworks relying on direct sparse solvers during solution of large-scale OPF problems Kardos et al. [2018]. On the other hand, iterative methods are popular in the numerical solution of partial differential equation (PDE) problems because of their intrinsic storage and computational requirements. The matrix is not formed explicitly, only an operator representing the matrix-vector product is provided. For example, in the case of saddle point systems arising from PDE problems on three dimensional meshes Benzi et al. [2005].

Iterative solvers face many difficulties when solving the KKT systems resulting from the IP methods. A possible reason is that, besides its poor conditioning, the matrix lacks the regular spectral properties of matrices obtained from discretizations of continuous operators. In the context of the IP methods, the iterative solution of the KKT system may be interpreted as an inexact search direction calculation. It was demonstrated that computational flexibility is greatly increased as inexact search direction calculations are allowed on certain problems (e.g. PDE-constrained optimization problems Curtis et al. [2012]). The termination criteria for the iterative process need to be carefully chosen in order to account also for the noncovex problems, which cannot rely purely on the residuals, as is the case for the convex problems. Additional conditions and procedures that aid the algorithms in converging toward minimizers need to be introduced in order to avoid saddle points or local maximizers, such as sufficient merit function approximation reduction termination tests introduced in Curtis, Nocedal and Wächter [2010]; Curtis, Schenk and Wächter [2010].

When an iterative solver is used to solve the KKT linear systems, an effective preconditioner is essential to keep the number of iterations low. For best performance, the preconditioner should be tailored to specific problems. Otherwise, there are two general-purpose preconditioning approaches. One is based on purely algebraic techniques (incomplete factorizations or sparse approximate inverses), and the other on algebraic multilevel methods. These require little knowledge of the problem and can be applied in a black-box fashion. On the other hand, when applied to saddle point systems, the black-box approach often performs poorly because of the indefiniteness and lack of diagonal dominance.

5.4 Quasi-Newton methods

The previous two sections were based on the fact that the second-order information (i.e. Hessians of the Lagrangian and constraints) is available. In many applications, such information is very expensive to compute or the storage required is too large Kourounis et al. [2014a]. Alternatively, the computational setup might require to avoid using the Hessian information, as explained below. This section briefly discusses OPF solution approaches based on quasi-Newton methods Gondzio and Sobral [2019], where the second-order information is based on the limited memory BFGS approximation Nocedal and Wright [2006]; Lewis and Overton [2013].

Augmented Lagrangian

An example of the OPF solution approach, where it could be favorable to avoid using the second-order information, would be formulating the problem in such a way that its structure is favorable for modern computing architectures. Modern HPC architectures are adopting hardware accelerators at an increasing pace due to their favorable energy efficiency and high floating point operation rate. This will be the case also for the first exascale computer "Aurora" that should get in production in 2021 Trader [2019]. In order to gain a performance benefit in an application code, the existing algorithms have to be redesigned to implement computational patterns favorable for these accelerated architectures. Current state-of-the-art optimization packages rely on sparse linear algebra kernels, which require indirect memory access and thus are not suitable for the accelerators. The quasi-Newton augmented Lagrangian method [Nocedal and Wright, 2006, Ch. 17] can be used as an alternative solution method to IP method which requires solution of the KKT sparse linear systems, including sparse Hessian of the Lagrangian and sparse Jacobian of the constraints. On the other hand, the augmented Lagrangian formulates an unconstrained optimization problem (except simple box bounds), thus eliminating the sparse Jacobians. Furthermore, if it is used in conjunction with a quasi-Newton method based on secant updates (e.g., BFGS), which forms a dense Hessian approximation, the computational kernels will have dense nature. However, explicit assembly of this matrix might quickly exhaust available memory for large-scale problems. Limited memory BFGS is based on a compact representation using the information in change of the gradient and iterates in a couple of previous iterations. Low rank approximations are matrix free and use only vector multiplications and additions. Preliminary experiments of such an approach demonstrated that the solver is experiencing con-

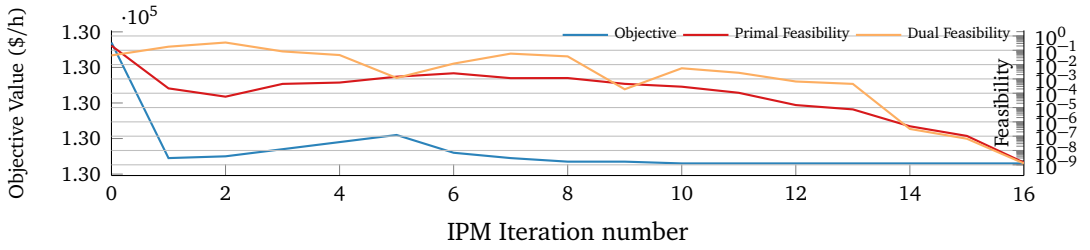
vergence difficulties for the nonconvex OPF problems, especially when it comes to reducing the residuals of the dual infeasibility and ends up performing an excessive number of iterations.

Reduced space IP method

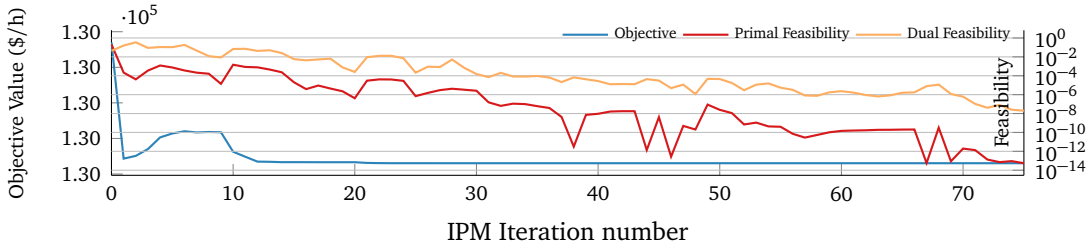
Solution of the OPF problems may be based also on a similar approach as used in the PDE-constrained optimization problems (e.g., maximization of the oil production Kourounis et al. [2014b]). These problems are often solved by reduced space optimization methods Akcelik et al. [2006]; Biegler et al. [2003]. In context of OPF problems, the unknown voltages and the nonlinear equality constraints, representing the power flow equations, are eliminated from the optimization problem and optimizer controls only the reduced set of the variables, so-called control or design variables. The eliminated state variables are treated explicitly during evaluation of the objective function value and its gradient. Given the known PF quantities, system of eliminated equality constraints is used to solve for the unknown state variables. Thus, the equality constraints are implicitly satisfied. However, this requires multiple factorizations of the constraint Jacobians in each IP iteration. In theory, this should not turn out as a serious limitation since the Jacobians are rather small (up to several thousand rows) and robust power flow solvers are available. The efficient evaluation of the gradient information, required by the optimization method, is achieved using the adjoint method. The computational cost of evaluating the inequality constraint (e.g., line flow limits) gradients can be improved using constraint lumping techniques Kourounis et al. [2014b]. The second-order derivatives are not evaluated exactly due to the excessive computational cost, only approximations such as limited memory BFGS are used. The study of such reduced space IP approach in Kardos et al. [2020a] demonstrated that the computational cost of evaluating the gradient information is excessive and the constraint lumping introduces approximations of nonsmooth functions, which leads to convergence difficulties of the IP method used for solution of the OPF problems.

It is well known that quasi-Newton's methods have worse convergence than their exact counterparts. The positive-definite BFGS approximation can be unsuitable when the Lagrangian Hessian is indefinite at the optimal solution. The convergence of quasi-Newton approaches, including the reduced space IP method, are illustrated in Figure 5.3. The quasi-Newton methods require significantly more iterations to reach the required tolerance $\epsilon_{tol} = 10^{-4}$, as shown in Figures 5.3b and 5.3c. The convergence problems might occur due to the fact that the BFGS cannot approximate the Hessian at late iterates (close to the optimal point)

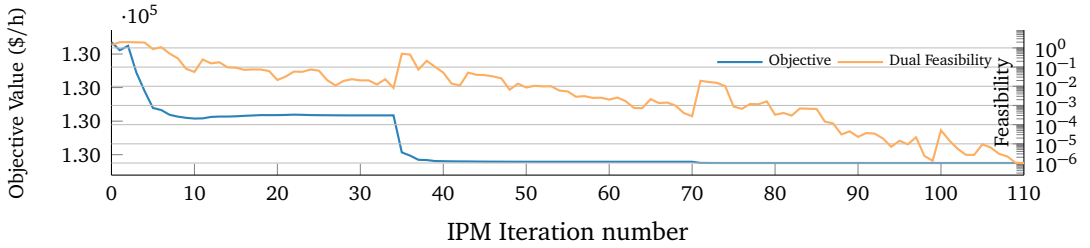
due to the well-known ill-conditioning of the Hessian in the IPM methods as the solution is approached.



(a) IP method with exact Hessian.



(b) Full-Space IP with quasi-Newton limited memory BFGS.



(c) Reduced-Space Quasi-Newton with constraint lumping.

Figure 5.3. Convergence trajectory for case118 power grid OPF benchmark.

Tables 5.1 and 5.2 provide additional information about the reduced space IP method convergence, including performance and precision comparison. In all cases, IPOPT was used as the solution method of the IP optimization problems. The full space IP with exact Hessian information is used as a reference (Exact). Next, the limited memory BFGS approximation of the Hessian in the full-space IP method (BFGS) is shown, followed by the reduced space IP method without constraint lumping (adjoint), and finally, the reduced space with the constraint lumping (lumped). In case of the smaller benchmark (case118), all four methods were able to successfully solve the problem, reaching NLP error in order of 10^{-5} and optimality gap with respect to the exact IP method in order of 10^{-11} in case

Table 5.1. Convergence of various IP solution methods for case118.

Method	Iterations	Time (s)	Stopping Reason	NLP Error	Optimality Gap
Exact	16	0.04	Solved	$1.18 \cdot 10^{-5}$	
BFGS	75	0.98	Solved	$8.19 \cdot 10^{-5}$	$-1.77 \cdot 10^{-11}$
Adjoint	113	5.34	Solved	$7.43 \cdot 10^{-5}$	$-3.52 \cdot 10^{-4}$
Lumped	107	2.02	Solved	$9.09 \cdot 10^{-6}$	$-3.52 \cdot 10^{-4}$

Table 5.2. Convergence of various IP solution methods for case2737.

Method	Iterations	Time (s)	Stopping Reason	NLP Error	Optimality Gap
Exact	26	1.09	Solved	$1.92 \cdot 10^{-5}$	
BFGS	56	7.86	Acceptable tol.	$1.69 \cdot 10^{-1}$	$5.2 \cdot 10^{-9}$
Adjoint	—	—	—	—	—
Lumped	140	30.50	Acceptable tol.	$6.45 \cdot 10^{-3}$	$-8.57 \cdot 10^{-3}$

of the BFGS and 10^{-4} for the reduced space methods. The number of iterations was 4.7 times larger for the BFGS and up to $6.7-7.0$ times for the reduced space approaches, compared to the reference exact solution. The constraint lumping reduced the computational time up to a factor of 2.6.

It was not possible to solve the benchmark case2737 up to the required tolerance. The IP iterations were stopped using the acceptable tolerances. When the algorithm encounters many iterations in a row that are considered acceptable (in this case 15), it will terminate before the desired convergence tolerance is met. This happens if objective function is not improved in many consecutive iterations and the relaxed tolerances are satisfied. The BFGS terminated with a significant NLP error in the dual feasibility of $1.69 \cdot 10^{-1}$. The adjoint method without the constraint lumping was computationally very expensive due to the evaluation of the constraint gradients (5 iterations took almost 200 seconds). However, the constraint lumping method significantly improves performance of the reduced space approach, outperforming also the BFGS approach in terms of the single iteration cost. Similarly, also the NLP error is significantly improved.

Finally, it was observed that approaches using the Hessian approximation are very sensitive to IP parameters such as initial point, barrier strategy and its initial value (monotone strategy should be preferred), length of the BFGS history, etc. Additional results and discussion can be found in Kardos et al. [2020a]. The remaining part of this document does not contain any follow-up of this research direction and focuses on direct sparse solution of the KKT system and its decomposition, considering the exact Hessian information.

Part III

High-performance IP algorithms and software for power grid problems

Chapter 6

Software packages

This chapter provides an overview of available power grid simulation packages and primal-dual IP optimization software. In general, an OPF problem can be formulated as a nonlinear optimal control problem with both equality and inequality constraints. Due to a large number of inequality constraints, IP methods are usually the most efficient solution method. IP methods are favorable for their approach to the inequality constraints, which are replaced by a logarithmic barrier function that penalizes the points approaching the boundary of the feasible region. Finally, the chapter is concluded with a brief review of the linear solvers used by the IP packages, since they heavily influence the performance and robustness of the overall IP software. The overall hierarchy of the software components is illustrated in Figure 6.1.

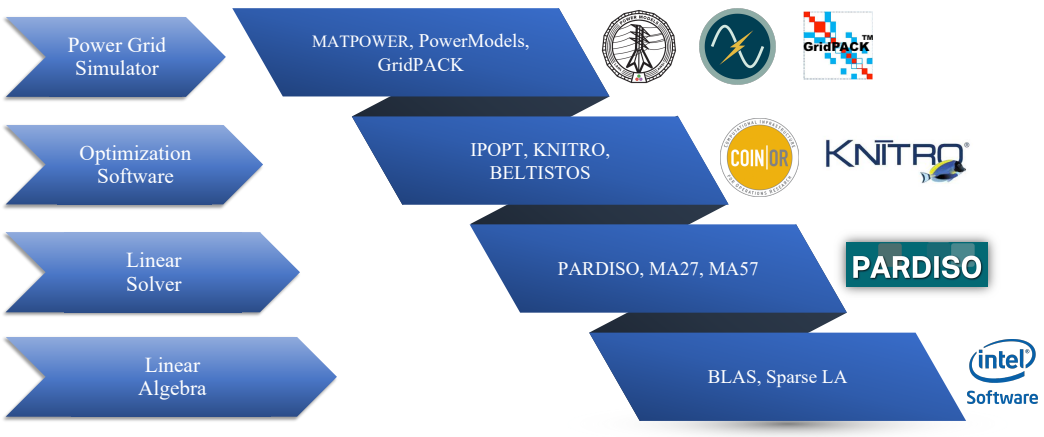


Figure 6.1. Software stack in power grid simulations.

6.1 Power grid simulation packages

Although many commercial software tools exist that support load flow and OPF simulation capabilities, including PowerFactory (DIgSILENT), PSS (Siemens), NEPLAN (NEPLAN AG), PSLF(GE ENERGY CONSULTING), ETAP Grid (ETAP), their licensing makes them prohibitive in many cases, thus focus is put on freely available software, which is introduced in the following subsections.

6.1.1 MATPOWER

MATPOWER Zimmerman et al. [2011]; Zimmerman and Murillo-Sanchez [2016] is a general purpose power system software, which employs all of the standard steady-state models typically used for power flow analysis. It is an open-source MATLAB-based package, and provides researchers and educators a platform for solving power flow and an extensible collection of OPF problems. Recently, several different formulations of the standard AC OPF problem were added, including polar and rectangular representations of complex voltage variables and both current and power versions of the nodal mismatch equations. The modular architecture of the MATPOWER code enables researchers to implement various OPF extensions or add user defined variables and functions. Interfaces to multiple high-performance nonlinear optimizers, such as FMINCON, IPOPT, KNITRO, BELTISTOS, and its default IP solver MIPS, are also available for its users. It also contains a library of several power networks of increasing complexity.

A subset of the MATPOWER power flow model was implemented in C++, used for evaluation of the HPC solution algorithms presented in this thesis. OPF extensions were also implemented, including SCOPF and MPOPF.

6.1.2 PowerModels

PowerModels Coffrin et al. [2017] is an open-source platform providing various OPF problem formulations, including several convex relaxations. JuMP Dunning et al. [2017], a domain-specific modeling language for mathematical optimization embedded in Julia is used to formulate the OPF model as an optimization problem. JuMP supports a variety of commercial or open-source solvers.

PowerModels provides similar modules to MATPOWER, but since it is implemented in Julia it allows conducting large-scale experiments where hundreds of independent tasks can be run in parallel. Associated MATPOWER's MATLAB licensing costs can be prohibitive in a nonacademic environments, while Julia is

provided under open source license. Additionally, it also provides second-order conic and quadratic relaxations of the OPF problem.

6.1.3 GridPACK^{TM}

GridPACK^{TM} Palmer et al. [2015]; Palmer et al. [2014] is an open source software framework for the development of power system simulation applications. The framework contains modules for setting up distributed power grid networks, and is built on top of third party libraries for communication (MPI), partitioning (Parmetis), distributed matrices and vectors, linear and nonlinear solvers (PETSc), and advanced programming constructs (Boost). The simulations are thus capable of running on high-performance computers. It was used in applications such as power flow, dynamic simulation, and state estimation. It also contains building blocks and interfaces that can be used to formulate a variety of optimization applications. The provided building blocks consist of constructs such as variable, expression, or constraint interfaces.

6.2 IP optimization packages

In what follows, several primal-dual IP software packages are described, which are used by many practitioners for solving NLP problems. The optimization software packages are summarized in Table 6.1, where the freely available solvers are highlighted. Structure exploiting capability is also indicated.

Table 6.1. Open source and commercial optimizers.

Optimizers	Version	Structured	License
IPOPT	3.12.5	no	Open source (EPL)
BELTISTOS	1.0	yes	Free academic use
KNITRO	11.0.1	no	Artelys
MIPS	1.2.2	no	Open source (BSD)*
FMINCON	2017b	no	MATLAB
PIPS		yes	Open source (UChicago Argonne)
OOQP		yes	Open source (University of Chicago)
OOPS		yes	Demo available

6.2.1 IPOPT

IPOPT Wächter and Biegler [2005, 2006]; Nocedal et al. [2009] is a software package for large-scale nonlinear optimization. It implements a primal-dual IP algorithm with a filter line-search method that aims to find a local solution of a given NLP problem. It includes additional algorithmic features such as second-order corrections, inertia correction of the KKT matrix, problem scalings, and various barrier update strategies. Furthermore, it provides tools such as derivative checker to verify the correctness of the user provided derivatives, or implements a first-order algorithm based on the Hessian approximation using limited memory BFGS (l-BFGS). It supports a variety of linear solvers, including HSL MA27, MA57, MUMPS, PARDISO, WSMP and others. Benchmarks Gould et al. [2007] have shown that MA57 and PARDISO are often the most reliable linear solvers for the indefinite problems within IPOPT.

6.2.2 BELTISTOS

BELTISTOS Kourounis and Schenk [2018]; Kourounis et al. [2018]; Kardos et al. [2020b] is a collection of high-performance OPF solution algorithms including extremely scalable and low memory MPOPF and SCOPF solvers. BELTISTOS-OPF encapsulates the most efficient algorithmic kernels implemented in IPOPT, which perform best for OPF problems on a wide variety of networks of increasing complexity. It attempts to maintain sufficiently accurate search directions by controlling sophisticated pivoting and scaling schemes provided by the direct sparse solver PARDISO. Additionally, when the search directions are not sufficiently accurate, they are refined by a sophisticated iterative refinement scheme Arioli and Scott [2014] or by performing iterative refinement using quadruple-precision floating-point arithmetic. Furthermore, special factorization schemes are applied for accelerating structured problems. BELTISTOS-MP implements structure exploiting and data compression algorithms designed for the particular structure of the MPOPF problems. BELTISTOS-MEM is a variant of the BELTISTOS-MP algorithm, which sacrifices redundant computation in favor of memory efficiency. BELTISTOS-SC addresses solution of the SCOPF problems by a parallel, structure exploiting IP algorithm, utilizing both shared and distributed memory environments, also supporting GPU for acceleration of the dense linear algebra.

6.2.3 KNITRO

KNITRO Byrd et al. [2006] is a commercial software package for solving large-scale mathematical NLP problems. KNITRO offers four different optimization algorithms for solving optimization problems, including an IP with an iterative or direct algorithm, SQP, and an active set algorithm. For the purpose of this study, the IPM algorithm with direct step was selected, computing new iterates by solving the primal-dual KKT matrix using direct linear algebra and sparse symmetric indefinite solvers such as MA27, MA57 or MKL PARDISO Intel [2019], or others. However, KNITRO may automatically switch to an iterative conjugate gradient solver if the direct step is suspected to be of poor quality, or if negative curvature is detected. KNITRO implements a couple of line-search methods, backtracking or cubic interpolation scheme. By default, the strategy is chosen automatically by KNITRO and is valid only for the IP algorithm with direct step or SQP. KNITRO also implements presolver that tries to simplify the model by removing variables or constraints. Other features include second-order corrections, derivative checker, multiple Hessian approximations (exact and l-BFGS, SR1), various barrier update strategies, or problem scaling.

6.2.4 MIPS

MIPS Wang et al. [2007], the MATPOWER IP solver, is a primal-dual IP solver for general NLP problems; however, OPF problems are the primary target of the solver. It is entirely implemented in MATLAB code and distributed with MATPOWER. Although it is released under open source license, it requires a MATLAB license for execution of the code. It implements an algorithm where the step control is not enabled by default, which often leads to numerical failure of the solver. It is strongly recommended to enable the additional step-size control in the MIPS algorithm. The exact derivatives are required (both first and second order) and it uses an adaptive strategy for the barrier parameter update (based on the duality measure scaled by a constant). Various linear solvers are supported, with MATLAB's backslash being the default solver. In addition, an interface to the PARDISO linear solver is available.

6.2.5 FMINCON

FMINCON Byrd et al. [2000] is a part of the MATLAB Optimization Toolbox MathWorks [2018]. By default FMINCON uses its IP solver, but other algorithms are available (including SQP, active set or trust region algorithms). The IP algorithm

applies projected conjugate gradient method to solve the KKT system in an iterative fashion. Finite difference derivative checker, or the l-BFGS algorithm are also available.

6.2.6 PIPS

PIPS Petra [2014] is a suite of parallel optimization solvers designed mainly for stochastic optimization problems. PIPS consists of multiple IP solvers for various problem classes and is designed to exploit particular problem structures using customized parallel linear algebra. PIPS-IPM Petra, Schenk and Anitescu [2014] is a parallel IP solver for stochastic LPs and convex QPs. The solver supports the PARDISO linear solver and exploits its functionality to greatly improve the performance of the local Schur complement contributions, which can be computed much faster then when used with other linear solvers (such as supported MA27, MA57, MA86, WSMP) and can thus offer HPC performance. It also supports GPUs for the acceleration of dense linear algebra. General NLP problems can be addressed by PIPS-NLP Chiang [2014]; Chiang et al. [2014], which implements a parallel IP method for structured problems. Supported linear solvers include MA27, MA57, MA86, Mumps, and UmfPack. PIPS-IPM and PIPS-NLP are both derivative works of OOQP Gertz and Wright [2003].

6.2.7 OOQP

OOQP Gertz and Wright [2003] is a package for solving solving convex QPs based on a primal-dual IP algorithm, where users may exploit problem structure by supplying linear algebra, problem data, and variable classes that are customized to their particular applications. The OOQP distribution contains default implementations that solve several important QP problem types, including general sparse and dense QPs and bound-constrained QPs. The MA27/57 solvers are supported. The larger goal of OOQP is to demonstrate the usefulness of object-oriented design principles in the context of optimization software and can incorporate other software for sparse systems and for parallel environments, without requiring substantial rewriting of the code.

6.2.8 OOPS

OOPS Gondzio and Grothey [2009] is a parallel IP code that exploits any problem structure and it solves LP, QP, and NLP problems. It also supports nested block-structured matrices that occur in problems such as multistage stochastic

programming. It provides an interface to a collection of linear algebra routines that need to be implemented for all supported structures.

