Lecture 1 - Introduction

Fabricio Oliveira (with small modifications by Nuutti Hyvönen)

Last update: September 7, 2022

Abstract. In this lecture, we introduce the concept of nonlinear optimisation problems and the difference between mathematical programming and optimisation. We also discuss a few important examples to motivate the future developments in the course.

Outline of this lecture

1 What is optimisation?

An optimisation is one of these words that has many meanings, depending on the context you take as a reference. In the context of this course, optimisation refers to *mathematical optimisation*, which is a discipline of applied mathematics.

In mathematical optimisation, we build upon concepts and techniques from calculus, analysis, linear algebra, and other domains of mathematics to develop methods that allow us finding values for variables within a given domain that maximise (or minimise) the value of a function. In specific, we are trying to solve the following general problem:

$$\begin{aligned} &\text{min. } f(x) \\ &\text{subject to: } x \in X. \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is a vector of n *variables*, $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a *function* to be optimised (minimised) and $X \subseteq \mathbb{R}^n$ is a *domain* containing acceptable values for x.

In a general sense, these problems can be solved by employing the following strategy:

1. Analysing properties of functions under specific domains and deriving the conditions that must be satisfied such that a point x is a candidate optimal point.
2. Applying numerical methods that iteratively searches for points satisfying these conditions.

This idea is central in several domains of knowledge, and very often are defined under area-specific nomenclature. Fields such as economics, engineering, statistics, machine learning and, perhaps more broadly, operations research, are intensive users and developers of optimisation theory and applications.

1.1 Mathematical programming and optimisation

Operations research and mathematical optimisation are somewhat intertwined, as they both were born around a similar circumstance.

I like to separate *mathematical programming* from (mathematical) *optimisation*. Mathematical programming is a modelling paradigm, in which we rely on (very powerful, I might add) analogies to model *real-world* problems. In that, we look at a given decision problem considering that

- *variables* represent *decisions*, as in a business decision or a course of action. Examples include setting the parameter of (e.g., prediction) model, production systems layouts, geometries of structures, topologies of networks, and so forth;

- *domain* represents business rules or *constraints*, representing logic relations, design or engineering limitations, requirements, and such;
- function is an *objective function* that provides a measure of solution quality.

With these in mind, we can represent the decision problem as a *mathematical programming model* of the form of (1) that can be solved using *optimisation* methods. From now on, we will refer to this specific class of models as mathematical optimisation models, or optimisation models for short. We will also use the term to *solve the problem* to refer to the task of finding optimal solutions to optimisation models.

This course is mostly focused on the optimisation techniques employed to find optimal solutions for these models. As we will see, depending on the nature of the functions f and g that are used to formulate the model, some methods might be more or less appropriate. Further complicating the issue, for models of a given nature, there might be alternative algorithms that can be employed and with no generalised consense whether one method is generally better performing than another.

1.2 Types of mathematical optimisation models

In general, the simpler the assumptions on the parts forming the optimisation model, the more efficient are the methods to solve such problems.

Let us define some additional notation that we will use from now on. Consider a model in the general form

$$\begin{aligned} \text{min. } & f(x) \\ \text{subject to: } & g_i(x) \leq 0,\ i = 1, \dots, m \\ & h_i(x) = 0,\ i = 1, \dots, l \\ & x \in X, \end{aligned}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is the objective function, $g : \mathbb{R}^m \mapsto \mathbb{R}^m$ is a collection of m inequality constraints and $h : \mathbb{R}^n \mapsto \mathbb{R}^l$ is a collection of l equality constraints.

Remark: in fact, every inequality constraint can be represented by an equality constraint by making $h_i(x) = g_i(x) + x_{n+1}$ and augmenting the decision variable vector $x \in \mathbb{R}^n$ to include the slack variable x_{n+1}. However, since these constraints are of very different nature, we will explicitly represent both whenever necessary.

The most general types of models are the following. We also use this as an opportunity to define some (admittedly confusing) nomenclature from the field of operations research that we will be using in these notes.

1. *Unconstrained models:* in these, the set $X = \mathbb{R}^n$ and $m = l = 0$. These are prominent in, e.g.,

machine learning and statistics applications, where f represents a measure of model fitness or prediction error.

2. *Linear programming (LP):* presumes linear objective function. $f(x) = c^\top x$ and constraints g and h affine, i.e., of the form $a_i^\top x - b_i$, with $a_i \in \mathbb{R}^n$ and $b \in \mathbb{R}$. Normally, $X = \{x \in \mathbb{R}^n \mid x_j \geq 0, j = 1, \ldots, n\}$ enforce that decision variables are constrained to be the non-negative orthant.

3. *Nonlinear programming (NLP):* some or all of the functions f, g, and h are nonlinear.

4. *Mixed-integer (linear) programming (MIP):* consists of an LP in which some (or all) of the variables are constrained to be integers. In other words, $X \subseteq \mathbb{R}^k \times \mathbb{Z}^{n-k}$. Very frequently, the integer variables are binary terms, i.e., $x_i \in \{0, 1\}$, for $i = 1, \ldots, n - k$ and are meant to represent true-or-false or yes-or-no conditions.

5. *Mixed-integer nonlinear programming (MINLP):* are the intersection of MIPs and NLPs.

Remark: notice that we use the vector notation $c^\top x = \sum_{j \in J} c_j x_j$, with $J = \{1, \ldots, N\}$. This is just a convenience for keeping the notation compact.

2 Examples of applications

We now discuss a few examples to illustrate the nature of the problems to which we will develop solution methods and their applicability to real-world contexts.

2.1 Resource allocation and portfolio optimisation

In a general sense, any mathematical optimisation model is an instantiation of the *resource allocation problem*. A resource allocation problem consists of how to design an optimal allocation of resources to tasks, such that a given outcome is optimised.

Classical examples typically include production planning settings, in which raw materials or labour resources are inputted into a system and a collection of products, a production plan, results from this allocation. The objective is to find the best production plan, that is, a plan with the maximum profit or minimum cost. Resource allocation problems can also appear in a less obvious setting, where the resources can be the capacity of transmission lines in an energy generation planning setting, for example.

Let $i \in I = \{1, \ldots, M\}$ be a collection of resources and $j \in J = \{1, \ldots, N\}$ be a collection of products. Suppose that, to produce one unit of product j, a quantity a_{ij} of resource i is required. Assume that the total availability of resource i is b_i and that the return per unit of product j is c_j.

Let x_j be the decision variable representing total of product j produced. The resource allocation problem can be modelled as

$$\text{max.} \quad \sum_{j \in J} c_j x_j \tag{2}$$

$$\text{subject to:} \quad \sum_{j \in J} a_{ij} x_j \leq b_i, \ \forall i \in I \tag{3}$$

$$x_j \geq 0, \ \forall j \in J. \tag{4}$$

Equation (2) represents the objective function, in which we maximise the total return obtained from a given production plan. Equation (3) quantify the resource requirements for a given production plan and enforce that such a requirement does not exceed the resource availability. Finally, constraint (4) defines the domain of the decision variables.

Notice that, as posed, the resource allocation problem is linear. This is perhaps the most basic, and also most diffused setting for optimisation models for which very reliable and mature technology is available. In this course, we will concentrate on methods that can solve variants of this model in which the objective function and/or the constraints are required to include nonlinear terms.

One classic variant of resource allocation that include nonlinear terms is the *portfolio optimisation problem*. In this, we assume that a collection of assets $j \in J = \{1, \ldots, N\}$ are available for investment. In this case, capital is the single (actual) resource to be considered. Each asset has random return R_j, with an expected value $\mathbb{E}[R_j] = \mu_j$. Also, the covariance between two assets $i, j \in J$ is given by $\sigma_{ij} = \mathbb{E}[(R_i - \mu_i)(R_j - \mu_j)]$, which can be denoted as the covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_{11} & \ldots & \sigma_{1N} \\ \vdots & \ddots & \vdots \\ \sigma_{N1} & \ldots & \sigma_{NN} \end{bmatrix}.$$

Markowitz (1952) proposed using $x^\top \Sigma x$ as a risk measure that captures the variability in the return of the assets. Given the above, the optimisation model that provides the investment portfolio with the least risk, given a minimum requirement ϵ in terms of expected returns is given by

$$\text{min.} \quad x^\top \Sigma x \tag{5}$$

$$\text{subject to:} \quad \mu^\top x \geq \epsilon \tag{6}$$

$$\sum_{j \in J} x_j = 1 \tag{7}$$

$$x_j \geq 0, \ \forall j \in J. \tag{8}$$

Objective function (5) represents the portfolio risk to be minimised, while constraint (6) enforces that the expected return must be at least ϵ. Notice that ϵ can be seen as a resource that has to be

(at least) completely depleted, if one wants to do a parallel with the resource allocation structure discussed early. Constraints (7)–(8) define the domain of the decision variables. Notice how the problem is posed in a scaled form, where $x_j \in [0, 1]$ represents a percentage of a hypothetical available capital for investment.

In this example, the problem is nonlinear due to the quadratic nature of the objective function $x^\top \Sigma x = \sum_{i,j \in J} \sigma_{ij} x_i x_j$. As we will see later on, there are efficient methods that can be employed to solve quadratic problems like this.

2.2 The pooling problem: refinery operations planning

The *pooling problem* is another example of a resource allocation problem that naturally presents nonlinear constraints. In this case, the production depends on *mixing operations*, known as pooling, to obtain certain product specification for a given property.

As an illustration, suppose that products $j \in J = \{1, \dots, N\}$ are produced by mixing byproducts $i \in I_j \subseteq I = \{1, \dots, M\}$. Assume that the qualities of byproducts q_i are known and that there is no reaction between byproducts. Each product is required to have a property value q_j within an acceptable range $[\underline{q}_j, \overline{q}_j]$ to be classified as product j. In this case, mass and property balances are calculated as

$$x_j = \sum_{i \in I_j} x_i, \ \forall j \in J \tag{9}$$

$$q_j = \frac{\sum_{i \in I_j} q_i x_i}{x_j}, \ \forall j \in J. \tag{10}$$

These can then incorporated into the resource allocation problem accordingly. One key aspect associated with pooling problem formulations is that the property balances represented by (10) define *nonconvex* feasibility regions. As we will see later, convexity is a powerful property that allows for developing efficient solution methods and its absence typically compromises computational performance and tractability in general.

2.3 Robust optimisation

Robust optimisation is a subarea of mathematical programming concerned with models that support decision-making under *uncertainty*. In specific, the idea is to devise a formulation mechanism that can guarantee feasibility of the optimal solution in face of variability, ultimately taking a risk-averse standpoint.

Consider the resource allocation problem from Section 2.1. Now, suppose that the parameters $\tilde{a}_i \in \mathbb{R}^N$ associated with a given constraint $i \in I = \{1, \dots, M\}$ are uncertain with a unknown

probability distribution. The resource allocation problem can then be formulated as

$$\begin{aligned} \text{max.} \quad & c^\top x \\ \text{subject to:} \quad & \tilde{a}_i^\top x \leq b_i, \ \forall i \in I \\ & x_j \geq 0, \ \forall j \in J. \end{aligned}$$

Let us assume that the only information available are observations $\hat{a}_i$, from which we can estimate a nominal value $\overline{a}_i$. This is illustrated in Figure 1, in which 100 random observations are generated for $\tilde{a}_i = [\tilde{a}_{i1}, \tilde{a}_{i2}]$ with $\tilde{a}_{i1} \sim \text{Normal}(10, 2)$ and $\tilde{a}_{i2} \sim \text{Normal}(5, 3)$ for a single constraint $i \in I$. The nominal values are assumed to have coordinates given by the average values used in the Normal distributions. Our objective is to develop a model that incorporates a given level of protection in

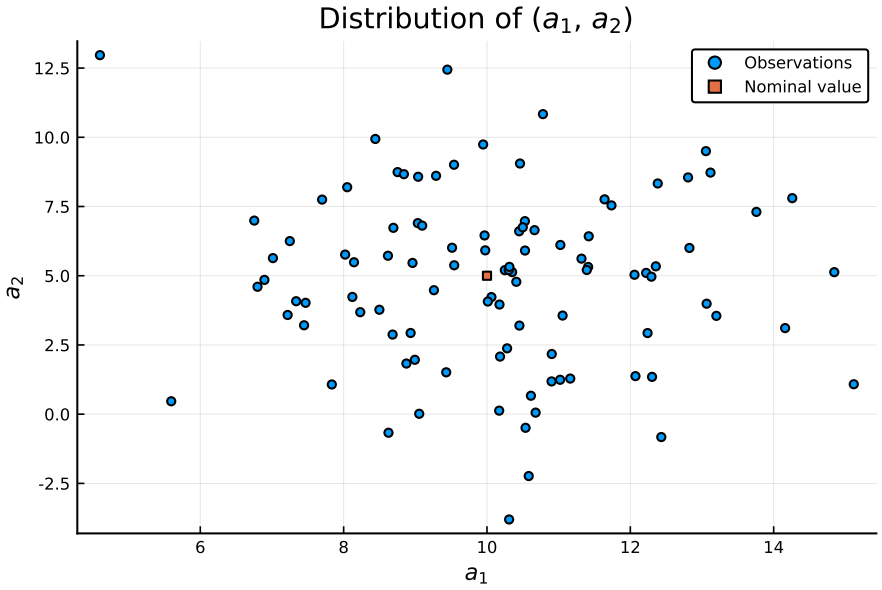


Figure 1: One hundred random realisations for $\tilde{a}_i$.

terms of feasibility guarantees. That is, we would like to develop a model that provides solutions that are guaranteed to remain feasible if the realisation of $\tilde{a}_i$ falls within an *uncertainty set* ϵ_i of size controlled by the parameter Γ_i. The idea is that the bigger the uncertainty set ϵ_i, the more robust is the solution, which typically comes at the expense of accepting solutions with expected worse performance.

The tractability of robust optimisation models depends on the geometry of the uncertainty set employed. Let us assume in what follows that

$$\epsilon_i = \{\overline{a}_i + P_i u \mid \|u\|_2 \leq \Gamma_i\} \tag{11}$$

is an ellipsoid with a characteristic matrix P_i and a parameter $\Gamma_i > 0$ that scales the size of the

ellipsoid. The singular values and corresponding left-singular vectors of P_i show how the ellipsoid extends in different directions from $\overline{a}_i$. In fact, for any ellipsoid ϵ_i, the characteristic matrix P_i can be chosen to be symmetric and positive semidefinite, in which case the singular values and singular vectors of P_i coincide with its eigenvalues and eigenvectors.

Remark: An alternative (perhaps more frequent) characterisation of an ellipsoid $\epsilon \subset \mathbb{R}^n$ centred at $\overline{x}$ is given by $\epsilon = \left\{x \in \mathbb{R}^n \mid (x-\overline{x})^\top A(x-\overline{x}) = \Gamma^2\right\}$. By choosing $A = (PP^\top)^{-1}$, we recover the representation in (11), assuming P is invertible.

We can now formulate the *robust counterpart*, which consists of a risk-averse version of the original resource allocation problem. In that, we try to anticipate the worst possible outcome and make decisions that are both optimal and guarantee feasibility in this worst-case sense. This standpoint translates into the following optimisation model.

$$\begin{aligned} \text{max.} \quad & c^\top x \\ \text{subject to:} \quad & \max_{a_i \in \epsilon_i} \left\{a_i^\top x\right\} \leq b_i, \ \forall i \in I \\ & x_j \geq 0, \ \forall j \in J. \end{aligned} \tag{12}$$

Notice how the constraint (12) has an embedded optimisation problem, turning into a *bi-level optimisation* problem. This highlights the issue associated with tractability, since solving the whole problem strongly depends on deriving tractable equivalent reformulations.

Assuming that the uncertainty set ϵ_i is an ellipsoid, the following result holds.

$$\max_{a_i \in \epsilon_i} \left\{a_i^\top x\right\} = \overline{a}_i^\top x + \max_{u} \left\{u^\top P_i x : \|u\|_2 \leq \Gamma_i\right\} \tag{13}$$

$$= \overline{a}_i^\top x + \Gamma_i \|P_i x\|_2. \tag{14}$$

In (13), we recast the inner problem in terms of the ellipsoidal uncertainty set, ultimately meaning that we recast the inner maximisation problem in terms of variable u. Since the only constraint is $\|u\|_2 \leq \Gamma_i$, in (14) we can derive a closed form for the inner optimisation problem.

With the closed form derived in (14), we can reformulate the original bi-level problem as a tractable single-level problem of the following form

$$\begin{aligned} \text{max.} \quad & c^\top x \\ \text{subject to:} \quad & \overline{a}_i^\top x + \Gamma_i \|P_i x\|_2 \leq b_i, \ \forall i \in I \\ & x_j \geq 0, \ \forall j \in J. \end{aligned} \tag{15}$$

Notice how the term $\Gamma_i \|P_i x\|_2$ creates a buffer for constraint (15), ultimately preventing the complete depletion of the resource. Clearly, this will lead to a suboptimal solution when compared to the original deterministic at the expense of providing protection against deviations in coefficients a_i. This difference is often referred to as the *price of robustness*.

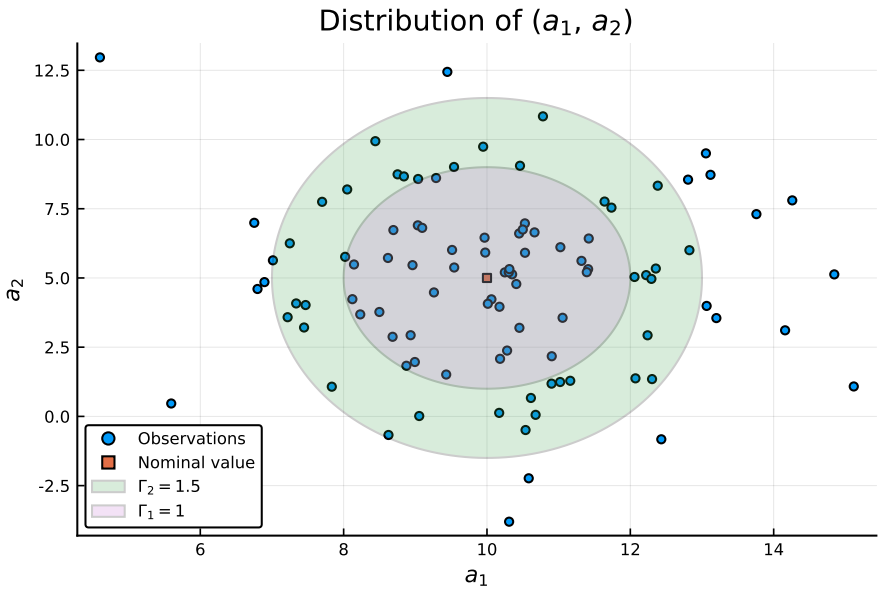


Figure 2: One hundred random realisations for $\tilde{a}_i$.

In Figure 2, we show the ellipsoidal sets for two levels of Γ_i for a single constraint i. We define

$$\epsilon_i = \left\{ \begin{bmatrix} 10 \\ 5 \end{bmatrix} + \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right\} \tag{16}$$

using the average and standard deviation of the original distributions that generated the observations. We plot the ellipsoids for $\Gamma_1 = 1$ and $\Gamma_2 = 1.5$, illustrating how the protection level increases as Γ increases. This can be inferred since the uncertainty set covers more of the observations and the formulation is such that feasibility is guaranteed for any observation within the uncertainty set.

2.4 Classification: support-vector machines

This is an example in which the resource allocation structure within the optimisation model is not as obvious. Suppose we are given a data set $D \subset \mathbb{R}^n$ with $|D| = N + M$ that can be divided into two disjunct sets $I^- = \{x_1, \ldots, x_N\}$ and $I^+ = \{x_1, \ldots, x_M\}$.

Each element in D is an observation of a given set of n features with values represented by a $x \in \mathbb{R}^n$ that has been classified as belonging to set I^- and I^+. Because of the availability of labelled data, classification is said to be an example of supervised learning in the field of machine learning.

Figure 3 illustrates this situation for $n = 2$, in which the orange dots represent points classified as belonging to I^- (negative observations) and the blue dots represent points classified as belonging to I^+ (positive observations).

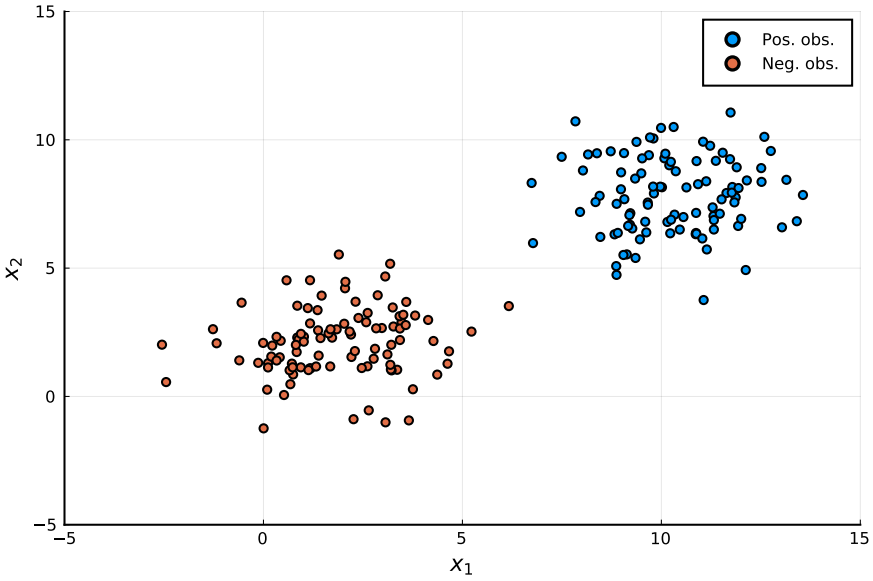


Figure 3: Two hundred observations for x_i classified to belong to I^- (orange) or I^+ (blue).

Our task is to obtain a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ from a given family of functions that is capable to, given an observed set of features $\hat{x}$, classify whether it belongs to I^- or I^+. In other words, we want to calibrate f such that

$$f(x_i) < 0, \ \forall x_i \in I^-, \ \text{and} \ f(x_i) > 0, \ \forall x_i \in I^+. \tag{17}$$

This function would then act as a classifier that could be employed to any new observation $\hat{x}$ made. If f is presumed to be an affine function of the form $f(x) = a^\top x - b$, then we obtain a *linear classifier*.

Our objective is to obtain $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$ such that misclassification error is minimised. Let us define the error measure as

$$e^-(x_i \in I^-; a, b) := \begin{cases} 0, \ \text{if} \ a^\top x_i - b \leq 0, \\ a^\top x_i - b, \ \text{if} \ a^\top x_i - b > 0. \end{cases}$$

$$e^+(x_i \in I^+; a, b) := \begin{cases} 0, \ \text{if} \ a^\top x_i - b \geq 0, \\ b - a^\top x_i, \ \text{if} \ a^\top x_i - b < 0. \end{cases}$$

Using this error measure, we can define constraints that capture deviation on each measure by means of nonnegative slack variables. Let $u_i \geq 0$ for $i = 1, \ldots, N$ and $v_i \geq 0$ for $i = 1, \ldots, M$ be slack variables that measure the *misclassification error* for $x_i \in I^-$ and $x_i \in I^+$, respectively.

The optimisation problem that finds optimal parameters a and b can be stated as

$$\text{min.} \quad \sum_{i=1}^{M} u_i + \sum_{i=1}^{N} v_i \tag{18}$$

$$\text{subject to:} \quad a^\top x_i - b - u_i \leq 0, \ i = 1, \ldots, M \tag{19}$$

$$a^\top x_i - b + v_i \geq 0, \ i = 1, \ldots, N \tag{20}$$

$$\|a\|_2 = 1 \tag{21}$$

$$u_i \geq 0, \ i = 1, \ldots, N \tag{22}$$

$$v_i \geq 0, \ i = 1, \ldots, M \tag{23}$$

$$a \in \mathbb{R}^n, \ b \in \mathbb{R}. \tag{24}$$

The objective function (18) accumulates the total misclassification error. Constraint (19) allows for capturing the misclassification error for each $x_i \in I^-$. Notice that $u_i = \max\{0, a^\top x_i - b\} = e^-(x_i \in I^-; a, b)$. Likewise, constraint (20) guarantees that $v_i = e^+(x_i \in I^+; a, b)$. To avoid trivial solutions in which $(a, b) = (0, 0)$, the normalisation constraint $\|a\|_2 = 1$ is imposed in constraint (21), which turns the model nonlinear.

Solving the model (18)–(24) provides optimal (a, b) which translates into the classifier represented as the green line in Figure 4.

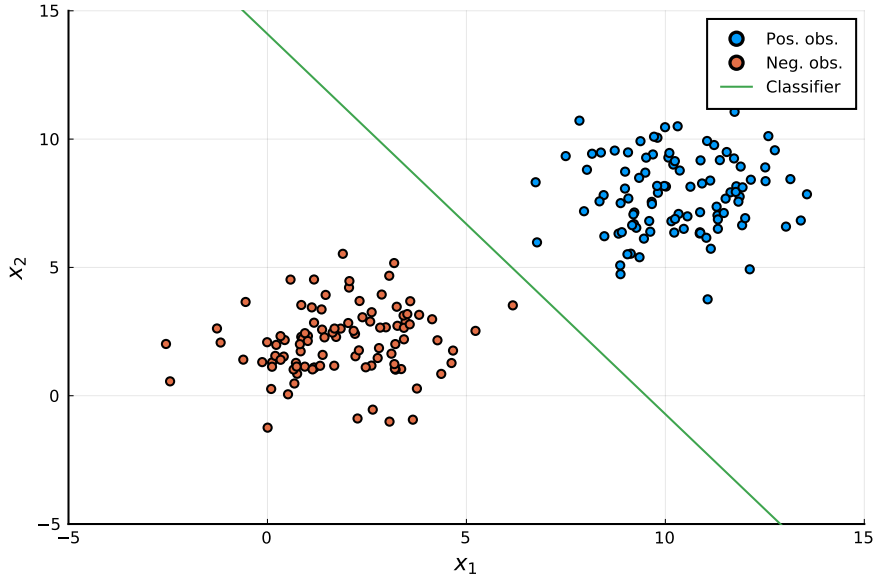


Figure 4: Two hundred observations for x_i classified to belong to I^- (orange) or I^+ (blue) with a classifier (green).

A variant referred to as *robust classifier* penalises not only the the misclassification error, but also the observations within a given slab $S = \{x \in \mathbb{R}^n \mid -1 \leq a^\top x - b \leq 1\}$. Notice that, being the two

lines defined by $f^-(x) : a^\top x - b = -1$ and $f^+(x) : a^\top x - b = +1$, the distance between the two hyperplanes is given by $\frac{2}{\|a\|_2}$.

Accordingly, we redefine our error measures as follows.

$$e^-(x_i \in I^-; a, b) := \begin{cases} 0, \text{ if } a^\top x_i - b \leq -1, \\ |a^\top x_i - b + 1|, \text{ if } a^\top x_i - b > -1. \end{cases}$$

$$e^+(x_i \in I^+; a, b) := \begin{cases} 0, \text{ if } a^\top x_i - b \geq 1, \\ |b - a^\top x_i + 1|, \text{ if } a^\top x_i - b < 1. \end{cases}$$

By doing so, a penalty is applied not only to those points that were misclassified but also to those points correctly classified that happen to be inside the slab S. To define an optimal robust classifier, one must trade off the size of the slab, which is inversely proportional to $\|a\|$, and the total of observations that fall in the slab S. The formulation for the robust classifier then becomes

$$\text{min.} \quad \sum_{i=1}^{M} u_i + \sum_{i=1}^{N} v_i + \gamma \|a\|_2^2 \tag{25}$$

$$\text{subject to:} \quad a^\top x_i - b - u_i \leq -1, \ i = 1, \ldots, M \tag{26}$$

$$a^\top x_i - b + v_i \geq 1, \ i = 1, \ldots, N \tag{27}$$

$$u_i \geq 0, \ i = 1, \ldots, N \tag{28}$$

$$v_i \geq 0, \ i = 1, \ldots, M \tag{29}$$

$$a \in \mathbb{R}^n, \ b \in \mathbb{R}. \tag{30}$$

In objective function (25), the errors accumulated in variables u_i, $i = 1, \ldots, N$ and v_i, $i = 1, \ldots, M$ and the squared norm $\|a\|_2^2$ are considered simultaneously. The term γ is a scalar used to impose an emphasis on minimising the norm $\|a\|_2$ and incentivising a larger slab S (recall that the slab is large for smaller $\|a\|_2$). The squared norm $\|a\|_2^2$ is considered instead as a means to recover differentiability, as the norm $\|a\|_2$ is not differentiable. Later on, we will see how beneficial it is for optimisation methods to be able to assume differentiability. Moreover, note how in constraints (26) and (27) u and v also accumulate penalties for correctly classified x_i that happen to be between the slab S, that is, that have term $a^\top x - b$ larger/ smaller than -1/ +1. Figure 5 shows a robust classifier an arbitrary value of γ.

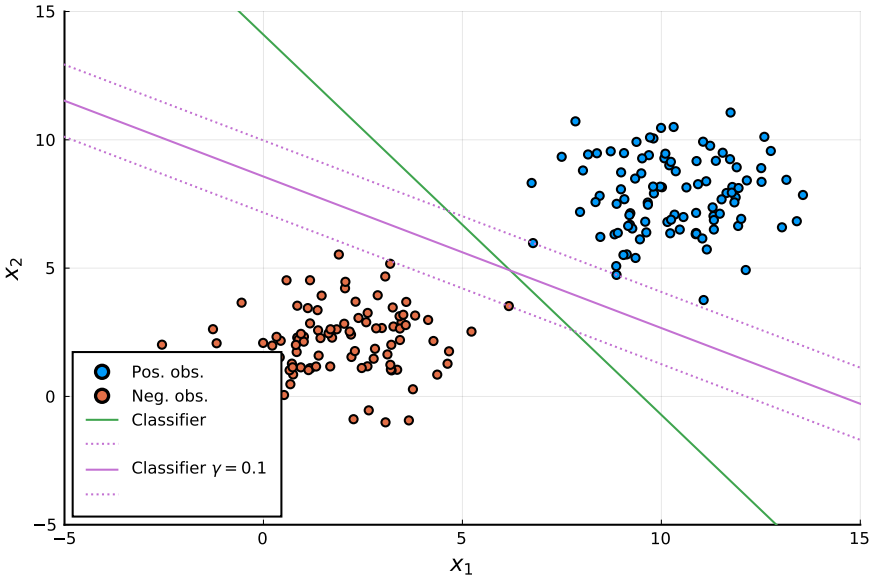


Figure 5: Two hundred observations for x_i classified to belong to I^- (orange) or I^+ (blue).

Remark: robust classifiers are known in the machine learning literature as *support vector machines*, where the support vectors are the observations that support the slab.

Lecture 2 - Convex sets

Fabricio Oliveira (with small modifications by Nuutti Hyvönen)

Last update: September 10, 2022

Abstract. In this lecture, we discuss concepts related to sets, which will later in the course serve as a framework for dealing with constraints in optimisation problems. We will also see that the convexity of sets can be used to infer the convexity of functions, a topic we will discuss in detail in the next class. We describe how to identify convexity in sets and discuss set-related concepts that will be useful when discussing optimality conditions later. We also discuss the concepts of separation and support, which are central in several results in optimisation theory, such as the Farkas theorem.

Outline of this lecture

1 Convexity and optimisation

Convexity is perhaps the most important property that the elements forming an optimisation problem can present. Paraphrasing Tyrrell Rockafellar:

> ... in fact, the great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity.

The importance of convexity will become clear later in the course. In a nutshell, the existence of convexity allows us to infer global properties of a solution (i.e., that holds for all of its domain) by considering exclusively local information (such as gradients, for example). This is critical in the context of optimisation, since most of the methods we know to perform well in practice are designed to find solutions that satisfy local optimality conditions. Once convexity is attested, one can then guarantee that these local solutions are in fact globally optimal without exhaustively exploring the solution space.

For a problem of the form

$$\begin{aligned}(P): \quad & \text{min. } f(x) \\ & \text{subject to: } x \in X\end{aligned}$$

to be convex, we need to verify whether f is a *convex function* and X is a *convex set*. If both statements hold true, we can conclude that P is a *convex problem*. We start looking into how to identify convex sets, since we can use the convexity of sets to infer the convexity of functions.

2 Identifying convexity of sets

Before we formally define convex sets, let us first look at the idea of *combinations*. For that, let $S \subseteq \mathbb{R}^n$ be a set and $x_j \in S$ for $j = 1, \ldots, k$ be an arbitrary finite collection of vectors (i.e., n-dimensional "points") belonging to S. Then, we have that:

- A *linear combination* of x_j, $j = 1, \ldots, k$, is

$$x = \sum_{j=1}^{k} \lambda_j x_j, \tag{1}$$

  where $\lambda_j \in \mathbb{R}$ for $j = 1, \ldots, k$.

- An *affine combination* is a linear combination, with the additional constraint that $\sum_{j=1}^{k} \lambda_j = 1$.

- A *conic combination* is a linear combination with the additional condition that $\lambda_j \geq 0$ for $j = 1, \ldots, k$.
- And finally, a *convex combination* is both an affine and a conic combination, i.e., it is a linear combination with the additional conditions that $\sum_{j=1}^{k} \lambda_j = 1$ and $\lambda_j \geq 0$ for $j = 1, \ldots, k$. (Observe that this immediately implies that $\lambda_j \in [0, 1]$ for $j = 1, \ldots, k$.)

We say that a set is convex if it contains the line segment between any two points belonging to the set:

Definition 1 (Convex sets)**.** *A set $S \subseteq \mathbb{R}^n$ is said to be convex if $x = \lambda x_1 + (1 - \lambda)x_2$ belongs to S for all $x_1, x_2 \in S$ and $0 \leq \lambda \leq 1$.*

It is equivalent to say that a set is convex if it contains all convex combinations of points in the set:

Corollary 2 (Convex sets)**.** *A set $S \subseteq \mathbb{R}^n$ is convex if and only if all (finite) convex combinations of its elements belong to S.*

Proof for those who are interested. (i) Assume first that $S \subseteq \mathbb{R}^n$ contains all finite convex combinations of its elements, that is, for any $k \in \mathbb{N}$,

$$\sum_{j=1}^{k} \lambda_j x_j \in S$$

if $x_1, \ldots, x_k \in S$ and $\lambda_j \geq 0$, $j = 1, \ldots, k$, satisfy $\sum_{j=1}^{k} \lambda_j = 1$. In particular, this means that $x = \lambda x_1 + (1 - \lambda)x_2$ belongs to S for any $x_1, x_2 \in S$ and $0 \leq \lambda \leq 1$. Thus, S is convex. (Observe that $\lambda \geq 0$, $1 - \lambda \geq 0$ and $\lambda + (1 - \lambda) = 1$, which means that the considered x is a convex combination of x_1 and x_2.)

(ii) Assume next that $S \subseteq \mathbb{R}^n$ is convex, and let us prove via mathematical induction that S contains all convex combinations of its elements. To begin with, observe that S obviously contains all convex combinations of any $x_1 \in S$ since the only convex combination of a single vector is that vector itself. This is the *base case* for our induction argument.

Fix $l \in \mathbb{N}$ and assume that

$$\sum_{j=1}^{l} \mu_j y_j \in S \tag{2}$$

for any $y_1, \ldots, y_l \in S$ and $\mu_j \geq 0$, $j = 1, \ldots, l$, that satisfy $\sum_{j=1}^{l} \mu_j = 1$. That is, we assume all "l-dimensional" convex combinations of elements in S belong to S, which is our *induction hypothesis*. The aim is to use (2) and the convexity of S to prove that all "$(l + 1)$-dimensional" convex combinations of elements in S also belong to S, which is the *induction step*.

To this end, let $x_1, \ldots, x_{l+1} \in S$ be arbitrary and assume that $\lambda_j \geq 0$, $j = 1, \ldots, l + 1$, satisfy $\sum_{j=1}^{l+1} \lambda_j = 1$. Without loss of generality, we may assume that $0 \leq \lambda_{l+1} < 1$. Indeed, due to the

conditions posed on λ_j, $j = 1, \ldots, l+1 \geq 2$, all of them cannot equal 1, and we can choose the numbering so that λ_{l+1} has this property. We have

$$x := \sum_{j=1}^{l+1} \lambda_j x_j = \lambda_{l+1} x_{l+1} + (1 - \lambda_{l+1}) \sum_{j=1}^{l} \mu_j x_j, \tag{3}$$

where $\mu_j = \lambda_j / (1 - \lambda_{l+1})$. Obviously, $\mu_j \geq 0$ for all $j = 1, \ldots, l$ and furthermore,

$$\sum_{j=1}^{l} \mu_j = \frac{1}{1 - \lambda_{l+1}} \sum_{j=1}^{l} \lambda_j = \frac{1}{1 - \lambda_{l+1}} \Big(\sum_{j=1}^{l+1} \lambda_j - \lambda_{l+1} \Big) = \frac{1 - \lambda_{l+1}}{1 - \lambda_{l+1}} = 1.$$

According to the induction hypothesis (2), the l-dimensional convex combination $\sum_{j=1}^{l} \mu_j x_j$ belongs to S, as does x_{l+1} by assumption. Because S is convex, the arbitrary "$(l+1)$-dimensional" convex combination x in (3) also belongs to S. This completes the induction step and the proof. □

Definition 1 and Corollary 2 are useful as they allow to show that some set operations preserve convexity.

2.1 Convexity-preserving set operations

Lemma 3 (Convexity-preserving operations)**.** *Let S_1 and S_2 be convex sets in $\mathbb{R}^n$. Then, the sets resulting from the following operations are also convex.*

1. *Intersection: $S = S_1 \cap S_2$;*
2. *Minkowski addition: $S = S_1 + S_2 = \{x_1 + x_2 : x_1 \in S_1, x_2 \in S_2\}$;*
3. *Minkowski difference: $S = S_1 - S_2 = \{x_1 - x_2 : x_1 \in S_1, x_2 \in S_2\}$;*
4. *Affine transformation: $S = \{Ax + b : x \in S_1\}$.*

Figures 1 and 2 illustrate the concept behind some of these set operations. Showing that the sets resulting from the operations in Lemma 3 are convex typically entails showing that convex combinations of elements in the resulting set S also belong to S_1 and S_2.

2.2 Examples of convex sets

There are several familiar sets that are known to be convex. Having the knowledge that these sets are convex is useful as a building block for determining the convexity of more complicated sets.

Some important examples of convex sets include:

- Them empty set $\emptyset$, any singleton $\{\overline{x}\}$ and the whole space $\mathbb{R}^n$;

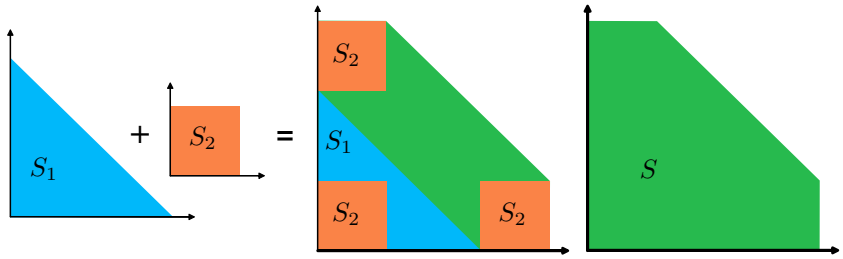


Figure 1: Minkowski sum of two convex sets.

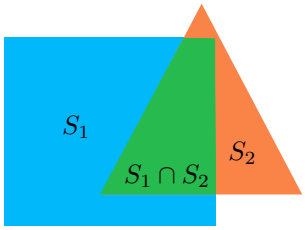


Figure 2: Intersection of two convex sets.

- hyperplanes: $H = \{x : p^\top x = \alpha\} \subset \mathbb{R}^n$, where $\mathbb{R}^n \ni p \neq 0$ is a normal vector and $\alpha \in \mathbb{R}$ is a scalar. Notice that H can be equivalently represented as $H = \{x \in \mathbb{R}^n : p^\top (x - \overline{x}) = 0\}$ for any $\overline{x} \in H$;
- halfspaces: $S = \{x : p^\top x \leq \alpha\} \subset \mathbb{R}^n$;
- polyhedral sets: $P = \{x : Ax \leq b\} \subset \mathbb{R}^n$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^n$;
- norm-induced sets (balls): $B = \{x : \|x - \overline{x}\| \leq \alpha\} \subseteq \mathbb{R}^n$, where $\|\cdot\|$ is any norm and α a scalar;
- norm cones: $C = \{(x, \alpha) \in \mathbb{R}^{n+1} : \|x\| \leq \alpha\}$;

For example, let us consider the polyhedral set $P = \{x \in \mathbb{R}^n : Ax \leq b\} \subset \mathbb{R}^n$ with A being a $m \times n$ matrix. Notice that S is the intersection of a collection of half-spaces $H_i = \{x \in \mathbb{R}^n : a_i^\top x \leq b_i\}$, where a_i are vectors from the rows of the matrix A and b_i are the components of the column vector b. We know that H_i are convex sets, thus $P = \cap_{i=1}^m H_i$ is also convex, as the intersection of sets is a convexity-preserving set operation.

2.2.1 Hyperplanes and halfspaces

Hyperplanes and halfspaces will play a central role in the developments we will see in our course. Therefore, let us take a moment and discuss some important aspects related these convex sets. First, notice that, geometrically, a hyperplane $H \subset \mathbb{R}^n$ can be interpreted as the set of points with

a *constant* inner product to a given vector $p \in \mathbb{R}^n$, while $\overline{x}$ determines the offset of the hyperplane from the origin. That is,

$$H = \{x : p^\top (x - \overline{x}) = 0\} \equiv \overline{x} + p^\perp,$$

where $p^\perp$ is the orthogonal complement of p, i.e., the set of vectors orthogonal to p, which is given by $\{x \in \mathbb{R}^n : p^\top x = 0\}$.

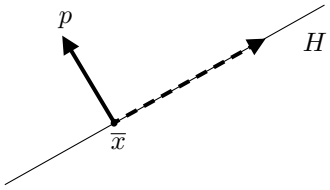


Figure 3: A hyperplane $H = \{x \in \mathbb{R}^n : p^\top (x - \overline{x}) = 0\}$ with normal vector p displaced to $\overline{x}$.

Analogously, a (closed) halfspace can be represented as $S = \{x \in \mathbb{R}^n : p^\top (x - \overline{x}) \leq 0\}$ where $p^\top x = p^\top \overline{x}$ is the hyperplane that forms the boundary of the halfspace. This definition suggests a simple geometrical interpretation: the halfspace S consists of $\overline{x}$ plus any vector with an obtuse or right angle (i.e., greater of equal to 90°) with the outward normal vector p.

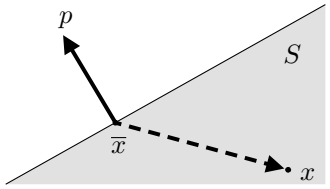


Figure 4: A halfspace $S = \{x \in \mathbb{R}^n : p^\top (x - \overline{x}) \leq 0\}$ defined by the same hyperplane H. Notice how the vectors p and $x - \overline{x}$ form angles greater or equal than $90°$.

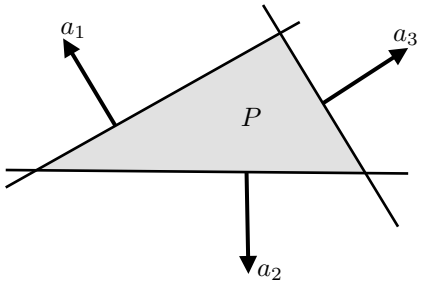


Figure 5: A polyhedron P formed by the intersection of three halfspace. Each hyperplane $H_i = \{x \in \mathbb{R}^n : a_i^\top x \leq b_i\}$, for $i = 1, 2, 3$, has a normal vector a_i, and has an offset from the origin b_i (which cannot be seen once project on a 2-dimensional plane as in the picture).

