1 Introduction

1.1 Recent history

Streetlights, subways, the Internet, this book you are reading now – it is difficult to imagine life without such amenities, all enabled by electric power. To support such a vast set of technologies, electric power systems have grown into some of the most complex and expensive machines in existence. While much of this growth resembles an organic process more than deliberate design, the advent of computing is enabling us more and more to direct the evolution of power systems toward greater efficiency, reliability, and versatility.

At the time of writing, the complexity of power systems is poised to take off. This is largely due to shifts toward renewable energy production and the active involvement of power consumers through demand response, as well as our still-developing handle on economic deregulation. To meet these challenges, new computational tools will be developed, and the most ubiquitous computation in power systems is optimization. An objective of this book is to simplify and unify various topics in power system optimization so as to provide a firm foundation for future developments.

At the heart of most power system optimizations are the equations of the steady-state, single-phase approximation to alternating current power flow in a network. Well-known problems like optimal power flow, reconfiguration, and transmission planning all consist of details layered on top of power flow. Nodal prices, a core component of electricity markets, are obtained from the dual of optimal power flow. It is therefore most unfortunate that the power flow equations are nonconvex, making all of these optimizations extremely difficult. We are thus faced with a tradeoff between realistic models that are too hard to solve at practical scales and tractable approximations.

For many years, linear programming (LP) was the most general efficiently solvable optimization class, and so many large-scale power system models were based on linear power flow approximations or even simpler descriptions like network flow or a real power balance. At the other extreme, a number of nonlinear programming (NLP) algorithms were developed for exact, nonconvex models. These approaches invariably encountered difficulty scaling to larger problem sizes due to the underlying NP-hardness of nonconvex optimization. This led some to resort to so-called metaheuristic algorithms, which make little use of problem structure and give little indication of their performance. Beyond their scalability issues, NLP and metaheuristic approaches can be tiresome to implement because they often require the user to program both the

mathematical model and algorithm. On the other hand, one would rarely write their own algorithm to solve an LP because of the many available professional-grade commercial and academic implementations.

In 1984, there was a turning point for convex optimization when Karmarkar invented the first practical, polynomial-time interior point method for LP [1]. Over the next decade, second-order cone programming (SOCP) and semidefinite programming (SDP) emerged as convex generalizations of LP also admitting polynomial-time interior point methods [2]. Of equal importance, SOCP and SDP are now featured in a number of standard software packages, making them similarly user friendly.

The enhanced modeling capabilities of SOCP and SDP brought about an explosion of research applications, some of which can be found in the standard text [3]. In 2006, ripples from the previous twenty years were felt in power systems when power flow in radial networks was posed as an SOCP in Jabr [4] and again in 2008 when an SDP approximation of optimal power flow was developed in Bai, Wei, Fujisawa, and Wang [5]. A substantial body of research has materialized in the short time since then, both theoretically characterizing the new SOCP and SDP power flow approximations and applying them in a variety of power system contexts. For most power system optimization problems, we now have a spectrum of LP, SOCP, SDP, and NLP models to choose from, each with a different balance of realism and scalability.

1.2 Structure and outline

This book only contains models and, with the exception of Chapters 2 and 7, rarely mentions algorithms. This is *not* because the algorithms are not worth knowing or decoupled from modeling; one can *always* do better by formulating optimization models and algorithms jointly. Rather, here this wisdom is applied by formulating models so that they can be solved by certain algorithms. This approach is a luxury we can afford because optimization is a relatively mature field: for a desired level of scalability, it identifies the corresponding tradeoff between efficiency and descriptiveness. Here, this manifests as a hierarchy of convex optimization classes, the main elements of which are LP, SOCP, and SDP. LP is the most efficient and least descriptive, SDP vice versa, and SOCP is in between. As discussed in the previous section, once a model has been formulated within one of these classes, it can be conveniently solved using standard software packages. By tailoring our models to these classes, we arrive at tractable formulations far more easily than if we were to design both model and algorithm from scratch.