6.3 Linear solvers

The solution of linear systems of equations is the cornerstone of a robust high-performance optimization package. Several sparse direct linear solvers are described in the following section, focusing on solvers used within this thesis. Specific performance benchmarks of the linear solvers, applied to the solution of OPF problems, are presented in section 8.2.4.

6.3.1 PARDISO

PARDISO Schenk and Gärtner [2004]; Schenk and Gärtner [2006b] is a thread-safe, high-performance, robust, memory efficient software for solving large sparse symmetric and unsymmetric linear systems of equations on shared-memory multiprocessors. PARDISO uses a combination of left- and right-looking Level-3 BLAS supernode techniques, utilizing OpenMP directives to achieve multithreaded parallelism. The solver uses diagonal pivoting or 1×1 and 2×2 Bunch–Kaufman pivoting for symmetric indefinite matrices, typical for IP methods, and an approximation of the solution is found by forward and backward substitution and iterative refinement. Additionally, symmetric weighted matching algorithms are used to improve the pivoting accuracy.

IPOPT, BELTISTOS, PIPS-IPM, and MIPS contain ready to use interfaces to the solver. Additionally, PARDISO is distributed with additional fully integrated components such as the parallel threaded implementation of the Schur-complement algorithm, called PARDISO-SCHUR Petra, Schenk, Lubin and Gärtner [2014], which is available for sparse symmetric or unsymmetric matrices. PARDISO-INV's Verbosio et al. [2017] selected inversion method provides an efficient way for computing, e.g., the diagonal elements of the inverse of a sparse matrix, and is available for sparse symmetric or unsymmetric matrices.

6.3.2 The Harwell Subroutine Library

The Harwell Subroutine Library (HSL) HSL [2002] is a Fortran library for many areas in scientific computing, including direct and iterative solvers, or various auxiliary linear algebra functions. It is probably best known for its codes for the direct solution of sparse linear systems, including multifrontal algorithm with

approximate minimum degree ordering implemented in MA57. The system can be optionally prescaled by using various scaling routines, while ordering options are provided including hooks to MeTiS Gupta et al. [1997]; Karypis and Kumar [1998]. Evaluation of the individual solvers in terms of robustness and performance is provided in Gould and Scott [2004]. HSL packages are available at no cost for academic research and teaching.

IPOPT provides support for a wide variety of linear solvers, including HSL linear solvers MA27, MA57. KNITRO may utilize routines MA27 or MA57 in order to solve linear systems arising at every iteration of the algorithm. MATLAB uses the MA57 routines for real sparse symmetric matrices (operator `ldl`).

Chapter 7

Structure exploiting solution methods

Computers have evolved significantly over the past decade, at an even faster pace than modern power grids. Multicore and manycore computer architectures and distributed compute clusters are ubiquitous among scientists and engineers, while at the same time no significant performance gains are expected for sequential codes. Historically, the easy performance gains could be achieved due to the increased clock frequencies of newer processors, which was the case until roughly 2005 when the CPU clock settled at around 3 GHz and stopped increasing Rupp [2018]. Significant performance gains, however, may be achieved by algorithmic redesign tailored to the particular application that is also able to utilize multicore and manycore architectures with deep memory hierarchies.

IP methods have been the most robust and successful tools for large-scale nonconvex optimization, given that IP methods can easily exploit the problem structure Gondzio and Grothey [2009]. The practical efficiency of the IP algorithms highly depends on the linear algebra kernels used and performance gains must be realized through the use of sophisticated algorithms utilizing parallelism across all cores of processors or distributed memory clusters. For large-scale optimization problems, the computation of the search direction (4.16) determines the overall runtime. In such cases, the KKT system is usually very large but sparse, especially for coupled problems such as SCOPF or MPOPF. Direct sparse linear solvers are the standard choice employed for the solution of the KKT; however, the solution quickly becomes intractable due to extensive memory and time requirements. Hence, any attempt at accelerating the solution should be focused on the efficient solution of the KKT linear system. Figure 7.1 demonstrates how various IP method components contribute to the overall time for various OPF benchmarks. The number of IP iterations was fixed at five. Note that the solution of the linear system represents the majority of the overall time.

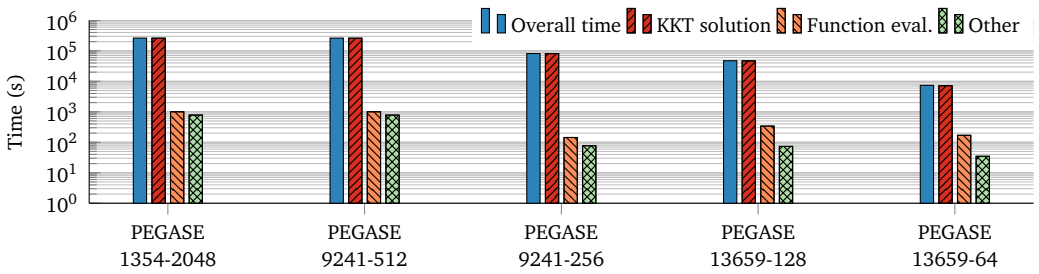


Figure 7.1. Computational complexity of the IP method components.

Real-world real-time implementation of coupled OPF problems for large-scale energy systems still remain computationally intractable. The coupled OPF problems are ubiquitous in various domains, owing to the presence of smart loads and energy storage devices such as batteries for demand shaping and deferral. Additional time couplings of the OPF problem at each time period are introduced by generator ramp rate limits. The higher the number of time periods considered, the larger the resulting optimal control problem becomes. For a significantly large number of time periods the problem becomes notoriously difficult to solve and for this purpose several approximations and simplifications are currently employed by the industry in order to meet real-time responses. Furthermore, the system operators have to foresee possible contingency events and operate the grid in a such a way that its operation will remain secure in the event of any contingency. However, addition of a large number of contingency scenarios results in significantly larger problem sizes, rendering the problem solution computationally intractable.

7.1 Revealing the structure of coupled OPF problems

A widespread approach for solving KKT systems consists of employing black-box techniques such as direct sparse solvers, due to their accuracy and robustness. The direct sparse solvers obtain the solution of the linear system by factorization and subsequent forward-backward substitutions. The factorization is a computationally expensive operation, with complexity $\mathcal{O}(\frac{2}{3}n^3)$, commonly introducing significant fill-in, which may quickly exhaust available memory on shared memory machines for large-scale linear systems. Furthermore, these solvers are not aware of the underlying structural properties of the KKT systems arising from many engineering problems which make it possible to significantly decrease time to solution by employing structure-exploiting algorithms and distributed memory

computers.

The appropriate structure emerges from the fact that each of the variables in the SCOPF optimization vector 3.24 $\left[\boldsymbol{x}, \boldsymbol{\lambda}_{\varepsilon}, \boldsymbol{\lambda}_{I}\right]$ or the MPOPF optimization vector 3.32 $\left[\boldsymbol{x}, \boldsymbol{\lambda}_{\varepsilon}, \boldsymbol{\lambda}_{I}, \boldsymbol{\lambda}_{A}\right]$ correspond to some contingency scenario $c = 0, 1, \ldots, N_c$, or the time period $n = 1, 2, \ldots, N$:

$$\boldsymbol{x} = \left[\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N_c}, \boldsymbol{x}_g\right], \tag{7.1}$$

$$\boldsymbol{\lambda}_{\varepsilon} = \left[\boldsymbol{\lambda}_{\varepsilon 0}, \ldots, \boldsymbol{\lambda}_{\varepsilon N_c}\right], \tag{7.2}$$

$$\boldsymbol{\lambda}_{I} = \left[\boldsymbol{\lambda}_{I0}, \ldots, \boldsymbol{\lambda}_{IN_c}\right], \tag{7.3}$$

$$\boldsymbol{x} = \left[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\right], \tag{7.4}$$

$$\boldsymbol{\lambda}_{\varepsilon} = \left[\boldsymbol{\lambda}_{\varepsilon 1}, \ldots, \boldsymbol{\lambda}_{\varepsilon N}\right], \tag{7.5}$$

$$\boldsymbol{\lambda}_{I} = \left[\boldsymbol{\lambda}_{I1}, \ldots, \boldsymbol{\lambda}_{IN}\right], \tag{7.6}$$

$$\boldsymbol{\lambda}_{A} = \left[\boldsymbol{\lambda}_{A1}, \ldots, \boldsymbol{\lambda}_{AN}\right]. \tag{7.7}$$

In order to reveal the scenario-local structure of the Hessian (4.21), the variables corresponding to the same contingency or time period are grouped together, i.e.,

$$\boldsymbol{u}_c = \left[\boldsymbol{x}_c,\ \boldsymbol{\lambda}_{\varepsilon c},\ \boldsymbol{\lambda}_{Ic}\right], \tag{7.8}$$

$$\boldsymbol{u}_n = \left[\boldsymbol{x}_n,\ \boldsymbol{\lambda}_{\varepsilon n},\ \boldsymbol{\lambda}_{In}\right], \tag{7.9}$$

and, thus, the global ordering will be

$$\boldsymbol{u} = \left[\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{N_c}, \boldsymbol{u}_g\right], \tag{7.10}$$

$$\boldsymbol{u} = \left[\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N, \boldsymbol{u}_g\right], \tag{7.11}$$

where the coupling variables $\boldsymbol{u}_g$ are placed at the end of the new optimization vector $\boldsymbol{u}$. Coupling in the SCOPF problem, $\boldsymbol{u}_g = \boldsymbol{x}_g$, is introduced by the two nonanticipatory constraints (3.21h) and (3.21i). The coupling in a case of the MPOPF problem, $\boldsymbol{u}_g = \boldsymbol{\lambda}_A$, is introduced by the linear energy constraints (3.25h). Under the new orderings (7.10) and (7.11), the Hessian matrix of the KKT system (4.21) becomes of an arrowhead structure (also described as bordered block-diagonal Duff and Scott [2005] or dual block-angular Petra, Schenk and Anitescu [2014]),

$$\begin{pmatrix} \boldsymbol{A}_0 & & & & \boldsymbol{B}_0^{\intercal} \\ & \boldsymbol{A}_1 & & & \boldsymbol{B}_1^{\intercal} \\ & & \ddots & & \vdots \\ & & & \boldsymbol{A}_n & \boldsymbol{B}_n^{\intercal} \\ \boldsymbol{B}_0 & \boldsymbol{B}_1 & \ldots & \boldsymbol{B}_n & \boldsymbol{C} \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{u}_0 \\ \Delta\boldsymbol{u}_1 \\ \vdots \\ \Delta\boldsymbol{u}_n \\ \Delta\boldsymbol{u}_g \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_{N_c} \\ \boldsymbol{b}_C \end{pmatrix}, \tag{7.12}$$

also illustrated in Figures 7.2 and 7.3. The block matrices $\boldsymbol{A}_i$ for the SCOPF problem are

$$\boldsymbol{A}_i = \begin{pmatrix} \tilde{\boldsymbol{H}}_{x_i,x_i} & \boldsymbol{J}_{\varepsilon_i,x_i}^{\intercal} & \boldsymbol{J}_{I_i,x_i}^{\intercal} \\ \boldsymbol{J}_{\varepsilon_i,x_i} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{J}_{I_i,x_i} & \boldsymbol{0} & -L_{s_i}^{-1} \end{pmatrix}. \tag{7.13}$$

Similarly, the block matrices A_i for the MPOPF problem are

$$\boldsymbol{A}_i = \begin{pmatrix} \tilde{\boldsymbol{H}}_{x_i,x_i} & \boldsymbol{J}^{\intercal}_{\varepsilon_i,x_i} & \boldsymbol{J}^{\intercal}_{I_i,x_i} & \mathbf{0} \\ \boldsymbol{J}_{\varepsilon_i,x_i} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{J}_{I_i,x_i} & \mathbf{0} & -L^{-1}_{s_i} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{L}_{A_i} \end{pmatrix}. \tag{7.14}$$

The matrices incorporate the Hessian of the Lagrangian with respect to the scenario-local variables $\tilde{\boldsymbol{H}}_{x_i,x_i} = \nabla^2_{x_i x_i}\mathcal{L} + X_i^{-1}Z_i$ and the Jacobians of the constraints for the ith scenario with respect to the local variables $\boldsymbol{J}_{\varepsilon_i,x_i} = \nabla_{x_i}\boldsymbol{c}_{\varepsilon i}$ and $\boldsymbol{J}^{\intercal}_{I_i,x_i} = \nabla_{x_i}\boldsymbol{c}_{Ii}$, as well as the diagonal entries corresponding to the eliminated slack variables. In the case of the SCOPF problem, the block $\boldsymbol{C} = \nabla^2_{x_g x_g}\mathcal{L} + X_g^{-1}Z_g$ contains Hessian of the Lagrangian with respect to the coupling variables $\boldsymbol{x}_g$, while in the case of the MPOPF problem it is a block of zeros. The off-diagonal blocks in the arrowhead SCOPF system are

$$\boldsymbol{B}_i = \begin{pmatrix} \tilde{\boldsymbol{H}}_{x_g,x_i} \\ \boldsymbol{J}^{\intercal}_{\varepsilon_i,x_g} \\ \boldsymbol{J}^{\intercal}_{I_i,x_g} \end{pmatrix}^{\intercal}, \quad \boldsymbol{B}_i^{\intercal} = \begin{pmatrix} \tilde{\boldsymbol{H}}_{x_i,x_g} \\ \boldsymbol{J}_{\varepsilon_i,x_g} \\ \boldsymbol{J}_{I_i,x_g} \end{pmatrix}, \tag{7.15}$$

where $\tilde{\boldsymbol{H}}_{x_i,x_g} = \nabla^2_{x_i x_g}\mathcal{L}$ represents the off-diagonal blocks of the Hessian of Lagrangian with respect to the local and coupling variables and $\boldsymbol{J}_{\varepsilon_i,x_g} = \nabla_{x_g}\boldsymbol{c}_{\varepsilon i}$ and $\boldsymbol{J}_{I_i,x_g} = \nabla_{x_g}\boldsymbol{c}_{Ii}$ are the Jacobians of the ith scenario with respect to the coupling variables. The MPOPF coupling matrices $\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_N \in \mathbb{R}^{NN_S \times N_A}$, where N_A is the size of the diagonal blocks in (7.12), contain the constant subblocks, which arise from the particular form of the linear constraints (3.25h) representing the evolution of stored energy.

7.2 Schur complement decomposition

The direct factorization of the full KKT system is not feasible for large-scale SCOPF problems due to their growing size with the number of contingencies and associated factorization fill-in that quickly exhausts the available memory. The systems with the arrowhead structure, such as (7.12), are well suited to be solved by the Schur complement (SC) technique. The SC arises as the result of performing a block Gaussian elimination, thus reducing the problem from the full KKT system into smaller problems corresponding to the diagonal blocks $\boldsymbol{A}_i$ and the dense SC system $\boldsymbol{S}$. The solution is obtained by a sequence of partial block

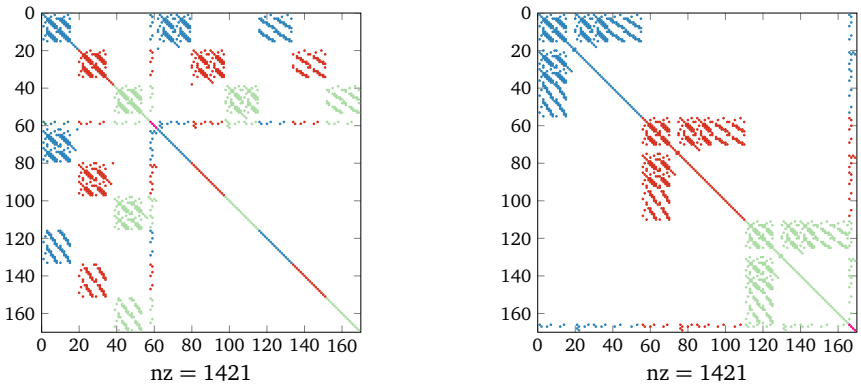


Figure 7.2. Symmetrized SCOPF system (4.21) and its permutation (7.12).

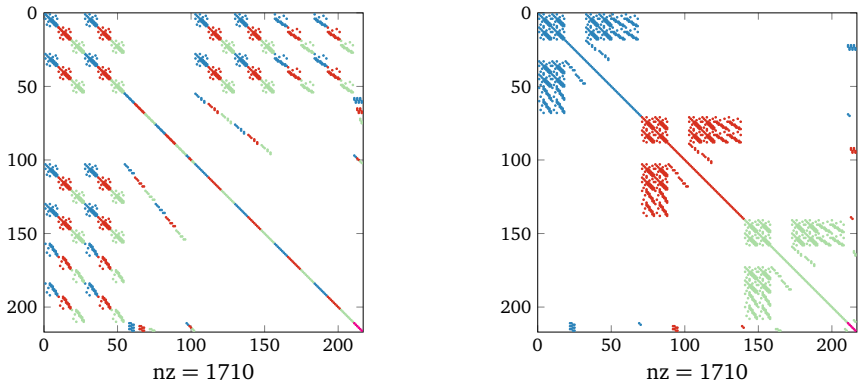


Figure 7.3. Symmetrized MPOPF system (4.21) and its permutation (7.12).

elimination steps, which are decoupled, aiming to form the SC of the system. This way, the factorization of the full KKT system is avoided. Instead, only the smaller diagonal blocks need to be factorized, as described in the Algorithm 1.

In the first step, the SC $\boldsymbol{S}$ is formed,

$$\boldsymbol{S} = \boldsymbol{C} - \sum_{i=0}^{N_c} \boldsymbol{B}_i \boldsymbol{A}_i^{-1} \boldsymbol{B}_i^{\mathsf{T}}, \tag{7.16}$$

which in the general case becomes a dense matrix. In case of the preventive SCOPF problem formulation, the size of the coupling stays constant, independently of the number of contingency scenarios. Therefore, the size of the SC does not increase with an increasing number of contingencies. It can therefore be solved using dense $\boldsymbol{LDL}^{\mathsf{T}}$ factorization and back substitution algorithms. In case of the MPOPF problem, the dense SC $\boldsymbol{S}$ requires additional treatment, since the coupling increases with increasing number of time periods. The additional

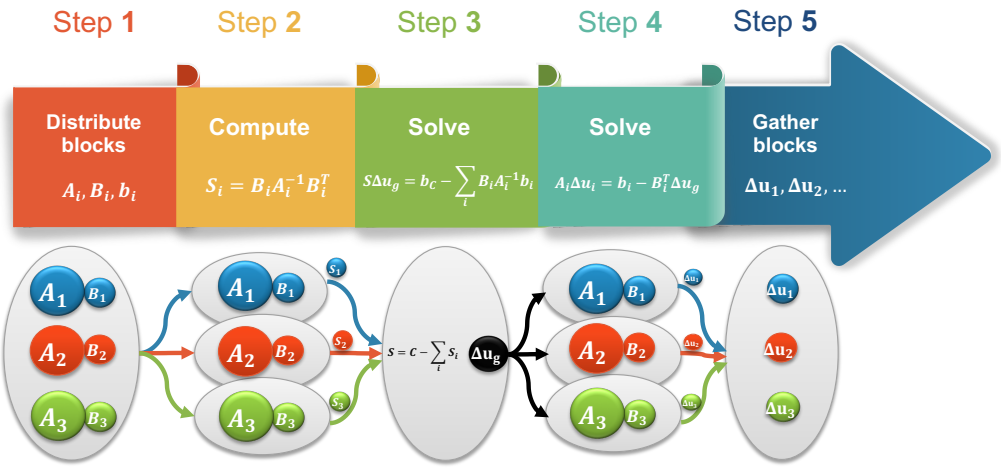


Figure 7.4. Schur complement procedure for solving the KKT system.

treatment is discussed in section 7.4. The solution of the dense Schur system,

$$\boldsymbol{S}\Delta\boldsymbol{u}_g = \boldsymbol{b}_C - \sum_{i=0}^{N_c} \boldsymbol{B}_i \boldsymbol{A}_i^{-1} \boldsymbol{b}_i, \tag{7.17}$$

yields a part of the solution corresponding to the coupling variables $\Delta\boldsymbol{u}_g$, which is used to obtain all the local solutions $\Delta\boldsymbol{u}_i$ by solving

$$\boldsymbol{A}_i \Delta\boldsymbol{u}_i = \boldsymbol{b}_i - \boldsymbol{B}_i^{\intercal} \Delta\boldsymbol{u}_g. \tag{7.18}$$

Since the block contributions to the SC $\boldsymbol{B}_i \boldsymbol{A}_i^{-1} \boldsymbol{B}_i^{\intercal}$ are independent, they can be evaluated in parallel, as well as the residuals $\boldsymbol{B}_i \boldsymbol{A}_i^{-1} \boldsymbol{b}_i$ and the solutions $\Delta\boldsymbol{u}_i$ which can be computed independently at each process. Interprocess communication occurs because the local SC contributions and SC residuals need to be assembled by the *master* process, and during the broadcast of the SC solution to the remaining processes, as illustrated in Figure 7.4.

Remark 3 *Note that the computational efficiency obtained by exploiting the block-diagonal structure, such as* (7.12)*, is determined by the number of the coupling variables* $|\boldsymbol{u}_g|$*. If coupling is large, then the Schur decomposition will not be efficient compared to the direct factorization techniques because of the cubic complexity of dense factorizations* (7.17)*.*

The most expensive step of the presented computational scheme is evaluation of the local contributions to the SC $\boldsymbol{B}_i \boldsymbol{A}_i^{-1} \boldsymbol{B}_i^{\intercal}$ in (7.16). The standard approach

Algorithm 1 Parallel SC decomposition for solving the KKT system.

Require: KKT system with arrowhead structure (7.12), right-hand side $\boldsymbol{b}$
Ensure: $\Delta \boldsymbol{u} := KKT^{-1}\boldsymbol{b}$

1: Distribute blocks from the KKT system (7.12) evenly across $\mathcal{P}$ processes, where $\mathcal{N}_p$ is the set of diagonal blocks assigned to process $p \in \mathcal{P}$
2: Factorize $\boldsymbol{A}_i = \boldsymbol{L}_i D_i \boldsymbol{L}_i^\intercal$ for each $i \in \mathcal{N}_p$
3: Compute $\boldsymbol{S}_i = \boldsymbol{B}_i \boldsymbol{A}_i^{-1} \boldsymbol{B}_i^\intercal$ for each $i \in \mathcal{N}_p$
4: Accumulate $\boldsymbol{C}_p = \sum_{i \in \mathcal{N}_p} \boldsymbol{S}_i$
5: **if** *master* **then**
6: Reduce $\boldsymbol{S} = \boldsymbol{C} - \sum_{p \in \mathcal{P}} \boldsymbol{C}_p$
7: **end if**
8: Compute $\boldsymbol{r}_i = \boldsymbol{B}_i \boldsymbol{A}_i^{-1} \boldsymbol{b}_i$ for each $i \in \mathcal{N}_p$
9: Accumulate $\boldsymbol{r}_p = \sum_{p \in \mathcal{P}} \boldsymbol{r}_i$
10: **if** *master* **then**
11: Reduce $\boldsymbol{r} = \sum_{p \in \mathcal{P}} \boldsymbol{r}_p$
12: Factorize $\boldsymbol{S} = L_s D_s L_s^\intercal$
13: Solve $\boldsymbol{S} \Delta \boldsymbol{u}_g = \boldsymbol{b}_C - \boldsymbol{r}$
14: Broadcast solution $\boldsymbol{u}_g$ to all $p \in \mathcal{P}$
15: **end if**
16: Solve $\boldsymbol{A}_i \Delta \boldsymbol{u}_i = \boldsymbol{B}_i \boldsymbol{u}_g - \boldsymbol{b}_i$ for each $i \in \mathcal{N}_p$

uses a direct sparse solver, such as PARDISO, to factorize the symmetric matrix $\boldsymbol{A}_i = \boldsymbol{L}_i D_i \boldsymbol{L}_i^\intercal$ and perform multiple forward-backward substitutions with all right-hand side (RHS) vectors in $\boldsymbol{B}_i^\intercal$, followed by multiplication from the left by $\boldsymbol{B}_i$. This approach, however, does not exploit sparsity of the problem in $\boldsymbol{B}_i^\intercal$ blocks, since the linear solver treats the RHS vectors as being dense.

