CONVEX OPTIMIZATION METHODS FOR ADAPTIVE RADIATION THERAPY

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Anqi Fu
May 2021

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Stephen Boyd, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

John Duchi

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Balasubramanian Narasimhan

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Lei Xing

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Balasubramanian Narasimhan

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

In 2020, over 1.8 million new cases of cancer were diagnosed in the United States, and of these cases, a majority were treated with radiation therapy. Radiation treatment involves fundamental tensions between tumor coverage and damage to surrounding healthy tissue. To mitigate the ill effects, treatment is typically carried out over several weeks, giving the patient time to recover in between sessions. However, this recovery time comes at the risk of tumor proliferation. The clinician's task is to design an individualized treatment plan that adapts to changes in the patient's health caused by these factors, as well as other unforeseen forces, to achieve the best therapeutic outcome.

In this dissertation, we present an optimization-based framework for adaptive radiation treatment planning. We focus on two specific planning challenges: (1) satisfying dose-volume (*i.e.*, percentile) constraints and (2) handling nonlinear patient health dynamics. For each situation, we show how to formulate the treatment planning problem as a nonconvex optimization problem and obtain a good estimate of the solution by solving a series of convex approximations. We demonstrate the effectiveness of our method on several clinical examples. Finally, we release an open-source Python software package that implements our method using generic convex solvers.

The last part of this dissertation concerns applications of optimization beyond radiation therapy. We develop a domain-specific language (DSL) for formulating and solving a broad class of convex optimization problems. Then, we describe an implementation of our DSL in R, a popular programming language for statistical modeling and analysis. Our resulting software package, CVXR, allows users to construct optimization problems in a natural mathematical syntax. CVXR automatically verifies

the problem's convexity and converts it into the standard form required by a specific solver. We illustrate CVXR's modeling framework with a variety of examples drawn from statistics, finance, engineering, and radiation treatment planning.

Acknowledgments

Advisers. First and foremost, I would like to thank my Ph.D adviser, Stephen Boyd. Stephen has been with me at every step of my academic journey. He has spent countless hours revising manuscripts, critiquing slides, discussing research ideas, and patiently explaining concepts from mathematical optimization to electrical circuits. He holds his students to high standards in both research and writing, emphasizing the importance of clarity when communicating our results. His ability to reduce a seemingly intractable problem to a simple, concise formulation is awe-inspiring. I have learned an incredible amount from him about what it means to be an excellent researcher and teacher.

But that's not all. Stephen has also shown me how to be a good citizen of the academic community. He encourages his students to make their research accessible by developing and sharing open-source software implementations of our results. Indeed, it was one of our conversations about accessibility that planted the seed for the CVXR software package, which I present in Chapter 4. He promotes collaboration across disciplines, connecting students with intellectuals and industry professionals from a wide array of backgrounds who would benefit from our work. I consider Stephen to be my role model in scholarly leadership. I am deeply indebted to him for his time and dedication to my education.

I would also like to thank Lei Xing, another one of my advisers who has been with me from the beginning of my journey. Lei welcomed me into his research lab at a critical juncture in my career and generously provided funding while I transitioned from a new student to a productive scholar. I learned a great deal from him about medical

physics and radiation therapy; most of the clinical examples in this dissertation originated from conversations with him. Finally, I thank Balasubramanian Narasimhan for his mentorship, career guidance, and warm encouragement during my graduate school years. He is my co-author and co-developer on the CVXR project, and we spent many evenings ironing out software bugs together in his office. Naras has gone above and beyond as my adviser, connecting me with research and internship opportunities, advocating for me on fellowship applications, and introducing me to his colleagues in the statistics community. I am very grateful for his support.

Committee. I thank my committee members, Sanjay Lall and John Duchi, for their thoughtful discussions and advice on my research, especially the material in Chapter 3.

Collaborators. I've been privileged to work with many talented researchers during the course of my Ph.D. In particular, I am thankful to Barış Ungun, with whom I co-authored the ConRad paper that forms the basis for Chapter 2 of this dissertation, Michael Folkerts and Peng Dong for providing the anonymized clinical datasets for my treatment planning examples, and Junzi Zhang, who I collaborated with on another project involving large-scale convex optimization. I would also like to thank Trevor Hastie, Robert Tibshirani, John Chambers, and David Donoho for their thoughtful feedback on the CVXR paper and software. Thank you as well to Steven Diamond, John Miller, and Paul Kunsberg Rosenfield for their contributions to CVXR's development. I am particularly indebted to Steven for his work on CVXPY; most of CVXR's code, documentation, and examples were ported from his Python library.

Classmates and colleagues. I give thanks to my fellow classmates and alumni of Stephen Boyd's research group, who have taught me so much. Thank you especially to Madeleine Udell, Ernest Ryu, Jaehyun Park, Nicholas Moehle, Enzo Busseti, Youngsuk Park, Jonathan Tuck, Jongho Kim, Shane Barratt, and Akshay Agrawal. I also thank my colleagues Julie Josse and Joseph Salmon for inviting me to France and introducing me to the optimization community there. Through them, I made

many fruitful connections with researchers across Europe.

Friends. I am grateful to my friends Elaine Baculi, Sabrina Weiss, and Jeanine Brown for their continual moral support. They kept me grounded throughout my graduate school career, reminding me to take time off to rest and enjoy life. I am especially thankful to Elaine for all the sweets she brought me to fuel my late-night writing sessions – large parts of this dissertation owe their existence to her pastries!

Family. Lastly and most importantly, I would like to thank my parents, Han Ni and Richard Fu, and my sister, Betsy Fu, for their unconditional love and support. They celebrated my successes with me and commiserated over my setbacks at every step of my journey. Their empathy, guidance, and unwavering belief in me formed the bedrock of my academic career. I would not be where I am today without them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The bulk of this dissertation concerns the use of optimization in radiation therapy. External beam radiation therapy is the treatment of diseased tissue with beams of ionizing radiation delivered from a source outside the patient. When radiation passes through the patient, it damages both healthy and diseased tissue. A treatment plan must be carefully designed to minimize harm to healthy organs, while delivering enough dose to kill the diseased cells. With recent hardware advances, delivery beams can be positioned and shaped with sophistication, and clinicians increasingly rely on optimization techniques to guide their treatment decisions. We focus on one part of the treatment planning process: the selection of an optimal intensity profile for every radiation beam.

The basic radiation treatment planning problem is convex and tractable. However, additional clinical objectives, penalty terms, and constraints render it nonconvex. In Chapter 2, the nonconvexity we address comes from dose-volume constraints, or constraints on the dose delivered to a percentile of an anatomical structure. We handle such constraints by replacing them with a convex restriction and using the solution to the restricted problem to obtain a plan that meets the dose-volume constraints with minimal gap. The material in this chapter is based on joint work with Barış Ungun, Lei Xing, and Stephen Boyd in Fu et al. (2019). My main contributions to this project were the mathematical modeling, algorithm characterization, and write-up.

In Chapter 3, we consider the problem of radiation treatment planning over time.

The nonconvexity we address in this setting arises from patient health dynamics that are nonlinear. We formulate the dynamic treatment planning problem as an optimal control problem and show how to obtain an approximate solution by solving a sequence of convex optimization problems. We then propose a solution method based on the operator splitting algorithm ADMM. Our method is fast, scalable, and robust, adapting readily to changes in the patient's condition between treatment sessions. The material in this chapter is drawn from joint work with Lei Xing and Stephen Boyd in Fu et al. (2021). I co-developed the planning algorithm, implemented it in Python, and ran all the experiments.

In Chapter 4, we shift gears to study convex optimization beyond radiation therapy. Specifically, we focus on the design of a domain-specific language (DSL) for disciplined convex optimization in R, a programming language widely used by statisticians. We describe our implementation of CVXR, an object-oriented R package that allows users to formulate and solve a broad class of convex optimization problems. We demonstrate CVXR's efficacy with several examples from statistics, finance, engineering, and medicine. The material in this chapter is adapted from joint work with Balasubramanian Narasimhan and Stephen Boyd in Fu et al. (2020a). I was the principal software developer and tester of CVXR, along with the author of a number of examples.

Chapter 2

Radiation Treatment Planning with Dose-Volume Constraints

2.1 Introduction

In this chapter, we present a convex optimization framework for radiation treatment planning with dose constraints. We describe a method for uniformly handling mean dose, maximum dose, minimum dose, and dose-volume (*i.e.*, percentile) constraints as part of our convex formulation. Since dose-volume constraints are nonconvex, we replace them with a convex restriction. This restriction is, by definition, conservative; to mitigate its impact on the objective, we develop a two-pass planning algorithm that allows each dose-volume constraint to be met exactly on a second pass if its corresponding restriction is feasible on the first pass. In another variant, we add slack variables to each dose constraint to prevent the problem from becoming infeasible when the user specifies an incompatible set of constraints or when the constraints are made infeasible by our restriction. Finally, we introduce ConRad, a Python-embedded open-source software package for convex radiation treatment planning. ConRad implements the methods described above and allows users to construct and plan cases through a simple interface.

Radiation treatment planning is a well-studied problem; see Shepard et al. (1999) for a comprehensive survey of several problem formulations in the literature, including

linear (Rosen et al., 1991; Hölder, 2003) and quadratic programming models (J. et al., 1990; Xing and Chen, 1996; Xing et al., 1998). Generally, linear models minimize the weighted sum of doses or the maximum deviation from a prescribed dose, while quadratic models minimize the weighted sum of squared difference between actual and prescribed dose. These formulations incorporate linear bounds on the dose to each structure. Solutions can be rapidly found using various interior point methods, such as primal-dual (Aleman et al., 2010), projected gradient (Aleman et al., 2013), and interior point constraint generation (Oskoorouchi et al., 2011).

To address conflicting clinical goals, researchers have proposed models with multiple objectives and constraints. By varying the weight on each objective, one can produce a set of solutions on the Pareto frontier (Hamacher and Küfer, 2002; Halabi et al., 2006a). A large number of treatment evaluation criteria can be transformed into convex criteria within this framework (Romeijn et al., 2004). Although multi-objective optimization offers flexibility, calculating thousands of points on the Pareto frontier proves computationally inefficient in practice, and expert judgment is still required to select a clinically acceptable plan from the set of mathematically optimal plans.

All the methods discussed so far hinge on a convex problem formulation. However, many clinically relevant constraints are nonconvex. One such type is the dose-volume constraint, which bounds the dose delivered to a given percentage of a patient's anatomy (Zarepisheh et al., 2014). A review of some models for handling this class of constraints is provided in Ehrgott et al. (2008). The simplest approach is to add a nonlinear, volume-sensitive penalty to the objective function (Cho et al., 1998; Spirou and Chui, 1998). Then, a local search algorithm, such as the conjugate gradient method (Xing and Chen, 1996; Xing et al., 1998; Shepard et al., 2000b) or simulated annealing (Webb, 1989, 1992; Mageras and Mohan, 1993), is used to solve the optimization problem. Unfortunately, since this formulation is nonconvex, these algorithms often produce a local minimum, resulting in a sub-optimal treatment plan (Deasy, 1997; Wu and Mohan, 2002). Another method is to directly model the dose-volume constraint with a set of binary decision variables. Each variable indicates whether a voxel should be included in the fraction of a structure's volume that must

fulfill the dose bound (Langer et al., 1990; Lee et al., 2000, 2003). Given the dimensions of patient data, this results in a large-scale mixed-integer programming problem, which is prohibitively expensive to compute for most clinical cases.

A more promising approach is to replace each dose-volume constraint with a convex approximation. This allows users to take advantage of large-scale convex optimization algorithms to quickly generate treatment plans. For instance, Halabi et al. (2006a) substitutes a ramp function for the indicator that a particular voxel violates its desired dose-volume threshold, then penalizes the total number of voxel violations in the objective. Other researchers have employed the conditional value-at-risk (CVaR), a metric that represents the average tail loss in a probability distribution (Rockafellar and Uryasev, 2000). It is convex in the loss variable and thus offers a computationally suitable alternative to the dose-volume constraint. In the recent literature, CVaR has been used to formulate linear constraints on the average dose in the upper and lower tails of a structure's dose distribution, leading to significant improvements in treatment plans (Romeijn et al., 2003, 2006; Chan et al., 2014). However, CVaR functions are parametric, and implementations of this model require a heuristic search over the parameter space to obtain a good approximation of the dose-volume constraint (Ahmed et al., 2010).

Perhaps the method most similar to ours is Zarepisheh et al. (2013). In this paper, the authors propose constraining the dose moments to equal those of the desired dose-volume histogram curve. They derive a convex relaxation of these constraints, then solve their treatment planning problem in two phases: the first phase adds slack variables to the moment bounds, so a solution is always feasible, while the second phase tightens these bounds in order to improve upon plan quality whenever possible. Using only three moments, their technique is able to closely match the reference histogram curves in a prostate cancer case.

2.2 Problem description

During external beam radiation therapy, ionizing radiation travels through a patient, depositing energy along the beam paths. Radiation damages both diseased and

healthy tissue, but clinicians aim to damage these tissues differentially, exploiting the fact that cancer cells typically have faulty cell repair mechanisms and exhibit a lower tolerance to radiation than healthy cells. The goal is to focus radiation beams such that enough dose is delivered to kill diseased tissue, while avoiding as much of the surrounding healthy organs as possible. The clinician separates these structures into one or more planning target volumes (PTVs) to be irradiated at a prescribed dose level and several organs-at-risk (OARs) to spare from radiation.

Before treatment, the patient is positioned on a couch. Photon, electron, proton, or heavier particle beams are generated with a particle accelerator and coupled to a mechanized gantry that contains additional hardware components, which shape and focus the beams. The gantry typically rotates around one central axis (but may have additional rotational and translational degrees of freedom (Mackie et al., 1993; Glide-Hurst et al., 2013; Jr et al., 1998)), so that by controlling the gantry and couch, beams can be delivered from almost any angle and location around the patient.

Delivery strategies vary from using a large number of apertures (beam shapes) delivered sequentially from a few beam angles, as in intensity-modulated radiation therapy (IMRT), to calculating a single optimal aperture at a large number of angles, as in volumetric modulated arc therapy (VMAT). For a given delivery strategy, the goal of treatment planning is to determine the optimal beam angles, shapes and intensities that most closely approximate a desired dose distribution to the targeted volumes. In this work, we consider the task of optimizing intensities for a given set of beams of known positions and shapes, *i.e.*, calculating optimal beam weights. This is applicable to the fluence map optimization (FMO) step in IMRT planning, the FMO step in direct aperture optimization for modalities such as VMAT, 4π, or SPORT (Bedford, 2009; Dong et al., 2013; Li and Xing, 2013), as well as inverse planning problems for other common modalities such as stereotactic radiosurgery (Shepard et al., 2000a; Schweikard et al., 2006) or proton beam therapy (Oelfke and Bortfeld, 2001).

2.3 Clinical planning

2.3.1 Dose physics

Prior to treatment, medical images—such as CT, MRI and PET scans—are collected to form a three-dimensional image of the patient's anatomy. This representation is discretized into regular volume elements, or voxels. The anatomy is then delineated by clinicians into various structures, and the dose to each structure is considered during planning. Although the structure contours drawn by clinicians may overlap, in this work, we associate each voxel with a single structure.

Dose calculation algorithms range from analytical approximations to Monte Carlo simulations, but in all cases, they provide a model with a linear mapping from beam intensities to delivered voxel doses. The dose within each voxel is assumed to be uniform. For each candidate beam, we have an aperture shape that may be further subdivided, *e.g.*, into regular rectangular subdivisions called beamlets. The intensities of these beams (or beamlets) are represented in a vector. A patient-specific dose deposition matrix maps this vector of radiation intensities to the vector of doses delivered per voxel.

2.3.2 Dose objectives

Given a fixed number of candidate beams, our goal is to determine the beam intensities that satisfy a clinical objective defined in terms of the dose delivered to each voxel in the patient anatomy.

Every PTV is prescribed a desired dose, which we wish to deposit uniformly throughout the target. Delivering too high or too low a dose of radiation has different clinical consequences, so we introduce separate underdose and overdose penalties for every PTV. In the case of OARs, a lower dose is always preferable, so we penalize any dose above zero and omit an underdose penalty term. In addition to clinical considerations, such as the patient's medical history and past courses of radiation therapy, different organs usually exhibit different levels of sensitivity to radiation. For these reasons, we allow the penalties for each OAR to be scaled independently,

allowing the planner to adjust the relative importance of meeting the dose targets for each structure separately.

We apply the penalty associated with each structure to every voxel in that structure, and the objective function of our treatment planning problem tallies these dose penalties over all voxels in a patient's anatomy.

2.3.3 Dose constraints

In addition to dose penalties, we allow for hard constraints on the amount of radiation delivered to portions of the patient anatomy. For example, the clinician may only consider plans in which the spinal column receives a dose below a certain level because any more will increase the likelihood of injury beyond an acceptable limit. Basic constraints of this nature take the form of bounds on the mean, minimum, and maximum dose to a structure. More generally, bounds can be enforced on the dose to a fraction of the voxels in a structure. These dose-volume constraints restrict the relative volume that receives radiation beyond a particular threshold, giving the clinician precise control over the dose distribution. This is especially important when sparing OARs, since some organs are able to sustain high levels of uniform radiation, while others will fail unless the radiation is contained to a small fraction of the tissue.

Clinicians typically use a dose-volume histogram (DVH) to assess the quality of a treatment plan. For every structure, the DVH specifies the percentage of its volume that receives at least a certain dose. A point (x, y) on the curve indicates that $y\%$ of the total voxels in the structure receives a dose of at least x Gy. Ideally, we want our structures to receive exactly the prescribed dose throughout their volumes. If the prescription is d Gy, then our optimal DVH curve for the PTV is a step function with a drop at $(d, 100)$, and our optimal DVH for each OAR exhibits a drop at $(0, 100)$.

Dose constraints restrict the shape and location of points on the DVH curve. In Figure 2.1, a lower dose-volume constraint, $D(90) \geq 60$, is represented by the right-facing arrow centered at $(60, 90)$. This ensures that a minimum of 90% of the structure's volume receives at least 60 Gy, *i.e.*, $y \geq 90$ along the vertical line $x = 60$. The PTV's DVH curve is pushed rightward by this type of constraint. Similarly,

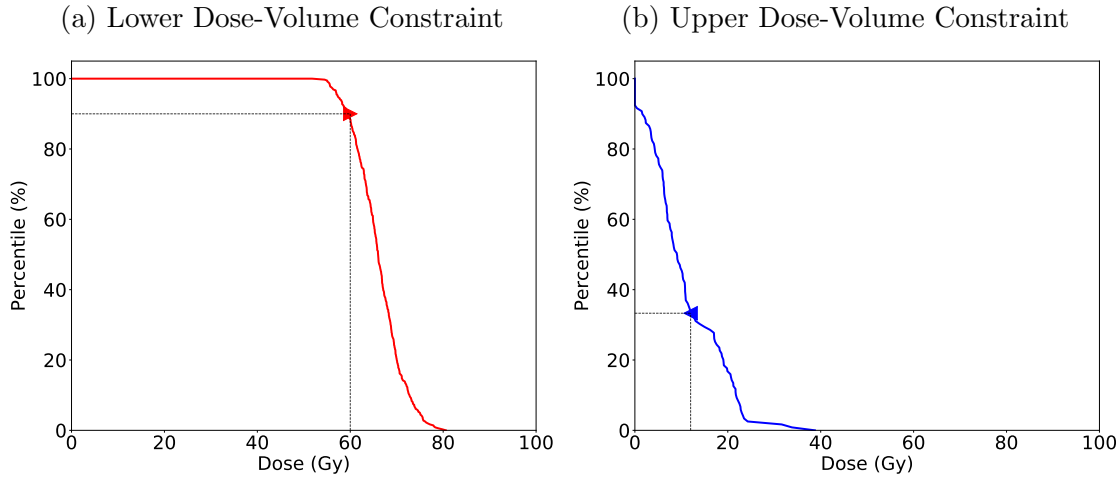


Figure 2.1: (a) A lower DVH constraint ensures at least 90% of the structure's volume receives at least 60 Gy. The dotted line intersects the curve at $(60, 90)$. (b) An upper DVH constraint allows at most 33% of the volume to receive at least 12 Gy.

an upper dose-volume constraint, $D(33) \leq 12$, is labeled with a left-facing arrow at $(12, 33)$, which pushes the OAR's DVH curve leftward, representing the restriction that $y \leq 33$ along the line $x = 12$. Together, the DVH curves and their respective dose constraints enable the clinician to easily visualize trade-offs when formulating a treatment plan.

2.4 Convex formulation

Consider a case with m voxels inside a patient volume and n candidate treatment beams. Our goal is to determine the beam intensities $x \in \mathbf{R}^n_+$ that deliver a vector of voxel doses $y \in \mathbf{R}^m_+$, which meet a set of clinical objectives. We are given a case-specific dose influence matrix $A \in \mathbf{R}^{m \times n}_+$ that approximates the relationship between the beams and doses linearly as $y = Ax$. We refer to the rows of A as $a_i \in \mathbf{R}^n_+$ for

$i = 1, \ldots, m$. The basic inverse treatment planning problem is of the form

$$\begin{array}{ll} \text{minimize} & f(y) \\ \text{subject to} & y = Ax \\ & x \succeq 0 \end{array}$$

with respect to x and y, where $f : \mathbf{R}^m \to \mathbf{R}$ is a convex loss function chosen to penalize voxel doses based on the goals of the clinician. Here, the inequality on x is understood to be applied element-wise. In a typical case, a patient is prescribed a treatment plan that can be characterized by a vector of doses $d \in \mathbf{R}^m_+$ to each voxel. Our function f then penalizes the deviation of the calculated dose y from the prescribed dose d, taking into account the different structures inside a patient.

In our formulation, we consider a loss function $f(y) = \sum_{i=1}^m f_i(y_i)$ where $y_i = a_i^T x$ and each f_i is a piecewise-linear function

$$f_i(y_i) = w_i^-(y_i - d_i)_- + w_i^+(y_i - d_i)_+.$$

The parameters w_i^- and w_i^+ are the non-negative weights on the underdose and overdose, respectively. This penalty structure is common in the literature (Lim and Cao, 2012; Chen et al., 2012) and provided the most efficient software implementation.

Prior to treatment planning, the m voxels in a patient volume are grouped into S distinct, non-overlapping sets representing the planning target volume (PTV), organs-at-risk (OARs), and generic non-target tissue (often labeled "body"). Each set $\mathcal{V}_s$ contains all the voxel indices i within a corresponding internal structure with index s. Together, $\{\mathcal{V}_s\}_1^S$ forms a partition of the patient volume, *i.e.*, $\bigcup_1^S \mathcal{V}_s$ covers all voxel indices and $\mathcal{V}_{s_1} \bigcap \mathcal{V}_{s_2} = \emptyset$ for $s_1 \neq s_2$. We assume the indices are ordered such that $s = 1, \ldots, P \leq S$ are targets and the rest non-targets.

For simplicity, we choose our voxel doses and penalties to be uniform within each structure. We let d_s represent the prescribed dose, and w_s^- and w_s^+ the underdose and overdose penalties for all voxels $i \in \mathcal{V}_s$. The loss function for our basic inverse

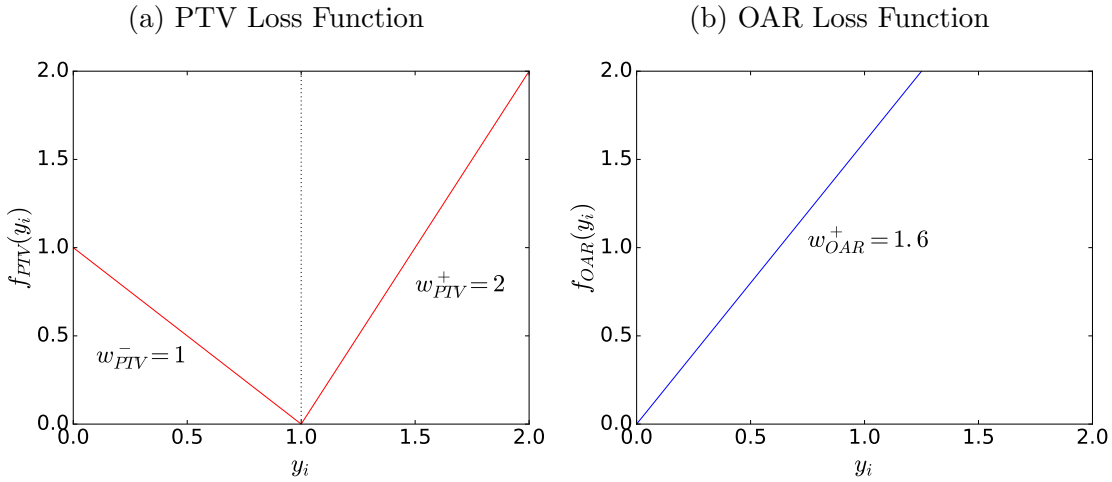


Figure 2.2: (a) The loss function for a PTV prescribed $d_s = 1$ with penalties $w_s^- = 1$ and $w_s^+ = 2$, and (b) the loss function for an OAR with penalty $w_{s'}^+ = 1.6$.

planning problem is $f(y) = \sum_{s=1}^{S} f_s(y_i)$ where

$$f_s(y_i) = \sum_{i \in \mathcal{V}_s} f_i(y_i) = \sum_{i \in \mathcal{V}_s} \{w_s^-(y_i - d_s)_- + w_s^+(y_i - d_s)_+\}.$$

A non-target structure s is always prescribed a dose of zero, and since $y \geq 0$, its individual loss simplifies to $f_i(y_i) = w_s^+ y_i$. Thus, its only contribution to the objective is through its total dose. An example of these loss functions is given in Figure 2.2. We can collapse the sum of non-target losses into a single linear term,

$$\sum_{s=P+1}^{S} f_s(y_i) = \sum_{s=P+1}^{S} w_s^+ \left(\sum_{i \in \mathcal{V}_s} y_i \right) = \sum_{s=P+1}^{S} w_s^+ z_s = c^T z,$$

where $c = (w_{P+1}^+, \ldots, w_S^+)$ and $z = \left(\sum_{i \in \mathcal{V}_{P+1}} y_i, \ldots, \sum_{i \in \mathcal{V}_S} y_i \right)$. Our objective is then

$$f(y) = \sum_{s=1}^{P} \sum_{i \in \mathcal{V}_s} \{w_s^-(y_i - d_s)_- + w_s^+(y_i - d_s)_+\} + c^T z.$$

This formulation is closely related to quantile regression (Davino et al., 2013). In

the latter, we minimize $\sum_i \phi(y_i - v - d_i)$ with respect to (x, v) where

$$\phi(u) = \tau(u)_+ + (1-\tau)(u)_- = \frac{1}{2}|u| + \left(\tau - \frac{1}{2}\right) u$$

is the tilted ℓ_1 penalty with $\tau \in (0,1)$. For our inverse planning problem, the residual $r_i := y_i - v - d_i$ can be interpreted as the difference between calculated and desired doses, allowing for a uniform dose offset $v \in \mathbf{R}$ within each structure. We rewrite our individual loss as

$$\begin{aligned} f_i(y_i - v) &= w_i^-(r_i)_- + w_i^+(r_i)_+ \\ &= (w_i^- + w_i^+)\left(\tfrac{w_i^-}{w_i^- + w_i^+}(r_i)_- + \tfrac{w_i^+}{w_i^- + w_i^+}(r_i)_+\right) \\ &= (w_i^- + w_i^+)\left(\tfrac{1}{2}|r_i| + \left(\tfrac{w_i^+}{w_i^- + w_i^+} - \tfrac{1}{2}\right) r_i\right), \end{aligned}$$

and the loss function becomes

$$f_s(y_i - v) = \sum_{i \in \mathcal{V}_s} f_i(y_i - v) = (w_s^- + w_s^+) \sum_{i \in \mathcal{V}_s} \left(\frac{1}{2}|r_i| + \left(\tau_s - \frac{1}{2}\right) r_i\right),$$

where $\tau_s := \frac{w_s^+}{w_s^- + w_s^+} \in (0,1)$. For $r_i \neq 0$, the first order condition with respect to v is

$$\frac{\partial f_s(y_i - v)}{\partial v} = \tau_s |\{i : r_i > 0\}| - (1 - \tau_s)|\{i : r_i < 0\}| = 0,$$

which implies $\tau_s |\mathcal{V}_s| = |\{i : r_i < 0\}|$, *i.e.*, in a given structure s, the τ_s-quantile of optimal residuals is zero. Although our original loss does not include v, we can use this as a rule of thumb for selecting relative dose penalties (w_s^-, w_s^+).

2.5 Dose constraints

2.5.1 Percentile

A percentile constraint, otherwise known as a dose-volume constraint, bounds the dose delivered to a given percentile of a patient structure. This allows us to set a limit on the fraction of total voxels that are under- or overdosed with respect to a

user-provided threshold. For clinicians, this provides a way to shape the dose-volume histogram directly rather than by searching through combinations of objective weights to achieve desired dose statistics. Given a structure s and dose vector y, let $D_s(p, y)$ represent the minimum dose delivered to p percent of all voxels in s, *i.e.*, $D_s(p, y)$ is the greatest lower bound on the dose received by $p\%$ of the tissue.

To formalize this notion, we define an exact value count function $v_s : \mathbf{R}_+^m \times \mathbf{R}_+ \to \mathbf{Z}_+$, which computes the total number of voxels $i \in \mathcal{V}_s$ that receive a dose above $b \in \mathbf{R}_+$. Let $g(u) = \mathbb{1}\{u \geq 0\}$, then

$$v_s(y, b) = \sum_{i \in \mathcal{V}_s} \mathbb{1}\{y_i \geq b\} = \sum_{i \in \mathcal{V}_s} g(y_i - b)$$

and our p-th percentile dose is

$$D_s(p, y) = \max\{b \in \mathbf{R}_+ : v_s(y, b) \geq \phi_s(p)\} \quad \text{where} \quad \phi_s(p) := \frac{p}{100}|\mathcal{V}_s|.$$

Observe that $D_s(p, y) \geq 0$ is finite and weakly decreasing in p.

Our goal is to bound $D_s(p, y)$. For example, we may want at least 30% of the voxels in structure s to receive a dose above 25 Gy; this is identical to $D_s(30, y) \geq 25$. Let $\ell < u$ be non-negative scalar values. An lower dose-volume constraint, $D_s(p, y) \geq \ell$, requires the number of voxels in s that receive a dose above ℓ to be at least $p\%$ of the total voxels in the structure. Similarly, an upper dose-volume constraint, $D_s(p, y) \leq u$, requires the number of voxels $i \in \mathcal{V}_s$ with a dose above u to be at most $p\%$ of voxels in $\mathcal{V}_s$, or equivalently, at least $100 - p\%$ of the voxels to receive a dose under u. Thus, the inequalities

$$D_s(p, y) \leq u \quad \Leftrightarrow \quad v_s(y, u) \leq \phi_s(p) \quad \Leftrightarrow \quad v_s(-y, -u) \geq \phi_s(100 - p)$$

are equivalent, as are

$$D_s(p, y) \geq \ell \quad \Leftrightarrow \quad v_s(y, \ell) \geq \phi_s(p) \quad \Leftrightarrow \quad v_s(-y, -\ell) \leq \phi_s(100 - p).$$

In general, this is a hard combinatorial problem: the brute force approach for a

single upper dose-volume constraint, for example, is to solve all $\binom{|\mathcal{V}_s|}{\phi}$ convex problems obtained by choosing subsets of $\phi = \lceil \phi_s(p) \rceil$ voxels to constrain below u, which is prohibitively large given the size of patient geometries.

2.5.2 Mean, minimum, and maximum

In a few special cases, we can set convex constraints on the dose. Let the average, minimum, and maximum dose delivered to all voxels in structure s be

$$D_s^{\text{avg}}(y) = \frac{1}{|\mathcal{V}_s|} \sum_{i \in \mathcal{V}_s} y_i, \quad D_s^{\min}(y) = \min_{i \in \mathcal{V}_s} \{y_i\}, \quad D_s^{\max}(y) = \max_{i \in \mathcal{V}_s} \{y_i\}.$$

A lower bound $b \in \mathbf{R}_+$ on the minimum dose is equivalent to requiring $y_i \geq b$ for all $i \in \mathcal{V}_s$, and similarly for an upper bound on the maximum dose. Thus, we can enforce linear constraints on these dose statistics in our problem. Our nonconvex formulation with exact dose-volume constraints is

$$\begin{array}{lll} \text{minimize} & f(y) & \\ \text{subject to} & y = Ax & \\ & x \succeq 0 & \\ & \ell_{s,k} \leq D_s(p_{s,k}, y) \leq u_{s,k}, & k = 1, \ldots, K_s, \quad s = 1, \ldots, S \\ & \ell_s^{\text{avg}} \leq D_s^{\text{avg}}(y) \leq u_s^{\text{avg}}, & s = 1, \ldots, S \\ & D_s^{\min}(y) \geq \ell_s^{\min}, & s = 1, \ldots, S \\ & D_s^{\max}(y) \leq u_s^{\max}, & s = 1, \ldots, S, \end{array} \tag{2.1}$$

where (x, y) are our variables, and for each structure s, we index the parameters of its dose-volume constraints with $k = 1, \ldots, K_s$.