2.2.2 Norm balls and norm cones

A (closed) Euclidean ball (or simply ball) of radius ϵ in $\mathbb{R}^n$ has the form

$$B(\overline{x}, r) = \{x \in \mathbb{R}^n : \|x - \overline{x}\|_2 \leq \epsilon\} \equiv \{x \in \mathbb{R}^n : (x - \overline{x})^\top (x - \overline{x}) \leq \epsilon^2\}$$

As one might suspect, balls are convex, which can be proved by noting that

$$\begin{aligned}\|\lambda x_1 + (1-\lambda)x_2 - \overline{x}\|_2 &= \|\lambda(x_1 - \overline{x}) + (1-\lambda)(x_2 - \overline{x})\|_2 \\ &\leq \lambda\|x_1 - \overline{x}\|_2 + +(1-\lambda)\|x_2 - \overline{x}\|_2 \leq \epsilon.\end{aligned}$$

Notice that between the first and the second line, we use the triangle inequality, which states that $\|x + y\| \leq \|x\| + \|y\|$ for any two vectors x and y and any norm (including the Euclidean norm).

Euclidean balls are a special case of norm balls, which are defined as $B(\overline{x}, r) = \{x \in \mathbb{R}^n : \|x - \overline{x}\| \leq \epsilon\}$ where $\|\ \cdot\ \|$ is any norm on $\mathbb{R}^n$.

A related set is the norm cone, defined as $C(x, \alpha) = \{(x, \alpha) \in \mathbb{R}^{n+1} : \|x\| \leq \alpha\}$, where α is a scalar. For example, the second-order cone (also known as the ice cream cone or Lorentz cone) is the norm cone for the Euclidean norm.

Remark. *Norm induced sets (balls or cones) are convex for any norm* $\|x\|_p = (\sum_{i=1}^n x_i^p)^{\frac{1}{p}}$ *for* $x \in \mathbb{R}^n$ *and* $p \geq 1$.

3 Convex hulls

The *convex hull* of a set S, denoted as $\mathbf{conv}(S)$, is the set formed by all convex combinations of points in S. As the name suggests, $\mathbf{conv}(S)$ is a convex set, regardless of S being convex or not.

Another interpretation for $\mathbf{conv}(S)$ is to think of it as the tightest enveloping (convex) set that contains S. Notice that, if S is convex, then $S = \mathbf{conv}(S)$. Formally, convex hulls are defined as follows.

Definition 4 (Convex hull of a set)**.** *Let $S \subseteq \mathbb{R}^n$ be an arbitrary set. The convex hull of S, denoted by $\mathbf{conv}(S)$, is the collection of all convex combinations of elements in S. That is, $x \in \mathbf{conv}(S)$ if and only if for some $k \in \mathbb{N}$,*

$$x = \sum_{j=1}^{k} \lambda_j x_j$$

for $x_1, \ldots, x_k \in S$ and $\lambda_j \geq 0$, $j = 1, \ldots, k$, that satisfy $\sum_{j=1}^{k} \lambda_j = 1$.

From Definition 4, one can show that the convex hull $\mathbf{conv}(S)$ can also be defined as the intersection of all convex sets containing S. Perhaps the easiest way to visualise this is to think of the infinitely many half-space containing S and their intersection, which can only be S. Figure 6 illustrates the convex hull $\mathbf{conv}(S)$ of an nonconvex set S.

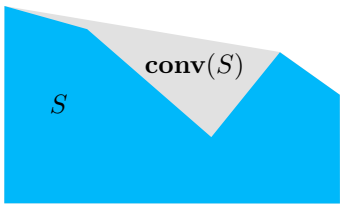


Figure 6: Example of an arbitrary set S (in solid blue) and its convex hull $\mathbf{conv}(S)$ (combined blue and grey areas).

The notion of convex hulls is a powerful tool in optimisation. One important application is using $\mathbf{conv}(S)$ to obtain approximations for a nonconvex S that can be exploited to solve an optimisation problem with constraint set defined by S. This is the underpinning technique in many important optimisation methods for such as branch-and-bound-based methods for nonconvex problems and decomposition methods (i.e., methods that solve large problems by breaking it into smaller parts that are presumably easier to solve).

In specific, let us consider the convex hull of a finite collection of discrete points. Some of these sets are so important in optimisation that they have their own names.

Definition 5. *Let $S = \{x_1, \ldots, x_{n+1}\} \subset \mathbb{R}^n$. Then $\mathbf{conv}(S)$ is called a* polytope*. If $x_1, \ldots, x_{n+1}$ are affinely independent (i.e., $x_2 - x_1, \ldots, x_{n+1} - x_1$ are linearly independent) then $\mathbf{conv}(S)$ is called a* simplex *with vertices $x_1, \ldots, x_{n+1}$.*

4 Closure and interior of sets

Many of the set-related results we will see in this course depends on the characteristics of the set itself. Often, assuming properties such as closedness or compactness considerably ease technical derivations.

4.1 Closure, interior and boundary of a set

Let us define some properties that will be useful in this course. For that, we will use an open ϵ-neighbourhood of $x \in \mathbb{R}^n$ (which is a norm ball of radius ϵ centred in x) defined as

$$N_\epsilon(x) = \{y : \|y - x\| < \epsilon\}.$$

Let $S \subseteq \mathbb{R}^n$ be an arbitrary set. We can use N_ϵ to define the following concepts related to S.

1. *Interior of* S: the interior of S, denoted $\mathbf{int}(S)$, is the set

   $$\mathbf{int}\, S = \{x \in S : N_\epsilon(x) \subset S \text{ for some } \epsilon > 0\}.$$

   If S is the same as its own interior, then we say that S is *open*. Some authors say that S is solid if it has a nonempty interior, that is, $\mathbf{int}(S) \neq \emptyset$. Notice that the interior of S is a subset of S, that is, $\mathbf{int}(S) \subseteq S$.

2. *Boundary of* S: the boundary of S, denoted $\mathbf{bou}(S)$, is defined by

   $$\mathbf{bou}(S) = \{x \in \mathbb{R}^n : N_\epsilon(x) \text{ contains some } y \in S \text{ and some } z \notin S \text{ for every } \epsilon > 0\}.$$

3. *Closure of* S: the closure of S, denoted $\mathbf{clo}(S)$, is defined as

   $$\mathbf{clo}(S) = \{x \in \mathbb{R}^n : S \cap N_\epsilon(x) \neq \emptyset \text{ for every } \epsilon > 0\}.$$

   Notice that the closure might contain points that do not belong to S. We say that a set is *closed* if $S = \mathbf{clo}(S)$, that is, the set itself is its own closure.

4. Boundedness: We say that S is bounded if $S \subset N_R(0)$ for some $R > 0$.

In our finite-dimensional setting, we say that a set is *compact* if it is both *closed* and *bounded*. Compact sets appear very frequently in real-world applications of optimisation, since typically one can assume the existence of bounds for decision variables (such as nonnegativity or maximum physical bounds or, at an extreme case, smallest/ largest computational constants). Another frequent example of bounded set is the convex hull of a collection of discrete points, which is called by some authors *polytopes* (effectively bounded polyhedral sets).

Let us consider the following example. Let $S = \{(x_1, x_2) \in \mathbb{R}^n : x_1^2 + x_2^2 \leq 1\}$. Then, we have that:

1. $\mathbf{clo}(S) = \{(x_1, x_2) \in \mathbb{R}^n : x_1^2 + x_2^2 \leq 1\}$. Since $S = \mathbf{clo}(S)$, S is closed.
2. $\mathbf{int}(S) = \{(x_1, x_2) \in \mathbb{R}^n : x_1^2 + x_2^2 < 1\}$.
3. $\mathbf{bou}(S) = \{(x_1, x_2) \in \mathbb{R}^n : x_1^2 + x_2^2 = 1\}$.
4. S is compact, since it is closed and, obviously, bounded.

Notice that, if S is closed, then $\mathbf{bou}(S) \subset S$. That is, its boundary is part of the set itself. Moreover, it can be shown that $\mathbf{clo}(S) = \mathbf{bou}(S) \cup S$ is the smallest closed set containing S.

In case S is convex, one can infer the convexity of the interior $\mathbf{int}(S)$ and its closure $\mathbf{clo}(S)$. The following theorem summarises this result.

Theorem 6. *Let $S \subseteq \mathbb{R}^n$ be a convex set with $\mathbf{int}(S) \neq \emptyset$. Let $x_1 \in \mathbf{clo}(S)$ and $x_2 \in \mathbf{int}(S)$. Then $x = \lambda x_1 + (1 - \lambda)x_2 \in \mathbf{int}(S)$ for all $\lambda \in (0, 1)$.*

Theorem 6 is useful for inferring the convexity of the elements related to S. We summarise the key results in the following corollary.

Corollary 7. *Let S be a convex set with $\mathbf{int}(S) \neq \emptyset$. Then*

1. *$\mathbf{int}(S)$ is convex;*
2. *$\mathbf{clo}(S)$ is convex;*
3. *$\mathbf{clo}(\mathbf{int}(S)) = \mathbf{clo}(S)$;*
4. *$\mathbf{int}(\mathbf{clo}(S)) = \mathbf{int}(S)$.*

4.2 The Weierstrass theorem

The Weierstrass theorem is a result that guarantees the existence of optimal solutions for optimisation problems. To make it more precise, let

$$(P): \quad \text{min. } \{f(x) : x \in S\}$$

be our optimisation problem. If an optimal solution x^* exists, then $f(x^*) \leq f(x)$ for all $x \in S$ and $z = f(x^*) = \min\{f(x) : x \in S\}$.

Notice the difference between min. (an abbreviation for minimise) and the operator $\min$. The first is meant to represent the problem of minimising the function f in the domain S, while $\min$ is shorthand for minimum, in this case z, assuming that it is attainable.

It might be that an optimal solution is not attainable, but a bound can be obtained for the optimal solution value. The greatest lower bound for z is its *infimum* (or *supremum* for maximisation problems), denoted by $\inf$. That is, if $z = \inf\{f(x) : x \in S\}$, then $z \leq f(x)$ for all $x \in S$ and there is no $\overline{z} > z$ such that $\overline{z} \leq f(x)$ for all $x \in S$. We might sometimes use the notation

$$(P): \ z = \inf\{f(x) : x \in S\}$$

to represent optimisation problems for which one cannot be sure whether an optimal solution is attainable. The Weierstrass theorem describes the situations in which those minimums (or maximums) are guaranteed to be attained, which is the case whenever S is compact.

Theorem 8 (Weierstrass theorem)**.** *Let $S \neq \emptyset$ be a compact set, and let $f : S \to \mathbb{R}$ be continuous on S. Then there exists a solution $\overline{x} \in S$ to min. $\{f(x) : x \in S\}$.*

Figure 7 illustrates three examples. In the first (on the left) the domain $[a, b]$ is compact, and thus the minimum of f is attained at b. In the other two, $[a, b)$ is open and therefore, Weierstrass theorem does not hold. In the middle example, one can obtain $\inf f$, which is not the case for the last example on the right.

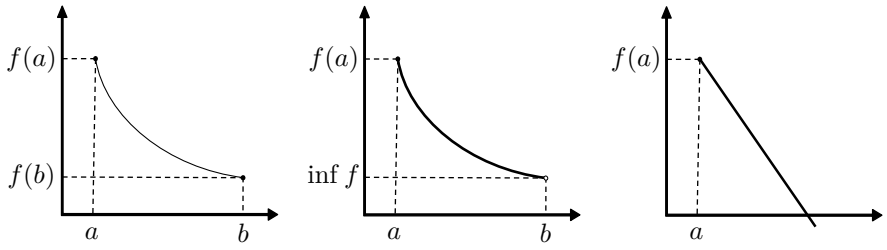


Figure 7: Examples of attainable minimum (left) and infimum (centre) and an example where neither are attainable (right).

5 Separation and support of sets

The concepts of *separation* and *support* of sets are key for establishing optimality conditions later in this course. We are interested in mechanisms that allow one to infer whether there exists hyperplanes separating points from sets (or sets from sets). We will also be interested in means to, given a point $x \notin S$, find the closest to point not belonging to S.

5.1 Hyperplanes and closest points

We start with how to identify closest points to sets.

Theorem 9 (Closest-point theorem)**.** *Let $S \neq \emptyset$ be a closed convex set in $\mathbb{R}^n$ and $y \notin S$. Then, there exists a unique point $\overline{x} \in S$ with minimum distance from y. In addition, $\overline{x}$ is the minimising point if and only if*

$$(y - \overline{x})^\top (x - \overline{x}) \leq 0, \text{ for all } x \in S$$

Simply put, if S is a closed convex set, then $\overline{x} \in S$ will be the closest point to $y \notin S$ if the vector $y - \overline{x}$ is such that if it forms an angle that is greater or equal than 90° with all other vectors $x - \overline{x}$ for $x \in S$. Figure 8 illustrates this logic.

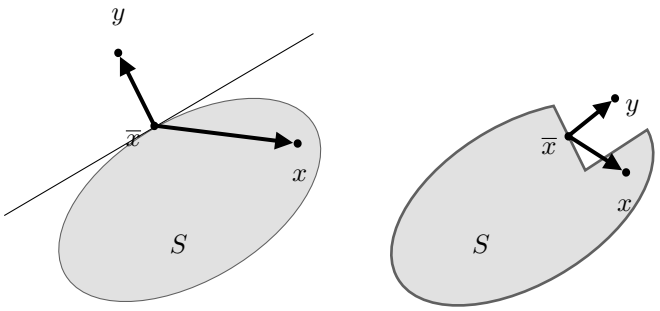


Figure 8: Closest-point theorem for a closed convex set (on the left). On the right, an illustration on how the absence of convexity invalidates the result.

Notice that S lies in the half-space $(y - \overline{x})^\top (x - \overline{x}) \leq 0$ defined by the hyperplane $p^\top (x - \overline{x}) = 0$ with normal vector $p = (y - \overline{x})$. We will next revise the concepts of half-spaces and hyperplanes, since they will play a central role in the derivations in this course.

5.2 Halfspaces and separation

We can use halfspaces to build the concept of separation. Let us start by recalling that a hyperplane $H = \{x : p^\top x = \alpha\}$ with normal vector $p \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$ defines two (closed) half-spaces $H^+ = \{x : p^\top x \geq \alpha\}$ and $H^- = \{x : p^\top x \leq \alpha\}$. Figure 9 illustrates the concept. Notice how the vector p points toward the half-space H^+.

Any hyperplane H can be defined in reference to a point $\overline{x} \in H$ by noticing that

$$p^\top (x - \overline{x}) = p^\top x - p^\top \overline{x} = \alpha - \alpha = 0.$$

From that, the half-spaces defined by H can be equivalently be given as $H^+ = \{x : p^\top (x - \overline{x}) \geq 0\}$ and $H^- = \{x : p^\top (x - \overline{x}) \leq 0\}$.

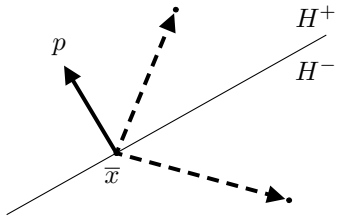


Figure 9: Normal vectors, hyperplane and halfspaces

We can now define the separation of convex sets.

Definition 10. *Let S_1 and S_2 be nonempty sets in $\mathbb{R}^n$. The hyperplane $H = \{x : p^\top x = \alpha\}$ is said to* separate *S_1 and S_2 if $p^\top x \geq \alpha$ for each $x \in S_1$ and $p^\top x \leq \alpha$ for each $x \in S_2$. In addition, the following apply:*

1. **Proper separation:** *$S_1 \cup S_2 \not\subset H$;*
2. **Strict separation:** *$p^\top x < \alpha$ for each $x \in S_1$ and $p^\top x > \alpha$ for each $x \in S_2$;*
3. **Strong separation:** *$p^\top x \geq \alpha + \epsilon$ for some $\epsilon > 0$ and $x \in S_1$, and $p^\top x \leq \alpha$ for each $x \in S_2$.*

Figure 10 illustrates the three types of separation in Definition 10. On the left, proper separation is illustrated, which is obtained by any hyperplane that does not contain both S_1 and S_2, but that might contain points from either or both. In the middle, sets S_1 and S_2 belong to two distinct half-spaces in a strict sense. On the right, strict separation holds with an additional margin $\epsilon > 0$, which is defined as strong separation.

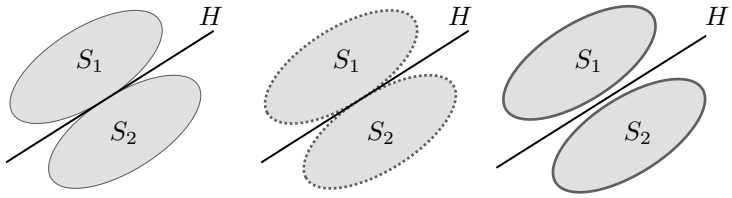


Figure 10: Three types of separation between S_1 and S_2.

A powerful yet simple result that we will use later is that, for a closed convex set S, there always exists a hyperplane separating S and a point y that does not belong to S.

Theorem 11 (Separation theorem)**.** *Let $S \neq \emptyset$ be a closed convex set in $\mathbb{R}^n$ and $y \notin S$. Then, there exists a nonzero vector $p \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$ such that $p^\top x \leq \alpha$ for each $x \in S$ and $p^\top y > \alpha$.*

Proof. Theorem 9 guarantees the existence of a unique minimising $\overline{x} \in S$ such that $(y - \overline{x})^\top (x - \overline{x}) \leq 0$ for each $x \in S$. Let $p = (y - \overline{x}) \neq 0$ and $\alpha = (y - \overline{x})^\top \overline{x} = p^\top \overline{x}$. Then we get $p^\top x \leq \alpha$ for each $x \in S$, while $p^\top y - \alpha = (y - \overline{x})^\top (y - \overline{x}) = \|y - \overline{x}\|^2 > 0$. □

This is the first proof we look at in these notes, and the reason for that is its importance in many of the results we will discuss further. The proof first looks at the problem of finding a minimum distance point as an optimisation problem and uses the Weierstrass theorem (in our finite-dimensional setting, Theorem 9 is a consequence of the Weierstrass theorem stated in Theorem 8) to guarantee that such a $\overline{x}$ exists. Being a minimum distance point, we know from Theorem 9 that $(y-\overline{x})^\top(x-\overline{x}) \leq 0$ holds. Now by defining p and α as in the proof, one might notice that

$$(y-\overline{x})^\top(x-\overline{x}) \leq 0 \;\Leftrightarrow\; (y-\overline{x})^\top x \leq (y-\overline{x})^\top \overline{x} \;\Leftrightarrow\; p^\top x \leq p^\top \overline{x} = \alpha.$$

The inequality $p^\top y > \alpha$ is demonstrated to hold in the final part by noticing that

$$\begin{aligned} p^\top y - \alpha &= (y-\overline{x})^\top y - \overline{x}^\top(y-\overline{x}) \\ &= y^\top(y-\overline{x}) - \overline{x}^\top(y-\overline{x}) \\ &= (y-\overline{x})^\top(y-\overline{x}) = \|y-\overline{x}\|^2 > 0. \end{aligned}$$

Theorem 11 has interesting consequences. For example, one can apply it to every point in the boundary $\mathbf{bou}(S)$ to show that S is formed by the intersection of all half-spaces containing S.

Another interesting result is the existence of strong separation. If $y \notin \mathbf{clo}(\mathbf{conv}(S))$, then one can show that strong separation between y and S exists since there will surely be a distance $\epsilon > 0$ between y and S.

5.3 Farkas' theorem

Farkas' theorem plays a central role in deriving optimality conditions. It can assume several alternative forms, which are typically referred to as Farkas' lemmas. In essence, the Farkas' theorem is used to demonstrate that a given system of linear equations has a solution if and only if a related system can be shown to have no solutions and vice-versa.

Theorem 12. *Let A be an $m \times n$ matrix and c be an n-vector. Then exactly one of the following two systems has a solution:*

$$\begin{aligned} &(1): \; Ax \leq 0,\; c^\top x > 0,\; x \in \mathbb{R}^n \\ &(2): \; A^\top y = c,\; y \geq 0,\; y \in \mathbb{R}^m. \end{aligned}$$

Proof. Suppose (2) has a solution. Let x be such that $Ax \leq 0$. Then $c^\top x = (A^\top y)^\top x = y^\top Ax \leq 0$. Hence, (1) has no solution.

Next, suppose (2) has no solution. Let $S = \{x \in \mathbb{R}^n : x = A^\top y,\; y \geq 0\}$. Notice that S is closed and convex and that $c \notin S$. By Theorem 11, there exists $p \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$ such that $p^\top c > \alpha$ and $p^\top x \leq \alpha$ for $x \in S$.

As $0 \in S$, $\alpha \geq 0$ and so $p^\top c > 0$. Also, $\alpha \geq p^\top A^\top y = y^\top Ap$ for all $y \geq 0$. This implies that $Ap \leq 0$, and thus p satisfies (1). □

The first part of the proof shows that, if we assume that system (2) has a solution, then $c^\top x > 0$ cannot hold for $y \geq 0$. The second part uses the separation theorem (Theorem 11) to show that c can be seen as a point not belonging to the closed convex set S for which there is a separation hyperplane and that the existence of such plane implies that system (1) must hold. The set S is closed and convex since it is a conic combination of rows a_i, $i = 1, \ldots, m$, of A. Using the fact that $0 \in S$, one can show that $\alpha \geq 0$. The last part uses the identity $p^\top A^\top = (Ap)^\top$ and the fact that $(Ap)^\top y = y^\top Ap$. Since the components of y can be arbitrarily large positive numbers and α is a nonnegative constant, $y^\top Ap \leq \alpha$ holds for all $y \geq 0$ if and only if $y^\top Ap \leq 0$ for all $y \geq 0$, requiring that $Ap \leq 0$. Indeed, if $\tilde{y}^\top Ap \geq \epsilon$ were true for some $\epsilon > 0$ and $\tilde{y} \geq 0$, then choosing $y = \beta\tilde{y} \geq 0$, $\beta > 0$, and letting $\beta \to \infty$ would lead to arbitrarily large values for $y^\top Ap \geq \beta\epsilon$, which would contradict $y^\top Ap \leq \alpha$.

Farkas' theorem has an interesting geometrical interpretation arising from this proof, as illustrated in Figure 11. Consider the cone C formed by the rows of A

$$C = \left\{c \in \mathbb{R}^n : c = A^\top y,\ y \geq 0\right\} = \left\{c \in \mathbb{R}^n : c_j = \sum_{i=1}^{m} a_{ij} y_i,\ j = 1, \ldots, n,\ y_i \geq 0,\ i = 1, \ldots, m\right\}.$$

The *polar cone* of C, denoted C^0, is formed by the all vectors having angles of 90° or more with all vectors in C. That is,

$$C^0 = \{x \in \mathbb{R}^n : c^\top x \leq 0\ \forall c \in C\} = \{x : Ax \leq 0\}.$$

Notice that (1) has a solution if the intersection between the polar cone C^0 and the positive half-space $H^+ = \left\{x \in \mathbb{R}^n : c^\top x > 0\right\}$ is not empty. If (2) has a solution, as in the beginning of the proof, then $c \in C$ and the intersection $C^0 \cap H^+ = \emptyset$. Now, if (2) does not have a solution, that is, $c \notin C$, then one can see that $C^0 \cap H^+$ cannot be empty, meaning that (1) has a solution.

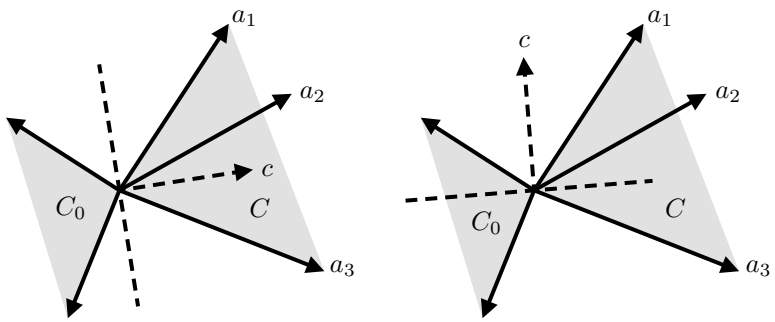


Figure 11: Geometrical illustration of the Farkas' theorem. On the left, system (2) has a solution, while on the right, system (1) has a solution.

5.4 Supporting hyperplanes

There is an important connection between the existence of hyperplanes that support a whole set and optimality conditions of points. Let us first define supporting hyperplanes.

Definition 13 (Supporting hyperplane)**.** *Let $S \neq \emptyset$ be a set in $\mathbb{R}^n$, and let $\overline{x} \in \mathbf{bou}(S)$. $H = \left\{x \in \mathbb{R}^n : p^\top(x - \overline{x}) = 0\right\}$ is a supporting hyperplane of S at $\overline{x}$ if either $S \subseteq H^+$ (i.e., $p^\top(x-\overline{x}) \geq 0$ for $x \in S$) or $S \subseteq H^-$.*

Figure 12 illustrates the concept of supporting hyperplanes. Notice that supporting hyperplanes might not be unique, with the geometry of the set S playing an important role in that matter.

Let us define the function $f(x) = p^\top x$ with $x \in S$. One can see that the optimal solution $\overline{x}$ given by

$$\overline{x} = \underset{x \in S}{\operatorname{argmax}}\, f(x)$$

is a point $x \in S$ for which p is a supporting hyperplane. A simple geometric analogy is to think that the f increases value as one moves in the direction of p. The constraint $x \in S$ will eventually prevent the movement further from S and this last contact point is precisely $\overline{x}$. This is a useful concept for optimising problem using gradients of functions, as we will discuss later in the course.

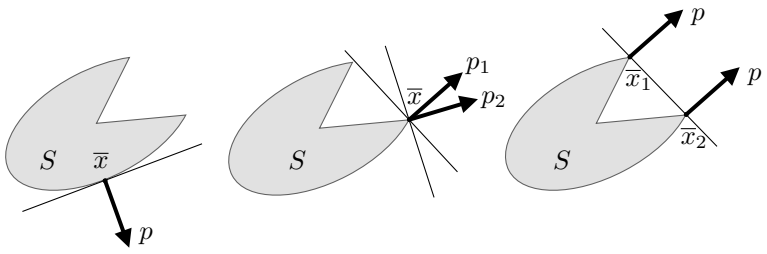


Figure 12: Supporting hyperplanes for an arbitrary set. Notice how a single point might have multiple supporting planes (middle) or different points might have the same supporting hyperplane (right)

One characteristic that convex sets present that will be of great importance when establishing optimality conditions is the existence of supporting hyperplanes at every boundary point.

Theorem 14 (Support of convex sets)**.** *Let $S \neq \emptyset$ be a convex set in $\mathbb{R}^n$, and let $\overline{x} \in \mathbf{bou}(S)$. Then there exists $p \neq 0$ such that $p^\top(x - \overline{x}) \leq 0$ for each $x \in \mathbf{clo}(S)$.*

The proof follows straightforwardly from Theorem 11. Figure 13 provides an illustration of the theorem.

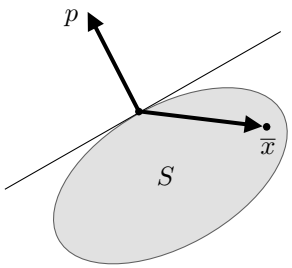


Figure 13: Supporting hyperplanes for convex sets. Notice how every boundary point has at least one supporting hyperplane

Lecture 3 - Convex functions

Fabricio Oliveira (with small modifications by Nuutti Hyvönen)

Last update: September 22, 2022

Abstract. In this lecture, we focus on the convexity of functions. First, we define what is a convex function and discuss examples. We also discuss the connection between convex function, lower level sets and epigraphs. Next, we develop the notion of subgradients, in connection to supporting hyperplanes. We also discuss the notion of gradients as being unique subgradients for differentiable functions. Second order differentiability is also considered as a mechanism to determine the convexity of functions. We finalise discussing the notion of quasiconvexity and how some nonconvex functions can be still present convex (sub)level sets, which can be exploited for developing optimisation methods.

Outline of this lecture

1 Convexity in functions

Now we turn our attention to identifying the convexity of functions. Consider the general problem

$$\begin{aligned}(P): \quad & \text{min. } f(x) \\ & \text{subject to: } g(x) \leq 0 \\ & x \in X\end{aligned}$$

with $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$ and $X \subseteq \mathbb{R}^n$. Assuming X is a convex set, the next step towards attesting that (P) is a convex problem is to check whether f and g are convex. It is important to emphasise (perhaps redundantly at this point) how crucial is for us to be able to attest the convexity (P), since it allows us to generalise local optimality results to the whole domain of the problem.

The convexity of functions has a different definition than that used to define convex sets.

Definition 1 (Convexity of a function I)**.** *Let $f : S \to \mathbb{R}$ where $S \subseteq \mathbb{R}^n$ is a nonempty convex set. The function f is said to be* convex *on S if*

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$

for each $x_1, x_2 \in S$ and for each $\lambda \in [0, 1]$.

Very often, we use the term convex to loosely refer to concave functions, which must be done with caution. In fact, if f is convex, than $-f$ is concave and we say that (P) is a convex problem even if f is concave and we seek to maximise f instead. Also, linear functions are both convex and concave.

We say that a convex function is *strictly convex* if the inequality holds strictly in Definition 1 for each $\lambda \in (0, 1)$ (notice the open interval instead). In practice, it means that the function is guaranteed to not present flatness around its minimum (or maximum, for concave functions).

1.1 Example of convex functions

Some examples of convex function are:

1. $f(x) = a^\top x + b$;
2. $f(x) = e^x$;
3. $f(x) = x^p$ on $\mathbb{R}_+$ for $p \leq 0$ or $p \geq 1$; concave for $0 \leq p \leq 1$.
4. $f(x) = \|x\|_p$, $p \geq 0$ (p-norm);
5. $f(x) = -\log x$ and negative entropy $f(x) = -x \log x$ are concave;

6. $f(x) = \max\{x_1, \ldots, x_n\}$.

Knowing that these common functions are convex is helpful for identifying convexity in more complex functions formed by *composition*. By knowing that an operation between functions preserves convexity, we can infer the convexity of more complicated functions. The following are convexity preserving operations.

1. Let $f_1, \ldots, f_k : \mathbb{R}^n \to \mathbb{R}$ be convex. Then these are convex:
   - $f(x) = \sum_{j=1}^{k} \alpha_j f_j(x)$ where $\alpha_j \geq 0$ for $j = 1, \ldots, k$;
   - $f(x) = \max\{f_1(x), \ldots, f_k(x)\}$;
2. $f(x) = \frac{1}{g(x)}$ on S, where $g : \mathbb{R}^n \to \mathbb{R}$ is concave and $S = \{x : g(x) > 0\}$;
3. $f(x) = g(h(x))$, where $g : \mathbb{R} \to \mathbb{R}$ is a nondecreasing convex function and $h : \mathbb{R}^n \to \mathbb{R}$ is convex.
4. $f(x) = g(h(x))$, where $g : \mathbb{R}^m \to \mathbb{R}$ is convex and $h : \mathbb{R}^n \to \mathbb{R}^m$ is affine: $h(x) = Ax + b$ with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

1.2 Convex functions and their level sets

There is a strong connection between convexity of sets and the convexity of functions. Let us first consider *level sets*, which is one type of set spawned by functions.

Definition 2 (Lower level set)**.** *Let $S \subseteq \mathbb{R}^n$ be a nonempty set. The lower level set of $f : \mathbb{R}^n \to \mathbb{R}$ for $\alpha \in \mathbb{R}$ is given by*

$$S_\alpha = \{x \in S : f(x) \leq \alpha\}.$$

Figure 1 illustrates the lower level sets of two functions. The lower level set S_α can be seen as the projection of the function image onto the domain for a given level α.

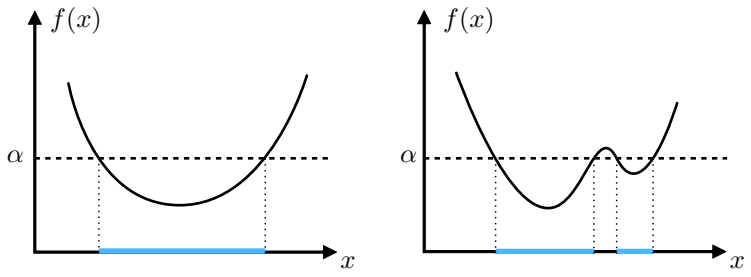


Figure 1: The lower level sets S_α (in blue) of two functions, given a value of α. Notice the nonconvexity of the level set of the nonconvex function (on the right)

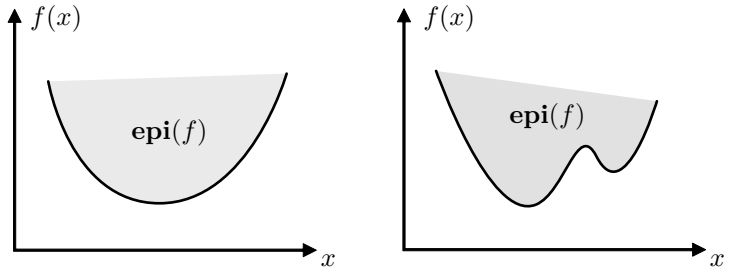


Figure 2: The epigraph $\mathbf{epi}(f)$ of a convex function is a convex set (in grey on the left).

Notice that, for convex functions, no discontinuity can be observed, making S_α convex. Lemma 3 states this property.

Lemma 3. *Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set and $f : S \to \mathbb{R}$ a convex function. Then, any level set S_α with $\alpha \in \mathbb{R}$ is convex.*

Proof. Let $x_1, x_2 \in S_\alpha$. Thus, $x_1, x_2 \in S$ with $f(x_1) \leq \alpha$ and $f(x_2) \leq \alpha$. Let $\lambda \in (0, 1)$ and $x = \lambda x_1 + (1 - \lambda)x_2$. Since S is convex, we have that $x \in S$. Now, by the convexity of f, we have

$$f(x) \leq \lambda f(x_1) + (1 - \lambda) f(x_2) \leq \lambda\alpha + (1 - \lambda)\alpha = \alpha$$

and thus $x \in S_\alpha$. □

Remark: notice that a convex lower level set does not necessarily mean that the function is convex. In fact, as we will see later, there are nonconvex functions that have convex level sets (the so-called quasiconvex functions).

1.3 Convex functions and their epigraphs

Epigraphs, on the other hand, can be used to show the convexity of functions. Let us first formally define epigraphs.

Definition 4 (Epigraph)**.** *Let $S \subseteq \mathbb{R}^n$ be a nonempty set and $f : S \to \mathbb{R}$. The epigraph of f is*

$$\mathbf{epi}(f) = \{(x, y) : x \in S,\ y \in \mathbb{R},\ y \geq f(x)\} \subseteq \mathbb{R}^{n+1}$$

Figure 2 illustrates the epigraphs of two functions. Notice that the second function (on the right) is not convex, and nor is its epigraph. In fact, we can use the convexity of epigraphs (and the technical results associated with the convexity of sets) to show the convexity of functions.

Theorem 5 (Convex epigraphs)**.** *Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set and $f : S \to \mathbb{R}$. Then f is convex if and only if $\mathbf{epi}(f)$ is a convex set.*

Proof. First, suppose f is convex and let $(x_1, y_1), (x_2, y_2) \in \mathbf{epi}(f)$ and $\lambda \in [0, 1]$. Then

$$\lambda y_1 + (1-\lambda)y_2 \geq \lambda f(x_1) + (1-\lambda)f(x_2) \geq f(\lambda x_1 + (1-\lambda)x_2).$$

As $\lambda x_1 + (1-\lambda)x_2 \in S$, $(\lambda x_1 + (1-\lambda)x_2, \lambda y_1 + (1-\lambda)y_2) \in \mathbf{epi}(f)$, i.e., $\mathbf{epi}(f)$ is convex.

Conversely, suppose $\mathbf{epi}(f)$ is convex. By the definition of an epigraph, for any $x_1, x_2 \in S$, it holds that $(x_1, f(x_1)) \in \mathbf{epi}(f)$, $(x_2, f(x_2)) \in \mathbf{epi}(f)$. Due to the assumed convexity of $\mathbf{epi}(f)$, $(\lambda x_1 + (1-\lambda)x_2, \lambda f(x_1) + (1-\lambda)f(x_2)) \in \mathbf{epi}(f)$ for any $\lambda \in [0, 1]$, implying that $\lambda f(x_1) + (1-\lambda)f(x_2) \geq f(\lambda x_1 + (1-\lambda)x_2)$, and thus f is convex. □

The proof starts with the implication "if f is convex, then $\mathbf{epi}(f)$ is convex". For that, it assumes that f is convex and uses it to show that any convex combination of (x_1, y_1), and (x_2, y_2) in $\mathbf{epi}(f) \subset \mathbb{R}^{n+1}$ will also be in $\mathbf{epi}(f)$, which is the definition of a convex set.

To prove the implication "if $\mathbf{epi}(f)$ is convex, then f is convex", we define a convex combination of points in $\mathbf{epi}(f)$ and use the definition of $\mathbf{epi}(f)$ to show that f is convex by setting $y = \lambda f(x_1) + (1-\lambda)f(x_2)$ and $x = \lambda x_1 + (1-\lambda)x_2$.

2 Differentiability of functions

2.1 Subgradients and supporting hyperplanes

Subgradients can be understood as supporting hyperplanes at the boundary of function epigraphs. They can be seen as first-order local approximations of the function, which is often helpful information for optimisation methods when searching for directions of improvement.

Definition 6 (Subgradients)**.** *Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set and $f : S \to \mathbb{R}$ a convex function. Then $\xi \in \mathbb{R}^n$ is a* subgradient *of f at $\overline{x} \in S$ if*

$$f(x) \geq f(\overline{x}) + \xi^\top (x - \overline{x}) \tag{1}$$

for all $x \in S$.

Inequality (1) is called the *subgradient inequality* and is going to be useful in several contexts later in this course. The set of subgradients ξ of f at $\overline{x} \in S$ is the *subdifferential*

$$\partial_f(\overline{x}) = \left\{\xi \in \mathbb{R}^n : f(x) \geq f(\overline{x}) + \xi^\top (x - \overline{x}),\ \forall x \in S\right\}.$$

Every convex function $f : S \to \mathbb{R}$ has at least one subgradient at any point $\overline{x}$ in the interior of the

convex set S. Requiring that $\overline{x} \in \mathbf{int}(S)$ allows us to disregard boundary points of S where $\partial_f(\overline{x})$ might be empty. Theorem 7 presents this result.

Theorem 7. *Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set and $f : S \to \mathbb{R}$ a convex function. Then for every $\overline{x} \in \mathbf{int}(S)$, there exists $\xi_{\overline{x}} \in \mathbb{R}^n$ such that*

$$H = \left\{(x,y) : y = f(\overline{x}) + \xi_{\overline{x}}^\top (x - \overline{x})\right\} = \left\{ (x,y) : \begin{bmatrix} \xi_{\overline{x}} \\ -1 \end{bmatrix}^\top \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \overline{x} \\ f(\overline{x}) \end{bmatrix} \right) = 0 \right\}$$

supports $\mathbf{epi}(f)$ at $(\overline{x}, f(\overline{x}))$. In particular,

$$f(x) \geq f(\overline{x}) + \xi_{\overline{x}}^\top (x - \overline{x}),\ \forall x \in S, \tag{2}$$

i.e., $\xi_{\overline{x}}$ is a subgradient for f at $\overline{x}$.

The proof consists of applying Theorem 5 and then using the support of convex sets theorem (Theorem 14 in Lecture 2) to show that the $\mathbf{epi}(f)$ has a supporting hyperplane at $(\overline{x}, f(\overline{x})) \in \mathbf{bou}(\mathbf{epi}(f))$. In particular, the role of the normal vector in Theorem 14 of Lecture 2 is played by

$$p = \begin{bmatrix} \xi_{\overline{x}} \\ -1 \end{bmatrix} \in \mathbb{R}^{n+1},$$

which has a negative component in the direction of the y-coordinate, and thus the (epi)graph of f lies in the corresponding 'negative halfspace'. This explains the direction of the inequality in (2). Notice also that one has the freedom to scale the normal vector defining a hyperplane, which explains why the last component of p can be assumed to be of absolute value 1.

2.2 Differentiability and gradients for convex functions

Let us first define differentiability of a function.

Definition 8. *Let $S \subseteq \mathbb{R}^n$ be a nonempty set. The function $f : S \to \mathbb{R}$ is differentiable at $\overline{x} \in \mathbf{int}(S)$ if there exists a vector $\nabla f(\overline{x})$, called a gradient vector, and a function $\alpha(\overline{x}; \cdot) : \mathbb{R}^n \to \mathbb{R}$ such that*

$$f(x) = f(\overline{x}) + \nabla f(\overline{x})^\top (x - \overline{x}) + \|x - \overline{x}\|\, \alpha(\overline{x}; x - \overline{x})$$

where $\lim_{x \to \overline{x}} \alpha(\overline{x}; x - \overline{x}) = 0$. If this is the case for all $\overline{x} \in \mathbf{int}(S)$, we say that the function is differentiable in S.

Notice that this definition is equivalent to the existence of a first-order (Taylor series) expansion, with an error term defined by α. It is straightforward to prove that the gradient vector at a given point is unique: If there were two different gradient vectors, by equating the corresponding first-order

expansions, one could easily show that one of the two error terms could not converge to zero as required by the definition. The components of the gradient are called the *partial derivatives* of f and denoted as $\frac{\partial f}{\partial x_i}$, $i = 1, \ldots, n$.

If f is differentiable in S, then its subdifferential $\partial(x)$ is a singleton (a set with a single element) for all $x \in S$. This is shown in Lemma 9

Lemma 9. *Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set and $f : S \to \mathbb{R}$ a convex function. Suppose that f is differentiable at $\overline{x} \in \mathbf{int}(S)$. Then $\partial_f(\overline{x}) = \{\nabla f(\overline{x})\}$, i.e., the subdifferential $\partial_f(\overline{x})$ is a singleton with $\nabla f(\overline{x})$ as its unique element.*

Proof. From Theorem 7, $\partial f(\overline{x}) \neq \emptyset$. Moreover, combining the existence of a subgradient ξ and differentiability of f at $\overline{x}$, we obtain for $d \in \mathbb{R}^n$ and a small enough $\lambda > 0$ that (Theorem 7 and Definition 8 with $\lambda d = x - \overline{x}$)

$$f(\overline{x} + \lambda d) \geq f(\overline{x}) + \lambda \xi^\top d, \tag{3}$$

$$f(\overline{x} + \lambda d) = f(\overline{x}) + \lambda \nabla f(\overline{x})^\top d + \lambda \|d\| \, \alpha(\overline{x}; \lambda d). \tag{4}$$

Subtracting (4) from (3), we get $0 \geq \lambda(\xi - \nabla f(\overline{x}))^\top d - \lambda \|d\| \alpha(\overline{x}; \lambda d)$. Dividing by $\lambda > 0$ and letting $\lambda \to 0^+$, we obtain $(\xi - \nabla f(\overline{x}))^\top d \leq 0$. Now, by setting $d = \xi - \nabla f(\overline{x})$, it becomes clear that $\xi = \nabla f(\overline{x})$. □

Notice that in the proof we use $\overline{x} + \lambda d$ to indicate that x is in direction d from $\overline{x}$, scaled by $\lambda > 0$. The fact that $\partial_f(x)$ is a singleton comes from the uniqueness of the solution for $(\xi - \nabla f(\overline{x}))^\top (\xi - \nabla f(\overline{x})) = 0$.