The resulting separation between models and algorithms should not be seen as a restriction but as a starting point for further specialization and extension. For example, once a problem has been formulated as an LP, uncertainty can be mechanistically incorporated using robust optimization (Section 7.1.2), or the problem can be split into different chunks for multiple processors or agents (Section 7.2).

A central motif in this book is posing complicated models as layers on top of more basic ones. To avoid rewriting the same constraints over and over, this book makes extensive use of *feasible sets* to package frequently occurring groups of constraints

for concise representation in other problems. Consequently, certain parts of this book are highly cumulative. Each chapter is concluded by a summary highlighting important points and open problems, as well as a small selection of exercises. In addition to those given, which are all analytical, students are of course encouraged to implement each chapter's models and examples using their preferred optimization platform.

The chapter structure is summarized below.

Chapter 2: This chapter provides a minimal introduction to optimization and power system modeling. In particular, it defines LP, SOCP, and SDP, and the tools used to construct convex relaxations. It also derives the quadratic steady-state, single-phase approximation to power flow in a network.

Chapter 3: This chapter defines the basic, nonconvex optimal power flow problem. It then derives classical linear approximations and modern SOC and SD relaxations. The constraints in these models form the foundation of all subsequent chapters in this book.

Chapter 4: This chapter constructs linear, SOC, and SD versions of a number of central optimization problems in power system operations using the approximations from the previous chapter. Here and in the next chapter we encounter a number of mixed-integer constraints, which make these problems challenging even with linear power flow constraints. This chapter also takes brief detours though inventory control and linear quadratic regulation.

Chapter 5: Similar to the last chapter, this chapter constructs infrastructure planning problems around the power flow approximations of Chapter 3. In this chapter, every problem is extremely difficult due to the integer constraints required to describe component installation.

Chapter 6: This chapter discusses electricity markets, in which *nodal prices* are obtained from the dual of optimal power flow. Because each convex approximation has strong duality, prices are guaranteed to support *economic dispatch*. This chapter also discusses why basic economic assumptions never hold in practice and briefly summarizes some game theoretic analyses of market power.

Chapter 7: This final chapter surveys some promising directions for future work.

1.3 On approximations

Every model in this book is an approximation. In fact, every mathematical model ever is an approximation, which inevitably fails to capture some fine physical detail. However, it is worth stating this explicitly here because every model in this book involves an approximation of one particular model: the steady-state, single-phase description of electric power in a network of conductors. We derive this model in Section 2.5 and place it in the context of optimization in Section 3.1.

The steady-state, single-phase description of electric power is a special approximation because it derives from very natural physical assumptions and enables power system engineers to take a large step from simulation to design, which is what this book is all

about. By definition, optimization is the highest (mathematical) form of design. Unfortunately, the steady-state description of electric power isn't quite right for optimization because it is nonconvex. This book attempts to make the mildest further adjustments necessary for this model and those built atop it to enjoy all that optimization has to offer.

With this perspective in mind, it is helpful to remember the following two statements when using this book.

- We would always use a more realistic description of electric power like an unbalanced steady-state or transient model were it practical to do so. So, when it is practical, use one of them and not the convex approximations in this book. We briefly elaborate on this in Section 3.1.1.
- While immensely important, the steady-state description of electric power is not sacrosanct. It is an approximation like all of the other models in this book, which just happens to be (slightly) closer to reality. From this perspective, the convex approximations in this book are no less valid than the model from which they are derived.

References

[1] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '84. New York: ACM, 1984, pp. 302–311.

[2] Y. Nesterov and A. Nemirovski, "Interior point polynomial methods in convex programming," *SIAM Studies in Applied Mathematics*, vol. 13, 1994.

[3] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York: Cambridge University Press, 2004.

[4] R. Jabr, "Radial distribution load flow using conic programming," *IEEE Transactions on Power Systems*, vol. 21, no. 3, pp. 1458 –1459, Aug. 2006.