2.5.3 Convex restriction

To address the nonconvexities in problem (2.1), we introduce a convex restriction that provides an effective heuristic for satisfying the dose-volume constraints. Our restricted constraint overestimates the number of voxels that are underdosed with

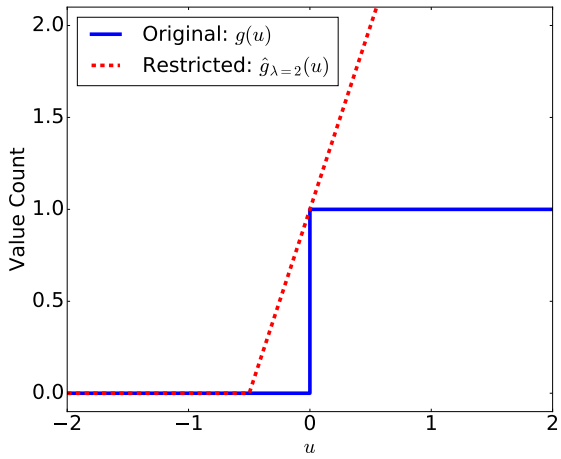


Figure 2.3: The indicator function $g(u)$ (solid) and hinge loss $\hat{g}_\lambda(u)$ with $\lambda = 2$ (dashed). Note that $\hat{g}_\lambda(u) \geq g(u)$ for all $u \in \mathbf{R}$, so the hinge loss provides a convex restriction on the dose-volume constraint.

respect to d by replacing the indicator g in v with a family of hinge loss functions

$$\hat{g}_\lambda(u) = (1 + \lambda u)_+ = \max(1 + \lambda u, 0),$$

parametrized by $\lambda > 0$, giving us a restricted value count for structure s of

$$\hat{v}_s(y, b; \lambda) = \sum_{i \in \mathcal{V}_s} \hat{g}_\lambda(y_i - b) = \sum_{i \in \mathcal{V}_s} (1 + \lambda(y_i - b))_+.$$

If $u > 0$, then $g(u) = 1 < 1 + \lambda u = \hat{g}_\lambda(u)$, and if $u \leq 0$, then $g(u) = 0 \leq \hat{g}_\lambda(u)$. Hence, $g(u) \leq \hat{g}_\lambda(u)$ for all $u \in \mathbf{R}$ and $\lambda > 0$ (Figure 2.3). Evaluating at $u_i = y_i - b$ and summing over all voxels $i \in \mathcal{V}_s$, we obtain

$$v_s(y, b) = \sum_{i \in \mathcal{V}_s} g(y_i - b) \leq \sum_{i \in \mathcal{V}_s} \hat{g}_\lambda(y_i - b) = \hat{v}_s(y, b; \lambda).$$

An upper bound on the restricted value count at a given point thus ensures the exact value count is bounded above as well.

We can guarantee our dose-volume constraints hold by enforcing specific limits

on $\hat{v}$. If $\hat{v}_s(y, u; \lambda) \leq \phi_s(p)$, then $v_s(y, u) \leq \phi_s(p)$, and the upper dose-volume constraint, $D_s(p, y) \leq u$, is satisfied. Similarly, $\hat{v}_s(-y, -\ell; \lambda) \leq \phi_s(100 - p)$ implies that $D_s(p, y) \leq \ell$. To simplify notation, we rewrite $\hat{v}_s(y, b; \lambda) \leq \phi$ as

$$\sum_{i \in \mathcal{V}_s} (1 + \lambda(y_i - b))_+ \leq \phi.$$

Since $\lambda > 0$, we can divide both sides of the inequality by λ. Letting $\alpha := \frac{1}{\lambda}$ and gathering all terms on the left-hand side, we obtain the inequality

$$\sum_{i \in \mathcal{V}_s} (\alpha + (y_i - b))_+ - \alpha\phi \leq 0.$$

The left-hand side of this inequality is a sum of convex functions of (α, y), and hence convex. Note that while λ was a parameter of our restricted value count function, $\alpha > 0$ can be an optimization variable, since the left-hand term is jointly convex in α and y. Additionally, we can replace the constraint $\alpha > 0$ with $\alpha \geq 0$ because when $\alpha = 0$, the constraint simplifies to $(y_i - b)_+ \leq 0$, which is equivalent to $y_i \leq b$ for all $i \in \mathcal{V}_s$. Certainly in this case, the condition $D_s(p, y) \leq b$ holds. Thus, by defining the functions

$$\hat{D}_s^+(p, y, b, \alpha) = \sum_{i \in \mathcal{V}_s} (\alpha + (y_i - b))_+ - \alpha\phi_s(p)$$

for upper constraints and

$$\hat{D}_s^-(p, y, b, \alpha) = \sum_{i \in \mathcal{V}_s} (\alpha - (y_i - b))_+ - \alpha\phi_s(100 - p),$$

for lower constraints, each convex restriction can be represented by inequalities in terms of these functions: for $\alpha \geq 0$, $\hat{D}_s^+(p, y, u, \alpha) \leq 0$ implies $D_s(p, y) \leq u$, and

$\hat{D}_s^-(p, y, \ell, \alpha) \leq 0$ implies $D_s(p, y) \geq \ell$. Our convex formulation with restricted dose-volume constraints is

$$\begin{array}{lll} \text{minimize} & f(y) & \\ \text{subject to} & y = Ax & \\ & x \succeq 0, \quad \alpha \succeq 0 & \\ & \hat{D}_s^+\left(p_{s,k}, y, u_{s,k}, \alpha_{s,k}^{(u)}\right) \leq 0, & k = 1, \ldots, K_s^{(u)}, \quad s = 1, \ldots, S \\ & \hat{D}_s^-\left(p_{s,k}, y, \ell_{s,k}, \alpha_{s,k}^{(\ell)}\right) \leq 0, & k = 1, \ldots, K_s^{(\ell)}, \quad s = 1, \ldots, S \\ & \ell_s^{\text{avg}} \leq D_s^{\text{avg}}(y) \leq u_s^{\text{avg}}, & s = 1, \ldots, S \\ & D_s^{\min}(y) \geq \ell_s^{\min}, & s = 1, \ldots, S \\ & D_s^{\max}(y) \leq u_s^{\max}, & s = 1, \ldots, S, \end{array} \tag{2.2}$$

where for every structure s, we index the parameters of its upper dose-volume constraints with $k = 1, \ldots, K_s^{(u)}$, and its lower dose-volume constraints with $k = 1, \ldots, K_s^{(\ell)}$. We include a separate optimization variable, $\alpha_{s,k}$, in each dose-volume constraint to represent the inverse slope of its convex restriction and stack these variables in a vector $\alpha := (\alpha^{(\ell)}, \alpha^{(u)})$. Optimizing over α in addition to (x, y) ensures we obtain the best hinge loss approximation to the value count function. The above formulation is a restriction of our original problem: if (x, y, α) is feasible for problem (2.2), then (x, y) is feasible for problem (2.1).

2.6 Refinements

2.6.1 Two-pass refinement

A solution (x^*, y^*, α^*) to problem (2.2) satisfies our restricted dose-volume constraints, so it is feasible for our original problem (2.1) with exact dose-volume constraints. However, since the convex restriction enforces an upper bound on the restricted value count function $\hat{v}$, the feasible set of problem (2.2) is a subset of the feasible set of problem (2.1), and (x^*, y^*) may not be optimal for the latter. One way to improve our solution is to bound only the minimum number of voxels in each structure required to satisfy the dose-volume constraint. A good heuristic is to select those voxels i that

receive a dose y_i^*, which satisfies the associated dose-volume bound by the largest margin, and re-solve the problem with the convex restriction replaced by bounds on just these voxels. The solution of this second pass, (x^{**}, y^{**}), will achieve an objective value $f(y^{**}) \leq f(y^*)$ while still satisfying our exact dose-volume constraints.

To make this precise, consider the lower dose-volume constraint $D_s(p, y) \geq \ell$. This is equivalent to $y_i \geq \ell$ for at least $\phi_s(p)$ voxels in structure s. Given y^* from our first pass optimization, we compute the margin $\xi_i^* = (y_i^* - \ell)$ and select the $q_s = \lceil \phi_s(p) \rceil$ voxels $i \in \mathcal{V}_s$ with the largest values of ξ_i^*. Call this subset $\mathcal{Q}_s^- \subseteq \mathcal{V}_s$. Now, we replace $D_s(p, y) \geq \ell$ in problem (2.1) with the precise voxel constraints $y_i \geq \ell$ for all $i \in \mathcal{Q}_s^-$. On the second pass,

$$v_s(y, \ell) = \sum_{i \in \mathcal{V}_s} \mathbb{1}\{y_i \geq \ell\} \geq \sum_{i \in \mathcal{Q}_s^-} \mathbb{1}\{y_i \geq \ell\} = q_s \geq \phi_s(p),$$

so our upper dose-volume constraint is satisfied. An analogous argument with $q_s = \lceil \phi_s(100 - p) \rceil$ and $\xi_i^* = (u - y_i^*)$ produces the subset $\mathcal{Q}_s^+$ for an upper dose-volume constraint $D_s(p, y) \leq u$. Given a solution (x^*, y^*, α^*) to problem (2.2), we repeat this process with every such constraint to obtain the second-pass problem formulation

$$\begin{array}{lll}
\text{minimize} & f(y) & \\
\text{subject to} & y = Ax & \\
& x \succeq 0 & \\
& y_i \leq u_{s,k} \ \forall i \in \mathcal{Q}_{s,k}^+, & k = 1, \ldots, K_s^{(u)}, \quad s = 1, \ldots, S \\
& y_i \geq \ell_{s,k} \ \ \forall i \in \mathcal{Q}_{s,k}^-, & k = 1, \ldots, K_s^{(\ell)}, \quad s = 1, \ldots, S \\
& \ell_s^{\text{avg}} \leq D_s^{\text{avg}}(y) \leq u_s^{\text{avg}}, & s = 1, \ldots, S \\
& D_s^{\min}(y) \geq \ell_s^{\min}, & s = 1, \ldots, S \\
& D_s^{\max}(y) \leq u_s^{\max}, & s = 1, \ldots, S,
\end{array} \tag{2.3}$$

where the voxel subsets are indexed with $k = 1, \ldots, K_s^{(u)}$ for upper dose-volume constraints, and $k = 1 \ldots, K_s^{(\ell)}$ for lower dose-volume constraints. We can warm start our solver at (x^*, y^*) to speed up the second pass optimization.

Algorithm 2.6.1 *Two-pass algorithm.*

given a dose matrix $A \in \mathbf{R}^{m \times n}$, a prescribed dose vector $d \in \mathbf{R}^m$,
and a set of dose-volume constraints $\mathcal{C}$.

1. *First pass.* Obtain the solution (x^*, y^*, α^*) to problem (2.2).

for each $(\ell, p, s) \in \mathcal{C}$ **do**

2a. *Compute margins.* Calculate $\xi_i^* = y_i^* - \ell$ for all $i \in \mathcal{V}_s$.

2b. *Sort margins.* Sort $\{\xi_i^*\}_{i \in \mathcal{V}_s}$ in ascending order to form a set ξ_s.

2c. *Identify voxel subset.* Select the $\lceil \phi_s(p) \rceil$ largest values $\xi_i \in \xi_s$ and include their indices i in $\mathcal{Q}_{s,k}^-$.

end for

for each $(u, p, s) \in \mathcal{C}$ **do**

3a. *Compute margins.* Calculate $\xi_i^* = u - y_i^*$ for all $i \in \mathcal{V}_s$.

3b. *Sort margins.* Sort $\{\xi_i^*\}_{i \in \mathcal{V}_s}$ in ascending order to form a set ξ_s.

3c. *Identify voxel subset.* Select the $\lceil \phi_s(100 - p) \rceil$ largest values $\xi_i \in \xi_s$ and include their indices i in $\mathcal{Q}_{s,k}^+$.

end for

4. *Second pass.* Obtain the solution (x^{**}, y^{**}) to problem (2.3) using (x^*, y^*) as a warm start point.

2.6.2 Dose constraints with slack

If our dose constraints are too strict, problem (2.2) may not have a solution. This can arise even if the feasible set for our original problem (2.1) is non-empty, since our convex restriction enforces more stringent bounds on the dose distribution. To ensure the first pass of our algorithm always supplies a solution, we introduce a slack variable $\delta \in \mathbf{R}_+$ to the bounds of each dose constraint, mapping lower bounds $\ell \mapsto (\ell - \delta)$ and upper bounds $u \mapsto (u + \delta)$. This creates soft constraints that need not be met precisely by the solution. Our problem reformulated with restricted dose-volume constraints

and slack is

$$\begin{array}{ll} \text{minimize} & f(y) \\ \text{subject to} & y = Ax \\ & x \succeq 0, \quad \alpha \succeq 0, \quad \delta \succeq 0 \\ & \hat{D}_s^+\left(p_{s,k}, y, u_{s,k} + \delta_{s,k}^{(u)}, \alpha_{s,k}^{(u)}\right) \leq 0, \qquad k = 1, \ldots, K_s^{(u)}, \quad s = 1, \ldots, S \\ & \hat{D}_s^-\left(p_{s,k}, y, \ell_{s,k} - \delta_{s,k}^{(\ell)}, \alpha_{s,k}^{(\ell)}\right) \leq 0, \qquad k = 1, \ldots, K_s^{(\ell)}, \quad s = 1, \ldots, S \\ & \ell_s^{\text{avg}} - \delta_s^{\text{avg},(\ell)} \leq D_s^{\text{avg}}(y) \leq u_s^{\text{avg}} + \delta_s^{\text{avg},(u)}, \quad s = 1, \ldots, S \\ & D_s^{\min}(y) \geq \ell_s^{\min} - \delta_s^{\min}, \qquad s = 1, \ldots, S \\ & D_s^{\max}(y) \leq u_s^{\max} + \delta_s^{\max}, \qquad s = 1, \ldots, S, \end{array} \tag{2.4}$$

Note that $\delta := (\delta^{(u)}, \delta^{(\ell)})$ is a variable in the optimization, and the value of each $\delta_{s,k}$ indicates the amount (in units of delivered dose, *e.g.*, Gy) by which each bound is weakened in the solution.

We can incorporate soft constraints into the two-pass algorithm as well. On the first pass, we solve problem (2.4) to obtain the optimal variables (x^*, y^*, α^*) and the optimal slacks δ^*. Our margin for selecting $\mathcal{Q}_s$ is now computed with respect to the slack bound, *i.e.*, $\xi_i^* = (y_i^* - \ell + \delta^*)$ for lower-volume dose constraints, and $\xi_i^* = (u + \delta^* - y_i^*)$ for upper dose-volume constraints. Finally, we weaken the bounds in problem (2.3) by δ^*, giving us the reformulated second pass optimization with slack dose-volume constraints

$$\begin{array}{ll} \text{minimize} & f(y) \\ \text{subject to} & y = Ax \\ & x \succeq 0 \\ & y_i \leq u_{s,k} + \delta_{s,k}^{(u)*} \ \forall i \in \mathcal{Q}_{s,k}^+, \qquad k = 1, \ldots, K_s^{(u)}, \quad s = 1, \ldots, S \\ & y_i \geq \ell_{s,k} - \delta_{s,k}^{(\ell)*} \ \forall i \in \mathcal{Q}_{s,k}^-, \qquad k = 1, \ldots, K_s^{(\ell)}, \quad s = 1, \ldots, S \\ & \ell_s^{\text{avg}} - \delta_s^{\text{avg},(\ell)*} \leq D_s^{\text{avg}}(y) \leq u_s^{\text{avg}} + \delta_s^{\text{avg},(u)*}, \quad s = 1, \ldots, S \\ & D_s^{\min}(y) \geq \ell_s^{\min} - \delta_s^{\min *}, \qquad s = 1, \ldots, S \\ & D_s^{\max}(y) \leq u_s^{\max} + \delta_s^{\max *}, \qquad s = 1, \ldots, S. \end{array} \tag{2.5}$$

Algorithm 2.6.2 *Two-pass algorithm with slack.*

given a dose matrix $A \in \mathbf{R}^{m \times n}$, a prescribed dose vector $d \in \mathbf{R}^m$,
and a set of dose-volume constraints $\mathcal{C}$.

1. *First pass.* Obtain the solution $(x^*, y^*, \alpha^*, \delta^*)$ to problem (2.4).

for each $(\delta^*, \ell, p, s) \in \mathcal{C}$ **do**

2a. *Compute margins.* Calculate $\xi_i^* = y_i^* - \ell + \delta^*$ for all $i \in \mathcal{V}_s$.

2b. *Sort margins.* Sort $\{\xi_i^*\}_{i \in \mathcal{V}_s}$ in ascending order to form a set ξ_s.

2c. *Identify voxel subset.* Select the $\lceil \phi_s(p) \rceil$ largest values $\xi_i \in \xi_s$ and include their indices i in $\mathcal{Q}_{s,k}^-$.

end for

for each $(\delta^*, u, p, s) \in \mathcal{C}$ **do**

3a. *Compute margins.* Calculate $\xi_i^* = u + \delta^* - y_i^*$ for all $i \in \mathcal{V}_s$.

3b. *Sort margins.* Sort $\{\xi_i^*\}_{i \in \mathcal{V}_s}$ in ascending order to form a set ξ_s.

3c. *Identify voxel subset.* Select the $\lceil \phi_s(100 - p) \rceil$ largest values $\xi_i \in \xi_s$ and include their indices i in $\mathcal{Q}_{s,k}^+$.

end for

4. *Second pass.* Obtain the solution (x^{**}, y^{**}) to problem (2.5) using (x^*, y^*) as a warm start point.

2.7 Implementation

We implement our radiation treatment planning methodology with ConRad, a Python-embedded open-source software package based on the convex programming library, CVXPY (Diamond and Boyd, 2016), using the convex solvers SCS (O'Donoghue et al., 2016) and ECOS (Domahidi et al., 2013). ConRad provides a simple, intuitive interface for ingesting patient data, constructing plans based on a clinical prescription, and visualizing the dose-volume histograms of the result. It allows the user to add dose constraints using syntax familiar to clinicians. Since ConRad is an ordinary Python library, it can be easily integrated into existing data processing pipelines.

The following code imports a prescription, solves for the optimal treatment plan without dose constraints, and plots the DVH curves for all the patient structures. The $m \times n$ dose-influence matrix `A` can be encoded as a NumPy `ndarray` or any of several sparse representations in Python. The m-length vector `voxel_labels` enumerates the index of the assigned structure for each voxel in the patient volume.

```
import conrad

# Construct the case with no dose constraints.
> case = conrad.Case()
> case.prescription = "/Documents/prescriptions/rx_patient_01.yaml"
> case.physics.dose_matrix = A
> case.physics.voxel_labels = voxel_labels
> graphics = conrad.CasePlotter(case)

# Solve with a single pass and no slack.
> status, run = case.plan(solver = "ECOS", use_slack = False,
                          use_2pass = False)
> print("Problem feasible?:\n{}".format(status))
> print("Dose summary:\n{}".format(case.dose_summary_string))

# Display color-coded plot of all DVH curves.
> graphics.plot(run, show = True)
```

A `Case` object comprises `Anatomy`, `Physics`, `Prescription` and `PlanningProblem` objects. Prior to planning, the case's `Anatomy` and `Physics` objects must be built. The user can either build the case's `Anatomy` by adding structures programmatically (with data on each structure's name, index, identity as target/non-target, and desired dose) or by ingesting a prescription, which can be supplied as a Python dictionary or as a YAML or JSON file formatted for ConRad's parser. The minimum information required for the case's `Physics` object are the $m \times n$ dose matrix and a m-length vector of voxel labels. The case's `Prescription` object can be used to populate the

`Anatomy` object or to keep track of clinical guidelines and objectives. It can also be left empty.

The case's `PlanningProblem` object builds and solves optimization problems based on the structures in the case anatomy and any constraints assigned to those structures. The `PlanningProblem` is not exposed to the user. Instead, users form a treatment plan by calling the case's `plan()` method, which returns a `bool` status indicating whether the specified problem was feasible, along with a `RunRecord` object that carries solver performance data, optimal variables, and DVH curves.

Before planning a case, the user can add, remove, or modify dose constraints to any structure. Thus, even when a case has an assigned prescription, the dose constraints attached to each structure in the case anatomy may differ from the constraints specified in the prescription. For example, the prescribed constraints may correspond to clinical guidelines, while the constraints used during planning may be chosen arbitrarily by the user to obtain plans with desirable dose properties.

After planning a case, users can plot the DVH curves, retrieve and print summaries of dose statistics for each structure, and when applicable, display a report of whether the current plan satisfies each constraint listed in the prescription. A case can be re-planned with different objective weights or dose constraints on any structure. The ConRad library provides a `PlanningHistory` object to retain and manage results from prior runs.

The following code adds a dose-volume constraint to the PTV from our previous case, allowing at most 20% of the PTV's voxels to receive more than 70 Gy. Algorithm 2.6.1 is then applied to obtain an optimal beam output.

```
# Constrain at most 20% of PTV voxels to receive dose above 70 Gy.
> case.anatomy["PTV"].constraints += D(20) <= 70 * Gy

# Solve with two-pass algorithm and no slack.
> _, run = case.plan(solver = "ECOS", use_slack = False,
                     use_2pass = True)
> print("x PASS 1: {}".format(run.x_pass1))
> print("x PASS 2: {}".format(run.x_pass2))
```

```
# Plot DVH curves from first (dashed) and second pass (solid).
> graphics.plot(run, show = False, ls = "--")
> graphics.plot(run, second_pass = True, show = True, clear = False,
                legend = True)
```

2.8 Examples

2.8.1 Basic functionality

We present results illustrating the methods described in this chapter: approximating dose-volume constraints via convex restrictions, two-pass refinement of plans with dose-volume constraints, and handling incompatible constraints with slack variables.

Problem instances. We demonstrate the basic functionality on a head-and-neck case expressed as a VMAT aperture re-weighting problem. The case contains 360 apertures in four arcs, 270,000 voxels distributed across 17 planning structures, including the PTV treated to 66 Gy, two auxiliary targets treated to 60 Gy, several OARs, and generic body voxels.

To test the handling of dose-volume constraints, we plan the case with no dose constraints and then re-plan with a single dose-volume constraint applied to the PTV, namely $D(20) \leq 70$ Gy. We run the two-pass algorithm and compare the plans obtained by applying restricted and exact versions of the aforementioned dose-volume constraint. Finally, we test the slack method by planning the case with two incompatible dose-volume constraints: $D(98) \geq 66$ Gy on the PTV and $D(20) \leq 20$ Gy on the spinal cord. We compare the results from enforcing the PTV constraint alone, both constraints without slack, and both constraints with slack allowed.

Computational details. The size of the dose matrix passed from ConRad to the convex solvers in the backend varied depending on the dose constraints. When no minimum, maximum, or dose-volume constraints were applied to a non-target structure, the submatrix for that structure was replaced with a mean dose representation,

thereby eliminating $|\mathcal{V}_s| - 1$ rows from the problem matrix. In particular, the matrix representing the full dose on the targets and mean dose for non-targets had dimensions 11,141 $\times$ 360, and the matrix including the full dose on the spinal cord was 15,000 $\times$ 360. The ConRad problem request was formulated as a convex program in CVXPY and passed to a GPU-based implementation of the convex solver SCS. Calculations were performed on a cluster with 32-core, 2.20 GHz Intel Xeon E5-4620 CPU and a nVidia TitanX graphics card.

Clinical results: dose-volume constraints. Figure 2.4a depicts the DVH curves of the plan produced without any dose constraints. The PTV curve is shown in red with a dotted vertical line marking its prescribed dose of 66 Gy. The rest are DVHs for the OARs and generic body voxels. This solution to the unconstrained problem already gives a fairly good treatment plan. The DVH of the right cochlea and left parotid are pushed far left, so only 15–20% of their volume exceed 10 Gy, and almost no voxels are dosed above 30 Gy. The spinal cord receives somewhat more radiation, while the worst case is the brain with a slow, nearly linear drop-off to about 75 Gy.

The PTV curve begins to fall at 66 Gy, but does not reach zero until nearly 95 Gy. To reduce this overdosing, we add a dose-volume constraint that limits no more than 20% of the PTV to receive over 70 Gy, as indicated by the red, left-pointing arrow in Figure 2.4b, and re-plan the case. The resulting DVH curves are depicted as solid lines in Figure 2.4c, while the dashed curves represent the original plan. Under the new plan, the PTV curve has been pushed left at the arrow, and its drop-off around 66 Gy is steeper, meaning all voxel doses are closer to the prescription. Moreover, our OARs are minimally affected. We have reduced overdosing to the PTV without significantly increasing radiation to other organs.

Clinical results: two-pass algorithm. A close inspection of Figure 2.4c reveals a gap between the PTV curve and the DVH constraint arrow. This is due to the conservative nature of the convex restriction, which overestimates the number of voxel violations. We can eliminate this gap and improve our overall objective with the two-pass algorithm. Figure 2.5a shows the plan from the first pass. There is a

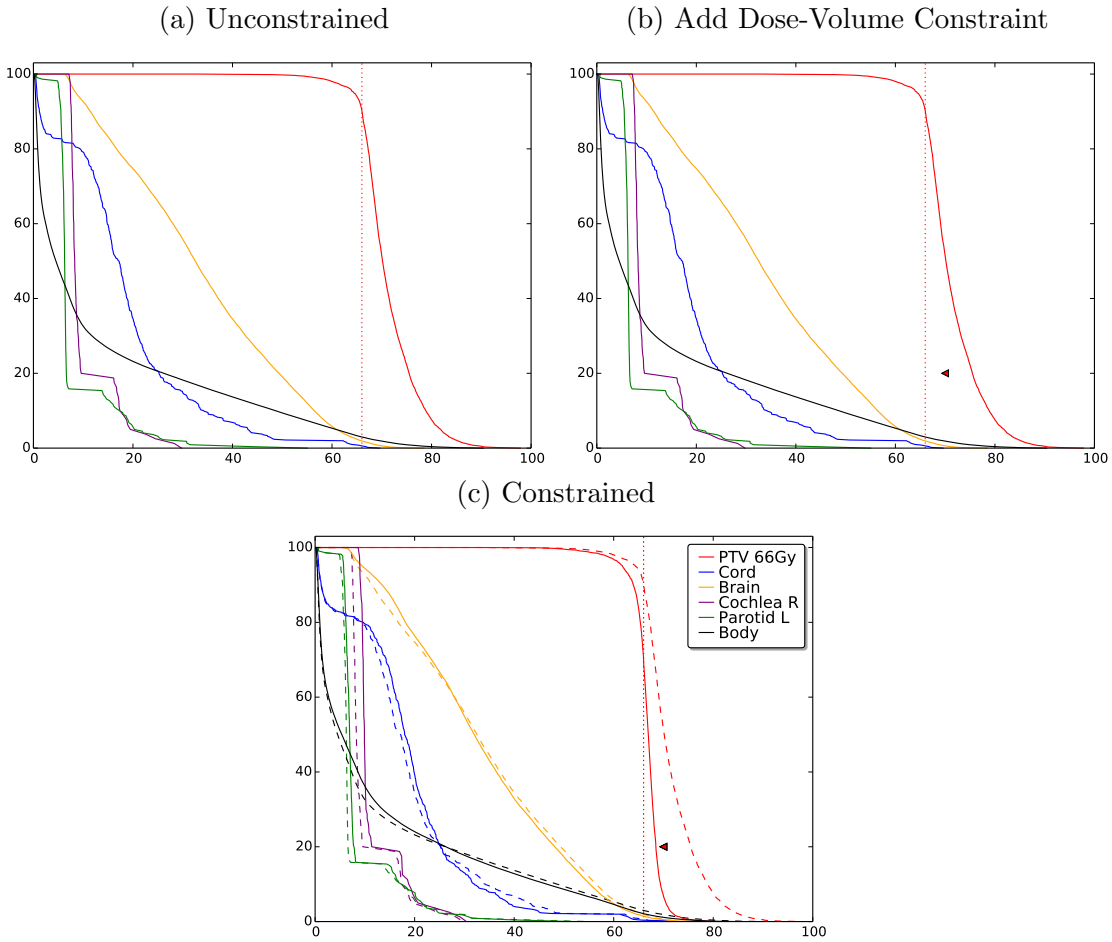


Figure 2.4: DVH curves for PTV and several OARs from the 4-arc VMAT head-and-neck case. (a) Plan without any dose constraints. (b) Add a dose-volume constraint $D(20) \leq 70$ Gy. (c) Re-plan with the new constraint. The unconstrained plan is shown in dashed lines.

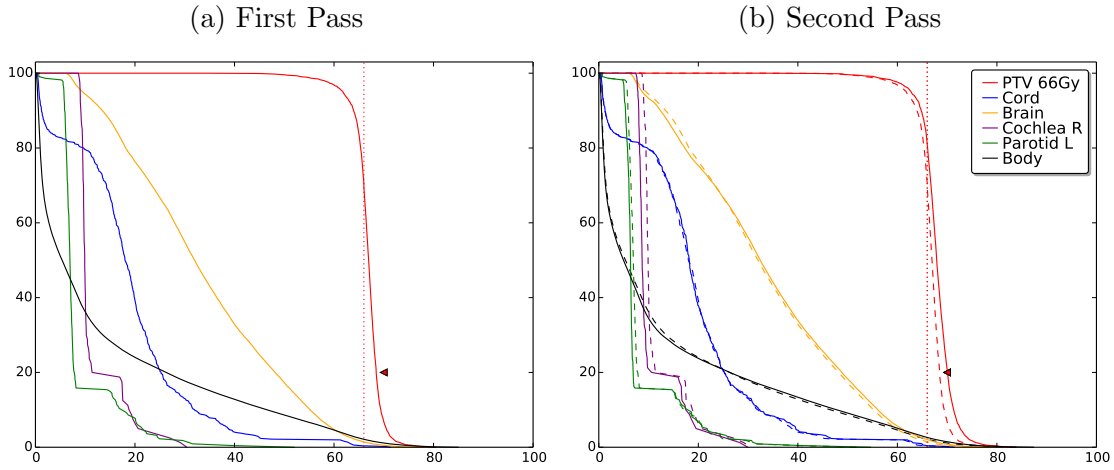


Figure 2.5: DVH curves for a two-pass algorithm with a single dose-volume constraint on the PTV. (a) On the first pass, the constraint is met with a margin of about 0.5 Gy. (b) On the second pass, the constraint is met tightly with small gains elsewhere.

margin of about 0.5 Gy between the PTV curve and arrow, meaning at most 20% of the PTV receives over 69.5 Gy, a more restrictive solution than required by our constraint $D(20) \leq 70$ Gy.

This margin disappears in Figure 2.5b. Here, the dashed lines depict the first pass solution, and the solid lines come from the second pass. The second pass PTV curve falls precisely on the center of the left-facing arrow, meaning the dose-volume constraint is tight. In addition, the DVH curves for the right cochlea and left parotid have shifted leftward, indicating they now receive less radiation. By replacing our convex restriction with exact voxel constraints, we are able to make gains in our OAR clinical objectives while still fulfilling the DVH constraint.

Clinical results: constraints with slack. So far, we have specified only one dose constraint. A problem may become infeasible when multiple constraints are enforced, either because the user-supplied bounds are too extreme or the convex restriction too severe in its overestimation of the voxel count. In such cases, we can still produce a plan that approximately conforms to the desired specifications by enabling slack constraints.

Figure 2.6a depicts a plan created without slack. The single PTV constraint, $D(98) \geq 66$ Gy, is met with margin. However, when we add the OAR constraint $D(20) \leq 20$ Gy, as symbolized by the blue arrow in Figure 2.6b, and re-plan the case, the optimizer tells us that the problem is infeasible. It is impossible to meet both (convex restricted) dose-volume constraints exactly. We thus re-plan allowing for slack bounds on these constraints. The resulting DVH curves are plotted in Figure 2.6c with dotted lines representing the original plan and solid lines for the new plan. The PTV constraint has been relaxed by about 3 Gy, as symbolized by the red arrow shifting left behind the solid red curve to $(98, 63)$. This small concession allows us to satisfy the OAR constraint by a wide margin.