Figure 3 illustrates subdifferential sets for three distinct points of a piecewise linear function. The picture schematically represents a multidimensional space x as a one-dimensional projection (you can imagine this picture as being a section in one of the x dimensions). For the points in which the function is not differentiable, the subdifferential set contains an infinite number of subgradients. At points in which the function is differentiable (any mid-segment point) the subgradient is unique (a gradient) and the subdifferential is a singleton.

If $f : S \to \mathbb{R}$ is a convex differentiable function, then Theorem 7 can be combined with Lemma 9 to express one of the most powerful results relating f and its affine (first-order) approximation at $\overline{x}$.

Theorem 10 (Convexity of a function II)**.** *Let $S \subseteq \mathbb{R}^n$ be a nonempty convex open set, and let $f : S \to \mathbb{R}$ be differentiable on S. The function f is convex if and only if for any $\overline{x} \in S$, we have*

$$f(x) \geq f(\overline{x}) + \nabla f(\overline{x})^\top (x - \overline{x}), \ \forall x \in S. \tag{5}$$

Proof. Since $S = \mathbf{int}(S)$ is open, the fact that (5) is satisfied by any differentiable convex function follows by combining Theorem 7 and Lemma 9.

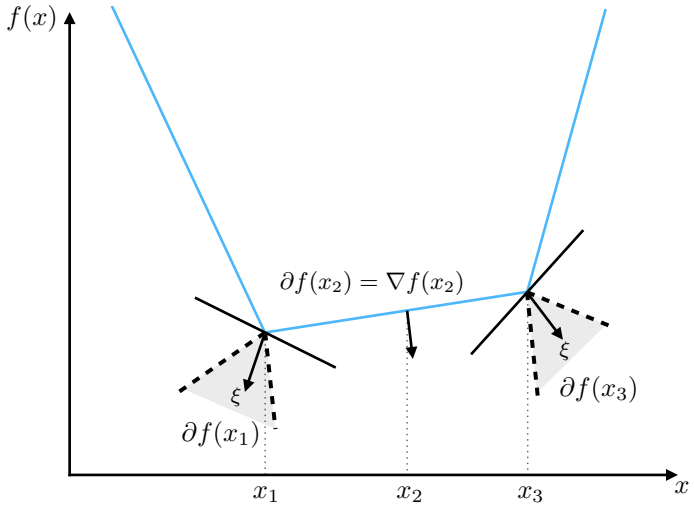


Figure 3: A representation of the subdifferential (in grey) for nondifferentiable (x_1 and x_3) and differentiable (x_2) points. The depicted vectors are not subgradients but the corresponding normal vectors $(\xi, -1)$ of tangent planes supporting $\mathbf{epi}(f)$ at the considered points; see Theorem 7.

To prove the other direction, i.e., that (5) guaranteees a (differentiable) $f : S \to \mathbb{R}^n$ is convex, we proceed as follows: Let $x_1, x_2 \in S$ be arbitrary and note that $\overline{x} = \lambda x_1 + (1-\lambda)x_2$, $\lambda \in [0,1]$, also belongs to S due to its convexity. Let $\xi = \nabla f(\overline{x})$ and notice that for $x = x_1$ (5) gives

$$f(x_1) \geq f(\overline{x}) + (1-\lambda)\xi^\top (x_1 - x_2), \tag{6}$$

and for $x = x_2$, it yields

$$f(x_2) \geq f(\overline{x}) + \lambda\xi^\top (x_2 - x_1). \tag{7}$$

Multiplying (6) by λ and (7) by $(1-\lambda)$, and adding them together leads to

$$\lambda f(x_1) + (1-\lambda) f(x_2) \geq f(\overline{x}) = f(\lambda x_1 + (1-\lambda)x_2),$$

which is means that f is convex. □

The general idea corresponding to Theorem 10 can also be formulated without assuming differentiability by resorting to subgradients.

Corollary 11 (Convexity of a function III)**.** *Let $S \subseteq \mathbb{R}^n$ be a nonempty convex open set, and let $f : S \to \mathbb{R}$. The function f is convex if and only if it has a subgradient at every $\overline{x} \in S$, that is, for every $\overline{x} \in S$, there exists $\xi_{\overline{x}} \in \mathbb{R}^n$ such that*

$$f(x) \geq f(\overline{x}) + \xi_{\overline{x}}^\top (x - \overline{x}), \ \forall x \in S. \tag{8}$$

Proof. The fact that (8) is satisfied by any convex function follows from Theorem 7 since $S = \mathbf{int}(S)$

is open. The other direction can be proven in exactly the same way as in the proof of Theorem 10 by choosing $\xi = \xi_{\overline{x}}$ instead of $\xi = \nabla f(\overline{x})$. □

2.3 Second-order differentiability

We say that a function is *twice-differentiable* if it has a second-order Taylor expansion. Having second-order expansions can be useful in that it allows for encoding curvature information in the approximation, which is characterised by the *Hessian*, and to verify convexity (or strict convexity) by testing for postive semi-definiteness (or positive definiteness).

Second-order differentiability can be defined as follows.

Definition 12 (Second-order differentiability)**.** *Let $S \subseteq \mathbb{R}^n$ be a nonempty set, and let $f : S \to \mathbb{R}$. Then f is twice differentiable at $\overline{x} \in \mathbf{int}(S)$ if there exists a vector $\nabla f(\overline{x}) \in \mathbb{R}^n$, an $n \times n$ matrix $H(\overline{x})$ (the Hessian), and a function $\alpha(\overline{x}; \cdot) : \mathbb{R}^n \to \mathbb{R}$ such that*

$$f(x) = f(\overline{x}) + \nabla f(\overline{x})^\top (x - \overline{x}) + \frac{1}{2}(x - \overline{x})^\top H(\overline{x})(x - \overline{x}) + \|x - \overline{x}\|^2 \alpha(\overline{x}; x - \overline{x})$$

where $\lim_{x \to \overline{x}} \alpha(\overline{x}; x - \overline{x}) = 0$. If this is the case for all $\overline{x} \in \mathbf{int}(S)$, we say that the function is twice differentiable in S.

The Hessian matrix $H(\overline{x})$ at $\overline{x}$ is written elementwise as

$$H(\overline{x}) = \begin{bmatrix} f_{11}(\overline{x}) & \dots & f_{1n}(\overline{x}) \\ \vdots & \ddots & \vdots \\ f_{n1}(\overline{x}) & \dots & f_{nn}(\overline{x}) \end{bmatrix},$$

where $f_{ij}(\overline{x}) = \frac{\partial^2 f(\overline{x})}{\partial x_i \partial x_j}$ are *the second partial derivatives* of f at $\overline{x}$. It is known that the Hessian $H(\overline{x})$ must be symmetric at $\overline{x}$ if the second partial derivatives are continuous in some open neighborhood of $\overline{x}$.

We say that $H(\overline{x})$, or any matrix, is *positive semi-definite* (PSD) if $x^\top H(\overline{x}) x \geq 0$ for $x \in \mathbb{R}^n$. Having a positive semi-definite Hessian for all $x \in S$ implies that the function is convex in S.

Theorem 13. *Let $S \subseteq \mathbb{R}^n$ be a nonempty convex open set, and let $f : S \to \mathbb{R}$ be twice differentiable on S. Then f is convex if and only if the Hessian matrix is positive semidefinite at each point in S.*

Proof. Suppose f is convex, pick $\overline{x} \in S$ and let $x \in \mathbb{R}^n$ be arbitrary. Since S is open, $\overline{x} + \lambda x \in S$ if $|\lambda| \neq 0$ is small enough. From Theorem 10 and the twice-differentiability of f, we have

$$f(\overline{x} + \lambda x) \geq f(\overline{x}) + \lambda \nabla f(\overline{x})^\top x \tag{9}$$

$$f(\overline{x} + \lambda x) = f(\overline{x}) + \lambda \nabla f(\overline{x})^\top x + \frac{1}{2}\lambda^2 x^\top H(\overline{x}) x + \lambda^2 \|x\|^2 \alpha(\overline{x}; \lambda x) \tag{10}$$

Subtracting (9) from (10), we get $\frac{1}{2}\lambda^2 x^\top H(\overline{x})x + \lambda^2\|x\|^2\alpha(\overline{x};\lambda x) \geq 0$. Diving by $\lambda^2 > 0$ and letting $\lambda \to 0$, it follows that $x^\top H(\overline{x})x \geq 0$.

Conversely, assume that $H(\hat{x})$ is positive semidefinite for all $\hat{x} \in S$. Let $\overline{x}, x \in S$ be arbitrary. Using the mean value theorem and the second-order expansion, one can show that

$$f(x) = f(\overline{x}) + \nabla f(\overline{x})^\top (x - \overline{x}) + \frac{1}{2}(x - \overline{x})^\top H(\hat{x})(x - \overline{x}),$$

where $\hat{x} = \lambda\overline{x} + (1 - \lambda)x$ for some $\lambda \in (0, 1)$ that depends on $\overline{x}$ and x. Note that $\hat{x} \in S$ by convexity, and $H(\hat{x})$ is positive semidefinite by assumption. Thus, $(x - \overline{x})^\top H(\hat{x})(x - \overline{x}) \geq 0$, implying $f(x) = f(\overline{x}) + \nabla f(\overline{x})^\top (x - \overline{x}) \geq 0$, which by Theorem 10 means that f is convex. □

The proof uses a trick we have seen before. First, we assume convexity and use the definition of convexity provided by Theorem 10 combined with an alternative definition for $(x - \overline{x})$ to show that $x^\top H(\overline{x})x \geq 0$. That is, instead of using the reference points x and $\overline{x}$, we incorporate a step size λ from $\overline{x}$ in the direction of x.

To show the other direction of implication, that is, that $x^\top H(\overline{x})x \geq 0$ implies convexity, we use the *mean value theorem*. The mean value theorem states that there must exist a point $\hat{x}$ between x and $\overline{x}$ for which the second order approximation is exact. From these, we can derive the definition of convexity, as in Theorem 10.

Checking for positive semi-definiteness can be done computationally even for large symmetric matrices, but one should be aware that in a floating point arithmetic such computations are never exact. Computing all eigenvalues of the investigated matrix and checking whether they are nonnegative is the straightforward choice, yet typically not the most efficient one. For more information on these kinds of computational considerations for matrices, we refer to courses on numerical matrix computation (e.g., MS-E1651 at the Aalto University). Some nonlinear optimisation solvers are capable of returning warning messages (or errors even) pointing out a lack of convexity by testing (under a certain threshold) for positive semi-definiteness.

3 Quasiconvexity

Quasiconvexity can be seen as the generalisation of convexity to a wider class of functions that are not convex, but share similar properties that allow for defining global optimality conditions. One class of such functions are named *quasiconvex*. Let us first technically define quasiconvex functions.

Definition 14 (quasiconvex functions)**.** *Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set and $f : S \to \mathbb{R}$. Function f is quasiconvex if, for each $x_1, x_2 \in S$ and $\lambda \in [0, 1]$, we have*

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \max\{f(x_1), f(x_2)\}. \tag{11}$$

A function f is quasiconcave if $-f$ is quasiconvex. A function is quasilinear if it is both quasiconvex and quasiconcave.

Figure 4 illustrates a quasiconvex function. Notice that, for any pair of points x_1 and x_2 in the domain of f, the graph of the function is always below the maximum between $f(x_1)$ and $f(x_2)$. This is precisely what renders convex the lower level sets of quasiconvex functions. Notice that, on the other hand, the epigraph $\mathbf{epi}(f)$ is not a convex set.

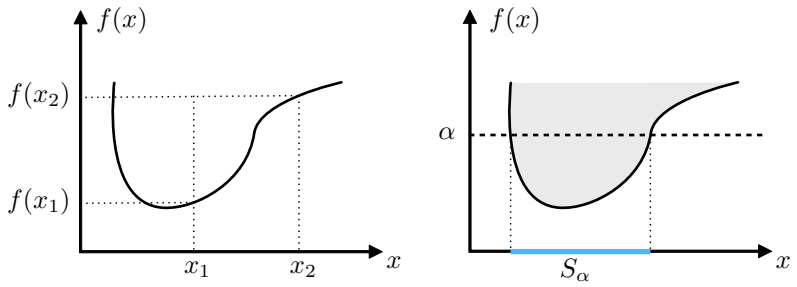


Figure 4: A quasiconvex function with its epigraph (in grey) and lower level set (in blue).

Examples of quasiconvex, quasiconcave and quasilinear functions include:

1. $f(x) = \sqrt{\|x\|_1}$ is quasiconvex.
2. Any monotone function, e.g., $f(x) = \log x$ for $x > 0$, is quasilinear.
3. $f(x) = \inf\{z \in \mathbb{Z} : z \geq x\}$ is quasilinear.
4. $f(x_1, x_2) = x_1 x_2$ is quasiconcave on $S = \{(x_1, x_2) \in \mathbb{R}^2 : x_1, x_2 > 0\}$.
5. $f(x_1, x_2) = \log(x_1^2 + x_2^2)$ is quasiconvex.

An important property of quasiconvex functions is that their level sets are convex.

Theorem 15. *Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set and $f : S \to \mathbb{R}$. Function f is quasiconvex if and only if $S_\alpha = \{x \in S : f(x) \leq \alpha\}$ is convex for all $\alpha \in \mathbb{R}$.*

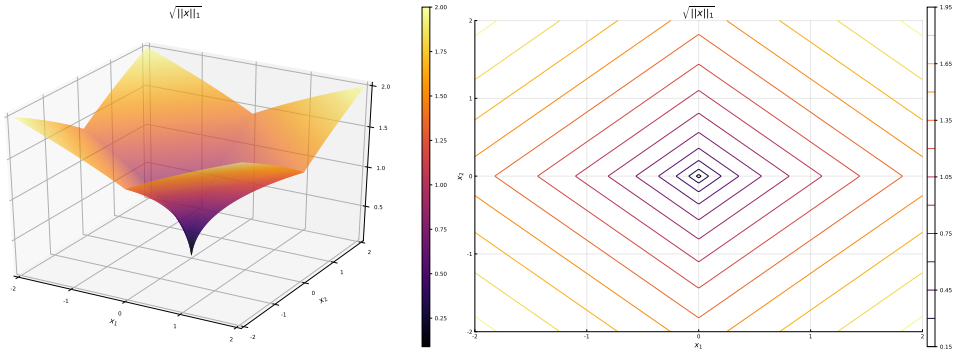
Proof. Suppose f is quasiconvex and let $x_1, x_2 \in S_\alpha$. Thus, $x_1, x_2 \in S$ and $\max\{f(x_1), f(x_2)\} \leq \alpha$. Let $x = \lambda x_1 + (1-\lambda)x_2$ for $\lambda \in [0,1]$. As S is convex, $x \in S$. By quasiconvexity of f, $f(x) \leq \max\{f(x_1), f(x_2)\} \leq \alpha$. Hence, $x \in S_\alpha$ and S_α is convex.

Conversely, assume that S_α is convex for any $\alpha \in \mathbb{R}$. Let $x_1, x_2 \in S$, and let $x = \lambda x_1 + (1-\lambda)x_2$ for $\lambda \in [0,1]$. Note that, for $\alpha = \max\{f(x_1), f(x_2)\}$, we have $x_1, x_2 \in S_\alpha$. The convexity of S_α implies that $x \in S_\alpha$, and thus $f(x) \leq \alpha = \max\{f(x_1), f(x_2)\}$, which implies that f is quasiconvex. □

The proof relies on the convexity of the domain S to show that a convex combination from point in the level set S_α also belongs to S_α. To show the other way around, we simply need to define $\alpha = \max\{f(x_1), f(x_2)\}$ to see that a convex level set S_α implies that f is quasiconvex.

Quasiconvex functions have an interesting first-order condition that arises from the convexity of its level sets.

Theorem 16. *Let $S \subseteq \mathbb{R}^n$ be a nonempty open convex set, and let $f : S \to \mathbb{R}$ be differentiable on S. Then f is quasiconvex if and only if, for $x_1, x_2 \in S$ and $f(x_1) \leq f(x_2)$, $\nabla f(x_2)^\top (x_1 - x_2) \leq 0$.*



(a) surface plot (b) level curves

Figure 5: Surface plot and level curves for $f(x) = \sqrt{\|x\|_1}$

The condition in Theorem 16 is in fact sufficient for global optimality if one can show that f is in fact *strictly quasiconvex*, that is when (11) holds strictly for any $\lambda \in (0, 1)$. Figure 5a and 5b show an example of a strict quasiconvex function and its level curves, illustrating that, despite the lack of convexity, the level sets are convex.

Strictly quasiconvex functions is a subset of a more general class of functions named *pseudoconvex*, for which the conditions in Theorem 16 are sufficient for global optimality.

Lecture 4 - Optimality conditions

Fabricio Oliveira (with small modifications by Nuutti Hyvönen)

Last update: September 27, 2022

Abstract. In this lecture, we use the concepts of convexity and differentiability to derive optimality conditions. We start by looking into how the presence of convexity can be used to derive general optimality conditions for unconstrained (and also constrained, which we will consider later in the course) problems. Next, we use the more general and easier to verify concept of differentiability to derive necessary and sufficient local optimality conditions and conclude showing how convexity can be used to show that first-order optimality conditions are in fact necessary and sufficient for global optimality.

Outline of this lecture

1 Recognising optimality

We now turn our focus to recognising whether a given point satisfies necessary and/or sufficient conditions for optimality. Even though these conditions can be used to test if a candidate point is optimal for a problem, its most important use is serving as a framework for directing solution methods in their search for optimal solutions.

Before we proceed, let us define the terminology we will use to refer to solutions. Let $f : \mathbb{R}^n \to \mathbb{R}$. Consider the problem (P) : min. $\{f(x) : x \in S\}$.

1. a *feasible solution* is any $\overline{x} \in S$;
2. a *local optimal solution* is a feasible solution $\overline{x} \in S$ that has a neighbourhood $N_\epsilon(\overline{x}) = \{x : \|x - \overline{x}\| \leq \epsilon\}$ for some $\epsilon > 0$ such that $f(\overline{x}) \leq f(x)$ for each $x \in S \cap N_\epsilon(\overline{x})$.
3. a *global optimal solution* is a feasible solution $\overline{x} \in S$ with $f(\overline{x}) \leq f(x)$ for all $x \in S$. Or alternatively, is a local optimal solution for which $S \subseteq N_\epsilon(\overline{x})$.

Figure 1 illustrates the concepts above. Solution x_1 is an unconstrained global minimum, but it is not a feasible solution considering the feasibility set S. Solution x_2 is a local maximum for any neighbourhood $N_\epsilon(x_2)$ only encompassing the points within the same plateau. Solution x_3 is a local minimum, while x_4 is neither a local or a global optimum in the unconstrained case, but it is a global maximum in the constrained case. Finally, x_5 is the global minimum in the constrained case.

2 The role of convexity in optimality conditions

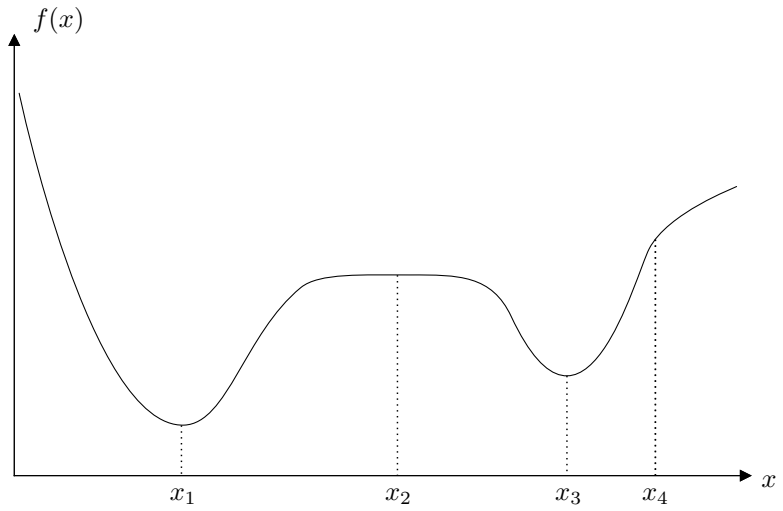
We can now state what is possibly the most important result in optimisation. In a nutshell, this results allows one promote local optimality to global optimality in the presence of convexity.

Theorem 1 (global optimality of convex problems)**.** *Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set and $f : S \to \mathbb{R}$ convex on S. Consider the problem (P) : min. $\{f(x) : x \in S\}$. Suppose $\overline{x}$ is a local optimal solution to P. Then $\overline{x}$ is a global optimal solution.*

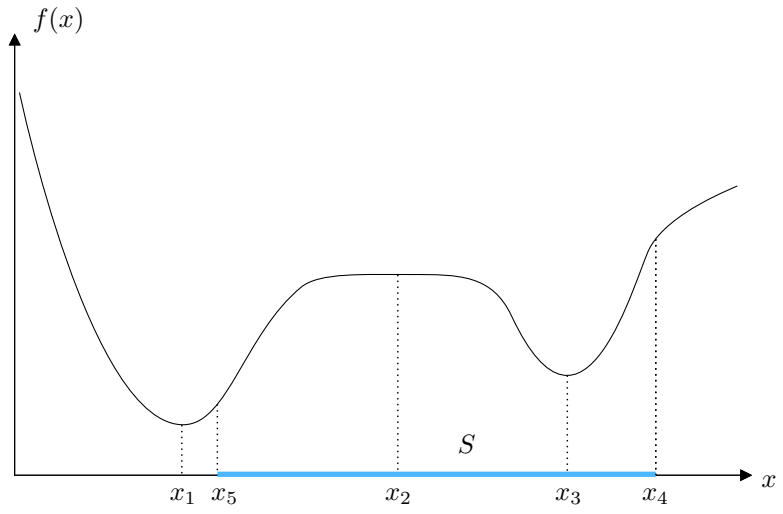
Proof. Since $\overline{x}$ is a local optimal solution, there exists $N_\epsilon(\overline{x})$ such that, for each $x \in S \cap N_\epsilon(\overline{x})$, $f(\overline{x}) \leq f(x)$. By contradiction, suppose $\overline{x}$ is not a global optimal solution. Then, there exists a solution $\hat{x} \in S$ so that $f(\hat{x}) < f(\overline{x})$. Now, for any $\lambda \in [0, 1]$, the convexity of f implies:

$$f(\lambda\hat{x} + (1 - \lambda)\overline{x}) \leq \lambda f(\hat{x}) + (1 - \lambda)f(\overline{x}) < \lambda f(\overline{x}) + (1 - \lambda)f(\overline{x}) = f(\overline{x})$$

However, for $\lambda > 0$ sufficiently small, $\lambda\hat{x} + (1 - \lambda)\overline{x} \in S \cap N_\epsilon(\overline{x})$ due to the convexity of S, which contradicts the local optimality of $\overline{x}$. Thus, $\overline{x}$ is a global optimum. □



(a) Unconstrained optimisation problem



(b) Constrained optimisation problem

Figure 1: Points of interest in optimisation. Points x_1, x_2 and x_3 are local optima in the unconstrained problem. Once a constraint set S is imposed, x_4 and x_5 become points of interest and x_1 becomes infeasible.

The proof is built using contradiction. That is, we show that for a solution to be a local optimum in a convex problem, not being a global solution contradicts its local optimality, originally true by assumption. This is achieved using the convexity of f and showing that some convex combinations between a hypothetical better solution $\hat{x}$ and $\overline{x}$ would have to be both in $N_\epsilon(\overline{x})$ and better than $\overline{x}$, contradicting the local optimality of $\overline{x}$.

3 Optimality condition of convex problems

We first look at optimality conditions in a general sense to then translate the concept to unconstrained and constrained problems specifically. Taking this more general standpoint is also helpful to understand how these can be specialised in the absence of a closed domain or in the presence of differentiability. We assume convexity for now, and later we will discuss further the consequences of the absence of convexity. Note that unconstrained problems have convex feasibility set (i.e., the whole $\mathbb{R}^n$), and thus what follows can be generalised to unconstrained optimisation problems.

Theorem 2 (optimality condition for convex problems)**.** *Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set and $f : \mathbb{R}^n \to \mathbb{R}$ convex on S. Consider the problem (P) : min. $\{f(x) : x \in S\}$. Then, $\overline{x} \in S$ is an optimal solution to (P) if and only if f has a subgradient ξ at $\overline{x}$ such that $\xi^\top(x - \overline{x}) \geq 0$ for all $x \in S$.*

Proof. Suppose that $\xi^\top(x - \overline{x}) \geq 0$ for all $x \in S$, where ξ is a subgradient of f at $\overline{x}$. By the definition of a subgradient, we have for all $x \in S$

$$f(x) \geq f(\overline{x}) + \xi^\top(x - \overline{x}) \geq f(\overline{x}),$$

and hence $\overline{x}$ is optimal.

Conversely, suppose that $\overline{x} \in S$ is a global optimum for P. Construct the sets:

$$\Lambda_1 = \{(x - \overline{x}, y) : x \in \mathbb{R}^n,\ y > f(x) - f(\overline{x})\},$$
$$\Lambda_2 = \{(x - \overline{x}, y) : x \in S,\ y \leq 0\}.$$

Note that Λ_1 and Λ_2 are convex: $\Lambda_1 \in \mathbb{R}^{n+1}$ is the Minkowski difference between the *open* epigraph of the convex function $x \mapsto f(x) - f(\overline{x})$ and the singleton $\{(\overline{x}, 0)\}$, whereas $\Lambda_2 \in \mathbb{R}^{n+1}$ is the Cartesian product of the convex set $S - \{\overline{x}\}$ and the closed negative coordinate axis. By the optimality of $\overline{x}$, the last component of any element in Λ_1 is positive, and thus $\Lambda_1 \cap \Lambda_2 = \emptyset$.

Using the *separation theorem*,[1] there exists a hyperplane that separates Λ_1 and Λ_2. That is, there

[1] The version of the separation theorem needed here is that two nonempty disjoint convex sets are separated by a hyperplane; see, e.g., the text book of the course.

exist a normal vector $0 \neq p \in \mathbb{R}^{n+1}$ and $\alpha \in \mathbb{R}$ such that

$$p^\top z \leq \alpha,\ \forall z \in \Lambda_1, \quad \text{and} \quad p^\top z \geq \alpha,\ \forall z \in \Lambda_2.$$

Writing $p = (\xi_0, \mu)$, with $\xi_0 \in \mathbb{R}^n$ and $\mu \in \mathbb{R}$, these conditions can be equivalently given as

$$\xi_0^\top (x - \overline{x}) + \mu y \leq \alpha,\ \forall x \in \mathbb{R}^n,\ \forall y > f(x) - f(\overline{x}), \tag{1}$$
$$\xi_0^\top (x - \overline{x}) + \mu y \geq \alpha,\ \forall x \in S,\ \forall y \leq 0. \tag{2}$$

Letting $x = \overline{x}$ and $y = 0$ in (2), we get $\alpha \leq 0$. Next, letting $x = \overline{x}$ and $y = \epsilon > 0$ in (1), we obtain $\alpha \geq \mu\epsilon$, meaning, in particular, that $\mu \leq 0$. Furthermore, as $\alpha \geq \mu\epsilon$ holds for any $\epsilon > 0$, we must have $\alpha \geq 0$, i.e., $\alpha = 0$ due to the already established nonpositivity of α.

If $\mu = 0$, we get from (1) that $\xi_0^\top (x - \overline{x}) \leq 0$ for all $x \in \mathbb{R}^n$. By letting $x = \overline{x} + \xi_0$, it follows that $\xi_0^\top (x - \overline{x}) = \|\xi_0\|^2 \leq 0$, and thus $\xi_0 = 0$. Since $(\xi_0, \mu) \neq 0$, we must thus have $\mu < 0$.

Dividing (1) and (2) by $-\mu \in \mathbb{R}_+$ and denoting $\xi = \frac{-\xi_0}{\mu}$, we obtain:

$$\xi^\top (x - \overline{x}) \leq y,\ \forall x \in \mathbb{R}^n,\ \forall y > f(x) - f(\overline{x}), \tag{3}$$
$$\xi^\top (x - \overline{x}) \geq y,\ \forall x \in S,\ \forall y \leq 0. \tag{4}$$

Letting $y = 0$ in (4), we get $\xi^\top (x - \overline{x}) \geq 0$ for all $x \in S$. It thus remains to prove that ξ is a subgradient at $\overline{x}$. To this end, fix an arbitrary $x \in \mathbb{R}^n$. According to (3), $y \geq \xi^\top (x - \overline{x})$ for every $y > f(x) - f(\overline{x})$. Taking a sequence $\{y_j\}_{j=1}^\infty$ such that $y_j \to f(x) - f(\overline{x})$ from above as $j \to \infty$, we thus get

$$f(x) - f(\overline{x}) = \lim_{j \to \infty} y_j \geq \xi^\top (x - \overline{x}) \qquad \Longrightarrow \qquad f(x) \geq f(\overline{x}) + \xi^\top (x - \overline{x}).$$

As the same argument is valid for any $x \in \mathbb{R}^n$, we have established the *subgradient inequality*. Thus, ξ is a subgradient for $f : \mathbb{R}^n \to \mathbb{R}$ at $\overline{x}$ with $\xi^\top (x - \overline{x}) \geq 0$ for all $x \in S$, which completes the proof. □

In the first part of the proof, we use the definition of a subgradient for a convex function to show that $\xi^\top (x - \overline{x}) \geq 0$ for a subgradient ξ of f at $\overline{x}$ implies that $f(\overline{x}) \leq f(x)$ for all $x \in S$.

The second part of the proof assumes that $\overline{x}$ is a global optimum and uses the separation theorem for *disjoint convex sets* in a creative way to show that a subgradient with the required property must exist at $\overline{x}$. This is achieved by introducing the two sets Λ_1 and Λ_2. Notice that $\overline{x}$ being optimal implies that $y > f(x) - f(\overline{x}) \geq 0$, which leads to the conclusion that $\Lambda_1 \cap \Lambda_2 = \emptyset$, demonstrating the existence of a separating hyperplane between them, as concretised by (1) and (2). We can easily show that α in those has to be 0 and μ nonnegative. In particular, note that this means the separating hyperplane passes through $(x - \overline{x}, y) = (0, 0)$, which is a natural conclusion since

obviously $(0,0) \in \mathbf{clo}(\Lambda_1) \cap \Lambda_2$, that is, the only "route" between the two sets passes through the origin of $\mathbb{R}^{n+1}$.

Next we demonstrate $\mu < 0$, so that we can divide (1) and (2) by μ to get closer to the required inequalities. We show that by contradiction: $\mu = 0$ would imply $\xi_0 = 0$, which disagrees with the existence of a nonzero normal vector $p \neq 0$ in the separation theorem. The inequality $\xi^\top(x - \overline{x}) \geq 0$ for all $x \in S$ immediately follows. Finally, as $y \geq \xi^\top(x - \overline{x})$ for any $y > f(x) - f(\overline{x})$, we can take a limit to deduce $f(x) - f(\overline{x}) \geq \xi^\top(x - \overline{x})$ for any $x \in \mathbb{R}^n$, which leads to the subgradient inequality.

Notice that this result provides necessary and sufficient conditions for optimality for convex problems. These conditions can also be extended to the unconstrained case as well, as indicated by Corollary 3.

Corollary 3 (optimality in open sets)**.** *Under the conditions of Theorem 2, if S is open, $\overline{x}$ is an optimal solution to P if and only if $0 \in \partial f(\overline{x})$.*

Proof. From Theorem 2, $\overline{x}$ is optimal if and only if there exists a subgradient ξ at $\overline{x}$ with $\xi^\top(x - \overline{x}) \geq 0$ for all $x \in S$. Since S is open, $x = \overline{x} - \lambda\xi \in S$ for small enough $\lambda > 0$, and thus $-\lambda\|\xi\|^2 \geq 0$, implying $\xi = 0$. □

Notice that, if S is open, then the only way to attain the condition $\xi^\top(x - \overline{x}) \geq 0$ is if $\xi = 0$ itself. This is particularly relevant in the context of nondifferentiable functions, as we will see later. Another important corollary is the classic optimality condition $\nabla f(\overline{x}) = 0$, which we state below for completeness.

Corollary 4 (optimality for differentiable functions)**.** *Suppose that $S \subseteq \mathbb{R}^n$ is a nonempty convex set and $f : \mathbb{R}^n \to \mathbb{R}$ a differentiable function that is convex on S. Then $\overline{x} \in S$ is optimal if and only if $\nabla f(\overline{x})^\top(x - \overline{x}) \geq 0$ for all $x \in S$. Moreover, if S is open, then $\overline{x} \in S$ is optimal if and only if $\nabla f(\overline{x}) = 0$.*

The proof for Corollary 4 is the same as Theorem 2 under a setting where $\partial(x) = \{\nabla f(x)\}$ due to the differentiability of f.

Let us consider two examples. First, consider the problem

$$\begin{aligned} \text{min.} \quad & \left(x_1 - \frac{3}{2}\right)^2 + (x_2 - 5)^2 \\ \text{subject to:} \quad & -x_1 + x_2 \leq 2 \\ & 2x_1 + 3x_2 \leq 11 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$

Figure 2 presents a plot of the feasible region S, which is form by the intersection of the two

halfspaces, and the level curves of the objective function, with some of the values indicated in the curves. Notice that that this is a convex problem.

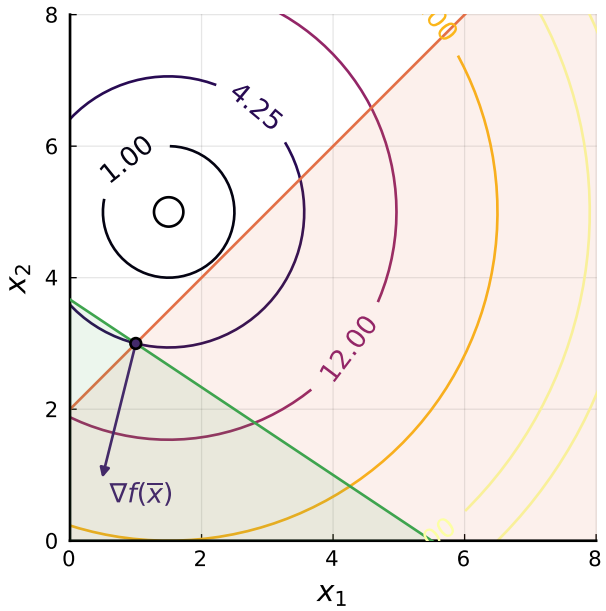


Figure 2: Example 1

The arrow shows the gradient $\nabla f(\overline{x})$ at $\overline{x} = (1, 3)$. Notice that this point is special since at that point, no vector $x - \overline{x}$ can be found forming an angle greater than 90° with $\nabla f(\overline{x})$, that is $\nabla f(\overline{x})^\top (x - \overline{x}) \geq 0$ for any $x \in S$, which means that $\overline{x}$ is optimal. Since the problem is convex, that is in fact the global optimum for this problem.

Figure 3 shows a similar situation, but now with one of the constraints being nonlinear. Notice that of the two points highlighted ($(1, 2)$ in orange and $(2, 1)$ in purple), the optimality condition only holds for $(2, 1)$. For example, for $x = (2, 1)$ and $\overline{x} = (1, 2)$ the vector $x - \overline{x}$ forms a angle greater than 90° with the gradient of f at $\overline{x}$, $\nabla f(\overline{x})$, and thus the condition $\nabla f(\overline{x})^\top (x - \overline{x}) \geq 0$ does not hold for all S. The condition does hold for $\overline{x} = (2, 1)$, as can be seen in Figure 3.

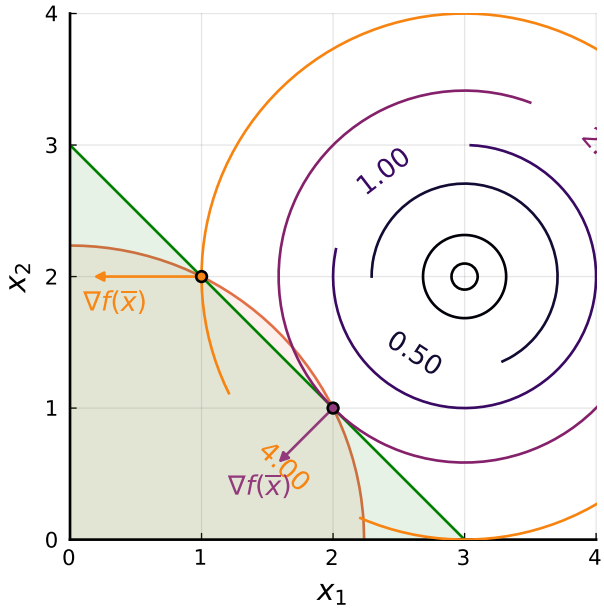


Figure 3: Example 2

A geometrical interpretation of the optimality condition $\xi^\top(x - \overline{x}) \geq 0$ is as follows. If there exists a subgradient ξ (or a gradient $\nabla f(\overline{x})$ if f is differentiable) that serves as a separating hyperplane between the level curve of f at $\overline{x}$ and the feasible region S, then there can be no feasible point further into the lower level set defined by that level curve. Ultimately, this means that there is no feasible point with smaller objective function value to be found. This is why the separation theorem from Lecture 2 plays an important role here, since it can be used to state that the feasible options have been exhausted in terms of potential directions of decrease of objective function value.

3.1 Optimality conditions for unconstrained problems

We have developed most of the concepts required to state optimality conditions for unconstrained optimisation problems, as presented in Corollaries 3 and 4. We now take an alternative route in which we do not take into account the feasibility set, but only the differentiability of f. This will be useful as it will allow us to momentarily depart from the assumption of convexity, which was used to state Theorem 2.

3.1.1 First-order optimality conditions

Let us start defining what it means to be a *descent direction*.

Theorem 5 (descent direction)**.** *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable at $\overline{x}$. If there is d such that*

$\nabla f(\overline{x})^\top d < 0$, there exists $\delta > 0$ such that $f(\overline{x} + \lambda d) < f(\overline{x})$ for each $\lambda \in (0, \delta)$, so that d is a descent direction of f at $\overline{x}$.

Proof. By differentiability of f at $\overline{x}$, we have that

$$\frac{f(\overline{x} + \lambda d) - f(\overline{x})}{\lambda} = \nabla f(\overline{x})^\top d + \|d\|\alpha(\overline{x}; \lambda d)$$

for any $\lambda > 0$. Since $\nabla f(\overline{x})^\top d < 0$ and $\alpha(\overline{x}; \lambda d) \to 0$ as λ tends to zero, we must have $f(\overline{x} + \lambda d) - f(\overline{x}) < 0$ for all $0 < \lambda < \delta$ for some $\delta > 0$. □

The proof uses the first-order expansion around $\overline{x}$ to show that, with f being differentiable, the condition $\nabla f(\overline{x})^\top d < 0$ implies that $f(\overline{x} + \lambda d) < f(\overline{x})$ for small enough $\lambda > 0$. Put in words, a small enough step in the direction d decreases the objective function value.

We can derive the first-order optimality condition in Corollary 4 as a consequence from Theorem 5. Notice, however, that since convexity is not assumed, all we can say is that this condition is necessary (but not sufficient) for local optimality.

Corollary 6 (first-order necessary condition)**.** *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable at $\overline{x}$. If $\overline{x}$ is a local minimum, then $\nabla f(\overline{x}) = 0$.*

Proof. By contradiction, suppose that $\nabla f(\overline{x}) \neq 0$ at a local minimum $\overline{x} \in \mathbb{R}^n$. Letting $d = -\nabla f(\overline{x})$, we have that $\nabla f(\overline{x})^\top d = -\|\nabla f(\overline{x})\|^2 < 0$. By Theorem 5, there exists a $\delta > 0$ such that $f(\overline{x} + \lambda d) < f(\overline{x})$ for all $\lambda \in (0, \delta)$, thus contradicting the local optimality of $\overline{x}$. □

Notice that Corollary 6 only holds in one direction. The proof uses contradiction once again: having $\nabla f(\overline{x}) \neq 0$ contradicts the assumed local optimality of $\overline{x}$. To show this, we simply demonstrate that $-\nabla f(\overline{x})$ is a proper descent direction.

3.1.2 Second-order optimality conditions

We now derive necessary conditions for local optimality of $\overline{x}$ based on second-order differentiability. As we will see, it requires that the Hessian $H(\overline{x})$ of $f(x)$ at $\overline{x}$ is positive semidefinite.

Theorem 7 (second-order necessary condition)**.** *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable at $\overline{x}$. If $\overline{x}$ is a local minimum, then $H(\overline{x})$ is positive semidefinite.*

Proof. Suppose $\overline{x} \in S$ is a local minimum and take an arbitrary direction $d \in \mathbb{R}^n$. As f is twice differentiable, we have:

$$f(\overline{x} + \lambda d) = f(\overline{x}) + \lambda \nabla f(\overline{x})^\top d + \frac{1}{2}\lambda^2 d^\top H(\overline{x}) d + \lambda^2 \|d\|^2 \alpha(\overline{x}; \lambda d).$$

Since $\overline{x}$ is a local minimum, $f(\overline{x} + \lambda d) \geq f(\overline{x})$ for small enough $\lambda \neq 0$ and Corollary 6 further guarantees that $\nabla f(\overline{x}) = 0$. Rearranging terms and dividing by $\lambda^2 > 0$, we thus obtain

$$\frac{f(\overline{x} + \lambda d) - f(\overline{x})}{\lambda^2} = \frac{1}{2} d^\top H(\overline{x}) d + \|d\|^2 \alpha(\overline{x}; \lambda d).$$

Since $\alpha(\overline{x}; \lambda d) \to 0$ as $\lambda \to 0$, we have that $d^\top H(\overline{x}) d \geq 0$. □

The second-order conditions can be used to attest local optimality of $\overline{x}$. In the case where $H(\overline{x})$ is positive definite, then this second order condition becomes *sufficient* for local optimality, since it implies that the function is 'locally convex' for a small enough neighbourhood $N_\epsilon(\overline{x})$.

In case f is convex, then the first-order condition $\nabla f(x) = 0$ becomes also sufficient for attesting the global optimality of $\overline{x}$. Recall that a twice differentiable f is convex if and only if $H(x)$ is positive semidefinite for all $x \in \mathbb{R}^n$, meaning that in this case the second-order necessary conditions are also satisfied at $\overline{x}$.

Theorem 8. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable and convex. Then $\overline{x}$ is a global minimum if and only if $\nabla f(\overline{x}) = 0$.*

Proof. By Corollary 6, if $\overline{x}$ is a global minimum, then $\nabla f(\overline{x}) = 0$.

Assume next that $\nabla f(\overline{x}) = 0$. Since f is convex,

$$f(x) \geq f(\overline{x}) + \nabla f(\overline{x})^\top (x - \overline{x}), \ \forall x \in \mathbb{R}^n.$$

By assumption $\nabla f(\overline{x})^\top (x - \overline{x}) = 0$ for all $x \in \mathbb{R}^n$, thus showing that $f(\overline{x}) \leq f(x)$ for all $x \in \mathbb{R}^n$.

The claim also directly follows from Corollary 3 by choosing the considered open convex set to be $S = \mathbb{R}^n$ and recalling that for a differentiable convex f the subdifferential is a singleton $\partial f(\overline{x}) = \{\nabla f(\overline{x})\}$. □

Lecture 5 - Unconstrained optimisation methods I

Fabricio Oliveira (with small modifications by Nuutti Hyvönen)

Last update: October 1, 2022

Abstract. In this lecture, we present methods for solving unconstrained optimisation problems. We start by defining a general optimisation algorithm that will serve as a reference for deriving variants of optimisation methods. We concentrate first on how to generate step sizes using variants of unidimensional optimisation methods, the so called line searches. We also present the Armijo rule as an inexact line search method, widely used in state-of-the-art implementations of optimisation algorithms. Next, we focus on three variants of multidimensional methods, namely the coordinate descent (derivative free), the gradient descent method that relies in first order approximations, and the Newton's method, which relies on second-order information. We also discuss the effects of having exact and inexact line searches in each of these methods.