An alternative approach, implemented in PARDISO, addresses these limitations by performing an incomplete factorization of the augmented matrix Petra, Schenk, Lubin and Gärtner [2014]:

$$\boldsymbol{M}_i = \begin{pmatrix} \boldsymbol{A}_i & \boldsymbol{B}_i^\intercal \\ \boldsymbol{B}_i & 0 \end{pmatrix}, \tag{7.19}$$

exploiting also the sparsity of $\boldsymbol{B}_i^\intercal$. The factorization of the augmented matrix $\boldsymbol{M}_i$ is stopped after pivoting reaches the last diagonal entry of $\boldsymbol{A}_i$. At this point, the term $-\boldsymbol{B}_i \boldsymbol{A}_i^{-1} \boldsymbol{B}_i^\intercal$ is computed and resides in the (2, 2) block of $\boldsymbol{M}_i$. By exploiting the sparsity not only in $\boldsymbol{A}_i$, but also in $\boldsymbol{B}_i$ it is possible to reduce memory traffic by using in-memory sparse matrix compression techniques, which render this approach quite favorable for multicore parallelization.

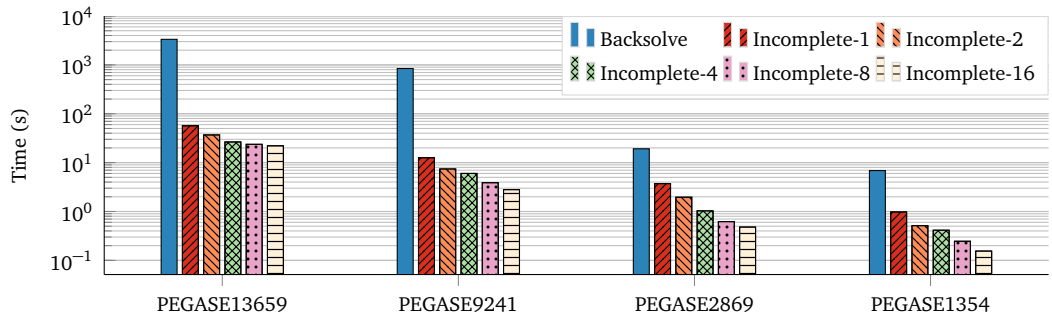


Figure 7.5. Evaluation of the $\boldsymbol{B}_i\boldsymbol{A}_i^{-1}\boldsymbol{B}_i^T$ terms using backsolve and incomplete factorization of the augmented matrix $\boldsymbol{M}_i$ with multiple threads proposed in Petra, Schenk, Lubin and Gärtner [2014].

Note that by using an incomplete factorization approach to the augmented matrix $\boldsymbol{M}_i$ for directly computing the local SC $\boldsymbol{S}_i$, some perturbation is introduced to the local SC to allow for more efficient pivoting during their computation. However, also the factorization of the diagonal blocks is perturbed, which can be implicitly obtained from the incomplete factorization approach. Therefore, the iterative refinement is essential and needs to be applied to each solve operation with the diagonal block $\boldsymbol{A}_i$ in Algorithm 1. However, introducing the iterative refinement might introduce load imbalance for the parallel computations, since each solve with $\boldsymbol{A}_i$ for different i might require different number of iterative refinement iterations to reach the desired residual.

Performance of computing the terms $\boldsymbol{B}_i\boldsymbol{A}_i^{-1}\boldsymbol{B}_i^{\intercal}$ is shown in Figure 7.5 for various benchmarks. The standard (so-called "backsolve") approach and the multicore incomplete factorization are shown. The latter is also shown for increasing number of cores. This demonstrates that the incomplete factorization approach is orders of magnitude faster, especially for the large problems. Due to the extensive memory requirements for storing the RHS vectors in the "backsolve" approach, only its single-core execution is demonstrated.

7.3 Solution algorithms for SCOPF problems

The parallel SC framework was applied to AC SCOPF problems and associated KKT systems in Kang [2015]; Jiang and Xu [2014]. The studies do not evaluate the algorithm on large-scale problems and demonstrate scaling only up to 16 or 8 MPI processes, respectively. The former work focuses on solving the

SC equations implicitly using an iterative quasi-Newton preconditioned conjugate gradient method. Structured non-convex optimization of large-scale energy systems using PIPS-NLP was performed in Schanen et al. [2018]; Chiang et al. [2014]. The parallel interior point optimization solver for nonlinear programming leverages the dual-block angular structure specific to the problem formulation by applying the SC for efficient parallelization of the linear solves. It was illustrated how different model structures arise in power system domains and how these can be exploited to achieve high computational efficiency. Stochastic optimization problems on high-performance computers have been treated similarly in Petra, Schenk, Lubin and Gärtner [2014]; Petra, Schenk and Anitescu [2014]. The numerical experiments suggest that supercomputers can be efficiently used to solve power grid optimization problems with thousands of scenarios under the strict time requirements of power grid operators, particularly due to improved linear algebra on a shared memory level.

The proposed additional SC scheme in this work differentiates the solution algorithm from the one suggested in the previous work, as summarized in the recent manuscript Kardoš et al. [2020] and sections 5.2.1 and 8.3.1 in this document. The benefits come from a more efficient direct sparse approach for the solution of the underlying sparse KKT system, introduced in (4.19)–(4.22). The proposed method employs one additional SC carefully chosen so that the sparsity of the KKT matrix is maintained. This way the size of the linear system decreases and this allows for memory savings and increased computational performance.

7.4 Structure exploiting algorithms for MPOPF

For the MPOPF problems, the size of the dense SC $\boldsymbol{S}$ grows very quickly, not only with the size of the network but also proportionally to the number of installed storage devices and the number of time periods NN_S. As the number of time periods N or storage devices N_S increases, the solution approach based on Algorithm 1 results in a less efficient algorithm than the direct sparse approach employing PARDISO on the original KKT system (4.15), both with respect to computational time and memory consumption despite the benefits of the Schur decomposition. However, the MPOPF problem, unlike the SCOPF problem, can be optimized even further by exploiting the particular structure of the off-diagonal blocks $\boldsymbol{B}_n$, as proposed in Kourounis et al. [2018]. The dense SC consists of smaller blocks, which appear repeatedly in the matrix. The SC can thus be computed, stored and factorized in more economical manner, compared to the naive black-box approach.

7.4.1 Distribution system flexibility

The structure of the MPOPF problem extended by the flexibility provision (introduced in section 3.3.2) is very similar to the MPOPF structure presented previously in Kourounis et al. [2018]. Structure of both problems is demonstrated in Figures 7.6 and 7.7. There are additional (separable) constraints and variables in each diagonal block and extra coupling variables, corresponding to the Lagrange multipliers of the linear constraints (3.25h) considering also the flexibility provision. The KKT system can be permuted to the arrowhead form and the structure exploiting approach, as presented before, can be applied in a similar manner.

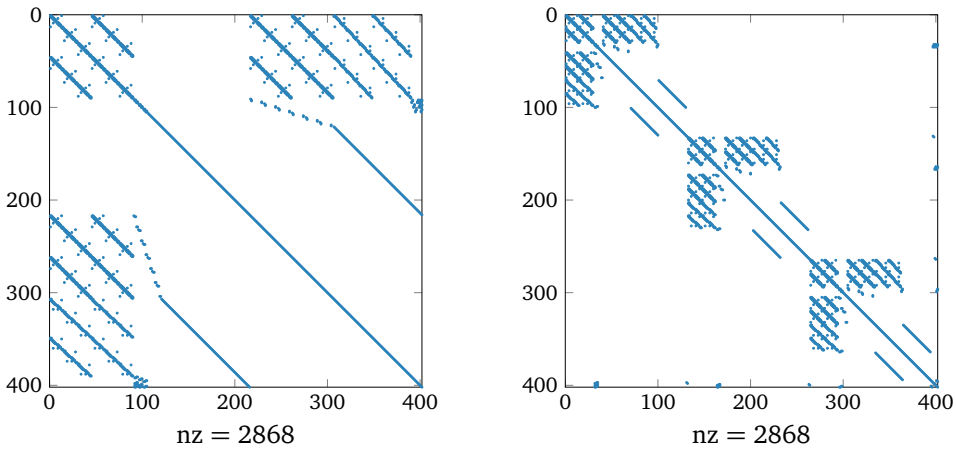


Figure 7.6. The KKT system for the standard MPOPF and its permutation.

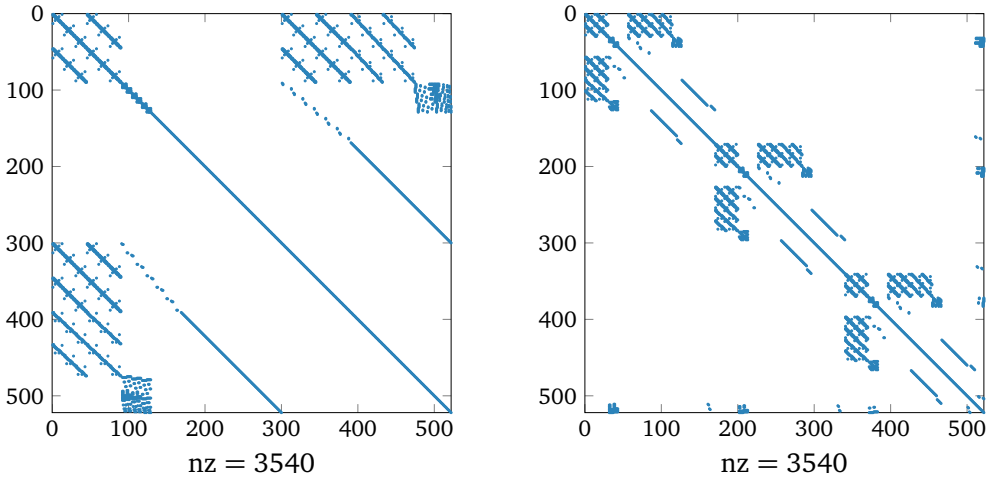


Figure 7.7. The KKT system considering the flexibility and its permutation.

Chapter 8

Numerical Results

In this chapter, the hardware and software setup used in the experiments is described, followed by introducing a concept of performance profiles used to present the results in a comprehensive manner. The numerical experiments follow, demonstrating the computational efficiency of the algorithms presented in this work.

The benchmarks, performed in section 8.2, are focused on the single period OPF problems without any contingency scenarios. The benchmarks are designed to investigate various parameters influencing the robustness and performance of the IP frameworks. The factors studied are the convergence tolerance, initial guess, and the OPF formulation. The first set of results is concluded by performing a comparison of the robustness and performance of various commercial and academic IP frameworks. The results are based on a preprint Kardos et al. [2018]; Kardos et al. [2020b], submitted to the ACM Transactions on Mathematical Software journal.

The impact of structure exploiting techniques of the KKT system solution on the overall computational time of the IP algorithm is demonstrated in section 8.2.5. It is shown that the distributed solution approach based on the Schur complement decomposition leads to the same SCOPF solution as the black-box solution methods based on factorization of the full KKT system. Time-to-solution and scaling of the parallel and distributed memory solution is reported for multiple large-scale SCOPF problems. Benefits of the improved sparse linear algebra and slack variables elimination in computation of the local Schur complement contributions are summarized. Furthermore, the bottlenecks of the decomposition scheme and their impact on performance are discussed. The results are based on the IEEE Transactions on Power Systems paper Kardoš et al. [2020]. The study is concluded by a series of performance and scaling benchmarks, pre-

sented in preprint Kardos et al. [2018], based on a structure exploiting solution for the MPOPF problems proposed in Kourounis et al. [2018]; Kardos et al. [2020b].

8.1 Benchmarking environment

A set of reference grids provided by the MATPOWER library Josz et al. [2016]; Zimmerman et al. [2011]; Fliscounakis et al. [2013a]; Birchfield et al. [2017] is used during the experiments. Various characteristics for a set of selected twenty-five benchmark cases are listed in Table 2.1 in chapter 2. In what follows, the grid name will be suffixed by the number of considered transmission line contingencies.

The IP frameworks and linear solvers used in this chapter were introduced in Chapter 6. The focus here is put on the KKT system solver used within the IP frameworks. The black-box solution of the KKT is used in IPOPT if used with one of the supported direct sparse solvers (e.g. PARDISO –which is considered as a default linear solver in later text, unless otherwise specified) or BELTISTOS-OPF. On the other hand, the parallel solution of the SCOPF problem based on the Schur complement decomposition, presented in algorithm 1 in chapter 7, is used within BELTISTOS-SC-N, where N denotes the number of parallel processes. We further distinguish between standard "backsolve" approach to local Schur complement computation, indicated by BELTISTOS-SC-N-std, while the approach where the local Schur complement contributions are obtained by incomplete factorization of the augmented matrix (7.19) is denoted as BELTISTOS-SC-N-aug. Multicore augmented factorization approach with T threads is further denoted as BELTISTOS-SC-N-aug-T. The two versions of the MPOPF structure exploiting algorithm proposed in Kourounis et al. [2018], based on the SC decomposition and its memory efficient version, are also distinguished. The notation is summarized in Table 8.1.

Computing environment

The large-scale scaling experiments are performed on multicore Cray XC40 compute nodes of "Piz Daint" at the Swiss National Supercomputing Centre CSCS - Swiss National Supercomputing Centre [2018], which consist of Intel Xeon E5-2695 v4 at 2.10 GHz with 18 cores and 64 GB RAM. In all experiments, Intel MKL implementation of BLAS and LAPACK (version 2017 update 3) was used. The GPU accelerators installed at compute nodes are NVIDIA Tesla P100 with 16

Table 8.1. Summary of the notation.

Method	Description
IPOPT	IPOPT with direct sparse solver PARDISO.
BELTISTOS-OPF	BELTISTOS with direct sparse solver PARDISO.
BELTISTOS-SC-N-std	BELTISTOS using Algorithm 1 with N processes.
BELTISTOS-SC-N-aug	Algorithm 1 with (7.19) using N processes.
BELTISTOS-SC-N-aug-T	Multicore execution with T threads.
BELTISTOS-MP	Algorithm proposed in Kourounis et al. [2018].
BELTISTOS-MEM	Algorithm proposed in Kourounis et al. [2018].

GB memory using CUDA Toolkit version 8.

The single period and multiperiod single-node simulations are performed on a workstation equipped with an Intel Xeon CPU E7-4880 v2 at 2.50 GHz and 1 TB memory, located at Technische Universität Braunschweig, Germany.

Performance profiles

Performance profiles are used in order to evaluate the quality of the different optimization methods for OPF problems. Performance profiles provide compact comparison of the benchmark problems using different optimization packages. These profiles were first proposed in Dolan and Moré [2002] for benchmarking optimization software and used, e.g., to evaluate the performance of various sparse direct linear solvers and optimizers Gould and Scott [2004]; Scott et al. [2006]; More and Wild [2009].

The profiles are generated by running the set of optimizers $\mathcal{M}$ on a set of OPF problems $\mathcal{S}$ and recording information of interest, e.g., time to solution or memory consumption. Let us assume that a power flow optimizer $m \in \mathcal{M}$ reports a statistic $\theta_{ms} \geq 0$ for the OPF problem $s \in \mathcal{S}$; smaller statistics θ_{ms} indicate better solution strategies. We can further define $\tilde{\theta}_s = \min_{m\in\mathcal{M}}\{\theta_{ms}\}$, which represents the best statistic for a given OPF problem s. Then for $\alpha \geq 1$ and each $m \in \mathcal{M}$ and $s \in \mathcal{S}$

$$k(\theta_{ms}, \tilde{\theta}_s, \alpha) = \begin{cases} 1 & \theta_{ms} \leq \alpha \cdot \tilde{\theta}_s, \\ 0 & \theta_{ms} > \alpha \cdot \tilde{\theta}_s. \end{cases} \tag{8.1}$$

is defined. The performance profile $p_m(\alpha)$ of the power flow optimizer m is then defined by

$$p_m(\alpha) = \frac{\sum_{s\in\mathcal{S}} k(\theta_{ms}, \tilde{\theta}_s, \alpha)}{|\mathcal{S}|}. \tag{8.2}$$

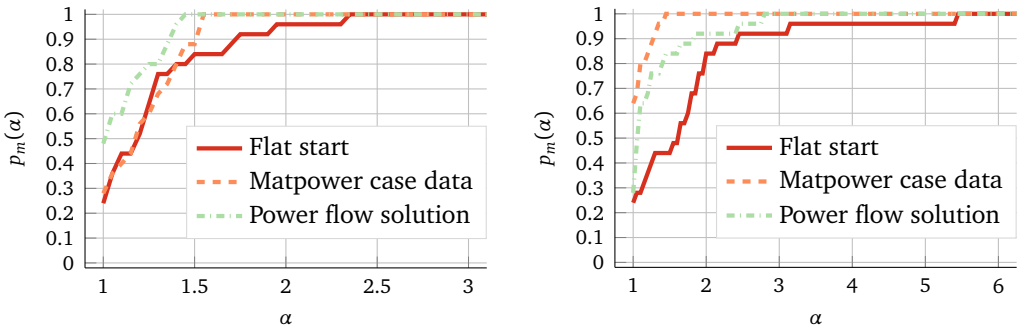
Thus, in these profiles, the value of $p_m(\alpha)$ indicates the fraction of all examples which can be solved within a factor of α of the time the best solver needed; e.g., $p_m(1)$ gives the fraction of which optimizer m is the most effective package and $p_m^* := \lim_{\alpha\to\infty} p_m(\alpha)$ indicates the fraction for which the algorithm succeeded. If we are just interested in the number of wins on $\mathcal{S}$, we need only compare the values of $p_m(1)$ for all the solvers $i \in \mathcal{M}$, but if we are interested in optimizers with a high probability of success on the set $\mathcal{S}$, we should choose those for which p_m^* is largest. Thereby, for a selected test set, performance profiles provide a very useful and convenient means of assessing the performance of optimizers relative to the best optimizer on each example from that set Gould and Scott [2016]. When commenting, e.g., on a performance profile presented in their paper, Dolan and Moré state that it "gives a clear indication" of the relative performance of each optimizer Dolan and Moré [2002] and one can determine which optimizer has the highest probability $p_m(f)$ of being within a factor f of the best optimizer for f in a chosen interval. In this paper performance profiles are used to compare various aspects of problem formulation, problem setup, and performance of several optimizers on sets of smooth power flow problems. Our results provide estimates for the best configuration of the problems and identification of the optimizer with the greatest performance benefits.

8.2 OPF problem solution

This section discusses various areas that have an impact on the convergence properties of the IP method on the nonlinear level, e.g., initial point selection and convergence tolerance. This can help power system engineers and other practitioners to make an informed decision when solving the AC OPF problems using an IP approach.

8.2.1 Choice of an initial point

The gradient-based optimization methods, such as IP methods, are sensitive to the starting point used as an initial guess usually provided by the user as a parameter to the optimization method. IP methods are known to suffer the lack of an efficient warm starting scheme which would enable the use of information from a previous solution of a similar problem Gondzio and Grothey [2008]. An advanced starting point which is close to the boundary of the feasible region, as is typical, might lead to blocking of the search direction. Therefore, it is required that the initial point is sufficiently inside the variable bounds. Note that if the ini-



(a) Overall time (b) Number of iterations

Figure 8.1. Performance profiles for the three initial guesses using BELTISTOS-OPF considering all benchmarks.

tial point does not strictly satisfy the variable bound constraints, IPOPT enforces these by shifting the corresponding entries of the initial guess that do not satisfy this criterion in order to be sufficiently inside the bounds. This is necessary in order to be able to evaluate the logarithmic barrier function (4.6), which is defined only for the points satisfying the bound constraints. On the other hand, the convergence from poor starting points is achieved by penalty merit functions to enforce progress toward the solution Wächter and Biegler [2006]. However, a wise choice of the initial guess may significantly reduce the amount of iterations to convergence. In context of the OPF problems, several heuristics may be devised for selection of an initial point, based on setting the variables inside their bounds or by selecting such point that satisfies the nonlinear equality constraints. Several strategies of the primal variables initialization are discussed next.

The computationally simplest strategy is to set bounded control variables to the midpoint of their allowed range, or close to a bound if bounded only from one side, i.e. so called “flat start”. This option is the default in MATPOWER and it does not satisfy any nonlinear constraints. The option “MATPOWER case data” uses the values of variables specified in the input MATPOWER case and the third option “power-flow solution” is the solution of the power flow equations for the given case initialized from the “MATPOWER case data”, which guarantees that the nonlinear power balance constraints are satisfied.

In order to evaluate the influence of the initial guess, the OPF problems are solved from three different initial guesses currently provided by the MATPOWER option `opf.start`. The performance profiles for the number of iterations and overall time of the OPF benchmarks starting from different points are presented in Figures 8.1a and 8.1b. The default OPF formulation with polar voltage coor-

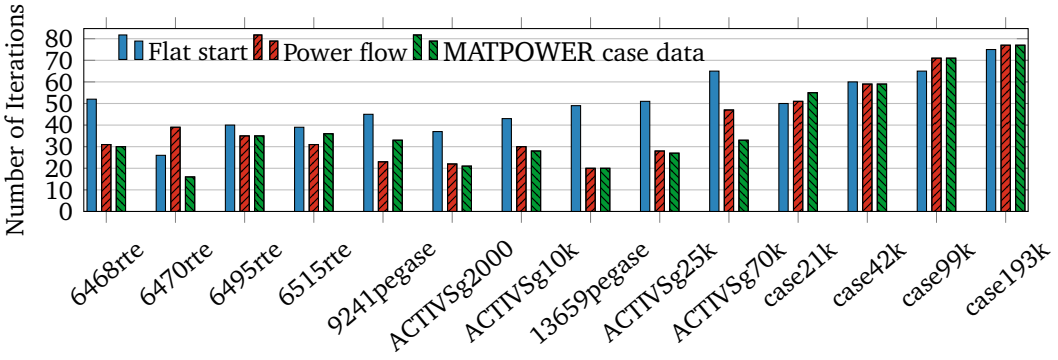


Figure 8.2. BELTISTOS-OPF iterations for each initial guess

Table 8.2. Number of solved benchmarks out of twenty-five test cases for different starting points. Gray background indicates use of the PARDISO solver.

Optimizer	Flat start	MATPOWER case data	Power flow solution
MIPS-MATLAB'\'	16	23	23
MIPS-PARDISO	16	23	24
IPOPT-PARDISO	23	25	25
IPOPT-MA57	21	22	21
BELTISTOS-OPF	25	25	25
FMINCON	18	21	20
KNITRO	20	23	24

dinates and power balance equations was considered. The results in Figures 8.1 and 8.2 were obtained using BELTISTOS-OPF, since it was the most successful optimizer, as demonstrated in Table 8.2. The best initial guesses in our set of benchmark cases were the options "MATPOWER case data" and "power-flow solution". Both optimizers BELTISTOS-OPF and IPOPT-PARDISO starting from these initial guesses solved all twenty-five cases requiring fewer iterations on average than optimizing from the "flat start". In what follows, the option "MATPOWER case data" is used in order to avoid the computational overhead of solving the power flow equations. The option "MATPOWER case data" assumes that the case is well constructed and contains high-quality data, which might not always be the case. The power flow solution would be a more appropriate choice in such situations.