[5] X. Bai, H. Wei, K. Fujisawa, and Y. Wang, "Semidefinite programming for optimal power flow problems," *International Journal of Electrical Power and Energy Systems*, vol. 30, no. 6–7, pp. 383–392, 2008.

2 Background

This chapter summarizes the basic technical concepts used throughout this book. As stated in the Introduction, this book focuses on modeling, so most algorithmic aspects are left "under the hood." Because this book is intended to appeal to anyone familiar with power systems or optimization, background material on both topics is covered, albeit at the minimum depth necessary to access the later material.

2.1 Convexity and computational complexity

We begin with a few core concepts. A point $x_0 \in X$ is a *global minimum* of the function $f(x)$ over the set $X \subseteq \mathbb{R}^n$ if $f(x_0) \leq f(x)$ for all $x \in X$. If $f(x)$ is continuous and X is compact, which is to say closed and bounded, such a point is guaranteed to exist. x_0 is a *local minimum* of $f(x)$ if there exists an $\epsilon > 0$ for which $f(x_0) \leq f(x)$ for all $x \in X$ satisfying $\|x - x_0\| \leq \epsilon$. All minima of a convex function achieve the same function value and are therefore global. In general, a function may have multiple local and global minima.

To find a global minimum of a convex function, choose a descending algorithm, let it run free, and in a perfect world it will eventually end up there. (In the real world, large problem sizes or bad numerical conditioning can derail any algorithm.) The intuitive simplicity of convexity translates to a genuine computational advantage, which is evinced by the powerful algorithms that exist for convex optimization and the extreme difficulty of nonconvex optimization. This section gives some basic characterizations of convexity and describes the varieties of convex optimization problems encountered in this book. For more comprehensive coverage, the reader is referred to endnotes [1–5], and to endnotes [6, 7] for theoretical treatments of convex functions and sets.

A function f is convex if, for any two points in its domain, x and y,

$$f(\alpha x + (1-\alpha)y) \geq \alpha f(x) + (1-\alpha)f(y) \quad \text{for all } \alpha \in [0,1].$$

This means that any point on the straight line between $(x,f(x))$ and $(y,f(y))$ is greater than or equal to the function value at the corresponding point between x and y. When f is twice-differentiable, it is convex if and only if its Hessian is positive semidefinite:

$$\nabla^2 f(x) \succeq 0.$$

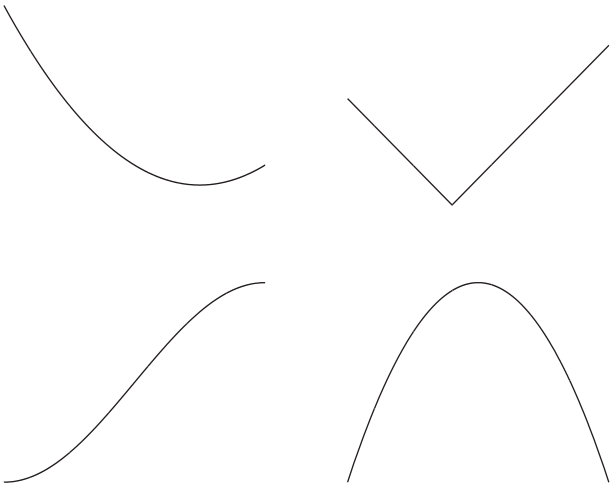

Fig. 2.1 Convex (top) and nonconvex (bottom) functions. The top left function is strictly convex, and the top right is not. The bottom right function is strictly concave.

f is strictly convex if the $\geq$ in the first condition and $\succeq$ in the second are respectively replaced by $>$ and $\succ$. A strictly convex function has a unique global minimum. Some single variable convex and nonconvex functions are shown in Figure 2.1.