2.8.2 Problem scaling

Problem instances. We assess the performance of our algorithm on a larger prostate FMO problem. This case contains 74,453 voxels $\times$ 34,848 beamlets, encompassing a single PTV treated to 75.6 Gy, five OARs with various dose constraints, and generic body voxels. Approximately 226 million (roughly 10.6%) of the entries in the dose matrix are non-zero. In our experiments, we used only a subset of 10,000 beamlets from this matrix.

We plan the case with the prescription detailed in Table 2.1, which is adapted from the QUANTEC guidelines in Marks et al. (2010). The computational details are the same as in the head-and-neck case. As before, we analyze the results from a single pass and two-pass algorithm with and without slack allowed. We then re-plan the case with only the PTV dose constraint and compare its runtime and OAR overdose to the plans produced from the full prescription.

Timing results. Our algorithm produces a plan that satisfies all dose constraints using a single pass with slack enabled. The optimization finishes in 426.9 seconds, about 2.5x the runtime of the head-and-neck case. A second pass takes approximately the same amount of time and does not result in significantly larger constraint margins. The presence or absence of slack also has little impact on the runtime, which varies by at most 7 seconds.

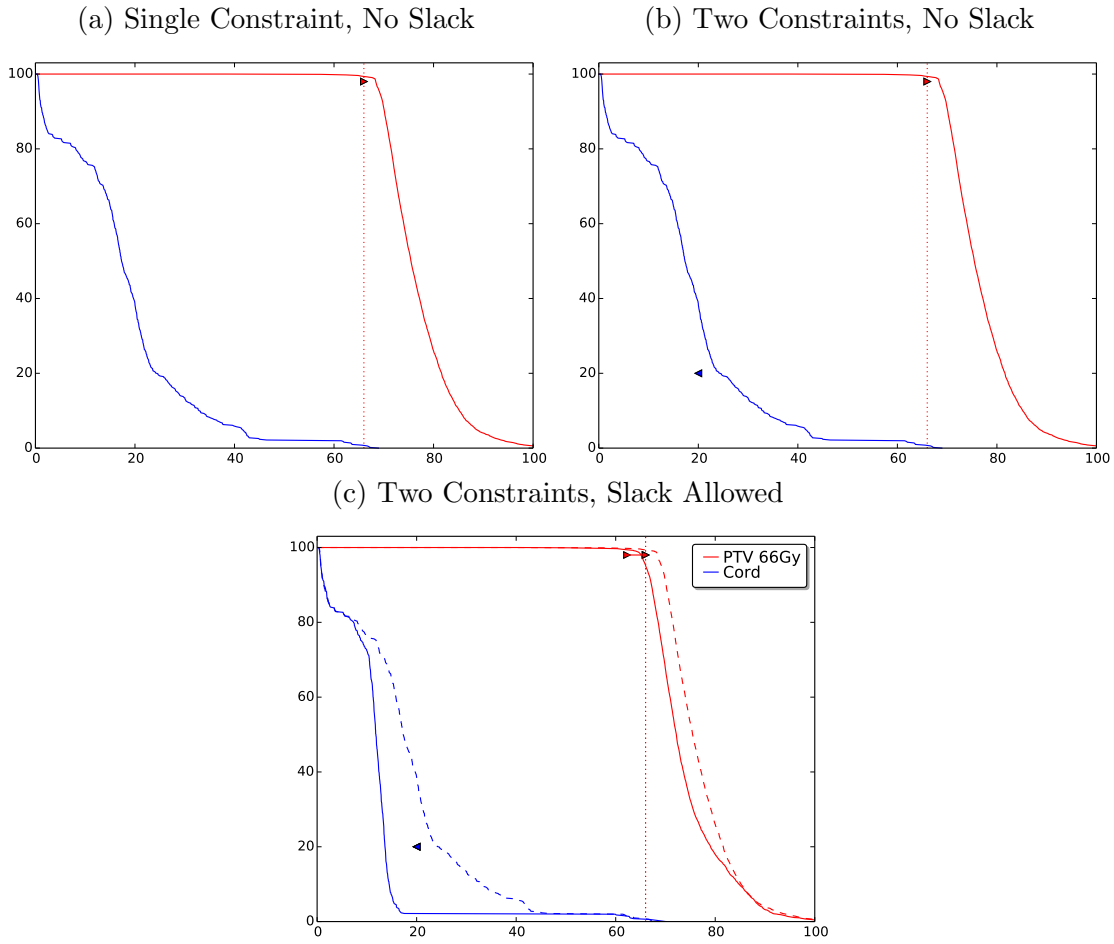


Figure 2.6: DVH curves for the PTV and spinal cord. (a) Plan without slack, constraining $D(98) \geq 66$ Gy in the PTV. (b) A constraint $D(20) \leq 20$ Gy is added to the spinal cord, rendering the problem infeasible. (c) Re-plan with slack allowed. The spinal cord constraint is met, but the PTV constraint relaxes by about 3 Gy.

Table 2.1: Prostate FMO Prescription

Structure	Target?	Dose (Gy)	Constraints (Gy)
Prostate	Yes	75.6	$D^{\text{avg}} \geq 75.6$
Urethra	No	0	$D^{\text{avg}} < 52.5$
Bladder	No	0	$D(85) < 80$ $D(75) < 75$ $D(65) < 70$ $D(50) < 65$
Rectum	No	0	$D(90) < 75$ $D(85) < 70$ $D(50) < 65$
L. Femoral Head	No	0	$D(95) < 50$
R. Femoral Head	No	0	$D(95) < 50$
Body	No	0	$D^{\text{avg}} < 52.5$

If we drop all except the mean dose constraints, the problem collapses into a linear program, greatly decreasing the runtime. The size of this reduction depends on the characteristics of the affected structures and the dose influence matrix. For example, in the head-and-neck case, the runtime falls by 87% to a mere 24 seconds. Conversely, when adding dose-volume constraints, the initial constraint on a structure will have a greater impact on runtime than subsequent additions.

2.9 Conclusion

We have developed a convex formulation for the FMO problem that incorporates dose-volume constraints. Our model replaces each exact dose-volume constraint with a convex restriction, which overestimates the number of voxels that violate the clinician's desired threshold. This allows us to solve the problem quickly and efficiently using standard convex optimization algorithms. We also introduce two refinements: a two-pass algorithm and a model with slack. In the former, we improve our initial solution by re-optimizing with the restrictions replaced by bounds on a subset of voxels, enabling us to achieve a better objective that still satisfies the dose-volume constraints. The latter allows for soft bounds and is useful if the restricted constraints

render the problem infeasible. We demonstrate the efficacy of our method on a VMAT head-and-neck case and a prostate case. Our algorithm consistently produces good treatment plans that fulfill all dose constraints when feasible. In problems with infeasible constraints, we are able to generate plans that minimize the dose violation while taking into account clinical goals, allowing clinicians to easily visualize trade-offs and select the plan that is best for the patient.

A variety of extensions to our two-pass algorithm are possible. For instance, we could rewrite the original problem as a mixed-integer linear program and use the solution of the convex restriction to warm start a branch-and-bound solver. More broadly, we could apply this starting point to accelerate any number of iterative approaches in the literature. Dose-volume constraints are often assigned different priorities in practice, and our algorithm may be easily adapted to accommodate such user-defined preferences, either through new penalties, changes in the slack, or additional passes that impose the constraints in a lexicographic order. These hybrid methods, which combine convex approximations with nonconvex solution methods, offer an important avenue for future research.

Chapter 3

Adaptive Radiation Treatment Planning

3.1 Introduction

In this chapter, we describe a method for radiation treatment planning that adapts dynamically to changes in the patient's health. Typically, radiation therapy takes place over multiple treatment sessions. A clinician will divide up the total prescribed dose into smaller dose fractions, which are then delivered over the course of several weeks or months. This permits normal tissue time to recover and repair sublethal cell damage, but also gives tumors an opportunity to proliferate, especially when the treatment course is long. A study of 4338 prostate cancer patients showed that biochemical failure increases by 6% for every 1 week increase in treatment time, with a dose equivalent of proliferation of 0.24 Gy/day (Thames et al., 2010). Thus, an important question in radiation treatment planning is how to choose the sequence of deliverable doses such that they balance these temporal effects on a patient's health.

We formulate the adaptive treatment planning problem as an optimal control problem with nonlinear patient health dynamics, which we derive from the linear-quadratic (LQ) model of post-irradiation cell survival (Fowler, 1989). As this formulation is nonconvex, we propose a method for obtaining an approximate solution by solving a sequence of convex optimization problems. Our method is fast, efficient, and

robust to model error, adapting readily to changes in the patient's health between sessions. Moreover, we show that it can be combined with the operator splitting method ADMM to produce an algorithm that is highly scalable and can handle large clinical cases, which involve tens of thousands of radiation beams. We introduce an open-source Python software package, AdaRad, that implements this method and demonstrate its performance on several examples.

Early clinical practitioners split the prescribed dose equally over a fixed number of sessions. While convenient, this method does not account for errors or uncertainty in the treatment process. For example, due to patient movement during radiation delivery, the expected dose may differ from the actual dose to an anatomical structure. If the actual dose is observable, a common way to compensate for this is to divide the residual dose (*i.e.*, the difference between the prescribed and cumulative actual dose) across the remaining sessions. This then becomes the new per-session dose goal. de la Zerda et al. (2007) solve for the beam intensities by minimizing the sum-of-squared difference between this goal dose and the expected dose. They compare the results when errors are perfectly known, so the expected dose is equal to the actual dose, with the results when errors are assumed to be zero. Ferris and Voelker (2004) take a similar approach, except the errors are modeled explicitly as a random shift in the surrounding voxels. Instead of the dose to each voxel, Sir et al. (2012) work with the equivalent uniform dose (EUD), a value that captures the biological effect of a dose distribution over a region. Their objective is to minimize the sum of the EUD over all treatment criteria subject to bounds on the EUD of the tumor and normal tissues. To solve this problem, they employ methods from approximate dynamic programming coupled with a discrete probabilistic model of the dose error.

The papers we have discussed so far only focus on the dose to the patient. Kim et al. (2009) introduce a Markov decision process model that includes both the dose (action) and the patient's health state. Each choice of dose induces a transition to a particular health state with some probability. Making this idea concrete, Mizuta et al. (2012) define the health of a tumor (resp. OAR) to be the radiation (resp. damage) effect of the delivered dose, as calculated from the linear-quadratic (LQ) model of cell survival (Fowler, 1989). They analyze a simple example with one tumor and one

OAR and find that the optimal fractionation scheme is either a single session delivery of the full dose or equal dose fractions, depending on the relationship between the LQ parameters. Bortfeld et al. (2015) extend this analysis to incorporate accelerated tumor repopulation and show that the dose per session increases over the treatment course. Using simulated annealing, Yang and Xing (2005) solve a similar treatment planning problem based on the LQR model, which captures all 4 Rs (repair of sublethal damage, repopulation, redistribution, and reoxygenation) of cellular radiation response (Brenner et al., 1995).

These analyses provide insight into the tradeoffs between hypo- and hyper-fractionation in a simple setting. However, most clinical cases are more complex, involving multiple tumors, OARs, and nonlinear constraints. For instance, dose-volume (*i.e.*, percentile) constraints are widely used to limit the radiation exposure of a percentage of an anatomical structure, such as the spine. These constraints are nonconvex, but can be approximated by a convex restriction (Halabi et al., 2006b; Zarepisheh et al., 2013; Fu et al., 2019). In Saberian et al. (2016), the authors consider a dynamic setting with multiple OARs and dose-volume constraints. Starting from a given set of beam intensities, they solve for the optimal number of sessions and OAR sparing factors. They also derive sufficient conditions under which the optimal treatment consists of equal dose fractions. In a follow-up paper (Saberian et al., 2017), the authors integrate the spatial and temporal aspects of the problem, treating both beam intensities and number of sessions as variables. Restricting their attention to equal fractions, they propose a two-stage solution algorithm: in the first stage, they solve for the optimal beams given each potential fixed number of sessions, and in the second stage, they select the number of sessions based on the optimal objectives from the first stage. They show that their method achieves better tumor ablation than conventional IMRT or the spatiotemporally separated method.

Perhaps the paper most similar to our work in this chapter is Kim et al. (2012). In it, the authors propose a stochastic control formulation of the adaptive treatment planning problem with multiple tumors and OARs. They estimate the radiation response of the tumors with a log-linear cell kill model and the response of the OARs with the standard LQ model. Their goal is to minimize the expected number of

tumor cells at the end of treatment subject to bounds on the radiobiological impact on the OARs. Uncertainty arises in the cell model parameters, which may fluctuate randomly between sessions, representing unpredictable changes in the patient's health status. The authors fix the number of sessions and focus on optimizing with respect to the beam intensities. They show that their problem is convex, so can be solved using a combination of standard stochastic control methods and off-the-shelf convex solvers, and provide several examples demonstrating the effectiveness of their approach.

3.2 Problem formulation

In radiation treatment, beams of ionizing radiation are delivered to a patient from an external source. The goal is to damage or kill diseased tissue, while minimizing harm to surrounding healthy organs. A course of treatment is generally divided into T sessions. At the start of session t, the clinician chooses the intensity levels of the n beams, denoted by $b_t \in \mathbf{R}_+^n$. Typically, $T \approx 20$ and n is on the order of 10^3 to 10^4. We are interested in determining the best sequence of beam intensities $b = (b_1, \ldots, b_T)$, otherwise known as a *treatment plan*, subject to upper bounds $B_t \in \bar{\mathbf{R}}_+^n$ on b_t for $t = 1, \ldots, T$.

Anatomy and doses. The beams irradiate an area containing K anatomical structures, labeled $i \in \{1, \ldots, K\}$, where usually $K < 10$. A subset $\mathcal{T} \subset \{1, \ldots, K\}$ are targets/tumors and the rest are OARs. The dose delivered to each structure is linear in the beam intensities. We write the dose vector $d_t = A_t b_t$ with $A_t \in \mathbf{R}_+^{K \times n}$ a known matrix that characterizes the physical effects and define $d = (d_1, \ldots, d_T)$. Notice that since b_t and A_t are nonnegative, $d_t \geq 0$.

In every session, we impose a penalty on d_t via a *dose penalty function* $\phi_t : \mathbf{R}^K \to \mathbf{R} \cup \{\infty\}$. A common choice is

$$\phi_t(d_t) = \theta_t^T d_t + \xi_t^T d_t^2,$$

where $\theta_t \in \mathbf{R}^K$ and $\xi_t \in \mathbf{R}_+^K$ are constants. Here d_t^2 denotes the elementwise square

of the dose vector. The total dose penalty over all sessions is

$$\phi(d) = \sum_{t=1}^{T} \phi_t(d_t).$$

Additionally, we enforce upper bound constraints $d_t \leq D_t$, where $D_t \in \bar{\mathbf{R}}_+^K$ is the maximum dose in session t.

Health dynamics. To assess treatment progress, we examine the health status of each anatomical structure and encode these statuses in a vector $h_t \in \mathbf{R}^K$. For now, the details of this encoding do not matter. Typically, h_{ti} represents an estimate of the total surviving cells in structure i. Hence if $i \in \mathcal{T}$, a smaller h_{ti} is desirable (since the tumor is shrinking), while if $i \notin \mathcal{T}$, a larger h_{ti} is desirable.

From an initial h_0, the health status evolves in response to the radiation dose and various other biophysical factors that depend on the patient's anatomy, generating a health trajectory $h = (h_1, \ldots, h_T)$. Here we represent its dynamics as

$$h_t = f_t(h_{t-1}, d_t), \quad t = 1, \ldots, T, \tag{3.1}$$

where $f_t : \mathbf{R}^K \times \mathbf{R}^K \to \mathbf{R}^K$ is a known mapping function. In this chapter, we focus on the linear-quadratic (LQ) model in which

$$f_{ti}(h_{t-1}, d_t) = h_{(t-1)i} - \alpha_{ti} d_{ti} - \beta_{ti} d_{ti}^2 + \gamma_{ti}, \quad i = 1, \ldots, K, \quad t = 1, \ldots, T \tag{3.2}$$

with constants $\alpha_t \in \mathbf{R}^K, \beta_t \in \mathbf{R}_+^K$, and $\gamma_t \in \mathbf{R}^K$. This model is commonly used to approximate cellular response to radiation (Fowler, 1989; Thames and Hendry, 1987; Brenner, 2008). Specifically, in the LQ + time framework (Travis and Tucker, 1987), h_{ti} is the log of the fraction of surviving cells in structure i after a dose d_{ti}, while α_{ti}/β_{ti} and γ_{ti} are constants related to the structure's survival curve and repair/repopulation rate, respectively. Notice that equation (3.2) implies that the health status of each structure evolves independently of the others.

Health penalty and constraints. In order to control the patient's health, we introduce a *health penalty function* $\psi_t : \mathbf{R}^K \to \mathbf{R} \cup \{\infty\}$ that imposes a penalty on h_t. Moreover, we assume that

$$\psi_t(h_t) = \psi_t(h_{t1}, \ldots, h_{tK}) \text{ is monotonically } \begin{cases} \text{increasing in } h_{ti} & i \in \mathcal{T} \\ \text{decreasing in } h_{ti} & i \notin \mathcal{T} \end{cases} \tag{3.3}$$

for $t = 1, \ldots, T$. This means that for a target, the health penalty increases as the health status increases, while for an organ-at-risk, the health penalty decreases as the health status increases. The assumption is reasonable if, for instance, h_t is a measure of cell survival in session t, so a lower (higher) status is desirable for a target (organ-at-risk). An example of a penalty function that satisfies (3.3) is

$$\psi_t(h_t) = \overline{w}^T (h_t - h_t^{\text{goal}})_+ + \underline{w}^T (h_t - h_t^{\text{goal}})_-,$$

where $h_t^{\text{goal}} \in \mathbf{R}^K$ is the desired health status and $\underline{w} \in \mathbf{R}_+^K$ and $\bar{w} \in \mathbf{R}_+^K$ are parameters with $\underline{w}_i = 0$ for $i \in \mathcal{T}$ and $\overline{w}_i = 0$ for $i \notin \mathcal{T}$. Here $(x)_+ = \max(x, 0)$ applied elementwise to x. The total health penalty is

$$\psi(h) = \sum_{t=1}^{T} \psi_t(h_t).$$

In addition, we enforce bounds $H_t \in \bar{\mathbf{R}}^K$ on the health status such that $h_{ti} \leq H_{ti}$ for $i \in \mathcal{T}$ and $h_{ti} \geq H_{ti}$ for $i \notin \mathcal{T}$.

Optimal control problem. Given an initial health status h_0, we wish to select a treatment plan that minimizes the total penalty across all sessions. Thus, our problem is

$$\begin{array}{lll} \text{minimize} & \sum_{t=1}^T \phi_t(d_t) + \sum_{t=1}^T \psi_t(h_t) & \\ \text{subject to} & h_t = f_t(h_{t-1}, d_t), & t = 1, \ldots, T, \\ & h_{ti} \leq H_{ti},\ i \in \mathcal{T}, \quad h_{ti} \geq H_{ti},\ i \notin \mathcal{T}, & t = 1, \ldots, T, \\ & d_t = A_t b_t, \quad 0 \leq d_t \leq D_t, \quad 0 \leq b_t \leq B_t, & t = 1, \ldots, T \end{array} \tag{3.4}$$

with variables $(b_1, \ldots, b_T), (d_1, \ldots, d_T)$, and $(h_1, \ldots, h_T)$. This is a discrete-time optimal control problem. If ϕ_t and ψ_t are convex and f_t is affine, *e.g.*, f_t is given by (3.2) with quadratic dose effect $\beta_t = 0$, it is also convex and can be solved directly using standard convex solvers.

3.3 Lossless relaxation

For the remainder of this chapter, we restrict our attention to a convex objective function and linear-quadratic health dynamics (3.2). In this case, condition (3.3) allows us to relax the health dynamics constraint so problem (3.4) can be written equivalently as

$$\begin{array}{lll} \text{minimize} & \sum_{t=1}^T \phi_t(d_t) + \sum_{t=1}^T \psi_t(h_t) & \\ \text{subject to} & h_{ti} \geq f_{ti}(h_{t-1}, d_t),\ i \in \mathcal{T}, & t = 1, \ldots, T, \\ & h_{ti} \leq f_{ti}(h_{t-1}, d_t),\ i \notin \mathcal{T}, & t = 1, \ldots, T, \\ & h_{ti} \leq H_{ti},\ i \in \mathcal{T}, \quad h_{ti} \geq H_{ti},\ i \notin \mathcal{T}, & t = 1, \ldots, T, \\ & d_t = A_t b_t, \quad 0 \leq d_t \leq D_t, \quad 0 \leq b_t \leq B_t, & t = 1, \ldots, T. \end{array} \tag{3.5}$$

The equality constraint $h_t = f_t(h_{t-1}, d_t)$ has been replaced with two inequality constraints: a lower bound for targets and an upper bound for OARs. Notice that the first inequality is the only nonconvex constraint in (3.5). Our relaxed problem has the same solution set as (3.4) because these two inequalities are tight at the optimum.

Proposition 1. *Let $(b^\star, d^\star, h^\star)$ be a solution to problem (3.5). If conditions (3.2) and (3.3) hold,*

$$h_t^\star = f_t(h_{t-1}^\star, d_t^\star), \quad t = 1, \ldots, T.$$

Proof. Suppose there exist some $t \in \{1, \ldots, T\}$ and $i \in \mathcal{T}$ such that $h_{ti}^\star > f_{ti}(h_{t-1}^\star, d_t^\star)$. Then, we can choose an $\epsilon > 0$ such that $h_{ti}^\star > h_{ti}^\star - \epsilon > f_{ti}(h_{t-1}^\star, d_t^\star)$. Since $f_{si}(h_{s-1}, d_s)$ is nondecreasing in $h_{(s-1)i}$ for all $s \in \{1, \ldots, T\}$, the point $(b^\star, d^\star, \hat{h})$ with

$$\hat{h}_{sj} = \begin{cases} h_{sj}^\star - \epsilon & s = t,\ j = i \\ h_{sj}^\star & \text{otherwise} \end{cases}$$

is feasible for problem (3.5) because

$$\hat{h}_{ti} > f_{ti}(h^\star_{t-1}, d^\star_t) \geq f_{ti}(\hat{h}_{t-1}, d^\star_t), \quad h^\star_{(t+1)i} \geq f_{(t+1)i}(h^\star_t, d^\star_t) \geq f_{(t+1)i}(\hat{h}_t, d^\star_t),$$

and $\hat{h}_{ti} < h^\star_{ti} \leq H_{ti}$. Moreover, by condition (3.3), $\psi_t(\hat{h}_t) < \psi_t(h^\star_t)$ so $(b^\star, d^\star, \hat{h})$ achieves a lower objective value than $(b^\star, d^\star, h^\star)$, contradicting our original assumption. An analogous argument holds for $t \in \{1, \ldots, T\}$ and $i \notin \mathcal{T}$ such that $h^\star_{ti} < f_{ti}(h^\star_{t-1}, d^\star_t)$ with $\hat{h}_{ti} = h^\star_{ti} + \epsilon$. □

3.4 Sequential convex optimization

3.4.1 Algorithm description

Problem (3.5) is in general nonconvex because the target's health dynamics constraint

$$h_{ti} \geq f_{ti}(h_{t-1}, d_t), \ i \in \mathcal{T}, \quad t = 1, \ldots, T \tag{3.6}$$

is nonconvex when any $\beta_t \neq 0$. However, we can derive an estimate of its optimum by solving a sequence of convex approximations. Each approximation is formed by linearizing the health dynamics function (3.2) around a fixed dose point and replacing the right-hand side of (3.6) with this linearization minus a slack variable. The slack allows for a degree of error in the approximation and is penalized in the objective.

More precisely, let $d^s_t \in \mathbf{R}^K$ for $t = 1, \ldots, T$. Define the linearized dynamics function

$$\hat{f}_{ti}(h_{t-1}, d_t; d^s_t) = h_{(t-1)i} - \alpha_{ti} d_{ti} - \beta_{ti} d^s_{ti}(2d_{ti} - d^s_{ti}) + \gamma_{ti}, \quad i = 1, \ldots, K. \tag{3.7}$$

This function is an upper bound on the LQ function (3.2) because $\beta_t \geq 0$. We replace the nonconvex constraint (3.6) in problem (3.5) with the affine constraint

$$h_{ti} = \hat{f}_{ti}(h_{t-1}, d_t; d^s_t) - \delta_{ti}, \ i \in \mathcal{T}, \quad t = 1, \ldots, T, \tag{3.8}$$

where $\delta_t \in \mathbf{R}^K_+$ is a slack variable. (The inequality can been tightened into an equality

due to Proposition 1). Convex approximation s is then

$$\begin{array}{lll} \text{minimize} & \sum_{t=1}^T \phi_t(d_t) + \sum_{t=1}^T \psi_t(h_t) + \lambda \sum_{t=1}^T \mathbf{1}^T \delta_t & \\ \text{subject to} & h_{ti} = \hat{f}_{ti}(h_{t-1}, d_t; d_t^s) - \delta_{ti},\ i \in \mathcal{T}, \quad \delta_t \geq 0 & t = 1, \ldots, T, \\ & h_{ti} \leq f_{ti}(h_{t-1}, d_t),\ i \notin \mathcal{T}, & t = 1, \ldots, T, \\ & h_{ti} \leq H_{ti},\ i \in \mathcal{T}, \quad h_{ti} \geq H_{ti},\ i \notin \mathcal{T}, & t = 1, \ldots, T, \\ & d_t = A_t b_t, \quad 0 \leq d_t \leq D_t, \quad 0 \leq b_t \leq B_t, & t = 1, \ldots, T \end{array} \tag{3.9}$$

with variables $(b_1, \ldots, b_T)$, $(d_1, \ldots, d_T)$, $(h_1, \ldots, h_T)$, and $(\delta_1, \ldots, \delta_T)$ and slack penalty parameter $\lambda > 0$. This problem is convex and can be solved using standard convex solvers. Given a solution to (3.9), we set the next linearization point $d^{s+1} = (d_1^{s+1}, \ldots, d_T^{s+1})$ equal to the optimal dose.

Algorithm 3.4.1 *Sequential convex optimization.*

input: initial point d^0, parameter $\lambda > 0$.

for $s = 0, 1, \ldots$ **do**

1. *Linearize.* For $t = 1, \ldots, T$, form the linearization (3.7) around d_t^s.
2. *Solve.* Set d^{s+1} equal to an optimal dose of problem (3.9).

until stopping criterion (3.10) is satisfied.

Algorithm 3.4.1 is a special case of the convex-concave procedure (CCP) (Yuille and Rangarajan, 2003; Lipp and Boyd, 2016; Shen et al., 2016), which is itself a form of majorization-minimization (Hunter and Lange, 2004; Sun et al., 2017). CCP is a heuristic for finding a local optimum of a nonconvex optimization problem. It is guaranteed to converge; indeed, when certain differentiability conditions are met, it converges to a stationary point (Sriperumbudur and Lanckriet, 2009). As a descent algorithm, CCP is usually terminated when the change in the objective falls below some user-specified threshold $\epsilon > 0$, *i.e.*,

$$p_{\text{opt}}^s - p_{\text{opt}}^{s+1} < \epsilon, \tag{3.10}$$

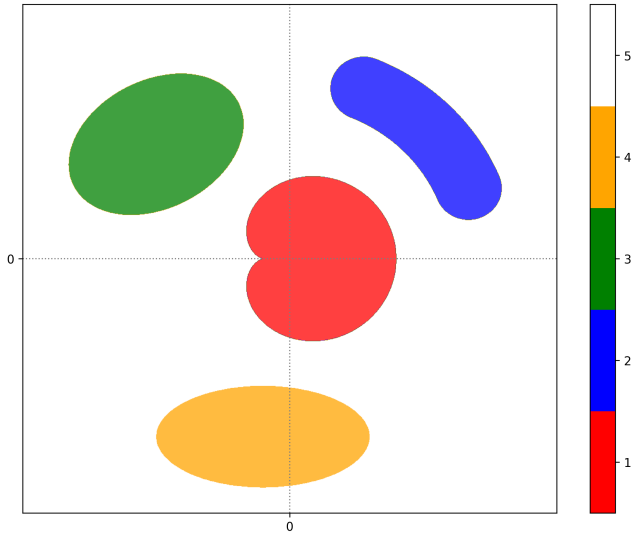


Figure 3.1: Anatomical structures for Example 3.4.2. Red is the target ($i = 1$), while green ($i = 2$), blue ($i = 3$), and orange ($i = 4$) are specific OARs. White denotes the non-target body voxels ($i = 5$).

where p^s_{opt} is the optimal objective of problem (3.9). In our simple experiments, we have found that an initial linearization point of $d^0 = 0$ and threshold of $\epsilon = 10^{-3}$ produce good results.

3.4.2 Illustrative example

Problem instance. We consider an example with $n = 1000$ beams divided into 50 bundles of 20 parallel beams each, positioned evenly around a half-circle. There are $K = 5$ structures, a single target $\mathcal{T} = \{1\}$ and four OARs (including generic body voxels) depicted in Figure 3.1. Treatment takes place over $T = 20$ sessions, so the basic problem has $nT + 2KT = 20200$ variables.

The patient's initial health status is $h_0 = (1, 0, 0, 0, 0)$. His status evolves according

to equation (3.2) with

$$\begin{aligned}\alpha_t &= (0.01, 0.50, 0.25, 0.15, 0.005),\\ \beta_t &= (0.001, 0.05, 0.025, 0.015, 0.0005),\\ \gamma_t &= (0.05, 0, 0, 0, 0)\end{aligned}$$

over all sessions $t = 1, \ldots, T$.

We set the health penalty function to

$$\psi_t(h_t) = (h_{t1})_+ + \sum_{i=2}^{5} (h_{ti})_-, \quad t = 1, \ldots, T.$$

This function penalizes positive statuses of the target and negative statuses of the OARs. Moreover, we constrain the target's health status to be $h_{t1} \leq 2.0$ for $t = 1, \ldots, 15$ and $h_{t1} \leq 0.05$ for the remaining sessions, and we enforce a bound on the other structures' health statuses of $(h_{t2}, h_{t3}, h_{t4}, h_{t5}) \geq (-1.0, -2.0, -2.0, -3.0)$. Thus,

$$H_t = \begin{cases} (2.0, -1.0, -2.0, -2.0, -3.0) & t = 1, \ldots, 15 \\ (0.05, -1.0, -2.0, -2.0, -3.0) & t = 16, \ldots, T. \end{cases}$$

For the dose penalty function, we choose

$$\phi_t(d_t) = \sum_{i=1}^{4} d_{ti}^2 + 0.25 d_{t5}^2, \quad t = 1, \ldots, T.$$

In addition, we restrict the dose and beam intensity to be no more than $D_t = 20$ and $B_t = 1.0$, respectively, over all sessions t.