The OPF problems are non-convex and different local minima may be reached from different starting points. We observed that the relative differences between

solutions obtained from different initial guesses or different optimizers were less than 10^{-5}, see Kardos et al. [2018] for more details.

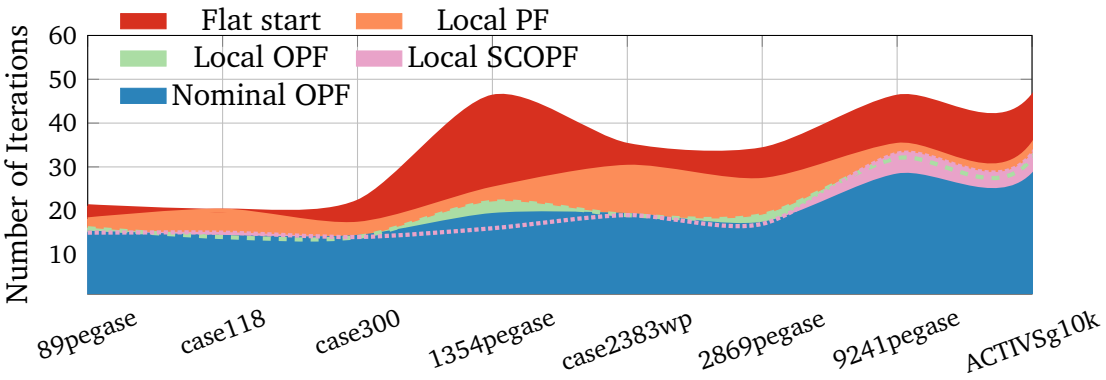


Figure 8.3. Number of iterations until convergence for different initial guesses. Considering SCOPF problem with 10 line contingency scenarios.

Similar initial point selection heuristics might be devised for the SCOPF problem. As demonstrated in Figure 8.3, the number of iterations until convergence may vary significantly with different starting points. We assume the following starting point choices: local solutions of the PF equations for each contingency scenario, solution of the OPF problem for the nominal case, solution of local OPF for each contingency scenario, and finally, the solution of local SCOPF problems considering the nominal case and the contingency of the given scenario. The results in Figure 8.3 indicate, the solution of the nominal OPF problem represents a good trade-off between the complexity and quality of the approximation. It provides convergence similar to more expensive approximations, such as local OPF or even local SCOPF with a single contingency, yet being cheaper to compute. The computation of the nominal OPF may be further improved by using the PF solution as an initial guess for the OPF problem computation and using relaxed convergence tolerances.

8.2.2 Convergence tolerance

Before proceeding with numerical experiments, the selection of the convergence tolerance is analyzed. The convergence tests implemented by optimizers vary in some details, e.g. scaling of the residual errors or type of the norms, making the user specified tolerance not equivalent amongst the optimizers. Second, various stopping tolerances were used in previous OPF studies, ranging from 10^{-3} to 10^{-8} Castillo and O'Neill [2013]. The selection of the convergence criteria has to

consider multiple factors, including required precision of the solution, numerical issues associated with very tight tolerances and computation time. For very tight tolerances, the linear systems become very ill-conditioned and the effect of the round-off error becomes pronounced, thus influencing the numerical stability.

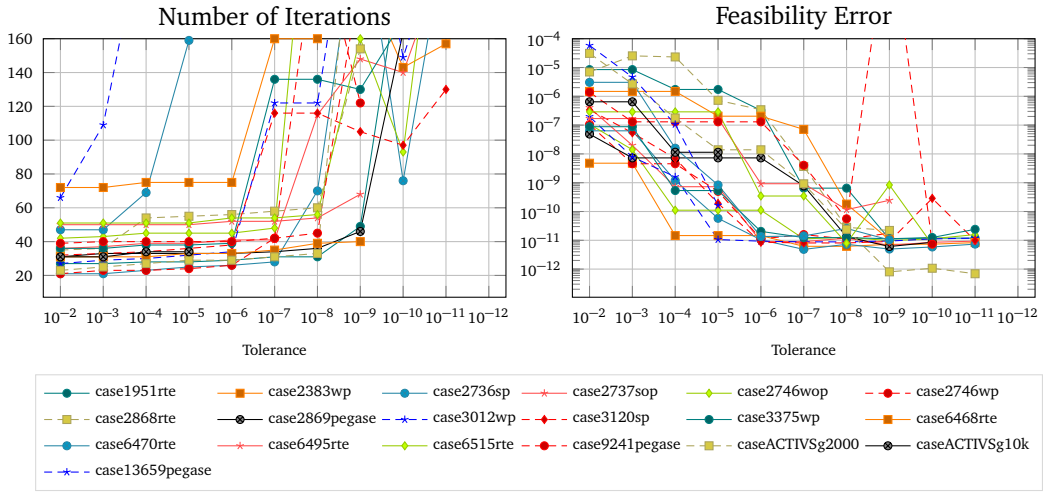


Figure 8.4. Convergence for different tolerances - IPOPT (considering also the restoration phase).

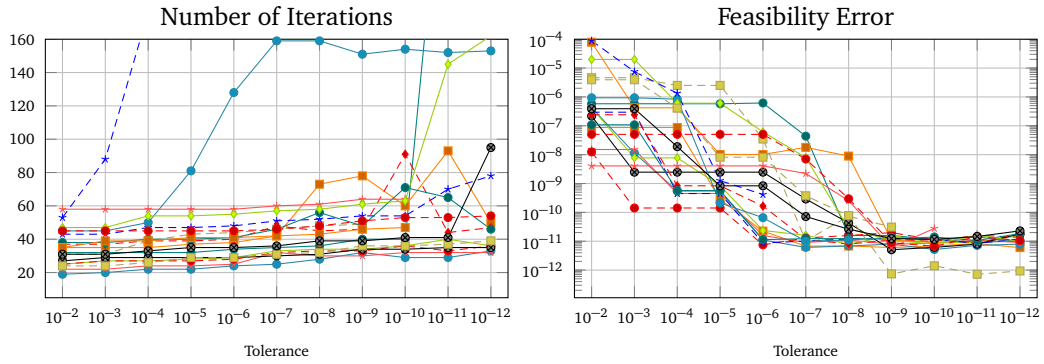


Figure 8.5. Convergence for different tolerances - BELTISTOS-OPF.

Many optimizers implement “acceptable” termination criteria, such that it will terminate before the desired convergence tolerance is met (e.g. there is no improvement in the objective function or feasibility norms over some specified number of iterations or the step becomes too small). This is useful in cases where the algorithm might not be able to achieve the desired level of accuracy. For the

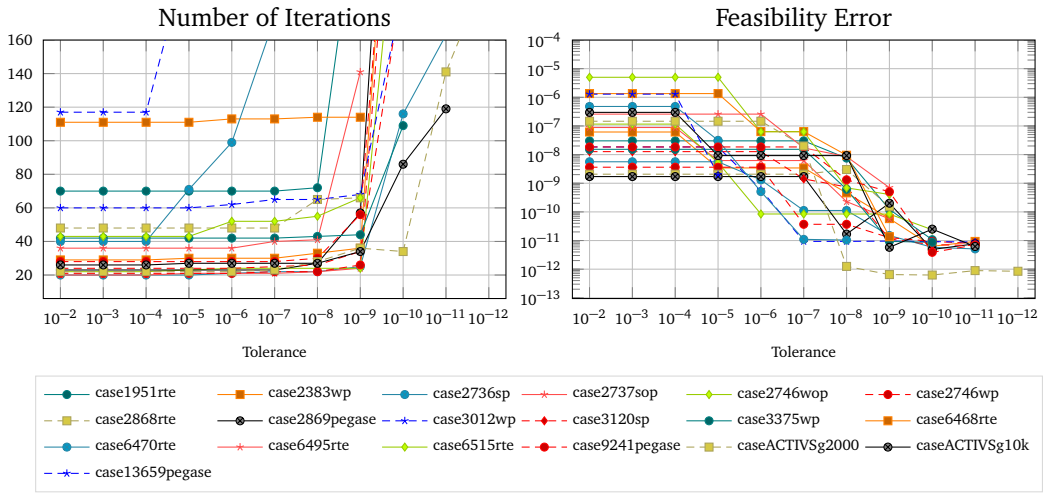


Figure 8.6. Convergence for different tolerances - KNITRO.

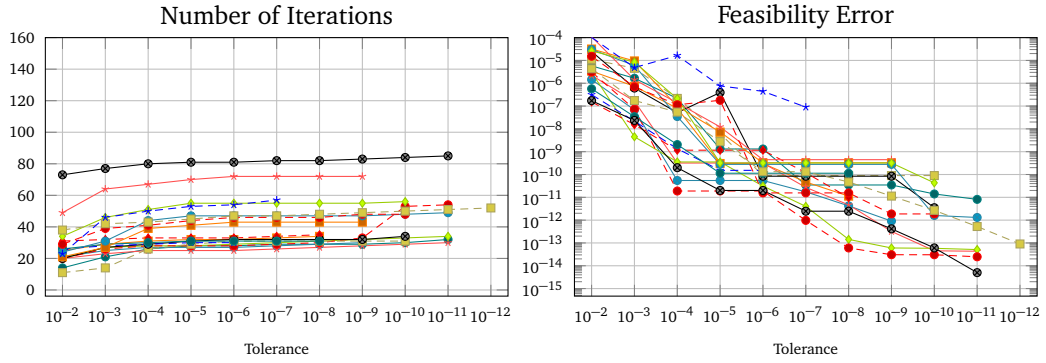


Figure 8.7. Convergence for different tolerances - MIPS.

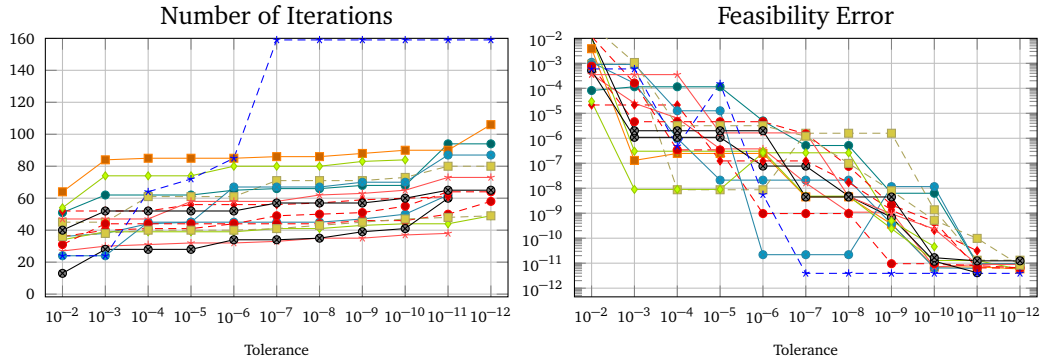


Figure 8.8. Convergence for different tolerances - FMINCON.

purpose of this section, such heuristics were disabled and optimizers are allowed to terminate only if the desired tolerance was reached. We observed that there is no significant improvement of the objective function value for tight tolerances. The absolute and relative errors of the objective function value for tolerances 10^{-4} and 10^{-9} are less than order of 10^{-3} ($/MWh) and 10^{-9}, respectively. In several cases, the objective function slightly increased for tighter tolerances, in similar orders of magnitude. The constraint violations (feasibility error) and number of iterations for different tolerances and optimizers are shown in Figures 8.4–8.8. The figures illustrate that even for a modest tolerance 10^{-4} the constraint violations are usually much smaller and in most cases tightening the tolerance involves only a few additional iterations in order of 1. For tight tolerances below 10^{-9}, optimizers start to be numerically unstable and the number of iterations starts to significantly grow or optimizers terminate with an error message. We thus use a modest value of the tolerance to focus on the performance of the optimizer, and isolate and issues related to the numerical errors.

8.2.3 OPF formulations

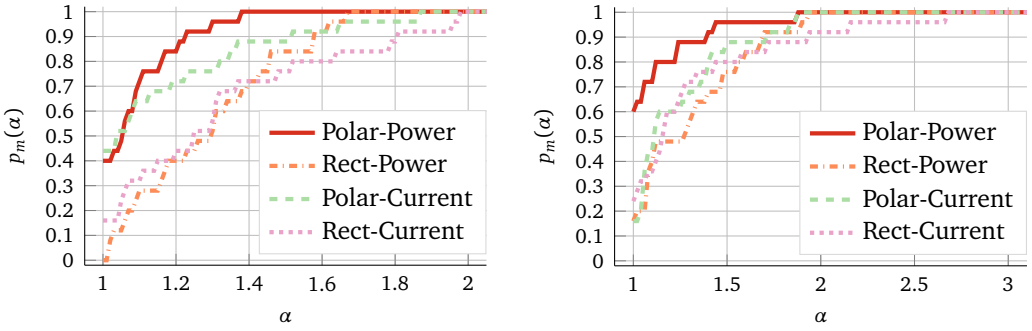
As introduced in chapter 3, different representations of the complex voltage variables or of the nodal balance equations can be used to formulate the OPF problem. Different OPF formulations will result in different constraint functions and feasible regions and consequently different constraint Jacobians with various properties, sparsity structure, and conditioning and which might pose distinct challenges to the optimizers and influence the their convergence. The corresponding MATPOWER options are `opf.v_cartesian`, specifying whether to use polar or rectangular voltage coordinates, and option `opf.current_balance`, which selects either a current or power balance formulation for AC OPF. Table 8.3 provides a summary of the optimizer success rate of solving different OPF formulations on the set of twentyfive benchmark cases. As can be seen in the table, robust optimizers such as BELTISTOS-OPF or IPOPT-MA57 (which, however, failed to solve most of the large-scale benchmarks) are marginally influenced by the choice of the formulation, while for the rest of the solvers the choice of the formulation can significantly influence whether the case can be successfully solved. While KNITRO and FMINCON perform better with rectangular voltage formulation, IPOPT-PARDISO and MIPS provide better success rate for power-based nodal equations.

The performance profiles for various OPF formulations presented in Figure 8.9 were obtained using the BELTISTOS-OPF optimizer, which successfully solved all benchmark cases with all possible OPF formulations. There is a significant dif-

Table 8.3. Number of solved benchmarks out of twenty-five test cases for different OPF formulations. Gray background indicates usage of PARDISO.

Optimizer	Polar Power	Polar Current	Rectangular Power	Rectangular Current
MIPS-MATLAB'\'	23	21	21	20
MIPS-PARDISO	23	20	23	17
IPOPT-PARDISO	25	20	25	22
IPOPT-MA57	22	21	21	21
BELTISTOS-OPF	25	25	25	25
FMINCON	21	18	25	25
KNITRO	23	24	25	24

ference between the polar and rectangular voltage formulations in terms of the overall time required until convergence. The polar formulations were observed to lead up to twice as fast solution times when compared to the rectangular voltage formulations.

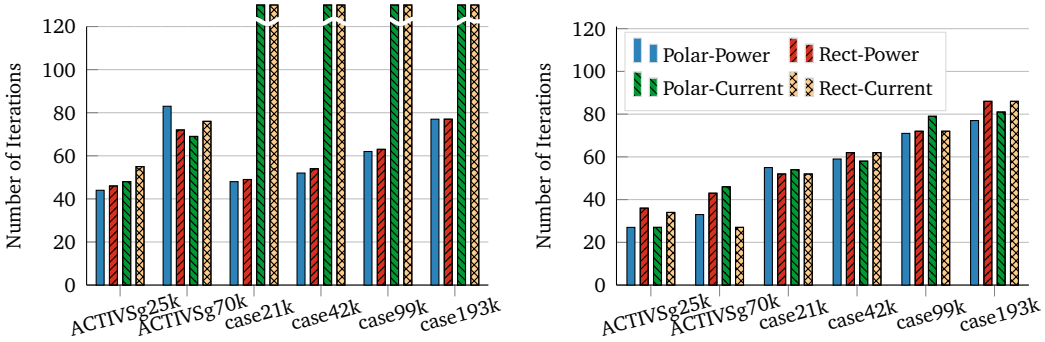


(a) Overall time (b) Number of iterations

Figure 8.9. BELTISTOS-OPF performance profiles for various OPF formulations starting from the MATPOWER case data.

When it comes to the other optimizers, IPOPT-MA57 also displays slightly slower convergence with rectangular voltage formulation for small and medium-sized benchmarks. Neither OPF formulation was solved for large-scale benchmarks case21k–case193k due to prohibitive time requirements. IPOPT-PARDISO does not seem to be influenced by voltage formulation, although it fails for large-scale benchmarks using current nodal balance equations, as can be observed in Figure 8.10a. The MIPS solver performs better with polar voltage coordinates, while the opposite is true for FMINCON and KNITRO, which successfully converge for more benchmarks with rectangular voltage coordinates, as presented in Ta-

ble 8.3. There is also a non-negligible influence of the nodal balance formulation. All optimizers prefer power based formulation of the nodal balance equations. The power balance was observed to be more robust and exhibit faster solution times in conjunction with both polar and rectangular voltage formulations.



(a) IPOPT-PARDISO (b) BELTISTOS-OPF

Figure 8.10. Optimizers' iteration count for large-scale benchmarks for various OPF formulations, starting from MATPOWER case data.

8.2.4 Optimization software

In this section, an emphasis is put on robustness, performance, and memory efficiency of the entire optimization software set on the set of large-scale benchmarks from Table 2.1. The "MATPOWER case data" has been used as an initial guess due to the superior performance as it was demonstrated in section 8.2.3. The OPF formulation with polar voltage and nodal power balance constraints was used for this set of benchmarks. The memory usage was collected using the Linux utility `time`, which reports maximum resident set size of the process during its lifetime. Table 8.6 shows the summary of timing results for the large-scale test cases and all optimizers. An extensive list of all results can be found in Kardos et al. [2018]. The performance profiles for each of the three aspects for the large-scale benchmarks for overall timing, iteration count, and memory consumption are shown in Figures 8.11, 8.12, and 8.13, respectively. The performance profiles clearly indicate that the BELTISTOS-OPF, IPOPT-PARDISO and the KNITRO optimizers converged to the optimal solution for all large-scale benchmark cases, followed by MIPS-PARDISO and FMINCON, which numerically failed for a single case. IPOPT-MA57 was not competitive, both in terms of robustness and performance, failing for the three large-scale cases and being slower up to a factor of three hundred or more, as can be seen in Table 8.6.

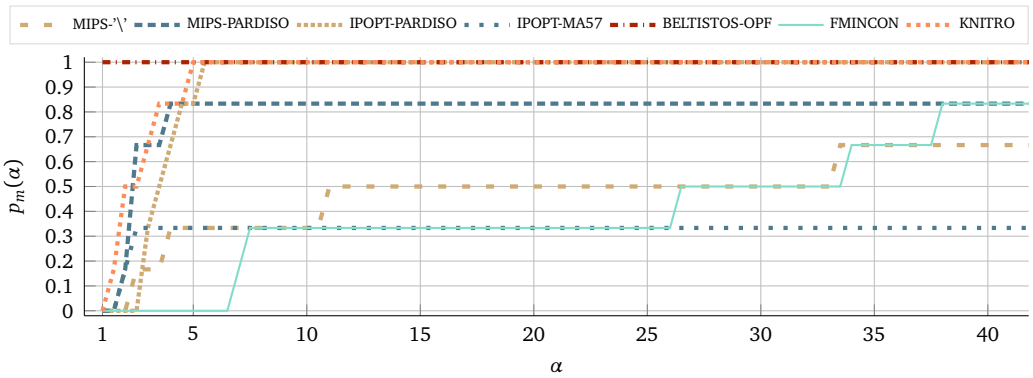


Figure 8.11. Overall time profile for large-scale benchmarks.

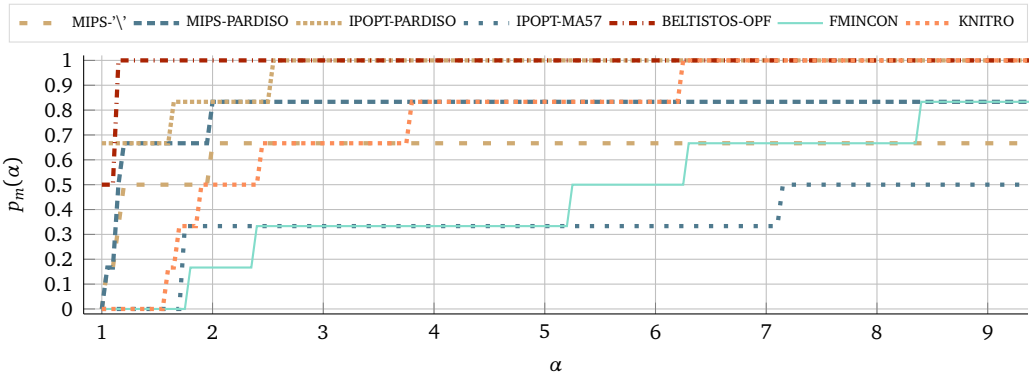


Figure 8.12. Iterations profile for large-scale benchmarks.

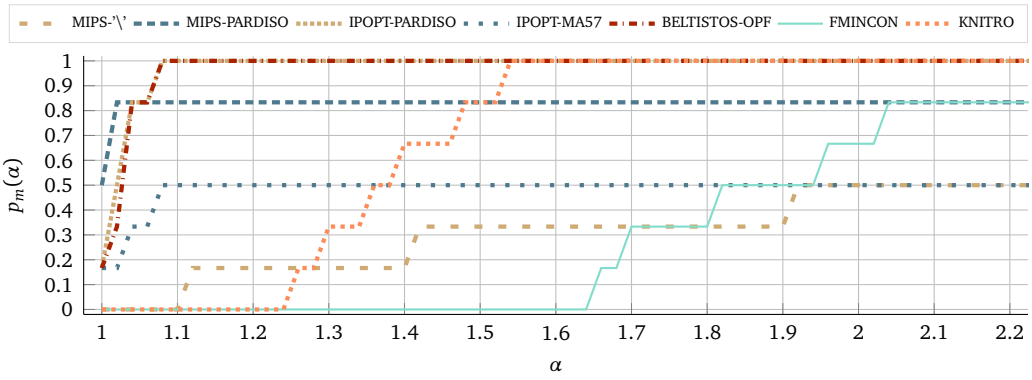


Figure 8.13. Memory efficiency profile for large-scale benchmarks.

Figure 8.11 reveals that BELTISTOS-OPF was the fastest optimizer for all large-scale cases. MIPS-PARDISO was slower by a factor of 4 when compared to the BELTISTOS-OPF, while KNITRO and IPOPT-PARDISO were up to 5.5 times slower. MIPS-'\' and FMINCON were slower up to a factor of 35 or 40, respectively. Concerning the number of iterations, BELTISTOS-OPF is also the best optimizer, while

IPOPT-PARDISO and MIPS-PARDISO perform up to 2.5 times more iterations. Regarding the memory requirements, MIPS-PARDISO is the most efficient optimizer for roughly 50% of the benchmark cases, very closely followed by BELTISTOS-OPF and IPOPT-PARDISO. MIPS with the default linear solver required up to 7 times more memory while solving the largest benchmark.