Outline of this lecture

1 A prototype of an optimisation method

Most, if not all, optimisation methods are based on the conceptual notion of successively obtaining *directions* of potential improvement and suitable *step sizes* in this direction, until a convergence or termination criterion (collectively called stopping criteria) is satisfied.

Considering what we have seen so far, we have now the concepts required for describing several unconstrained optimisation methods. We start by posing a conceptual optimisation algorithm in a pseudocode structure. This will be helpful in identifying the elements that differentiate the methods we will discuss.

Algorithm 1 Conceptual optimisation algorithm

1: **initialise.** iteration count $k = 0$, starting point x_0
2: **while** stopping criteria are not met **do**
3: compute direction d_k
4: compute step size λ_k
5: $x_{k+1} = x_k + \lambda_k d_k$
6: $k = k + 1$
7: **end while**
8: **return** x_k.

Algorithm 1 has two main elements, namely the computation of the direction d_k and the step size λ_k at each iteration k. In what follows, we present some univariate optimisation methods that can be employed to calculate step sizes λ_k. These methods are commonly referred to as *line search methods*.

2 Line search methods

Finding an optimal step size λ_k is in itself an optimisation problem. The name line search refers to the fact that it consists of a unidimensional search as $\lambda_k \in \mathbb{R}$.

Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable. We define the unidimensional function $\theta : \mathbb{R} \to \mathbb{R}$ as

$$\theta(\lambda) = f(x + \lambda d).$$

Assuming differentiability, we can use the first-order necessary condition $\theta'(\lambda) = 0$ to obtain optimal values for the step size λ. This means solving the system

$$\theta'(\lambda) = d^\top \nabla f(x + \lambda d) = 0,$$

which might pose challenges. First, $d^\top \nabla f(x+\lambda d)$ is often nonlinear in λ, with optimal solutions not

trivially resting at boundary points for an explicit domain of λ. Moreover, recall that $\theta'(\lambda) = 0$ is not a sufficient condition for optimality in general, unless properties such as convexity can be inferred.

In what follows, we assume that strict quasiconvexity holds and therefore $\theta'(\lambda) = 0$ becomes necessary and sufficient for optimality. In some contexts, unidimensional strictly quasiconvex functions are called *unimodal*.

Theorem 1 establishes the mechanism underpinning line search methods. In that, we use the assumption that the function has a unique minimum (a consequence of being strictly quasiconvex) to successively reduce the search space until the optimum is contained in a sufficiently small interval of length l, i.e., until the optimum is determinend within an acceptable tolerance.

Theorem 1 (Line search reduction)**.** *Let $\theta : \mathbb{R} \to \mathbb{R}$ be strictly quasiconvex over the interval $[a, b]$, and let $\lambda, \mu \in [a, b]$ be such that $\lambda < \mu$. If $\theta(\lambda) \geq \theta(\mu)$, then $\theta(z) > \theta(\lambda)$ for all $z \in [a, \lambda)$. If $\theta(\lambda) \leq \theta(\mu)$, then $\theta(z) > \theta(\mu)$ for all $z \in (\mu, b]$.*

Proof. Assume $\theta(\lambda) \geq \theta(\mu)$ for $\lambda, \mu \in [a, b]$ such that $\lambda < \mu$, and let $z \in [a, \lambda)$ be arbitrary. As $\lambda \in (z, \mu)$, there exists $0 < \tau < 1$ such that $\lambda = \tau z + (1 - \tau)\mu$. By virtue of the strict quasiconvexity of θ,

$$\theta(\lambda) = \theta(\tau z + (1 - \tau)\mu) < \max\{\theta(z), \theta(\mu)\}.$$

Because $\theta(\lambda) \geq \theta(\mu)$ by assumption, it must thus hold $\theta(z) > \theta(\lambda)$. The second part of the claim follows via an analogous argument. □

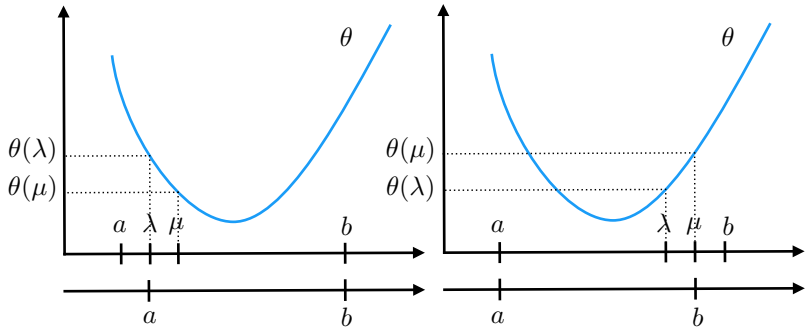


Figure 1: Applying Theorem 1 allows to iteratively reduce the search space.

Figure 1 provides an illustration of Theorem 1. The line below the x-axis illustrates how the search space can be reduced between two successive iterations. In fact, most line search methods will iteratively reduce the search interval (represented by $[a, b]$) until the interval is sufficiently small to be considered "a point" (i.e., is smaller than a set threshold l).

Line searches are *exact* if optimal (up to the chosen tolerance) step sizes λ_k^* are calculated at each iteration k of the conceptual Algorithm 1, and inexact if arbitrarily good approximations for λ_k^*

are used instead. As we will see, there is a trade-off between the number iterations required for convergence and the time taken per iteration, which must be taken into account when choosing between exact and inexact line searches.

2.1 Exact line searches

Exact methods are designed to return the optimal step value λ^* within a pre-specified tolerance l. In practice, it means that these methods return an interval $[a_k, b_k]$ such that $b_k - a_k \leq l$.

2.1.1 Uniform search

The uniform search consists of breaking the search domain $[a, b]$ into $N \in \mathbb{N}$ slices of uniform width $\delta = \frac{|b-a|}{N}$. This leads to a one-dimensional grid with the grid points $a_n = a_0 + n\delta$, $n = 0, \ldots, N$, where $a_0 = a$ and $a_N = b$. We can then set $\hat{\lambda}$ to be

$$\hat{\lambda} = \arg\min_{i=0,\ldots,n} f(a_i).$$

From Theorem 1, we know that the optimal step size $\lambda^* \in [\hat{\lambda} - \delta, \hat{\lambda} + \delta]$. The process can then be repeated, by recursively defining $a = \hat{\lambda} - \delta$ and $b = \hat{\lambda} + \delta$ (see Figure 2) until $|a - b|$ is less than a prespecified tolerance l. Without a sufficient number repetitions, the uniform search becomes an inexact search, that is, the prespecified tolerance l is not reached.

This type of search is particularly useful when setting values for hyperparameters in algorithms (that is, user defined parameters that influence the behaviour of the algorithm) or performing any sort of search in a grid structure. One concept related to this type of search is what is known as the *coarse-to-fine approach*. Coarse-to-fine approaches use sequences of increasingly fine approximations (i.e., gradually increasing n) to obtain computational savings in terms of function evaluations. In fact, the number of function evaluations a line search method executes is one of the indicators of its efficiency.

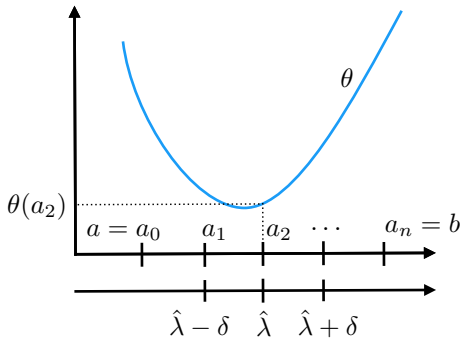


Figure 2: Grid search with 5 points; Note that $\theta(a_2) = \min_{i=0,\ldots,n} \theta(a_i)$.

2.1.2 Dichotomous search

The *dichotomous search* is a sequential line search method whose aim is to (almost) halve the search interval at each iteration based on two new function evaluations.

The word dichotomous refers to the mutually exclusive parts into which the search interval is divided at each iteration. We initiate the search by defining a distance margin ϵ as well as the original reference points $\lambda_0 = \frac{a_0+b_0}{2} - \epsilon$ and $\mu_0 = \frac{a_0+b_0}{2} + \epsilon$, where $a_0 = a$ and $b_0 = b$ are the endpoints of the original search interval. We then proceed iteratively as follows starting from $k = 0$:

1. If $\theta(\lambda_k) < \theta(\mu_k)$, then *move to the left* by setting $a_{k+1} = a_k$ and $b_{k+1} = \mu_k$;
2. Otherwise, if $\theta(\lambda_k) > \theta(\mu_k)$, then *move to the right* by setting $a_{k+1} = \lambda_k$ and $b_{k+1} = b_k$.
3. Define $\lambda_{k+1} = \frac{a_{k+1}+b_{k+1}}{2} - \epsilon$ and $\mu_{k+1} = \frac{a_{k+1}+b_{k+1}}{2} + \epsilon$.

To be quite precise, one should also introduce a criterion for resolving a tie $\theta(\lambda_k) = \theta(\mu_k)$; at a tie, one can, e.g., choose to always "move to the left", meaning that $\theta(\lambda_k) < \theta(\mu_k)$ is replaced by $\theta(\lambda_k) \leq \theta(\mu_k)$ in item 1, or to always "move to the right" by dropping "if $\theta(\lambda_k) > \theta(\mu_k)$" from item 2.[1] Once the new search interval $[a_{k+1}, b_{k+1}]$ is updated, new reference points λ_{k+1} and μ_{k+1} are calculated and the process is repeated until $|a - b| \leq l$.

The method is summarised in Algorithm 2, where the choice is to "move to the right" at a tie. Notice that one can calculate what will be the width $|a_{k+1} - b_{k+1}|$ at any given iteration k:

$$b_{k+1} - a_{k+1} = \frac{1}{2^k}(b_0 - a_0) + 2\epsilon\left(1 - \frac{1}{2^k}\right).$$

This is useful in that it allows predicting the number of iterations Algorithm 2 will require before convergence. Figure 3 illustrates the process for two distinct functions. Notice that the employment

[1] Under our assumption of strict quasiconvexity, a tie would actually theoretically mean that the minimiser lies in the interval (λ_k, μ_k), and one could thus choose $a_{k+1} = \lambda_k$ and $b_{k+1} = \mu_k$; see Theorem 1.

of the central point $\frac{a+b}{2}$ as the reference to define the points λ and μ turns the method robust in terms of interval reduction at each iteration.

Algorithm 2 Dichotomous search

1: **initialise.** distance margin $\epsilon > 0$, tolerance $l > 0$, $[a_0, b_0] = [a, b]$, $k = 0$
2: **while** $b_k - a_k > l$ **do**
3: $\lambda_k = \frac{a_k+b_k}{2} - \epsilon$, $\mu_k = \frac{a_k+b_k}{2} + \epsilon$
4: **if** $\theta(\lambda_k) < \theta(\mu_k)$ **then**
5: $a_{k+1} = a_k$, $b_{k+1} = \mu_k$
6: **else**
7: $a_{k+1} = \lambda_k$, $b_{k+1} = b_k$
8: **end if**
9: $k = k + 1$
10: **end while**
11: **return** $\overline{\lambda} = \frac{a_k+b_k}{2}$

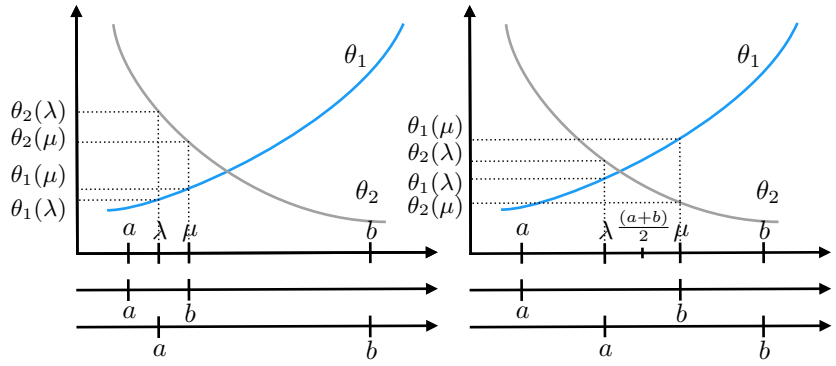


Figure 3: Using the midpoint $(a + b)/2$ and Theorem 1 to reduce the search space.

2.1.3 Golden section search*

The *golden section search* is named after the *golden ratio* $\varphi = \frac{1+\sqrt{5}}{2}$, the reciprocal of which (i.e., $\alpha \approx 0.618$) is used as the ratio of reduction for the search interval $[a, b]$ at each iteration. The golden section search requires only a single new function evaluation at each iteration. This property typically makes it more efficient than the dichotomous search, although the reduction in the length of the search interval at a single iteration in the golden section search is less than in the dichotomous search (assuming a "small enough" $\epsilon > 0$ is employed for the latter).

Suppose that we, once again, rely on two reference points $\lambda_k < \mu_k$ at the $(k + 1)$th iteration of the considered line search. The golden section search is based on imposing two requirements:

1. The relative reduction in the width of the search interval does not depend on whether $\theta(\lambda_k) > \theta(\mu_k)$ or vice-versa.

2. At each iteration, we perform only a single new function evaluation:

   (a) If $\theta(\lambda_k) > \theta(\mu_k)$, then we choose $\lambda_{k+1} = \mu_k$ as well as $a_{k+1} = \lambda_k$ and $b_{k+1} = b_k$, i.e., only $\theta(\mu_{k+1})$ needs to be evaluated at the next interation since $\theta(\lambda_{k+1}) = \theta(\mu_k)$ is already known. ("move to the right")

   (b) If $\theta(\lambda_k) < \theta(\mu_k)$, then we choose $\mu_{k+1} = \lambda_k$ as well as $a_{k+1} = a_k$ and $b_{k+1} = \mu_k$, i.e., only $\theta(\lambda_{k+1})$ needs to be evaluated at the next iteration since $\theta(\mu_{k+1}) = \theta(\lambda_k)$ is already known. ("move to the left")

The remaining question is how to choose μ_{k+1} in option 2(a) and λ_{k+1} in option 2(b) so that the first requirement is satisfied.

From requirement 1 and the update options 2(a) and 2(b), we infer that the conditions $b_{k+1} - a_{k+1} = b_k - \lambda_k = \mu_k - a_k$ must be satisfied. To find the interval reduction rate $\alpha \in (0, 1)$ that would allow this, we write $\mu_k = a_k + \alpha(b_k - a_k)$ and, consequently, $\lambda_k = a_k + (1 - \alpha)(b_k - a_k)$. Notice that these choices result in the same reduction rate $b_{k+1} - a_{k+1} = \alpha(b_k - a_k)$ independently of whether $\theta(\lambda_k) > \theta(\mu_k)$ or vice-versa. (As we aim for $\lambda_k < \mu_k$, we are actually only interested in reduction rates in the interval $(0.5, 1)$.)

Based on the above considerations, we can now solve for the ratio α that allows the method to function as desired. Let us assume that $\theta(\lambda_k) < \theta(\mu_k)$, which leads to the choices $\mu_{k+1} = \lambda_k$, $a_{k+1} = a_k$ and $b_{k+1} = \mu_k$ in 2(b). We write μ_{k+1} in two ways using only α, a_k and b_k:

$$\mu_{k+1} = \lambda_k = a_k + (1 - \alpha)(b_k - a_k)$$

and

$$\begin{aligned}\mu_{k+1} &= a_{k+1} + \alpha(b_{k+1} - a_{k+1}) = a_k + \alpha(\mu_k - a_k) \\ &= a_k + \alpha(a_k + \alpha(b_k - a_k) - a_k) = a_k + \alpha^2(b_k - a_k).\end{aligned}$$

Equating these and dividing by $b_k - a_k \neq 0$ leads to the quadratic equation $\alpha^2 + \alpha - 1 = 0$ to which $\alpha = \frac{\sqrt{5}-1}{2} = \frac{2}{1+\sqrt{5}} = \frac{1}{\varphi} \approx 0.618$ is the positive solution. It is straightforward to check that the same result would be obtained if one considered the case $\theta(\mu_k) < \theta(\lambda_k)$ instead.

Algorithm 3 summarises the golden section search. Notice that at each iteration, only a single additional function evaluation is required.

Algorithm 3 Golden section search

initialise. tolerance $l > 0$, $[a_0, b_0] = [a, b]$, $\alpha = 0.618$, $k = 0$
$\lambda_k = a_k + (1 - \alpha)(b_k - a_k)$, $\mu_k = a_k + \alpha(b_k - a_k)$
while $b_k - a_k > l$ **do**
    if $\theta(\lambda_k) > \theta(\mu_k)$ **then**
        $a_{k+1} = \lambda_k$, $b_{k+1} = b_k$, $\lambda_{k+1} = \mu_k$, and
        $\mu_{k+1} = a_{k+1} + \alpha(b_{k+1} - a_{k+1})$. Calculate $\theta(\mu_{k+1})$.
    else
        $a_{k+1} = a_k$, $b_{k+1} = \mu_k$, $\mu_{k+1} = \lambda_k$, and
        $\lambda_{k+1} = a_{k+1} + (1 - \alpha)(b_{k+1} - a_{k+1})$. Calculate $\theta(\lambda_{k+1})$.
    end if
    $k \leftarrow k + 1$
end while
return $\overline{\lambda} = \frac{a_k + b_k}{2}$

Comparing the introduced methods for a given tolerance l, the required number of function evaluations is:

$$\min \left\{ n : \begin{array}{l} \text{uniform: } n \geq \frac{b-a}{l/2} - 1 \\ \text{dichotomous: } (1/2)^{n/2} \leq \frac{l}{b-a} \\ \text{golden section: } (0.618)^{n-1} \leq \frac{l}{b-a} \end{array} \right\},$$

where it is assumed that ϵ in the dichotomous search is so small compared to the tolerance l that its effect does not have to be taken into account. Moreover, the implementation of the uniform search does not utilize recursion. For example, suppose we set $[a, b] = [-10, 10]$ and $l = 10^{-6}$. Then the number of iterations required for convergence is

- uniform: $n = 4 \times 10^6$;
- dichotomous: $n = 49$;
- golden section: $n = 36$.

A variant of the golden section method uses Fibonacci numbers to define the ratio of interval reduction. Despite being marginally more efficient in terms of function evaluations, the overhead of calculating Fibonacci numbers has to be taken into account.

2.1.4 Bisection search

Unlike the previous methods, the bisection search relies on derivative information to infer how the search interval should be reduced. For that, we assume $\theta(\lambda)$ is differentiable and convex.

We proceed as follows. If $\theta'(\lambda_k) = 0$, then λ_k is a minimiser. Otherwise

1. if $\theta'(\lambda_k) > 0$, then, for $\lambda > \lambda_k$, we have $\theta'(\lambda_k)(\lambda - \lambda_k) > 0$, which implies $\theta(\lambda) \geq \theta(\lambda_k)$ since θ is convex. Therefore, the new search interval becomes $[a_{k+1}, b_{k+1}] = [a_k, \lambda_k]$.

2. If $\theta'(\lambda_k) < 0$, we have $\theta'(\lambda_k)(\lambda - \lambda_k) > 0$ (and thus $\theta(\lambda) \geq \theta(\lambda_k)$) for $\lambda < \lambda_k$. Thus, the new search interval becomes $[a_{k+1}, b_{k+1}] = [\lambda_k, b_k]$.

Mimicking the dichotomous search, we set $\lambda_k = \frac{1}{2}(b_k + a_k)$, which provides a robust guarantee on the search interval reduction. Notice that the dichotomous search can be seen as a bisection search in which the derivative information is estimated using the difference of function evaluation at two distinct points. Algorithm 4 summarises the bisection method.

Algorithm 4 Bisection method

initialise. tolerance $l > 0$, $[a_0, b_0] = [a, b]$, $k = 0$
while $b_k - a_k > l$ **do**
  $\lambda_k = \frac{(b_k + a_k)}{2}$ and evaluate $\theta'(\lambda_k)$
  if $\theta'(\lambda_k) = 0$ **then** return λ_k
  else if $\theta'(\lambda_k) > 0$ **then**
    $a_{k+1} = a_k$, $b_{k+1} = \lambda_k$
  else
    $a_{k+1} = \lambda_k$, $b_{k+1} = b_k$
  end if
  $k \leftarrow k + 1$
end while
return $\overline{\lambda} = \frac{a_k + b_k}{2}$

2.2 Inexact line search

Often, it is worth sacrificing the optimality of the step size λ_k for the overall efficiency of the solution method in terms of solution time.

There are several heuristics that can be employed to define step sizes and their performance is related to how the directions d_k are defined in Algorithm 1. Next, we present the *Armijo rule*, arguably the most commonly used technique to obtain step sizes in efficient implementations of optimisation methods.

2.2.1 Armijo rule

The Armijo rule is a condition that is tested to decide whether a current step size $\overline{\lambda}$ is acceptable or not. The step size $\overline{\lambda}$ is considered acceptable if

$$f(x + d\overline{\lambda}) - f(x) \leq \alpha\overline{\lambda}\nabla f(x)^\top d.$$

One way of understanding the Armijo rule is to consider what it means in terms of the function $\theta(\lambda) = f(x + \lambda d)$, i.e.,

$$\begin{aligned} &\theta(\overline{\lambda}) - \theta(0) \leq \alpha\overline{\lambda}\theta'(0), \quad \text{or} \\ &\theta(\overline{\lambda}) \leq \theta(0) + \alpha\overline{\lambda}\theta'(0). \end{aligned} \tag{1}$$

That is, $\theta(\overline{\lambda})$ has to be less than a deflected linear extrapolation of θ at $\lambda = 0$. The deflection is defined by the prespecified parameter α that typically lies in the interval $(0, 1)$. In case $\overline{\lambda}$ does not satisfy the test in (1), $\overline{\lambda}$ is reduced by a multiplicative factor $\beta \in (0, 1)$ until the test in (1) is satisfied.

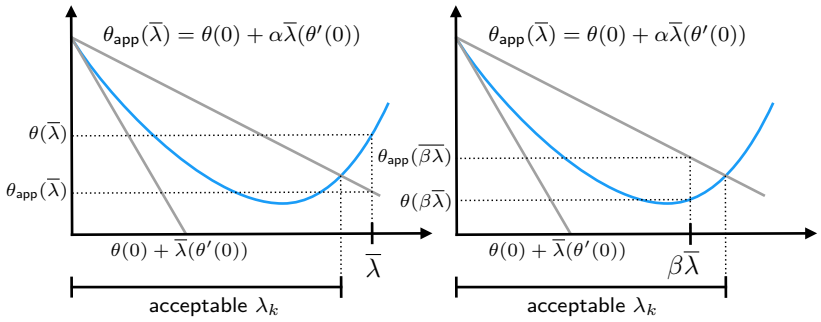


Figure 4: At first $\lambda_0 = \overline{\lambda}$ is not acceptable; after reducing the step size to $\lambda_1 = \beta\overline{\lambda}$, it enters the acceptable range where $\theta(\lambda_k) \leq \theta_{\text{app}}(\lambda_k) = \theta(0) + \alpha\lambda_k(\theta'(0))$.

In Figure 4, we can see the acceptable region for the Armijo test. At first, $\overline{\lambda}$ does not satisfy the condition (1). It is then reduced to $\beta\overline{\lambda}$, which, in turn, satisfies (1). In this case, the step size λ_k would have been chosen as $\beta\overline{\lambda}$. Suitable values for α are in the range $(0, 0.5]$ and for β in the range $(0, 1)$, with a trade-off between precision (higher values) and number of tests before acceptance (lower values).

The Armijo rule is called *backtracking* in some contexts, due to the successive reduction of the step size caused by the factor $\beta \in (0, 1)$. Some variants might also include rules that prevent the step size from becoming too small, such as $\theta(\delta\overline{\lambda}) \geq \theta(0) + \alpha\delta\overline{\lambda}\theta'(0)$, with $\delta > 1$.

3 Unconstrained optimisation methods

We now focus on developing methods that can be employed to optimise $f : \mathbb{R}^n \to \mathbb{R}$. We start with the coordinate descent method, which is derivative free, and then discuss the gradient method and Newton's method. In essence, the main difference between the three methods is how the directions d_k in Algorithm 1 are determined. Also, all of these methods rely on line search to define optimal

step sizes; the line search method of choice can be any of the algorithms introduced above or any other unidimensional optimisation method.

3.1 Coordinate descent

The *cordinate descent method* relies on a simple yet powerful idea. By focusing on one coordinate at a time, the method chooses in turns directions d such that $d_i = 1$ for a single coordinate i and $d_j = 0$ for $j \neq i$. As one would suspect, the order in which the coordinates are selected influences the performance of the algorithm. Some known variants include:

1. **Cyclic:** coordinates are considered in the trivial order $1, \ldots, n$;
2. **Double-sweep:** swap the coordinate order at each iteration;
3. **Gauss–Southwell:** choose components with largest $\frac{\partial f(x)}{\partial x_i}$;
4. **Stochastic:** coordinates are selected at random.

Algorithm 5 summarises the general structure of the coordinate descent method. Here, we denote the standard coordinate vectors as $\mathrm{e}_1, \ldots, \mathrm{e}_n$. Notice that the for-loop starting in line 4 uses the cyclic variant of the coordinate descent method.

Algorithm 5 Coordinate descent method (cyclic)

1: **initialise.** tolerance $\epsilon > 0$, initial point x^0, iteration count $k = 0$
2: **while** $\|x^k - x^{k-1}\| > \epsilon$ or $k = 0$ **do**
3: $\quad y = x^k$
4: $\quad$ **for** $j = 1, \ldots, n$ **do**
5: $\quad\quad \overline{\lambda} = \operatorname{argmin}_{\lambda \in \mathbb{R}} \{f(y + \lambda \mathrm{e}_j)\}$
6: $\quad\quad y = y + \overline{\lambda} \mathrm{e}_j$
7: $\quad$ **end for**
8: $\quad x^{k+1} = y$
9: $\quad k = k + 1$
10: **end while**
11: **return** x^k

This version of the coordinate descent does not employ any information on the derivatives of f, and thus the line search step on line 5 must take into account both directions, not just $\lambda > 0$.

Figure 5 shows the progress of the algorithm when applied to solve

$$f(x) = e^{(-(x_1-3)/2)} + e^{((4x_2+x_1)/10)} + e^{((-4x_2+x_1)/10)}$$

using the golden section method as line search.

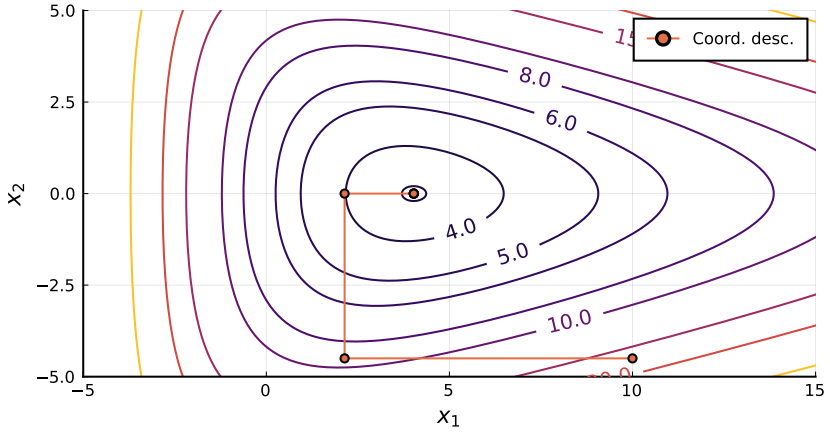


Figure 5: Coordinate descent method applied to f. Convergence is observed in 4 steps for a tolerance $\epsilon = 10^{-5}$

The coordinate descent method is the strategy employed in several other methods, such as the *Gauss–Seidel* method for solving linear system of equations, which is why some references refer to each of these iterations as Gauss–Seidel steps. Also, when a collection of coordinates is used to derive a direction, the term *block coordinate descent* is used, though a method for deriving directions for each block is still necessary, for example the gradient method presented next.

3.2 Gradient (descent) method

The *gradient descent* uses (minus one times) the gradient of the considered function as the search direction d. Before we present the method, let us consider a result that justifies the use of gradients to derive search directions.

Lemma 2. *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable at $x \in \mathbb{R}^n$ and $\nabla f(x) \neq 0$. Then $\overline{d} = -\frac{\nabla f(x)}{\|\nabla f(x)\|}$ is the direction of steepest descent of f at x.*

Proof. From differentiability of f, we have

$$f'(x; d) := \lim_{\lambda \to 0^+} \frac{f(x + \lambda d) - f(x)}{\lambda} = \nabla f(x)^\top d,$$

where $f'(x; d)$ denotes the directional derivative of f in the direction d. Thus,

$$\overline{d} = \operatorname*{argmin}_{\|d\|=1} f'(x; d) = \operatorname*{argmin}_{\|d\|=1} \left\{\nabla f(x)^\top d\right\} = -\frac{\nabla f(x)}{\|\nabla f(x)\|}.$$

In other words, the negative gradient determines the direction in which f decreases fastest. □

In the proof, we use the differentiability of f to define the directional derivative for f in the direction d. It is given by $\nabla f(x)^\top d$. If we minimise this term with respect to d satisfying $\|d\|_2 = 1$, we get $d = -\frac{\nabla f(x)}{\|\nabla f(x)\|}$, i.e., a vector of length one pointing to the opposite direction compared to $\nabla f(x)$.

This result provides us with the insight that it may be reasonable to define the search direction at x to be $-\nabla f(x)$, i.e., the direction of steepest descent. This is why the gradient method is called the method of *steepest descent* in some references, though gradient and steepest descent might refer to different methods in specific contexts. Notice that for maximation problems one should use the opposite direction $\nabla f(x)$ instead.

Using the gradient $\nabla f(x)$ for defining the search direction is also of convenience as it allows for the definition of a straightforward convergence condition. Notice that, if $\nabla f(x_k) = 0$, then the algorithm stalls, as $x_{k+1} = x_k + \lambda_k d_k = x_k$. In other words, the algorithm converges to points $\overline{x} \in \mathbb{R}^n$ that satisfy the first-order necessary conditions $\nabla f(\overline{x}) = 0$.

The gradient method has many known variants that try to mitigate issues associated with its poor convergence caused by the natural 'zigzagging' behaviour of the algorithm (see, e.g., the gradient method *with momentum* and the *Nesterov* method). There are also variants that only consider the partial derivatives of some (and not all) of the dimensions $i = 1, \ldots, n$ at a time, forming *blocks* of coordinates at each iteration. If these blocks are randomly formed, the associated methods are known as *stochastic gradient* methods.

In Algorithm 6 we provide a pseudocode for the gradient method. In line 2, the stopping condition for the while-loop is equivalent to testing $\nabla f(\overline{x}) = 0$ for a tolerance ϵ.

Algorithm 6 Gradient method

1: **initialise.** tolerance $\epsilon > 0$, initial point x_0, iteration count $k = 0$.
2: **while** $\|\nabla f(x_k)\| > \epsilon$ **do**
3: $\quad d = -\frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}$
4: $\quad \overline{\lambda} = \operatorname{argmin}_{\lambda \in \mathbb{R}_+} \{f(x_k + \lambda d)\}$
5: $\quad x_{k+1} = x_k + \overline{\lambda} d$
6: $\quad k = k + 1$
7: **end while**
8: **return** x_k.

Figure 6 presents the progress of the gradient method using exact (bisection) and inexact (Armijo rule with $\alpha = 0.1$ and $\beta = 0.7$) line searches. As can be expected, when an inexact line search is employed, the method slightly overshoots some of the steps, taking a few more iterations to converge.

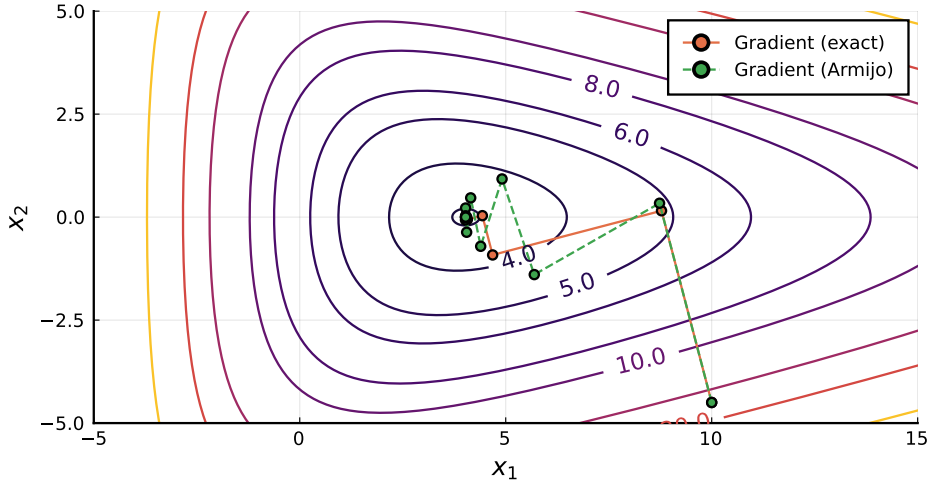


Figure 6: Gradient method applied to f. Convergence is observed in 10 steps using exact line search and in 19 using Armijo's rule (for $\epsilon = 10^{-5}$)

3.3 Newton's method

One can think of gradient methods as using first-order information to derive directions of improvement, while *Newton's method* also incorporates second-order information. This can be shown to produce better convergence properties, but at the expense of the extra computational burden incurred by calculating and manipulating Hessian matrices.

The basic idea of the Newton's method is the following. Consider the second-order approximation of f at x_k, that is,

$$f(x) \approx q(x) = f(x_k) + \nabla f(x_k)^\top (x - x_k) + \frac{1}{2}(x - x_k)^\top H(x_k)(x - x_k).$$

The method uses as the search direction d the vector pointing from x_k to the optimum of the above quadratic approximation around x_k. This can be obtained from the first-order condition $\nabla q(x) = 0$ (assuming convexity/concavity):

$$\nabla q(x) = \nabla f(x_k) + H(x_k)(x - x_k) = 0. \tag{2}$$

Assuming that $H^{-1}(x_k)$ exists, we can use (2) to obtain the following update rule, which is known as the *Newton step*:

$$x_{k+1} = x_k - H^{-1}(x_k)\nabla f(x_k). \tag{3}$$

Notice that the "pure" Newton's method described by (3) does not only define the search direction but also the step size, i.e., it gives an explicit formula for x_{k+1}. However, in practice the method

typically uses $d = -H^{-1}(x_k)\nabla f(x_k)$ as the search direction combined with a line search to obtain an optimal step size and to prevent divergence (that is, converge to $-\infty$) in cases where the second-order approximation might lead to divergence. Choosing the step size $\lambda = 1$ renders the pure Newton's method, as given in (3). The Newton's method can also be interpreted as employing the Newton–Raphson method to solve the system of equations that describe the first order conditions for f.

Figure 7 shows the calculation of the direction $d = -H^{-1}(x_k)\nabla f(x_k)$ for the first iteration of the Newton's method. Notice that the direction is the same as that for the minimum of the quadratic approximation $q(x)$ at x_k. The employment of a line search allows for avoiding overshooting the exact minimum, making the search more efficient.

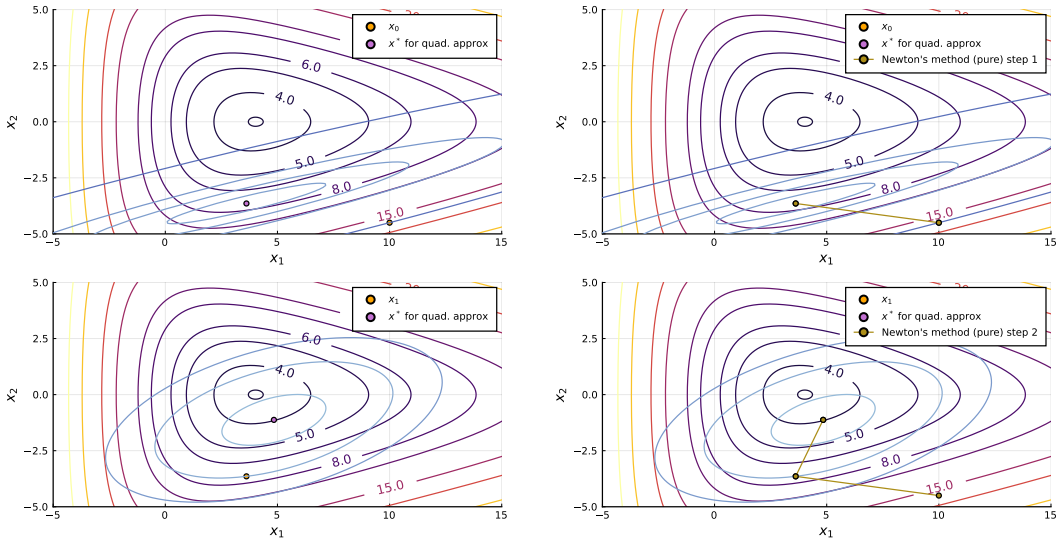


Figure 7: The calculation of the direction $d = x^* - x_0$ in the first two iterations of the Newton's method with step size λ fixed to 1 (the pure Newton's method, in left to right, top to bottom order). Notice in blue the level curves of the quadratic approximation of the function at the current point x_k and how it improves from one iteration to the next.

The Newton's method might diverge if the initial point is too far from the optimum and fixed step sizes (such as $\lambda = 1$) are used, since the minimum of the quadratic approximation and that of the actual target function can become drastically and increasingly disparate. Levenberg-Marquardt method and other trust-region-based variants address convergence issues of the Newton's method. As a general rule, combining the method with an exact line search or a criterion for step-size acceptance that requires improvement (such as employing the Armijo rule for defining the step sizes) is often sufficient for guaranteeing convergence. Figure 8 compares the convergence of the pure Newton's method and a method employing an exact line search.

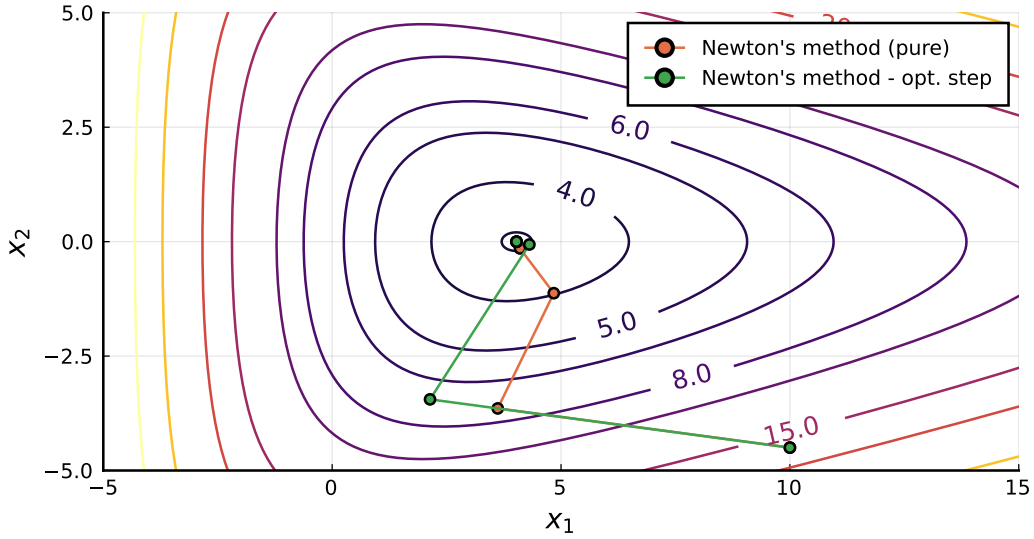


Figure 8: A comparison of the trajectory of both Newton's method variants. Notice that in the method using the exact line search, while the direction $d = x^* - x_0$ is utilised, the step size is larger in the first iteration.

Algorithm 7 presents a pseudocode for the Newton's method. Notice that in line 3, an inversion operation is required. One might be cautious about this operation, since as $\nabla f(x_k)$ tends to zero, the Hessian $H(x_k)$ may become singular, potentially causing numerical instability.

Algorithm 7 Newton's method

1: **initialise.** tolerance $\epsilon > 0$, initial point x_0, iteration count $k = 0$.
2: **while** $\|\nabla f(x_k)\| > \epsilon$ **do**
3: $\quad d = -H^{-1}(x_k)\nabla f(x_k)$
4: $\quad \overline{\lambda} = \operatorname{argmin}_{\lambda \in \mathbb{R}_+} \{f(x_k + \lambda d)\}$
5: $\quad x_{k+1} = x_k + \overline{\lambda} d$
6: $\quad k = k + 1$
7: **end while**
8: **return** x_k

Figure 9 shows the progression of the Newton's method for f with exact and inexact line searches.

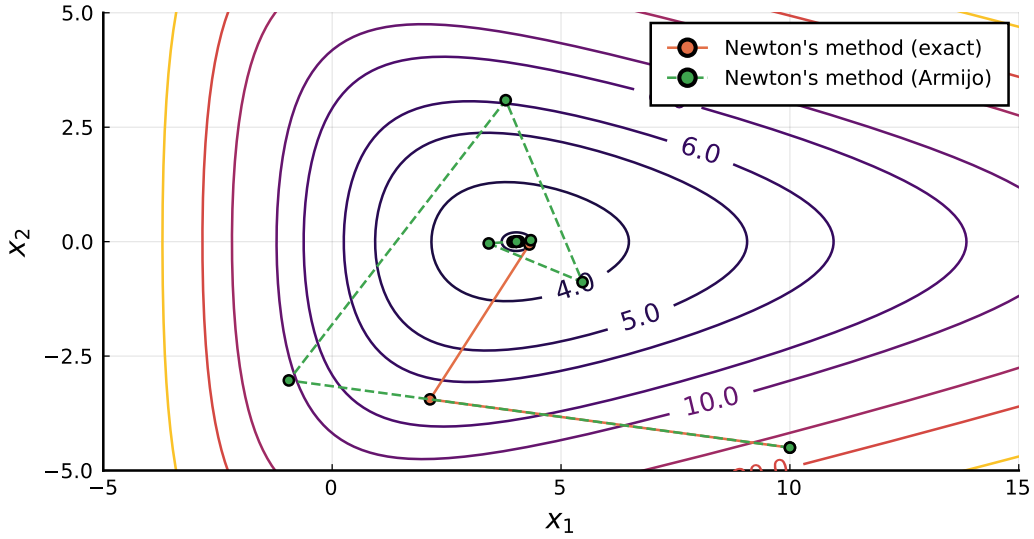


Figure 9: Newton's method applied to f. Convergence is observed in 4 steps using exact line search and 27 using Armijo's rule ($\epsilon = 10^{-5}$)

Lecture 6 - Unconstrained optimisation methods II

Fabricio Oliveira (with modifications by Nuutti Hyönen)

Last update: October 9, 2022

Abstract. We discuss two additional unconstrained optimisation methods that build upon more sophisticated ideas to encode curvature information when deducing search directions. The first is the conjugate gradient method that uses the notion of conjugate directions to devise search directions. The second is a class of methods known as quasi-Newton methods, which rely on approximations for the Hessian matrix that preclude the need of calculating matrix inverses and second-order derivatives. We also discuss the important concepts of complexity, convergence, and conditioning, illustrating how these can influence the performance of an optimisation method.