Computational details. We implemented Algorithm 3.4.1 in Python using CVXPY (Diamond and Boyd, 2016) and solved problem (3.9) with MOSEK (Andersen and Andersen, 2000). From an initial $d^0 = 0$ and $\lambda = 10^4$, the algorithm converged in 11 iterations to a threshold of $\epsilon = 10^{-3}$. Total runtime was approximately 17 seconds on a 64-bit Ubuntu OS desktop with 8 4-core Intel i7-4790k / 4.00 GHz CPUs and

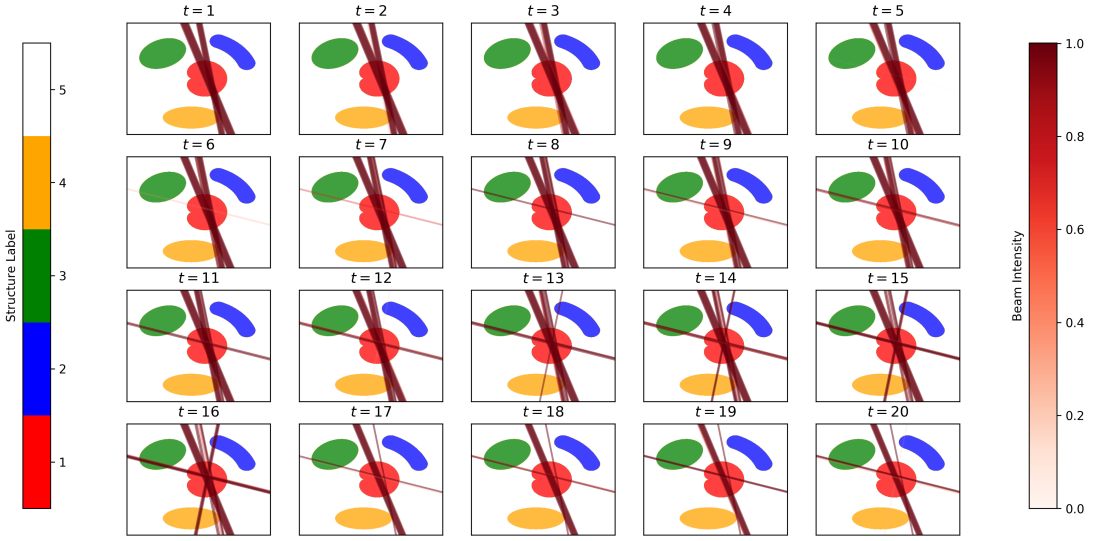


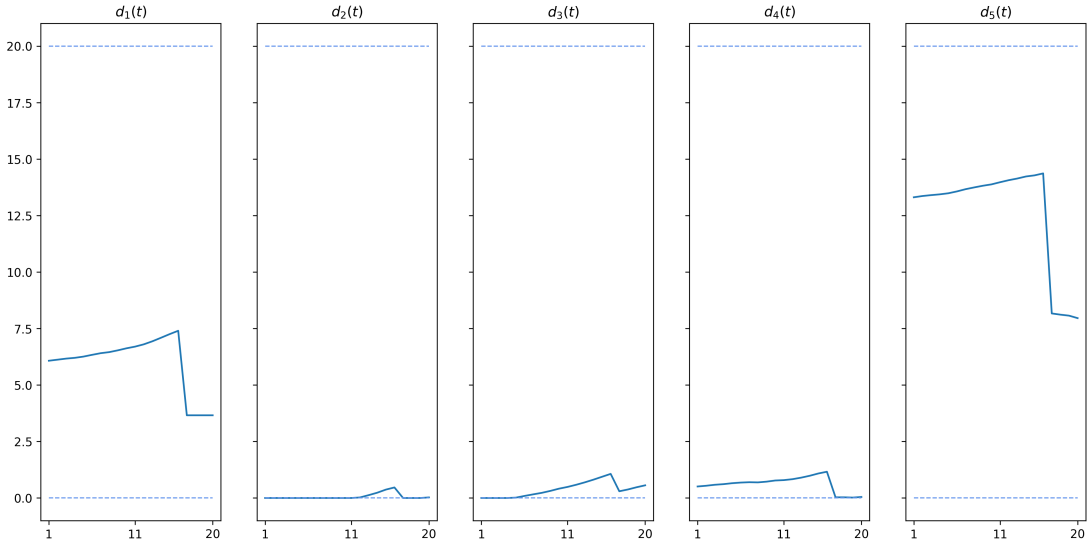
Figure 3.2: Optimal beam intensities for Example 3.4.2.

16 GB of RAM.

Results and analysis. The optimal treatment plan is depicted in Figure 3.2. Beams are densely clustered diagonal from the vertical, striking the target while largely sparing the OARs. As the sessions continue, the number of beams slowly increases, damaging some of the less sensitive organs ($i = 3$ and 4). Then at $t = 16$, when the target's health bound becomes more stringent, the beam density drops precipitously so that only a narrow bundle remains focused on the target, keeping its health status at the desired level.

Figure 3.3 shows the radiation dose and health status resulting from this plan. The latter was computed by plugging the optimal dose into equation (3.2). Total dose to the target ($i = 1$) and body voxels ($i = 5$) far exceed the dose to any other structures. By the end of treatment, the target's health status has fallen to a steady 0.05, while the health statuses of the OARs remain within their respective lower limits.

(a) Dose Trajectories



(b) Health Trajectories

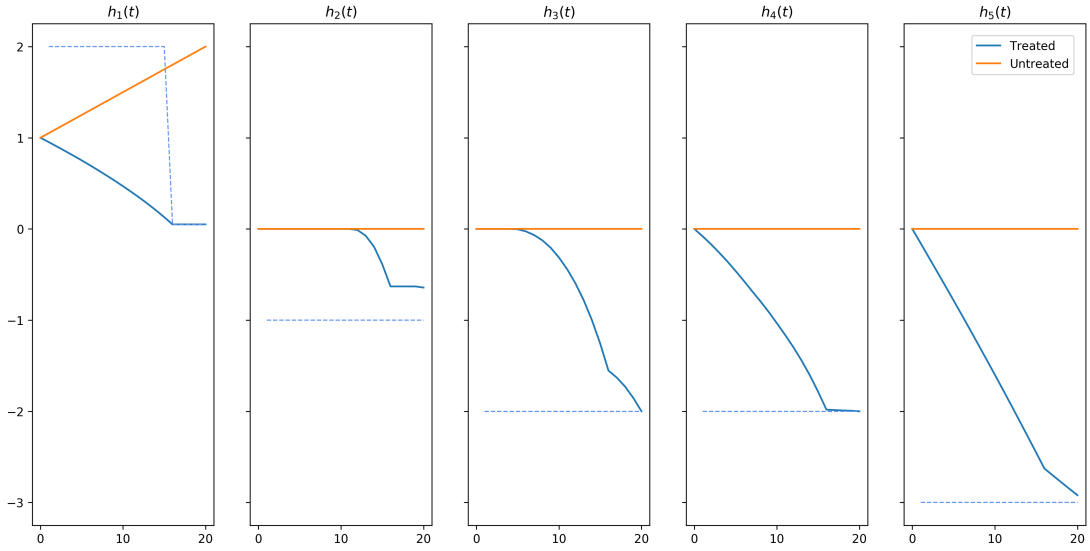


Figure 3.3: Optimal (a) radiation dose and (b) health status trajectories for Example 3.4.2.

3.5 Model predictive control

3.5.1 Algorithm description

So far, we have assumed that at the time of planning, f_t perfectly captures the health dynamics from $t = 1, \ldots, T$. This is rarely true in practice. A patient's anatomy changes unpredictably between sessions, affecting the dispersion of radiation beams and the course of their health status. We can incorporate these changes into problem (3.4) using model predictive control (MPC).

MPC is a powerful technique for automatic control of complex, nonlinear, stochastic systems. It performs extremely well even when the dynamics are approximated by a simple model, since the system's state is updated regularly and new information is incorporated into the solution. This is particularly fitting for radiation treatment planning.

As is customary in MPC, we first convert the state variable constraints in the original problem into soft constraints, *i.e.*, we remove the inequality constraints on h in (3.4) and add a penalty for violating them to the objective. Let $c_\tau : \mathbf{R}^K \to \mathbf{R}$ be the corresponding *health violation penalty function*, defined as

$$c_\tau(h_\tau) = \sum_{i \in \mathcal{T}} (h_{\tau i} - H_{\tau i})_+ + \sum_{i \notin \mathcal{T}} (H_{\tau i} - h_{\tau i})_+, \quad \tau = 1, \ldots, T.$$

This penalty function allows us to accommodate new and unexpected changes in the patient's health, such as the metastasis of a tumor that renders it impossible to control without exceeding the health damage limit of an OAR.

We are now ready to describe MPC for our model. At the beginning of each session t, we observe A_t, f_t, and the patient's true health status, h_{t-1}, then form the problem

$$\begin{array}{lll} \text{minimize} & \sum_{\tau=t}^{T} \phi_\tau(d_\tau) + \sum_{\tau=t}^{T} \psi_\tau(h_\tau) + \eta \sum_{\tau=t}^{T} c_\tau(h_\tau) & \\ \text{subject to} & h_\tau = f_t(h_{\tau-1}, d_\tau), & \tau = t, \ldots, T, \\ & d_\tau = A_t b_\tau, \quad 0 \leq d_\tau \leq D_\tau, \quad 0 \leq b_\tau \leq B_\tau, & \tau = t, \ldots, T \end{array} \tag{3.11}$$

with variables $(b_t, \ldots, b_T)$, $(d_t, \ldots, d_T)$, and $(h_t, \ldots, h_T)$ and violation penalty parameter $\eta > 0$. Since c_τ is convex, problem (3.11) is convex and can be solved using a slight variation on Algorithm 3.4.1. Let $\bar{b} = (\bar{b}_t, \ldots, \bar{b}_T)$ be the optimal treatment plan. We carry out only the first treatment, $\bar{b}_t$, and update our observations A_{t+1}, f_{t+1}, and h_t based on the patient's response. This process repeats until all T sessions have been completed.

3.5.2 Illustrative example

Problem instance. We return to the setting of Example 3.4.2, except now, the health dynamics are modeled with some error. Specifically, let h_{t-1} be the patient's health status at the beginning of session t and d_t the dose delivered during session t. Our model predicts the status will become $\hat{h}_t = f_t(h_{t-1}, d_t)$. In fact, at the beginning of the next session, we observe the true health status to be

$$(h_t)_i = \begin{cases} \max(\hat{h}_t + \omega_t, 0)_i & i \in \mathcal{T} \\ \min(\hat{h}_t + \omega_t, 0)_i & i \notin \mathcal{T} \end{cases},$$

where $\omega_t \in \mathbf{R}^K$ is drawn from $N(\mu, \sigma^2 I)$. This random process continues for $t = 1, \ldots, T$.

For this example, we choose $\mu = 0$ and $\sigma = 0.1$. The rest of the functions and parameter values are identical to 3.4.2. In particular, we still employ the LQ model (3.2) with constant α_t, β_t, and γ_t even though the health status is now stochastic. We plan the treatment using MPC with $\eta = 10^4$ and compare the results to those generated by the naive approach, which simply solves problem (3.4) once prior to session 1.

Computational details. We solved problem (3.11) using Algorithm 3.4.1 with $\lambda = 10^4$ and $\epsilon = 10^{-3}$. For the initial dose in session 1, we chose $d^0 = 0$. In each subsequent session t, we set d^0 to be the (truncated) optimal dose point from the previous session, $(d_t^\star, \ldots, d_T^\star)$. With these parameters, the algorithm took an average of 7 iterations per session to achieve convergence; most runs completed in only 3–4

iterations. The total runtime was 116 seconds.

Results and analysis. Figure 3.4 depicts the treatment plan output by MPC. Most beams are aimed slightly diagonal from the vertical, similar to the naive plan (Figure 3.2) up to session 14. Then, the bundles of beams start to grow sparser and fan out, hitting more areas of the OARs. This sparse irradiation pattern continues until the final session, when there is a brief spike in intensity to bring the target's health status into the desired range.

In Figure 3.5a, we plot the dose trajectories of the MPC plan (green) and the naive plan (blue). The MPC curves are more jagged with a large spike at the end of treatment. However, in each structure, the area under the MPC and naive dose curves remains on par. Thus, we conclude that the MPC plan delivers about the same amount of radiation as the naive plan, only spread across a wider range of beam angles/intensities so as to compensate for uncertainty in the health dynamics model.

This strategy results in better patient health as shown in Figure 3.5b. The MPC plan reduces the target's health status to 0.05, while maintaining the health status of the OARs at a high level. Indeed, the health of these organs under the MPC plan exceeds their health under the naive plan by a significant margin in all but structure 4, where the two are relatively equal up until the last session.

3.6 Operator splitting

MPC enables us to robustly handle uncertainty over time. However, another challenge in radiation treatment planning is the sheer size of problems, which makes them computationally difficult to solve in practice. A typical case with $K = 15$ and $n = 10^4$ requires approximately 10^5 floating-point operations for the beam-to-dose calculation alone. Over a month of sessions, that comes out to 4.5 million operations on a single machine.

In this section, we propose a fast, efficient method for solving the radiation treatment planning problem using operator splitting. Our method is distributed and scales readily with the number of beams as well as the length of treatment. It can

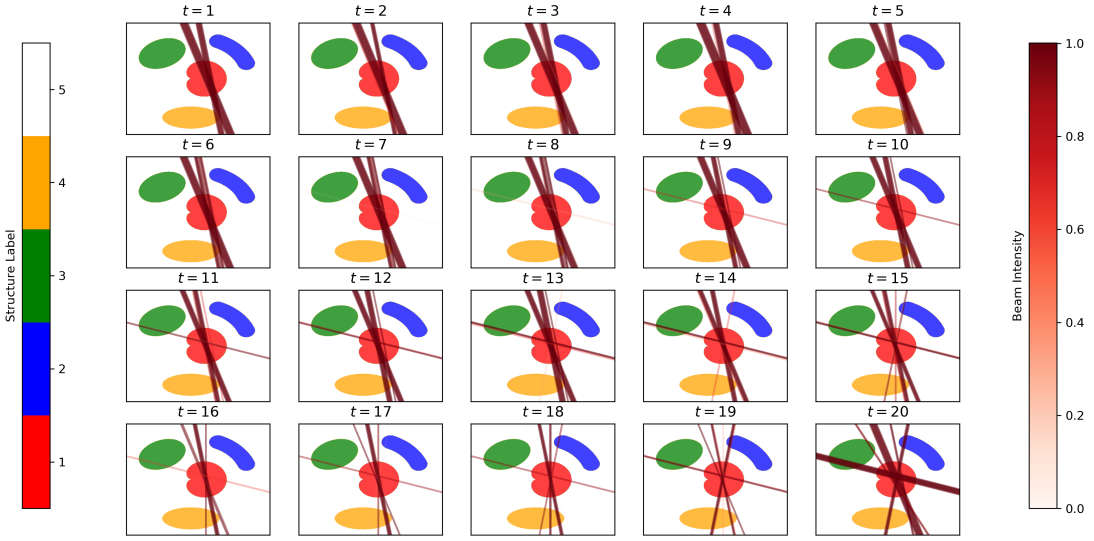


Figure 3.4: Optimal beam intensities for Example 3.5.2 using MPC.

be applied both to the original problem (3.4) and the soft constrained MPC variant (3.11). Below, we describe the mathematical details for the former; the latter is a straightforward extension.

3.6.1 Consensus form

We first rewrite problem (3.4) in an equivalent consensus form:

$$\begin{array}{lll} \text{minimize} & \sum_{t=1}^{T} \phi_t(d_t) + \sum_{t=1}^{T} \psi_t(h_t) & \\ \text{subject to} & h_t = f_t(h_{t-1}, \tilde{d}_t), \quad 0 \leq \tilde{d}_t \leq D_t, & t = 1, \ldots, T, \\ & h_{ti} \leq H_{ti}, \ i \in \mathcal{T}, \quad h_{ti} \geq H_{ti}, \ i \notin \mathcal{T}, & t = 1, \ldots, T, \\ & d_t = A_t b_t, \quad 0 \leq d_t \leq D_t, \quad 0 \leq b_t \leq B_t, & t = 1, \ldots, T, \\ & d_t = \tilde{d}_t, & t = 1, \ldots, T \end{array} \tag{3.12}$$

with additional variable $\tilde{d} = (\tilde{d}_1, \ldots, \tilde{d}_T)$. This splits the problem into two parts, one that encapsulates the radiation physics and the other that contains the health dynamics. The parts share no variables. They are only linked by the consensus constraint, $d_t = \tilde{d}_t$, which requires their doses be equal.

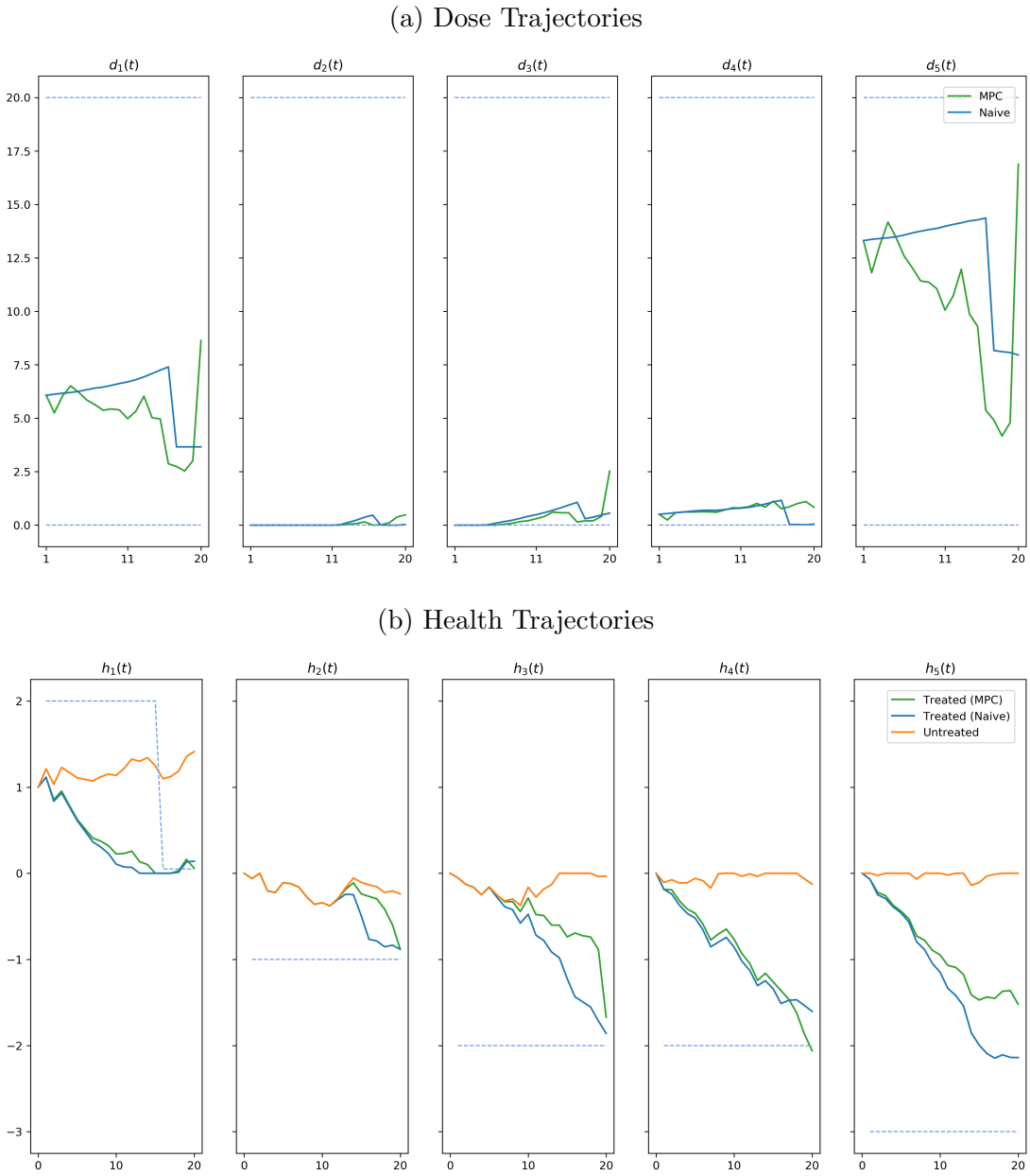


Figure 3.5: Optimal (a) radiation dose and (b) health status trajectories for Example 3.5.2 using MPC (green) and a naive planning approach (blue). The MPC plan's health trajectories all remain within the desired bounds, despite the error in the health dynamics model.

3.6.2 ADMM

We solve problem (3.12) using an iterative algorithm called the alternating direction method of multipliers (ADMM) (Boyd et al., 2010). In ADMM, the beams and health statuses are optimized separately, taking into account the difference between their resulting dose values. This difference is associated with a dual variable $u = (u_1, \ldots, u_T)$, where each $u_t \in \mathbf{R}^K$, which is updated every iteration in order to promote consensus.

Algorithm 3.6.1 *ADMM algorithm.*

input: initial point $(\tilde{d}^0, u^0)$, parameter $\rho > 0$.

for $k = 0, 1, \ldots$ **do**

1. *Calculate beams.* For $t = 1, \ldots, T$, set the value of (b_t^{k+1}, d_t^{k+1}) to a solution of the problem

$$\begin{array}{ll} \text{minimize} & \phi_t(d_t) + \frac{\rho}{2}\|d_t - \tilde{d}_t^k - u_t^k\|_2^2 \\ \text{subject to} & d_t = A_t b_t, \quad 0 \leq d_t \leq D_t, \quad 0 \leq b_t \leq B_t. \end{array}$$

2. *Calculate health trajectory.* Set the value of $(h^{k+1}, \tilde{d}^{k+1})$ to a solution of the problem

$$\begin{array}{lll} \text{minimize} & \sum_{t=1}^T \psi_t(h_t) + \frac{\rho}{2}\|\tilde{d} - d^{k+1} + u^k\|_2^2 & \\ \text{subject to} & h_t = f_t(h_{t-1}, \tilde{d}_t), \quad 0 \leq \tilde{d}_t \leq D_t, & t = 1, \ldots, T, \\ & h_{ti} \leq H_{ti}, \ i \in \mathcal{T}, \quad h_{ti} \geq H_{ti}, \ i \notin \mathcal{T}, & t = 1, \ldots, T. \end{array}$$

3. *Update dual variables.* $u^{k+1} := u^k + \tilde{d}^{k+1} - d^{k+1}$.

until stopping criterion (3.17) is satisfied.

Here $1/\rho > 0$ may be interpreted as the step size. Notice that the first step of Algorithm 3.6.1 can be parallelized across sessions. We impose the dose bound constraint on both the beam and health subproblems because it produces faster convergence in practice.

Initialization. For complex problems, the initial dose point $\tilde{d}^0$ can have a significant impact on the performance of Algorithm 3.6.1. Below, we describe one heuristic that

produces a good starting point by solving a series of simple optimization problems. We begin by solving the static treatment planning problem

$$\begin{array}{ll} \text{minimize} & \phi_1(d_1) + \psi_1(h_1) + \mu \mathbf{1}^T \zeta \\ \text{subject to} & h_1 = f_1(h_0, d_1), \quad \zeta \geq 0, \\ & h_{1i} \leq H_{Ti},\ i \in \mathcal{T}, \quad h_{1i} \geq H_{Ti} - \zeta_i,\ i \notin \mathcal{T}, \\ & d_1 = A_1 b_1, \quad 0 \leq d_1 \leq \sum_{t=1}^T D_t, \quad 0 \leq b_1 \leq \sum_{t=1}^T B_t \end{array} \tag{3.13}$$

with respect to $b_1 \in \mathbf{R}^n, d_1 \in \mathbf{R}^K, h_1 \in \mathbf{R}^K$, and $\zeta \in \mathbf{R}^K$, where $\mu > 0$ is a slack penalty parameter. A reasonable choice for $\mu = \frac{1}{K - |\mathcal{T}|}$, assuming there is at least one non-target structure. Problem (3.13) is convex and can be easily handled on a single machine (*e.g.*, via interior-point methods) for up to 10^5 beams. Denote the optimal beam intensities by b^{stat}.

Next, we consider the dynamic treatment planning problem in which the beams for each session are restricted to be a scalar multiple of b^{stat},

$$\begin{array}{lll} \text{minimize} & \sum_{t=1}^T \phi_t(d_t) + \sum_{t=1}^T \psi_t(h_t) + \mu \sum_{t=1}^T \mathbf{1}^T \zeta_t & \\ \text{subject to} & h_t = f_t(h_{t-1}, d_t), \quad \zeta_t \geq 0, & t = 1, \ldots, T, \\ & h_{ti} \leq H_{ti},\ i \in \mathcal{T}, \quad h_{ti} \geq H_{ti} - \zeta_{ti},\ i \notin \mathcal{T}, & t = 1, \ldots, T, \\ & d_t = \nu_t A_t b^{\text{stat}}, \quad 0 \leq d_t \leq D_t, \quad \nu_t \geq 0, & t = 1, \ldots, T \end{array} \tag{3.14}$$

with variables $(\nu_1, \ldots, \nu_T), (d_1, \ldots, d_T), (h_1, \ldots, h_T)$, and $(\zeta_1, \ldots, \zeta_T)$, where each $\nu_t \in \mathbf{R}$ and $\zeta_t \in \mathbf{R}^K$. This problem can be solved using a slight variation on Algorithm 3.4.1. (For the initial CCP point, we may use the optimal time-invariant $\nu_t = \nu$ when $\beta_t = 0$; finding this value entails solving a small convex problem). Since there are only $O(TK)$ variables, convergence is generally quick, taking less than 5 iterations in our experiments. We use the resulting doses as our initial dose point for ADMM, *i.e.*, $\tilde{d}_t^0 = \nu_t^\star A_t b^{\text{stat}}$ for $t = 1, \ldots, T$.

Besides providing a good starting point, this initialization heuristic also gives us a way to quickly tune problem parameters. If the health trajectory from $\tilde{d}^0$ is poor, it is much faster to modify weights and re-solve problems (3.13) and (3.14) than it is to re-run the full ADMM algorithm.

Stopping criterion. If problem (3.12) is convex, then under mild conditions, ADMM converges to a solution assuming one exists. Moreover, the primal and dual residuals

$$r^k_{\text{prim}} = d^k - \tilde{d}^k \tag{3.15}$$
$$r^k_{\text{dual}} = \rho(\tilde{d}^k - \tilde{d}^{k-1}) \tag{3.16}$$

also converge to zero. Thus, a reasonable stopping criterion is

$$\|r^k_{\text{prim}}\|_2 \leq \epsilon_{\text{prim}} \quad \text{and} \quad \|r^k_{\text{dual}}\|_2 \leq \epsilon_{\text{dual}}, \tag{3.17}$$

where $\epsilon_{\text{prim}} > 0$ and $\epsilon_{\text{dual}} > 0$ are tolerances for primal and dual feasibility, respectively. Typically, these tolerances are chosen with respect to absolute and relative cutoffs $\epsilon_{\text{abs}} > 0$ and $\epsilon_{\text{rel}} > 0$ using the relation

$$\epsilon_{\text{prim}} = \epsilon_{\text{abs}}\sqrt{TK} + \epsilon_{\text{rel}} \max(\|d^k\|_2, \|\tilde{d}^k\|_2)$$
$$\epsilon_{\text{dual}} = \epsilon_{\text{abs}}\sqrt{TK} + \epsilon_{\text{rel}}\|u^k\|_2.$$

A common choice for $\epsilon_{\text{rel}} = 10^{-3}$, while the choice for ϵ_{abs} depends on the scale of the treatment planning problem (Boyd et al., 2010, Section 3.3.1).

Convergence and choice of ρ. When the problem is convex, *i.e.*, the health dynamics function is affine, Algorithm 3.6.1 converges to a solution for any $\rho > 0$, although the value of ρ may have an impact on the practical convergence rate. When the problem is nonconvex, ADMM is a heuristic and the final beam/dose plan can depend directly on ρ (Boyd et al., 2010, Section 9). The question of how to choose ρ is still unsettled; see Ghadimi et al. (2015); Xu et al. (2017b,a) for further discussion on the topic. We have found that for data on the order of one, values of ρ between 10^{-2} and 10^2 work reasonably well.

Table 3.1: Prostate IMRT: LQ Model Parameters

i	Structure	α_{ti}	β_{ti}	γ_{ti}
1	Prostate	0.15	0.05	$\begin{cases} 0 & t \leq 28 \\ 0.0173 & t > 28 \end{cases}$
2	Urethra	1	0.2	0
3	Bladder	1	0.2	0
4	Rectum	1	0.2	0
5	L. Femoral Head	1	0.25	0
6	R. Femoral Head	1	0.25	0
7	Body	1	0.3333	0

3.6.3 Clinical example

Problem instance. We test our method on a fluence map optimization of a prostate cancer IMRT case with $n = 34848$ beams and $K = 7$ structures consisting of a single PTV ($i = 1$), five OARs, and generic body voxels ($i = 7$). Treatment is carried out over $T = 45$ sessions, so the planning problem has about 1.6 million variables. The matrix A_t remains constant over time and maps the beam intensities to the *average* dose per structure, *i.e.*, $(d_t)_i$ is the total dose to structure i divided by the number of voxels in i. Each beam's intensity cannot exceed $B_t = 0.025$.

The LQ model parameters, initial health status, and dose and health status bounds can be found in Tables 3.1 and 3.2; these have been adapted from prior clinical datasets (Kehwar, 2005; Gao et al., 2010; Marks et al., 2010; van Leeuwen et al., 2018). We choose the health and dose penalty functions to be

$$\psi_t(h_t) = (h_{t1})_+ + \frac{1}{6}\sum_{i=2}^{7}(h_{ti})_-, \quad \phi_t(d_t) = \sum_{i=1}^{6} d_{ti}^2 + 0.25 d_{t7}^2, \quad t = 1, \ldots, T.$$

These penalties place greater importance on reducing the health status of the PTV compared to sparing the OARs or generic body tissue.

Computational details. The computational setup is the same as in Example 3.4.2. To solve the ADMM subproblems, we used MOSEK and ran CCP ($\lambda = 10^4$) on the

Table 3.2: Prostate IMRT: Health and Dose Parameters

i	Structure	h_{0i}	H_{ti}	D_{ti}
1	Prostate	5.8579	$\begin{cases} 5.8579 & t \leq 14 \\ 4.4716 & 15 \leq t \leq 31 \\ 0 & t > 31 \end{cases}$	10
2	Urethra	0	-4.8	10
3	Bladder	0	-4.8	10
4	Rectum	0	-4.8	10
5	L. Femoral Head	0	-3.0	10
6	R. Femoral Head	0	-3.0	10
7	Body	0	-6.0	10

health trajectory subproblem. With $\rho = 80$, ADMM converged in 82 iterations to cutoffs of $\epsilon_{\text{abs}} = 10^{-2}$ and $\epsilon_{\text{rel}} = 10^{-3}$. The normed residuals, $\|r^k_{\text{prim}}\|_2$ and $\|r^k_{\text{dual}}\|_2$, are shown in Figure 3.6. Total runtime was about 43 minutes, with the bulk of that time spent on the main ADMM loop (initialization took only 32 seconds). By contrast, a straightforward application of Algorithm 3.4.1 to this problem required over an hour.

Results and analysis. Figure 3.7 depicts the dose trajectories resulting from the initial plan (green) and the final plan output by ADMM (blue). The initial plan is essentially a piecewise equal-dose fractionation scheme, reflected by the flat plateaus in the corresponding dose trajectories. This already gives us a good approximation of the final plan: both plans maintain a relatively high dose to the PTV of about 0.9 Gy until session 31, then drop off sharply to the same constant doses thereafter. However, during the high dose phase, the final plan gradually increases the dosage over time to all structures except the bladder ($i = 3$). By adapting dynamically to changes in the patient's anatomy, it is able to deliver more dose per session and thus achieve better tumor control, while still respecting the limits on the OARs' health statuses.

Indeed, we see in Figure 3.8 that the final plan exactly attains the desired PTV health status of zero for $t > 31$. It must sacrifice some OARs to do this, reducing the health statuses of the urethra, rectum, and right femoral head ($i = 2, 4$, and 6) to

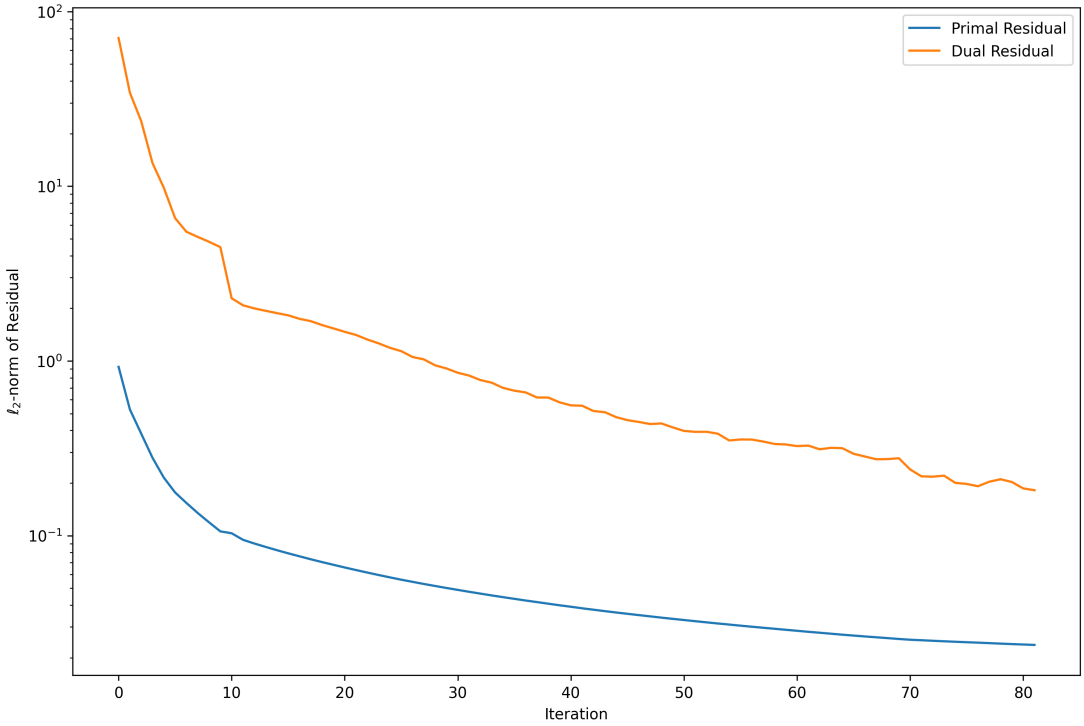


Figure 3.6: Primal and dual residual ℓ_2-norms for Example 3.6.3.

their lower bounds, but never violates those bounds. In fact, by shifting radiation to other structures, the final plan actually improves the health of the bladder over that from the initial plan, which results in a $h_3(t)$ far below the limit of -4.8 for $t \geq 35$. Overall, it is clear that the combination of a solid initialization heuristic and ADMM produces a treatment plan that satisfies or even exceeds all of our clinical goals.