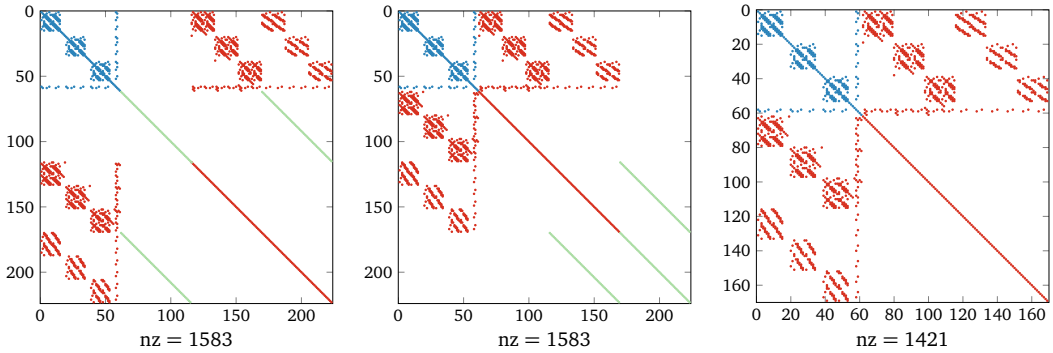


Figure 8.14. Structure of the SCOPF KKT system (4.16) with two contingencies, reordered according to (4.19), and, finally, the reduced KKT with the slacks removed (4.21).

8.2.5 Solution of the KKT linear system

Since a sparse linear system is solved at each iteration of an IP solver, the robustness and performance of the optimizer heavily depends on the sparse direct solver used for the computation of the search direction. The linear system solutions with high accuracy provide better search directions and thus help to accelerate convergence. Similarly, the efficient linear system solution strategy eliminates the major computational bottleneck of the IP method for large-scale problems.

The OPF problems and the associated linear systems are the 'simplest' among the three problems discussed in this document. Direct sparse solvers, based on the black-box solution strategy, may be used, to solve the problem. Still, significant differences are observed between the linear solvers.

The difference between the MIPS performance using the default backslash solver '\' compared to the PARDISO solver is visible for the large-scale cases in Table 8.4, where PARDISO outperforms the MATLAB backslash operator by more than a factor of 20. PARDISO also reduces the number of iterations until convergence for some cases due to special handling of ill-conditioned systems and thus

Table 8.4. Overall time (s) of MIPS with different linear solvers.

	Polar-Power		Polar-Current		Rectangular-Power		Rectangular-Current	
Benchmark	PARDISO	'\'	PARDISO	'\'	PARDISO	'\'	PARDISO	'\'
case1951rte	6.6	5.6	4.1	3.0	8.4	11.7	58.0	41.1
case2383wp	5.9	4.7	5.4	4.3	6.1	4.6	8.2	6.7
case2736sp	6.7	5.2	5.2	3.8	6.4	4.9	9.8	7.7
case2737sop	6.1	5.0	5.3	4.0	5.5	4.2	6.0	4.8
case2746wop	6.8	5.5	4.9	3.9	5.4	4.1	6.1	4.7
case2746wp	7.0	5.9	5.9	4.3	6.7	4.8	7.0	4.9
case2868rte	6.3	4.8	—	—	6.9	5.0	—	29.2
case2869pegase	22.9	26.9	17.2	14.3	18.8	17.0	—	—
case3012wp	7.2	6.3	6.1	5.5	7.2	6.3	7.6	6.6
case3120sp	27.9	30.0	10.3	9.3	12.4	11.4	11.8	10.3
case3375wp	8.3	7.6	8.6	7.7	8.6	8.0	9.8	9.0
case6468rte	19.3	17.5	27.7	88.1	—	—	102.7	99.8
case6470rte	20.8	19.5	—	—	107.5	—	—	—
case6495rte	33.0	29.8	42.3	43.0	95.0	73.4	—	73.4
case6515rte	—	30.0	24.1	20.6	—	—	80.6	68.6
case9241pegase	57.6	50.5	89.4	104.0	88.0	106.7	—	—
caseACTIVSg2000	6.1	6.0	16.6	16.4	4.9	4.9	15.5	16.2
caseACTIVSg10k	36.1	32.1	58.6	53.1	59.4	65.4	76.6	68.5
case13659pegase	42.2	37.2	—	—	115.6	—	—	—
ACTIVSg25k	119.0	109.4	118.4	104.7	164.4	156.7	208.9	192.0
ACTIVSg70k	—	—	627.7	1,075.5	879.1	1,079.4	1,476.0	1,322.4
case21k	140.6	171.0	126.7	142.3	141.6	183.9	142.3	171.0
case42k	518.2	2,639.5	—	2,672.0	438.3	2,897.3	541.5	2,601.0
case99k	1,612.3	25,936.4	4,077.3	18,011.5	1,524.2	19,458.4	—	18,472.1
case193k	4,484.0	—	—	—	3,530.6	81,198.0	—	—

more accurate descent directions are provided to the optimizer. A similar effect has been observed when using another optimizer such as IPOPT with two different linear solvers e.g. HSL MA57 and PARDISO, shown in Table 8.5. The MA57 was shown to have the best performance among the HSL solvers on an extensive set of test problems in Gould and Scott [2004]. Using IPOPT with PARDISO results in feasible computation time and successful convergence also for the large-scale networks, as summarized in the Table 8.6.

8.3 SCOPF problem solution

Solution of the KKT system is the most expensive part of the optimization process for large-scale optimal control problems in terms of computational work. Using a general direct sparse solver, that does not take into account the matrix structure, the solution of the KKT system consumes up to 99.99% of the overall run time, as demonstrated previously in Figure 7.1. The other contributions from function evaluations (e.g., objective, Hessian of Lagrangian, or Jacobian of constraints),

Table 8.5. Overall time (s) of IPOPT with different linear solvers.

	Polar-Power		Polar-Current		Rectangular-Power		Rectangular-Current	
Benchmark	PARDISO	MA57	PARDISO	MA57	PARDISO	MA57	PARDISO	MA57
case1951rte	7.16	4.77	3.62	2.61	5.31	2.56	3.60	2.58
case2383wp	9.66	5.20	5.97	3.90	12.81	4.21	13.95	3.95
case2736sp	6.54	3.28	4.00	2.15	3.61	2.54	3.75	2.28
case2737sop	6.09	3.84	3.94	1.80	3.77	2.25	4.03	2.20
case2746wop	5.43	3.08	3.32	1.50	2.89	3.08	3.16	2.96
case2746wp	7.46	3.21	3.59	1.67	4.20	2.90	3.70	2.87
case2868rte	9.21	4.48	16.97	4.38	12.08	4.21	13.62	4.05
case2869pegase	10.37	4.94	9.10	3.35	12.65	4.76	32.17	3.83
case3012wp	12.95	5.67	6.57	3.97	15.04	4.57	8.92	4.66
case3120sp	10.83	5.53	9.78	4.14	16.84	5.96	19.69	4.92
case3375wp	13.06	6.28	—	4.39	12.42	4.67	205.81	4.52
case6468rte	12.23	7.91	8.91	5.26	11.52	9.08	10.36	11.37
case6470rte	41.56	10.96	50.25	6.92	36.67	10.66	54.12	11.42
case6495rte	21.73	12.41	10.39	8.75	15.52	184.76	15.51	162.49
case6515rte	16.85	11.16	11.67	8.10	12.15	118.26	11.39	51.35
case9241pegase	45.79	21.78	98.38	14.05	49.23	21.55	179.06	16.33
case_ACTIVSg2000	7.45	3.70	4.66	2.60	5.69	3.25	5.22	3.05
case_ACTIVSg10k	26.64	16.21	19.06	11.60	18.60	12.71	17.46	14.63
case_ACTIVSg25k	83.68	62.12	63.04	33.39	67.87	49.59	94.37	51.31
case13659pegase	266.01	30.32	259.19	15.69	400.67	27.97	559.03	22.44
case_ACTIVSg70k	491.83	198.12	309.33	149.67	358.52	330.07	483.38	292.97
case21k	262.59	—	—	—	166.43	—	—	—
case42k	747.87	75,386.85	—	—	621.89	—	—	—
case99k	3,101.47	—	—	—	3,080.08	—	—	—
case193k	10,105.87	—	—	—	12,489.93	—	—	—

Table 8.6. Overall time (s) for large-scale benchmarks (Polar-Power form.)

Benchmark	MIPS-'\'	MIPS-PARDISO	IPOPT-PARDISO	IPOPT-MA57	BELT.-OPF	FMINCON	KNITRO
ACTIVSg25k	109.4	119.0	83.7	62.1	30.3	210.1	50.2
ACTIVSg70k	—	—	491.8	198.1	111.1	786.1	167.6
case21k	171.0	140.6	262.6	—	81.4	—	219.8
case42k	2,639.5	518.2	747.9	75,386.9	251.1	8,428.3	1,162.0
case99k	25,936.4	1,612.3	3,101.5	—	782.4	20,554.6	2,558.9
case193k	—	4,484.0	10,105.9	—	1,854.4	70,150.9	2,654.4

vector updates, or convergence criterion evaluation inside IP method, are orders of magnitude smaller. Reduction in the KKT solution time therefore significantly improves the overall run time.

8.3.1 Impact of the slack variables elimination

Figure 8.14 illustrates the symmetric KKT structure of the SCOPF problem for a simple power grid, together with the reduced variant, where the slack variables are eliminated according to the discussion in section 5.2.1. The elimina-

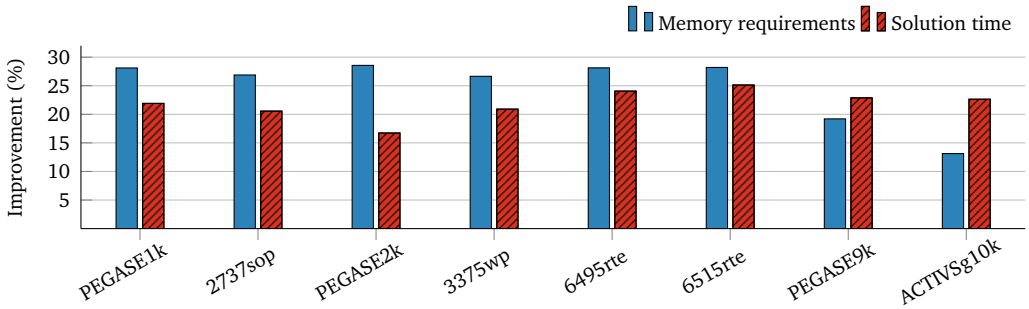


Figure 8.15. Improvement rate considering elimination of the slack variables.

tion threshold $\tau = 10^{-8}$ was used. Realistic power grids are significantly larger and contain proportionally more nonzero entries, but the structure remains very similar.

The expected benefits of solving the reduced KKT system compared to the original system are savings both in terms of memory requirements for storing the sparse $\boldsymbol{L}$ factor of the $\boldsymbol{LDL}^\intercal$ factorization of the symmetric indefinite system, and possibly faster factorization and solution times due to a smaller number of required floating point operations. The numerical evaluation of the benefits of solving the reduced system are summarized in Figure 8.15.

The elimination of the slack variables from the KKT system reduces its dimension by approximately 30% with 13% fewer nonzeros in the KKT system and up to 12% fewer nonzeros in the $\boldsymbol{L}$ factor, resulting in up to 28% memory savings for each of the diagonal blocks. The majority of the time in per-block computation is spent in factorization of the augmented matrix (7.19). Factorization time of the reduced system is improved up to 25% and the overall time of the local computation in Algorithm 1 (steps 2, 8, and 16) is improved by up to 24%. Considering that the assembly of the local SC is the main bottleneck and available memory on the compute nodes is limited, the achieved memory savings and time gain is significant, leading to more efficient and scalable code. We note that the benefit of solving the reduced system strongly depends on the quality of fill-in minimization reordering of the linear system performed during the factorization phase of the $\boldsymbol{A}_i$ blocks. For matrices which are well-conditioned, the fill-in of the elimination is similar for both the standard KKT system (4.16) and its reduced variant (4.21). This was observed for the PEGASE 13,659 bus case. However, ill-conditioned matrices will have a much better fill-in reduction and higher speedups between 20% - 30% were observed for all other networks. Please note that matrices from IP methods are typically very ill-conditioned.

The comparison of different solution algorithms is not valid unless all of them

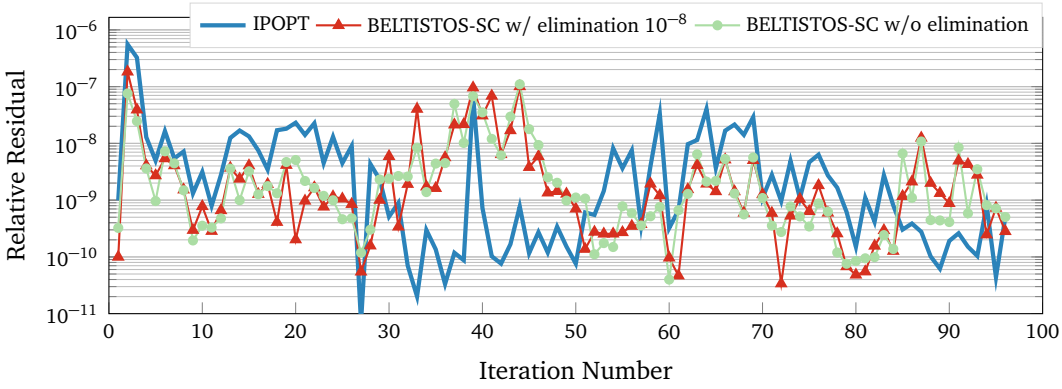


Figure 8.16. Residuals of the KKT systems for PEGASE1354-100 case using different solution approaches – the IPOPT solver and the BELTISTOS-SC-aug with and without elimination of the slack variables.

make the optimization algorithm converge to the same optimal solution. The Schur complement based solution strategy from Algorithm 1 is just a different way of solving the KKT linear system. Thus theoretically, it should not change the convergence properties of the IP method. Nevertheless, due to the floating point arithmetics, the round-off errors may introduce significant contribution so that at some point the IP software package used with a different linear system solution strategy might take different trajectory, but it was observed only for very ill-conditioned linear system. The residuals for the KKT system solutions arising in PEGASE1354-100 benchmark are shown in Figure 8.16. In order to improve robustness of the implementation, it is very important to use the iterative refinement in order to obtain highly accurate solutions. In IPOPT algorithm, the iterative refinement (IR) is computed for the original, non-reduced, KKT system (4.15). The Figure 8.16 presents residual of the first solution only, since the number of IR might vary between the two approaches.

Figure 8.17 shows the convergence trajectory of the objective function and infeasibility norms for the PEGASE1354-100 benchmark. Similar information is shown for IEEE188-159 benchmark in Figure 8.18. Both figures are demonstrating the convergence for the direct factorization approach, as implemented in PARDISO linear solver, and distributed BELTISTOS-SC-4-aug approach (using 4 parallel processes). The two approaches take the same trajectories during the optimization process and converge to the same solution. Note the sudden changes in the infeasibility norms in Figure 8.17 after iteration 60. These are the result of switching from one barrier sub-problem to the next one with decreased barrier

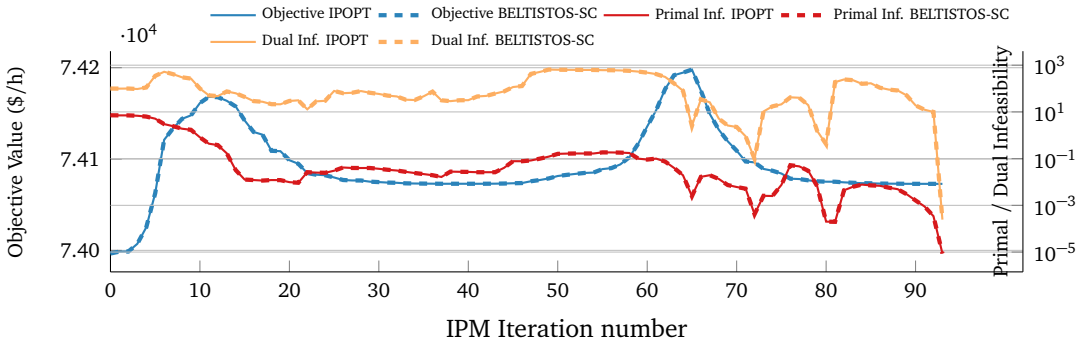


Figure 8.17. Convergence trajectories for PEGASE1354-100 benchmark over the IP iterations (both approaches overlap).

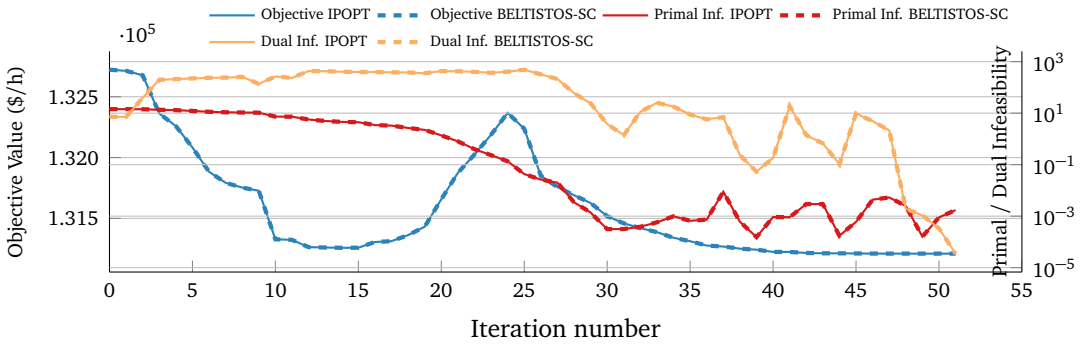
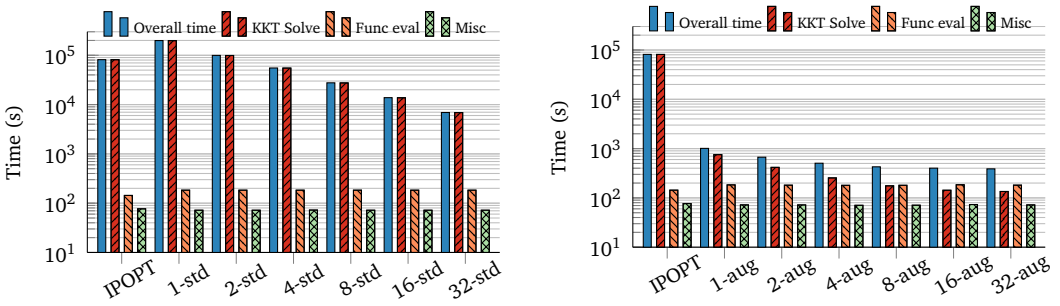


Figure 8.18. Convergence trajectories for IEEE118-159 benchmark over the IP iterations (both approaches overlap).

parameter μ. In both figures it is interesting to observe that even though the optimization problem is formulated as a minimization, the objective function value is actually increasing in some iterations. This is a result of the fact that the optimization process is driven by satisfying the constraints with higher priority than improving the objective function value.

Figure 8.19 demonstrates that by using the parallel BELTISTOS-SC solver with an increasing number of processes, the KKT solution time decreases and thus the overall run-time of the optimization is reduced accordingly. The performance of the BELTISTOS-SC-std approach is shown in Figure 8.19a. The single-process performance is several times slower than the IPOPT approach, but with increasing number of parallel processes the overall time scales down proportionally and outperforms the IPOPT approach. On the other hand, the single-process performance of the BELTISTOS-SC-aug approach, shown in Figure 8.19b, is almost two



(a) Standard "backsolve" (b) Incomplete factorization approach

Figure 8.19. Overview of the IP components for PEGASE9241-256 benchmark, where the IPOPT solver is compared to the parallel BELTISTOS-SC-N-std and BELTISTOS-SC-N-aug algorithms.

orders of magnitude faster than the IPOPT approach and it is possible to further decrease the solution time with additional parallel processes.

Depending on the problem size, there is an upper limit to the number of parallel processes that are able to efficiently utilize available computing resources, since there are serial components in the IP algorithm. Such a situation can be observed in Figure 8.19b, where the function evaluations and other IPOPT internal computations became as significant as the solution phase for 8 and more parallel processes. The effect of parallelization after this point will be deteriorated by these serial components. Another component influencing the parallel efficiency is assembly of the KKT system, which is performed solely at the master process and individual blocks need to be distributed to the available processes. The distribution step involves inherent interprocess communication and thus deteriorates the performance with increasing number of processes. The communication overhead becomes significant, especially if the workload per node is small, e.g., only a couple of scenarios per node.

8.3.2 GPU acceleration

The bottleneck of the parallel Schur algorithm is the dense linear algebra associated with the global part of the solution in (7.12), which is solved exclusively by a single process. The dense Schur system needs to be factorized and a forward-backward substitution needs to be performed with the computed factors, corresponding to steps 12 and 13 in Algorithm 1, respectively. GPU accelerated cuSolveDN library for dense linear systems is used to reduce the time of this serial component of the algorithm in order to increase parallel efficiency and further

decrease overall time to solution. Table 8.7 summarizes the dense linear algebra

Table 8.7. Solution and memory transfer times of the dense Schur complement systems of size N.

		CPU	GPU		
Benchmark	**N**	**Solution (ms)**	**Solution (ms)**	**Memory (ms)**	**Speedup**
PEGASE1354	518	19.7	13.1	0.8	1.41
PEGASE2869	1,018	131.1	21.2	1.4	5.80
PEGASE9241	2,888	416.3	50.7	7.4	7.16
ACTIVSg10k	3,663	925.9	71.5	11.3	11.18
PEGASE13659	8,182	8,641.0	231.0	54.0	30.33

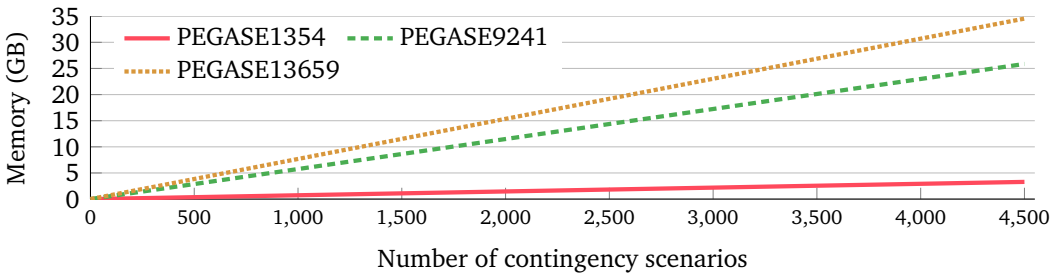


Figure 8.20. Memory required for storing the upper-triangular KKT system.

solution time, using the GPU accelerated library compared to the Intel MKL LAPACK implementation for CPU. N denotes size of the dense Schur complement, and the GPU related memory transfers are presented separately from the solution time. As a remark, the size of the dense Schur system is constant, independent on the number of considered contingencies for each grid. We observed speedup of up to 30× using GPU for the largest grid, considering also the penalty of the memory transfers between CPU and GPU. The GPU acceleration is vital, since the improved sparse linear algebra reduces computational time to such extent that the dense computation times start to present significant contribution to the overall run time, especially for the large-scale power networks.

8.3.3 Performance case study

Strong scaling efficiency of the BELTISTOS-SC-aug solver is measured on “Piz Daint” computer, using an increasing number of compute cores and distributed

memory compute nodes. The number of considered contingency scenarios was

Table 8.8. Properties of the large-scale benchmarks.