Outline of this lecture

1 Unconstrained optimisation methods

We will now discuss two variants of the gradient and Newton's methods that try to exploit the computational simplicity of gradient methods while encoding curvature information like Newton's method, but without explicitly relying on second-order derivatives (i.e., Hessian matrices).

1.1 Conjugate gradient method

The conjugate gradient method uses the notion of *conjugacy* to guide the search for optimal solutions. The original motivation for the method comes from quadratic problems, for which one can use conjugacy to separate the search for the optimum of $f : \mathbb{R}^n \to \mathbb{R}$ into n exact steps.

1.1.1 The concept of conjugacy

Let us first define the concept of conjugacy.

Definition 1. *Let H be an $n \times n$ symmetric matrix. The vectors $d_1, \ldots, d_k$ are called H-conjugate if they are linearly independent and $d_i^\top H d_j = 0$ for all $i, j = 1, \ldots, k$ such that $i \neq j$.*

Definition 1 implicitly includes the assumption that the vectors $d_1, \ldots, d_n$ are nonzero because a set of vectors cannot be linearly independent if any of them is the zero vector. Notice also that H-conjugacy (or simply conjugacy) is a generalisation of orthogonality under the linear transformation imposed by the matrix H. In particular, orthogonal vectors are H-conjugate for $H = I$. Figure 1 shows examples of H-conjugate vectors d_1 and d_2, with the considered H being the Hessian of the convex quadratic function shown in the background. Observe that using d_1 and d_2 as consecutive line search directions would take us to the minimiser of the quadratic function in two steps.

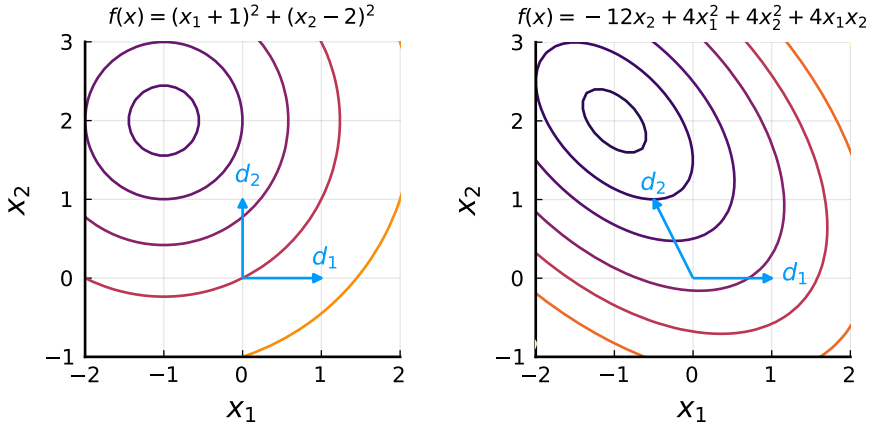


Figure 1: d_1 and d_2 are H-conjugates; on the left, $H = I$.

More generally, one can use H-conjugate directions to find in (at most) n steps the minimiser for a quadratic function $f(x) = c^\top x + \frac{1}{2}x^\top Hx$, where $H \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix, implying that f is strictly convex. Suppose we know directions $d_1, \ldots, d_n$ that are H-conjugate. Then, given an initial point $x_0 \in \mathbb{R}^n$, any point $x \in \mathbb{R}^n$ can be expressed as $x = x_0 + \sum_{j=1}^n \lambda_j d_j$ since H-conjugate vectors are by definition linearly independent, and thus $x - x_0$ can be given as their linear combination. Moreover, the initial point x_0 itself can be written as a similar linear combination, i.e., as $x_0 = \sum_{j=1}^n \mu_j d_j$ for some coefficients $\mu_1, \ldots, \mu_n \in \mathbb{R}$.

We can then reformulate $f(x)$ as a function of the "step size vector" $\lambda \in \mathbb{R}^n$, i.e.,

$$\begin{aligned}
f(x) = F(\lambda) &= c^\top \Big(x_0 + \sum_{j=1}^n \lambda_j d_j\Big) + \frac{1}{2}\Big(x_0 + \sum_{j=1}^n \lambda_j d_j\Big)^\top H\Big(x_0 + \sum_{k=1}^n \lambda_k d_k\Big) \\
&= c^\top \Big(\sum_{j=1}^n (\mu_j + \lambda_j) d_j\Big) + \frac{1}{2}\Big(\sum_{j=1}^n (\mu_j + \lambda_j) d_j\Big)^\top H\Big(\sum_{k=1}^n (\mu_k + \lambda_k) d_k\Big) \\
&= \sum_{j=1}^n (\mu_j + \lambda_j) c^\top d_j + \frac{1}{2}\sum_{j=1}^n \sum_{k=1}^n (\mu_j + \lambda_j)(\mu_k + \lambda_k) d_j^\top H d_k \\
&= \sum_{j=1}^n (\mu_j + \lambda_j) c^\top d_j + \frac{1}{2}\sum_{j=1}^n (\mu_j + \lambda_j)^2 d_j^\top H d_j \\
&= \sum_{j=1}^n \big((\mu_j + \lambda_j) c^\top d_j + \frac{1}{2}(\mu_j + \lambda_j)^2 d_j^\top H d_j\big),
\end{aligned}$$

where the penultimate step is a consequence of $d_1, \ldots, d_n$ being H-conjugate. This reformulation reveals an important property that having conjugate directions $d_1, \ldots, d_n$ allows us to explore: separability. Indeed,

$$F(\lambda) = \sum_{j=1}^n F_j(\lambda_j), \tag{1}$$

where

$$F_j(\lambda_j) = (\mu_j + \lambda_j) c^\top d_j + \frac{1}{2}(\mu_j + \lambda_j)^2 d_j^\top H d_j,$$

and the coefficients $\mu_1, \ldots, \mu_n$ are defined by the initial point x_0. Obviously, the minimiser for F — and thus also for the original f — can be obtained by separately minimising each of F_j with respect to the corresponding coefficient λ_j. What is slightly less obvious, but equally important, is that this minimiser could also be deduced via performing n line searches for the original f starting from x_0 and moving in turns into the directions $d_1, \ldots, d_n$.

To demonstrate this, assume that we start the minimisation process at x_0 and perform a line search for f in the direction d_1. This means that we are considering points of the form $x_0 + \lambda_1 d_1$, where $\lambda_1 \in \mathbb{R}$. Comparing with (1) and the definition of the function F reveals that we are looking for $\overline{\lambda}_1 \in \mathbb{R}$ that minimises

$$\theta_1(\lambda_1) = F_1(\lambda_1) + \sum_{j=2}^n F_j(0). \tag{2}$$

As the latter term on the right-hand side of (2) is independent of λ_1, it is enough to minimize the first term, i.e., F_1. After achieving this, one ends up at the point $x_1 = x_0 + \overline{\lambda}_1 d_1$, and performing next a line search for F from x_1 to the direction d_2 corresponds to minimising

$$\theta_2(\lambda_2) = F_1(\overline{\lambda}_1) + F_2(\lambda_2) + \sum_{j=3}^{n} F_j(0),$$

where the second term is the only one that depends on λ_2, and thus the second step size $\overline{\lambda}_2$ is the minimiser of F_2. At this point the optimization procedure for f is at $x_2 = x_1 + \overline{\lambda}_2 d_2 = x_0 + \overline{\lambda}_1 d_1 + \overline{\lambda}_2 d_2$, and a line search in the direction d_3 corresponds to minimising

$$\theta_3(\lambda_3) = \sum_{j=1}^{2} F_1(\overline{\lambda}_j) + F_3(\lambda_3) + \sum_{j=4}^{n} F_j(0),$$

that is, the line search produces the minimiser of F_3, say, $\overline{\lambda}_3$. This procedure can obviously be repeated n times after which the global search is at the point

$$\overline{x} = x_0 + \sum_{j=1}^{n} \overline{\lambda}_j d_j.$$

where $\overline{\lambda}_1, \ldots, \overline{\lambda}_n$ have been obtained via consecutive line searches for f, but they are also the minimisers of $F_1, \ldots, F_n$, respectively. In particular, due to (1), the vector $(\overline{\lambda}_1, \ldots, \overline{\lambda}_n) \in \mathbb{R}^n$ is the minimiser of F, and thus $\overline{x} \in \mathbb{R}^n$ is the global minimiser for the original quadratic function f.

As H was assumed to be positive definite, the first-order condition for f is necessary and sufficient for optimality. The same also applies to F_j, $j = 1, \ldots, n$, since they are unidimensional quadratic functions with the second derivatives $d_j^\top H d_j > 0$, respectively. We can thus calculate the minimisers $\overline{\lambda}_j$, $j = 1, \ldots, n$, from

$$F_j'(\overline{\lambda}_j) = c^\top d_j + (\mu_j + \overline{\lambda}_j) d_j^\top H d_j = c^\top d_j + x_0^\top H d_j + \overline{\lambda}_j d_j^\top H d_j = 0, \tag{3}$$

where we used the fact that

$$x_0^\top H d_j = \Big(\sum_{k=1}^{n} \mu_k d_k \Big)^\top H d_j = \mu_j d_j^\top H d_j$$

due to the H-conjugacy of $d_1, \ldots, d_n$. Solving (3) for λ_j gives

$$\overline{\lambda}_j = -\frac{c^\top d_j + x_0^\top H d_j}{d_j^\top H d_j}, \qquad j = 1, \ldots, n. \tag{4}$$

By mimicking the interpretation as sequential line searches discussed above, this result can be used to devise an iterative method that produces the optimal solution for a strictly convex quadratic

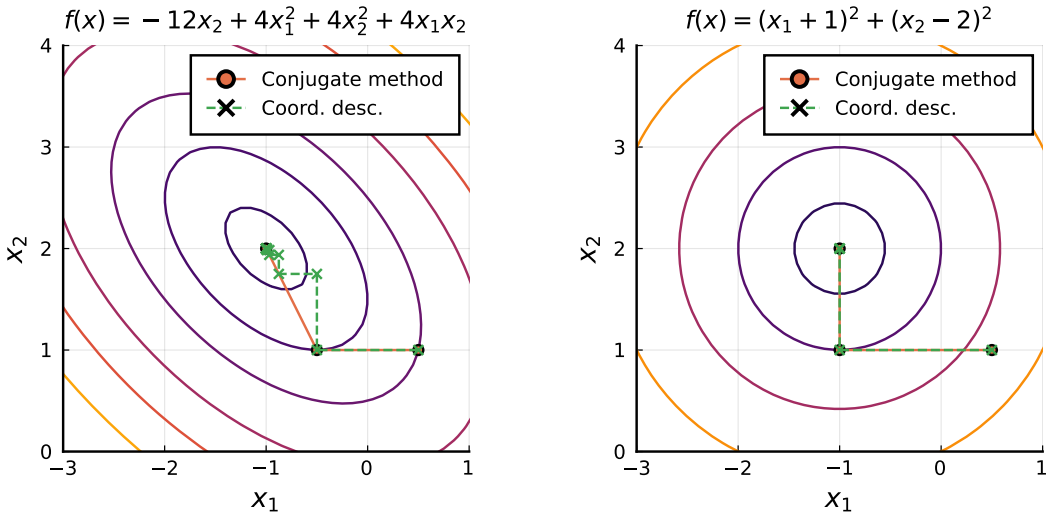


Figure 2: Optimising f with the conjugate method and coordinate descent (left). For $H = I$, both methods converge in two steps (right)

function in (at most) n iterations. For an initial point x_0 and a collection of H-conjugate directions $d_1, \ldots, d_n$, the method consists of successively executing the step (cf. (4))

$$x_{k+1} = x_k + \overline{\lambda}_{k+1} d_{k+1}, \text{ where } \overline{\lambda}_{k+1} = -\frac{c^\top d_{k+1} + x_k^\top H d_{k+1}}{d_{k+1}^\top H d_{k+1}} = -\frac{c^\top d_{k+1} + x_0^\top H d_{k+1}}{d_{k+1}^\top H d_{k+1}}. \quad (5)$$

The equivalence between the two forms for $\overline{\lambda}_{k+1}$ is explained by

$$x_k^\top H d_{k+1} = \Big(x_0 + \sum_{j=1}^{k} \overline{\lambda}_k d_k\Big)^\top H d_{k+1} = x_0^\top H d_{k+1},$$

which follows from the H-conjugacy of $d_1, \ldots, d_n$.

Notice the resemblance this method holds with coordinate descent. In case $H = I$, the coordinate directions given by the standard basis vectors $\mathrm{e}_1, \ldots, \mathrm{e}_n$ are H-conjugate and thus, the coordinate descent method converges in n iterations. Figure 2 illustrates this behaviour for $n = 2$. Notice that, on the left, the conjugate method converges in exactly two iterations, while coordinate descent takes several steps before finding the minimum. On the right, the methods become equivalent since the coordinate directions are conjugate for $H = I$. Observe also that the gradient method would converge in a single step for the right-hand image of Figure 2, whereas it would suffer from zig-zagging for the setting in the left-hand image.

1.1.2 Generating conjugate directions

As demonstrated in the preceding section, the availability of n conjugate directions for the Hessian of a convex quadratic function enables minimising it in n steps without having to explicitly operate with the inverse of the Hessian as is the case for Newton's method. From the standpoint of introducing an actual algorithm based on this observation, the missing part is generating H-conjugate directions. This can be done by resorting to an adaptation of the *Gram–Schmidt* procedure, typically employed to generate orthonormal bases; one can interpret the search for H-conjugate directions as a task to build orthogonal basis with respect to the inner product defined by the positive definite matrix H. Notice that one does not actually require the availability of all n conjugate directions at the start of the iteration (5), but it suffices to introduce a new nonzero direction d_{k+1} that is H-conjugate to all previous (H-conjugate) search direction $d_1, \ldots, d_k$ only when executing the step described in (5), that is, one does not need to know the 'future directions' $d_{k+2}, \ldots, d_n$ at that point.

We intend to build a collection of conjugate directions $d_1, \ldots, d_n$ based on knowing a set of linearly independent vectors $\xi_1, \ldots, \xi_n$. The method proceeds as follows:

1. Set $d_1 = \xi_1$ as the first direction.
2. At a given iteration $1 \leq k \leq n$, we aim to represent the new direction d_{k+1} as a linear combination of a 'new' ξ_{k+1} and the previous H-conjugate directions $d_1, \ldots, d_k$:

$$d_{k+1} = \xi_{k+1} + \sum_{l=1}^{k} \alpha_k^l d_l, \quad \text{where } \alpha_k^1, \ldots, \alpha_k^k \in \mathbb{R}. \tag{6}$$

   Notice that this construction implicitly implies that $\sum_{l=1}^{k} \alpha_k^l d_l$ belongs to $\operatorname{span}\{\xi_1, \ldots, \xi_k\}$, and thus d_{k+1} must be nonzero since ξ_{k+1} does not belong to $\operatorname{span}\{\xi_1, \ldots, \xi_k\}$ by assumption.

3. To have H-conjugacy between d_{k+1} and all previous search directions, it must hold

$$\begin{aligned} {d_{k+1}}^\top H d_i &= {\xi_{k+1}}^\top H d_i + \left(\sum_{l=0}^{k} \alpha_k^l d_l \right)^\top H d_i \\ &= {\xi_{k+1}}^\top H d_i + \sum_{l=0}^{k} \alpha_k^l d_l^\top H d_i \\ &= {\xi_{k+1}}^\top H d_i + \alpha_k^i d_i^\top H d_i = 0, \qquad i = 1, \ldots, k, \end{aligned} \tag{7}$$

   where we used the assumed H-conjugacy of the previous search directions. The value of α_k^i can now be solved from (7),

$$\alpha_k^i = -\frac{\xi_{k+1}^\top H d_i}{d_i^\top H d_i}, \tag{8}$$

   for all $i = 1, \ldots, k$.

1.1.3 Gradients and conjugate directions

The next piece required for developing a method that could exploit conjugacy is choosing a collection of linearly independent vectors $\xi_1, \ldots, \xi_n$ that is used for generating conjugate directions. Observe that the construction of the $(k+1)$:th direction d_{k+1} in (6) actually only requires the knowledge of the first $k+1$ of these linearly independent vectors — or even more precisely, one only needs a new vector ξ_{k+1} that does not belong to $\operatorname{span}(d_1, \ldots, d_k)$ ensuring that $d_{k+1} \neq 0$. In the setting of developing an unconstrained optimisation method, the steepest descent directions $-\nabla f(x_0), \ldots, -\nabla f(x_k)$ at the previous iterates can play the part of $\xi_1, \ldots, \xi_{k+1}$, which is a key result in Theorem 2.

Theorem 2. *Let $f(x) = c^\top x + \frac{1}{2} x^\top H x$, where $H \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix. Let $d_1, \ldots, d_n$ be H-conjugate, let $x_0 \in \mathbb{R}^n$ be an arbitrary starting point, and let $x_1, \ldots, x_n$ be defined by the iteration* (5)*. Then, the following hold:*

1. *For each $k = 1, \ldots, n$, the iterate x_k solves the problem*

$$\text{min. } \{f(x) : x \in \{x_0\} + \operatorname{span}(d_1, \ldots, d_k)\}. \tag{9}$$

2. *$\nabla f(x_k)^\top d_j = 0$ for $j = 1, \ldots, k$.*

Proof. The first part of the theorem states that x_k, $k = 1, \ldots, n$, minimises f among the vectors that are of the form $x_0 + \sum_{j=1}^k \lambda_k d_k$ for some $\lambda_1, \ldots, \lambda_k \in \mathbb{R}$. However, recalling (1) and the definition of F, we have

$$f\Big(x_0 + \sum_{j=1}^k \lambda_k d_k\Big) = \sum_{j=1}^k F_j(\lambda_j) + \sum_{j=k+1}^n F_j(0),$$

which is minimised by choosing $\lambda_j = \overline{\lambda}_j$ for $j = 1, \ldots, k$ due to the definition of $\overline{\lambda}_j$ as the minimiser of F_j for $j = 1, \ldots, n$. This means that the vector

$$\overline{x}_k = x_0 + \sum_{j=1}^k \overline{\lambda}_j d_j \tag{10}$$

is the solution of problem (9). However, it is easy to see that this is the same vector as x_k defined by the iteration (5), i.e., $x_k = \overline{x}_k$.

To prove the second part of the claim, notice first of all that

$$\nabla f(x) = c + Hx, \qquad x \in \mathbb{R}^n.$$

Hence, (10) and the symmetry of H give

$$\nabla f(x_k)^\top d_j = c^\top d_j + x_0^\top H d_j + \sum_{l=1}^{k} \overline{\lambda}_l d_l^\top H d_j = c^\top d_j + x_0^\top H d_j + \overline{\lambda}_j d_j^\top H d_j \quad \text{for } j = 1, \dots k,$$

where we once again used the H-conjugacy of $d_1, \dots, d_k$. By virtue of (3), this means that

$$\nabla f(x_k)^\top d_j = F_j'(\overline{\lambda}_j) = 0 \quad \text{for } j = 1, \dots, k,$$

which completes the proof. □

The first part of Theorem 2 states that x_k is the minimiser of the target function f on the k-dimensional hyperplane spanned by the conjugate directions $d_1, \dots, d_k$ and passing through the "basis point" x_0. This phenomenon is sometimes called the *expanding manifold property*, since at each iteration the considered hyperplane expands in one independent (conjugate) direction. When $k = n$, the hyperplane becomes the whole of $\mathbb{R}^n$ due to the linear independence of $d_1, \dots, d_n$, which is compatible with the iteration converging to a global minimiser in at most n steps.

The second part of the theorem states that the gradient of f at the latest iterate x_k is orthogonal to all previously employed search directions $d_1, \dots, d_k$. In particular, this means that either $-\nabla f(x_k)$ is zero or it cannot be given as a linear combination of $d_1, \dots, d_k$. In other words, either the iteration has already converged to the minimiser, i.e. $\nabla f(x_k) = 0$, or $-\nabla f(x_k)$ can be used in the role of ξ_{k+1} in the construction of d_{k+1} in (6).

Remark. *If one employes the steepest descent directions $-\nabla f(x_0), \dots, -\nabla f(x_{n-1})$ in the role of the linearly independent auxiliary vectors $\xi_1, \dots, \xi_n$ when constructing the conjugate search directions $d_1, \dots, d_n$ as described in Section 1.1.2, it can be shown relatively easily that the vectors $Hd_1, \dots, Hd_{k-1}$ belong to $\operatorname{span}(d_1, \dots, d_k)$ for any $k = 1, \dots, n$ (unless the algorithm converges prematurely, i.e., $\nabla f(x_k) = 0$ for some $k < n$). This is related to the so-called* Krylov subspaces *and explains why the conjugate gradient method intoduced in the next subsection belongs to the family of* Krylov subspace methods. *In particular, with this choice $Hd_1, \dots, Hd_{k-1}$ are orthogonal to $\xi_{k+1} = -\nabla f(x_k)$ due to the second part of Theorem 2. In terms of* (8)*, this means that*

$$\alpha_k^i = 0 \qquad \text{for } i = 1, \dots, k-1,$$

an the update rule in (6) *becomes simply*

$$d_{k+1} = \xi_{k+1} + \alpha_k d_k = -\nabla f(x_k) + \alpha_k d_k, \tag{11}$$

with

$$\alpha_k := \alpha_k^k = -\frac{\xi_{k+1}^\top H d_k}{d_k^\top H d_k} = \frac{\nabla f(x_k)^\top H d_k}{d_k^\top H d_k}. \tag{12}$$

In other words, the new H-conjugate search direction is given as a certain linear combination of the

steepest descent direction at the current iterate $-\nabla f(x_k)$ *and the previous search direction* d_k *— unless* $\nabla f(x_k) = 0$*, i.e., the algorithm has already converged, and it can be terminated.*

1.1.4 Conjugate gradient method

We have now all parts required for describing the *conjugate gradient method*. The method uses the gradients $\nabla f(x_k)$ as linearly independent vectors to generate conjugate directions, which are then used as search directions d_k.

Let us summarise the observations presented above. The method operates by generating a sequence of iterates

$$x_{k+1} = x_k + \overline{\lambda}_{k+1} d_{k+1}, \qquad k = 0, \ldots, k-1, \tag{13}$$

with the initialisation $d_1 = -\nabla f(x_0)$ and $\overline{\lambda}_{k+1}$ denoting the optimal step size in the direction d_{k+1}. Given a current iterate x_k with $-\nabla f(x_k) \neq 0$, we use (11) and (12) to generate the new conjugate direction d_{k+1}, i.e.,

$$d_{k+1} = -\nabla f(x_k) + \alpha_k d_k, \quad \text{with } \alpha_k = \frac{\nabla f(x_k)^\top H d_k}{d_k^\top H d_k}. \tag{14}$$

Since $\nabla f(x_k) - \nabla f(x_{k-1}) = H(x_k - x_{k-1}) = \overline{\lambda}_k H d_k$ and $d_k = -\nabla f(x_{k-1}) + \alpha_{k-1} d_{k-1}$, the formula for α_k can alternatively be written as:

$$\begin{aligned}
\alpha_k &= \frac{\nabla f(x_k)^\top (\nabla f(x_k) - \nabla f(x_{k-1}))}{(-\nabla f(x_{k-1}) + \alpha_{k-1} d_{k-1})^\top (\nabla f(x_k) - \nabla f(x_{k-1}))} \\
&= \frac{\|\nabla f(x_k)\|^2 - \nabla f(x_k)^\top \nabla f(x_{k-1})}{-\nabla f(x_{k-1})^\top \nabla f(x_k) + \|\nabla f(x_{k-1})\|^2 + \alpha_{k-1} d_{k-1}^\top \nabla f(x_k) - \alpha_{k-1} d_{k-1}^\top \nabla f(x_{k-1})} \\
&= \frac{\|\nabla f(x_k)\|^2 - \nabla f(x_k)^\top \nabla f(x_{k-1})}{\|\nabla f(x_{k-1})\|^2 - \nabla f(x_k)^\top \nabla f(x_{k-1})}
\end{aligned}$$

where the last relation follows from the second part of Theorem 2. Moreover, by the update rule (14), we have $\nabla f(x_{k-1}) = \alpha_{k-1} d_{k-1} - d_k$, which is also orthogonal to $\nabla f(x_k)$ due to Theorem 2. Hence, the formula for α_k simplifies into

$$\alpha_k = \frac{\|\nabla f(x_k)\|^2}{\|\nabla f(x_{k-1})\|^2}, \tag{15}$$

for $k = 1, \ldots, n$.

For a convex quadratic problem, i.e., when $f(x) = c^\top x + \frac{1}{2} x^\top H x$ for a positive definite H as has been assumed all the time above, the optimal step size $\overline{\lambda}_{k+1}$ for the line search in the direction d_{k+1}

in (13) can be written explicitly as indicated in (5):

$$\begin{aligned}\overline{\lambda}_{k+1} &= -\frac{c^\top d_{k+1} + x_k^\top H d_{k+1}}{d_{k+1}^\top H d_{k+1}} = -\frac{\nabla f(x_k)^\top d_{k+1}}{d_{k+1}^\top H d_{k+1}} \\ &= -\frac{\nabla f(x_k)^\top (-\nabla f(x_k) + \alpha_k d_k)}{d_{k+1}^\top H d_{k+1}} = \frac{\|\nabla f(x_k)\|^2}{d_{k+1}^\top H d_{k+1}} \end{aligned} \tag{16}$$

by virtue of Theorem 2. However, this line search step is often left unsolved in the formulation of the conjugate gradient method for optimisation:[1] Although the deductions presented above do not directly hold for more general target functions, one can anyway apply the method to other types of functions as well — often with good success. However, for a non-quadratic function one should typically apply some line search method for deducing the optimal step size instead of directly resorting to (16).

Algorithm 1 summarises the conjugate gradient method for a general target function f.

Algorithm 1 Conjugate gradient method

initialise. tolerance $\epsilon > 0$, initial point x_0, $k = 1$
while $\|\nabla f(x_k)\| > \epsilon$ **do**
    $y_0 = x_{k-1}$
    $d_1 = -\nabla f(y_0)$
    for $j = 1, \ldots, n$ **do**
        $\overline{\lambda}_j = \operatorname{argmin}_{\lambda \geq 0} \{f(y_{j-1} + \lambda d_j)\}$
        $y_j = y_{j-1} + \overline{\lambda}_j d_j$
        $d_{j+1} = -\nabla f(y_j) + \alpha_j d_j$, where $\alpha_j = \frac{\|\nabla f(y_j)\|^2}{\|\nabla f(y_{j-1})\|^2}$.
    end for
    $x_k = y_n$, $k = k + 1$
end while
return x_k.

The conjugate gradient method using (15) is due to Fletcher and Reeves. An alternative version of the method uses

$$\alpha_k = \frac{\nabla f(x_k)^\top (\nabla f(x_k) - \nabla f(x_{k-1}))}{\|\nabla f(x_{k-1})\|},$$

which is known for having better numerical properties for solving problems that are not quadratic.

Figure 3 illustrates the behaviour of the conjugate gradient method when applied to minimising $f(x) = e^{(-(x_1-3)/2)} + e^{((4x_2+x_1)/10)} + e^{((-4x_2+x_1)/10)}$ using both exact and inexact line searches. If $f : \mathbb{R}^n \to \mathbb{R}$ is a quadratic function, then the method is guaranteed to converge in at most n iterations with an exact line search, i.e., (16). However, the method can be applied to any differentiable function f, in which setting it behaves as successively minimising quadratic approximations of f — in a similar

[1] The conjugate gradient method is also used for solving (sparse and large) sets of linear equations.

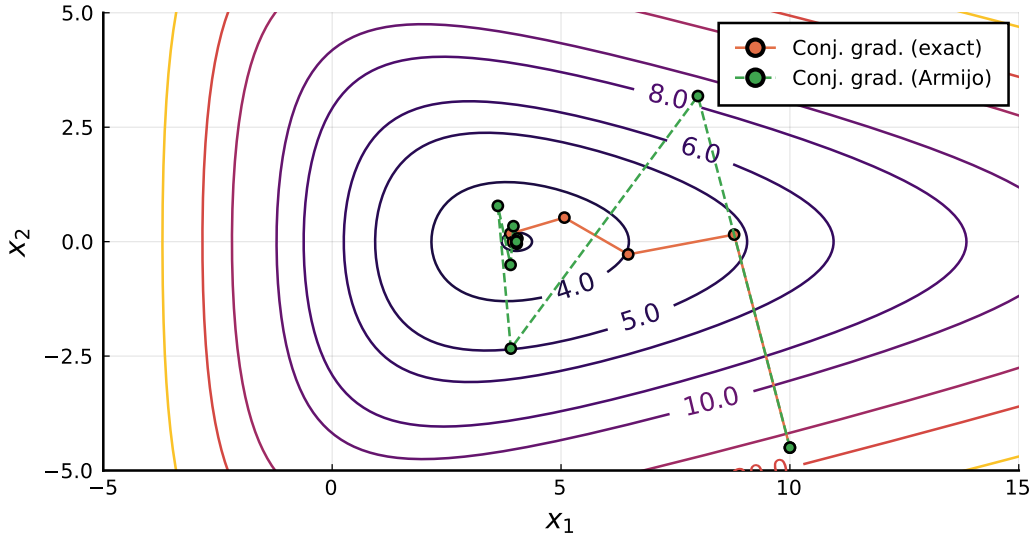


Figure 3: Conjugate gradient method applied to f. Convergence is observed in 24 steps using exact line search and 28 using Armijo's rule ($\epsilon = 10^{-6}$)

fashion to that of Newton's method, but without requiring second-order (Hessian) information, which is the most demanding aspect associated with Newton's method. When employed to non-quadratic functions, the process of deducing conjugate directions is restarted at the current point x_k after n steps (represented in the loop staring in Line 5 in Algorithm 1). Often, it would actually be advantageous to restart the computation of conjugate directions much earlier than after a full set of n steps in the inner loop when non-quadratic functions are considered, but we do not stress this matter further in these lecture notes.

Equation (14) exposes an important property of the conjugate gradient method. In general, the employment of second-order terms is helpful for the optimisation method because it encodes *curvature information* on the definition of the search direction. The conjugate gradient method is also capable of encoding curvature information, not by using Hessians, but by taking a linear combination of the current direction of steepest descent $-\nabla f(x_{k+1})$ and the previous direction d_k, which compensates for the curvature encoded in H (which is the Hessian of the quadratic approximation).

1.2 Quasi-Newton methods

Quasi-Newton method is a term referring to a class of methods that use approximations for the inverse of the Hessian of f at x_k, $H^{-1}(x_k)$, that do not explicitly require second-order information nor expensive inversion operations. To be more precise, a quasi-Newton method considers search directions

$$d_{k+1} = -D_k \nabla f(x_k), \qquad k = 1, 2, \ldots,$$

where D_k acts as an approximation for the inverse of the Hessian $H^{-1}(x_k)$. The method thus tries to mimic Newton's method in its choice of the search direction, which explains its name.

1.2.1 Basic idea with a quadratic target function

In this section, we consider a quadratic target function $f(x) = c^\top x + \frac{1}{2}x^\top Hx$, with $H \in \mathbb{R}^{n\times n}$ symmetric and positive definite, since one can prove the convergence of the approximate (inverse) Hessians in a quasi-Newton method for quadratic functions. However, the quasi-Newton methods themselves are applicaple to minimisation of more general functions, and thus we will also consider more general target functions in the next section.

To compute D_k, we use (approximate) curvature information encoded in the previous search directions, in an attempt to approximate second-order derivatives. To that end, let us fix some notation:

$$\begin{aligned} p_k &= \overline{\lambda}_k d_k = x_k - x_{k-1} \\ q_k &= \nabla f(x_k) - \nabla f(x_{k-1}) = H(x_k - x_{k-1}) = Hp_k, \end{aligned}$$

where $\overline{\lambda}_k$ denotes the chosen step size in the direction d_k to move from the iterate x_{k-1} to the next one x_k. Note that the second equality on the last line is valid because the Hessian of the considered f is a constant matrix. Starting from an initial symmetric positive definite approximation for the inverse of the Hessian D_0 (e.g., $D_0 = I$), a quasi-Newton method proceeds by successively updating

$$D_k = D_{k-1} + C_k, \tag{17}$$

with C_k being symmetric and such that it only uses the information in p_k, q_k and D_{k-1}. Furthermore, we are looking for such an update scheme that D_n equals H^{-1} unless $\nabla f(x_k) = 0$ for some $k < n$, in which case the iteration can be terminated before performing n steps. Recall that Newton's method converges in one iteration for a convex quadratic target function, and hence the condition $D_n = H^{-1}$ guarantees the convergence of a quasi-Newton method for such a target function in $n + 1$ steps.

To achieve the desired propery of D_n, we require that p_j, $j = 1, \ldots, k$, are eigenvectors of $D_k H$ with unit eigenvalue, that is,

$$D_k H p_j = p_j, \text{ for } j = 1, \ldots, k. \tag{18}$$

In other words, the restriction of $D_k H$ onto the subspace $\operatorname{span}(p_1, \ldots, p_k)$ is required to be the identity operator (this interpretation is concretised below by (19) for $k = n$). Note also that $\operatorname{span}(p_1, \ldots, p_k) = \operatorname{span}(d_1, \ldots, d_k)$ unless one of the previous step sizes $\overline{\lambda}_j$ is zero, i.e., $p_j = 0$.

To simplify the analysis, let us assume that our algorithm produces linearly independent search directions $d_1, \ldots, d_n$ and that the corresponding step sizes $\overline{\lambda}_j$ are nonzero; for the quasi-Newton

methods introduced below, it could be shown that d_{k+1} being representable as a linear combination of the previous search directions would mean that x_{k+1} is the minimiser of our quadratic target function f and that $\overline{\lambda}_{k+1} = 0$ cannot occur (unless $\nabla f(x_k) = 0$). Hence, either our assumption of linear independence of the n search directions and nonzero step sizes is valid, or the algorithm converges in fewer than n iterations (for the quadratic target function). For $k = n$, the requirement (18) is equivalent to

$$D_n H\Big(\sum_{l=1}^{n} \mu_l p_l\Big) = \sum_{l=1}^{n} \mu_l p_l \tag{19}$$

for all $\mu_1, \ldots, \mu_m \in \mathbb{R}$. Since the step sizes are assumed to be nonzero, the update vectors $p_1, \ldots, p_n$ inherit their linear independence from the assumed linear independence of $d_1, \ldots, d_n$, and thus $p_1, \ldots, p_n$ form a basis for $\mathbb{R}^n$. Hence, (19) means $D_n H y = y$ for any $y \in \mathbb{R}^n$, i.e., $D_n = H^{-1}$.

Because by definition

$$H[p_1, \ldots, p_n] = [Hp_1, \ldots, Hp_n] = [q_1, \ldots, q_n],$$

it also holds that

$$[p_1, \ldots, p_n] = H^{-1}[q_1, \ldots, q_n] = D_n[q_1, \ldots, q_n],$$

and thus,

$$H^{-1} = D_n = [p_1, \ldots, p_n][q_1, \ldots, q_n]^{-1}. \tag{20}$$

Notice that under our assumptions, the update vectors $p_1, \ldots, p_n$ are linearly independent, and thus $Hp_1, \ldots, Hp_n$, i.e., $q_1, \ldots, q_n$, are also linearly independent as images of linearly independent vectors under an invertible linear transform. Hence, $[q_1, \ldots, q_n] \in \mathbb{R}^{n \times n}$ is invertible and the formula on the right-hand side of (20) makes sense. The condition (20) is called the *secant condition* as a reference to the approximation to the second-order derivative.

1.2.2 BFGS and DFP methods

There are many ways to define the update C_k in (17) so that (18) is satisfied. The *Davidon–Fletcher–Powell* (DFP) method is one classical quasi-Newton method, and it employs the update rule

$$C_k = C_k^{\mathrm{DFP}} := \frac{p_k p_k^\top}{p_k^\top q_k} - \frac{D_{k-1} q_k q_k^\top D_{k-1}}{q_k^\top D_{k-1} q_k}. \tag{21}$$

One can verify that $D_k = D_{k-1} + C_k^{\mathrm{DFP}}$ satisfies conditions (18) by showing (i) that $(D_{k-1} + C_k^{\mathrm{DFP}})H$ inherits $p_1, \ldots, p_{k-1}$ as eigenvectors with eigenvalue 1 from $D_{k-1}H$, i.e., $p_1, \ldots, p_{k-1}$ belong to the nullspace of $C_k^{\mathrm{DFP}}H$, and (ii) that $(D_{k-1} + C_k^{\mathrm{DFP}})Hp_k = p_k$, i.e., $C_k^{\mathrm{DFP}}Hp_k =$

$p_k - D_{k-1}Hp_k$. That is,

(i) $C_k^{\mathrm{DFP}} Hp_j = \dfrac{p_k p_k^\top Hp_j}{p_k^\top q_k} - \dfrac{D_{k-1}q_k p_k^\top H D_{k-1}Hp_j}{q_k^\top D_{k-1}q_k} = 0, \quad \text{for } j = 1, \ldots, k-1;$

(ii) $C_k^{\mathrm{DFP}} Hp_k = C_k^{\mathrm{DFP}} q_k = \dfrac{p_k p_k^\top q_k}{p_k^\top q_k} - \dfrac{D_{k-1}q_k q_k^\top D_{k-1}q_k}{q_k^\top D_{k-1}q_k} = p_k - D_{k-1}q_k = p_k - D_{k-1}Hp_k.$

Demonstrating that condition (i) holds is not quite straightforward, whereas (ii) follows immediately as seen above. What is more, $D_k = D_{k-1} + C_k^{\mathrm{DFP}}$ is known to be symmetric and positive definite if D_{k-1} is symmetric and positive definite and $p_k^\top q_k > 0$ (i.e., for our quadratic target function $p_k^\top q_k = p_k^\top Hp_k > 0$, which holds unless $p_k = 0$ meaning that the algorithm has converged).

Let us then consider a general twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$. Arguably a more standard way of building quasi-Newton type update rules for an approximation of the inverse Hessian of f is to only consider the latest secant equation

$$D_k q_k = (D_{k-1} + C_k)q_k = p_k, \tag{22}$$

where $p_k = x_k - x_{k-1}$ and $q_k = \nabla f(x_k) - \nabla f(x_{k-1}) = H(\hat{x})p_k$ for some $\hat{x}$ on the line segment between x_{k-1} and x_k due to the mean value theorem. Obviously, (22) is underdetermined, i.e., there are inifinitely many ways of choosing C_k. Hence, (22) is accompanied by a requirement that C_k is symmetric and *as small as possible* in some matrix norm. The rationale behind this approach is that one wants to incorporate the information on the curvature of f encoded in the pair p_k, q_k into the algorithm, but at the same time one wants to be conservative and modify the previous approximation of the inverse Hessian, D_{k-1}, as little as possible. (Observe that the Hessian H of a general f is a function of x, and thus, unlike for the quadratic target function, the 'correct' approximation for the Hessian changes as the iteration moves around $\mathbb{R}^n$.)

Choosing different matrix norms for minimising the size of the update C_k results in different update rules. In particular, choosing a certain weighted Frobenius norm leads to the *Broyden–Fletcher–Goldfarb–Shanno* (BFGS) method:

$$C_k = C_k^{\mathrm{BFGS}} = \frac{p_k p_k^\top}{p_k^\top q_k}\left(1 + \frac{q_k^\top D_{k-1}q_k}{p_k^\top q_k}\right) - \frac{D_{k-1}q_k p_k^\top + p_k q_k^\top D_{k-1}}{p_k^\top q_k}, \tag{23}$$

which is usually favored over the DFP method. Like the DFP method, also the BFGS method maintains symmetry and positive definiteness of the approximate inverse Hessian in the sense that D_k is symmetric and positive definite if D_{k-1} is symmetric and positive definite and $p_k^\top q_k > 0$. It is of interest to note that the DFP method could be derived by looking for a symmetric C_k that satisfies (22) and minimises the update in the approximate Hessian, $D_k^{-1} - D_{k-1}^{-1}$, in a certain weighted Frobenius norm. The DFP and BFGS method can thus be considered to be dual to each other: the update in the approximation of the inverse Hessian in the BFGS method is formed by following the same logic that is employed when forming the update for the approximation of the

Hessian itself in the DFP method.

Sometimes it is useful to present the BFGS method as a scheme for updating an approximation for the Hessian instead of the inverse Hessian. If B_{k-1} is the current approximation for the Hessian in the BFGS method, i.e., $B_{k-1} = D_{k-1}^{-1}$, then

$$D_k = B_k^{-1} = \left(B_{k-1} + \overline{C}_k^{\mathrm{BFGS}}\right)^{-1},$$

where

$$\overline{C}_k^{\mathrm{BFGS}} = \frac{q_k q_k^\top}{q_k^\top p_k} - \frac{B_{k-1} p_k p_k^\top B_{k-1}}{p_k^\top B_{k-1} p_k}. \tag{24}$$

The equivalence between the update rules (23) and (24) follows from the *Sherman–Morrison formula* or the *Woodbury matrix identity*. One can observe a resemblance between the update rule for the approximate Hessian in the BFGS method (24) and the update rule for the inverse Hessian in the DFP method (21), which is due to the aforementioned duality of these methods. In the same way, one could write an explicit formula for the additive update in the Hessian in the DFP method by resorting to (21) and the Sherman–Morrison formula — and the resulting update rule would resemble that for the inverse Hessian in the BFGS method (23).

The BFGS and DFP methods are members of the Broyden family of quasi-Newton methods that correspond to the following update in the inverse Hessian:

$$C_k^{\mathrm{B}} = C_k^{\mathrm{DFP}} + \phi \frac{\tau_j v_k v_k^\top}{p_k^\top q_k}, \tag{25}$$

where $v_k = p_k - \left(\frac{1}{\tau_k}\right) D_{k-1} q_k$ and $\tau_k = \frac{q_j^\top D_{k-1} q_k}{p_k^\top q_k}$. If the free parameter $\phi \in [0, 1]$ is chosen to be $\phi = 0$, (25) becomes the DFP method, whereas $\phi = 1$ produces the BFGS method. Compared to the DFP method, the extra term in the Broyden family of updates is designed to help with mitigating numerical difficulties from near-singular approximations. It can be shown that all update schemes in the Broyden family satisfy the quasi-Newton conditions (18) and (20) for a quadratic target function with a positive definite Hessian.

The BFGS method with the additive update for the inverse Hessian (21) is presented in Algorithm 2.