3.7 Implementation

We provide an implementation of our adaptive radiation treatment planning method in AdaRad, an open-source Python software package based on CVXPY (Diamond and Boyd, 2016). Our implementation is fully distributed, leveraging Python's built-in multiprocessing library to execute solves in parallel. Users can quickly import patient data, define clinical goals, construct treatment plans, and visualize the results. They

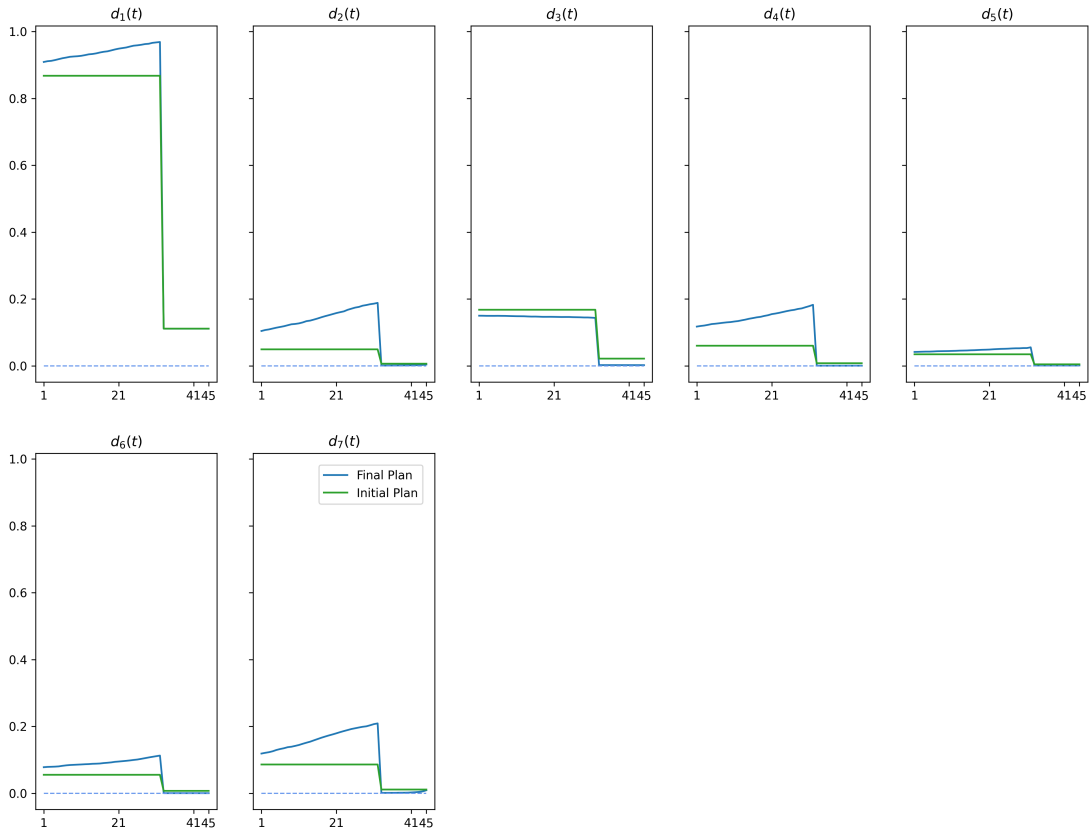


Figure 3.7: Optimal radiation dose trajectory for Example 3.6.3. The initial plan (green) depicts the dose output by the initialization heuristic described in Section 3.6.2, while the final plan (blue) depicts the dose output by the ADMM algorithm.

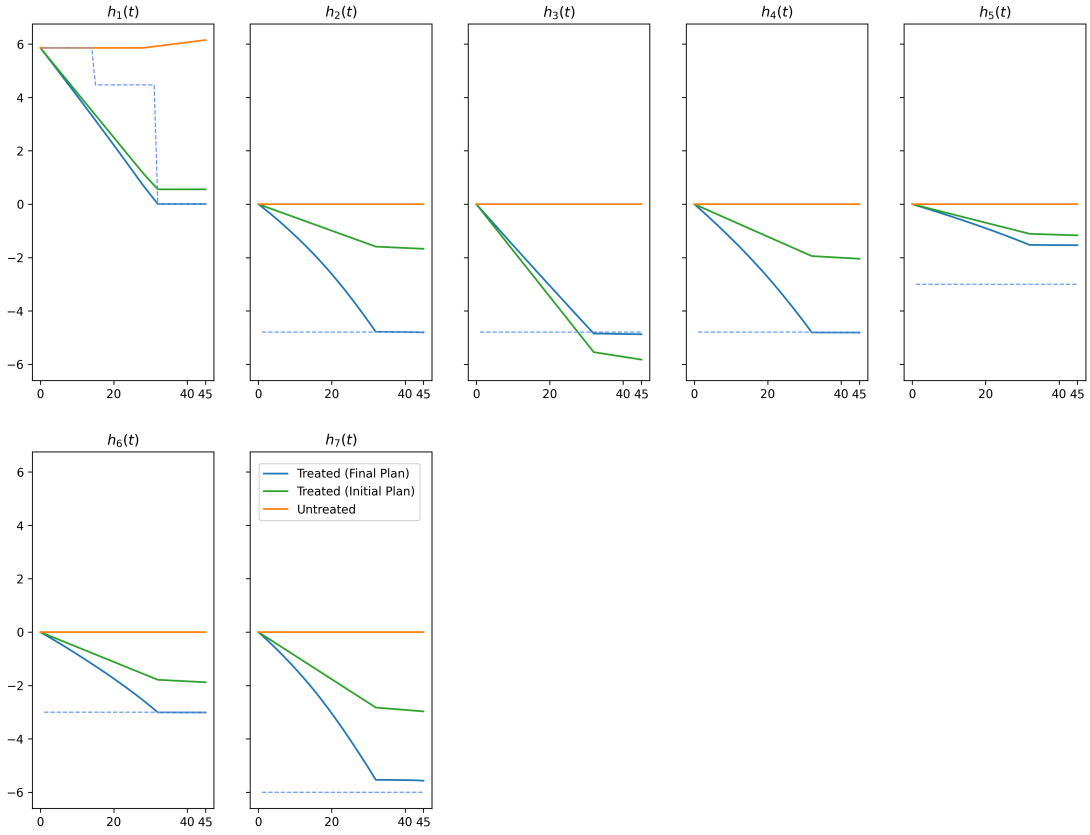


Figure 3.8: Optimal health trajectories resulting from the doses in Figure 3.7.

can also rapidly modify and re-plan a case, allowing for comparisons between different prescriptions and treatment lengths. Moreover, since AdaRad is a Python library, it can be easily integrated with other libraries (*e.g.*, for image processing) used in radiation therapy.

The code below imports some patient data and a prescription, solves for the optimal treatment plan, and plots the resulting dose and health trajectories.

```
import adarad, numpy
from adarad import Case, CasePlotter

# Construct the clinical case.
> case = Case()
> case.import_file("/examples/patient_01-case.yaml")
> case.physics.dose_matrix = numpy.load("/examples/patient_01-dmat.npy")

# Solve using ADMM algorithm.
> status, result = case.plan(slack_weight = 50, max_iter = 100,
                             solver = ECOS, use_admm = True)
> print("Solve status: {}".format(status))
> print("Solve time: {}".format(result.solver_stats.solve_time))
> print("Iterations: {}".format(result.solver_stats.num_iters))

# Plot the dose and health trajectories.
> caseviz = CasePlotter(case)
> caseviz.plot_treatment(result, stepsize = 10)
> caseviz.plot_health(result, stepsize = 10)
```

In this example, the dose matrix A_t is the same for all t and stored in a single `*.npy` file. AdaRad also supports other sparse data representations, such as `scipy.csc_matrix`. To specify a time-varying dose matrix, the user would input a list of matrices in order $[A_1, \ldots, A_T]$.

We start by constructing a `Case`, which contains `Anatomy`, `Physics`, and `Prescription`

objects. The `Anatomy` and `Physics` must be defined prior to planning, either by manually specifying them in the code or importing a case description. A description is a YAML file that contains at minimum the keys `treatment_length` and `structures`, where the latter is a list of anatomical structures $i = 1, \ldots, K$, each of which has a `name`, `is_target` boolean indicator, and `alpha`, `beta`, and `gamma` values corresponding to the LQ model parameters. The initial health status and health and dose bounds may also be specified.

Once the `Case` is defined, we can solve for the optimal treatment plan. The `plan` function implements Algorithms 3.4.1 and 3.6.1 (the latter with `use_admm = True`). It takes as optional input `d_init`: the initial dose point, `use_slack`: a boolean indicating whether to include slack variable δ, `slack_weight`: the slack penalty parameter λ, `max_iter`: the maximum number of iterations, and `solver`: the convex solver to use for the beam and health subproblems. In the above example, we call the solver ECOS (Domahidi et al., 2013), one of several free, open-source solvers packaged with CVXPY. If MOSEK is installed, we can call it as well by passing `solver = MOSEK` into the planning function.

After the algorithm finishes, `plan` saves the results in `case.current_plan` and returns the final solve status along with a `RunRecord` object that carries solver performance data, such as the total runtime, and the optimal variable values. To visualize the resulting plan, we instantiate a `CasePlotter` object and call `plot_treatment` and `plot_health` on the `RunRecord` to display the dose and health trajectories, respectively. We can also extract the optimal beams, doses, and health statuses with, *e.g.*, `result.beams` for further processing.

If we wish to explore alternate plans, we can easily modify the dose and health status constraints of any structure and re-plan the case. Re-planning is generally fast, since AdaRad uses the previously stored solution as a warm start point. In a typical workflow, we may import a prescription formed from general clinical guidelines, then repeatedly adjust the dose/health status bounds until we obtain a treatment plan with our desired properties. The `case.current_plan` will be updated with the new optimal values after each run. To keep a history of plans for comparison, we can save our results in the `Case` by calling `save_plan` before re-optimizing. The code below

provides an example of changing the upper dose bound on the PTV to $D_{ti} = 10$ Gy for all sessions and plotting the dose and health trajectories under this new constraint alongside the trajectories of the original plan.

```
# Save previous treatment plan.
> case.save_plan("Original Plan")

# Constraint allows maximum of 10 Gy per session on the PTV.
> case.prescription["PTV"].dose_upper = 10

# Re-plan the case with new dose constraint.
> status2, result2 = case.plan(slack_weight = 50, max_iter = 100,
                               solver = ECOS, use_admm = True)
> print("Solve status: {}".format(status2))

# Compare original and new treatment plans.
> caseviz.plot_treatment(result2, stepsize = 10, label = "New Plan",
                         plot_saved = True)
> caseviz.plot_health(result2, stepsize = 10, label = "New Plan",
                      plot_saved = True)
```

For more details on AdaRad's functions as well as additional examples, see the documentation at `https://github.com/anqif/adarad`.

3.8 Conclusion

To achieve the best outcomes, radiation therapy must adapt to new information about the patient's health and anatomy during treatment. We have described one method for adaptive radiation treatment planning using an operator splitting algorithm. Our method is highly scalable, parallelizable, and can efficiently handle a large number of beams and sessions. Moreover, it is robust to errors in the patient's health response

model, as well as other sources of uncertainty in the clinic. We demonstrated its effectiveness on a large prostate cancer case and showed that the resulting plan improves markedly on a standard equal-dose fractionation scheme.

Chapter 4

A Domain Specific Language for Convex Optimization

4.1 Introduction

In this chapter, we shift our focus to general convex optimization, particularly optimization applied to problems in statistical modeling. Some examples of optimization-based models are least squares, ridge and lasso regression, isotonic regression, Huber regression, support vector machines, and sparse inverse covariance estimation. Koenker and Mizera (2014) discuss the role of convex optimization in statistics and provide a survey of packages for solving such problems in R (R Core Team, 2020). Our package, CVXR (Fu et al., 2020b), solves a broad class of convex optimization problems, which includes those noted above as well as many other models and methods in statistics.

Similar systems already exist, such as CVX (Grant and Boyd, 2014) and YALMIP (Lofberg, 2004) in MATLAB (The MathWorks Inc., 2019), CVXPY (Diamond and Boyd, 2016) in Python (van Rossum et al., 2011), and Convex.jl (Udell et al., 2014) in Julia (Bezanson et al., 2012). CVXR brings these capabilities to R, providing a domain-specific language (DSL) that allows users to easily formulate and solve new problems for which custom code does not exist. As an illustration, suppose we are given $X \in \mathbf{R}^{m \times n}$ and $y \in \mathbf{R}^m$, and we want to solve the ordinary least squares (OLS)

problem

$$\underset{\beta}{\text{minimize}} \quad \|y - X\beta\|_2^2$$

with optimization variable $\beta \in \mathbf{R}^n$. This problem has a well-known analytical solution, which can be determined using `lm` in the default stats package. In CVXR, we can solve for β using the code

```
R> beta <- Variable(n)
R> obj <- sum((y - X %*% beta)^2)
R> prob <- Problem(Minimize(obj))
R> result <- solve(prob)
```

The first line declares our variable, the second line forms our objective function, the third line defines the optimization problem, and the last line solves this problem by converting it into a second-order cone program and sending it to one of CVXR's solvers. The results are retrieved with

```
R> result$value             # Optimal objective
R> result$getValue(beta)    # Optimal variables
R> result$solve_time        # Solver runtime
```

This code runs slower and requires additional set-up at the beginning. So far, it does not look like an improvement on `stats::lm`. However, suppose we add a constraint to our problem:

$$\begin{aligned} \underset{\beta}{\text{minimize}} \quad & \|y - X\beta\|_2^2 \\ \text{subject to} \quad & \beta_j \leq \beta_{j+1}, \quad j = 1, \ldots, n-1. \end{aligned}$$

This is a special case of isotonic regression. Now, we can no longer use `stats::lm` for the optimization. We would need to find another R package tailored to this type of problem such as nnls (Mullen and van Stokkum, 2012) or write our own custom solver. With CVXR though, we need only add the constraint as a second argument to the problem:

```
R> prob <- Problem(Minimize(obj), list(diff(beta) >= 0))
```

Our new problem definition includes the coefficient constraint, and a call to `solve` will produce its solution. In addition to the usual results, we can get the dual variables with

```
R> result$getDualValue(constraints(prob)[[1]])
```

This example demonstrates CVXR's chief advantage: flexibility. Users can quickly modify and re-solve a problem, making our package ideal for prototyping new statistical methods. Its syntax is simple and mathematically intuitive. Furthermore, CVXR combines seamlessly with native R code as well as several popular packages, allowing it to be incorporated easily into a larger analytical framework. The user can, for instance, apply resampling techniques like the bootstrap to estimate variability, as we show in Section 4.3.2.

DSLs for convex optimization are already widespread on other application platforms. In R, users have access to the packages listed in the CRAN Task View for *Optimization and Mathematical Programming* (Theußl et al., 2020). Packages like optimx (Nash and Varadhan, 2011) and nloptr (Johnson, 2008) provide access to a variety of general algorithms, which can handle nonlinear and certain classes of nonconvex problems. CVXR, on the other hand, offers a language to express convex optimization problems using R syntax, along with a tool for analyzing and restructuring them for the solver best suited to their type. ROI (Theußl et al., 2017) is perhaps the package closest to ours in spirit. It offers an object-oriented framework for defining optimization problems, but still requires users to explicitly identify the type of every objective and constraint, whereas CVXR manages this process automatically.

In the next section, we provide a brief mathematical overview of convex optimization. Interested readers can find a full treatment in Boyd and Vandenberghe (2004). Then we give a series of examples ranging from basic regression models to semidefinite programming, which demonstrate the simplicity of problem construction in CVXR. Finally, we describe the implementation details before concluding. Our package and the example code for this chapter are available on the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=CVXR` and the official CVXR site at `https://cvxr.rbind.io`.

4.2 Disciplined convex optimization

The general convex optimization problem is of the form

$$\begin{array}{ll} \underset{v}{\text{minimize}} & f_0(v) \\ \text{subject to} & f_i(v) \leq 0, \quad i = 1, \ldots, M \\ & Av = b, \end{array}$$

where $v \in \mathbf{R}^n$ is our variable of interest, and $A \in \mathbf{R}^{m \times n}$ and $b \in \mathbf{R}^n$ are constants describing our linear equality constraints. The objective and inequality constraint functions $f_0, \ldots, f_M$ are convex, i.e., they are functions $f_i : \mathbf{R}^n \to \mathbf{R}$ that satisfy

$$f_i(\theta u + (1-\theta)v) \leq \theta f_i(u) + (1-\theta) f_i(v)$$

for all $u, v \in \mathbf{R}^n$ and $\theta \in [0, 1]$. This class of problems arises in a variety of fields, including machine learning and statistics.

A number of efficient algorithms exist for solving convex problems (Wright, 1997; Boyd et al., 2010; Andersen et al., 2011; Skajaa and Ye, 2015). However, it is unnecessary for the CVXR user to know the operational details of these algorithms. CVXR provides a DSL that allows the user to specify the problem in a natural mathematical syntax. This specification is automatically converted into the standard form ingested by a generic convex solver. See Section 4.4 for more on this process.

In general, it can be difficult to determine whether an optimization problem is convex. We follow an approach called disciplined convex programming (DCP; Grant et al., 2006) to define problems using a library of basic functions (atoms), whose properties like curvature, monotonicity, and sign are known. Adhering to the DCP rule,

$f(g_1, \ldots, g_k)$ is convex if f is convex and for each $i = 1, \ldots, k$, either

- g_i is affine,
- g_i is convex and f is increasing in argument i, or
- g_i is concave and f is decreasing in argument i,

we combine these atoms such that the resulting problem is convex by construction. Users will need to become familiar with this rule if they wish to define complex problems.

The library of available atoms is provided in the documentation. It covers an extensive array of functions, enabling any user to model and solve a wide variety of sophisticated optimization problems. In the next section, we provide sample code for just a few of these problems, many of which are cumbersome to prototype or solve with other R packages.

4.3 Examples

In the following examples, we are given a dataset (x_i, y_i) for $i = 1, \ldots, m$, where $x_i \in \mathbf{R}^n$ and $y_i \in \mathbf{R}$. We represent these observations in matrix form as $X \in \mathbf{R}^{m \times n}$ with stacked rows $x_i^\top$ and $y \in \mathbf{R}^m$. Generally, we assume that $m > n$.

4.3.1 Regression

Robust (Huber) regression

In Section 4.1, we saw an example of OLS in CVXR. While least squares is a popular regression model, one of its flaws is its high sensitivity to outliers. A single outlier that falls outside the tails of the normal distribution can drastically alter the resulting coefficients, skewing the fit on the other data points. For a more robust model, we can fit a Huber regression (Huber, 1964) instead by solving

$$\underset{\beta}{\text{minimize}} \quad \sum_{i=1}^m \phi(y_i - x_i^\top \beta)$$

for variable $\beta \in \mathbf{R}^n$, where the loss is the Huber function with threshold $M > 0$,

$$\phi(u) = \begin{cases} \frac{1}{2}u^2 & \text{if } |u| \leq M \\ M|u| - \frac{1}{2}M^2 & \text{if } |u| > M. \end{cases}$$

This function is identical to the least squares penalty for small residuals, but on large residuals, its penalty is lower and increases linearly rather than quadratically. It is thus more forgiving of outliers.

In CVXR, the code for this problem is

```
R> beta <- Variable(n)
R> obj <- sum(huber(y - X %*% beta, M))
R> prob <- Problem(Minimize(obj))
R> result <- solve(prob)
```

Note the similarity to the OLS code. As before, the first line instantiates the n-dimensional optimization variable, and the second line defines the objective function by combining this variable with our data using CVXR's library of atoms. The only difference this time is we call the `huber` atom on the residuals with threshold `M`, which we assume has been set to a positive scalar constant. Our package provides many such atoms to simplify problem definition for the user.

Quantile regression

Another variation on least squares is quantile regression (Koenker, 2005). The loss is the tilted l_1 function,

$$\phi(u) = \tau \max(u, 0) - (1 - \tau) \max(-u, 0) = \frac{1}{2}|u| + \left(\tau - \frac{1}{2}\right) u,$$

where $\tau \in (0, 1)$ specifies the quantile. The problem as before is to minimize the total residual loss. This model is commonly used in ecology, healthcare, and other fields where the mean alone is not enough to capture complex relationships between variables. CVXR allows us to create a function to represent the loss and integrate it seamlessly into the problem definition, as illustrated below.

```
R> quant_loss <- function(u, tau) 0.5 * abs(u) + (tau - 0.5) * u
R> obj <- sum(quant_loss(y - X %*% beta, t))
R> prob <- Problem(Minimize(obj))
R> result <- solve(prob)
```

Here `t` is the user-defined quantile parameter. We do not need to create a new '`Variable`' object, since we can reuse `beta` from the previous example.

By default, the `solve` method automatically selects the CVXR solver most specialized to the given problem's type. This solver may be changed by passing in an additional `solver` argument. For instance, the following line fits our quantile regression with SCS (O'Donoghue et al., 2016).

```
R> result <- solve(prob, solver = "SCS")
```

Elastic net regularization

Often in applications, we encounter problems that require regularization to prevent overfitting, introduce sparsity, facilitate variable selection, or impose prior distributions on parameters. Two of the most common regularization functions are the l_1-norm and squared l_2-norm, combined in the elastic net regression model (Hastie and Zou, 2005; Friedman et al., 2010),

$$\underset{\beta}{\text{minimize}} \quad \frac{1}{2m}\|y - X\beta\|_2^2 + \lambda(\frac{1-\alpha}{2}\|\beta\|_2^2 + \alpha\|\beta\|_1).$$

Here $\lambda \geq 0$ is the overall regularization weight and $\alpha \in [0, 1]$ controls the relative l_1 versus squared l_2 penalty. Thus, this model encompasses both ridge ($\alpha = 0$) and lasso ($\alpha = 1$) regression.

To solve this problem in CVXR, we first define a function that calculates the regularization term given the variable and penalty weights.

```
R> elastic_reg <- function(beta, lambda = 0, alpha = 0) {
+    ridge <- (1 - alpha) * sum(beta^2)
+    lasso <- alpha * p_norm(beta, 1)
+    lambda * (lasso + ridge)
+  }
```

Then, we add it to the scaled least squares loss.

```
R> loss <- sum((y - X %*% beta)^2)/(2 * m)
```

```
R> obj <- loss + elastic_reg(beta, lambda, alpha)
R> prob <- Problem(Minimize(obj))
R> result <- solve(prob)
```

The advantage of this modular approach is that we can easily incorporate elastic net regularization into other regression models. For instance, if we wanted to run regularized Huber regression, CVXR allows us to reuse the above code with just a single changed line,

```
R> loss <- sum(huber(y - X %*% beta, M))
```

Logistic regression

Suppose now that $y_i \in \{0, 1\}$ is a binary class indicator. One of the most popular methods for binary classification is logistic regression (Cox, 1958; Freedman, 2009). We model the conditional response as $y|x \sim \text{Bernoulli}(g_\beta(x))$, where $g_\beta(x) = \frac{1}{1+e^{-x^\top \beta}}$ is the logistic function, and maximize the log-likelihood function, yielding the optimization problem

$$\underset{\beta}{\text{maximize}} \quad \sum_{i=1}^{m} \{y_i \log(g_\beta(x_i)) + (1 - y_i) \log(1 - g_\beta(x_i))\}.$$

CVXR provides the `logistic` atom as a shortcut for $f(z) = \log(1 + e^z)$, so our problem is succinctly expressed as

```
R> obj <- -sum(X[y == 0, ] %*% beta) - sum(logistic(-X %*% beta))
R> prob <- Problem(Maximize(obj))
R> result <- solve(prob)
```

The user may be tempted to type `log(1 + exp(X %*% beta))` as in conventional R syntax. However, this representation of $f(z)$ violates the DCP composition rule, so the CVXR parser will reject the problem even though the objective is convex. Users who wish to employ a function that is convex, but not DCP compliant should check the documentation for a custom atom or consider a different formulation.

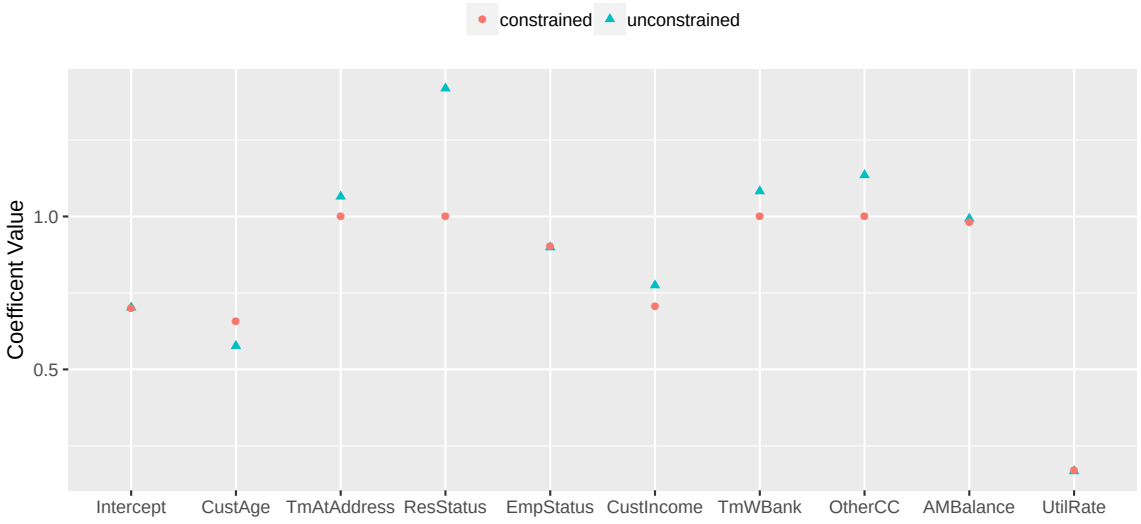


Figure 4.1: Logistic regression with constraints using data from The MathWorks Inc. (2018). The addition of constraint (4.1) moves the coefficients for customer age and customer income closer to each other.

We can retrieve the optimal objective and variables just like in OLS. More interestingly, we can evaluate various functions of these variables as well by passing them directly into `result$getValue`. For instance, the log-odds are

```
R> log_odds <- result$getValue(X %*% beta)
```

This will coincide with the ratio we get from computing the probabilities directly:

```
R> beta_res <- result$getValue(beta)
R> y_probs <- 1 / (1 + exp(-X %*% beta_res))
R> log(y_probs / (1 - y_probs))
```

We illustrate with a logistic regression fit from a credit scoring example (The MathWorks Inc., 2018). The nine regression coefficients other than the intercept are constrained to be in the unit interval. To reflect the correlation between two of the covariates, customer age (x_2) and customer income (x_6), an additional constraint is placed on the respective coefficients β_2 and β_6:

$$|\beta_2 - \beta_6| \leq 0.5. \tag{4.1}$$

The code below demonstrates how the latter constraint can be specified by seamlessly combining familiar R functions such as `abs` with standard indexing constructs.

```
R> constr <- list(beta[2:10] >= 0, beta[2:10] <= 1,
+    abs(beta[2] - beta[6]) <= 0.05)
R> prob <- Problem(Maximize(obj), constr)
R> result <- solve(prob)
R> beta_res_con <- result$getValue(beta)
```

Figure 4.1 compares the unconstrained and constrained fits and shows that the addition of constraint (4.1) pulls the coefficient estimates for customer age and customer income towards each other.

Many other classification methods belong to the convex framework. For example, the support vector classifier is the solution of a l_2-norm minimization problem with linear constraints, which we have already shown how to model. Support vector machines are a straightforward extension. The multinomial distribution can be used to predict multiple classes, and estimation via maximum likelihood produces a convex problem. To each of these methods, we can easily add new penalties, variables, and constraints in CVXR, allowing us to adapt to a specific dataset or environment.

Sparse inverse covariance estimation

Assume we are given i.i.d. observations $x_i \sim N(0, \Sigma)$ for $i = 1, \ldots, m$, and the covariance matrix $\Sigma \in \mathbf{S}_+^n$, the set of symmetric positive semidefinite matrices, has a sparse inverse $S = \Sigma^{-1}$. Let $Q = \frac{1}{m-1} \sum_{i=1}^m (x_i - \bar{x})(x_i - \bar{x})^\top$ be our sample covariance. One way to estimate Σ is to maximize the log-likelihood with an l_1-norm constraint (Yuan and Lin, 2007; Banerjee et al., 2008; Friedman et al., 2008), which amounts to the optimization problem

$$\begin{array}{ll} \underset{S}{\text{maximize}} & \log\det(S) - \text{tr}(SQ) \\ \text{subject to} & S \in \mathbf{S}_+^n, \quad \sum_{i=1}^n \sum_{j=1}^n |S_{ij}| \leq \alpha. \end{array}$$

The parameter $\alpha \geq 0$ controls the degree of sparsity. Our problem is convex, so we can solve it with

```
R> S <- Variable(n, n, PSD = TRUE)
R> obj <- log_det(S) - matrix_trace(S %*% Q)
R> constr <- list(sum(abs(S)) <= alpha)
R> prob <- Problem(Maximize(obj), constr)
R> result <- solve(prob, solver = "SCS")
```

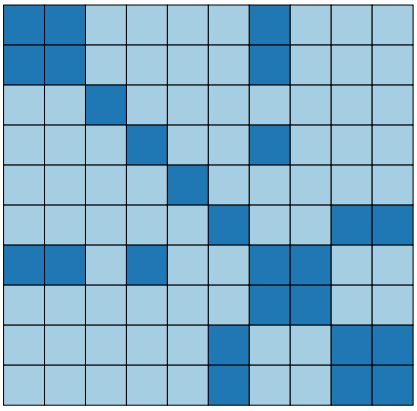
The `PSD = TRUE` argument to the `Variable` constructor restricts `S` to the positive semidefinite cone. In our objective, we use CVXR functions for the log-determinant and trace. The expression `matrix_trace(S %*% Q)` is equivalent to `sum(diag(S %*% Q))`, but the former is preferred because it is more efficient than making nested function calls. However, a standalone atom does not exist for the determinant, so we cannot replace `log_det(S)` with `log(det(S))` since `det` is undefined for a '`Variable`' object.

Figure 4.2 depicts the solutions for a particular dataset with $m = 1000, n = 10$, and S containing 26% non-zero entries represented by the black squares in the top left image. The sparsity of our inverse covariance estimate decreases for higher α, so that when $\alpha = 1$, most of the off-diagonal entries are zero, while if $\alpha = 10$, over half the matrix is dense. At $\alpha = 4$, we achieve the true percentage of non-zeros.

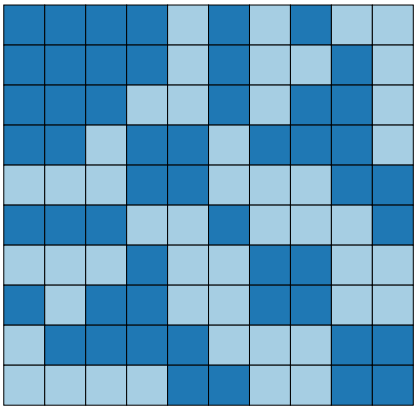
Saturating hinges

The following example comes from work on saturating splines in Boyd et al. (2018). Adaptive regression splines are commonly used in statistical modeling, but the instability they exhibit beyond their boundary knots makes extrapolation dangerous. One way to correct this issue for linear splines is to require they *saturate*: remain constant outside their boundary. This problem can be solved using a heuristic that is an extension of lasso regression, producing a weighted sum of hinge functions, which we call a *saturating hinge*.