A set X is convex if, for any x and y in X, any point $\alpha x + (1-\alpha)y$ with $\alpha \in [0,1]$ is also in X; in other words, all points on the line between x and y are also in X. Convex and nonconvex sets are shown in Figure 2.2. Now consider the set $X = \{x \mid g(x) \leq 0\}$; if g is convex, then so is X. While extremely useful in general, we will virtually never use these characterizations of convexity in this book because we will seek models that are convex by construction and hence do not need to be tested.

Suppose that we want to solve the following optimization problem:

$$\begin{aligned} &\underset{x}{\text{minimize}} && f(x) \\ &\text{subject to} && g_i(x) \leq 0. \end{aligned}$$

If $f(x)$ and each $g_i(x)$ are convex, any minimal solution is a global minimum attaining the lowest possible function value, and we call the above a *convex optimization problem*. We say that a point x is *feasible* if $g_i(x) \leq 0$ for all i, i.e., all constraints are satisfied. Equality constraints $h(x) = 0$ are equivalently expressed by the pair $h(x) \leq 0$ and $h(x) \geq 0$, from which it is evident that the only convex equality constraints are linear.[1]

In power system optimization problems, the objective function is usually some economic cost, and the constraints encode physical requirements derived from first principles. Consequently, we will usually assume approximate, convex objectives while regarding nonconvex constraints as fundamental. A primary focus of this book is therefore the nonconvex constraints arising in power system optimization problems, which

[1] Technically, a linear function is of the form $g(x) = \alpha x$. In optimization, it is standard to use the term "linear" to refer to affine functions of the form $g(x) = \alpha x + \beta$.

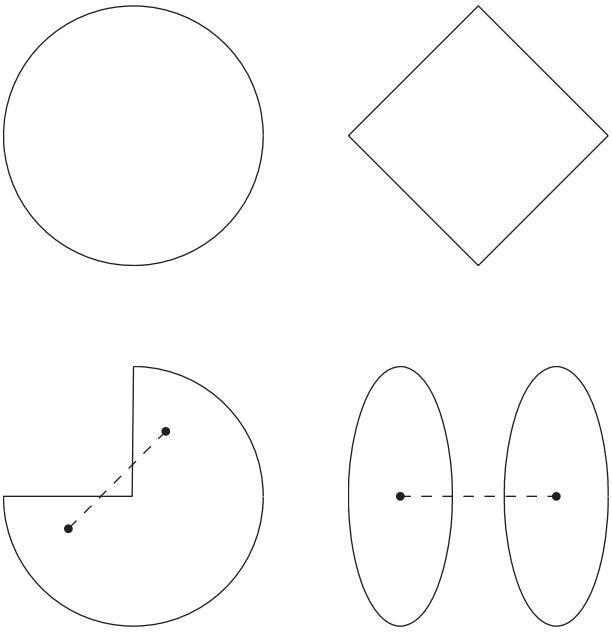

Fig. 2.2 Convex (top) and nonconvex (bottom) sets. Convexity violations are shown for the latter.

are organized into *feasible sets*. The feasible set of an optimization problem is the set of all feasible points, i.e.,

$$\{x \mid g_i(x) \leq 0\}.$$

Convex optimization problems are often efficiently solvable because it is unnecessary to check numerous local minima to find a global minimum. This sort of computational efficiency can be quantified by the concept of non-deterministic polynomial-time hardness, or NP-hardness for short [8–10] (actually, some of the problems we will cover are NP-complete, a subset of the NP-hard problems, but the difference is insignificant for our purposes). On a fundamental level, there is no way to efficiently solve a large instance of an NP-hard problem.

More precisely, suppose the difficulty of a problem grows with some parameter, n. A *polynomial-time* algorithm will solve said problem (terminate with a satisfactory answer) after a number of arithmetic operations, which is a polynomial in n, for example n^q, where q is some positive integer. An *exponential-time* algorithm will require a number of operations that is exponential in n; if you are unconvinced that this is a significant difference, compare q^n and n^q for any fixed q and increasing n (alternatively, try counting to 2^{100}).