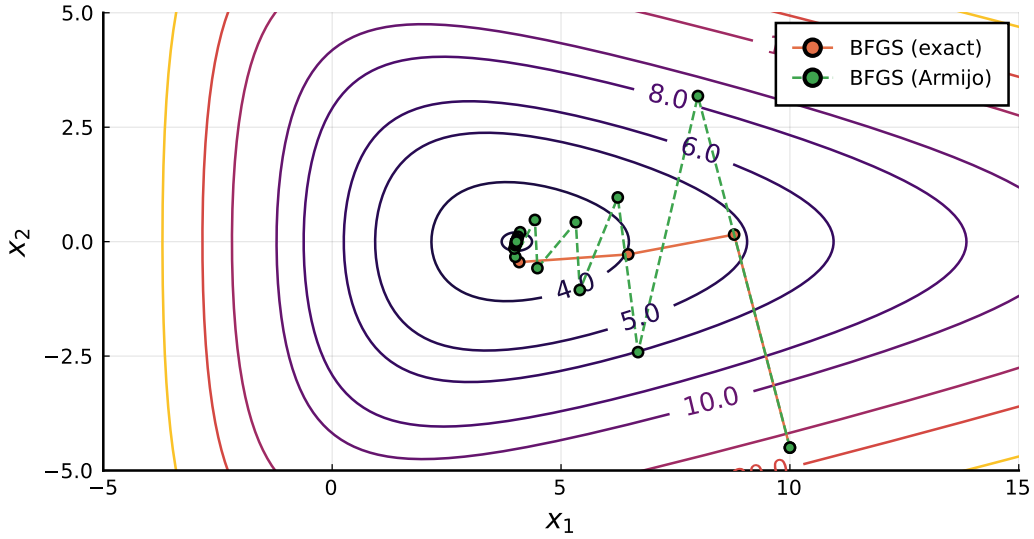


Figure 4: BFGS method applied to $f(x) = e^{(-(x_1-3)/2)} + e^{((4x_2+x_1)/10)} + e^{((-4x_2+x_1)/10)}$. Convergence is observed in 11 steps using exact line search and in 36 steps using Armijo's rule ($\epsilon = 10^{-6}$)

Algorithm 2 BFGS method

1: **initialise.** tolerance $\epsilon > 0$, initial point x_0, initial inverse Hessian D (e.g., $D = I$ or $D = H^{-1}(x_0)$), iteration count $k = 0$.
2: **while** $\|\nabla f(x_k)\| > \epsilon$ **do**
3: $\quad d = -D\nabla f(x_k)$
4: $\quad \overline{\lambda} = \operatorname{argmin}_{\lambda \in \mathbb{R}_+} \{f(x_k + \lambda d)\}$
5: $\quad p = \overline{\lambda} d$
6: $\quad x_{k+1} = x_k + p$
7: $\quad q = \nabla f(x_{k+1}) - \nabla f(x_k)$
8: $\quad D = D + \dfrac{pp^\top}{p^\top q}\left(1 + \dfrac{q^\top D q}{p^\top q}\right) - \dfrac{Dqp^\top + pq^\top D}{p^\top q}$
9: $\quad k = k + 1$
10: **end while**
11: **return** x_k

Figure 4 illustrates the behaviour of the BFGS method when applied to solve

$$f(x) = e^{(-(x_1-3)/2)} + e^{((4x_2+x_1)/10)} + e^{((-4x_2+x_1)/10)}$$

using both exact and inexact line searches. Notice how the combination of imprecisions in the approximation of $H^{-1}(x_k)$ and in the inexact line search turns the path 'noisy'. This combination (BFGS and Armijo rule) is, however, widely used in efficient implementations of several nonlinear optimisation methods.

A variant of BFGS, called the *limited memory* BFGS (l-BFGS) utilises efficient implementations that

do not require storing the whole approximation for the Hessian, but only a few most recent p_k and q_k vectors.

2 Complexity, convergence and conditioning

Several aspects must be considered when analysing the performance of algorithms under a given setting and, in each, a multitude of theoretical results that can be used to understand, even if to some extent, the performance of a given optimisation method.

We focus on three key properties that one should be aware when employing the methods we have seen to solve optimisation problems. The first two, *complexity* and *convergence* refer to the algorithm itself, but often involve considerations related to the function being optimised. *Conditioning*, on the other hand, is a characteristic exclusively related to the problem at hand. Knowing how the "three C's" can influence the performance of an optimisation problem is central in making good choices in terms of which optimisation method to employ.

2.1 Complexity

Algorithm complexity analysis is a discipline from computer science that focus on deriving worst-case guarantees in terms of the number of computational steps required for an algorithm to converge, given an input of known size. For that, we use the following definition to identify efficient, generally referred to as *polynomial*, algorithms.

Definition 3 (Polynomial algorithms)**.** *Given a problem P, a problem instance $X \in P$ with length $L(X)$ in binary representation, and an algorithm A that solves X, let $f_A(X)$ be the number of* elementary calculations *required to run A on X. Then, the running time of A on X is proportional to*

$$f_A^*(n) = \sup_X \{f_A(X) : L(X) = n\}.$$

Algorithm A is polynomial *for a problem P if $f_A^*(n) = O(n^p)$ for some integer p.*

Notice that this sort of analysis only render bounds on the worst-case performance. Though it can be informative under a general setting, there are several well known examples in that experimental practice does not correlate with the complexity analysis. One famous example is the simplex method for linear optimisation problems, which despite not being a polynomial algorithm, presents widely-demonstrated reliable (polynomial-like) performance.

2.2 Convergence

In the context of optimisation, *local analysis* is typically more informative regarding to the behaviour of optimisation methods. This analysis tend to disregard initial steps further from the initial points and concentrate on the behaviour of the sequence $\{x_k\}$ to a unique point $\overline{x}$.

The convergence is analysed by means of *rates of convergence* associated with *error functions* $e : \mathbb{R}^n \to \mathbb{R}$ such that $e(x) \geq 0$. Typical choices for e include:

- $e(x) = \|x - \overline{x}\|$;
- $e(x) = |f(x) - f(\overline{x})|$.

The sequence $\{e(x_k)\}$ is then compared to the geometric progression β^k, with $k = 1, 2, \ldots,$ and $\beta \in (0, 1)$. We say that a method presents *linear convergence* if exists $q > 0$ and $\beta \in (0, 1)$ such that $e(x) \leq q\beta^k$ for all k. An alternative way of posing this result is stating that

$$\lim_{k\to\infty} \sup \frac{e(x_{k+1})}{e(x_k)} \leq \beta.$$

We say that an optimisation method converges superlinearly if the rate of convergence tends to zero. That is, if exists $\beta \in (0, 1)$, $q > 0$ and $p > 1$ such that $e(x_k) \leq q\beta^{p^k}$ for all k. For $k = 2$, we say that the method presents quadratic convergence. Any p-order convergence is obtained if

$$\lim_{k\to\infty} \sup \frac{e(x_{k+1})}{e(x_k)^p} < \infty, \text{ which is true if } \lim_{k\to\infty} \sup \frac{e(x_{k+1})}{e(x_k)} = 0.$$

Linear convergence is the most typical convergence rate for nonlinear optimisation methods, which is satisfactory if β is not too close to one. Certain methods are capable of achieving superlinear convergence for certain problems, being Newton's method an important example.

In light of what we discussed, let us analyse the convergence rate of some of the methods earlier discussed. We start by posing the convergence of gradient methods.

Theorem 4 (Convergence of the gradient method)**.** *Let $f(x) = \frac{1}{2}x^\top Hx$ where H is a positive definite symmetric matrix. Suppose $f(x)$ is minimised with the gradient method using an exact line search. Let $\underline{\lambda} = \min_{i=1,\ldots,n} \lambda_i$ and $\overline{\lambda} = \max_{i=1,\ldots,n} \lambda_i$, where λ_i are eigenvalues of H. Then, for all k,*

$$\frac{f(x_{k+1})}{f(x_k)} \leq \left(\frac{\overline{\lambda} - \underline{\lambda}}{\overline{\lambda} + \underline{\lambda}}\right)^2$$

Theorem 4 implies that, under certain assumptions, the gradient methods present *linear convergence*. Moreover, this result shows that the convergence rate is *dependent* on the scaling of the function, since it depends on the ratio of eigenvalues of H, which in turn can be modified by scaling f. This results exposes an important shortcoming that gradient methods present: the dependence on the

conditioning of the problem, which we will discuss shortly. Moreover, this result can be extended to incorporate functions other than quadratic and also inexact line searches.

The convergence of Newton's method is also of interest since, under specific circumstances, it presents a quadratic convergence rate. Theorem 5 summarises these conditions.

Theorem 5 (Convergence of Newton's method - general case)**.** *Let $g : \mathbb{R}^n \to \mathbb{R}^n$ be differentiable, $\overline{x}$ such that $g(\overline{x}) = 0$, and let $\{e(x_k)\} = \{\|x_k - \overline{x}\|\}$. Moreover, let $N_\delta(\overline{x}) = \{x : \|x - \overline{x}\| \leq \delta\}$ for some $\delta > 0$. Then*

1. *There exists $\delta > 0$ such that if $x_0 \in N_\delta(\overline{x})$, the sequence $\{x_k\}$ with $x_{k+1} = x_k - (\nabla g(x_k)^\top)^{-1} g(x_k)$ belongs to $N_\delta(\overline{x})$ and converges to $\overline{x}$, while $\{e(x_k)\}$ converges superlinearly.*
2. *If for some $L > 0$, $M > 0$, and for all $x, y \in N_\delta(\overline{x})$, $\lambda \in (0, \delta]$*

$$\|\nabla g(x) - \nabla g(y)\| \leq L\|x - y\| \quad \textit{and} \quad \|(\nabla g(x_k)^\top)^{-1}\| \leq M,$$

then, if $x_0 \in N_\delta(\overline{x})$, we have for $k = 0, 1, \ldots$

$$\|x_{k+1} - \overline{x}\| \leq \frac{LM}{2}\|x_k - \overline{x}\|^2.$$

If $\frac{LM\delta}{2} < 1$ and $x_0 \in N_\delta(\overline{x})$, $\{e(x_k)\}$ converges quadratically.

Notice that the convergence of the method is analysed in two distinct phases. In the first phase, referred to as 'damped' phase, superlinear convergence is observed within the neighbourhood $N_\delta(\overline{x})$ defined by δ. The second phase is where quadratic convergence is observed and it happens when $\delta < \frac{2}{LM}$, which in practice can only be interpreted as small enough, as the constants L (the Lipschitz constant) and M (a finite bound for the norm of the Hessian) cannot be easily estimated in practical applications.

However, it is interesting to notice that the convergence result for Newton's method do not depend on the scaling of the problem, like the gradient method. This property, called *affine invariance* is one of the greatest features that Newton's method possess.

Figure 5 compare the convergence of four methods presented considering $f(x) = e^{(-(x_1-3)/2)} + e^{((4x_2+x_1)/10)} + e^{((-4x_2+x_1)/10)}$, employing exact line search and using $e(x) = \|x_k - \overline{x}\|$. Notice how the quadratic convergence of Newton's method compare with the linear convergence of the gradients method. The other two, conjugate gradients and BFGS, present superlinear convergence.

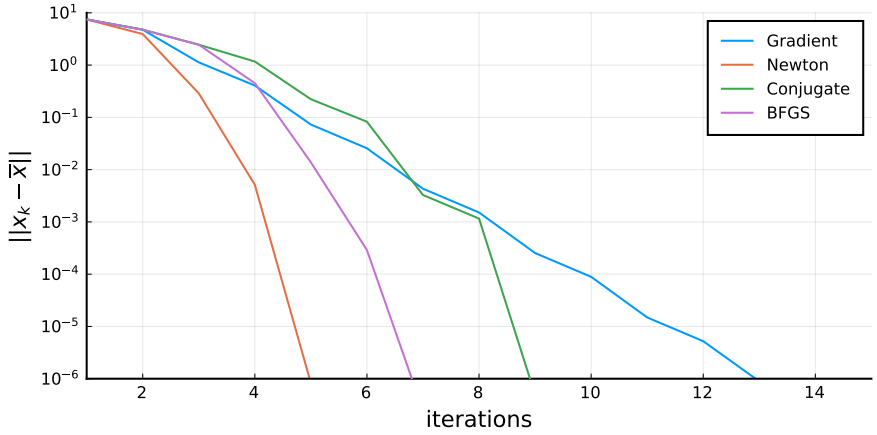


Figure 5: Convergence comparison for the four methods

2.3 Conditioning

The *condition number* of a symmetric matrix is given by

$$\kappa = \|A\|_2 \|A^{-1}\|_2 = \frac{\max_{i=1,\dots,n}\{\lambda_i\}}{\min_{i=1,\dots,n}\{\lambda_i\}} = \frac{\overline{\lambda}}{\underline{\lambda}}$$

The condition number κ is an important measure in optimisation, since it can be used to predict how badly scaled a problem might be. Large κ values mean that numerical errors will be amplified after repeated iterations, in particular matrix inversions.

Roughly speaking, having $\kappa \geq 10^k$ means that at each iteration, k digits of accuracy are lost. As general rule, one would prefer smaller κ numbers, but good values are entirely problem dependent.

One way of understanding the role that the conditioning number κ has is to think the role that the eigenvalues of the Hessian have in the shape of the level curves of quadratic approximations of a general function $f : \mathbb{R}^n \to \mathbb{R}$. First, let us consider the Hessian $H(x)$ at a given point $x \in \mathbb{R}^n$ is the identity matrix I, for which all eigenvalues are 1 and eigenvectors are e_i, $i = 1, \dots, n$, where e_i is the vector with component 1 in the position i and zero everywhere else. This means that in the direction of the n-eigenvectors, the ellipsoid formed by the level curves (specifically, the lower level sets) of f stretch by the same magnitude and, therefore, the level curves of the quadratic approximation are in fact a circle. Now, suppose that for one of the dimensions i of the matrix $H(x)$, we have one of the eigenvalues greater than 1. What we would see is that the level curves of the quadratic approximation will be more stretched in that dimension i than in the others. The reason for that is because the Hessian plays a role akin to that of a characteristic matrix in an ellipsoid (specifically due to the second order term $\frac{1}{2}(x - x_k)^\top H(x_k)(x - x_k)$ in the quadratic approximation).

Thus, larger κ will mean that the ratio between the eigenvalues is larger, which in turn implies that

there is eccentricity in the lower level sets (i.e., the lower level sets are far wider in one direction than in others), which ultimately implies that first-order methods struggle since often the gradients often point to directions that only show descent for small step sizes.

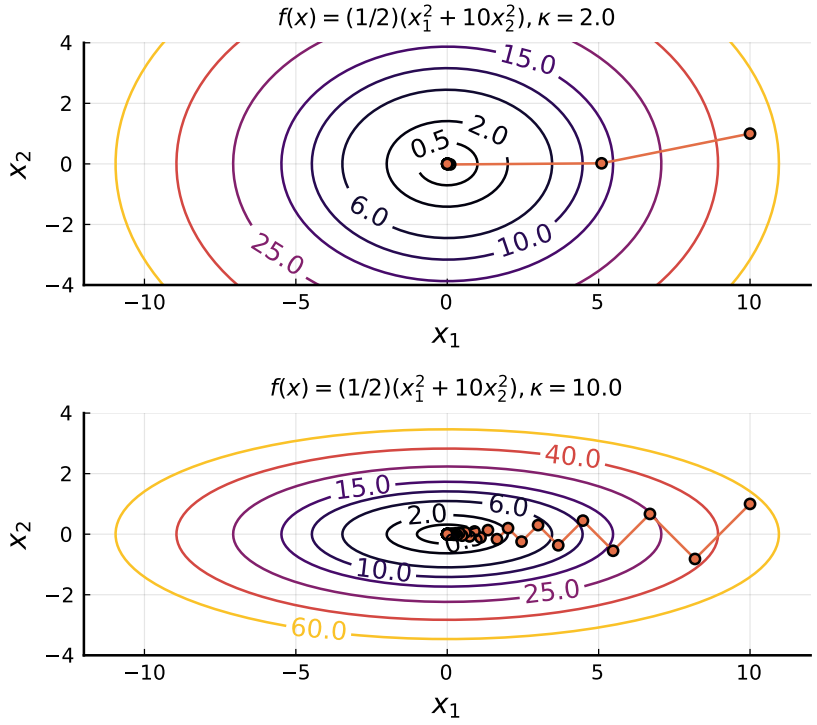


Figure 6: The gradient method with exact line search for different κ.

Figure 6 illustrates the effect of different condition numbers on the performance of the gradient method. As can be seen, the method require more iterations for higher conditioning numbers, in accordance to the convergence result presented in Theorem 4.

Lecture 7 - Optimality conditions for constrained problems

Fabricio Oliveira (with modifications by Nuutti Hyvönen)

Last update: October 19, 2022

Abstract. In this lecture, we discuss the optimality conditions for constrained optimisation problems. We show how geometrical optimality can be converted into an algebraic representation using the Fritz-John optimality conditions. Next, we discuss the Karush-Kuhn-Tucker optimality condition, which require further regularity conditions on the constraints to hold as necessary conditions for optimality. We show that these regularity conditions can be translated into constraint qualification conditions, and discuss the main constraint qualification conditions one could use in practice.

Outline of this lecture

1 Optimality for constrained problems

We now investigate how to derive optimality conditions for the problem

$$(P): \ \text{min.} \ \{f(x) : x \in S\}.$$

In particular, we are interested in understanding the role that the feasibility set S has on the optimality conditions of constrained optimisation problems in the form of P. Let us first define two geometric elements that we will use to derive the optimality conditions for P.

Definition 1 (cone of feasible directions)**.** *Let $S \subseteq \mathbb{R}^n$ be a nonempty set, and let $\overline{x} \in \mathbf{clo}(S)$. The* cone of feasible directions D *at $\overline{x} \in S$ is given by*

$$D = \{d \in \mathbb{R}^n : d \neq 0, \ \textit{and } \overline{x} + \lambda d \in S \ \textit{for all } \lambda \in (0, \delta) \ \textit{for some } \delta > 0\}.$$

Definition 2 (cone of descent directions)**.** *Let $S \subseteq \mathbb{R}^n$ be a nonempty set, $f : \mathbb{R}^n \to \mathbb{R}$, and $\overline{x} \in \mathbf{clo}(S)$. The* cone of improving (i.e., descent) directions F *at $\overline{x} \in S$ is*

$$F = \{d \in \mathbb{R}^n : f(\overline{x} + \lambda d) < f(\overline{x}) \ \textit{for all } \lambda \in (0, \delta) \ \textit{for some } \delta > 0\}.$$

These cones are geometrical descriptions of the regions that, from a given point $\overline{x}$, one can obtain feasible (D) and improving (F) solutions. This is useful in that it allows to express the optimality conditions for $\overline{x}$ as observing that $F \cap D = \emptyset$ holds. In other words, $\overline{x}$ is locally optimal if there exists no feasible direction that can provide improvement in the objective function value.

Although having a geometrical representation of such sets can be useful in solidifying the conditions for which a feasible solution is also optimal, we need to derive an *algebraic* representation of such sets that can be used in computations. To reach that objective, let us start by defining an algebraic representation for F. For that, let us assume that $f : S \subset \mathbb{R}^n \to \mathbb{R}$ is differentiable. Recall that d is a descent direction at $\overline{x}$ if $\nabla f(\overline{x})^\top d < 0$. Thus, we can define the set

$$F_0 = \left\{d \in \mathbb{R}^n : \nabla f(\overline{x})^\top d < 0\right\}$$

as an algebraic representation for F. Notice that F_0 is an open half-space formed by the hyperplane with normal $\nabla f(\overline{x})$. Figure 1 illustrates the condition $F_0 \cap D = \emptyset$. Theorem 3 establishes that the condition $F_0 \cap D = \emptyset$ is necessary for local optimality in constrained optimisation problems.

Theorem 3 (geometric necessary condition)**.** *Let $S \subseteq \mathbb{R}^n$ be a nonempty set, and let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable at $\overline{x} \in S$. If $\overline{x}$ is a local optimal solution to*

$$(P): \ \textit{min.} \ \{f(x) : x \in S\},$$

then $F_0 \cap D = \emptyset$, with $F_0 = \left\{d \in \mathbb{R}^n : \nabla f(\overline{x})^\top d < 0\right\}$ and D the cone of feasible directions at $\overline{x}$.

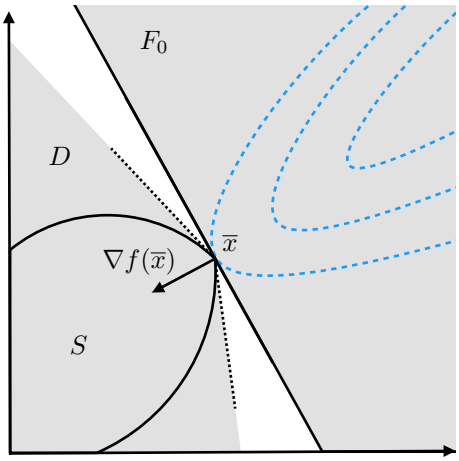


Figure 1: Illustration of the cones F_0 and D for the optimal point $\overline{x}$. Notice that D is an open set.

Proof. Assume that $\overline{x} \in S$ is a local optimal solution for P and let us prove the claim by contradiction. Suppose that $0 \neq d \in F_0 \cap D$. Hence, there exist $\delta_1 > 0$ such that $\overline{x} + \lambda d \in S$ for all $\lambda \in (0, \delta_1)$ and $\delta_2 > 0$ such that $f(\overline{x} + \lambda d) < f(\overline{x})$ for all $\lambda \in (0, \delta_2)$. By choosing $\delta = \min\{\delta_1, \delta_2\}$, it follows that for any $0 < \epsilon < \|d\|\delta$, there is a point $\hat{x} \in N_\epsilon(\overline{x}) \cap S$ such that $f(\hat{x}) < f(\overline{x})$. This contradicts the local optimality of $\overline{x}$ and completes the proof. □

As discussed earlier (in Lecture 4), if f and S are convex, this condition becomes sufficient for optimality. Moreover, if f is strictly convex, then $F = F_0$; otherwise, it may happen that F_0 is a proper subset of F (consider, e.g., $f(x) = x^3$ at the origin). If f is linear, it might be worth considering $F_0' = \left\{0 \neq d \in \mathbb{R}^n : \nabla f(\overline{x})^\top d \leq 0\right\}$ to allow for considering orthogonal directions.

1.1 Inequality constrained problems

In mathematical programming applications, the feasibility set S is typically expressed by a set of inequalities. Let us thus redefine P as

$$\begin{aligned} (P): \quad \text{min.} \quad & f(x) \\ \text{subject to:} \quad & g_i(x) \leq 0, \ i = 1, \ldots, m, \\ & x \in X, \end{aligned}$$

where $g_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$, are differentiable functions and $X \subset \mathbb{R}$ is an nonempty open set. The differentiability of g_i, $i = 1, \ldots, m$, allows for the definition of a proxy for D using the gradients

of the binding constraints $i \in I = \{i : g_i(\overline{x}) = 0\}$ at $\overline{x}$. This set, denoted by G_0, is defined as

$$G_0 = \left\{ d \in \mathbb{R}^n : \nabla g_i(\overline{x})^\top d < 0,\ i \in I \right\}. \tag{1}$$

The use of G_0 is a convenient algebraic representation, since it can be shown that $G_0 \subseteq D$, which is stated in Lemma 4. As $F_0 \cap D = \emptyset$ must hold for a local optimal solution $\overline{x} \in S$, it follows that $F_0 \cap G_0 = \emptyset$ must also hold.

Lemma 4. *Let $S = \{x \in X : g_i(x) \leq 0$ for all $i = 1, \ldots, m\}$, where $X \subset \mathbb{R}^n$ is a nonempty open set and $g_i : \mathbb{R}^n \to \mathbb{R}$ a differentiable function for all $i = 1, \ldots, m$. For a feasible point $\overline{x} \in S$, let $I = \{i : g_i(\overline{x}) = 0\}$ be the index set of the binding (or active) constraints and let G_0 be defined by* (1)*. Then $G_0 \subseteq D$, where D is the cone of feasible directions.*

The proof of Lemma 4 is based on the observation that the values of all g_i with $i \in I$ start to decrease when moving from $\overline{x} \in S$ to a direction $d \in G_0$. On the other hand, the inactive constraints, i.e. i such that $g_i(\overline{x}) < 0$, remain inactive in some nonempty open neighborhood of $\overline{x}$ due to the assumed continuity of g_i, $i = 1, \ldots, m$. Moreover, as X is open, one can move a small distance from $\overline{x}$ to any directions without leaving X.

In settings where g_i is affine for some $i \in I$, it may be worth considering $G_0' = \left\{ 0 \neq d \in \mathbb{R}^n : \nabla g_i(\overline{x})^\top d \leq 0,\ i \in I \right\}$ so that orthogonal feasible directions can also be represented. Notice that in this case $D \subseteq G_0'$.

2 Fritz–John conditions

The Fritz–John conditions are the algebraic conditions that must be met for $F_0 \cap G_0 = \emptyset$ to hold. These conditions are convenient as they only involve the gradients of the binding constraints, and they can thus be verified computationally.

Theorem 5 (Fritz–John necessary conditions)**.** *Let $X \subseteq \mathbb{R}^n$ be a nonempty open set, and let $f : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}$ be differentiable for all $i = 1, \ldots, m$. If a feasible $\overline{x}$ solves P locally, there exist scalars $u_0, \ldots, u_m$ such that*

$$\begin{aligned}
&u_0 \nabla f(\overline{x}) + \sum_{i=1}^{m} u_i \nabla g_i(\overline{x}) = 0, \\
&u_i g_i(\overline{x}) = 0,\ i = 1, \ldots, m, \\
&u_i \geq 0,\ i = 0, \ldots, m, \\
&u = (u_0, \ldots, u_m) \neq 0.
\end{aligned}$$

Proof. Let $I = \{i : g_i(\overline{x}) = 0\}$. Since $\overline{x}$ solves P locally, Theorem 3 guarantees that there is no

$d \in \mathbb{R}^n$ such that $\nabla f(\overline{x})^\top d < 0$ and $\nabla g_i(x)^\top d < 0$ for all $i \in I$.

Let A be the matrix whose rows are $\nabla f(\overline{x})^\top$ and $\nabla g_i(\overline{x})^\top$ for $i \in I$. Using a suitable version of the Farkas' theorem, one can show that as there is no $d \in \mathbb{R}^n$ such that $Ad < 0$, then there must exist a nonzero $p \geq 0$ such that $A^\top p = 0$. Seting $[u_0, u_{i_1}, \ldots, u_{i_{|I|}}]^\top = p$ for $I = \{i_1, \ldots, i_{|I|}\}$ and $u_i = 0$ for $i \notin I$, the result follows. □

The proof begins by observing that, if $\overline{x}$ is optimal, then the matrix

$$A = \begin{bmatrix} \nabla f(\overline{x})^\top \\ \nabla g_{i_1}(\overline{x})^\top \\ \vdots \\ \nabla g_{i_{|I|}}(\overline{x})^\top \end{bmatrix},$$

with $I = \{i_1, \ldots, i_{|I|}\}$, satisfies $Ad \not< 0$ for all $d \in \mathbb{R}^n$, because otherwise $F_0 \cap G_0 \neq \emptyset$ at $\overline{x}$. This is used with a variant of Farkas' theorem (known as Gordan's theorem) to show that the 'dual system' $A^\top p = 0$ has a nonzero solution such that $p \geq 0$ holds. By defining $[u_0, u_{i_1}, \ldots, u_{i_{|I|}}]^\top = p$ and enforcing that the other gradients $\nabla g_i(\overline{x})$, for $i \notin I$, do not actively contribute to the first Fritz–John condition by setting $u_i = 0$, this leads precisely to the whole set of Fritz–John conditions.

The scalars u_i, for $i = 0, \ldots, m$, are called Lagrange multipliers due to their connection to Lagrangian duality, as we will see later. Also, notice that for nonbinding constraints, i.e. $g_i(\overline{x}) < 0$ for $i \notin I$, the corresponding u_i must be zero in the Fritz–John conditions. This is called "complementary slackness".

The Fritz–John conditions are unfortunately too weak, which is a problematic issue in some rather common settings. In particular, observe that a point $\overline{x}$ satisfies the Fritz–John conditions if and only if $F_0 \cap G_0 = \emptyset$, which is trivially satisfied when $G_0 = \emptyset$. The Fritz–John conditions are thus trivially satisfied at points where some of the gradients vanish, that is, $\nabla f(\overline{x}) = 0$ or $\nabla g_i(\overline{x}) = 0$ for some $i = 1, \ldots, m$. Sets with no relative interior in the immediate vicinity of $\overline{x}$ also satisfy the Fritz–John conditions. Equality constraints can be particularly difficult from the standpoint of the Fritz–John conditions, as illustrated in Figure 2.

3 Karush–Kuhn–Tucker conditions

The Karush–Kuhn–Tucker (KKT) conditions can be understood as the Fritz–John conditions with an extra requirement of 'regularity' for $\overline{x} \in S$. Such a regularity requirement is called *constraint qualification* and, in a general sense, it is meant to prevent the trivial case $G_0 = \emptyset$, thus making the optimality conditions stronger (i.e., more stringent). This is achieved by forcing $u_0 = 1$ in Theorem 5 via assuming that the gradients $\nabla g_i(\overline{x})$ for $i \in I$ are linearly independent. This assumption of linear

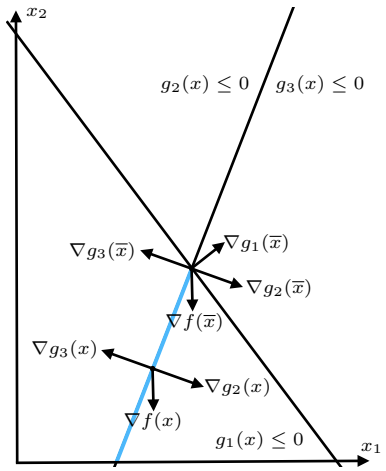


Figure 2: All points in the blue segment satisfy the Fritz-John conditions, including the minimiser $\overline{x}$. Note that a single equality constrained is expressed as two inequality constraints $g_2(x) \leq 0$ and $g_3(x) \leq 0$, with $g_2 = -g_3$.

independence is called *linearly independent constraint qualification* (LICQ), and it is one of several known constraint qualifications that can be used to guarantee the regularity of $\overline{x} \in S$.

Theorem 6 establishes the KKT conditions as necessary for the local optimality of $\overline{x}$ assuming that LICQ holds. For notational simplicity, we continue to assume that

$$(P): \quad \text{min. } \{f(x) : g_i(x) \leq 0,\ i = 1, \ldots, m,\ x \in X\}$$

and consider equality constraints separately below.

Theorem 6 (Karush–Kuhn–Tucker necessary conditions)**.** *Let $X \subseteq \mathbb{R}^n$ be a nonempty open set, and let $f : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}$ be differentiable for all $i = 1, \ldots, m$. In addition, for a feasible $\overline{x}$, let $I = \{i : g_i(\overline{x}) = 0\}$ and suppose that the gradients $\nabla g_i(\overline{x})$, $i \in I$, are linearly independent. If $\overline{x}$ solves P locally, there exist scalars $u_1, \ldots, u_m$ such that*

$$\begin{aligned} &\nabla f(\overline{x}) + \sum_{i=1}^{m} u_i \nabla g_i(\overline{x}) = 0, \\ &u_i g_i(\overline{x}) = 0,\ i = 1, \ldots, m, \\ &u_i \geq 0,\ i = 1, \ldots, m. \end{aligned}$$

Proof. By Theorem 5 and the complementary slackness property, there exist $\hat{u}_i$, $i \in \{0\} \cup I$, that

are not all zeros and satisfy

$$\hat{u}_0 \nabla f(\overline{x}) + \sum_{i \in I} \hat{u}_i \nabla g_i(\overline{x}) = 0,$$
$$\hat{u}_i \geq 0, \ i \in I.$$

Due to the assumed linear independence of $\nabla g_i(\overline{x})$, $i \in I$, either $\sum_{i \in I} \hat{u}_i \nabla g_i(\overline{x}) \neq 0$ or $u_i = 0$ for all $i \in I$ must hold, and thus $\hat{u}_0 > 0$. Letting $u_i = \hat{u}_i / u_0$ for $i \in I$ and $u_i = 0$ for $i \notin I$, the claim follows.

□

The proof builds upon the Fritz–John conditions: under the assumption that the gradients of the active constraints $\nabla g_i(\overline{x})$, $i \in I$, are linearly independent, all Lagrange multipliers can be rescaled so that $u_0 = 1$.

The general conditions including both inequality and equality constraints are posed as follows. Notice that the Lagrange multipliers v_i associated with the equality constraints $h_i(\overline{x}) = 0$, $i = 1, \ldots, l$, are irrestricted in sign, and the corresponding complementary slackness conditions are not explicitly stated since their hold redundantly. These conditions can be deduced by replacing an equality constraint $h(x) = 0$ with two equivalent inequalities $h(x) \leq 0$ and $-h(x) \leq 0$, writing down the conditions in Theorem 6, and combining the summands involving $\nabla h(\overline{x})$ and $-\nabla h(\overline{x})$. Also, notice that, in the absence of constraints, the KKT conditions reduce to the unconstrained first-order condition $\nabla f(\overline{x}) = 0$.

$$\nabla f(\overline{x}) + \sum_{i=1}^{m} u_i \nabla g_i(\overline{x}) + \sum_{i=1}^{l} v_i \nabla h_i(\overline{x}) = 0, \qquad \text{(dual feasibility 1)}$$
$$u_i g_i(\overline{x}) = 0, \qquad i = 1, \ldots, m, \qquad \text{(complementary slackness)}$$
$$g_i(\overline{x}) \leq 0, \qquad i = 1, \ldots, m, \qquad \text{(primal feasibility)}$$
$$h_i(\overline{x}) = 0, \qquad i = 1, \ldots, l, \qquad \text{(primal feasibility)}$$
$$u_i \geq 0, \qquad i = 1, \ldots, m, \qquad \text{(dual feasibility 2)}$$
$$\overline{x} \in X. \qquad \text{(primal feasibility)}$$

In case of mere inequality constraints, the KKT conditions can be interpreted geometrically as follows. Consider the cone spanned by the gradients of the active constraints at $\overline{x}$, i.e., $C(\overline{x}) = \left\{ \sum_{i \in I} u_i \nabla g_i(\overline{x}) : u_i \geq 0, \ i \in I \right\}$. A solution $\overline{x}$ then satisfies the KKT conditions if $-\nabla f(\overline{x}) \in C(\overline{x})$, which is equivalent to $-\nabla f(\overline{x}) = \sum_{i \in I} u_i \nabla g_i(\overline{x})$ for some $u_i \geq 0$, $i \in I$. Figure 3 illustrates this condition.

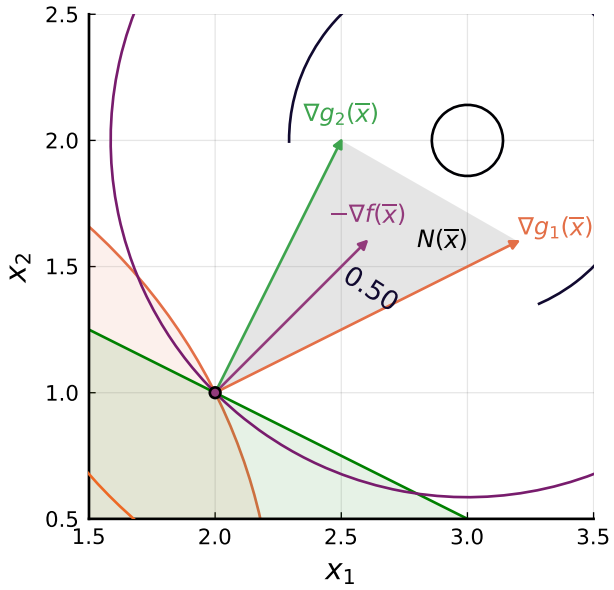


Figure 3: Graphical illustration of the KKT conditions at the optimal point $\overline{x}$.

4 Constraint qualification

Constraint qualification is a technical condition that needs to be assessed in the context of nonlinear optimisation problems. As we rely on an algebraic description of the set of directions G_0 that serves as proxy for D, it is important to be sure that the former is indeed a reliable description of the latter.

In specific, constraint qualification can be seen as a certification that the geometry of the feasible region and gradient information obtained from the constraints that form it are related at an optimal solution. Recall that gradients can only provide a *first-order* approximation of the feasible region, which might lead to mismatches. This is typically the case when the feasible region has cusps or isolated feasible points.

Constraint qualification can be seen as certificates for a proper relationship between the (closed) set of feasible directions

$$G_0' = \left\{0 \neq d \in \mathbb{R}^n : \nabla g_i(\overline{x})^\top d \leq 0,\ i \in I\right\}$$

and the cone of tangents (or the tangent cone)

$$T = \left\{d :\ d = \lim_{k\to\infty} \lambda_k(x_k - \overline{x}),\ \lim_{k\to\infty} x_k = \overline{x},\ x_k \in S,\ \lambda_k > 0\right\} \tag{2}$$

with $S = \{x : g_i(x) \leq 0,\ i = 1,\ldots,m;\ h_j(x) = 0,\ j = 1,\ldots,l;\ x \in X\}$. The cone of tangents is the cone representing all directions in which the feasible region allows for an arbitrarily small

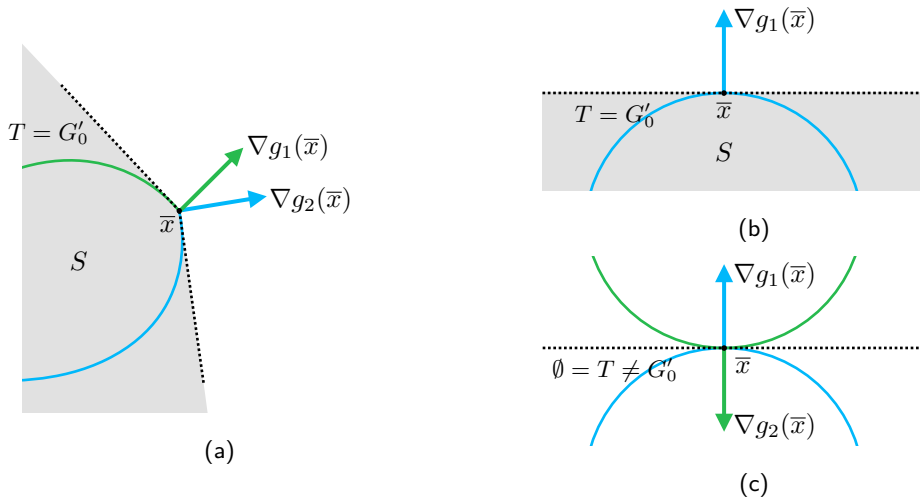


Figure 4: CQ holds for 4a and 4b, since the tangent cone T and the cone of feasible directions G_0' (denoted by the dashed black lines and grey area) match; for 4c, they do not match, as $T = \emptyset$

movement from the point $\overline{x}$ so that feasibility is retained. As the name suggests, at a boundary point $\overline{x}$ it is typically defined by the lines that tangent S at $\overline{x}$. However, if $\overline{x}$ is in the interior of $S \subseteq \mathbb{R}^n$, then $T = \mathbb{R}^n$.

One way of interpreting the cone of tangents as defined in (2) is the following: Consider a sequence of feasible points $\{x_k\}_{k=1}^{\infty} \subset S$ in any trajectory you like under the condition that the sequence converges to $\overline{x}$. Then, take the vector $x_k - \overline{x}$ and consider the limit of its *direction* as $k \to \infty$. The collection of all scalar multiples of such limit directions corresponding to all possible trajectories is what forms the cone of tangents.

Abadie's constraint qualification holds if $T = G_0'$ at $\overline{x}$. In the presence of equality constraints, the condition becomes $T = G_0' \cap H_0$, with

$$H_0 = \left\{ d : \nabla h_i(\overline{x})^\top d = 0,\ i = 1, \ldots, l \right\}.$$

Figure 4 illustrates the tangent cone T and the cone of feasible directions G_0' for cases when the constraint qualification holds (Figures 4a and 4b), i.e., $T = G_0'$, and a case when it does not hold (Figure 4c) as $T = \emptyset$ and G_0' is given by the dashed black line.

The importance of Abadie's constraint qualification is that it allows for generalising the KKT conditions by replacing the condition on the linear independence of the gradients $\nabla g_i(\overline{x})$ for $i \in I$. This allows us to state the KKT conditions as presented in Theorem 7.

Theorem 7 (Karush–Kuhn–Tucker necessary conditions II)**.** *Consider the problem*

$$(P): \quad \textit{min.} \ \{ f(x) : g_i(x) \leq 0,\ i = 1, \ldots, m,\ x \in X \}.$$

Let $X \subseteq \mathbb{R}^n$ be a nonempty open set, and let $f : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}$ be differentiable for all $i = 1, \ldots, m$. Suppose that Abadie's CQ holds at a feasible $\overline{x}$. If $\overline{x}$ solves P locally, there exist scalars $u_1, \ldots, u_m$, such that

$$\begin{aligned} &\nabla f(\overline{x}) + \sum_{i=1}^{m} u_i \nabla g_i(\overline{x}) = 0, \\ &u_i g_i(\overline{x}) = 0, \ i = 1, \ldots, m, \\ &u_i \geq 0, \ i = 1, \ldots, m. \end{aligned}$$

Despite being a more general result, Theorem 7 is of little use since Abadie's constraint qualification cannot be straightforwardly verified in practice. Alternatively, we can rely on verifiable constraint qualification conditions that imply Abadie's constraint qualification. Examples of such include

1. **Linear independence (LI)CQ:** holds at $\overline{x}$ if $\nabla g_i(\overline{x})$, for $i \in I$, as well as $\nabla h_i(\overline{x})$, $i = 1, \ldots, l$, are *linearly independent* (this was already considered in Theorem 6)
2. **Affine CQ:** holds for all $x \in S$ if g_i, $i = 1, \ldots, m$, and h_i, $i = 1, \ldots, l$, are *affine*.
3. **Slater's CQ:** holds for all $x \in S$ if g_i is *convex* for all $i = 1, \ldots, m$, h_i is *affine* for all $i = 1, \ldots, l$, and there exists $x \in S$ such that $g_i(x) < 0$ for all $i = 1, \ldots, m$.

Slater's constraint qualification is the most frequently used, in particular in the context of convex optimisation problems. One important point to notice is the requirement of not having an empty relative interior, which can be a source of error.

Consider, for example: $P = \left\{\text{min. } x_1 : x_1^2 + x_2 \leq 0, \ x_2 \geq 0\right\}$. Notice that P is convex and the KKT system for P at the only feasible solution $\overline{x} = (0, 0)$ is

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = 0; \quad u_1, u_2 \geq 0,$$

which has no solution. Thus, the KKT conditions are not necessary for the global optimality of $(0, 0)$. This is due to the lack of CQ, since the feasible region is the single point $(0, 0)$, and thus, e.g., Slater's CQ does not hold.

Corollary 8 summarises the setting in which one should expect the KKT conditions to be necessary an sufficient conditions for global optimality, i.e., convex optimisation.

Corollary 8 (Necessary and sufficient KKT conditions)**.** *Suppose that Slater's CQ holds. Then, if f is convex, the conditions of Theorem 7 are* necessary and sufficient *for $\overline{x}$ to be a global optimal solution.*

Lecture 8 - Lagrangian duality

Fabricio Oliveira (with modifications by Nuutti Hyvönen)

Last update: November 3, 2022

Abstract. In this lecture, we look into the notion of Lagrangian duality, which consists of a theoretical framework that allows deriving not only optimality conditions for constrained optimisation problems, but also solution methods build upon results from duality theory. First, we consider Lagrangian duality under the notion of relaxation, and show how primal and dual solutions are related. Then we describe the notion of strong duality, a key property that connects Lagrangian duality and optimality conditions for constrained problems. Lastly, we discuss the subgradient method, one of the most widespread methods for solving constrained optimisation problems using Lagrangian duals.

Outline of this lecture

1 The concept of relaxation

The idea of using relaxations is central in several constrained optimisation methods. In a general sense, it consists of techniques that remove constraints from the problem to allow for a version, i.e., a *relaxation*, that is simpler to solve and/or can provide information to be used for solving the original problem.

A classical example of the employment of relaxations for solving constrained problems is the branch-and-bound method that uses linear (continuous) relaxations of integer problems to guide the search for optimal solutions that are at the end required to be integers. However, there are several other examples of settings in which relaxations are purposely derived to lead to problems with a convenient structure that can be exploited.

Let $f : \mathbb{R}^n \to \mathbb{R}$ and $S \subseteq \mathbb{R}^n$. Consider the following problem:

$$(P): \quad \text{min. } \{f(x) : x \in S\}.$$

Definition 1 provides the conditions for P_R to be a *relaxation* of P, where

$$(P_R): \quad \text{min. } \{f_R(x) : x \in S_R\}$$

with $f_R : \mathbb{R}^n \to \mathbb{R}$, $S_R \subseteq \mathbb{R}^n$.