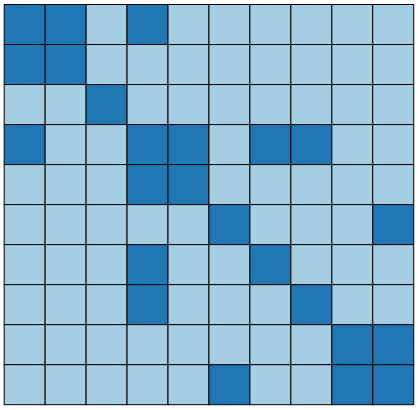
For simplicity, consider the univariate case with $n = 1$. Assume we are given knots $t_1 < t_2 < \cdots < t_k$ where each $t_j \in \mathbf{R}$. Let h_j be a hinge function at knot t_j,



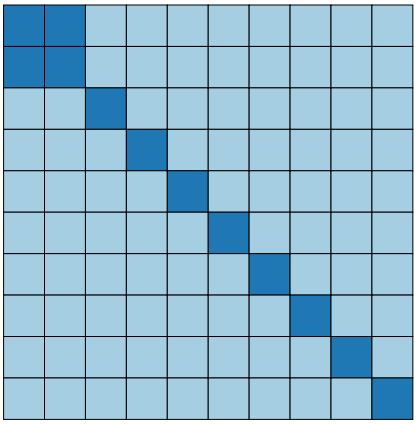
(a) True inverse.



(b) $\alpha = 10$.



(c) $\alpha = 4$.

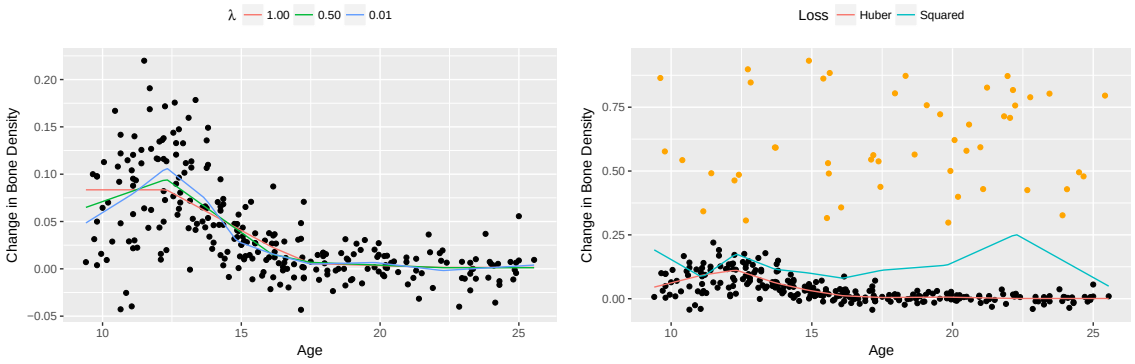


(d) $\alpha = 1$.

Figure 4.2: Sparsity patterns for (a) inverse of true covariance matrix, and estimated inverse covariance matrices with (b) $\alpha = 10$, (c) $\alpha = 4$, and (d) $\alpha = 1$. The light blue regions indicate where $S_{ij} = 0$.

i.e., $h_j(x) = \max(x - t_j, 0)$, and define $f(x) = w_0 + \sum_{j=1}^{k} w_j h_j(x)$. We want to solve

$$\begin{array}{ll} \underset{w_0,w}{\text{minimize}} & \sum_{i=1}^{m} \ell(y_i, f(x_i)) + \lambda \|w\|_1 \\ \text{subject to} & \sum_{j=1}^{k} w_j = 0 \end{array}$$



(a) Squared error loss. (b) Saturating hinges with outliers.

Figure 4.3: (a) Saturating hinges fit to the change in bone density for female patients with $\lambda = 0.01$ (blue), $\lambda = 0.5$ (green), and $\lambda = 1$ (red). (b) Hinges refit to the previous data with additional outliers (orange) using squared error (blue) and Huber loss (red).

for variables $(w_0, w) \in \mathbf{R} \times \mathbf{R}^k$. The function $\ell : \mathbf{R} \times \mathbf{R} \to \mathbf{R}$ is the loss associated with every observation, and $\lambda \geq 0$ is the penalty weight. In choosing our knots, we set $t_1 = \min(x_i)$ and $t_k = \max(x_i)$ so that by construction, the estimate $\hat{f}$ will be constant outside $[t_1, t_k]$.

We demonstrate this technique on the bone density data for female patients from Hastie et al. (2001, Section 5.4). There are a total of $m = 259$ observations. Our response y_i is the change in spinal bone density between two visits, and our predictor x_i is the patient's age. We select $k = 10$ knots about evenly spaced across the range of X and fit a saturating hinge with squared error loss $\ell(y_i, f(x_i)) = (y_i - f(x_i))^2$.

In R, we first define the estimation and loss functions:

```
R> f_est <- function(x, knots, w0, w) {
+    hinges <- sapply(knots, function(t) pmax(x - t, 0))
+    w0 + hinges %*% w
+  }
R> loss_obs <- function(y, f) (y - f)^2
```

This allows us to easily test different losses and knot locations later. The rest of the set-up is similar to previous examples. We assume that `knots` is a R vector

representing $(t_1, \ldots, t_k)$.

```
R> w0 <- Variable(1)
R> w <- Variable(k)
R> loss <- sum(loss_obs(y, f_est(X, knots, w0, w)))
R> reg <- lambda * p_norm(w, 1)
R> obj <- loss + reg
R> constr <- list(sum(w) == 0)
R> prob <- Problem(Minimize(obj), constr)
R> result <- solve(prob)
```

The optimal weights are retrieved using separate calls, as shown below.

```
R> w0s <- result$getValue(w0)
R> ws <- result$getValue(w)
```

We plot the fitted saturating hinges in Figure 4.3a. As expected, when λ increases, the spline exhibits less variation and grows flatter outside its boundaries. The squared error loss works well in this case, but as we saw previously in this section, the Huber loss is preferred when the dataset contains large outliers. We can change the loss function by simply redefining

```
R> loss_obs <- function(y, f, M) huber(y - f, M)
```

and passing an extra threshold parameter in when initializing `loss`. In Figure 4.3b, we have added 50 randomly generated outliers to the bone density data and plotted the re-fitted saturating hinges. For a Huber loss with $M = 0.01$, the resulting spline is fairly smooth and follows the shape of the original data, as opposed to the spline using squared error loss, which is biased upwards by a significant amount.

4.3.2 Nonparametric estimation

Log-concave distribution estimation

Let $n = 1$ and suppose x_i are i.i.d. samples from a log-concave discrete distribution on $\{0, \ldots, K\}$ for some $K \in \mathbf{Z}_+$. Define $p_k := \mathbf{Prob}(X = k)$ to be the probability mass

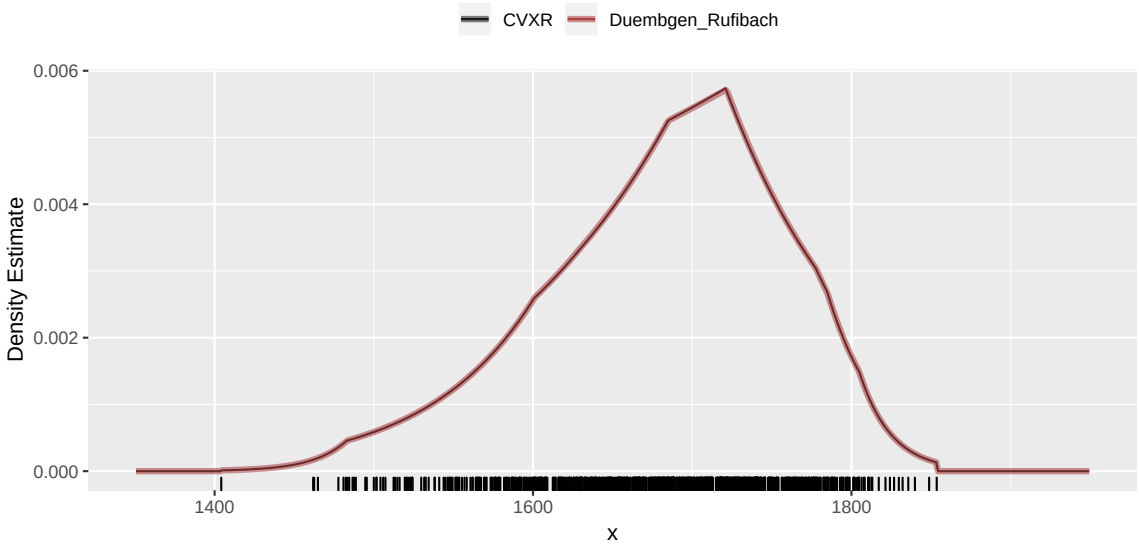


Figure 4.4: Log-concave estimation using the approach of Dümbgen and Rufibach (2011) and CVXR.

function. One method for estimating $(p_0, \ldots, p_K)$ is to maximize the log-likelihood function subject to a log-concavity constraint (Dümbgen and Rufibach, 2009), i.e.,

$$\begin{array}{ll} \underset{p}{\text{maximize}} & \sum_{k=0}^{K} M_k \log p_k \\ \text{subject to} & p \geq 0, \quad \sum_{k=0}^{K} p_k = 1, \\ & p_k \geq \sqrt{p_{k-1} p_{k+1}}, \quad k = 1, \ldots, K-1, \end{array}$$

where $p \in \mathbf{R}^{K+1}$ is our variable of interest and M_k represents the number of observations equal to k, so that $\sum_{k=0}^{K} M_k = m$. The problem as posed above is not convex. However, we can transform it into a convex optimization problem by defining new variables $u_k = \log p_k$ and relaxing the equality constraint to $\sum_{k=0}^{K} p_k \leq 1$, since the latter always holds tightly at an optimal solution. The result is

$$\begin{array}{ll} \underset{u}{\text{maximize}} & \sum_{k=0}^{K} M_k u_k \\ \text{subject to} & \sum_{k=0}^{K} e^{u_k} \leq 1, \\ & u_k - u_{k-1} \geq u_{k+1} - u_k, \quad k = 1, \ldots, K-1. \end{array}$$

If `counts` is the R vector of $(M_0, \ldots, M_K)$, the code for our convex problem is

```
R> u <- Variable(K+1)
R> obj <- t(counts) %*% u
R> constr <- list(sum(exp(u)) <= 1, diff(u[1:K]) >= diff(u[2:(K+1)]))
R> prob <- solve(Maximize(obj), constr)
R> result <- solve(prob)
```

Once the solver is finished, we can retrieve the probabilities directly with

```
R> pmf <- result$getValue(exp(u))
```

The above line transforms the variables u_k to e^{u_k} before calculating their resulting values. This is possible because `exp` is a member of CVXR's library of atoms, so it can operate directly on a '`Variable`' object such as `u`.

As an example, we consider the reliability data from Dümbgen and Rufibach (2011) that was collected as part of a consulting project at the Institute for Mathematical Statistics and Actuarial Science, University of Bern (Dümbgen and Rufibach, 2009). The dataset consists of $n = 786$ observations, and the goal is to fit a suitable distribution to this sample that can be used for simulations. For various reasons detailed in the paper, the authors chose a log-concave estimator, which they implemented in the R package logcondens (Dümbgen and Rufibach, 2011). Figure 4.4 shows that the curve obtained from the CVXR code above matches their results exactly.

Survey calibration

Calibration is a widely used technique in survey sampling. Suppose m sampling units in a survey have been assigned initial weights d_i for $i = 1, \ldots, m$, and furthermore, there are n auxiliary variables whose values in the sample are known. Calibration seeks to improve the initial weights d_i by finding new weights w_i that incorporate this auxiliary information while perturbing the initial weights as little as possible, i.e., the ratio $g_i = w_i/d_i$ must be close to one. Such reweighting improves precision of estimates (Lumley, 2010, Chapter 7).

		survey		CVXR	
School Type	Target Met?	Weight	Frequency	Weight	Frequency
E	Yes	29.00	15	29.00	15
H	No	31.40	13	31.40	13
M	Yes	29.03	9	29.03	9
E	No	28.91	127	28.91	127
H	Yes	31.50	12	31.50	12
M	No	31.53	24	31.53	24

Table 4.1: Raking weight estimates with survey package and CVXR for California Academic Performance Index data.

Let $X \in \mathbf{R}^{m \times n}$ be the matrix of survey samples, with each column corresponding to an auxiliary variable. Reweighting can be expressed as the optimization problem

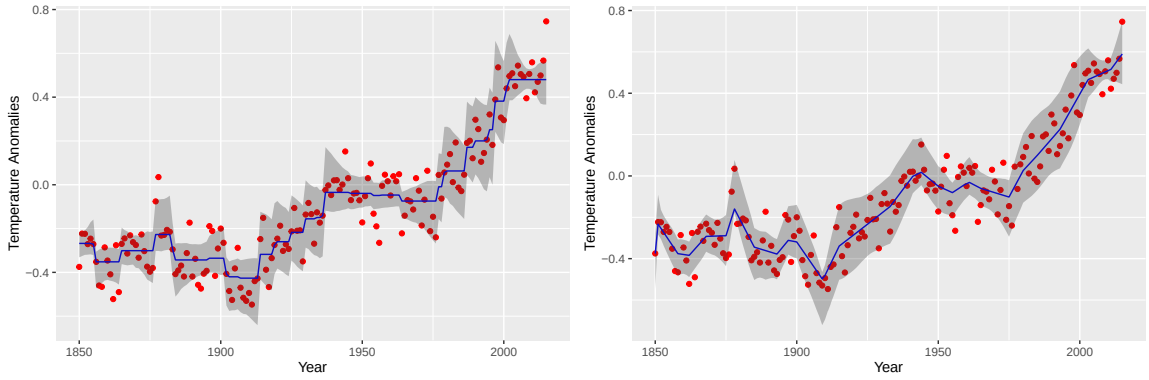
$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{m} d_i \phi(g_i) \\ \text{subject to} & A^\top g = r \end{array}$$

with respect to $g \in \mathbf{R}^m$, where $\phi : \mathbf{R} \to \mathbf{R}$ is a strictly convex function with $\phi(1) = 0$, $r \in \mathbf{R}^n$ are the known population totals of the auxiliary variables, and $A \in \mathbf{R}^{m \times n}$ is related to X by $A_{ij} = d_i X_{ij}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. A common technique is raking, which uses the penalty function $\phi(g_i) = g_i \log(g_i) - g_i + 1$.

We illustrate with the California Academic Performance Index data in the survey package (Lumley, 2004, 2020), which also supplies facilities for calibration via the function `calibrate`. Both the population dataset (`apipop`) and a simple random sample of $m = 200$ (`apisrs`) are provided. Suppose that we wish to reweight the observations in the sample using known totals for two variables from the population: `stype`, the school type (elementary, middle or high) and `sch.wide`, whether the school met the yearly target or not. This reweighting would make the sample more representative of the general population.

The code below solves the problem in CVXR, where we have used a model matrix to generate the appropriate dummy variables for the two factor variables.

```
R> m <- nrow(apisrs)
R> di <- apisrs$pw
```



(a) Nearly-isotonic. (b) Nearly-convex.

Figure 4.5: (a) A nearly-isotonic fit and (b) nearly-convex fit to global warming data on temperature anomalies for $\lambda = 0.44$. The 95% normal confidence intervals are shown in gray using $R = 400$ and $R = 200$ bootstrap samples, respectively.

```
R> formula <- ~ stype + sch.wide
R> r <- apply(model.matrix(object = formula, data = apipop), 2, sum)
R> X <- model.matrix(object = formula, data = apisrs)
R> A <- di * X
R> g <- Variable(m)
R> obj <- sum(di * (-entr(g) - g + 1))
R> constr <- list(t(A) %*% g == r)
R> prob <- Problem(Minimize(obj), constr)
R> result <- solve(prob)
R> w_cvxr <- di * result$getValue(g)
```

Table 4.1 shows that the results are identical to those obtained from survey. CVXR can also accommodate other penalty functions common in the survey literature, as well as additional constraints.

Nearly-isotonic and nearly-convex fits

Given a set of data points $y \in \mathbf{R}^m$, Tibshirani et al. (2011) fit a nearly-isotonic approximation $\beta \in \mathbf{R}^m$ by solving

$$\underset{\beta}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{m}(y_i - \beta_i)^2 + \lambda \sum_{i=1}^{m-1}(\beta_i - \beta_{i+1})_+,$$

where $\lambda \geq 0$ is a penalty parameter and $x_+ = \max(x, 0)$. Our CVXR formulation follows directly as shown below. The `pos` atom evaluates x_+ elementwise on the input expression.

```
R> near_fit <- function(y, lambda) {
+    m <- length(y)
+    beta <- Variable(m)
+    penalty <- sum(pos(diff(beta)))
+    obj <- 0.5 * sum((y - beta)^2) + lambda * penalty
+    prob <- Problem(Minimize(obj))
+    result <- solve(prob)
+    result$getValue(beta)
+  }
```

We demonstrate this technique on the global warming data provided by the Carbon Dioxide Information Analysis Center (CDIAC). Our data points are the annual temperature anomalies relative to the 1961–1990 mean. Combining `near_fit` with the boot package (Canty and Ripley, 2020), we can obtain the standard errors and confidence intervals for our estimate in just a few lines of code.

```
R> near_fit_stat <- function(data, index, lambda) {
+    sample <- data[index, ]                  # Bootstrap sample of rows
+    sample <- sample[order(sample$year), ]   # Order ascending by year
+    near_fit(sample$annual, lambda)
+  }
R> boot.out <- boot(CDIAC, near_fit_stat, R = 400, lambda = 0.44)
```

Figure 4.5a shows a nearly-isotonic fit with $\lambda = 0.44$ and 95% normal confidence bands, which were generated using $R = 400$ bootstrap samples. The curve follows the data well, but exhibits choppiness in regions with a steep trend.

For a smoother curve, we can solve for the nearly-convex fit described in the same paper:

$$\underset{\beta}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{m}(y_i - \beta_i)^2 + \lambda \sum_{i=1}^{m-2}(\beta_i - 2\beta_{i+1} + \beta_{i+2})_+$$

This replaces the first difference term with an approximation to the second derivative at β_{i+1}. In CVXR, the only change necessary is the penalty line in `near_fit`,

```
R> penalty <- sum(pos(diff(beta, differences = 2)))
```

The resulting curve is depicted in Figure 4.5b with 95% confidence bands generated from $R = 200$ samples. Note the jagged staircase pattern has been smoothed out. We can easily extend this example to higher-order differences or lags by modifying the arguments to `diff`.

4.3.3 Miscellaneous applications

Worst case covariance

Suppose we have i.i.d. samples $x_i \sim N(0, \Sigma)$ for $i = 1, \ldots, m$ and want to determine the maximum covariance of $y = w^\top x = \sum_{i=1}^{m} w_i x_i$, where $w \in \mathbf{R}^m$ is a given vector of weights. We are provided limited information on the elements of Σ. For example, we may know the specific value or sign of certain Σ_{jk}, which are represented by upper and lower bound matrices L and $U \in \mathbf{R}^{n \times n}$, respectively (Boyd and Vandenberghe, 2004, pp. 171–172). This situation can arise when calculating the worst-case risk of an investment portfolio (Lobo and Boyd, 2000). Formally, our optimization problem is

$$\begin{aligned} \underset{\Sigma}{\text{maximize}} \quad & w^\top \Sigma w \\ \text{subject to} \quad & \Sigma \in \mathbf{S}_+^n, \quad L_{jk} \leq \Sigma_{jk} \leq U_{jk}, \quad j, k = 1, \ldots, n. \end{aligned}$$

Consider the specific case

$$w = \begin{bmatrix} 0.1 \\ 0.2 \\ -0.05 \\ 0.1 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 0.2 & + & + & \pm \\ + & 0.1 & - & - \\ + & - & 0.3 & + \\ \pm & - & + & 0.1 \end{bmatrix},$$

where a $+$ means the element is non-negative, a $-$ means the element is non-positive, and a $\pm$ means the element can be any real number. In CVXR, this semidefinite program is

```
R> Sigma <- Variable(n, n, PSD = TRUE)
R> obj <- t(w) %*% Sigma %*% w
R> constr <- list(Sigma[1, 1] == 0.2, Sigma[1, 2] >= 0, Sigma[1, 3] >= 0,
+    Sigma[2, 2] == 0.1, Sigma[2, 3] <= 0, Sigma[2, 4] <= 0,
+    Sigma[3, 3] == 0.3, Sigma[3, 4] >= 0, Sigma[4, 4] == 0.1)
R> prob <- Problem(Maximize(obj), constr)
R> result <- solve(prob, solver = "SCS")
```

Our result for this numerical case is

$$\Sigma = \begin{bmatrix} 0.2000 & 0.0967 & 0.0000 & 0.0762 \\ 0.0967 & 0.1000 & -0.1032 & 0.0000 \\ 0.0000 & -0.1032 & 0.3000 & 0.0041 \\ 0.0762 & 0.0000 & 0.0041 & 0.1000 \end{bmatrix}$$

This example can be generalized to include arbitrary convex constraints on Σ. Furthermore, if we have a target estimate for the covariance, we can bound deviations from the target by incorporating penalized slack variables into our optimization problem.

Catenary problem

We consider a discretized version of the catenary problem in Griva and Vanderbei (2005). A chain with uniformly distributed mass hangs from the endpoints $(0, 1)$ and

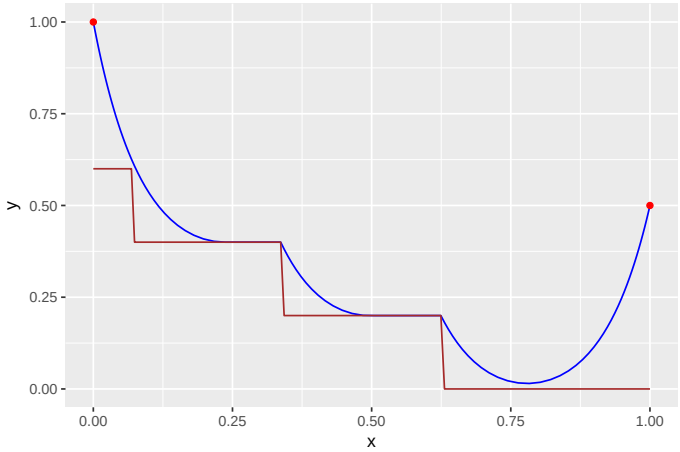


Figure 4.6: Solution of the catenary problem (blue) with a ground constraint (brown).

$(1, 1)$ on a 2-D plane. Gravitational force acts in the negative y direction. Our goal is to find the shape of the chain in equilibrium, which is equivalent to determining the (x, y) coordinates of every point along its curve when its potential energy is minimized.

To formulate this as an optimization problem, we parameterize the chain by its arclength and divide it into m discrete links. The length of each link must be no more than $h > 0$. Since mass is uniform, the total potential energy is simply the sum of the y-coordinates. Therefore, our problem is

$$\begin{array}{ll} \underset{x,y}{\text{minimize}} & \sum_{i=1}^{m} y_i \\ \text{subject to} & x_1 = 0, \quad y_1 = 1, \quad x_m = 1, \quad y_m = 1 \\ & (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 \leq h^2, \quad i = 1, \ldots, m-1 \end{array}$$

with variables $x \in \mathbf{R}^m$ and $y \in \mathbf{R}^m$. This basic catenary problem has a well-known analytical solution (Gelfand and Fomin, 1963), which we can easily verify with CVXR.

```
R> x <- Variable(m)
R> y <- Variable(m)
R> obj <- sum(y)
R> constr <- list(x[1] == 0, y[1] == 0, x[m] == 1, y[m] == 1,
+    diff(x)^2 + diff(y)^2 <= h^2)
```

```
R> prob <- Problem(Minimize(obj), constr)
R> result <- solve(prob)
```

A more interesting situation arises when the ground is not flat. Let $g \in \mathbf{R}^m$ be the elevation vector (relative to the x-axis), and suppose the right endpoint of our chain has been lowered by $\Delta y_m = 0.5$. The analytical solution in this case would be difficult to calculate. However, we need only add two lines to our constraint definition,

```
R> constr[[4]] <- (y[m] == 0.5)
R> constr <- c(constr, y >= g)
```

to obtain the new result. Figure 4.6 depicts the solution of this modified catenary problem for $m = 101$ and $h = 0.02$. The chain is shown hanging in blue, bounded below by the red staircase structure, which represents the ground.

Portfolio optimization

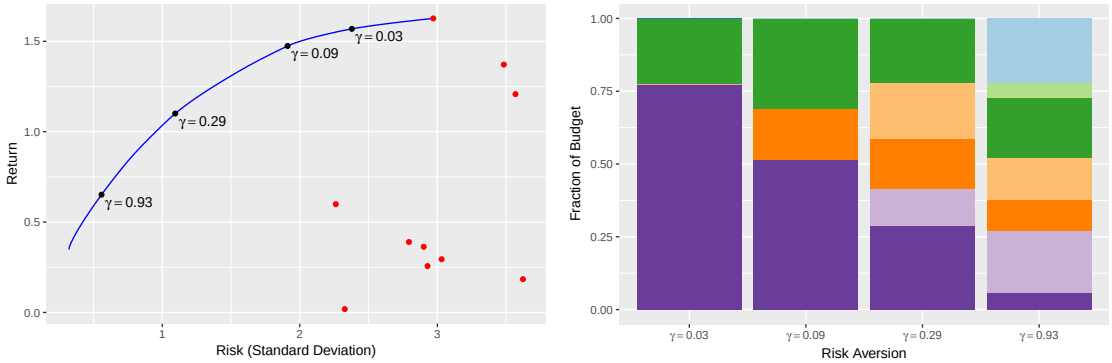
In this example, we solve the Markowitz portfolio problem under various different constraints (Markowitz, 1952; Roy, 1952; Lobo et al., 2007). We have n assets or stocks in our portfolio and must determine the amount of money to invest in each. Let w_i denote the fraction of our budget invested in asset $i = 1, \ldots, m$, and let r_i be the returns (i.e., fractional change in price) over the period of interest. We model returns as a random vector $r \in \mathbf{R}^n$ with known mean $\mathrm{E}[r] = \mu$ and covariance $\mathrm{Var}(r) = \Sigma$. Thus, given a portfolio $w \in \mathbf{R}^n$, the overall return is $R = r^\top w$.

Portfolio optimization involves a trade-off between the expected return $\mathrm{E}[R] = \mu^\top w$ and associated risk, which we take as the return variance $\mathrm{Var}(R) = w^\top \Sigma w$. Initially, we consider only long portfolios, so our problem is

$$\begin{array}{ll} \underset{w}{\text{maximize}} & \mu^\top w - \gamma w^\top \Sigma w \\ \text{subject to} & w \geq 0, \quad \sum_{i=1}^n w_i = 1, \end{array}$$

where the objective is the risk-adjusted return and $\gamma > 0$ is a risk aversion parameter.

```
R> w <- Variable(n)
```



(a) Risk-return curve. (b) Asset portfolios.

Figure 4.7: (a) Risk-return trade-off curve for various γ. Portfolios that invest completely in one asset are plotted in red. (b) Fraction of budget invested in each asset.

```
R> ret <- t(mu) %*% w
R> risk <- quad_form(w, Sigma)
R> obj <- ret - gamma * risk
R> constr <- list(w >= 0, sum(w) == 1)
R> prob <- Problem(Maximize(obj), constr)
R> result <- solve(prob)
```

In this case, it is necessary to specify the quadratic form with `quad_form` rather than the usual `t(w) %*% Sigma %*% w` because the latter will be interpreted by the CVXR parser as a product of two affine terms and rejected for not being DCP. We can obtain the risk and return by directly evaluating the value of the separate expressions:

```
R> result$getValue(risk)
R> result$getValue(ret)
```

Figure 4.7a depicts the risk-return trade-off curve for $n = 10$ assets and μ and $\Sigma^{1/2}$ drawn from a standard normal distribution. The x-axis represents the standard deviation of the return. Red points indicate the result from investing the entire budget in a single asset. As γ increases, our portfolio becomes more diverse (Figure 4.7b), reducing risk but also yielding a lower return.

Many variations on the classical portfolio problem exist. For instance, we could allow long and short positions, but impose a leverage limit $\|w\|_1 \leq L^{\max}$ by changing

```
R> constr <- list(p_norm(w, 1) <= Lmax, sum(w) == 1)
```

An alternative is to set a lower bound on the return and minimize just the risk. To account for transaction costs, we could add a term to the objective that penalizes deviations of w from the previous portfolio. These extensions and more are described in Boyd et al. (2017). The key takeaway is that all of these convex problems can be easily solved in CVXR with just a few alterations to the code above.

Kelly gambling

In Kelly gambling (Kelly, 1956), we are given the opportunity to bet on n possible outcomes, which yield a random non-negative return of $r \in \mathbf{R}^n_+$. The return r takes on exactly K values $r_1, \ldots, r_K$ with known probabilities $\pi_1, \ldots, \pi_K$. This gamble is repeated over T periods. In a given period t, let $b_i \geq 0$ denote the fraction of our wealth bet on outcome i. Assuming the nth outcome is equivalent to not wagering (it returns one with certainty), the fractions must satisfy $\sum_{i=1}^n b_i = 1$. Thus, at the end of the period, our cumulative wealth is $w_t = (r^\top b) w_{t-1}$. Our goal is to maximize the average growth rate with respect to $b \in \mathbf{R}^n$:

$$\begin{array}{ll} \underset{b}{\text{maximize}} & \sum_{j=1}^K \pi_j \log(r_j^\top b) \\ \text{subject to} & b \geq 0, \quad \sum_{i=1}^n b_i = 1. \end{array}$$

In the following code, `rets` is the K by n matrix of possible returns with rows r_j, while `ps` is the vector of return probabilities $(\pi_1, \ldots, \pi_K)$.

```
R> b <- Variable(n)
R> obj <- t(ps) %*% log(rets %*% b)
R> constr <- list(b >= 0, sum(b) == 1)
R> prob <- Problem(Maximize(obj), constr)
R> result <- solve(prob)
```

We solve the Kelly gambling problem for $K = 100$ and $n = 20$. The probabilities $\pi_j \sim \text{Uniform}(0, 1)$, and the potential returns $r_{ji} \sim \text{Uniform}(0.5, 1.5)$ except for $r_{jn} = 1$, which represents the payoff from not wagering. With an initial wealth of

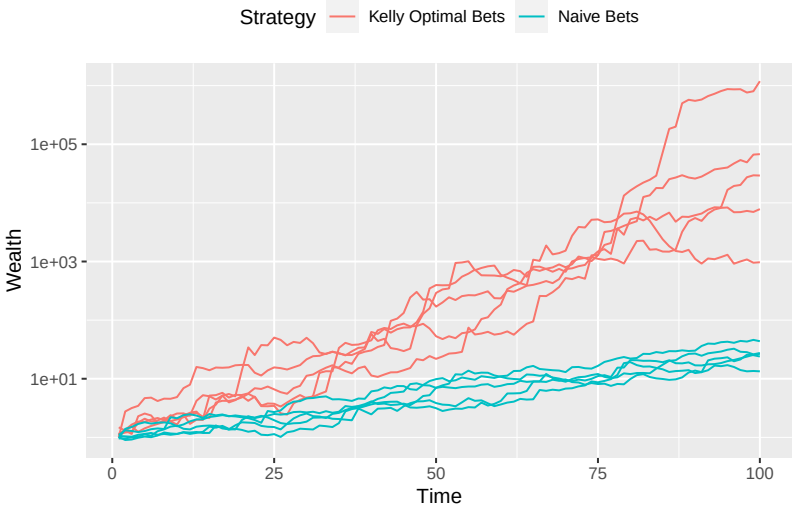


Figure 4.8: Wealth trajectories for the Kelly optimal bets (red) and naïve bets (cyan). The naïve betting scheme holds onto 15% of the wealth and splits the rest in direct proportion to the expected returns.