In the context of optimization, a polynomial-time algorithm requires a number of operations that is polynomial in both the number of variables and constraints. One reason nonconvex problems can be NP-hard is that the number of local minima may be exponential in either the number of constraints or variables. Although convexity is not

the precise boundary between hard and easy problems, there are broad classes of convex optimization problems that admit polynomial-time algorithms, and nonconvex ones usually do not.

2.2 Optimization classes

This section describes the classes of optimization problems encountered in this book.

2.2.1 Linear and quadratic programming

LP rightfully occupies the central spot among efficient convex optimization frameworks; the reader is referred to endnotes [11–14] for comprehensive treatments. Even Dantzig's simplex algorithm, which is known to have worst-case exponential complexity, remains the most competitive choice for many LPs and is at the foundation of mixed-integer algorithms. QP is LP with an additional quadratic term in the objective, and when the quadratic term is convex, QPs can be solved with similar algorithms and efficiency to LPs. The *standard form* of a generic QP is

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & x^T Q x + c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0. \end{aligned}$$

This is the format processed by most algorithms. Observe that if the rank of A is equal to the length of the vector x, $Ax = b$ is a fully determined system with only one solution. If it is positive, it is feasible and therefore optimal. If $Ax = b$ is overdetermined, no feasible solution exists and the problem is also trivial. It is only when rank A is less than the dimension of x that the feasible set is more than a point and the above optimization is nontrivial. Because all linear constraints are convex, the feasible set of an LP or QP is always convex. The feasible set of a simple LP is shown in Figure 2.3.

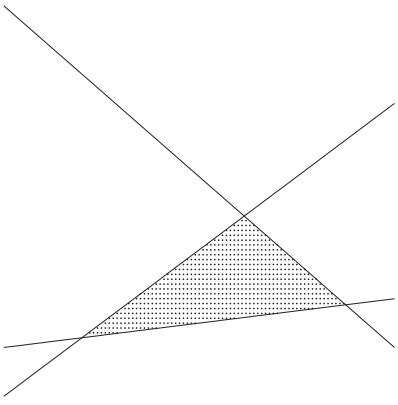

Fig. 2.3 The (convex) polyhedral feasible set described by an LP's inequality constraints.

When Q is a positive semidefinite matrix (which we define in Section 2.2.2 and denote $Q \succeq 0$), this problem is convex. When Q is indefinite or negative semidefinite, this problem is NP-hard [15]. When $Q = 0$, we obtain the standard form LP. All three cases are illustrated in Example 2.1. Putting a model into standard form can be tedious, but fortunately many software-modeling environments exist to automate this task.

The constraints of an LP or QP constitute a convex *polytope*, a set with only flat sides. The simplex algorithm operates by descending the corners or *vertices* of the polytope. In 1984, Karmarkar introduced the first practical polynomial-time LP algorithm [16]. It was an interior point method that, unlike the Simplex method, used Newton's method to navigate the interior of the feasible set and avoided constraints via barrier functions. This was a milestone in optimization and paved the way for polynomial-time algorithms for more general classes of convex optimization problems, in particular SOCP and SDP.

Example 2.1 *A small quadratic program.* Consider a two-dimensional QP with standard form parameters

$$Q = \begin{bmatrix} q & 0 \\ 0 & 0 \end{bmatrix}, \quad c = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad A = [1 \quad 1], \quad b = 1.$$

Observe that these parameters encode the one dimensional, non-standard form QP

$$\begin{aligned} &\underset{y}{\text{minimize}} && qy^2 + y \\ &\text{subject to} && 0 \leq y \leq 1. \end{aligned}$$

If $q \geq 0$, $Q \succeq 0$, and the local minimum $x = [0; 1]$ is the global minimum due to convexity. If $q = 0$, the problem is linear, and the solution is again $x = [0; 1]$. Now suppose that $q = -2$. Then both $x = [0; 1]$ and $x = [1; 0]$ are local minima, and we can ascertain that the latter is the global minimum by comparing the corresponding objective values.