Benchmark	N_c	**Variables**	**Constraints**	**KKT size**
PEGASE1354	2,048	$5.55 \cdot 10^6$	$1.37 \cdot 10^7$	$2.74 \cdot 10^7$
PEGASE1354	4,096	$1.11 \cdot 10^7$	$2.74 \cdot 10^7$	$5.48 \cdot 10^7$
PEGASE9241	256	$4.73 \cdot 10^6$	$1.29 \cdot 10^7$	$2.47 \cdot 10^7$
PEGASE9241	512	$9.47 \cdot 10^6$	$2.59 \cdot 10^7$	$5.18 \cdot 10^7$
PEGASE13659	128	$3.51 \cdot 10^6$	$8.74 \cdot 10^6$	$1.75 \cdot 10^7$

selected such that the problem does not exceed the memory limit of 64 GB available at each compute node since the current implementation requires assembly of the linear system on a single node with subsequent distribution of its parts to the remaining processes. This memory limit turned out to be very restrictive since it did not allow us to solve arbitrarily large problems and demonstrate scaling up to very large number of compute cores. The largest instance of the solved problem contained up to $1.11 \cdot 10^7$ variables and $2.74 \cdot 10^7$ constraints, resulting in the KKT system of size $5.48 \cdot 10^7$. The properties of the large-scale benchmark problems are summarized in Table 8.8. Number of considered contingencies N_c, overall number of variables, constraints and size of the resulting KKT system are presented. Memory requirements for storing the symmetric upper-triangular part of the KKT system for the benchmark cases with growing number of contingencies are illustrated in Figure 8.20.

Figure 8.21 shows the average wall time of the individual phases of the Algorithm 1, indicating also the ideal strong scaling. The algorithmic phases shown in the Figure are: the initialization phase (*init*), assembly of the Schur complement in steps 2–6 (*SC Assembly*), GPU accelerated Schur complement solution in steps 12–13 (*GPU Computation*), and solutions of the local parts of the system in steps 14–16 (*Local Solve*). The benchmarks were run with a single MPI process per node and each process contained 16 threads.

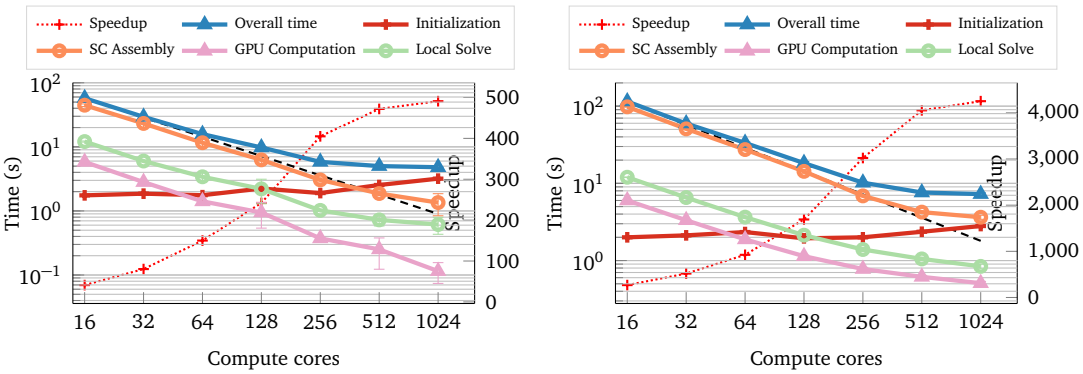
The BELTISTOS-SC-aug approach using a single process outperforms the sequential direct factorization by a factor of up to 40× and 270× for the two benchmarks, respectively. The observed speedup increased up to 500× and 4200×, respectively, with an increasing number of distributed memory nodes. The significant speedup difference between the two benchmark cases may be attributed to nonsatisfactory performance of the serial direct factorization for the PEGASE9241-

512 benchmark due to extensive fill-in. Although both KKT systems have similar size and number of nonzero entries, there is a factor of 13× difference between the factorization time of the two. The speedups are expected to grow significantly for larger problems, but due to the restrictive memory at the "Piz Daint" machine, it is not possible to demonstrate. The BELTISTOS-SC-aug solution time scales reasonably up to 512 cores at 32 compute nodes, which in terms of workload translates to 16 scenarios per node of PEGASE9241 or 128 scenarios per node of PEGASE1354 benchmark. At this point, the initialization phase requires approximately the same time as the most expensive part of the algorithm, which is the computation of the local contributions to the Schur complement. Thus, the initialization phase is the inherent bottleneck of this approach and determines the scalability limits. The acceleration and efficiency of the structure-exploiting algorithm stems from the reduced complexity associated with the factorization of the smaller sparse diagonal blocks instead of the original SCOPF KKT system (4.16) or its reduced variant (4.21) after it was permuted to the arrowhead structure (7.12). For sufficiently large power grids, however, the dense Schur complement system might become very large, and dominate the overall processing time in steps 12 and 13. Hardware accelerators such as GPUSs might be deployed to address the computational complexity of the dense linear algebra. Otherwise, the dimensions of the dense systems remain feasible for the majority of power grids, since the dimensions depend only on the power grid properties, not on the number of contingency scenarios.

The implementation of fully distributed and parallel IP framework was not attempted, since this was accomplished before, e.g. by PIPS Petra [2014]; Chiang et al. [2014]. The purpose was to show that more efficient implementation is possible by implementing sparse linear algebra routines which further improve intra-node performance compared to the previous work. The proposed method employs one additional Schur complement, chosen so that the sparsity of the KKT matrix is maintained. Consequently, the size of the linear system decreases and this allows for memory savings and increased computational performance.

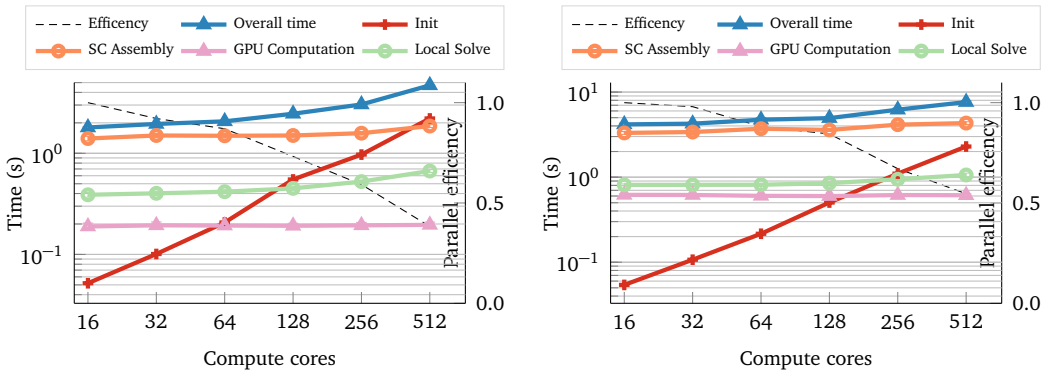
8.3.4 Swiss grid case study

The parallel SCOPF problem solution is demonstrated on a Swiss transmission system, consisting of 231 generator nodes (out of which 149 are domestic nodes) and 439 transmission lines (220 and 380 kV). The architecture of the power grid is illustrated in Figure 8.23. The SC decomposition approach, implemented in BELTISTOS-SC, is compared to the single core IPOPT in Figure 8.24. Results for two benchmarks are shown, considering 290 and 4,096 contingency scenarios.



(a) PEGASE1354-4096 (b) PEGASE9241-512

Figure 8.21. Strong scaling of the parallel BELTISTOS-SC-N-aug-16 solver and its components on two SCOPF benchmarks. Additionally, the speedup with respect to the serial IPOPT approach is shown.



(a) PEGASE1354, 128 cont. per node (b) PEGASE9241, 16 cont. per node

Figure 8.22. Weak scaling of the parallel BELTISTOS-SC-N-aug-16 solver and its components on two SCOPF benchmarks. Additionally, the parallel efficiency is shown.

For the latter benchmark, the single process BELTISTOS-SC speedup with respect to the IPOPT was 337-fold, while the speedup of the parallel run using 16 nodes with 256 cores overall was over 3500-fold. Similar speedups were observed as in the previous pan-European benchmarks.

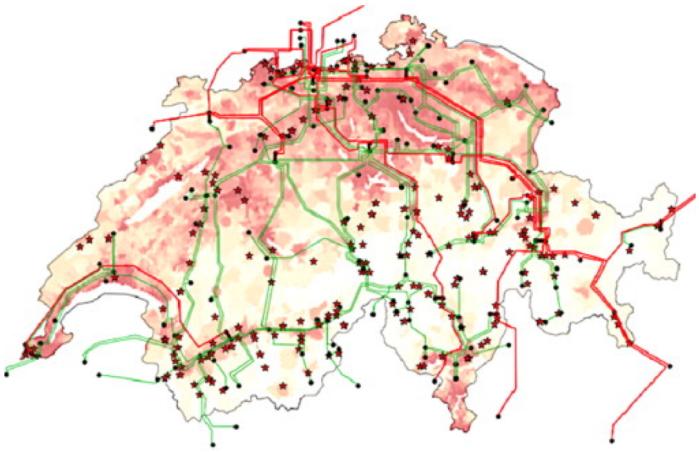
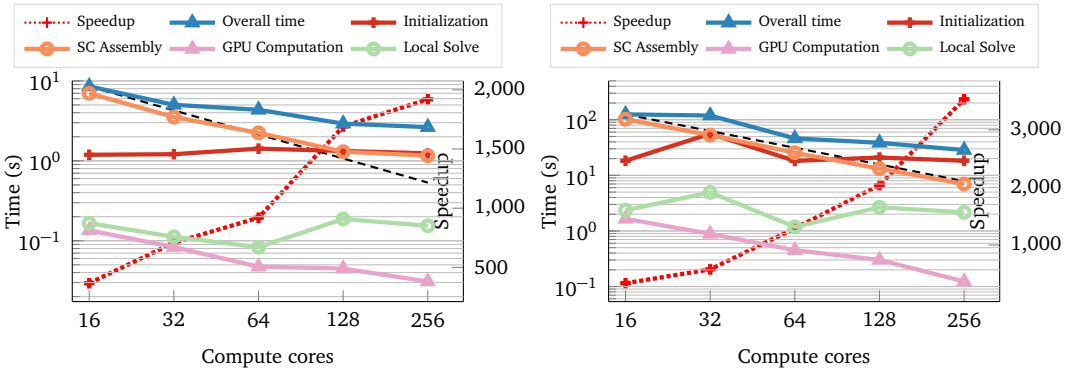


Figure 8.23. Switzerland's transmission system (image Singh et al. [2014]).



(a) Swiss grid — 290 contingencies. (b) Swiss grid — 4,096 contingencies.

Figure 8.24. Strong scaling of the parallel BELTISTOS-SC-N-aug-16 solver on two Swiss grid SCOPF benchmarks, including the speedup with respect to the serial IPOPT approach.

8.4 MPOPF problem solution

The computational complexity of the MPOPF problems grows quickly with increasing number of time periods. The generic optimization packages and black-box linear solvers become infeasible for large MPOPF problems due to the extensive memory requirements and solution times. In this section, BELTISTOS-OPF represents the black-box approaches. It is assumed that other black-box optimization packages will perform in a similar order of magnitude. The structure exploiting algorithm is represented by BELTISTOS-MP and its memory efficient

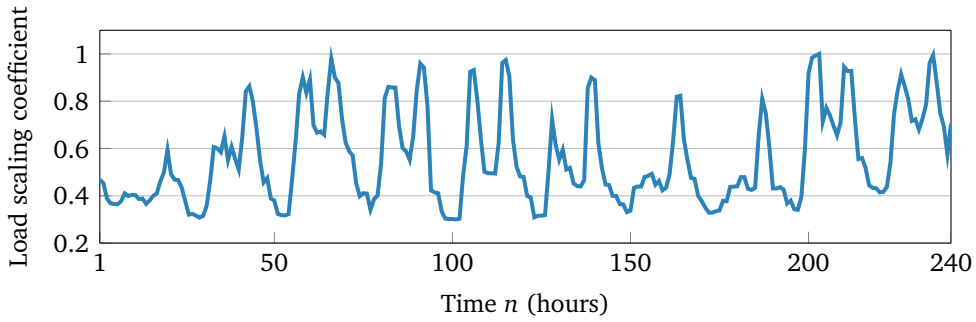


Figure 8.25. Hourly load scaling coefficient over a 10 day period.

variant BELTISTOS-MEM. Furthermore, due to the similarity of the BELTISTOS-OPF and BELTISTOS-MP (underlying data structures, BLAS implementation, linear solver, etc.), the comparison will most accurately represent the performance gap between the black-box and the structure exploiting approaches.

The benchmarks in this section focus on performance related to the KKT solution (factorization and solution phases) since these represent the bottleneck of the IP method for large-scale MPOPF problems. We do not consider common factors for both IP methods, including computation such as assembly of the MPOPF case or symbolic factorization routines, which are performed only once, or other routines (vector updates, evaluations of stopping criteria, etc.) performed in each IP iteration.

For all simulations, the length of the time period δt is set to one hour. The load scaling profile, shown in Figure 8.25, was used for all benchmarks to generate a time dependent load as a multiplicator of the nominal loads given in the MATPOWER case files. The profile is based on a load pattern for the Swiss canton Ticino.[1] The experiments in this section are run using only a single core.

The benchmark cases include one small and two medium size power grid networks, listed in table 8.9. The table illustrates MPOPF problem sizes for different numbers of time periods and numbers of storage devices.

8.4.1 Number of time periods and storage devices

We investigate the performance of BELTISTOS-OPF and BELTISTOS-MP for increasing problem sizes, changing the number of time periods and storage devices. The investigation is focused on the bottleneck of the IP method, which is the solution

[1]Data available at `https://www.swissgrid.ch/en/home/operation/grid-data/generation.html`.

Table 8.9. Selected MPOPF benchmark statistics including the number of time periods N, storage devices N_S, buses n_b, generators n_g, transmission lines n_l, and nonlinear equality and inequality constraints, as well as the number of linear constraints $|A(x)|$.

Benchmark	N	N_S	n_b	n_g	n_l	$\|x\|$	$\|g(x)\|$	$\|h(x)\|$	$\|A(x)\|$
case118	600	10	118	54	186	$2.30{\cdot}10^5$	$1.42{\cdot}10^5$	$2.23{\cdot}10^5$	$6.0{\cdot}10^3$
1369pegase	600	10	1,354	260	1,991	$1.96{\cdot}10^6$	$1.62{\cdot}10^6$	$2.39{\cdot}10^6$	$6.0{\cdot}10^3$
2869pegase	600	10	2,869	510	4,582	$4.08{\cdot}10^6$	$3.44{\cdot}10^6$	$5.50{\cdot}10^6$	$6.0{\cdot}10^3$
case118	1,200	10	118	54	186	$4.61{\cdot}10^5$	$2.83{\cdot}10^5$	$4.46{\cdot}10^5$	$1.2{\cdot}10^4$
1369pegase	1,200	10	1,354	260	1,991	$3.92{\cdot}10^6$	$3.25{\cdot}10^6$	$4.78{\cdot}10^6$	$1.2{\cdot}10^4$
2869pegase	1,200	10	2,869	510	4,582	$8.16{\cdot}10^6$	$6.89{\cdot}10^6$	$1.10{\cdot}10^7$	$1.2{\cdot}10^4$
case118	4,800	10	118	54	186	$1.84{\cdot}10^6$	$1.13{\cdot}10^6$	$1.79{\cdot}10^6$	$4.8{\cdot}10^4$
1369pegase	4,800	10	1,354	260	1,991	$1.57{\cdot}10^7$	$1.30{\cdot}10^7$	$1.91{\cdot}10^7$	$4.8{\cdot}10^4$
2869pegase	4,800	10	2,869	510	4,582	$3.26{\cdot}10^7$	$2.75{\cdot}10^7$	$4.40{\cdot}10^7$	$4.8{\cdot}10^4$

of the KKT system in each IP iteration. Since the KKT matrix changes numerically but not structurally at every iteration it is therefore reasonable to separate the symbolic factorization phase that determines a sparsity preserving pivot order from the numerical factorization phase. The symbolic factorization phase only needs to be done once at the beginning of the IP algorithm. In general, the matrix is indefinite, thus the pivot order strongly depends on the numerical values of the pivots, preventing the separation of symbolic and numerical factorization. However the augmented system matrix can be transformed into a quasi-definite matrix by use of a diagonal regularization Wächter and Biegler [2006], such that negligibly small terms are used for all acceptable pivots and the stronger regularization terms are used whenever a dangerously small pivot candidate appears. The separation of the symbolic factorization phase is thus valid.

The average time of the KKT system solution, consisting of the numerical factorization and forward-backward substitutions (excluding the symbolic factorization) is shown in Figure 8.26 for increasing number of time periods. Different power grids are shown, each containing ten storage devices. It is evident that BELTISTOS-MP outperforms the black-box solution approach implemented by BELTISTOS-OPF, providing orders of magnitude faster solution times. Figure 8.26 also provides comparison of the factorization and forward-backward substitution phases, illustrated by the horizontal line inside the bars (note the logarithmic scale on the y-axis). The factorization phase clearly dominates for the BELTISTOS-OPF black-box approach, therefore the forward-backward substitution phase is not visible in the figure. The factorization and the backsubstitution

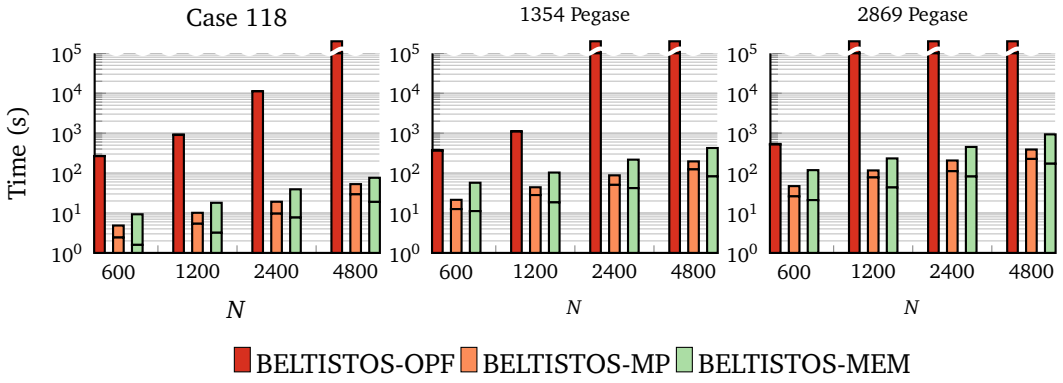


Figure 8.26. Average time per iteration for solving the KKT system. The horizontal line inside the bars marks the ratio between the factorization and solution phase (logarithmic scale). Number of storage devices fixed to $N_s = 10$.

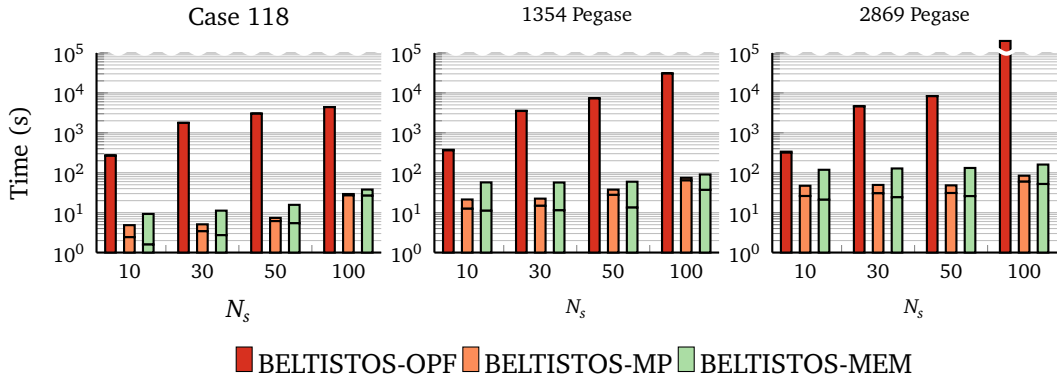


Figure 8.27. Average time per iteration for solving the KKT system. The horizontal line inside the bars marks the ratio between the factorization and solution phase (logarithmic scale). Number of time periods fixed to $N = 600$.

phases are comparable for the BELTISTOS-MP, with the backsubstitution phase taking more time for the memory efficient version of BELTISTOS-MP – BELTISTOS-MEM, since some portion of the computation needs to be recomputed redundantly in order to reduce the memory requirements of the algorithm. However, the performance benefit is still significant, compared to the standard solution method. We observe that the performance gap between BELTISTOS-MP and all other black-box solvers represented by BELTISTOS-OPF increases with increasing values of N. For $N = 4,800$ the BELTISTOS-OPF failed due to exceeding the memory limit during the symbolic factorization of the matrix. With increasing

benchmark size, the failure was also observed for $N = 2,400$ or $N = 1,200$ due to exceeding the available memory or the time limit. BELTISTOS-MP requires approximately 1% of the time needed by the best competitor for the smallest problem, with an increasing performance benefit for larger problems.

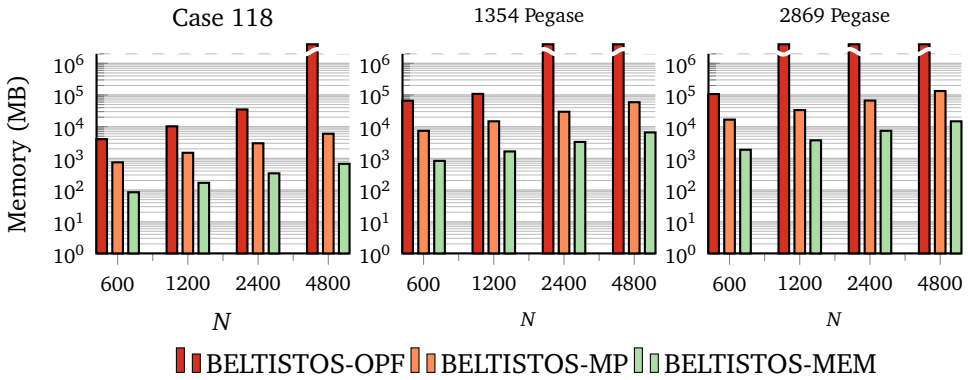


Figure 8.28. Memory requirements for solving the KKT system.

Next, the performance of BELTISTOS-MP is investigated for increasing number of storage devices. This is particularly important for storage sizing and placement problems, where the optimal size and location of the storage devices are sought for. The number of storage devices increases the number of coupling variables and thus the size of the Schur complement, therefore, also posing a bottleneck for large problems. Figure 8.27 shows the average solution time of the KKT system for an increasing number of storage devices N_s for different power grids, considering $N = 600$ time periods. In the case of the case1354pegase benchmark, the 10-fold increase in the storage devices resulted in an 81 times longer computation for the BELTISTOS-OPF, while only 3.7 times increase for the computation time of BELTISTOS-MP.

8.4.2 Memory complexity

The memory efficiency of the solvers is examined in this section. Figure 8.28 illustrates the memory requirements of each solver on the same set of MPOPF benchmarks as in Figure 8.26. Clearly, the black-box approach has the most extensive memory requirements, related to storing the factors for the full KKT systems. The structure-exploiting approach implemented in BELTISTOS-MP reduces the memory requirements by more than one order of magnitude by use of efficient linear algebra components adopted for the particular structure of the KKT system. Additionally, BELTISTOS-MP can be executed using the memory saving

computation model, represented by BELTISTOS-MEM. The memory requirements can be further reduced by releasing the memory required to store the L factors of the diagonal blocks, and recomputing the factorization during different phases of the Schur decomposition algorithm. Obviously, the redundant computations in the memory efficient algorithm are reflected in increased execution time. However, the execution time is still orders of magnitude faster than the BELTISTOS-OPF approach.