$w_0 = 1$, we simulate the growth trajectory of our Kelly optimal bets over $P = 100$ periods, assuming returns are i.i.d. over time.

```
R> bets <- result$getValue(b)
R> idx <- sample.int(K, size = P, probs = ps, replace = TRUE)
R> winnings <- rets[idx,] %*% bets
R> wealth <- w0 * cumprod(winnings)
```

For comparison, we also calculate the trajectory for a naïve betting scheme, which holds onto 15% of the wealth at the beginning of each period and divides the other 85% over the bets in direct proportion to their expect returns.

Growth curves for five independent trials are plotted in Figure 4.8. Red lines represent the wealth each period from the Kelly bets, while cyan lines are the result of the naïve bets. Clearly, Kelly optimal bets perform better, producing greater net wealth by the final period. However, as observed in some trajectories, wealth tends to drop by a significant amount before increasing eventually. One way to reduce this drawdown risk is to add a convex constraint as proposed in Busseti et al. (2016,

Section 5.3),

$$\log\left(\sum_{j=1}^{K}\exp(\log\pi_j - \lambda\log(r_j^\top b))\right) \leq 0,$$

where $\lambda \geq 0$ is the risk-aversion parameter. With CVXR, this can be accomplished in a single line using the `log_sum_exp` atom. Other extensions like wealth goals, betting restrictions, and VaR/CVaR bounds are also readily incorporated.

Channel capacity

The following problem comes from an exercise in Boyd and Vandenberghe (2004, pp. 207–208). Consider a discrete memoryless communication channel with input $X(t) \in \{1,\ldots,n\}$ and output $Y(t) \in \{1,\ldots,m\}$ for $t = 1, 2, \ldots$. The relation between the input and output is given by a transition matrix $P \in \mathbf{R}_+^{m\times n}$ with

$$P_{ij} = \mathbf{Prob}(Y(t) = i | X(t) = j), \quad i = 1,\ldots,m, \quad j = 1,\ldots,n.$$

Assume that X has a probability distribution denoted by $x \in \mathbf{R}^n$, i.e., $x_j = \mathbf{Prob}(X(t) = j)$ for $j = 1,\ldots,n$. A famous result by Shannon and Weaver (1949) states that the channel capacity is found by maximizing the mutual information between X and Y,

$$I(X,Y) = \sum_{j=1}^{n} x_j \sum_{i=1}^{m} P_{ij}\log_2 P_{ij} - \sum_{i=1}^{m} y_i \log_2 y_i,$$

where $y = Px$ is the probability distribution of Y. Since I is concave, this is equivalent to solving the convex optimization problem

$$\begin{array}{ll}\underset{x,y}{\text{maximize}} & \sum_{j=1}^{n} x_j \sum_{i=1}^{m} P_{ij}\log P_{ij} - \sum_{i=1}^{m} y_i \log y_i \\ \text{subject to} & x \geq 0, \quad \sum_{i=1}^{m} x_i = 1, \quad y = Px\end{array}$$

for $x \in \mathbf{R}^n$ and $y \in \mathbf{R}^m$. The associated code in CVXR is

```
R> x <- Variable(n)
R> y <- P %*% x
R> c <- apply(P * log2(P), 2, sum)
```

```
R> obj <- t(c) %*% x + sum(entr(y))
R> constr <- list(sum(x) == 1, x >= 0)
R> prob <- Problem(Maximize(obj), constr)
R> result <- solve(prob)
```

The channel capacity is simply the optimal objective, `result$value`.

Fastest mixing Markov chain

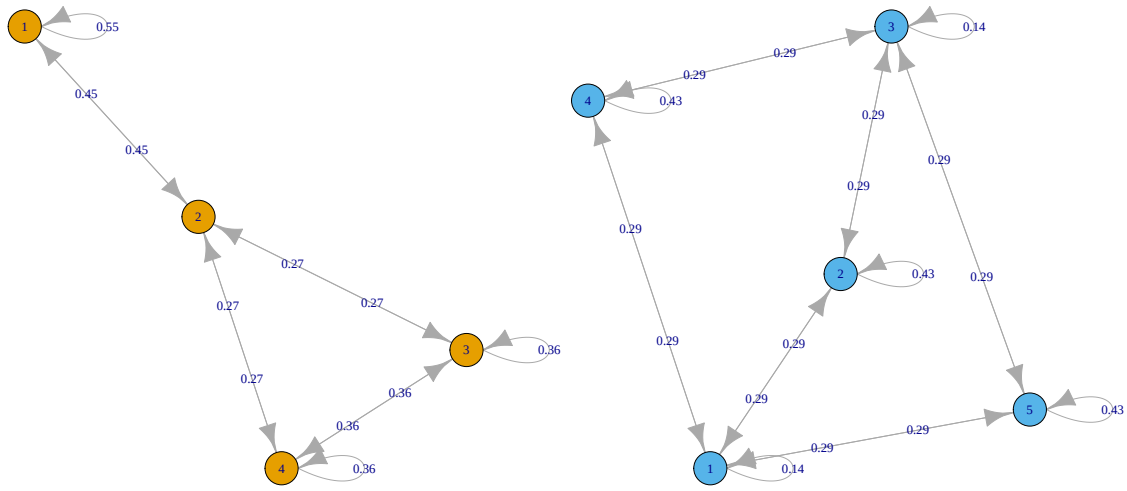
This example is derived from the results in Boyd et al. (2004, Section 2). Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected graph with vertices $\mathcal{V} = \{1, \ldots, n\}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Assume that $(i, i) \in \mathcal{E}$ for all $i = 1, \ldots, n$, and $(i, j) \in \mathcal{E}$ implies $(j, i) \in \mathcal{E}$. Under these conditions, a discrete-time Markov chain on $\mathcal{V}$ will have the uniform distribution as one of its equilibrium distributions. We are interested in finding the Markov chain, i.e., constructing the transition probability matrix $P \in \mathbf{R}_+^{n \times n}$, that minimizes its asymptotic convergence rate to the uniform distribution. This is an important problem in Markov chain Monte Carlo (MCMC) simulations, as it directly affects the sampling efficiency of an algorithm.

The asymptotic rate of convergence is determined by the second largest eigenvalue of P, which in our case is $\mu(P) := \sigma_{\max}(P - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)$ where $\sigma_{\max}(A)$ denotes the maximum singular value of A. As $\mu(P)$ decreases, the mixing rate increases and the Markov chain converges faster to equilibrium. Thus, our optimization problem is

$$\begin{array}{ll} \underset{P}{\text{minimize}} & \sigma_{\max}(P - \frac{1}{n}\mathbf{1}\mathbf{1}^\top) \\ \text{subject to} & P \geq 0, \quad P\mathbf{1} = \mathbf{1}, \quad P = P^\top \\ & P_{ij} = 0, \quad (i, j) \notin \mathcal{E}. \end{array}$$

The element P_{ij} of our transition matrix is the probability of moving from state i to state j. Our assumptions imply that P is non-negative, symmetric, and doubly stochastic. The last constraint ensures transitions do not occur between unconnected vertices.

The function $\sigma_{\max}$ is convex, so this problem is solvable in CVXR. For instance, the code for the Markov chain in Figure 4.9a is



(a) Triangle + 1 edge. (b) Bipartite 2 + 3.

Figure 4.9: Markov chains with transition probabilities that achieve the fastest mixing rate.

```
R> P <- Variable(n, n)
R> ones <- matrix(1, nrow = n, ncol = 1)
R> obj <- sigma_max(P - 1/n)
R> constr1 <- list(P >= 0, P %*% ones == ones, P == t(P))
R> constr2 <- list(P[1, 3] == 0, P[1, 4] == 0)
R> prob <- Problem(Minimize(obj), c(constr1, constr2))
R> result <- solve(prob, solver = "SCS")
```

where we have set $n = 4$. We could also have specified $P\mathbf{1} = \mathbf{1}$ with `sum_entries(P, 1) == 1`, which uses the `sum_entries` atom to represent the row sums.

It is easy to extend this example to other Markov chains. To change the number of vertices, we would simply modify `n`, and to add or remove edges, we need only alter the constraints in `constr2`. For instance, the bipartite chain in Figure 4.9b is produced by setting $n = 5$ and

```
R> constr2 <- list(P[1, 3] == 0, P[2, 4] == 0, P[2, 5] == 0, P[4, 5] == 0)
```

Radiation therapy dose scheduling

This example is a simple variation on the adaptive radiation treatment planning problem in Chapter 3. An oncology patient is given a dose of radiation $d_t \in \mathbf{R}_+$ in sessions $t = 1, \ldots, T-1$ with the goal of shrinking a tumor to some specified target size, while minimizing the damage to the patient's health. We must choose the doses d_t subject to the constraint $d_t \leq d^{\max}$, where $d^{\max}$ is a constant.

Let $S_t \in \mathbf{R}_+$ denote the tumor size in session t. This evolves in response to the radiation dose as

$$S_{t+1} = \alpha e^{-\beta d_t} S_t, \quad t = 1, \ldots, T-1,$$

where $\alpha > 1$ is the per-session tumor growth rate without radiation and $\beta > 0$ is a known constant. The initial tumor size $S_1 > 0$ is given; by the end of treatment, we wish to achieve $S_T \leq S^{\text{tar}}$, where S^{tar} is the target final tumor size.

Similarly, let $H_t \in \mathbf{R}_+$ denote some measure of the damage to the patient's health from radiation treatment. This evolves according to

$$H_{t+1} = \gamma e^{\delta d_t} H_t, \quad t = 1, \ldots, T-1,$$

where $\gamma \in (0, 1]$ is the per-session damage recovery rate without radiation and $\delta > 0$ is a known constant. We assume the initial damage $H_1 > 0$ is given.

Our goal is to find a sequence of doses that satisfies the constraints described above and minimizes the maximum damage across all sessions. By taking logs and changing variables to $\tilde{S}_t = \log S_t$ and $\tilde{H}_t = \log H_t$, we can pose the dose scheduling problem as

$$\begin{array}{ll} \text{minimize} & \max_{t=1,\ldots,T} \tilde{H}_t \\ \text{subject to} & \tilde{S}_{t+1} = \tilde{S}_t - \beta d_t + \log \alpha, \quad t = 1, \ldots, T-1 \\ & \tilde{H}_{t+1} = \tilde{H}_t + \delta d_t + \log \gamma, \quad t = 1, \ldots, T-1 \\ & \tilde{S}_1 = \log S_1, \quad \tilde{S}_T \leq \log S^{\text{tar}} \\ & \tilde{H}_1 = \log H_1, \quad 0 \preceq d \preceq d^{\max} \end{array}$$

with respect to $d = (d_1, \ldots, d_{T-1}), \tilde{S} = (\tilde{S}_1, \ldots, \tilde{S}_T)$, and $\tilde{H} = (\tilde{H}_1, \ldots, \tilde{H}_T)$. This

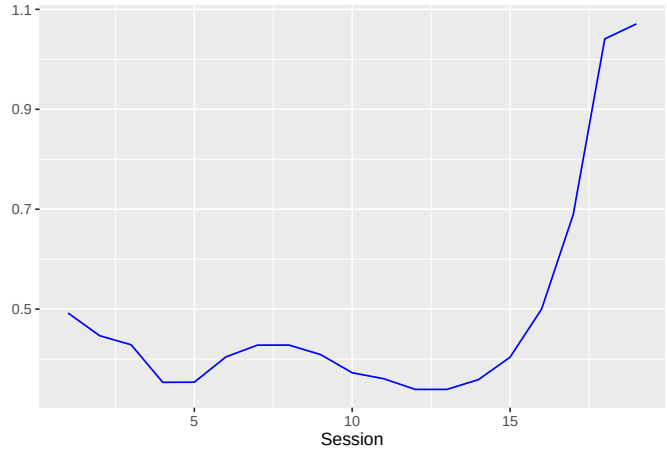
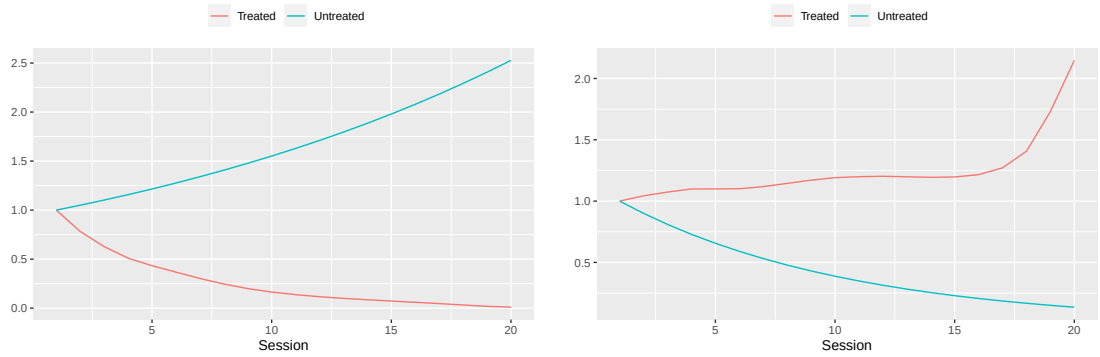


Figure 4.10: Optimal radiation dose plan.

problem is convex and hence can be solved using CVXR, as shown below.

```
R> d <- Variable(T_val-1, nonneg = TRUE)
R> S_l <- Variable(T_val)
R> H_l <- Variable(T_val)
R> obj <- max(H_l)
R> constr <- list()
R> for(t in seq_len(T_val-1)) {
+    constr <- c(constr, list(S_l[t+1] == S_l[t] - beta*d[t] + log(alpha),
+      H_l[t+1] == H_l[t] + delta*d[t] + log(gamma)))
+ }
R> constr <- c(constr, list(S_l[1] == log(S1),
+    S_l[T_val] <= log(S_tar), H_l[1] == log(H1), d <= d_max))
R> prob <- Problem(Minimize(obj), constr)
R> result <- solve(prob)
```

Figures 4.10 and 4.11 depict the solution of an instance with $d^{\max} = 1.2, \alpha = 1.05, \beta = 0.6, \gamma = 0.9, \delta = 0.3, S_1 = 1, S^{\text{tar}} = 0.01$, and $H_1 = 1$.



(a) Tumor size. (b) Health damage.

Figure 4.11: (a) Tumor size and (b) patient health damage under the optimal dose plan (red) and without any radiation treatment (blue).

4.4 Implementation

CVXR represents the atoms, variables, constraints, and other parts of an optimization problem using S4 class objects. S4 enables us to overload standard mathematical operations so CVXR combines seamlessly with native R code and other packages. When an operation is invoked on a variable, a new object is created that represents the corresponding expression tree with the operator as the root node and the arguments as leaves. This tree grows automatically as more elements are added, allowing us to encapsulate the structure of an objective function or constraint.

Once the user calls `solve`, DCP verification occurs. CVXR traverses the expression tree recursively, determining the sign and curvature of each sub-expression based on the properties of its component atoms. If the problem is deemed compliant, it is transformed into an equivalent cone program using graph implementations of convex functions (Grant et al., 2006). Then, CVXR passes the problem's description to the CVXcanon C++ library (Miller et al., 2015), which generates data for the cone program, and sends this data to the solver-specific R interface. The solver's results are returned to the user in a list. This object-oriented design and infrastructure were largely borrowed from CVXPY.

CVXR interfaces with the open-source cone solvers ECOS (Domahidi et al., 2013) and SCS (O'Donoghue et al., 2016) through their respective R packages. ECOS is

an interior-point solver, which achieves high accuracy for small and medium-sized problems, while SCS is a first-order solver that is capable of handling larger problems and semidefinite constraints. As noted by Domahidi et al. (2013, Section I.A), first-order methods can be slow if the problem is not well conditioned or if it has a feasible set that does not allow for an efficient projection, while interior-point methods have a convergence rate that is independent of the problem data and the particular feasible set. Furthermore, starting from version 0.99, CVXR also provides support for the commercial solvers MOSEK (Andersen and Andersen, 2000) and GUROBI (Gurobi Optimization, Inc, 2016) through binary R packages published by the respective vendors. It is not difficult to connect additional solvers so long as the solver has an API that can communicate with R. Users who wish to employ a custom solver may obtain the canonicalized data for a problem and solver combination directly with `get_problem_data(problem, solver)`. When more than one solver is capable of solving a problem, the `solver` argument to the `solve` function can be used to indicate a preference. Available solvers, depending on installed packages in a session, are returned via `installed_solvers()`. Interested users should consult tutorial examples on the web page `https://cvxr.rbind.io` for further guidance.

We have provided a large library of atoms, which should be sufficient to model most convex optimization problems. However, it is possible for a sophisticated user to incorporate new atoms into this library. The process entails creating a S4 class for the atom, overloading methods that characterize its DCP properties, and representing its graph implementation as a list of linear operators that specify the corresponding feasibility problem. For instance, the absolute value function $f(x) = |x|$ is represented by the `Abs` class, which inherits from `Atom`. We defined its curvature by overloading the S4 method `is_atom_convex`, used in the DCP verification step, to return `TRUE` when called on an `Abs` object. Then, we derived the graph form of the absolute value to be $f(x) = \inf\{t | -t \leq x \leq t\}$. This form's objective and constraints were coded into lists in the atom's `graph_implementation` function. A full mathematical exposition may be found in Grant et al. (2006, Section 10). In general, we suggest users try to reformulate their optimization problem first before attempting to add a novel atom.

4.4.1 Speed considerations

Usually, CVXR will be slower than a direct call to a solver, because in the latter case, the user would have already done the job of translating a mathematical problem into code and constraints ingestible by the solver. CVXR does this translation for the user starting from a DCP formulation of the problem by walking the abstract syntax tree, which represents the canonicalized objectives and constraints, and building appropriate matrix structures for the solver. The matrix data are passed to a compatible solver using either Rcpp (Eddelbuettel and François, 2011) or calls to a solver-specific R package. CVXR stores data in sparse matrices, thereby allowing large problems to be specified. However, the restrictions imposed by R on sparse matrices (Bates and Maechler, 2019) still apply: each dimension cannot exceed the integer limit of $2^{31} - 1$.

Currently, the canonicalization and construction of data in R for the solver dominates computation time, particularly for complex expressions that involve indexing into individual elements of a matrix or vector. Using available CVXR functions for vectorized operations provides substantial speed improvements.

CVXR also provides a `Parameter` object that can be combined with warm starts, if such an option is available in the solver. A `Parameter` is a constant expression whose value can be modified after a `Problem` is created. This can yield significant reductions in computation time when solving a family of parametrized problems. The code below exploits warm starts to solve a lasso problem with two different values of the penalization parameter λ.

```
R> beta <- Variable(n)
R> lambda <- Parameter(pos = TRUE)
R> obj <- 0.5 * sum((y - X %*% beta)^2) + lambda * p_norm(beta, 1)
R> constr <- list(beta >= 0)
R> prob <- Problem(Minimize(obj), constr)
R> value(lambda) <- 1   # First value of lambda
R> result <- solve(prob, solver = "OSQP")
R> value(lambda) <- 2   # Second value of lambda, warm start
R> result <- solve(prob, solver = "OSQP", warm_start = TRUE)
```

On a commodity Macintosh laptop, with $X \in \mathbf{R}^{2000 \times 500}$ and $y \in \mathbf{R}^{2000}$, the first solution took 7.153 seconds, while the second took only 0.763 seconds.

4.5 Conclusion

Convex optimization plays an essential role in many fields, particularly machine learning and statistics. CVXR provides an object-oriented language with which users can easily formulate, modify, and solve a broad range of convex optimization problems. While other R packages may perform faster on a subset of these problems, CVXR's advantage is its flexibility and simple intuitive syntax, making it an ideal tool for prototyping new models for which custom R code does not exist. For more information, see the official web page of the package on the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=CVXR` and documentation.

Chapter 5

Conclusion

5.1 Summary

We have presented a unified optimization-based framework for adaptive radiation treatment planning. Starting from a simple setting, we showed how to formulate the treatment planning problem as a convex optimization problem, which is both tractable and easy to solve. We then examined two potential sources of nonconvexity: dose-volume constraints and nonlinear patient health dynamics.

In considering the first case, we limited our attention to the static setting. We replaced each dose-volume constraint with a convex restriction based on the hinge loss function. This restriction overestimates the number of voxels that violate the clinician's desired dose threshold. Therefore, to mitigate its impact on the overall objective, we developed a two-pass algorithm that solves the restricted problem in the first pass, then uses this solution to meet the dose-volume constraints exactly in the second pass. We also introduced a slack refinement that ensures the first pass is always feasible. We tested our algorithm on two clinical cases and showed that it produces excellent treatment plans, which satisfy all dose-volume constraints when possible. If this is impossible (*i.e.*, the original problem is infeasible), it generates plans that minimize the total dose violation while taking into account other clinical goals. Our algorithm is implemented in the Python package, ConRad, along with a simple interface for constructing, visualizing, and comparing treatment plans.

We next turned our attention to the dynamic setting, where radiation is delivered across multiple treatment sessions. To model changes in the patient's condition between these sessions, we introduced a patient health status variable and health dynamics function. We then formulated the adaptive radiation treatment planning problem as an optimal control problem. In general, this problem is nonconvex because the patient health dynamics are described by a highly nonlinear process. We focused on the setting where the dynamics function is concave quadratic; this aligns with the standard linear-quadratic model of cell response to radiation. We proposed a method for obtaining a good proximate solution to the nonconvex optimal control problem by solving a sequence of convex approximations using an operator splitting algorithm. Our method is fast, robust, and highly scalable, handling cases that involve tens of thousands of radiation beams with ease. We implemented our method in the Python package, AdaRad, and demonstrated its performance on a large prostate cancer case.

Finally, in the last part of the dissertation, we moved beyond radiation therapy to general convex optimization. We developed a domain-specific language for formulating and solving a large class of convex optimization problems, which include statistical models like least-squares, ridge and lasso regression, Huber regression, and support vector machines. We implemented our DSL in CVXR, an R package built upon an object-oriented framework. CVXR allows users to express optimization problems in a simple mathematical syntax. Then, using a system called disciplined convex programming, it automatically verifies each problem's convexity and transforms that problem into the standard form required by a particular solver. We illustrated CVXR's modeling framework with a variety of examples from statistics, mechanics, engineering, and radiation treatment planning.

5.2 Future work

This dissertation lays the groundwork for future research in optimization and radiation therapy. In Chapters 2 and 3, we showed one way to handle nonconvex constraints in the treatment planning problem by replacing them with a convex approximation. A natural next step would be to combine the algorithms for dose-volume constraints

and nonlinear health dynamics so that both can be handled simultaneously in the adaptive setting. This could be achieved by modifying the sequential convex optimization method as follows: in each iteration, we linearize the health dynamics function, then apply the two-pass algorithm to deal with the dose-volume constraints in the linearized problem. More generally, if we have a convex surrogate for a nonconvex planning constraint, such as a conformality requirement on the dose distribution, we can use this hybrid approach to incorporate it into our adaptive treatment planning framework.

An avenue that deserves deeper exploration is the patient health dynamics model. In Chapter 3, we took the health status to be the cell survival rate according to the linear-quadratic model, but this simple model ignores factors like resensitization and repair of sublethal damage, which can be significant in certain tumors. Additionally, we have assumed that the health status of each anatomical structure evolves independently of the others, when this is rarely the case in reality. Accounting for these and other highly nonlinear effects will require us to rethink our convexification approach. Perhaps instead of forming a convex approximation, we can train a neural network on past patient data as a black box predictor of the health status. This would allow us to combine the best aspects of model-based optimization with model-free machine learning to produce treatment plans that are both well-tailored and interpretable.

Looking beyond radiation therapy, we can extend our work on treatment planning to other therapeutic agents. In practice, cancer is usually treated with a combination of different therapies, such as surgery, radiation therapy, and chemotherapy, which act synergistically to destroy diseased tissue more efficiently than a single mode of therapy. To take advantage of this synergy, we must first develop a model that captures the interactions between different modalities. Then, we can situate it in the framework from Chapter 3 to produce a method for adaptive multi-modal treatment planning. If successful, this method could open the door to more flexible, cross-disciplinary cancer treatment schemes in the future.

Appendix A

CVXR Atoms and Operators

A.1 Expressions and functions

CVXR uses the function information in this section and the DCP tools to assign expressions a sign and curvature. In what follows, the domain $\mathbf{S}^n$ refers to the set of symmetric matrices, with $\mathbf{S}^n_+$ and $\mathbf{S}^n_-$ referring to the set of positive semidefinite and negative semidefinite matrices, respectively.

A.1.1 Operators

The infix operators `+`, `-`, `*`, `%*%`, `/` are treated as functions. Both `+` and `-` are affine functions. In CVXR, `*` and `/` are affine because `expr1 * expr2` and `expr1 %*% expr2` are allowed only when one of the expressions is constant and `expr1 / expr2` is allowed only when `expr2` is a scalar constant.

The transpose of any expression can be obtained using `t(expr)`. Transpose is an affine function. The construct `expr^p` is equivalent to the function `power(expr, p)`.

A.1.2 Indexing and slicing

All non-scalar expressions can be indexed using `expr[i, j]`. Indexing is an affine function. The syntax `expr[i]` can be used as a shorthand for `expr[i, 1]` when `expr`

Function	Meaning	Domain	Sign	Curvature	Monotonicity

`geo_mean(x)`
`geo_mean(x, p)`
$p \in \mathbf{R}^n_+$, $p \neq 0$
	$x_1^{1/n} \cdots x_n^{1/n}$
	$(x_1^{p_1} \cdots x_n^{p_n})^{\frac{1}{\mathbf{1}^\top p}}$
		$x \in \mathbf{R}^n_+$	$+$	concave	$\nearrow$

`harmonic_mean(x)`	$\frac{n}{\frac{1}{x_1}+\cdots+\frac{1}{x_n}}$	$x \in \mathbf{R}^n_+$	$+$	concave	$\nearrow$
`lambda_max(X)`	$\lambda_{\max}(X)$	$X \in \mathbf{S}^n$	$\pm$	convex	none
`lambda_min(X)`	$\lambda_{\min}(X)$	$X \in \mathbf{S}^n$		concave	none
`lambda_sum_largest(X, k)` $k = 1, \ldots, n$	sum of k largest eigenvalues of X	$X \in \mathbf{S}^n$	$\pm$	convex	none
`lambda_sum_smallest(X, k)` $k = 1, \ldots, n$	sum of k smallest eigenvalues of X	$X \in \mathbf{S}^n$	$\pm$	concave	none
`log_det(X)`	$\log(\det(X))$	$X \in \mathbf{S}^n_+$	$\pm$	concave	none
`log_sum_exp(X)`	$\log\left(\sum_{ij} e^{X_{ij}}\right)$	$X \in \mathbf{R}^{m\times n}$	$\pm$	convex	$\nearrow$
`matrix_frac(x, P)`	$x^\top P^{-1} x$	$x \in \mathbf{R}^n$, $P \in \mathbf{S}^n_{++}$	$+$	convex	none
`max_entries(X)`	$\max_{ij}\{X_{ij}\}$	$X \in \mathbf{R}^{m\times n}$	same as X	convex	$\nearrow$
`min_entries(X)`	$\min_{ij}\{X_{ij}\}$	$X \in \mathbf{R}^{m\times n}$	same as X	concave	$\nearrow$
`mixed_norm(X, p, q)`	$\left(\sum_k \left(\sum_l \lvert x_{k,l}\rvert^p\right)^{q/p}\right)^{1/q}$	$X \in \mathbf{R}^{n\times n}$	$+$	convex	none
`cvxr_norm(x)` `cvxr_norm(x, 2)`	$\sqrt{\sum_i x_i^2}$	$X \in \mathbf{R}^n$	$+$	convex	$\nearrow$ for $x_i \geq 0$, $\swarrow$ for $x_i \leq 0$
`cvxr_norm(X, \fro")`	$\sqrt{\sum_{ij} X_{ij}^2}$	$X \in \mathbf{R}^{m\times n}$	$+$	convex	$\nearrow$ for $X_{ij} \geq 0$, $\swarrow$ for $X_{ij} \leq 0$
`cvxr_norm(X, 1)`	$\sum_{ij} \lvert X_{ij}\rvert$	$X \in \mathbf{R}^{m\times n}$	$+$	convex	$\nearrow$ for $X_{ij} \geq 0$, $\swarrow$ for $X_{ij} \leq 0$
`cvxr_norm(X, \inf")`	$\max_{ij}\{\lvert X_{ij}\rvert\}$	$X \in \mathbf{R}^{m\times n}$	$+$	convex	$\nearrow$ for $X_{ij} \geq 0$, $\swarrow$ for $X_{ij} \leq 0$
`cvxr_norm(X, \nuc")`	$\mathrm{tr}\left(\left(X^\top X\right)^{1/2}\right)$	$X \in \mathbf{R}^{m\times n}$	$+$	convex	none
`cvxr_norm(X)` `cvxr_norm(X, 2)`	$\sqrt{\lambda_{\max}\left(X^\top X\right)}$	$X \in \mathbf{R}^{m\times n}$	$+$	convex	none

Table A.1: Scalar functions.

Function	Meaning	Domain	Sign	Curvature	Monotonicity
`p_norm(X, p)` $p \geq 1$ or $p = \infty$	$\|X\|_p = \left(\sum_{ij} \|X_{ij}\|^p\right)^{1/p}$	$X \in \mathbf{R}^{m\times n}$	$+$	convex	$\nearrow$ for $X_{ij} \geq 0$, $\swarrow$ for $X_{ij} \leq 0$
`p_norm(X, p)` $p < 1$, $p \neq 0$	$\|X\|_p = \left(\sum_{ij} X_{ij}^p\right)^{1/p}$	$X \in \mathbf{R}^{m\times n}_+$	$+$	concave	$\nearrow$
`quad_form(x, P)` constant $P \in \mathbf{S}^n_+$	$x^\top P x$	$x \in \mathbf{R}^n$	$+$	convex	$\nearrow$ for $x_i \geq 0$, $\swarrow$ for $x_i \leq 0$
`quad_form(x, P)` constant $P \in \mathbf{S}^n_-$	$x^\top P x$	$x \in \mathbf{R}^n$	$-$	concave	$\nearrow$ for $x_i \geq 0$, $\swarrow$ for $x_i \leq 0$
`quad_form(c, X)` constant $c \in \mathbf{R}^n$	$c^\top X c$	$X \in \mathbf{R}^{n\times n}$	depends on c, X	affine	depends on c
`quad_over_lin(X, y)`	$\left(\sum_{ij} X_{ij}^2\right)/y$	$x \in \mathbf{R}^n$, $y > 0$	$+$	convex	$\nearrow$ for $X_{ij} \geq 0$, $\swarrow$ for $X_{ij} \leq 0$, $\swarrow$ in y
`sum_entries(X)`	$\sum_{ij} X_{ij}$	$X \in \mathbf{R}^{m\times n}$	same as X	affine	$\nearrow$
`sum_largest(X, k)` $k = 1, 2, \ldots$	sum of k largest X_{ij}	$X \in \mathbf{R}^{m\times n}$	same as X	convex	$\nearrow$
`sum_smallest(X, k)` $k = 1, 2, \ldots$	sum of k smallest X_{ij}	$X \in \mathbf{R}^{m\times n}$	same as X	concave	$\nearrow$
`sum_squares(X)`	$\sum_{ij} X_{ij}^2$	$X \in \mathbf{R}^{m\times n}$	$+$	convex	$\nearrow$ for $X_{ij} \geq 0$, $\swarrow$ for $X_{ij} \leq 0$
`matrix_trace(X)`	$\mathrm{tr}\,(X)$	$X \in \mathbf{R}^{n\times n}$	same as X	affine	$\nearrow$
`tv(x)`	$\sum_i \|x_{i+1} - x_i\|$	$x \in \mathbf{R}^n$	$+$	convex	none
`tv(X)`	$\sum_{ij} \left\| \begin{bmatrix} X_{i+1,j} - X_{ij} \\ X_{i,j+1} - X_{ij} \end{bmatrix} \right\|_2$	$X \in \mathbf{R}^{m\times n}$	$+$	convex	none
`tv(X1,...,Xk)`	$\sum_{ij} \left\| \begin{bmatrix} X^{(1)}_{i+1,j} - X^{(1)}_{ij} \\ X^{(1)}_{i,j+1} - X^{(1)}_{ij} \\ \vdots \\ X^{(k)}_{i+1,j} - X^{(k)}_{ij} \\ X^{(k)}_{i,j+1} - X^{(k)}_{ij} \end{bmatrix} \right\|_2$	$X^{(i)} \in \mathbf{R}^{m\times n}$	$+$	convex	none

Table A.2: More scalar functions.

is a column vector. Similarly, `expr[i]` is shorthand for `expr[1, i]` when `expr` is a row vector.