Observe that the only way to find the global minimum when $q = -2$ was to check both local minima. This kind of enumerative procedure may be straightforward in low-dimensional problems but would hardly be viable if x was a thousand-dimensional vector hiding in a polytope with more than 2^{1000} vertices to check. In fact, no efficient procedure exists for general problems of this nature because they are NP-hard.

Now observe that if $-1/2 \leq q \leq 0$, the problem is nonconvex, but $x = [0; 1]$ is the sole local and hence global solution. We are here reminded that convexity is a sufficient condition for global optimality of local optima but not a necessary one.

2.2.2 Cone programming

LP is the most basic type of cone programming. The constraint $x \geq 0$ says that x must occupy the cone of points with all nonnegative coordinates, also known as the nonnegative orthant. More generally, a set X is a cone if for any $x \in X$ and nonnegative

scalar α, $\alpha x \in X$. It is a convex cone if it also satisfies the definition of set convexity in Section 2.1.

This section discusses two convex conic optimization classes: *second-order cone* and *semidefinite programming*. They are successive generalizations of LP and share with it the virtue of polynomial-time interior point methods (but not practical simplex methods). Of course, with greater generality comes lower efficiency, so one would never solve an LP as an SOCP or an SOCP as an SDP.

Second-order cone programming

The SOC, also referred to as the Lorenz or ice cream cone, is defined as the set

$$\left\{(y,t) \in \mathbb{R}^{n+1} : \ \|y\| \leq t\right\}.$$

Clearly, if $\|y\| \leq t$, then $\|\alpha y\| \leq \alpha t$ for any $\alpha \geq 0$, so it satisfies the definition of a cone. Detailed expositions of SOCP are provided in endnotes [17, 18]. We can straightforwardly check its convexity. Suppose (y_1, t_1) and (y_2, t_2) are in the SOC and $\alpha \in [0, 1]$. Then, using the triangle inequality and homogeneity of the two-norm,

$$\begin{aligned}\|\alpha y_1 + (1-\alpha)y_2\| &\leq \alpha \|y_1\| + (1-\alpha)\|y_2\| \\ &\leq \alpha t_1 + (1-\alpha)t_2,\end{aligned}$$

i.e., the point $(\alpha y_1 + (1-\alpha)y_2, \alpha t_1 + (1-\alpha)t_2)$ is also in the SOC.

Setting $y = Ax + b$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and $t = c^T x + d$, $c \in \mathbb{R}^n$, $d \in \mathbb{R}$, we obtain the standard form SOC constraint:

$$\|Ax + b\| \leq c^T x + d. \tag{2.1}$$

An example SOC constraint is shown in Figure 2.4. Notice that if A is zero, this is a linear constraint. If $c = 0$, CQCP is retrieved, which we discuss further in Section 2.2.3.

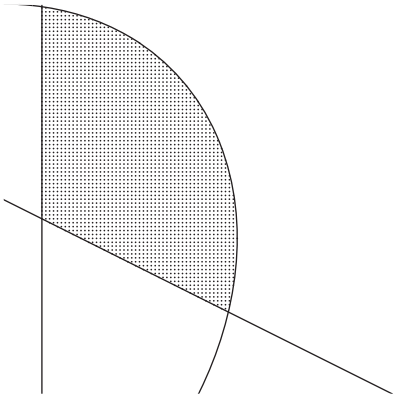

Fig. 2.4 The feasible set described by the linear constraints $x_1 \geq 0$, $1 - x_1 \leq 2x_2$, and an SOC constraint with the standard form parameterization $A = [1, 2; -1, 1]$, $b = [1, -1]^T$, $c = [1, 1]^T$, $d = 2$. Here, x_1 and x_2 are the horizontal and vertical axes, respectively.