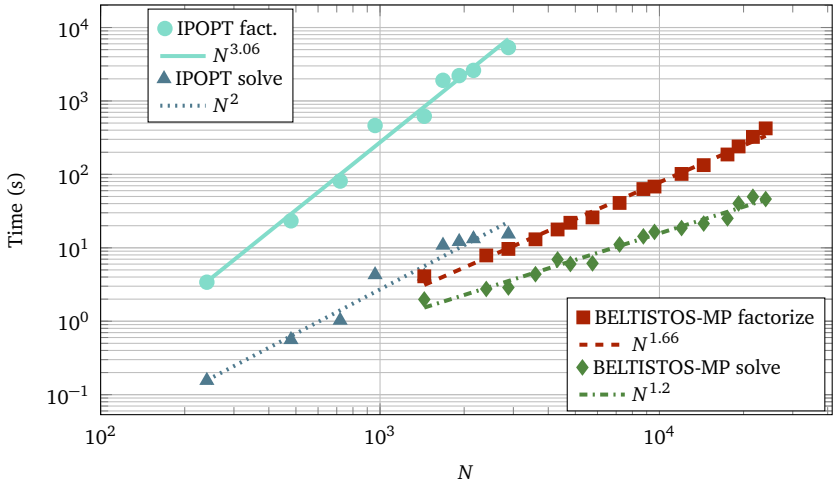


Figure 8.29. Regression analysis for the factorization and solution phases for both BELTISTOS-OPF and BELTISTOS-MP using case118.

8.4.3 Computational complexity

The complexity of the presented approach can be estimated by examining the complexity of the factorization steps. The number of nonzeros entries per row of OPF Hessians is approximately the same with that of Laplacian matrices discretized by finite element or finite volume methods on 3D meshes. Thus, the LU factorization for each one of the sparse diagonal blocks of the arrowhead KKT system, A_n, for most direct sparse solvers has a complexity of $O({N_D}^2)$ George [1973]; Liu [1990].

Consequently the complexity of the factorization for all the sparse diagonal blocks in Algorithm 1, will be $O(N{N_D}^2)$. However, for large N it is dominated by the factorization of the dense Schur complement matrix $\boldsymbol{S}$. The $LDL^\intercal$ factorization of $\boldsymbol{S}$ has complexity $O(n^3)$ for dense symmetric matrices of $\mathbb{R}^{n\times n}$ and the associated back substitution has complexity $O(n^2)$, with $n = NN_S$. This is

the case for each one of the standard methods, such as IPOPT or BELTISTOS-OPF. Exploiting the fact that the blocks below the main diagonal of each column of $\boldsymbol{S}$ are identical, which is a consequence of the structure inherent to the linear constraints (3.25h), the factorization can be performed in $O(n^2)$ operations and the back substitution in $O(n)$. Since the matrix $\boldsymbol{S}$ has a block structure with $N \times N$ blocks in $\mathbb{R}^{N_S \times N_S}$ each, the complexity for the factorization is $O(N^2 {N_S}^3)$ and for the back substitution $O(N {N_S}^2)$; see Rozlovzník et al. [2011] for a detailed discussion of block dense $LDL^\intercal$ algorithms. The reduction in the computational complexity and storage requirements of the Schur complement system, renders the overall approach significantly more economical in terms of overall running time and memory footprint.

In Figure 8.29, similar study on the complexity of the factorization and solution phases is shown for IPOPT and BELTISTOS-MP. As is expected, the complexity of IPOPT is cubic for the factorization phase and quadratic for the solution phase. However, although the factorization phase is expected to be quadratic for BELTISTOS-MP, very large N is needed for the Schur complement factorization to dominate the sparse factorization of the diagonal blocks A_n for which the memory on our workstation is not enough. Up to this point the factorization phase seems to have an overall complexity 1.6 in terms of running time instead of the estimated quadratic.

8.4.4 Swiss grid case study

The structure exploiting solution strategy is demonstrated on solutions of MPOPF problems using the Swiss transmission network. These benchmarks are similar and complement the SCOPF benchmarks in the previous section. The MPOPF problem consists of up to $4,800$ time periods and up to 50 installed storage devices. The load pattern for a ten day period is shown in Figure 8.30. The BELTISTOS-MP was the best solver in a set of benchmarks, achieving up to three orders of magnitude faster solution times compared to IPOPT. For larger problems IPOPT failed due to excessive memory requirements during the factorization of the KKT matrix. BELTISTOS-MEM achieves similar performance to BELTISTOS-MP, where the performance gap is most significant for benchmarks with $N_S = 50$ storage devices. The size of the blocks in the dense SC is a function of the number of storage devices, thus redundant factorization in the memory saving approach becomes pronounced for the problems with larger utilization of storage devices. The results are summarized in Figure 8.31.

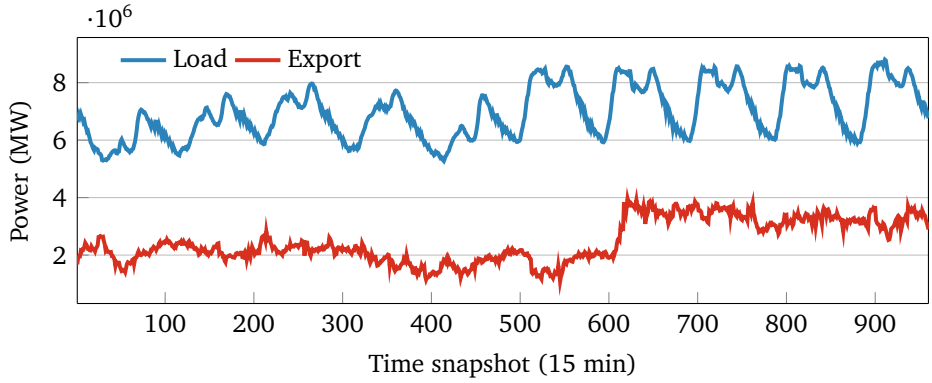


Figure 8.30. Swiss grid power load over a 10 day period (year 2014).

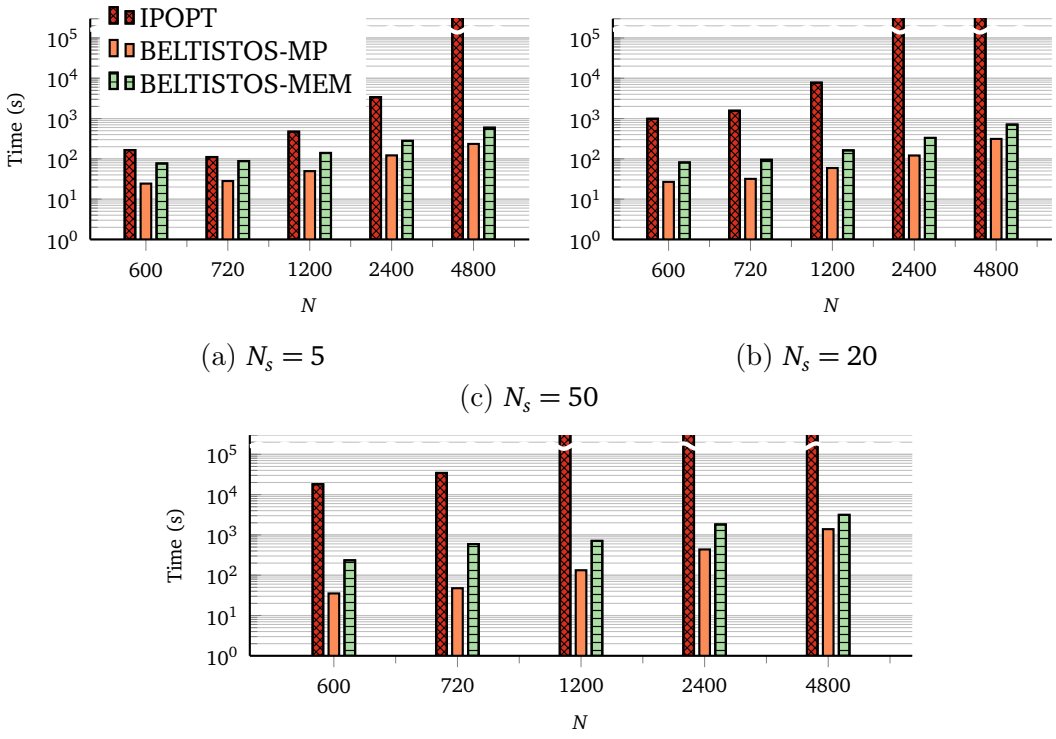


Figure 8.31. Average time per iteration for solving the KKT system of the Swiss power grid benchmark.

Chapter 9

Conclusions

This thesis was centered on computational methods for accelerating OPF problems, considering IP solution frameworks. The performed numerical experiments can be roughly split into two parts, single period OPF, and coupled SCOPF and MPOPF problems. The OPF problem was examined from multiple perspectives, starting from the formulation of the problem, analysis of various characteristics of the solution method, identifying aspects leading to improved performance and finally, considering linear algebra kernels and its application to modern, parallel multicore architectures with distributed memory hierarchies. The solution of the large-scale coupled OPF problem, such as SCOPF problem considered in this work, is possible by using sophisticated algorithmic improvements that achieved two orders of magnitude or more speedup even on single core, and is further increased using the parallel and distributed computing architectures.

A general assessment of the nonlinear IP frameworks used for solution of the OPF problems can be summarized such that all software packages demonstrate similar performance for small and medium-size networks. For smaller OPF problems, we find that, in general, there is little difference in terms of reliability and efficiency among the leading competitors IPOPT-PARDISO, KNITRO or BELTISTOS-OPF. It is important to note that the choice of the linear solver significantly influences the overall optimizer's performance, especially in case of large-scale problems. Significant performance differences between the optimizer convergence are observed for large networks. Two optimizers performed consistently better than the rest, namely KNITRO and BELTISTOS-OPF. KNITRO outperforms BELTISTOS-OPF for the OPF forumulation with Cartesian voltage coordinates, while BELTISTOS-OPF is better for polar voltage coordinates with nodal power balance. The choice of the initial guess was demonstrated to be crucial since it usually accelerates convergence to the optimal solution if chosen wisely.

In this respect, the solution of the power flow equations should always be preferred instead of averages of upper and lower bounds or flat starts.

Next, it was demonstrated that significant performance gains are possible for specific classes of coupled optimal control problems not only by exploiting supercomputers and parallel distributed or multithreaded programming, but also by deeper understanding of the problem structures and the design of algorithms adapted to them. Orders of magnitude faster execution time were achieved and solutions of very-large-scale problems became possible. The BELTISTOS-SC framework accelerated the solution of large-scale SCOPF problems through several algorithmic advancements. On the linear level, the solution of the SCOPF problem was accelerated by several orders of magnitude by adopting highly efficient sparse linear algebra algorithms. The numerical experiments demonstrated speedup up to 270-fold compared to the black-box approach, even for single-node execution. Finally, the parallel computing architectures were employed to achieve additional performance gains.

These findings recommend a deeper redesign of the entire power grid optimization software ecosystem, aiming to realign optimization algorithms with the realities of the underlying hardware constraints. Exploiting distributed multicore and manycore nodes for the solution of the KKT system drastically reduces the execution times and demonstrates significant progress towards the solution of large-scale power grid problems with the stringent time requirements of the power grid operators.

9.1 Outlook and discussion

The solution of power grid optimization problems based on sparse linear algebra and structure exploiting methods has been pushed to the limit and, as it seems, additional improvements are still possible on both linear and nonlinear level. The challenges are to design similar algorithms for problems such as MPOPF including generator ramping constraints, the planning and placement problem, stochastic OPF etc. Extensions of the presented methodology to other power grid problems are quite tempting:

Stochastic OPF Source of uncertainty related to the intermittent nature of renewable in feed and electricity demand, are modeled by stochastic scenarios aiming to capture the uncertainties considering various correlations. The structure of the resulting problem known as Stochastic OPF, also results in arrowhead sparse matrices and can benefit from Schur decompositions

as it has already been reported Petra, Schenk and Anitescu [2014]. However, further exploitation of the structure and matrices may be possible, reducing further the computing time and memory requirements.

Multiperiod OPF with generator ramp limits In Europe active power generation is determined by the market and remains fixed in the OPF problems solved for supporting TSOs daily operating decisions. Many real-life applications could benefit however, from extension of MPOPF should it model generator ramping constraints. These constraints introduce inter-temporal coupling of individual time periods, additional to the ones introduced by storage devices discussed previously.

Transient grid models The structure exploiting solver can be also applied within the SCCER-FURIES framework developed by other researchers Demiray and Andersson [2009], where a flexible, modular, and computationally efficient framework is suggested to allow transient simulations of power grid problems. The structure of the linear systems resulting from the time discretization is appropriate for the parallel Schur complement decomposition scheme.

The power grid models considered in this study are the standard public-access domain datasets. These datasets were designed for testing the OPF algorithms and asses the numerical behavior of the novel solution methods on mostly synthetic power grids. As a next step, the developed computational tools are used within the SCCER-FURIES Digitalization[1] project, which closely collaborates with other academic partners and tries to deploy the application with industrial partners. One of the aims of this digitalization project is to create a tool for the secure high-performance storage scheduling. This will be achieved through a complete problem formulation across multiple time scales available. The implementation of a fully distributed and parallel IP framework was not attempted as a part of this work, since this was accomplished before, e.g. by PIPS Petra [2014]; Chiang et al. [2014], and thus this represents a natural next step. The parallelism is applied on the level of the KKT system solution, while the IP algorithm is run sequentially by a single process, which introduces a considerable bottleneck and unnecessary data transfers. The purpose was to show that a more efficient implementation is possible by implementing sparse linear algebra routines which further improve intranode performance compared to previous work. The proposed method employs one additional Schur complement, chosen so that the sparsity of the KKT

[1] `https://search.usi.ch/en/projects/1074/sccer-furies-digitalization`

matrix is maintained. Consequently, the size of the linear system decreases and this allows for memory savings and increased computational performance.

Additionally, the efficient Schur decomposition heavily relies on the incomplete factorization approach of the augmented matrix for directly computing the local Schur complements. Some perturbation is introduced to the local Schur complements to allow for more efficient pivoting during their computation. Consequently, the factorization of the diagonal blocks, which can be implicitly obtained from the incomplete factorization approach, is also perturbed. Therefore, the iterative refinement is essential and needs to be applied to each solve operation with the diagonal block. Doing so might introduce load imbalance for the parallel computations, since different numbers of iterative refinement iterations might be performed for individual diagonal blocks to reach the desired residual. The resulting solution of the entire KKT system might contain considerable error if the iterative refinement is not allowed to perform a sufficient number of iterations. In order to address this shortcoming, although it was not observed in the numerical experiments, the distributed solution might be used as a very efficient implicit preconditioner applied in an iterative method for the entire KKT system.

It was attempted to further advance the solution of large-scale problems by exploring two additional methods using an alternative to the Schur decomposition approach on the linear level. The first method was based on nonlinear decomposition of the SCOPF problem, such that it is decomposed into solution of many embarrassingly parallel power flow problems. These are much easier to solve and many robust and fast power flow solvers exist. The approach, however, relies only on the first order information while the Hessians are approximated. The convergence characteristics of the method need to be assessed in more rigorous study in future work. It was observed that the method requires significantly more iterations to reach the desired tolerance and in certain cases the tolerance could not be reached. The second attempt was based on the idea of formulating the solution method such that the dense linear algebra becomes a computational kernel. In such a case, the computation can be greatly improved by leveraging the computational load to modern hardware accelerators. The formulation based on the augmented Lagrangian was explored, using the Hessian approximations based on limited memory secant updates. The open question for the future is rigorous mathematical analysis of the behavior of the method, especially on difficult nonconvex problems, such as OPF.

Bibliography

Akcelik, V., Biros, G., Ghattas, O., Hill, J., Keyes, D. and van Bloemen Waanders, B. [2006]. *Parallel Algorithms for PDE-Constrained Optimization*, Society for Industrial and Applied Mathematics, chapter 16, pp. 291–322.

Alsac, O., Bright, J., Prais, M. and Stott, B. [1990]. Further developments in lp-based optimal power flow, *IEEE Transactions on Power Systems* **5**(3): 697–711.

Arioli, M. and Scott, J. [2014]. Chebyshev acceleration of iterative refinement, *Numerical Algorithms* **66**(3): 591–608.

Babaeinejadsarookolaee, S., Birchfield, A., Christie, R. D., Coffrin, C., DeMarco, C., Diao, R., Ferris, M., Fliscounakis, S., Greene, S., Huang, R., Josz, C., Korab, R., Lesieutre, B., Maeght, J., Molzahn, D. K., Overbye, T. J., Panciatici, P., Park, B., Snodgrass, J. and Zimmerman, R. [2019]. The power grid library for benchmarking ac optimal power flow algorithms.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P. and Vance, P. H. [1998]. Branch-and-price: Column generation for solving huge integer programs, *Operations Research* **46**(3): 316–329.

Benzi, M., Golub, G. H. and Liesen, J. [2005]. Numerical solution of saddle point problems, *Acta Numerica* **14**: 1–137.

Biegler, L. T., Ghattas, O., Heinkenschloss, M. and van Bloemen Waanders, B. [2003]. Large-scale pde-constrained optimization: An introduction, *in* L. T. Biegler, M. Heinkenschloss, O. Ghattas and B. van Bloemen Waanders (eds), *Large-Scale PDE-Constrained Optimization*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 3–13.

Birchfield, A. B., Xu, T., Gegner, K. M., Shetye, K. S. and Overbye, T. J. [2017]. Grid structural characteristics as validation criteria for synthetic networks, *IEEE Transactions on Power Systems* **32**(4): 3258–3265.

Borges, C. L. T. and Alves, J. M. T. [2007]. Power system real time operation based on security constrained optimal power flow and distributed processing, *2007 IEEE Lausanne Power Tech*, pp. 960–965.

Boyd, S., Parikh, N., Chu, E., Peleato, B. and Eckstein, J. [2011]. Distributed optimization and statistical learning via the alternating direction method of multipliers, *Foundations and Trends in Machine Learning* **3**(1): 1–122.

Bunch, J. R. and Kaufman, L. [1977]. Some stable methods for calculating inertia and solving symmetric linear systems, *Mathematics of Computation* **31**(137): 163–179.

Burchett, R. C., Happ, H. H. and Wirgau, K. A. [1982]. Large scale optimal power flow, *IEEE Transactions on Power Apparatus and Systems* **PAS-101**(10): 3722–3732.

Byrd, R. H., Gilbert, J. C. and Nocedal, J. [2000]. A trust region method based on interior point techniques for nonlinear programming, *Mathematical Programming* **89**(1): 149–185.

Byrd, R. H., Hribar, M. E. and Nocedal, J. [1999]. An interior point algorithm for large-scale nonlinear programming, *SIAM Journal on Optimization* **9**(4): 877–900.

Byrd, R. H., Liu, G. and Nocedal, J. [1998]. On the local behavior of an interior point method for nonlinear programming, *Numerical Analysis 1997*, Addison Wesley Longman, pp. 37–56.

Byrd, R. H., Nocedal, J. and Waltz, R. A. [2006]. *Knitro: An Integrated Package for Nonlinear Optimization*, Springer US, Boston, MA, pp. 35–59.

Capitanescu, F., Glavic, M., Ernst, D. and Wehenkel, L. [2007]. Contingency filtering techniques for preventive security-constrained optimal power flow, *IEEE Transactions on Power Systems* **22**(4): 1690–1697.

Capitanescu, F., Ramos, J. M., Panciatici, P., Kirschen, D., Marcolini, A. M., Platbrood, L. and Wehenkel, L. [2011]. State-of-the-art, challenges, and future trends in security constrained optimal power flow, *Electric Power Systems Research* **81**(8): 1731 – 1741.

Capitanescu, F. and Wehenkel, L. [2008]. A new iterative approach to the corrective security-constrained optimal power flow problem, *IEEE Transactions on Power Systems* **23**(4): 1533–1541.

Capitanescu, F. and Wehenkel, L. [2013]. Experiments with the interior-point method for solving large scale optimal power flow problems, *Electric Power Systems Research* **95**: 276 – 283.

Carpentier, J. [1962]. Contribution to the economic dispatch problem, *Bulletin de la Societe Francoise des Electriciens* **3**(8): 431–447.

Castillo, A. and O'Neill, R. P. [2013]. Survey of approaches to solving the acopf, *Technical report*, Federal Energy Regulatory Commission.

Chakrabarti, S., Kraning, M., Chu, E., Baldick, R. and Boyd, S. [2014]. Security constrained optimal power flow via proximal message passing, *2014 Clemson University Power Systems Conference*, pp. 1–8.

Chen, D., Jiang, H., Li, Y. and Xu, D. [2017]. A two-layered parallel static security assessment for large-scale grids based on GPU, *IEEE Transactions on Smart Grid* **8**(3): 1396–1405.

Chiang, N., Petra, C. G. and Zavala, V. M. [2014]. Structured nonconvex optimization of large-scale energy systems using pips-nlp, *2014 Power Systems Computation Conference*, pp. 1–7.

Chiang, N.-Y. [2014]. PIPS-NLP.
URL: *https://www.mcs.anl.gov/ nychiang/pipsnlp.html*

Chiang, N.-Y. and Zavala, V. M. [2016]. An inertia-free filter line-search algorithm for large-scale nonlinear programming, *Computational Optimization and Applications* **64**(2): 327–354.

CIGRE Task Force [2014]. Benchmark systems for network integration of renewable and distributed energy resources. ELT_273_8.

Coffrin, C., Bent, R., Sundar, K., Ng, Y. and Lubin, M. [2017]. PowerModels.jl: An open-source framework for exploring power flow formulations.

Coffrin, C., Gordon, D. and Scott, P. [2014]. NESTA, the NICTA energy system test case archive.

Conn, A. R., Gould, N. I. M., Orban, D. and Toint, P. L. [2000]. A primal-dual trust-region algorithm for non-convex nonlinear programming, *Mathematical Programming* **87**(2): 215–249.

Costa, VM ; Martins, Nelson; Pereira, J. L. [1999]. Developments in the Newton Raphson power flow formulation based on current injections, *IEEE Transactions on Power Systems* **14**(4): 1320–1326.

CSCS - Swiss National Supercomputing Centre [2018]. Piz Daint.
URL: *https://www.cscs.ch/computers/piz-daint/*

Curtis, F. E., Huber, J., Schenk, O. and Wächter, A. [2012]. A note on the implementation of an interior-point algorithm for nonlinear optimization with inexact step computations, *Mathematical Programming* **136**(1): 209–227.

Curtis, F. E., Nocedal, J. and Wächter, A. [2010]. A matrix-free algorithm for equality constrained optimization problems with rank-deficient jacobians, *SIAM Journal on Optimization* **20**(3): 1224–1249.

Curtis, F. E., Schenk, O. and Wächter, A. [2010]. An interior-point algorithm for large-scale nonlinear optimization with inexact step computations, *SIAM J. Sci. Comput.* **32**(6): 3447–3475.

D'Apuzzo, M., De Simone, V. and di Serafino, D. [2010]. On mutual impact of numerical linear algebra and large-scale optimization with focus on interior point methods, *Computational Optimization and Applications* **45**(2): 283–310.

Demiray, T. and Andersson, G. [2009]. Optimization of numerical integration methods for the simulation of dynamic phasor models in power systems, *International Journal of Electrical Power and Energy Systems* **31**(9): 512 – 521. Power Systems Computation Conference (PSCC) 2008.

Dolan, E. D. and Moré, J. J. [2002]. Benchmarking optimization software with performance profiles, *Mathematical Programming* **91**(2): 201–213.

Du, Y., Li, F. and Huang, C. [2019]. Applying deep convolutional neural network for fast security assessment with n-1 contingency, *2019 IEEE Power Energy Society General Meeting (PESGM)*, pp. 1–5.

Duff, I. S. and Scott, J. A. [2005]. Stabilized bordered block diagonal forms for parallel sparse solvers, *Parallel Comput.* **31**(3-4): 275–289.