Non-scalar expressions can also be sliced using the standard R slicing syntax. For example, `expr[i:j, r]` selects rows `i` through `j` of column `r` and returns a vector.

CVXR supports advanced indexing using lists of indices or boolean arrays. The semantics are the same as in R. Any time R might return a numeric vector, CVXR returns a column vector.

A.1.3 Scalar functions

CVXR provides the scalar functions displayed in Tables A.1 and A.2, which take in one or more scalars, vectors, or matrices as arguments and return a scalar.

For a vector expression `x`, `cvxr_norm(x)` and `cvxr_norm(x, 2)` give the Euclidean norm. For a matrix expression `X`, however, `cvxr_norm(X)` and `cvxr_norm(X, 2)` give the spectral norm. The function `cvxr_norm(X, \fro")` gives the Frobenius norm and `cvxr_norm(X, \nuc")` the nuclear norm. The nuclear norm can also be defined as the sum of the singular values of `X`.

The functions `max_entries` and `min_entries` give the largest and smallest entry, respectively, in a single expression. These functions should not be confused with `max_elemwise` and `min_elemwise` (see Section A.1.4). The functions `max_elemwise` and `min_elemwise` return the maximum or minimum of a list of scalar expressions.

The function `sum_entries` sums all the entries in a single expression. The built-in R `sum` should be used to add together a list of expressions. For example, the following code sums three expressions.

```
R> sum(expr1, expr2, expr3)
```

Some functions such as `sum_entries`, `cvxr_norm`, `max_entries`, and `min_entries` can be applied along an axis. Given an m by n expression `expr`, the line `func(expr, axis = 1)` applies `func` to each row, returning an m by 1 expression. The line `func(expr, axis = 2)` applies `func` to each column, returning a 1 by n expression. For example, the following code sums along the columns and rows of a matrix variable:

Function	Meaning	Domain	Sign	Curvature	Monotonicity
`abs(x)`	$\lvert x\rvert$	$x \in \mathbf{R}$	$+$	convex	$\nearrow$ for $x \geq 0$, $\swarrow$ for $x \leq 0$
`entr(x)`	$-x\log(x)$	$x > 0$	$\pm$	concave	none
`exp(x)`	e^x	$x \in \mathbf{R}$	$+$	convex	$\nearrow$
`huber(x, M = 1)` $M \geq 0$	$\begin{cases} x^2 & \lvert x\rvert \leq M \\ 2M\lvert x\rvert - M^2 & \lvert x\rvert > M \end{cases}$	$x \in \mathbf{R}$	$+$	convex	$\nearrow$ for $x \geq 0$, $\swarrow$ for $x \leq 0$
`inv_pos(x)`	$1/x$	$x > 0$	$+$	convex	$\swarrow$
`kl_div(x, y)`	$x\log(x/y) - x + y$	$x > 0$, $y > 0$	$+$	convex	none
`log(x)`	$\log(x)$	$x > 0$	$\pm$	concave	$\nearrow$
`log1p(x)`	$\log(x+1)$	$x > -1$	same as x	concave	$\nearrow$
`logistic(x)`	$\log(1+e^x)$	$x \in \mathbf{R}$	$+$	convex	$\nearrow$
`max_elemwise(x1,..., xk)`	$\max\{x_1, \ldots, x_k\}$	$x_i \in \mathbf{R}$	$\max(\text{sign}(x_1))$	convex	$\nearrow$
`min_elemwise(x1,..., xk)`	$\min\{x_1, \ldots, x_k\}$	$x_i \in \mathbf{R}$	$\min(\text{sign}(x_1))$	concave	$\nearrow$
`multiply(c, x)` $c \in \mathbf{R}$	$c \times x$	$x \in \mathbf{R}$	$\text{sign}(cx)$	affine	depends on c
`neg(x)`	$\max\{-x, 0\}$	$x \in \mathbf{R}$	$+$	convex	$\swarrow$
`pos(x)`	$\max\{x, 0\}$	$x \in \mathbf{R}$	$+$	convex	$\nearrow$
`power(x, 0)`	1	$x \in \mathbf{R}$	$+$	constant	
`power(x, 1)`	x	$x \in \mathbf{R}$	same as x	affine	$\nearrow$
`power(x, p)` $p = 2, 4, 8, \ldots$	x^p	$x \in \mathbf{R}$	$+$	convex	$\nearrow$ for $x \geq 0$, $\swarrow$ for $x \leq 0$
`power(x, p)` $p < 0$	x^p	$x > 0$	$+$	convex	$\swarrow$
`power(x, p)` $0 < p < 1$	x^p	$x \geq 0$	$+$	concave	$\nearrow$
`power(x, p)` $p > 1,\ p \neq 2, 4, 8, \ldots$	x^p	$x \geq 0$	$+$	convex	$\nearrow$
`scalene(x, alpha, beta)` alpha ≥ 0, beta ≥ 0	$\alpha\text{pos}(x) + \beta\text{neg}(x)$	$x \in \mathbf{R}$	$+$	convex	$\nearrow$ for $x \geq 0$, $\swarrow$ for $x \leq 0$
`sqrt(x)`	$\sqrt{x}$	$x \geq 0$	$+$	concave	$\nearrow$
`square(x)`	x^2	$x \in \mathbf{R}$	$+$	convex	$\nearrow$ for $x \geq 0$, $\swarrow$ for $x \leq 0$

Table A.3: Elementwise functions.

Function	Meaning	Domain	Sign	Curvature	Monotonicity
`bmat([[X11,..., X1q], ..., [Xp1,..., Xpq]])`	$\begin{bmatrix} X^{(1,1)} & \cdots & X^{(1,q)} \\ \vdots & & \vdots \\ X^{(p,1)} & \cdots & X^{(p,q)} \end{bmatrix}$	$X^{(i,j)} \in \mathbf{R}^{m_i \times n_j}$	$\text{sign}\left(\sum_{ij} X_{11}^{(i,j)}\right)$	affine	$\nearrow$
`conv(c, x)` $c \in \mathbf{R}^m$	$c * x$	$x \in \mathbf{R}^n$	$\text{sign}(c_1 x_1)$	affine	depends on c
`cumsum_axis(X, axis = 1)`	cumulative sum along given axis	$X \in \mathbf{R}^{m \times n}$	same as X	affine	$\nearrow$
`diag(x)`	$\begin{bmatrix} x_1 & & \\ & \ddots & \\ & & x_n \end{bmatrix}$	$x \in \mathbf{R}^n$	same as x	affine	$\nearrow$
`diag(X)`	$\begin{bmatrix} X_{11} \\ \vdots \\ X_{nn} \end{bmatrix}$	$X \in \mathbf{R}^{n \times n}$	same as X	affine	$\nearrow$
`diff(X, k = 1, axis = 1)` $k = 0, 1, 2, \ldots$	kth order differences (argument k is actually named *differences* and *lag* can also be used) along given axis	$X \in \mathbf{R}^{m \times n}$	same as X	affine	$\nearrow$
`hstack(X1,..., Xk)`	$\begin{bmatrix} X^{(1)} \cdots X^{(k)} \end{bmatrix}$	$X^{(i)} \in \mathbf{R}^{m \times n_i}$	$\text{sign}\left(\sum_i X_{11}^{(i)}\right)$	affine	$\nearrow$
`kronecker(C, X)` $C \in \mathbf{R}^{p \times q}$	$\begin{bmatrix} C_{11}X & \cdots & C_{1q}X \\ \vdots & & \vdots \\ C_{p1}X & \cdots & C_{pq}X \end{bmatrix}$	$X \in \mathbf{R}^{m \times n}$	$\text{sign}(C_{11}X_{11})$	affine	depends on C
`reshape_expr(X, c(m', n'))`	$X' \in \mathbf{R}^{m' \times n'}$	$X \in \mathbf{R}^{m \times n}$ $m'n' = mn$	same as X	affine	$\nearrow$
`vec(X)`	$x' \in \mathbf{R}^{mn}$	$X \in \mathbf{R}^{m \times n}$	same as X	affine	$\nearrow$
`vstack(X1,..., Xk)`	$\begin{bmatrix} X^{(1)} \\ \vdots \\ X^{(k)} \end{bmatrix}$	$X^{(i)} \in \mathbf{R}^{m_i \times n}$	$\text{sign}\left(\sum_i X_{11}^{(i)}\right)$	affine	$\nearrow$

Table A.4: Vector and matrix functions.

```
R> X <- Variable(5, 4)
R> row_sums <- sum_entries(X, axis = 1)   # Has size (5, 1)
R> col_sums <- sum_entries(X, axis = 2)   # Has size (1, 4)
```

CVXR ensures the implementation aligns with the `base::apply` function. The default in most cases is `axis = NA`, which treats an input matrix as one long vector, basically the same as `base::apply` with `MARGIN = c(1, 2)`. The exception is `cumsum_axis` (see Table A.3), which cannot take `axis = NA` and will throw an error.

A.1.4 Elementwise functions

These functions operate on each element of their arguments and are displayed in Table A.3. For example, if `X` is a 5 by 4 matrix variable, then `abs(X)` is a 5 by 4 matrix expression. Also, `abs(X)[1, 2]` is equivalent to `abs(X[1, 2])`.

Elementwise functions that take multiple arguments, e.g., `max_elemwise` and `multiply`, operate on the corresponding elements of each argument. For instance, if `X` and `Y` are both 3 by 3 matrix variables, then `max_elemwise(X, Y)` is a 3 by 3 matrix expression, where `max_elemwise(X, Y)[2, 1]` is equivalent to `max_elemwise(X[2, 1], Y[2, 1])`. Thus all arguments must have the same dimensions or be scalars, which are promoted appropriately.

A.1.5 Vector and matrix functions

These functions, shown in Table A.4, take one or more scalars, vectors, or matrices as arguments and return a vector or matrix.

The input to `bmat` is a list of lists of CVXR expressions. It constructs a block matrix. The elements of each inner list are stacked horizontally, and then the resulting block matrices are stacked vertically.

The output of `vec(X)` is the matrix `X` flattened in column-major order into a vector.

The output of `reshape_expr(X, c(m1, n1))` is the matrix `X` cast into an `m1` by `n1` matrix. The entries are taken from `X` in column-major order and stored in the output in column-major order.

Bibliography

S. Ahmed, O. Gozbasi, M. Savelsbergh, I. Crocker, T. Fox, and E. Schreibmann. An automated intensity-modulated radiation therapy planning system. *INFORMS Journal on Computing*, 22(4):568–583, 2010.

D. M. Aleman, D. Glaser, H. E. Romeijn, and J. F. Dempsey. Interior point algorithms: Guaranteed optimality for fluence map optimization IMRT. *Physics in Medicine and Biology*, 55(18):5467–5482, 2010.

D. M. Aleman, V. V. Mišić, , and M. B. Sharpe. Computational enhancements to fluence map optimization for total marrow irradiation using IMRT. *Computers and Operations Research*, 40(9):2167–2177, 2013.

E. D. Andersen and K. D. Andersen. The MOSEK interior point optimizer for linear programming: An implementation of the homogeneous algorithm. In *High Performance Optimization*, pages 197–232. Springer-Verlag, 2000.

M. Andersen, J. Dahl, Z. Liu, and L. Vandenberghe. Interior-point methods for large-scale cone programming. *Optimization for Machine Learning*, 5583, 2011.

O. Banerjee, L. E. Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516, 2008.

D. Bates and M. Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2019. URL `https://CRAN.R-project.org/package=Matrix`. R package version 1.2-18.

J. L. Bedford. Treatment planning for volumetric modulated arc therapy. *Medical physics*, 36(11):5128–5138, 2009.

J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman. Julia: A fast dynamic language for technical computing. arXiv:1209.5145 [cs.PL], 2012. URL `http://arxiv.org/abs/1209.5145`.

T. Bortfeld, J. Ramakrishnan, J. N. Tsitsiklis, and J. Unkelbach. Optimization of radiation therapy fractionation schedules in the presence of tumor repopulation. *INFORMS Journal on Computing*, 27(4):788–803, 2015.

N. Boyd, T. Hastie, S. Boyd, B. Recht, and M. I. Jordan. Saturating splines and feature selection. *Journal of Machine Learning Reserach*, 18(197):1–32, 2018.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing Markov chain on a graph. *SIAM Review*, 46(4):667–689, 2004.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.

S. Boyd, E. Busseti, S. Diamond, R. N. Kahn, K. Koh, P. Nystrup, and J. Speth. Multi-period trading via convex optimization. *Foundations and Trends in Optimization*, 3(1):1–76, 2017.

D. J. Brenner. The linear-quadratic model is an appropriate methodology for determining isoeffective doses at large doses per fraction. *Seminars in Radiation Oncology*, 18(4):234–239, 2008.

D. J. Brenner, L. R. Hlatky, P. J. Hahnfeldt, E. J. Hall, and R. K. Sachs. A convenient extension of the linear-quadratic model to include redistribution and reoxygenation. *International Journal of Radiation Oncology, Biology, Physics*, 32(2): 379–390, 1995.

E. Busseti, E. K. Ryu, and S. Boyd. Risk-constrained Kelly gambling. *Journal of Investing*, 25(3):118–134, 2016.

A. Canty and B. Ripley. *boot: Bootstrap Functions (Originally by Angelo Canty for S)*, 2020. URL `https://CRAN.R-project.org/package=boot`. R package version 1.3-25.

T. C. Y. Chan, H. Mahmoudzadeh, and T. G. Purdie. A robust-CVaR optimization approach with applications to breast cancer therapy. *European Journal of Operational Research*, 238(3):876–885, 2014.

W. Chen, J. Unkelbach, A. Trofimov, T. Madden, H. Kooy, T. Bortfeld, and D. Craft. Including robustness in multi-criteria optimization for intensity-modulated proton therapy. *Physics in Medicine and Biology*, 57(3):591–608, 2012.

P. S. Cho, S. Lee, R. J. M. II, S. Oh, S. G. Sutlief, and M. H. Phillips. Optimization of intensity modulated beams with volume constraints using two methods: Cost function minimization and projections onto convex sets. *Medical Physics*, 25(4): 435–443, 1998.

D. R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society B*, 20(2):215–242, 1958.

C. Davino, M. Furno, and D. Vistocco. *Quantile regression*. Wiley, 2013.

A. de la Zerda, B. Armbruster, and L. Xing. Formulating adaptive radiation therapy (ART) treatment planning into a closed-loop control framework. *Physics in Medicine and Biology*, 52(14):4137–4153, 2007.

J. O. Deasy. Multiple local minima in radiotherapy optimization problems with dose-volume constraints. *Medical Physics*, 24(7):1157–1161, 1997.

S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *Proceedings of the European Control Conference*, pages 3071–3076, 2013.

P. Dong, P. Lee, D. Ruan, T. Long, H. E. Romeijn, Y. Yang, D. Low, P. Kupelian, and K. Sheng. 4π non-coplanar liver sbrt: a novel delivery technique. *International Journal of Radiation Oncology* Biology* Physics*, 85(5):1360–1366, 2013.

L. Dümbgen and K. Rufibach. Maximum likelihood estimation of a log-concave density and its distribution function: Basic properties and uniform consistency. *Bernoulli*, 15(1):40–68, 2009.

L. Dümbgen and K. Rufibach. logcondens: Computations related to univariate log-concave density estimation. *Journal of Statistical Software*, 39(6):1–28, 2011.

D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011.

M. Ehrgott, Ç. Güler, H. W. Hamacher, and L. Shao. Mathematical optimization in intensity modulated radiation therapy. *4OR*, 6(3):199–262, 2008.

M. Ferris and M. Voelker. Fractionation in radiation treatment planning. *Mathematical Programming*, 101(2):387–413, 2004.

J. F. Fowler. The linear-quadratic formula and progress in fractionated radiotherapy. *The British Journal of Radiology*, 62(740):679–694, 1989.

D. A. Freedman. *Statistical Models: Theory and Practice*. Cambridge University Press, 2009.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

A. Fu, B. Ungun, L. Xing, and S. Boyd. A convex optimization approach to radiation treatment planning with dose constraints. *Optimization and Engineering*, 20(1): 277–300, March 2019.

A. Fu, B. Narasimhan, and S. Boyd. CVXR: An R package for disciplined convex optimization. *Journal of Statistical Software*, 94(14):1–34, August 2020a.

A. Fu, B. Narasimhan, D. W. Kang, S. Diamond, and J. Miller. *CVXR: Disciplined Convex Optimization*, 2020b. URL `https://CRAN.R-project.org/package=CVXR`. R package version 1.0-1.

A. Fu, L. Xing, and S. Boyd. Operator splitting for adaptive radiation therapy with nonlinear health constraints, 2021. arXiv:2105.01286 [physics.med-ph].

M. Gao, N. A. Mayr, Z. . Huang, H. Zhang, and J. Z. Wang. When tumor repopulation starts? the onset time of prostate cancer during radiation therapy. *Acta Oncologica*, 49(8):1269–1275, 2010.

I. M. Gelfand and S. V. Fomin. *Calculus of Variations*. Prentice-Hall, 1963.

E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. Optimal parameter selection for the alternating direction method of multipliers (ADMM): Quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, 2015.

C. Glide-Hurst, M. Bellon, R. Foster, C. Altunbas, M. Speiser, M. Altman, D. Westerly, N. Wen, B. Zhao, and M. Miften. Commissioning of the Varian TrueBeam linear accelerator: A multi-institutional study. *Medical physics*, 40(3):031719, 2013.

M. Grant and S. Boyd. CVX: MATLAB software for disciplined convex programming, version 2.1, 2014. URL `https://cvxr.com/cvx`.

M. Grant, S. Boyd, and Y. Ye. Disciplined convex programming. In L. Liberti and N. Maculan, editors, *Global Optimization: From Theory to Implementation*, Nonconvex Optimization and its Applications, pages 155–210. Springer-Verlag, 2006.

I. A. Griva and R. J. Vanderbei. Case studies in optimization: Catenary problem. *Optimization and Engineering*, 6(4):463–482, 2005.

Gurobi Optimization, Inc. *Gurobi Optimizer Reference Manual*, 2016. URL `http://www.gurobi.com`.

T. Halabi, D. Craft, and T. Bortfeld. Dose-volume objectives in multi-criteria optimization. *Physics in Medicine and Biology*, 51(15):3809–3818, 2006a.

T. Halabi, D. Craft, and T. Bortfeld. Dose-volume objectives in multi-criteria optimization. *Physics in Medicine and Biology*, 51:3809–3818, 2006b.

H. W. Hamacher and K.-H. Küfer. Inverse radiation therapy planning—a multiple objective optimization approach. *Discrete Applied Mathematics*, 118(1):145–161, 2002.

T. Hastie and H. Zou. Regularization and variable selection via the elastic-net. *Journal of the Royal Statistical Society B*, 67(2):301–320, 2005.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

A. Hölder. Designing radiotherapy plans with elastic constraints and interior point methods. *Health Care Management Science*, 6(1):5–16, 2003.

P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

D. R. Hunter and K. Lange. A tutorial on MM algorithms. *The American Statistician*, 58(1):30–37, 2004.

T. B. J., Bürkelbach, R. Boesecke, and W. Schlegel. Methods of image reconstruction from projections applied to conformation radiotherapy. *Physics in Medicine and Biology*, 35(10):1423–1434, 1990.

S. G. Johnson. *The NLopt Nonlinear-Optimization Package*, 2008. URL `http://ab-initio.mit.edu/nlopt/`.

J. R. A. Jr, S. D. Chang, M. J. Murphy, J. Doty, P. Geis, and S. L. Hancock. The cyberKnife: A frameless robotic system for radiosurgery. *Stereotactic and functional neurosurgery*, 69(1-4):124–128, 1998.

T. Kehwar. Analytical approach to estimate normal tissue complication probability using best fit of normal tissue tolerance doses into the NTCP equation of the linear quadratic model. *Journal of Cancer Research and Therapeautics*, 1(3):168–179, 2005.

J. L. Kelly. A new interpretation of information rate. *Bell System Technical Journal*, 35(4):917–926, 1956.

M. Kim, A. Ghate, and M. H. Phillips. A Markov decision process approach to temporal modulation of dose fractions in radiation therapy planning. *Physics in Medicine and Biology*, 54(14):4455–4476, 2009.

M. Kim, A. Ghate, and M. H. Phillips. A stochastic control formalism for dynamic biologically conformal radiation therapy. *European Journal of Operational Research*, 219(3):541–556, 2012.

R. Koenker. *Quantile Regression*. Cambridge University Press, 2005.

R. Koenker and I. Mizera. Convex optimization in R. *Journal of Statistical Software, Articles*, 60(5):1–23, 2014.

M. Langer, R. Brown, M. Urie, J. Leong, M. Stracher, and J. Shapiro. Large scale optimization of beam weights under dose-volume restrictions. *International Journal of Radiation Oncology Biology Physics*, 18(4):887–893, 1990.

E. Lee, T. Fox, and I. Crocker. Optimization of radiosurgery treatment planning via mixed integer programming. *Medical Physics*, 27(5):995–1004, 2000.

E. Lee, T. Fox, and I. Crocker. Integer programming applied to intensity-modulated radiation therapy treatment planning. *Annals of Operations Research*, 119(1–4): 165–181, 2003.

R. Li and L. Xing. An adaptive planning strategy for station parameter optimized radiation therapy (SPORT): Segmentally boosted VMAT. *Medical physics*, 40(5): 050701, 2013.

G. Lim and W. Cao. A two-phase method for selecting IMRT treatment beam angles: Branch-and-Prune and local neighborhood search. *European Journal of Operational Research*, 217(3):609–618, 2012.

T. Lipp and S. Boyd. Variations and extension of the convex-concave procedure. *Optimization and Engineering*, 17(2):263–287, June 2016.

M. S. Lobo and S. Boyd. The worst-case risk of a portfolio, 2000. URL `http://stanford.edu/~boyd/papers/pdf/risk_bnd.pdf`.

M. S. Lobo, M. Fazel, and S. Boyd. Portfolio optimization with linear and fixed transaction costs. *Annals of Operations Research*, 152(1):341–365, 2007.

J. Lofberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *Proceedings of the IEEE International Symposium on Computed Aided Control Systems Design*, pages 294–289, 2004. URL `http://users.isy.liu.se/johanl/yalmip`.

T. Lumley. Analysis of complex survey samples. *Journal of Statistical Software*, 9(8): 1–19, 2004.

T. Lumley. *Complex Surveys: A Guide to Analysis Using R*. John Wiley & Sons, 2010.

T. Lumley. *survey: Analysis of Complex Survey Samples*, 2020. URL `https://CRAN.R-project.org/package=survey`. R package version 4.0.

T. R. Mackie, T. Holmes, S. Swerdloff, P. Reckwerdt, J. O. Deasy, J. Yang, B. Paliwal, and T. Kinsella. Tomotherapy: A new concept for the delivery of dynamic conformal radiotherapy. *Medical Physics*, 20(6):1709–1719, 1993.

G. S. Mageras and R. Mohan. Application of fast simulated annealing to optimization of conformal radiation treatments. *Medical Physics*, 20(3):639–647, 1993.

H. M. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.

L. B. Marks, E. D. Yorke, A. Jackson, R. K. T. Haken, L. S. Constine, A. Eisbruch, S. M. Bentzen, J. Nam, and J. O. Deasy. Use of normal tissue complication probability (NTCP) models in the clinic. *International Journal of Radiation Oncology, Biology, and Physics*, 76(3 Suppl):S10–S19, 2010.

J. Miller, P. Quigley, and J. Zhu. CVXcanon: Automatic canonicalization of disciplined convex programs, 2015. URL `https://stanford.edu/class/ee364b/projects/2015projects/reports/miller_quigley_zhu_report.pdf`.

M. Mizuta, S. Takao, H. Date, N. Kishimoto, K. L. Sutherland, R. Onimaru, and H. Shirato. A mathematical study to select fractionation regimen based on physical dose distribution and the linear-quadratic model. *International Journal of Radiation Oncology, Biology, Physics*, 84(3):829–833, 2012.

K. M. Mullen and I. H. M. van Stokkum. *nnls: The Lawson-Hanson Algorithm for Non-Negative Least Squares (NNLS)*, 2012. URL `https://CRAN.R-project.org/package=nnls`. R package version 1.4.

J. C. Nash and R. Varadhan. Unifying optimization algorithms to aid software system users: optimx for R. *Journal of Statistical Software*, 43(9):1–14, 2011.

B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 2016.

U. Oelfke and T. Bortfeld. Inverse planning for photon and proton beams. *Medical Dosimetry*, 26(2):113–124, 2001.

M. R. Oskoorouchi, H. R. Ghaffari, T. Terlaky, and D. M. Aleman. An interior point constraint generation algorithm for semi-infinite optimization with healthcare application. *Operations Research*, 59(5):1184–1197, 2011.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. URL `https://www.R-project.org/`.

R. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42, 2000.

H. E. Romeijn, R. K. Ahuja, J. F. Dempsey, A. Kumar, and J. G. Li. A novel linear programming approach to fluence map optimization for intensity modulated radiation therapy treatment planning. *Physics in Medicine and Biology*, 48(21): 3521–3542, 2003.

H. E. Romeijn, J. Dempsey, and J. Li. A unifying framework for multi-criteria fluence map optimization models. *Physics in Medicine and Biology*, 49(10):1991–2013, 2004.

H. E. Romeijn, R. K. Ahuja, J. F. J. F. Dempsey, and A. Kumar. A new linear programming approach to radiation therapy treatment planning problems. *Operations Research*, 54(2):201–216, 2006.

I. I. Rosen, R. G. Lane, S. M. Morrill, and J. A. Belli. Treatment plan optimization using linear programming. *Medical Physics*, 18(2):141–152, 1991.

A. D. Roy. Safety first and the holding of assets. *Econometrica*, 20(3):431–449, 1952.

F. Saberian, A. Ghate, and M. Kim. Optimal fractionation in radiotherapy with multiple normal tissues. *Mathematical Medicine and Biology*, 33(2):211–252, 2016.

F. Saberian, A. Ghate, and M. Kim. Spatiotemporally optimal fractionation in radiotherapy. *INFORMS Journal on Computing*, 29(3):422–437, 2017.

A. Schweikard, A. Schlaefer, and J. J. R. Adler. Resampling: An optimization method for inverse planning in robotic radiosurgery. *Medical Physics*, 33(11):4005–4011, 2006.

C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, 1949.

X. Shen, S. Diamond, Y. Gu, and S. Boyd. Disciplined convex-concave programming. In *Proceedings of the 55th IEEE Conference on Decision and Control*, pages 1009–1014, 2016.

D. M. Shepard, M. C. Ferris, G. H. Olivera, and T. R. Mackie. Optimizing the delivery of radiation therapy to cancer patients. *Siam Review*, 41(4):721–744, 1999.

D. M. Shepard, M. C. Ferris, R. Ove, and L. Ma. Inverse treatment planning for Gamma Knife radiosurgery. *Medical Physics*, 27(9):2146–2149, 2000a.

D. M. Shepard, G. H. Olivera, P. J. Reckwerdt, and T. R. Mackie. Iterative approaches to dose optimization in tomotherapy. *Physics in Medicine and Biology*, 45(1):69–90, 2000b.

M. Y. Sir, M. A. Epelman, and S. M. Pollock. Stochastic programming for off-line adaptive radiotherapy. *Annals of Operation Research*, 196(1):767–797, 2012.

A. Skajaa and Y. Ye. A homogeneous interior-point algorithm for nonsymmetric convex conic optimization. *Mathematical Programming*, 150(2):391–422, 2015.

S. V. Spirou and C. Chui. A gradient inverse planning algorithm with dose-volume constraints. *Medical Physics*, 25(3):321–333, 1998.

B. K. Sriperumbudur and G. R. Lanckriet. On the convergence of the concave-convex procedure. *Advances in Neural Information Processing Systems*, pages 1759–1767, 2009.

Y. Sun, P. Babu, and D. P. Palomar. Majorization-minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, 65(3):794–816, February 2017.

H. D. Thames and J. H. Hendry. *Fractionation in Radiotherapy*. Taylor and Francis, 1987.

H. D. Thames, D. Kuban, L. B. Levy, E. M. Horwitz, P. Kupelian, A. Martinez, J. Michalski, T. Pisansky, H. Sandler, W. Shipley, M. Zelefsky, and A. Zietman. The role of overall treatment time in the outcome of radiotherapy of prostate cancer: An analysis of biochemical failure in 4839 men treated between 1987 and 1995. *Radiotherapy and Oncology*, 96(1):6–12, 2010.

The MathWorks Inc. Using credit scorecards with constrained logistic regression coefficients, 2018. URL `https://www.mathworks.com/help/finance/examples/credit-scorecards-with-constrained-logistic-regression-coefficients.html`.

The MathWorks Inc. *MATLAB – The Language of Technical Computing, Version R2019a.* Natick, Massachusetts, 2019. URL `http://www.mathworks.com/products/matlab/`.

S. Theußl, F. Schwendinger, and K. Hornik. ROI: The R optimization infrastructure package. *Research Report Series / Department of Statistics and Mathematics*, 133, 2017. WU Vienna University of Economics and Business, Vienna.

S. Theußl, F. Schwendinger, and H. W. Borchers. CRAN task view: Optimization and mathematical programming, 2020. URL `https://CRAN.R-project.org/view=Optimization`. Version 2020-05-21.

R. J. Tibshirani, H. Hoefling, and R. Tibshirani. Nearly-isotonic regression. *Technometrics*, 53(1):54–61, 2011.

E. L. Travis and S. L. Tucker. Isoeffect models and fractionated radiation therapy. *International Journal of Radiation Oncology, Biology, Physics*, 13(2):283–287, 1987.

M. Udell, K. Mohan, D. Zeng, J. Hong, S. Diamond, and S. Boyd. Convex optimization in Julia. In *Proceedings of the Workshop for High Performance Technical Computing in Dynamic Languages*, pages 18–28, 2014.

C. M. van Leeuwen, A. L. Oei, J. Crezee, A. Bel, N. A. P. Franken, L. J. A. Stalpers,

and H. P. Kok. The alfa and beta of tumours: A review of parameters of the linear-quadratic model, derived from clinical radiotherapy studies. *Radiation Oncology*, 13(96):1–11, 2018.

G. van Rossum et al. *Python Programming Language*, 2011. URL `http://www.python.org`.

S. Webb. Optimization of conformal radiotherapy dose distribution by simulated annealing. *Physics in Medicine and Biology*, 34(10):1349–1370, 1989.

S. Webb. Optimization by simulated annealing of three-dimensional, conformal treatment planning for radiation fields defined by a multileaf collimator: II. inclusion of two-dimensional modulation of the X-ray intensity. *Physics in Medicine and Biology*, 37(8):1689–1704, 1992.

S. J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, 1997.

Q. Wu and R. Mohan. Multiple local minima in IMRT optimization based on dose-volume criteria. *Medical Physics*, 29(7):1514–1527, 2002.

L. Xing and G. T. Y. Chen. Iterative methods for inverse treatment planning. *Physics in Medicine and Biology*, 41(10):2107–2123, 1996.

L. Xing, R. J. Hamilton, D. Spelbring, C. A. Pelizzari, G. T. Y. Chen, and A. L. Boyer. Fast iterative algorithms for three-dimensional inverse treatment planning. *Medical Physics*, 25(10):1845–1849, 1998.

Y. Xu, M. Liu, Q. Lin, and T. Yang. ADMM without a fixed penalty parameter: Faster convergence with new adaptive penalization. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1267—-1277, 2017a.

Z. Xu, M. Figueiredo, and T. Goldstein. Adaptive ADMM with spectral penalty parameter selection. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 718–727, 2017b.

Y. Yang and L. Xing. Optimization of radiotherapy dose-time fractionation with consideration of tumor specific biology. *Medical Physics*, 32(12):3666–3677, 2005.

M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

A. L. Yuille and A. Rangarajan. The concave-convex procedure (CCCP). *Neural Computation*, 15(4):915–936, April 2003.

M. Zarepisheh, M. Shakourifar, G. Trigila, P. S. Ghomi, S. Couzens, A. Abebe, L. Noreña, W. Shang, S. B. Jiang, and Y. Zinchenko. A moment-based approach for DVH-guided radiotherapy treatment plan optimization. *Physics in Medicine and Biology*, 58(6):1869–1887, 2013.

M. Zarepisheh, T. Long, N. Li, Z. Tian, H. E. Romeijn, X. Jia, and S. B. Jiang. A DVH-guided IMRT optimization algorithm for automatic treatment planning and adaptive radiotherapy replanning. *Medical Physics*, 41(6):061711, 2014.