Definition 1 (Relaxation)**.** *P_R is a relaxation of P if and only if:*

1. *$f_R(x) \leq f(x)$ for all $x \in S$;*
2. *$S \subseteq S_R$.*

In specific, P_R is said to be a relaxation for P if $f_R(x)$ bounds $f(x)$ from below (in a minimisation setting) for all $x \in S$ and the enlarged feasible region S_R contains S.

The motivation for using relaxations arises from the possibility of finding a solution to the original problem P by solving P_R. Clearly, such a strategy would only make sense if P_R possess some attractive property or feature that we can use in our favour to, e.g., improve solution times or create separability that can be further exploited using parallelised computation (which we will discuss in more detail in the upcoming lectures). Theorem 2 presents the technical result that allows for using relaxations for solving P.

Theorem 2 (Relaxation theorem)**.** *Let us define*

$$(P): \quad \textit{min. } \{f(x) : x \in S\} \quad \textit{and} \quad (P_R): \quad \textit{min. } \{f_R(x) : x \in S_R\}.$$

If P_R is a relaxation of P, then the following hold:

1. *if P_R is infeasible (i.e., $S_R = \emptyset$), so is P (i.e., $S = \emptyset$);*
2. *if $\overline{x}_R$ is an optimal solution to P_R such that $\overline{x}_R \in S$ and $f_R(\overline{x}_R) = f(\overline{x}_R)$, then $\overline{x}_R$ is optimal to P as well.*

Proof. Result (1) follows since $S \subseteq S_R$. To show (2), notice that $f(\overline{x}_R) = f_R(\overline{x}_R) \leq f_R(x) \leq f(x)$ for all $x \in S$. □

The first part of the proof is a consequence of $S \subset S_R$, meaning that if $x \notin S$, then $x \notin S_R$. The second part combines the optimality of $\overline{x}_R$ (first inequality) and the definition of a relaxation (second inequality) to derive the optimality condition of $\overline{x}_R$ for P, which is $f(\overline{x}_R) \leq f(\overline{x})$ for all $x \in S$.

2 Lagrangian dual problems

Lagrangian duality is the body of theory supporting the use of *Lagrangian relaxations* to solve constrained optimisation problems. In what follows, we refer to the relaxation obtained using Lagrangian duality as the *(Lagrangian) dual* problem. Consequently, we refer to the original problem as the *primal* problem.

Let $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, $h : \mathbb{R}^n \to \mathbb{R}^l$, and assume that $X \subseteq \mathbb{R}^n$ is an open set. Then, consider P defined as

$$\begin{aligned} (P): \quad \text{min.} \quad & f(x) \\ \text{subject to:} \quad & g(x) \leq 0, \\ & h(x) = 0, \\ & x \in X. \end{aligned}$$

For a given set of *dual variables* $(u, v) \in \mathbb{R}^{m+l}$ with $u \geq 0$, the *Lagrangian relaxation* or *Lagrangian dual function* for P is

$$\theta(u, v) = \inf_{x \in X} \phi(x, u, v),$$

where

$$\phi(x, u, v) := f(x) + u^\top g(x) + v^\top h(x)$$

is the *Lagrangian function*.

Notice that the Lagrangian dual function $\theta(u, v)$ has a built-in optimisation problem in x, meaning that evaluating $\theta(u, v)$ still requires solving an optimisation problem, which amounts to finding the minimiser $\overline{x}$ for $\phi(x, u, v)$, given (u, v).

2.1 Weak and strong duality

Weak and strong duality are, to some extent, consequences of Theorem 2 and the fact that the Lagrangian relaxation is indeed a relaxation of P. We start with the equivalent to Definition 1, which is referred to as *weak duality*.

Theorem 3 (Weak Lagrangian duality)**.** *Let x be a feasible solution to P and let $(u, v) \in \mathbb{R}^{m+l}$ be such that $u \geq 0$. Then $\theta(u, v) \leq f(x)$.*

Proof. From feasibility, $u \geq 0$, $g(x) \leq 0$ and $h(x) = 0$. Thus, we have that

$$\theta(u, v) = \inf_{y \in X} \left\{ f(y) + u^\top g(y) + v^\top h(y) \right\} \leq f(x) + u^\top g(x) + v^\top h(x) \leq f(x).$$

which completes the proof. □

The proof uses the fact that the infimal of the Lagrangian function $\phi(\,\cdot\,, u, v)$, or in fact $\phi(x, u, v)$, is a lower bound for $f(x)$ for any primal feasible x and dual feasible $u \geq 0$ (a condition for the Lagrangian relaxation to be indeed a relaxation). This arises from observing that $g(x) \leq 0$ and $h(x) = 0$ for a feasible x. The *Lagrangian dual problem* is the problem used to obtain the best possible relaxation bound $\theta(u, v)$ for $f(x)$ in light of Theorem 3, that is, maximising $\theta(u, v)$ over feasible (u, v). This can be achieved by optimising $\theta(u, v)$ in the space of the dual variables (u, v), that is, aiming to solve

$$\begin{aligned} (D): \quad & \max_{u,v} \ \theta(u, v) = \max_{u,v} \ \left(\inf_{x \in X} \ \phi(x, u, v) \right) \\ & \text{subject to: } \ u \geq 0. \end{aligned}$$

The use of Lagrangian dual problems is an alternative for dealing with constrained optimisation problems, as they allow to convert the constrained primal into a (typically) unconstrained dual that is potentially easier to handle or present exploitable properties that can benefit specialised algorithms, such as separability.

Employing Lagrangian relaxations to solve optimisation problems is possible due to the following important results, which are posed as corollaries of Theorem 3.

Corollary 4 (Weak Lagrangian duality II)**.**

$$\sup_{u,v} \left\{ \theta(u, v) : u \geq 0 \right\} \leq \inf_{x} \left\{ f(x) : g(x) \leq 0, \ h(x) = 0, \ x \in X \right\}.$$

Proof. We have $\theta(u, v) \leq f(x)$ for any feasible x and (u, v), thus implying $\sup_{u,v} \{\theta(u, v) : u \geq 0\} \leq \inf_x \{f(x) : g(x) \leq 0, \ h(x) = 0, \ x \in X\}$. □

Corollary 5 (One-way strong Lagrangian duality)**.** *If $f(\overline{x}) = \theta(\overline{u}, \overline{v})$ for some $\overline{u} \geq 0$ and*

$$\overline{x} \in \{x \in X : g(x) \leq 0,\ h(x) = 0\},$$

then $\overline{x}$ and $(\overline{u}, \overline{v})$ are optimal solutions to P and D, respectively.

Proof. By Corollary 4, $\theta(u, v) \leq f(x)$ for any feasible (u, v) and x. Hence, for the equality $\theta(\overline{u}, \overline{v}) = f(\overline{x})$ to hold, $\theta(\overline{u}, \overline{v})$ must present the largest value for θ and $f(\overline{x})$ the smallest value for f over the respective sets of feasible variables. (If, e.g., there existed a feasible $\hat{x}$ such that $f(\hat{x}) < f(\overline{x})$, then $\theta(\overline{u}, \overline{v}) = f(\overline{x}) > f(\hat{x})$, which contradicts Corollary 4.) □

Notice that Corollary 5 implies that if the target function values for the primal and dual problems match, then the variables producing these values are solutions to the primal and dual problems, respectively. However, to use Lagrangian relaxations to solve constrained optimisation problems, we need the opposite clause to also hold, that is, we need the the primal and dual optimal solutions to produce matching values for the respective target functions. This is called *strong duality* and, unfortunately, it does not always hold.

2.1.1 Geometric interpretation of Lagrangian duality

To investigate the cases in which strong duality can hold, let us focus on a graphical interpretation of Lagrangian dual problems. For that, let us first define some auxiliary elements. For the sake of simplicity, consider (P) : min. $\{f(x) : g(x) \leq 0,\ x \in X\}$ with $f : \mathbb{R}^n \to \mathbb{R}$, a single constraint $g : \mathbb{R}^n \to \mathbb{R}$ and $X \subseteq \mathbb{R}^n$ an open set.

Let us define a set $G = \{(y, z) : y = g(x),\ z = f(x),\ x \in X\} \subset \mathbb{R}^2$ that consists of the images of all points $x \in X$ under $(f, g) : \mathbb{R}^n \to \mathbb{R}^2$. In this setting, solving P means finding $(y, z) \in G$ that has the smallest second component among those points that have a nonpositive first component. Figure 1 illustrate this setting. In the depicted 'convex' case, the minimum value for the target function, $f(\overline{x})$, corresponds to the z-coordinate of the lowest point $(\overline{y}, \overline{z}) = (0, \overline{z})$ in G on the z-axis. This means that the constraint $g(x) \leq 0$ is active at the optimum; if $g(\overline{x}) < 0$ at the optimum $\overline{x}$, the lowest point in the intersection of G and the closed left half-plane would have a negative y-component.

The value of the Lagrangian function at a fixed $u \geq 0$ is given by

$$\theta(u) = \inf_x \{f(x) + ug(x) : x \in X\}.$$

In other words, one seeks the smallest value for $f(x) + ug(x) = z + uy =: \zeta(y, z)$ over $(y, z) \in G$. The bilinear function $\zeta(y, z)$ takes a constant value in the (y, z)-plane on any line with the slope $-u$, that is, if $z = \alpha - uy$ for some $\alpha \in \mathbb{R}$, which can be straightforwardly verified: $\zeta(y, \alpha - uy) = \alpha - uy + uy = \alpha$. Take note that α is the z-coordinate of the point where the line $z = \alpha - uy$

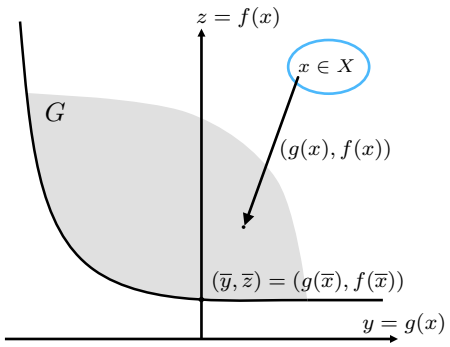


Figure 1: Illustration of the set G. Solving P amounts to finding the lowest point in the intersection of G and the closed left half-plane. In the illustrated case, this point is the lowest point on the z-axis still contained within G.

intersects the z-axis. In consequence,

$$\theta(u) = \inf_{\alpha \in \mathbb{R}} \{\alpha \in \mathbb{R} : \{(y, z) : z = \alpha - uy,\ y \in \mathbb{R}\} \cap G \neq \emptyset\}, \tag{1}$$

or in words, $\theta(u)$ is the lowest intersection point with the z-axis for those lines that have the slope $-u$ and pass through G. Figure 2 illustrates this construction.

In our simple setting, solving the dual problem

$$\begin{aligned} (D): \quad & \max_{u} \ \theta(u) \\ \text{subject to:} \quad & u \geq 0 \end{aligned}$$

thus corresponds to finding the nonpositive slope $-u$ for which the lowest intersection point with the z-axis for lines passing through G is as high as possible. In the setting of Figure 2, this corresponds to choosing the slope $-\overline{u}$ to be that of the tangent line to G at $(\overline{y}, \overline{z}) = (0, \overline{z})$, and this tangent line is exactly the one that realises the infimum in (1). This means that $\theta(\overline{u}) = \overline{z} = f(\overline{x})$, and the strong duality holds.

Assume next that G is still convex, but the lowest point, i.e., the point with the smallest z-component, in its intersection with the closed left half-plane is attained strictly left from the z-axis, say, at $(\overline{y}, \overline{z}) = (g(\overline{x}), f(\overline{x}))$, with $g(\overline{x}) < 0$. In this case, the line realising the maximum for θ is the horizontal line $z = f(\overline{x}) = \alpha$, which corresponds to the slope $-\overline{u} = 0$ (recall that u is constrained to be nonnegative, i.e., $-u$ must be nonpositive). In consequence, the strong duality still holds: $\theta(\overline{u}) = \theta(0) = f(\overline{x})$.

Note that in both of the cases considered above, the maximal value of θ actually corresponds to the lowest intersection point of a hyperplane supporting G with the z-axis (in the illustrated two-dimensional case, hyperplanes are lines). This observation can also be suitably generalised to

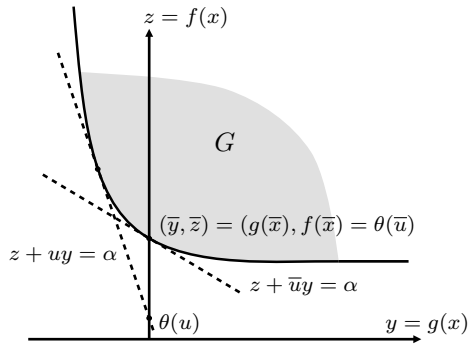


Figure 2: Solving the Lagrangian dual problem is the same as finding the coefficient u such that $z = \alpha - uy$ is a supporting hyperplane for G with the uppermost intercept α with the z-axis. Notice that in the illustrated 'convex' case, the hyperplane corresponding to the maximiser $\overline{u}$ supports G at the point $(\overline{y}, \overline{z}) = (g(\overline{x}), f(\overline{x}))$ that is the image of the solution $\overline{x}$ to P.

settings where the number of constraints is higher. As the existence of supporting hyperplanes at all boundary point is intimately connected to the convexity of the considered set and the convexity of G is obviously closely related to the convexity of f and g, it thus seems that the convexity of the considered optimisation problem altogether plays an essential role for the strong duality to hold.

The *perturbation function*

$$v(y) = \inf_x \{f(x) : g(x) \leq y,\ x \in X\}$$

is an analytical tool that helps understanding when strong duality holds, that is, when the optimal values of the primal and dual problems coincide. Specifically, notice that $v(y)$ is the greatest monotone nonincreasing lower envelope of G, that is,

$$v(y) \leq \inf\{z : (y, z) \in G\},$$

and $v(y_1) \geq v(y_2)$ for $y_1 \leq y_2$. Moreover, if/when $f(\overline{x}) = \theta(\overline{u})$ holds is related to the convexity of $v(y)$, which would imply

$$v(y) \geq v(\overline{y}) - \overline{u}(y - \overline{y}) \text{ for all } y \in \mathbb{R}, \tag{2}$$

where $-\overline{u}$ is a subgradient for v at $\overline{y}$. Notice that for a convex v, (2) is a consequence of Theorem 12 from Lecture 2 (stating that convex sets have supporting hyperplanes for all points on their boundary) and Theorem 5 in Lecture 3 (stating that convex functions have convex epigraphs). As v is nonincreasing, it must hold in (2) that $\overline{u} \geq 0$. Moreover, at the image $\overline{y} = g(\overline{x})$ of the primal solution $\overline{x}$ (if it exists) there is a supporting hyperplane (i.e., a line in our two-dimensional setting) for v that realises the optimal target value of the dual target function $\theta(\overline{u})$ as the z-coordinate of its intercection with the z-axis; see Figures 2 and 3.

A *duality gap* can exist when the perturbation function $v(y)$ does not have supporting hyperplanes at all points in its domain, which would be impossible if v is convex. Figure 3 illustrates a case where

v is not convex, resulting in strong duality to not hold: $\theta(\overline{u}) < f(\overline{x})$.

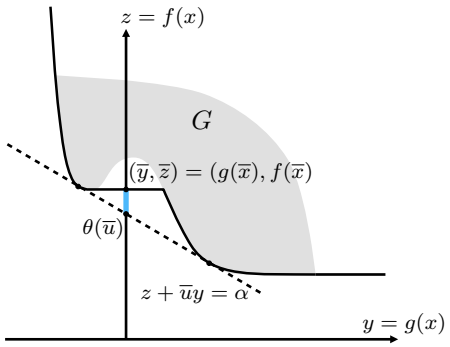


Figure 3: An example in which the perturbation function $v(y)$ and G are not convex. Notice the consequent mismatch between the lowest intercept of a supporting hyperplane with the z-axis and the z-coordinate of the lowest point contained in G. This mismatch corresponds to the duality gap.

Let us illustrate the above considerations with two numerical examples.

Example 1. First, consider the problem

$$\begin{aligned} (P): \quad & \text{min. } x_1^2 + x_2^2, \\ & \text{subject to: } x_1 + x_2 \geq 4, \\ & \qquad\qquad x_1, x_2 > 0. \end{aligned}$$

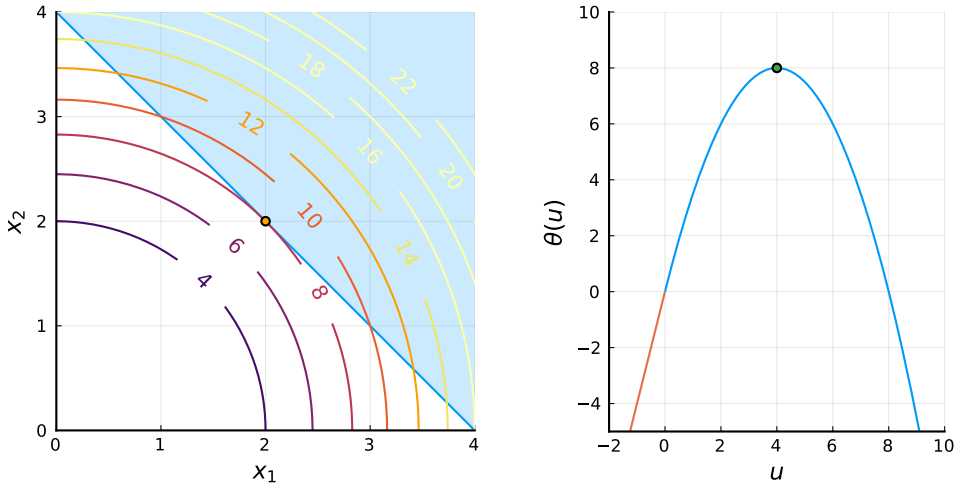
Interpreting the positivity constraints on x_1 and x_2 as the definition of the open set X, the Lagrangian dual function is given by

$$\begin{aligned} \theta(u) &= \inf_x \left\{ x_1^2 + x_2^2 + u(-x_1 - x_2 + 4) : x_1, x_2 > 0 \right\} \\ &= \inf_{x_1} \left\{ x_1^2 - ux_1 : x_1 > 0 \right\} + \inf_{x_2} \left\{ x_2^2 - ux_2 : x_2 > 0 \right\} + 4u \\ &= \begin{cases} -1/2u^2 + 4u & \text{if } u \geq 0, \\ 4u & \text{if } u < 0. \end{cases} \end{aligned}$$

Figures 4a and 4b provide graphical representations of the primal problem P and the dual problem D of maximising $\theta(u)$ over $u \geq 0$. As can be seen, both problems have as the optimal target function value $f(\overline{x}) = \theta(\overline{u}) = 8$, with the optimal solution $\overline{x} = (2, 2)$ for P and $\overline{u} = 4$ for D.

To draw the $G \subset \mathbb{R}^2$, i.e., the image of X under (g, f), we proceed as follows. First, notice that

$$v(y) = \inf_x \left\{ x_1^2 + x_2^2 : -x_1 - x_2 + 4 \leq y;\ x_1, x_2 > 0 \right\},$$



(a) P (b) D

Figure 4: The primal problem P as a constrained optimisation problem, and the dual problem D as an unconstrained optimisation problem.

which shows that $v(y) = 0$ for $y \geq 4$. For $y < 4$, $v(y)$ can be equivalently rewritten as

$$v(y) = \min_x \left\{ x_1^2 + x_2^2 : -x_1 - x_2 + 4 = y;\ x_1, x_2 > 0 \right\}$$

because the infimum must be attained when the sum of x_1 and x_2 is as small as possible — otherwise one could decrease the value of x_1 or x_2 a bit so that still $4 \leq y + x_1 + x_2$ but $x_1^2 + x_2^2$ would decrease.

Let $h_y(x) = -x_1 - x_2 + 4 - y$ and $f(x) = x_1^2 + x_2^2$, which are both convex in x. Not accounting for the positivity constraints on x_1 and x_2, the KKT optimality conditions for $\overline{x}$ to be the point realising the value of $v(y)$ are $h_y(\overline{x}) = 0$ and

$$\nabla f(\overline{x}) + u \nabla h_y(\overline{x}) = 0 \quad \Longrightarrow \quad \begin{cases} 2x_1 - u = 0 \\ 2x_2 - u = 0 \end{cases} \quad \Longrightarrow \quad \overline{x}_1 = \overline{x}_2 = u/2$$

for $u \in \mathbb{R}$. From the definition of $h(x)$, we see that $u = 4 - y$, and thus $\overline{x} = (\frac{4-y}{2}, \frac{4-y}{2})$, which satisfies the positivity constraints as we assumed $y < 4$, and $\overline{x}$ is thus the minimiser of f with the equality constraint $h_y(x) = 0$ over both $\mathbb{R}^2$ and its open positive quadrant.

Evaluating $f(\overline{x})$ gives

$$v(y) = \begin{cases} (4-y)^2/2 & \text{for } y < 4, \\ 0 & \text{for } y \geq 4. \end{cases}$$

Note that v is convex, and thus, in particular, $v(y) \geq v(\overline{y}) + v'(\overline{y})(y - \overline{y})$, where $\overline{y} = g(\overline{x}) =$

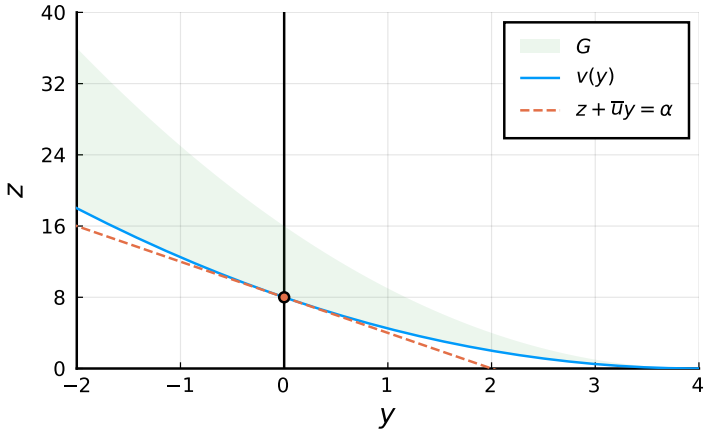


Figure 5: The set G, the perturbation function v and the supporting hyperplane (i.e., line) to G at $(\overline{y}, \overline{z}) = (g(\overline{x}), f(\overline{x}) = (0, 8)$ with the slope $-\overline{u} = -4$ for Example 1.

$\overline{x}_1 - \overline{x}_2 + 4 = 0$. Moreover,

$$v'(\overline{y}) = (4 - \overline{y}) = 4 = -4\overline{u}$$

as predicted by the above considerations. See Figure 5.

Example 2. Let us then consider a problem whose feasible set is not convex and, therefore, the set G will not be convex either. For that, consider the problem

$$\begin{aligned} (P): \quad & \text{min.} \quad -2x_1 + x_2 \\ & \text{subject to:} \quad x_1 + x_2 = 3, \\ & \qquad\qquad x \in X, \end{aligned}$$

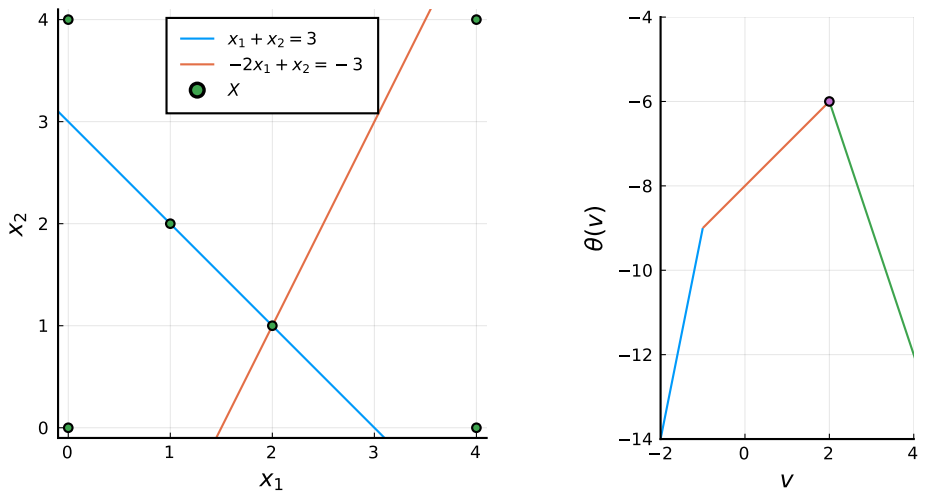
where $X = \{(0,0), (0,4), (4,4), (4,0), (1,2), (2,1)\}$.[1] By going through all points in X, it is easy to see that only $(1,2)$ and $(2,1)$ are feasible, of which $\overline{x} = (2,1)$ is the minimiser of the target function.

The Lagrangian dual function is given by

$$\begin{aligned} \theta(v) &= \min_x \{(-2x_1 + x_2) + v(x_1 + x_2 - 3) : (x_1, x_2) \in X\} \\ &= \begin{cases} -4 + 5v & \text{if } v \leq -1, \\ -8 + v & \text{if } -1 \leq v \leq 2, \\ -3v & \text{if } v \geq 2, \end{cases} \end{aligned}$$

which can be verified, e.g., by evaluating $(-2x_1 + x_2) + v(x_1 + x_2 - 3)$ for all $x \in X$ and comparing the resulting linear functions of $v \in \mathbb{R}$. For $v \leq -1$, the value of $\theta(v)$ is realised by $x = (4,4)$; for

[1] Observe that X is not open as is assumed at many points in these lecture notes. However, it does not affect the validity of this example.



(a) P (b) (D) : max. $\theta(v)$

Figure 6: The primal problem P as a constrained optimisation problem and the the dual problem D. Notice how the Lagrangian dual function is concave and piecewise linear, despite the nonconvex nature of P.

$-1 \leq v \leq 2$, the value of $\theta(v)$ is realised by $x = (4, 0)$; and for $v \geq 2$, the value of $\theta(v)$ is realised by $x = (0, 0)$.

Figure 6a provides a graphical representation of the problem. Notice that to obtain the Lagrangian dual function, one must simply take the lowermost segments of the hyperplanes obtained when considering each $x \in X$, which leads to a piecewise concave function, as represented in Figure 6b.

Similarly to the previous example, we can plot the set G, which in this case consists of the points $x \in X$ mapped as $(h(x), f(x))$, with $h(x) = x_1 + x_2 - 3$ and $f(x) = -2x_1 + x_2$. Notice that v in this case is defined by the points $(h(4, 0), f(4, 0)) = (1, -8)$, $(h(2, 1), f(2, 1)) = (0, -3)$ and $(h(0, 0), f(0, 0)) = (-3, 0)$ in the (y, z)-plane:

$$v(y) = \begin{cases} \infty & \text{for } y < -3 \\ 0 & \text{for } -3 \leq y < 0, \\ -3 & \text{for } 0 \leq y < 1, \\ -8 & \text{for } y \geq 1, \end{cases}$$

which is discontinuous and nonconvex. Clearly, $v(y)$ does not have a supporting hyperplane at the the point $(\overline{y}, \overline{z}) = (h(\overline{x}), f(\overline{x})) = (h(2, 1), f(2, 1)) = (0, -3)$, which illustrates the existence of a duality gap, as confirmed by the fact that $-3 = f(\overline{x}) > \theta(\overline{v}) = -6$.

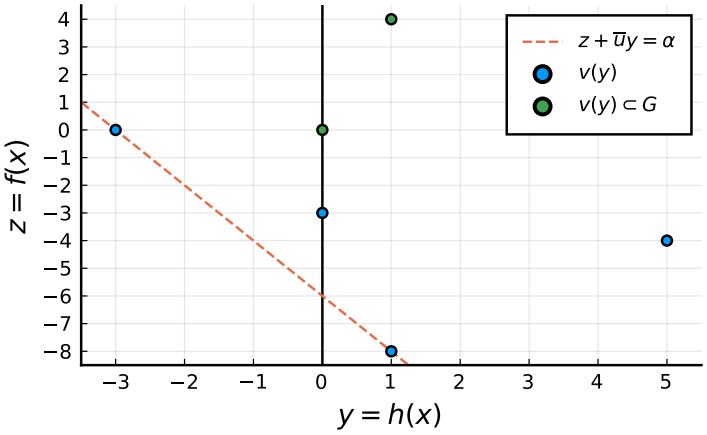


Figure 7: The set G for the second example. The three blue dots on the bottom left represent the values taken by the perturbtion function $v(y)$, which is not convex. Notice that the duality gap is represented by the difference between the intercept of $z = -6 - 2y$, which is the hyperplane supporting G (or the epigraph of v) and having the lowest interception point with the z-axis, and the optimal value of P, i.e., the z-coordinate of $(0, -3)$.

2.1.2 Strong duality

From the previous graphical interpretation and related examples, it becomes clear that there is a strong tie between strong duality and the convexity of P. This is formally described in Theorem 6.

Theorem 6. *Let $X \subseteq \mathbb{R}^n$ be a nonempty convex set. Moreover, let $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ be convex, let $h : \mathbb{R}^n \to \mathbb{R}^l$ be affine (i.e. $h(x) = Ax - b$), and assume there exists x such that $g(x) < 0$. In other words, assume that Slater's constraint qualification holds true. Then*

$$\inf_x \{f(x) : g(x) \leq 0,\ h(x) = 0,\ x \in X\} = \sup_{u,v} \{\theta(u, v) : u \geq 0\},$$

where $\theta(u, v) = \inf_{x \in X} \left\{f(x) + u^\top g(x) + v^\top h(x)\right\}$ is the Lagrangian function. Furthermore, if $\inf_x \{f(x) : g(x) \leq 0,\ h(x) = 0,\ x \in X\}$ is finite and achieved at $\overline{x}$, then $\sup_{u,v} \{\theta(u, v) : u \geq 0\}$ is achieved at $(\overline{u}, \overline{v})$ with $\overline{u} \geq 0$ and $\overline{u}^\top g(\overline{x}) = 0$.

The proof for the strong duality theorem follows the following outline:

1. Let $\gamma = \inf_x \{f(x) : g(x) \leq 0,\ h(x) = 0,\ x \in X\}$. Suppose that $-\infty < \gamma < \infty$, hence finite (for unbounded problems, $f(x) = -\infty$ implies $\theta(u, v) = -\infty$ since $\theta(u, v) \leq f(x)$ from Theorem 3; the right-hand side holds by assumption of the existence of a feasible point from Slater's constraint qualification).

2. Formulate the inconsistent system:

$$f(x) - \gamma < 0, \quad g(x) \leq 0, \quad h(x) = 0, \quad x \in X.$$

3. Use the separation theorem (or a variant form of Farkas theorem) to show that $(\overline{u}_0, \overline{u}, \overline{v})$ with $\overline{u}_0 > 0$ and $\overline{u} \geq 0$ exists such that, after scaling using $\overline{u}_0$ one obtains $\theta(\overline{u}, \overline{v}) := f(x) + \overline{u}^\top g(x) + \overline{v}^\top h(x) \geq \gamma,\ x \in X$, which requires the assumption of Slater's constraint qualification.

4. From weak duality (Theorem 3), we have that $\theta(\overline{u}, \overline{v}) \leq \gamma$, which combined with the above, yields $\theta(\overline{u}, \overline{v}) = \gamma$.

5. Finally, an optimal $\overline{x}$ solving the primal problem implies that $g(\overline{x}) \leq 0$, $h(\overline{x}) = 0$, $\overline{x} \in X$, and $f(x) = \gamma$. From 3, we have $\overline{u}^\top g(\overline{x}) \geq 0$. As $g(\overline{x}) \leq 0$ and $\overline{u} \geq 0$, $\overline{u}^\top g(\overline{x}) \geq 0 = 0$.

The proof uses a variant of the Farkas theorem that states the existence of a solution for the system $u_0(f(x) - \gamma) + u^\top g(x) \geq 0$, $x \in X$ with $(u_0, u, v) \neq 0$, what can be shown to be the case if Slater's constraint qualification holds. This, combined with weak duality stated in Theorem 3 yields strong duality.

2.2 Lagrangian duality: convergence and the KKT conditions

Weak duality can be used to derive stopping criteria for solution methods that can generate both primal and dual feasible solutions, also known as primal-dual pairs. Such methods are typically referred to as primal-dual methods, of which the primal-dual interior point method (which we will discuss in detail in upcoming lectures) is perhaps the most widely known.

For feasible x and (u, v), one can bound how suboptimal $f(x)$ is by noticing that

$$f(x) - f(\overline{x}) \leq f(x) - \theta(u, v),$$

which is a consequence of $f(\overline{x}) \geq \theta(u, v)$ (i.e., weak duality). We say that x is ϵ-optimal, with $\epsilon = f(x) - \theta(u, v)$. In essence, such (u, v) is a certificate for the level of (sub-)optimality of x, i.e., the existence of such a (u, v) proves that x is ϵ-optimal.

Let us then examine the connection of strong Lagrangian duality with the KKT conditions. Assume that strong duality holds, in which case the complementary slackenss condition $\overline{u}^\top g(\overline{x}) = 0$ is satisfied by an optimal primal-dual pair $(\overline{x}, (\overline{u}, \overline{v}))$; cf Theorem 6. Moreover, by definition, $\overline{x}$ and $(\overline{u}, \overline{v})$ are primal and dual feasible, respectively. Moreover, as $\overline{x}$ is a minimiser for $\phi(x, \overline{u}, \overline{v}) = f(x) + \overline{u}^\top g(x) + \overline{v}^\top h(x)$, it must hold that

$$\nabla f(\overline{x}) + \sum_{i=1}^{m} u_i \nabla g_i(\overline{x}) + \sum_{i=1}^{l} v_i \nabla h_i(\overline{x}) = 0.$$

Combining the above, we have listed all the KKT optimality conditions, which under the assumptions of Theorem 6 are known to be necessary and sufficient for global optimality. That is, in this case,

any primal dual pair for which the objective function values match will automatically be a point satisfying the KKT conditions and therefore globally optimal. This provides an alternative avenue to search for optimal solutions, relying on Lagrangian dual problems.

2.3 Saddle point optimality and KKT conditions*

An alternative perspective for establishing necessary and sufficient conditions for strong duality to hold involves identifying the existence of saddle points for the Lagrangian dual problem.

Let us first define saddle points in the context of Lagrangian duality. Let

$$(P): \quad \text{min. } \{f(x) : g(x) \leq 0, h(x) = 0, x \in X\}.$$

Let us define the Lagrangian function $\phi(x, u, v) = f(x) + u^\top g(x) + v^\top h(x)$. A solution $(\overline{x}, \overline{u}, \overline{v})$ is called a *saddle point* if $\overline{x} \in X$, $\overline{u} \geq 0$, and

$$\phi(\overline{x}, u, v) \leq \phi(\overline{x}, \overline{u}, \overline{v}) \leq \phi(x, \overline{u}, \overline{v})$$

for all $x \in X$ and (u, v) such that $u \geq 0$.

Notice that this definition implies that:

- $\overline{x}$ minimises $\phi(x, u, v)$ when (u, v) is fixed at $(\overline{u}, \overline{v})$;
- $(\overline{u}, \overline{v})$ maximises $\phi(x, u, v)$ when x is fixed at $\overline{x}$.

This insight allows for the development of methods that can alternatively solve the Lagrangian dual problem in the space of primal variables x and dual variables (u, v) in a block-coordinate descent fashion.

Theorem 7 establishes the relationship between the existence of saddle points for Lagrangian dual problems and zero duality gaps.

Theorem 7 (Saddle point optimality and zero duality gap)**.** *A solution $(\overline{x}, \overline{u}, \overline{v})$ with $\overline{x} \in X$ and $\overline{u} \geq 0$ is a saddle point for the Lagrangian function $\phi(x, u, v) = f(x) + u^\top g(x) + v^\top h(x)$ if and only if:*

1. $\phi(\overline{x}, \overline{u}, \overline{v}) =$ *min.* $\{\phi(x, \overline{u}, \overline{v}) : x \in X\}$
2. $g(\overline{x}) \leq 0$, $h(\overline{x}) = 0$, *and*
3. $\overline{u}^\top g(\overline{x}) = 0$

Moreover, $(\overline{x}, \overline{u}, \overline{v})$ is a saddle point if and only if $\overline{x}$ and $(\overline{u}, \overline{v})$ are optimal solutions for the primal (P) and dual (D) problems, respectively, with $f(\overline{x}) = \theta(\overline{u}, \overline{v})$.

From Theorem 7 it becomes clear that there is a strong connection between the existence of saddle points and the KKT conditions for optimality. Figure 8 illustrates the existence of a saddle point and the related zero optimality gap.

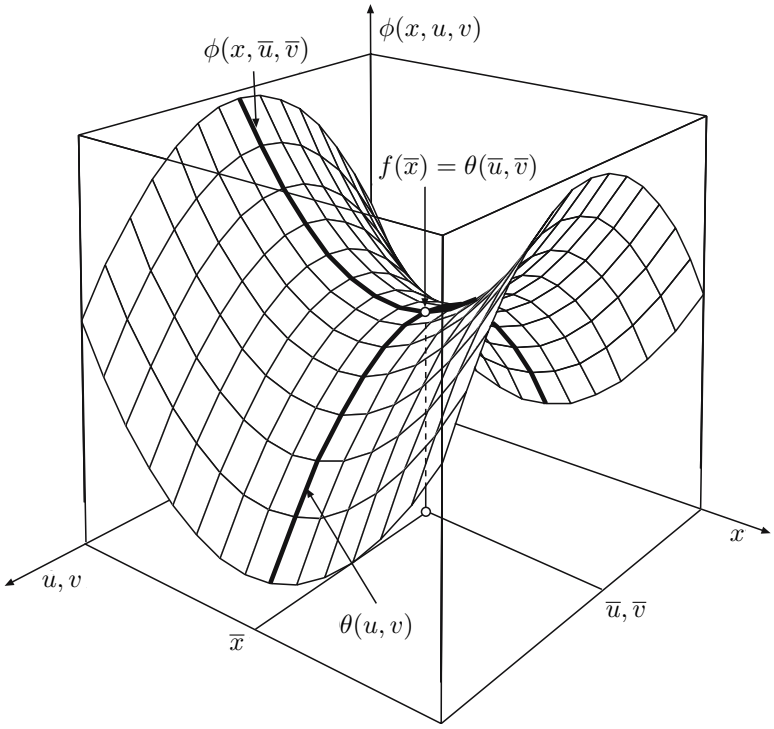


Figure 8: Illustration of a saddle point for the Lagrangian dual problem

3 Properties of Lagrangian functions

Lagrangian duals are a useful framework for devising solution methods for constrained optimisation problems if solving the dual problem can be done efficiently or it carries some exploitable structure.

One important property that Lagrangian dual functions present is that they are *concave piecewise linear* in the dual multipliers. Moreover, they are continuous and thus have subgradients everywhere.[2] Notice however that they are typically not differentiable, requiring the employment of a *nonsmooth optimisation* method to be appropriately optimised. Theorem 8 establishes the concavity of the Lagrangian dual function.

Theorem 8 (Concavity of Lagrangian dual functions)**.** *Let $X \subseteq \mathbb{R}^n$ be a nonempty compact set and let $f : \mathbb{R}^n \to \mathbb{R}$ and $\beta : \mathbb{R}^n \to \mathbb{R}^{m+l}$, defined by $\beta(x) = \binom{g(x)}{h(x)}$, be continuous. Then,*

[2]We abuse the terminology by calling ξ a subgradient for a concave function if $-\xi$ is a subgradient for minus that function. The appropriate term for ξ would arguably be "supergradient".

$\theta(w) = \inf_x \{f(x) + w^\top \beta(x) : x \in X\}$ *is concave in* $\mathbb{R}^{m+l}$

Proof. Homework. □

Theorem 9. *Let* $X \subset \mathbb{R}^n$ *be a nonempty compact set and let* $f : \mathbb{R}^n \to \mathbb{R}$ *and* $\beta : \mathbb{R}^n \to \mathbb{R}^{m+l}$*, defined by* $\beta(x) = \binom{g(x)}{h(x)}$*, be continuous. If* $\overline{x} \in X(\overline{w}) = \{x \in X : x = \arg\min \{f(x) + \overline{w}^\top \beta(x)\}\}$*, then* $-\beta(\overline{x})$ *is a subgradient for* $-\theta$ *at* $\overline{w}$*.*

Proof. Since f and β are continuous and X is compact, $X(\overline{w}) \neq \emptyset$ for any $\overline{w} \in \mathbb{R}^{m+l}$. Now, let $w, \overline{w} \in \mathbb{R}^{m+l}$ and $\overline{x} \in X(\overline{w})$. Then,

$$\begin{aligned}\theta(w) &= \inf \{f(x) + w^\top \beta(x) : x \in X\} \\ &\leq f(\overline{x}) + w^\top \beta(\overline{x}) \\ &= f(\overline{x}) + (w - \overline{w})^\top \beta(\overline{x}) + \overline{w}^\top \beta(\overline{x}) \\ &= \theta(\overline{w}) + (w - \overline{w})^\top \beta(\overline{x}).\end{aligned}$$

□

As $w \in \mathbb{R}^{m+l}$ is arbitrary, multiplying this inequality by -1 proves the claim.

Theorem 9 can be used to derive a simple optimisation method for Lagrangian functions using subgradient information that is easily available from the term $\beta(w)$.

3.1 The subgradient method

One challenging aspect concerning the maximization of Lagrangian dual functions is that they very often are not differentiable. This observation leads to an adaptation of the gradient method, considering subgradient information instead.

The main challenge with using subgradients (instead of gradients) is that subgradients are not guaranteed to be ascent directions (as opposed to gradients being steepest ascent directions). Nevertheless, for suitable step size choices, convergence can be observed if using subgradients in place of gradients. Figure 9 illustrates the fact that subgradients are not necessarily ascent directions.

Algorithm 1 summarises the subgradient method. The stopping criterion can be heuristically driven such as a maximum number of iterations or a given number of iterations without observable improvement in the value of $\theta(w)$.

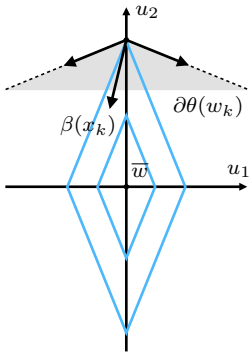


Figure 9: One possible subgradient $\beta(x_k)$ that is an ascent direction (for a maximisation problem). Notice that within the subdifferential $\partial\theta(w_k)$, there are also subgradients that are not ascent directions.

Algorithm 1 Subgradient method for maximising the dual function

---

1: **initialise.** initial point w_0, the best best-available lower-estimate LB_0 for $\theta(\overline{w})$, iteration count $k = 0$.
2: **while** stopping criterion is not satisfied **do**
3: $\quad x_k = \arg\min_x \left\{ f(x) + w_k^\top \beta(x) \right\}$
4: $\quad \theta(w_k) = f(x_k) + w_k^\top \beta(x_k)$
5: $\quad \mathrm{LB}_k = \max\left\{ \mathrm{LB}_k, \theta(w_k) \right\}$
6: $\quad$ update λ_k
7: $\quad w_{k+1} = w_k + \lambda_k \beta(x_k)$.
8: $\quad k \leftarrow k + 1$.
9: **end while**
10: **return** $\mathrm{LB}_k = \theta(w_k)$.

One critical aspect associated with the subgradient method is the step size update described in Step 6 of Algorithm 1. Theoretical convergence is guaranteed if Step 6 generates a sequence $\{\lambda_k\}$ such that $\sum_{k=0}^{\infty} \lambda_k \to \infty$ and $\lim_{k\to\infty} \lambda_k = 0$. However, discrepant performance can be observed for distinct parametrisations of the method.