Dunning, I., Huchette, J. and Lubin, M. [2017]. Jump: A modeling language for mathematical optimization, *SIAM Review* **59**(2): 295–320.

Engelmann, A., Jiang, Y., Mühlpfordt, T., Houska, B. and Faulwasser, T. [2019]. Toward distributed OPF using ALADIN, *IEEE Transactions on Power Systems* **34**(1): 584–594.

Erseghe, T. [2015]. A distributed approach to the opf problem, *EURASIP Journal on Advances in Signal Processing* **2015**(1): 45.

Fletcher, R. and Leyffer, S. [2002]. Nonlinear programming without a penalty function, *Mathematical Programming* **91**(2): 239–269.

Fletcher, R. and Leyffer, S. [2004]. Solving mathematical programs with complementarity constraints as nonlinear programs, *Optimization Methods and Software* **19**(1): 15–40.

Fliscounakis, S., Panciatici, P., Capitanescu, F. and Wehenkel, L. [2013a]. Contingency ranking with respect to overloads in very large power systems taking into account uncertainty, preventive, and corrective actions, *IEEE Transactions on Power Systems* **28**(4): 4909–4917.

Fliscounakis, S., Panciatici, P., Capitanescu, F. and Wehenkel, L. [2013b]. Contingency ranking with respect to overloads in very large power systems taking into account uncertainty, preventive, and corrective actions, *IEEE Transactions on Power Systems* **28**(4): 4909–4917.

Forsgren, A., Gill, P. E. and Wright, M. H. [2002]. Interior methods for nonlinear optimization, *SIAM Review* **44**(4): 525–597.

Fortenbacher, P. and Demiray, T. [2019]. Linear/quadratic programming-based optimal power flow using linear power flow and absolute loss approximations, *International Journal of Electrical Power & Energy Systems* **107**: 680 – 689.

Fortenbacher, P., Ulbig, A. and Andersson, G. [2018]. Optimal Placement and Sizing of Distributed Battery Storage in Low Voltage Grids Using Receding Horizon Control Strategies, *IEEE TRANSACTIONS ON POWER SYSTEMS* **33**(3).

Frank, S. and Rebennack, S. [2012]. A Primer on Optimal Power Flow: Theory, Formulation, and Practical Examples Title: A Primer on Optimal Power Flow: Theory, Formulation, and Practical Examples.

Frank, S., Steponavice, I. and Rebennack, S. [2012]. Optimal power flow: a bibliographic survey I, *Energy Systems* **3**(3): 221–258.

Fuchs, A., Garrison, J. and Demiray, T. [2017]. A security-constrained multi-period opf for the locational allocation of automatic reserves, *2017 IEEE Manchester PowerTech*, pp. 1–6.

George, A. [1973]. Nested dissection of a regular finite element mesh, *SIAM Journal on Numerical Analysis* **10**(2): 345–363.

Gertz, E. M. and Wright, S. J. [2003]. Object-oriented software for quadratic programming, *ACM Trans. Math. Softw.* **29**(1): 58–81.

Gondzio, J. and Grothey, A. [2008]. A new unblocking technique to warmstart interior point methods based on sensitivity analysis, *SIAM Journal on Optimization* **19**: 1184–1210.

Gondzio, J. and Grothey, A. [2009]. Exploiting structure in parallel implementation of interior point methods for optimization, *Computational Management Science* **6**(2): 135–160.

Gondzio, J. and Sobral, F. N. C. [2019]. Quasi-newton approaches to interior point methods for quadratic problems, *Computational Optimization and Applications* **74**(1): 93–120.

Gould, N. I. M., Orban, D., Sartenaer, A. and Toint, P. L. [2001]. Superlinear convergence of primal-dual interior point algorithms for nonlinear programming, *SIAM Journal on Optimization* **11**(4): 974–1002.

Gould, N. I. M. and Scott, J. A. [2004]. A numerical evaluation of HSL packages for the direct solution of large sparse, symmetric linear systems of equations, *ACM Trans. Math. Softw.* **30**(3): 300–325.

Gould, N. I. M., Scott, J. and Hu, Y. [2007]. A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations, *ACM Trans. Math. Softw.* **33**: 10.

Gould, N. and Scott, J. [2016]. A note on performance profiles for benchmarking software, *ACM Trans. Math. Softw.* **43**(2): 15:1–15:5.

Granville, S. [1994]. Optimal reactive dispatch through interior point methods, *IEEE Transactions on Power Systems* **9**(1): 136–146.

Gupta, A., Karypis, G. and Kumar, V. [1997]. Highly scalable parallel algorithms for sparse matrix factorization, *IEEE Transactions on Parallel and Distributed Systems* **8**(5): 502–520.

Helman, U. [2019]. Chapter 19 - distributed energy resources in the us wholesale markets: Recent trends, new models, and forecasts, *in* F. Sioshansi (ed.), *Consumer, Prosumer, Prosumager*, Academic Press, pp. 431 – 469.

Hogg, J. D. and Scott, J. A. [2013]. On the effects of scaling on the performance of ipopt.

HSL [2002]. HSL. A collection of Fortran codes for large scale scientific computation.
URL: *http://www.hsl.rl.ac.uk*

Hug-Glanzmann, G. and Andersson, G. [2009]. Decentralized optimal power flow control for overlapping areas in power systems, *IEEE Transactions on Power Systems* **24**(1): 327–336.

Huneault, M. and Galiana, F. D. [1991]. A survey of the optimal power flow literature, *IEEE Transactions on Power Systems* **6**(2): 762–770.

Intel [2019]. *Developer Reference for Intel Math Kernel Library*, Intel Corporation. Chapter 5: Sparse Solver Routines.
URL: *https://software.intel.com/en-us/download/developer-reference-for-intel-math-kernel-library-c*

Jiang, H. and Ralph, D. [2000]. Smooth SQP methods for mathematical programs with nonlinear complementarity constraints, *SIAM Journal on Optimization* **10**(3): 779–808.

Jiang, Q. and Xu, K. [2014]. A novel iterative contingency filtering approach to corrective security-constrained optimal power flow, *IEEE Transactions on Power Systems* **29**(3): 1099–1109.

John, E. and Yıldırım, E. A. [2008]. Implementation of warm-start strategies in interior-point methods for linear programming in fixed dimension, *Computational Optimization and Applications* **41**(2): 151–183.

Josz, C., Fliscounakis, S., Maeght, J. and Panciatici, P. [2016]. AC Power Flow Data in MATPOWER and QCQP Format: iTesla, RTE Snapshots, and PEGASE, *ArXiv e-prints* .

Joubert, C. J., Chokani, N. and Abhari, R. S. [2018]. Impact of large scale battery energy storage on the 2030 central european transmission grid, *2018 15th International Conference on the European Energy Market (EEM)*, pp. 1–5.

Kang, J. [2015]. *An Efficient Interior-Point Decomposition Algorithm for Parallel Solution of Large-Scale Nonlinear Problems with Significant Variable Coupling*, PhD thesis, Texas A & M University, http://hdl.handle.net/1969.1/156141.

Kaplunovich, P. and Turitsyn, K. [2016]. Fast and reliable screening of n-2 contingencies, *IEEE Transactions on Power Systems* **31**(6): 4243–4252.

Kardos, J., Kourounis, D. and Schenk, O. [2018]. Complete results for a numerical evaluation of interior point solvers for large-scale optimal power flow problems, *ArXiv e-prints: 1807.03964* .

Kardos, J., Kourounis, D. and Schenk, O. [2020a]. Reduced-space interior point methods in power grid problems, *ArXiv e-prints: 2001.10815* .

Kardos, J., Kourounis, D. and Schenk, O. [2020b]. Structure Exploiting Interior Point Methods, *ArXiv e-prints: 1907.05420* .

Kardoš, J., Kourounis, D. and Schenk, O. [2020]. Two-level parallel augmented schur complement interior-point algorithms for the solution of security constrained optimal power flow problems, *IEEE Transactions on Power Systems* **35**(2): 1340–1350.

Kargarian, A., Fu, Y. and Li, Z. [2015]. Distributed security-constrained unit commitment for large-scale power systems, *IEEE Transactions on Power Systems* **30**(4): 1925–1936.

Kargarian, A., Mohammadi, J., Guo, J., Chakrabarti, S., Barati, M., Hug, G., Kar, S. and Baldick, R. [2018]. Toward distributed/decentralized dc optimal power flow implementation in future electric power systems, *IEEE Transactions on Smart Grid* **9**(4): 2574–2594.

Karmarkar, N. [1984]. A new polynomial-time algorithm for linear programming, *Combinatorica* **4**(4): 373–395.

Karypis, G. and Kumar, V. [1998]. A fast and high quality multilevel scheme for partitioning irregular graphs, *SIAM J. Sci. Comput.* **20**(1): 359âĂŞ392.

Kourounis, D., Durlofsky, L. J., Jansen, J. D. and Aziz, K. [2014a]. Adjoint formulation and constraint handling for gradient-based optimization of compositional reservoir flow, *Computational Geosciences* **18**(2): 117–137.

Kourounis, D., Durlofsky, L. J., Jansen, J. D. and Aziz, K. [2014b]. Adjoint formulation and constraint handling for gradient-based optimization of compositional reservoir flow, *Computational Geosciences* **18**(2): 117–137.

Kourounis, D., Fuchs, A. and Schenk, O. [2018]. Toward the next generation of multiperiod optimal power flow solvers, *IEEE Transactions on Power Systems* **33**(4): 4005–4014.

Kourounis, D. and Schenk, O. [2018]. Optimal power flow solvers of extreme performance.
URL: *www.beltistos.com*

Lavaei, J. and Low, S. H. [2012]. Zero duality gap in optimal power flow problem, *IEEE Transactions on Power Systems* **27**(1): 92–107.

Lewis, A. S. and Overton, M. L. [2013]. Nonsmooth optimization via quasi-newton methods, *Mathematical Programming* **141**(1): 135–163.

Liu, J. W. [1990]. The role of elimination trees in sparse factorization, *SIAM Journal on Matrix Analysis and Applications* **11**(1): 134–172.

López, P. C., Sadikovic, R., Pinto, H. and Magnago, F. [2015]. Swiss tso experience with an ac security-constrained optimal power flow application for real-time security management, *2015 IEEE Eindhoven PowerTech*, pp. 1–6.

Low, S. H. [2014]. Convex relaxation of optimal power flow; part I: Formulations and equivalence, *IEEE Transactions on Control of Network Systems* **1**(1): 15–27.

Lu, C. N. and Unum, M. R. [1993]. Network constrained security control using an interior point algorithm, *IEEE Transactions on Power Systems* **8**(3): 1068–1076.

Macfie, P. J., Taylor, G. A., Irving, M. R., Hurlock, P. and Wan, H. [2010]. Proposed shunt rounding technique for large-scale security constrained loss minimization, *IEEE Transactions on Power Systems* **25**(3): 1478–1485.

Marano-Marcolini, A., Capitanescu, F., Martinez-Ramos, J. L. and Wehenkel, L. [2012]. Exploiting the Use of DC SCOPF Approximation to Improve Iterative AC SCOPF Algorithms, *IEEE Transactions on Power Systems* **27**(3): 1459–1466.

Marley, J. F., Molzahn, D. K. and Hiskens, I. A. [2017]. Solving multiperiod opf problems using an ac-qp algorithm initialized with an socp relaxation, *IEEE Transactions on Power Systems* **32**(5): 3538–3548.

MathWorks [2018]. Matlab™Optimization Toolbox User's Guide, Version 2. The MathWorks, Natick, MA, USA.

Mehrotra, S. [1992]. On the implementation of a primal-dual interior point method, *SIAM Journal on Optimization* **2**(4): 575–601.

Mohammadi, J., Hug, G. and Kar, S. [2013]. A benders decomposition approach to corrective security constrained opf with power flow control devices, *2013 IEEE Power Energy Society General Meeting*, pp. 1–5.

Mohammadi, J., Hug, G. and Kar, S. [2018]. Agent-based distributed security constrained optimal power flow, *IEEE Transactions on Smart Grid* **9**(2): 1118–1130.

Mohler, D. and Sowder, D. [2014]. Chapter 23 - Energy storage and the need for flexibility on the grid, *in* L. E. Jones (ed.), *Renewable Energy Integration*, Academic Press, Boston, pp. 285 – 292.

Momoh, J. [2000]. *Electric Power System Applications of Optimization*, Power Engineering (Willis), Taylor & Francis.

Momoh, J. A., El-Hawary, M. and Adapa, R. [1999]. A review of selected optimal power flow literature to 1993. Part II: Newton, linear programming and interior point methods, *IEEE Trans. on Power Syst.* **14**(1): 105–111.

Monticelli, A., Pereira, M. V. F. and Granville, S. [1987]. Security-constrained optimal power flow with post-contingency corrective rescheduling, *IEEE Transactions on Power Systems* **2**(1): 175–180.

More, J. J. and Wild, S. M. [2009]. Benchmarking derivative-free optimization algorithms, *SIAM Journal on Optimization* **20**(1): 172–191.

Nocedal, J., Wächter, A., and Waltz, R. A. [2009]. Adaptive barrier update strategies for nonlinear interior methods, *SIAM Journal on Optimization* **19**(4): 1674–1693.

Nocedal, J. and Wright, S. J. [2006]. *Numerical Optimization*, second edn, Springer, New York, NY, USA.

Palmer, B., Perkins, W., Chen, Y., Jin, S., Callahan, D., Glass, K., Diao, R., Rice, M., Elbert, S., Vallem, M. and Huang, Z. [2014]. Gridpack: A framework for developing power grid simulations on high performance computing platforms, *2014 Fourth International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing*, pp. 68–77.

Palmer, B., Perkins, W., Chen, Y., Jin, S., Callahan, D., Glass, K., Diao, R., Rice, M., Elbert, S., Vallem, M. and Huang, Z. [2015]. Gridpack: A framework for developing power grid simulations on high performance computing platforms, *International Journal of High Performance Computing Applications* **30**.

Park, C., Knazkins, V., Sevilla, F. R. S., Korba, P. and Poland, J. [2015]. On the estimation of an optimum size of energy storage system for local load shifting, *2015 IEEE Power Energy Society General Meeting*, pp. 1–5.

Petra, C. [2014]. PIPS.
URL: *https://github.com/Argonne-National-Laboratory/PIPS*

Petra, C. G., Schenk, O. and Anitescu, M. [2014]. Real-time stochastic optimization of complex energy systems on high-performance computers, *Computing in Science Engineering* **16**(5): 32–42.

Petra, C. G., Schenk, O., Lubin, M. and Gärtner, K. [2014]. An augmented incomplete factorization approach for computing the schur complement in stochastic optimization, *SIAM Journal on Scientific Computing* **36**(2): C139–C162.

Phan, D. T. and Sun, X. A. [2015]. Minimal impact corrective actions in security-constrained optimal power flow via sparsity regularization, *IEEE Transactions on Power Systems* **30**(4): 1947–1956.

Platbrood, L., Capitanescu, F., Merckx, C., Crisciu, H. and Wehenkel, L. [2014]. A generic approach for solving nonlinear-discrete security-constrained optimal power flow problems in large-scale systems, *IEEE Transactions on Power Systems* **29**(3): 1194–1203.

Qiu, W., Flueck, A. J. and Tu, F. [2005]. A new parallel algorithm for security constrained optimal power flow with a nonlinear interior point method, *Power Engineering Society General Meeting, 2005. IEEE*, IEEE, pp. 447–453.

R, S., Kumar, R. S. and Mathew, A. T. [2013]. Online static security assessment module using artificial neural networks, *IEEE Transactions on Power Systems* **28**(4): 4328–4335.

Rahmaniani, R., Crainic, T. G., Gendreau, M. and Rei, W. [2017]. The benders decomposition algorithm: A literature review, *European Journal of Operational Research* **259**(3): 801 – 817.

Rodrigues, M., Saavedra, O. R. and Monticelli, A. [1994]. Asynchronous programming model for the concurrent solution of the security constrained optimal power flow problem, *IEEE Transactions on Power Systems* **9**(4): 2021–2027.

Rozlovzník, M., Shklarski, G. and Toledo, S. [2011]. Partitioned triangular tridiagonalization, *ACM Trans. Math. Softw.* **37**(4): 38:1–38:16.

Rudion, K., Orths, A., Styczynski, Z. A. and Strunz, K. [2006]. Design of benchmark of medium voltage distribution network for investigation of dg integration, *2006 IEEE Power Engineering Society General Meeting*, pp. 6 pp.–.

Rüeger, C., Dobrowolski, J., Korba, P. and Sevilla, F. R. S. [2019]. Lyapunov exponent for evaluation and ranking of the severity of grid events on extra-large power systems, *2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*, pp. 1–5.

Rupp, K. [2018]. Karl rupp, computational scientist.
URL: *https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/*

Sauter, P. S., Braun, C. A., Kluwe, M. and Hohmann, S. [2017]. Comparison of the Holomorphic Embedding Load Flow Method with Established Power Flow Algorithms and a New Hybrid Approach, *IEEE Green Technologies Conference*, IEEE Computer Society, pp. 203–210.

Schanen, M., Gilbert, F., Petra, C. and Anitescu, M. [2018]. Toward multi-period ac-based contingency constrained optimal power flow at large scale, *20th Power systems computation conference, PSCC*, pp. 1–7.

Schenk, O. and Gärtner, K. [2004]. Solving unsymmetric sparse systems of linear equations with pardiso, *Future Generation Computer Systems* **20**(3): 475 – 487. Selected numerical algorithms.

Schenk, O. and Gärtner, K. [2006a]. On fast factorization pivoting methods for sparse symmetric indefinite systems, *Electronic Transactions on Numerical Analysis* **23**: 158–179.

Schenk, O. and Gärtner, K. [2006b]. On Fast Factorization Pivoting Methods for Sparse Symmetric Indefinite Systems, *Elec. Trans. Numer. Anal.* **23**: 158–179.

Schenk, O., Gärtner, K., Fichtner, W. and Stricker, A. [2001]. Pardiso: a high-performance serial and parallel sparse linear solver in semiconductor device

simulation, *Future Generation Computer Systems* **18**(1): 69 – 78. I. High Performance Numerical Methods and Applications. II. Performance Data Mining: Automated Diagnosis, Adaption, and Optimization.

Scott, J. A., Hu, Y. and Gould, N. I. M. [2006]. *An Evaluation of Sparse Direct Symmetric Solvers: An Introduction and Preliminary Findings*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 818–827.

Semerow, A., Höhn, S., Luther, M., Sattinger, W., Abildgaard, H., Garcia, A. D. and Giannuzzi, G. [2015]. Dynamic study model for the interconnected power system of continental europe in different simulation tools, *2015 IEEE Eindhoven PowerTech*, pp. 1–6.

Sevilla, F. R. S., Korba, P., Uhlen, K., Hillberg, E., Lindahl, G. and Sattinger, W. [2017]. Evaluation of the entso-e initial dynamic model of continental europe subject to parameter variations, *2017 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pp. 1–2.

Sevilla, F. R. S., Parra, D., Wyrsch, N., Patel, M. K., Kienzle, F. and Korba, P. [2018]. Techno-economic analysis of battery storage and curtailment in a distribution grid with high pv penetration, *Journal of Energy Storage* **17**: 73 – 83.

Shchetinin, D., De Rubira, T. T. and Hug, G. [2019]. On the construction of linear approximations of line flow constraints for ac optimal power flow, *IEEE Transactions on Power Systems* **34**(2): 1182–1192.

Singh, A., Willi, D., Chokani, N. and Abhari, R. S. [2014]. Optimal power flow analysis of a switzerland's transmission system for long-term capacity planning, *Renewable and Sustainable Energy Reviews* **34**: 596 – 607.

Sperstad, I. and Korpas, M. [2019]. Energy Storage Scheduling in Distribution Systems Considering Wind and Photovoltaic Generation Uncertainties, *Energies* **12**(7): 1231.

Stott, B. and Hobson, E. [1978]. Power system security control calculations using linear programming, part I, *IEEE Transactions on Power Apparatus and Systems* **PAS-97**(5): 1713–1720.

Swiss Federal Office of Energy [2018]. Energy strategy 2050.
URL: *https://www.bfe.admin.ch/bfe/en/home/policy/energy-strategy-2050.html*

Tahanan, M., van Ackooij, W., Frangioni, A. and Lacalandra, F. [2015]. Large-scale unit commitment under uncertainty, *4OR* **13**(2): 115–171.

Tinoco De Rubira, T. and Hug, G. [2016]. Adaptive certainty-equivalent approach for optimal generator dispatch under uncertainty, *European Control Conference (ECC)* .

Torres, G. L. and Quintana, V. H. [1998]. An interior-point method for nonlinear optimal power flow using voltage rectangular coordinates, *IEEE Transactions on Power Systems* **13**(4): 1211–1218.

Trader, T. [2019]. It's official: Aurora on track to be first us exascale computer in 2021.
URL: *https://www.hpcwire.com/2019/03/18/its-official-aurora-on-track-to-be-first-u-s-exascale-computer-in-2021/*

Trias, A. [2015]. Fundamentals of the Holomorphic Embedding Load-Flow Method.

Vanderbei, R. J. [1999]. Loqo:an interior point code for quadratic programming, *Optimization Methods and Software* **11**(1-4): 451–484.

Vargas, L. S., Quintana, V. H. and Vannelli, A. [1993]. A tutorial description of an interior point method and its applications to security-constrained economic dispatch, *IEEE Transactions on Power Systems* **8**(3): 1315–1324.

Venkatasubramanian, M. and Tomsovic, K. [2005]. 7 - power system analysis, *in* W.-K. CHEN (ed.), *The Electrical Engineering Handbook*, Academic Press, Burlington, pp. 761 – 778.

Verbosio, F., Coninck, A. D., Kourounis, D. and Schenk, O. [2017]. Enhancing the scalability of selected inversion factorization algorithms in genomic prediction, *Journal of Computational Science* **22**: 99 – 108.

von Meier, A. [2006]. *Electric Power Systems: A Conceptual Introduction*, Wiley Survival Guides in Engineering and Science, Wiley.

Wächter, A. and Biegler, L. T. [2005]. Line search filter methods for nonlinear programming: motivation and global convergence, *SIAM J. Optim.* **16**(1): 1–31 (electronic).

Wächter, A. and Biegler, L. T. [2006]. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, *Math. Program.* **106**(1, Ser. A): 25–57.

Wang, H., Murillo-Sanchez, C. E., Zimmerman, R. D. and Thomas, R. J. [2007]. On computational issues of market-based optimal power flow, *IEEE Trans. on Power Syst.* **22**(3): 1185–1193.

Wright, M. [2005]. The interior-point revolution in optimization: History, recent developments, and lasting consequences, *Bulletin of the American Mathematical Society* **42**(1): 39–56.

Wright, M. H. [1998]. Ill-conditioning and computational error in interior methods for nonlinear programming, *SIAM J. on Optimization* **9**(1): 84–111.

Wright, S. [1997]. *Primal-Dual Interior-Point Methods*, Society for Industrial and Applied Mathematics.

Zhou, G., Feng, Y., Bo, R., Chien, L., Zhang, X., Lang, Y., Jia, Y. and Chen, Z. [2017]. GPU-accelerated batch-ACPF solution for N-1 static security analysis, *IEEE Transactions on Smart Grid* **8**(3): 1406–1416.

Zimmerman, R. D., Murillo-Sanchez, C. E. and Thomas, R. J. [2011]. MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education, *IEEE Trans. on Power Syst.* **26**(1): 12–19.

Zimmerman, R. and Murillo-Sanchez, C. [2016]. *Matpower 6.0 User's manual*, Power Systems Engineering Research Center.