The classical step update rule employed for the subgradient method is known as the Polyak rule, a version of which is given by

$$\lambda_{k+1} = \frac{\alpha_k(\delta + \mathrm{LB}_k - \theta(w_k))}{\|\beta(x_k)\|^2} \tag{3}$$

with $\alpha_k \in (0, 2)$ and a suitable small $\delta > 0$. Recall that LB_k is the best-available lower-estimate for $\theta(\overline{w})$. This rule is inspired by the following result.

Proposition 10 (Improving step size)**.** *Consider Algorithm 1. If w_k is not optimal, then for all optimal dual solutions $\overline{w}$, we have*

$$\|w_{k+1} - \overline{w}\| < \|w_k - \overline{w}\|$$

for all step sizes λ_k such that

$$0 < \lambda_k < \frac{2(\theta(\overline{w}) - \theta(w_k))}{\|\beta(x_k)\|^2}.$$

Proof. We have that

$$\begin{aligned}\|w_{k+1} - \overline{w}\|^2 &= \|w_k + \lambda_k \beta(x_k) - \overline{w}\|^2 \\ &= \|w_k - \overline{w}\|^2 - 2\lambda_k(\overline{w} - w_k)^\top \beta(x_k) + \lambda_k^2 \|\beta(x_k)\|^2.\end{aligned}$$

By Theorem 9 and the subgradient inequality at w_k: $\theta(\overline{w}) - \theta(w_k) \leq (\overline{w} - w_k)^\top \beta(x_k)$. Thus,

$$\|w_{k+1} - \overline{w}\|^2 \leq \|w_k - \overline{w}\|^2 - 2\lambda_k(\theta(\overline{w}) - \theta(w_k))^\top + \lambda_k^2 \|\beta(x_k)\|^2.$$

Parametrising the last two terms by $\gamma_k = \frac{\lambda_k \|\beta(x_k)\|^2}{\theta(\overline{w}) - \theta(w_k)}$ leads to

$$\|w_{k+1} - \overline{w}\|^2 \leq \|w_k - \overline{w}\|^2 - \frac{\gamma_k(2 - \gamma_k)(\theta(\overline{w}) - \theta(w_k))^2}{\|\beta(x_k)\|^2}.$$

Notice that if $0 < \lambda_k < \frac{2(\theta(\overline{w}) - \theta(w_k))}{\|\beta(x_k)\|^2}$, then $0 < \gamma_k < 2$, and thus, $\|w_{k+1} - \overline{w}\| < \|w_k - \overline{w}\|$. □

In practice, since $\theta(\overline{w})$ is not known, it must be replaced by the proxy $\mathrm{LB}_k + \delta$ in (3).

Lecture 9 - Constrained optimisation methods: penalty methods

Fabricio Oliveira (with modifications by Nuutti Hyvönen)

Last update: November 9, 2022

Abstract. In this lecture, we discuss the strategy of employing penalty functions to solve constrained optimisation problems. We also discuss the concept of penalised functions and demonstrate their asymptotical convergence properties. We consider a specific penalty method that has ties with Lagrangian duality, the augmented Lagrangian method of multipliers (ALMM). Moreover, we present a variant of ALMM, the alternating direction method of multipliers (ADMM), that allows for parallelisation of problems with suitable structure.

Outline of this lecture

1 Penalty functions

The employment of penalty functions is a paradigm for solving constrained optimisation problems. The central idea of this paradigm is to convert the constrained optimisation problem into an unconstrained optimisation problem that is augmented with a *penalty function*, which penalises violations of the original constraints. The role of the penalty function is to allow steering the search towards feasible solutions in the search for optimal ones.

Consider the problem

$$(P) : \text{min. } \{f(x) : g(x) \leq 0,\ h(x) = 0,\ x \in X\}.$$

A *penalised version* of P is given by

$$(P_\mu) : \text{min. } \{f(x) + \mu\alpha(x) : x \in X\},$$

where $\mu > 0$ is a *penalty term* and $\alpha(x) : \mathbb{R}^n \to \mathbb{R}$ is a *penalty function* of the form

$$\alpha(x) = \sum_{i=1}^{m} \phi(g_i(x)) + \sum_{i=1}^{l} \psi(h_i(x)). \tag{1}$$

For $\alpha(x)$ to be a suitable penalty function, one must make sure that $\phi : \mathbb{R} \to \mathbb{R}$ and $\psi : \mathbb{R} \to \mathbb{R}$ are continuous and satisfy

$$\phi(y) = 0 \text{ if } y \leq 0 \text{ and } \phi(y) > 0 \text{ if } y > 0, \tag{2}$$
$$\psi(y) = 0 \text{ if } y = 0 \text{ and } \psi(y) > 0 \text{ if } y \neq 0. \tag{3}$$

Typical options are $\phi(y) = ([y]^+)^p$ with $p \in \mathbb{N}$ and $\psi(y) = |y|^p$ with $p = 1$ or $p = 2$. Here, $[y]^+$ takes the value 0 if $y \leq 0$ and the value y otherwise.

Figure 1 illustrates the solution of (P) : min. $\left\{x_1^2 + x_2^2 : x_1 + x_2 = 1,\ x \in \mathbb{R}^2\right\}$ using a penalty-based approach; notice that the solution of P is $\overline{x} = (\frac{1}{2}, \frac{1}{2})$ that corresponds to the objective function value $f(\overline{x}) = \frac{1}{2}$. Using $\alpha(x_1, x_2) = (x_1 + x_2 - 1)^2$, the penalised auxiliary problem P_μ becomes (P_μ) : min. $\left\{x_1^2 + x_2^2 + \mu(x_1 + x_2 - 1)^2 : x \in \mathbb{R}^2\right\}$. Since f_μ is convex and differentiable, necessary and sufficient optimality conditions $\nabla f_\mu(x) = 0$ imply:

$$x_1 + \mu(x_1 + x_2 - 1) = 0$$
$$x_2 + \mu(x_1 + x_2 - 1) = 0,$$

which gives $x_1 = x_2 = \frac{\mu}{2\mu+1}$.

As μ increases, the solution of the unconstrained penalised problem, represented by the level curves, moves closer to the solutions of the original constrained problem P, represented by the dot on the

hyperplane defined by $x_1 + x_2 = 1$.

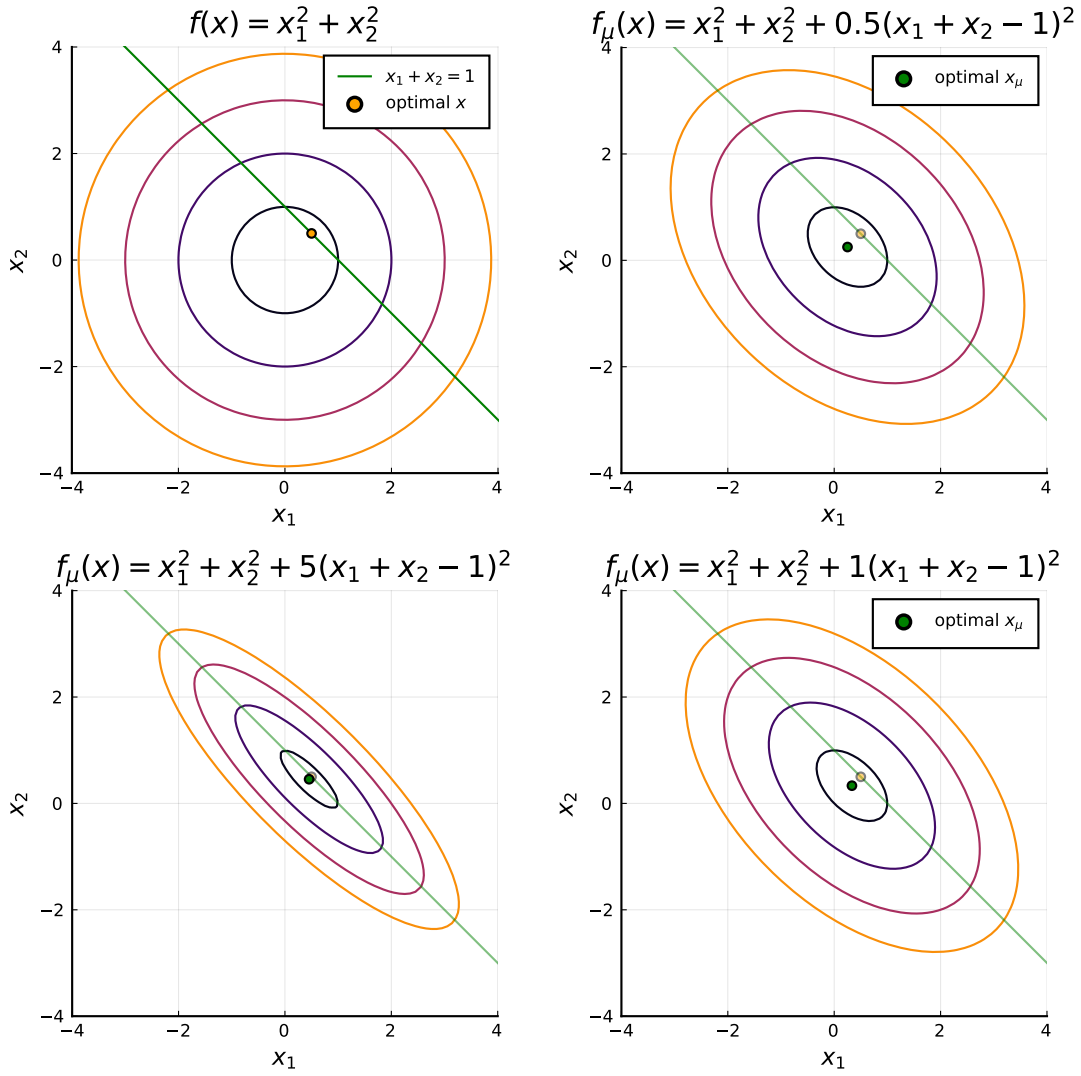


Figure 1: Solving the constrained problem P (top left) by gradually increasing the penalty term μ (0.5, 1, and 5, in clockwise order)

1.1 Geometric interpretation

To give a geometric interpretation for a penalised problem with a quadratic penalty function and mere equality constraints, notice that

$$\begin{aligned}
&\min_{x\in X}\left\{f(x)+\mu\sum_{i=1}^{l}h_i(x)^2\right\}\\
&\quad=\min_{x\in X,\,\epsilon\in\mathbb{R}^l}\left\{f(x)+\mu\|\epsilon\|^2 : h_i(x)=\epsilon_i,\ i=1,\dots,l\right\}\\
&\quad=\min_{\epsilon\in\mathbb{R}^l}\left\{\mu\|\epsilon\|^2+\min_{x\in X}\left\{f(x) : h_i(x)=\epsilon_i,\ i=1,\dots,l\right\}\right\}\\
&\quad=\min_{\epsilon\in\mathbb{R}^l}\left\{\mu\|\epsilon\|^2+v(\epsilon)\right\},
\end{aligned}\tag{4}$$

where

$$v(\epsilon)=\min_{x\in X}\left\{f(x) : h_i(x)=\epsilon_i,\ i=1,\dots,l\right\}$$

gives the optimal objective function value for ϵ-perturbed constraints and no penalisation.

To facilitate visualisation, consider $l=1$, let

$$x_\mu=\arg\min_{x\in X}\left\{f(x)+\mu h(x)^2\right\}$$

be a solution to the penalised problem and denote by $\epsilon_\mu=h(x_\mu)$ the corresponding discrepancy in the equality constraint. Due to the reformulated form (4) of the penalised problem for $l=1$, it holds that

$$f(x_\mu)+\mu h(x_\mu)^2=\mu\epsilon_\mu^2+v(\epsilon_\mu),\quad\text{i.e.,}\quad f(x_\mu)=v(\epsilon_\mu).$$

Moreover, as ϵ_μ is the minimiser for (4), it must hold

$$\frac{\mathrm{d}}{\mathrm{d}\epsilon}\left(\mu\epsilon^2+v(\epsilon)\right)|_{\epsilon=\epsilon_\mu}=0,\quad\text{i.e.,}\quad v'(\epsilon_\mu)=-2\mu\epsilon_\mu\tag{5}$$

assuming sufficient regularity from f and h.

Let us then consider the set $G=\{(\epsilon,z) : \epsilon=h(x),\ z=f(x),\, x\in X\}\subset\mathbb{R}^2$ in the (ϵ,z)-plane, where the horizontal axis corresponds to the value of h, i.e. the discrepancy in the equality constraint, and the vertical axis to the value of the objective function f. The function $v(\epsilon)$ gives the minimimal value for $f(x)$ under the constraint $h(x)=\epsilon$, i.e., $v(\epsilon)$ is the lowest point in G at the horizontal coordinate ϵ. According to the above analysis $(h(x_\mu),f(x_\mu))=(\epsilon_\mu,v(\epsilon_\mu))$, that is, the solution of the penalised problem with the penalty term $\mu>0$ corresponds to the value ϵ_μ for the constraint h and to the value $v(\epsilon_\mu)$ for the original objective function f. Moreover, denoting the optimal value of the penalised objective function by $k_\mu=f(x_\mu)+\mu h(x_\mu)^2$, one sees that the parabola

$$z=q_\mu(\epsilon):=k_\mu-\mu\epsilon^2\tag{6}$$

matches v at $\epsilon = \epsilon_\mu$ both in value,

$$q_\mu(\epsilon_\mu) = k_\mu - \mu\epsilon_\mu^2 = f(x_\mu) + \mu h(x_\mu)^2 - \mu\epsilon_\mu^2 = f(x_\mu) = v(\epsilon_\mu),$$

and in slope

$$q'_\mu(\epsilon_\mu) = -2\mu\epsilon_\mu = v'(\epsilon_\mu)$$

by virtue of (5). Finding the minimal value for $f(x) + \mu h(x)^2$ thus consists in choosing the constant term k_μ in (6) so that the corresponding parabola tangents the (convex) graph of v. As $\mu \to \infty$, the parabola becomes sharper and sharper, and $v(\epsilon_\mu)$ *converges* to the optimal value of the objective function f under the original equality constraint $h(x) = 0$; at the limit, the parabola becomes needle-like along the z-axis, and its only intersection point with the graph of v is $(0, v(0))$. Figure 2 illustrates this behaviour for the setting of Figure 1 as explained in detail in the following.

Let us then return to the setting of Figure 1, with $f(x) = x_1^2 + x_2^2$ and $h(x) = x_1 + x_2 - 1$. The corresponding set G, function v, parabola q_μ and discrepancy ϵ_μ (for $\mu_2 > \mu_1$ and $\overline{\mu} = \infty$) are illustrated in Figure 2. It is straightforward to deduce that in this setting

$$v(\epsilon) = \frac{(1+\epsilon)^2}{2}$$

corresponding to

$$\frac{1}{2}(1+\epsilon, 1+\epsilon) = \arg\min_{x \in \mathbb{R}^2} \left\{ x_1^2 + x_2^2 : x_1 + x_2 - 1 = \epsilon \right\}.$$

Moreover, since the minimiser of the penalised target function, i.e. $x_\mu = (\frac{\mu}{2\mu+1}, \frac{\mu}{2\mu+1})$, is known from above, we have

$$\epsilon_\mu = h(x_\mu) = \frac{2\mu}{2\mu+1} - 1 = -\frac{1}{2\mu+1},$$

which obviously converges to 0 as μ tends to infinity. Moreover, according to (6), the equation of the parabola tangenting the graph of v at ϵ_μ is

$$z = f(x_\mu) + \mu h(x_\mu)^2 - \mu\epsilon^2 = \frac{2\mu^2}{(2\mu+1)^2} + \frac{\mu}{(2\mu+1)^2} - \mu\epsilon^2 = \frac{2\mu^2+\mu}{(2\mu+1)^2} - \mu\epsilon^2.$$

Note that the intersection of this parabola with the z-axis converges to

$$\lim_{\mu\to\infty} \frac{2\mu^2+\mu}{(2\mu+1)^2} = \frac{1}{2},$$

which is the value of the target function $f(x) = x_1^2 + x_2^2$ at optimum $\overline{x} = (\frac{1}{2}, \frac{1}{2})$ of the original equality-constrained minimisation problem P.

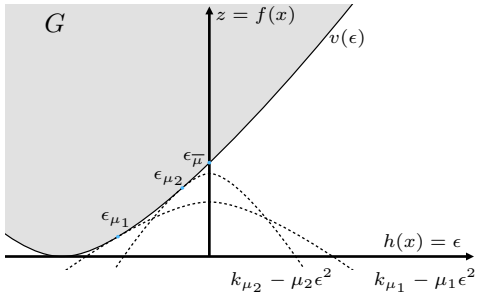


Figure 2: Geometric representation of penalised problems in the coordinates $(z, \epsilon) = (h(x), f(x))$.

1.2 Penalty function methods

The convergent behaviour of the penalised problem as the penalty term μ increases inspires the development of a simple yet powerful method for optimising constrained optimisation problems.

That is, consider the problem P defined as

$$\begin{aligned}(P) : \text{min. } & f(x) \\ & g_i(x) \leq 0,\ i = 1, \ldots, m, \\ & h_i(x) = 0,\ i = 1, \ldots, l, \\ & x \in X.\end{aligned}$$

We seek to solve P by solving $\sup_{\mu} \{\theta(\mu)\}$ for $\mu > 0$, where

$$\theta(\mu) = \min \{f(x) + \mu\alpha(x) : x \in X\}$$

and $\alpha(x)$ is a penalty function as defined in (1). For that to be possible, we need first to state a convergence result guaranteeing that

$$\min \{f(x) : g(x) \leq 0,\ h(x) = 0,\ x \in X\} = \sup_{\mu \geq 0} \theta(\mu) = \lim_{\mu \to \infty} \theta(\mu).$$

In practice, that would mean that μ_k can be increased at each iteration k until a suitable tolerance is achieved. Theorem 1 states the convergence of penalty based methods.

Theorem 1 (Convergence of penalty-based methods)**.** *Consider the problem P and assume it is feasible. Let $f : X \to \mathbb{R}$, $g_i : X \to \mathbb{R}$ for $i = 1, \ldots, m$, and $h_i : X \to \mathbb{R}$ for $i = 1, \ldots, l$ be continuous and $X \subset \mathbb{R}^n$ a compact set. Suppose that, for each $\mu > 0$, there exists $x_\mu = \arg\min \{f(x) + \mu\alpha(x) : x \in X\}$, where α is a suitable penalty function satisfying* (2)–(3)*. Then*

$$\min_{x} \{f(x) : g(x) \leq 0,\ h(x) = 0,\ x \in X\} = \sup_{\mu \geq 0} \{\theta(\mu)\} = \lim_{\mu \to \infty} \theta(\mu),$$

where $\theta(\mu) = \min_x \{f(x) + \mu\alpha(x) : x \in X\} = f(x_\mu) + \mu\alpha(x_\mu)$. *Moreover, the limit* $\overline{x} \in X$ *of any converging sequence* $\{x_{\mu_k}\}_{k=1}^{\infty}$, *with* $\mathbb{R}_+ \ni \mu_k \to \infty$ *as* $k \to \infty$, *is feasible and optimal for the original problem* P, *and* $\mu_k\alpha(x_{\mu_k}) \to 0$ *as* $k \to \infty$.

Proof. We first show that $\theta(\mu)$ is a nondecreasing function of μ. From the definition of $\theta(\lambda)$ it follows that

$$f(x_\mu) + \lambda\alpha(x_\mu) \geq f(x_\lambda) + \lambda\alpha(x_\lambda) = \theta(\lambda) \tag{7}$$

for $0 < \lambda < \mu$. Adding and subtracting $\mu\alpha(x_\mu)$ in the left side of (7) leads to

$$\theta(\mu) \geq f(x_\mu) + \mu\alpha(x_\mu) + (\lambda - \mu)\alpha(x_\mu) = f(x_\mu) + \lambda\alpha(x_\mu) \geq f(x_\lambda) + \lambda\alpha(x_\lambda) = \theta(\lambda)$$

because $\alpha(x_\mu) \geq 0$ by (2)–(3) and $\lambda - \mu < 0$ by assumption.

Pick any $y \in X$ such that $g(y) \leq 0$ and $h(y) = 0$; such y exists as P is feasible. According to (2)–(3), it holds that $\alpha(x) = 0$, and thus

$$f(y) = f(y) + \mu\alpha(y) \geq \min_x \{f(x) + \mu\alpha(x) : x \in X\} = \theta(\mu), \tag{8}$$

which means that $\theta(\mu)$ is bounded from above. As θ is nondecreasing and bounded from above,

$$\sup_{\mu \geq 0} \theta(\mu) = \lim_{\mu \to \infty} \theta(\mu).$$

For that to be the case, we must have that $\alpha(x_\mu) \to 0$ as $\mu \to \infty$ because otherwise $\mu\alpha(x_\mu)$ would diverge and $\theta(\mu)$ could not stay bounded as $\mu \to \infty$. (Note that f is bounded over the compact set X as it is continuous.)

As y is an arbitrary point satisfying the constraints of the original minimisation problem, it follows from (8) that

$$\min_x \{f(x) : g(x) \leq 0,\ h(x) = 0,\ x \in X\} \geq \lim_{\mu \to \infty} \theta(\mu). \tag{9}$$

As $\{x_\mu\}_{\mu \in \mathbb{R}_+} \subset X$ and X is compact, any sequence $\{x_{\mu_j}\}_{j=1}^{\infty}$, with $\lim_{j \to \infty} \mu_j = \infty$, contains a convergent infinite subsequence. Consider any such converging (sub)sequence $\{x_{\mu_k}\}_{k=1}^{\infty}$ with the limit $\overline{x}$. Since $\alpha(x_{\mu_k})$ is known to converge to zero as $\mu_k \to \infty$ and α is continuous by (2)–(3), it follows that $\alpha(\overline{x}) = 0$, and thus $\overline{x}$ is feasible for the original problem P. Moreover,

$$\lim_{\mu \to \infty} \theta(\mu) = \sup_{\mu \geq 0} \theta(\mu) \geq \theta(\mu_k) = f(x_{\mu_k}) + \mu\alpha(x_{\mu_k}) \geq f(x_{\mu_k}).$$

Since $x_{\mu_k} \to \overline{x}$ as $k \to \infty$ and f is continuous, this implies that

$$\lim_{\mu \to \infty} \theta(\mu) = \sup_{\mu \geq 0} \theta(\mu) \geq f(\overline{x}) + \lim_{k \to \infty} \mu_k\alpha(x_{\mu_k}) \geq f(\overline{x}).$$

Combined with (9) and the feasiblity of $\overline{x}$ for P, we have that

$$f(\overline{x}) = \min_x \{f(x) : g(x) \leq 0,\ h(x) = 0,\ x \in X\}$$

and $\lim_{k\to\infty} \mu_k \alpha(x_{\mu_k}) = 0$. □

The proof starts by demonstrating the nonincreasing behaviour of penalty functions and nondecreasing behaviour of $\theta(\mu)$ to allow for convergence. By noticing that

$$f(x_\mu) + \lambda\alpha(x_\mu) + \mu\alpha(x_\mu) - \mu\alpha(x_\mu) = \theta(\mu) + (\lambda - \mu)\alpha(x_\mu) \geq f(x_\lambda) + \lambda\alpha(x_\lambda) = \theta(\lambda)$$

and that $\lambda - \mu < 0$, we can infer that $\theta(\mu) \geq \theta(\lambda)$. It is also interesting to notice how the objective function $f(x)$ and the 'infeasibility' $\alpha(x)$ behave as we increase the penalty coefficient μ. For that, notice that using the same trick as in the proof, for two distinct values $0 < \lambda < \mu$, we have

1. $f(x_\mu) + \lambda\alpha(x_\mu) \geq f(x_\lambda) + \lambda\alpha(x_\lambda)$,
2. $f(x_\lambda) + \mu\alpha(x_\lambda) \geq f(x_\mu) + \mu\alpha(x_\mu)$.

In 1, we use the fact that $x_\lambda = \arg\min_x \theta(\lambda) = \arg\min_x \{f(x) + \lambda\alpha(x)\}$ and therefore, $f(x_\lambda) + \lambda\alpha(x_\lambda)$ must be less or equal to $f(x) + \lambda\alpha(x)$ for any $x \in X$ or, in particular, for $x = x_\mu \in X$. The same logic is employed in 2, but by changing the roles of λ and μ. Adding 1 and 2, we obtain $(\mu - \lambda)(\alpha(x_\lambda) - \alpha(x_\mu)) \geq 0$ and conclude that $\alpha(x_\mu) \leq \alpha(x_\lambda)$ for $\mu > \lambda$, i.e., that $\alpha(x)$ is nonincreasing in μ.

Moreover, from the first inequality, we have that $f(x_\mu) \geq f(x_\lambda)$. Notice how this goes in line with what one would expect from the method: as we increase the penalty coefficient μ, the optimal infeasibility, measured by $\alpha(x_\mu)$, decreases, while the objective function value $f(x_\mu)$ worsens as it is slowly "forced" to be closer to the original feasible region.

Note that the assumption of compactness plays a central role in this proof, such that $\theta(\mu)$ can be evaluated for any μ as $\mu \to \infty$. Though this is a strong assumption, it tends to not be so restrictive in practical cases, since variables typically have finite lower and upper bounds. Finally, notice that $\alpha(\overline{x}) = 0$ implies that $\overline{x}$ is feasible for g_i for $i = 1, \ldots, m$, and h_i for $i = 1, \ldots, l$, and thus optimal for P. This is related to the following corollary.

Corollary 2. *If $\alpha(x_\mu) = 0$ for some μ, then x_μ is optimal for P.*

Proof. Denote by S the subset of X that consists of those points that are feasible for P, that is, the points in $S \subset X$ satisfy the inequality and equality constraints of P. The problem

$$\min_{x\in X}\ \{f(x) + \mu\alpha(x) : x \in X\}$$

is a relaxation of P as defined in Definition 1 of Lecture 8. Indeed, (1) $f(x) + \mu\alpha(x) = f(x)$ for all $x \in S$ because $\alpha(x) = 0$ for $x \in S$. Moreover, the feasible set of the penalised problem is $S_R = X$, and so (2) $S \subseteq S_R$. Since $\alpha(x_\mu) = 0$, i.e., $x_\mu \in S$, and

$$\min_{x \in X} \{f(x) + \mu\alpha(x) : x \in X\} = f(x_\mu) + \mu\alpha(x_\mu) = f(x_\mu),$$

the 'penalised minimiser' x_μ is optimal for P as well due to Theorem 2 of Lecture 8. □

A technical detail of the proof of Theorem 1 is that the convergence is asymptotical, i.e., by making μ arbitrarily large, x_μ can be forced to be arbitrarily close to the true minimiser $\overline{x}$ and $\theta(\mu)$ can be forced to be arbitrarily close the optimal objective value $f(\overline{x})$. In practice, this strategy tends to be prone to computational instability.

The computational instability arises from the influence that the penalty term exerts in some of the eigenvalues of the Hessian of the penalised problem. Let $H_\mu(x_\mu)$ be the Hessian of the penalised function at x_μ. Recall that conditioning is measured by $\kappa = \frac{\max_{i=1,\ldots,n} |\lambda_i|}{\min_{i=1,\ldots,n} |\lambda_i|}$, where $\{\lambda_i\}_{i=1,\ldots,n}$ are the eigenvalues of $H_\mu(x_\mu)$. Since the influence of penalisation is often more pronounced on some of the eigenvalues than others, the conditioning of the problem is affected, which might lead to numerical instabilities. This phenomenon is visible in Figure 1, where one can notice the elongated profile of the to-be-minimised function as the penalty term μ increases.

Let us consider the setting of Figure 1 in more detail from this standpoint. The Hessian of the penalised function $f_\mu(x) = x_1^2 + x_2^2 + \mu(x_1 + x_2 - 1)^2$ is

$$\nabla^2 f_\mu(x) = \begin{bmatrix} 2(1+\mu) & 2\mu \\ 2\mu & 2(1+\mu) \end{bmatrix}.$$

Solving $\mathbf{det}(\nabla^2 f_\mu(x) - \lambda I) = 0$, we obtain $\lambda_1 = 2$ and $\lambda_2 = 2(1 + 2\mu)$, with the corresponding eigenvectors $(1, -1)$ and $(1, 1)$, which gives $\kappa = 1 + 2\mu$. This illustrates that the condition number is asymptotically proportional to the penalty term $\mu > 0$.

2 Augmented Lagrangian method of multipliers

For simplicity, consider the (primal) problem

$$(P) : \text{min. } \{f(x) : h_i(x) = 0,\ i = 1, \ldots, l,\ x \in X\}, \tag{10}$$

with mere equality constraints. The augmented Lagrangian method of multipliers arises from the idea of seeking for a penalisation formulation that would allow for exact convergence for a finite penalty coefficient.

Considering the geometrical interpretation in Figure 2 for $l = 1$, one notices that shifting a considered parabola suitably right and upward would move the point where it tangents the graph of v (closer) to the z-axis that corresponds to the value $\epsilon = h(x) = 0$. Motivated by this observation, consider minimising

$$f_{\mu,w}(x) = f(x) + wh(x) + \mu h(x)^2, \qquad w \in \mathbb{R},\ \mu \in \mathbb{R}_+, \tag{11}$$

instead of $f(x) + \mu h(x)^2$. Denoting the minimimum value of $f_{\mu,w}(x)$ by $k_{\mu,w}$, as in Figure 2 we see that the parabola

$$z = q_{\mu,w}(\epsilon) := k_{\mu,w} - w\epsilon - \mu\epsilon^2 = \Big(k_{\mu,w} + \frac{w^2}{4\mu}\Big) - \mu\Big(\frac{w}{2\mu} + \epsilon\Big)^2$$

should touch the (convex) graph of v at the horizontal position $\epsilon = h(x_{\mu,w})$, with $x_{\mu,w}$ being the point where the minimum of $f_{\mu,w}$ is attained. Compared to the parabolas in Figure 2 attaining their maxima at $\epsilon = 0$, the maximum of $q_{\mu,w}$ is attained at $\epsilon = -\frac{w}{2\mu}$. Hence, including the term $wh(x)$ in (11) allows to shift the considered tangenting parabola vertically — and choosing w 'appropriately' allows to force the minimising point $x_{\mu,w}$ to satisfy the constraint $h(x_{\mu,w}) = 0$; see Figure 3. In fact, w has an interpretation as a Lagrange multiplier of a problem closely related to P, as we will reason below.

Motivated by the above observations let use define the *augmented Lagrangian*

$$L_\mu(x, w) = f(x) + \sum_{i=1}^{l} w_i h_i(x) + \mu \sum_{i=1}^{l} h_i(x)^2, \qquad x \in \mathbb{R}^n,\ w \in \mathbb{R}^l,$$

for the original form of P with l equality constraints in (10). The term augmented Lagrangian refers to the fact that f_μ is the Lagrangian function for the problem P augmented with a penalty term:

$$(P_\mu) : \text{min. } \left\{ f(x) + \mu \sum_{i=1}^{l} h_i(x)^2 : h_i(x) = 0,\ i = 1, \ldots, l,\ x \in X \right\}.$$

Observe that this problem is equivalent to P since the penalty term in P_μ has no effect on the value of the target function for any feasible x.

The augmented Lagrangian has many important properties. For example, if $(\overline{x}, \overline{w})$ is a KKT solution for the original P, then the gradient of the augmented Lagrangian with respect to x vanishes at $(\overline{x}, \overline{w})$:

$$\nabla_x L_\mu(\overline{x}, \overline{w}) = \nabla f(\overline{x}) + \sum_{i=1}^{l} \overline{w}_i \nabla h_i(\overline{x}) + 2\mu \sum_{i=1}^{l} h_i(\overline{x}) \nabla h_i(\overline{x}) = \nabla f(\overline{x}) + \sum_{i=1}^{l} \overline{w}_i \nabla h_i(\overline{x}) = 0$$

since $h(\overline{x}) = 0$ due to the primal feasibility included in the KKT conditions. This implies that (under appropriate assumptions on convexity) the optimal solution $\overline{x}$ can be recovered by using a finite penalty term μ, unlike with the original penalty-based method.

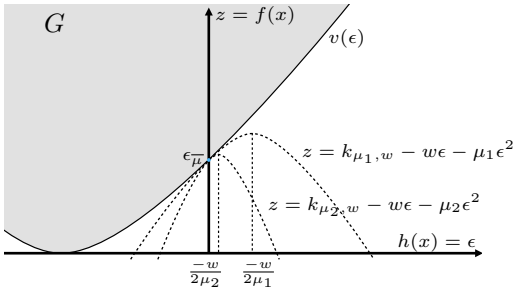


Figure 3: Geometric representation of augmented Lagrangians in the coordinates $(z, \epsilon) = (h(x), f(x))$.

2.1 Augmented Lagrangian method of multipliers

We can employ an unconstrained optimisation method to solve the augmented Lagrangian function[1]

$$L_\mu(x, v) = f(x) + \sum_{i=1}^{l} v_i h_i(x) + \mu \sum_{i=1}^{l} h_i(x)^2,$$

which relies on strong duality and corresponds to a search for KKT points (or primal-dual pairs) $(\overline{x}, \overline{v})$ by iteratively operating in the primal (x) and dual (v) spaces. In what follows, we denote the unaugmented Lagrangian by $L(x, v) = L_0(x, v)$.

In particular, the strategy consists of

1. *Primal space:* minimise $L_\mu(x, v^k)$ using a suitable unconstrained optimisation method to determine x^{k+1}.
2. *Dual space:* perform a dual variable update, moving to a (potential) ascent direction of $L_\mu(x^{k+1}, v)$ at v^k so that the primal optimality condition $\nabla_x L(x^{k+1}, v^{k+1}) = 0$ is satisfied for the unaugmented Lagrangian.

This strategy is akin to applying the subgradient method to solving the augmented Lagrangian dual: the update step for the dual variable is given by

$$v^{k+1} = v^k + 2\mu h(x^{k+1}).$$

The motivation for the dual update stems from the following observations:

1. $h(x^{k+1})$ is a subgradient (i.e., a supergradient) for $\theta(v) = \inf_x\{L_\mu(x, v)\}$ at v^k since x^{k+1} minimises $L_\mu(x, v^k)$ with respect to x; cf. Theorem 9 of Lecture 8.

[1] "Solving" means minimising with respect to the primal variable x and (then) maximising with respect to the dual variable v. Note also that v no longer refers to the function illustrated in Figures 2 and 3, but it now plays the role of a dual variable corresponding to the equality constraints.

2. The step size is devised such that the primal optimality condition for the original Lagrangian function holds at the new iterate: $\nabla_x L(x^{k+1}, v^{k+1}) = 0$.

Part 2 refers to the following:

$$\begin{aligned}\nabla_x L(x^{k+1}, v^{k+1}) &= \nabla f(x^{k+1}) + \sum_{i=1}^{l} v_i^{k+1} \nabla h_i(x^{k+1}) \\ &= \nabla f(x^{k+1}) + \sum_{i=1}^{l} (v_i^k + 2\mu h_i(x^{k+1})) \nabla h_i(x^{k+1}) \\ &= \nabla f(x^{k+1}) + \sum_{i=1}^{l} v_i^k \nabla h_i(x^{k+1}) + \sum_{i=1}^{l} 2\mu h_i(x^{k+1}) \nabla h_i(x^{k+1}) \\ &= \nabla_x L_\mu(x^{k+1}, v^k) = 0,\end{aligned}$$

where the final conclusion is due to x^{k+1} being a minimiser of $L_\mu(x, v^k)$ with respect to x. That is, by employing the dual update $v^{k+1} = v^k + 2\mu h(x^{k+1})$, the optimality for the unaugmented Lagrangian function follows from the optimality condition of the augmented Lagrangian at the previous iterate for the dual variable.

Algorithm 1 summarises the augmented Lagrangian method of multipliers (ALMM).

Algorithm 1 Augmented Lagrangian method of multipliers (ALMM)

1: **initialise.** tolerance $\epsilon > 0$, initial dual solution v^0, iteration count $k = 0$
2: **while** $\|h(x^k)\| > \epsilon$ **do**
3: $\quad x^{k+1} = \arg\min L_\mu(x, v^k)$
4: $\quad v^{k+1} = v^k + 2\mu h(x^{k+1})$
5: $\quad k = k + 1$
6: **end while**
7: **return** x^k.

The method can be specialised such that μ is individualised for each constraint and updated proportionally to the observed infeasibility $h_i(x)$. Another important point about the augmented Lagrangian method of multipliers is that linear convergence is to be expected, due to the gradient-like step taken to find optimal dual variables. This is often the case with traditional Lagrangian duality based approaches.

2.2 Alternating direction method of multipliers – ADMM

ADMM is a distributed version of the augmented Lagrangian method of multipliers, and it is more suited to large problems with a decomposable structure.

Assume the optimisation problem P is of the following form:

$$\begin{aligned}(P): \text{min.} \quad & f(x) + g(y) \\ \text{subject to:} \quad & Ax + By = c.\end{aligned}$$

We would like to be able to solve the problem separately for x and y, which could, in principle, be achieved using ALMM. However, the inclusion of the penalty term prevents the problem from being completely separable. To see that, let

$$\phi_\mu(x, y, v) = f(x) + g(y) + v^\top(c - Ax - By) + \mu(c - Ax - By)^2$$

be the augmented Lagrangian function. The penalty term $\mu(c - Ax - By)^2$ prevents the separation of the problem in terms of the x and y variables. However, separability can be recovered if one employs a coordinate descent approach in which three blocks of variables are considered separately: x, y, and v. The ADMM is summarised in Algorithm 2.

Algorithm 2 ADMM

1: **initialise.** tolerance $\epsilon > 0$, initial dual and primal solutions v^0 and y^0, $k = 0$
2: **while** $\|c - Ax^k - By^k\|$ and $\|y^{k+1} - y^k\| > \epsilon$ **do**
3: $\quad x^{k+1} = \arg\min \phi_\mu(x, y^k, v^k)$
4: $\quad y^{k+1} = \arg\min \phi_\mu(x^{k+1}, y, v^k)$
5: $\quad v^{k+1} = v^k + 2\mu(c - Ax^{k+1} - By^{k+1})$
6: $\quad k = k + 1$
7: **end while**
8: **return** (x^k, y^k).

One important feature regarding ADMM is that the coordinate descent steps are taken in a cyclic order, not requiring more than one (x, y) update step. Variants consider more than one of these steps, but no clear benefit in practice has been observed. Moreover, μ can be updated according to the amount of infeasibility observed at iteration k, but no generally good update rule is known.

ADMM is particularly relevant as a method for (un)constrained problems in which it might expose a structure that can be exploited, such as being able to give the solutions of some of the subproblems (in Lines 3 and 4 in Algorithm 2) in closed forms.

Lecture 10 - Barrier methods

Fabricio Oliveira (with modifications by Nuutti Hyvönen)

Last update: November 18, 2022

Abstract. In this lecture, we consider a class of methods known as the barrier methods. We describe the behaviour of such methods in terms of finding optimal solutions for constrained optimisation problems, and provide a general convergence result. Then, we describe the most widespread barrier method, generally known as the interior point method for solving LPs. We develop the main settings of the algorithm and discuss some implementation aspects.

Outline of this lecture

1 Barrier functions

In essence, barrier methods also use proxies for the constraints in the objective function, so that an unconstrained optimisation problem can be solved instead. However, the concept of barrier functions is different from penalty functions: the former are defined to *prevent* the solution search method from leaving the feasible region, which is why some of these methods are also called *interior point methods*.

Consider the primal problem P being defined as

$$\begin{aligned} (P): \quad & \text{min.} \quad f(x) \\ & \text{subject to:} \quad g(x) \leq 0, \\ & \qquad\qquad\quad x \in X. \end{aligned}$$

We define the *barrier problem* BP as

$$\begin{aligned} (BP): \quad & \inf_{\mu} \ \theta(\mu) \\ & \text{subject to:} \quad \mu > 0 \end{aligned}$$

where $\theta(\mu) = \inf_x \{f(x) + \mu B(x) : g(x) < 0,\ x \in X\}$ and $B(x)$ is a *barrier function*. The barrier function is such that its value approaches $+\infty$ as the boundary of the region $\{x : g(x) \leq 0\}$ is approached from its interior. Notice that, in practice, this means that the constraint $g(x) < 0$ can be dropped, as it is automatically enforced by the barrier function.

The barrier function $B : \mathbb{R}^m \to \mathbb{R}$ is of the form

$$B(x) = \sum_{i=1}^{m} \phi(g_i(x))$$

where

$$\begin{cases} \phi(y) \geq 0 & \text{if } y < 0; \\ \phi(y) \to \infty & \text{when } y \to 0^-. \end{cases} \tag{1}$$

Examples of barrier functions that exhibit this behaviour are

- $B(x) = -\sum_{i=1}^{m} \frac{1}{g_i(x)}$
- $B(x) = -\sum_{i=1}^{m} \ln(\min\{1, -g_i(x)\})$

Perhaps the most important barrier function is the *Frisch's log barrier function*, used in the highly successful primal-dual interior point methods. We will describe its use later. The log barrier is

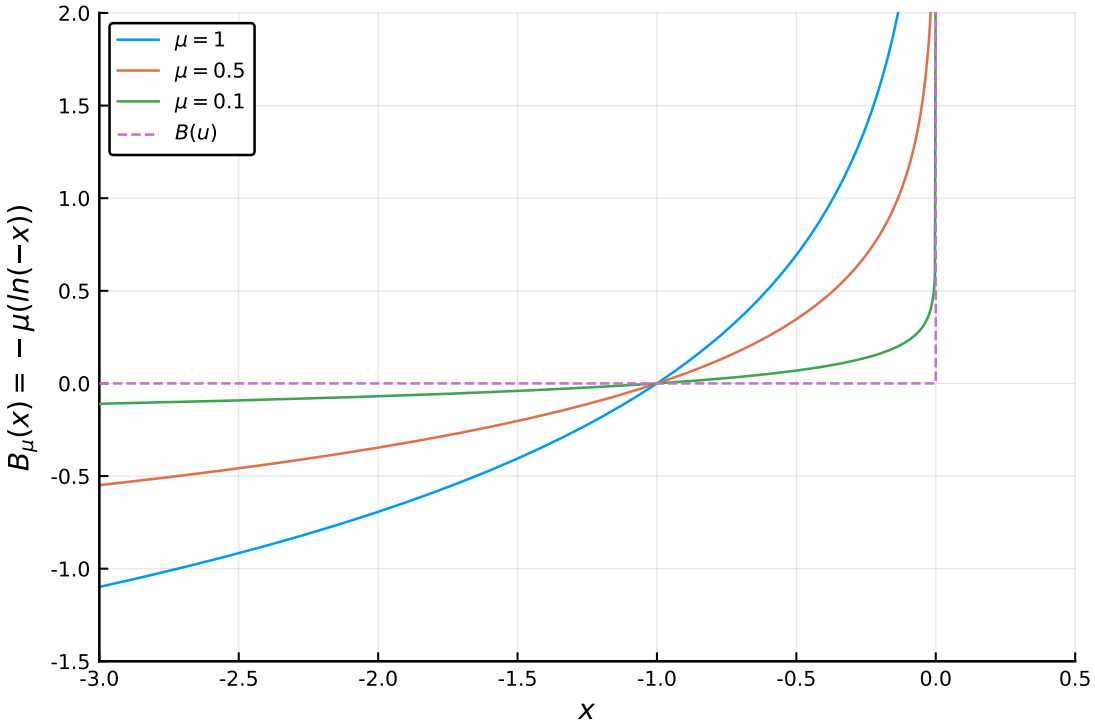


Figure 1: The product of μ and the log barrier function for different values of μ

defined as

$$B(x) = -\sum_{i=1}^{m} \ln(-g_i(x));$$

note that the log barrier is not strictly speaking a barrier function since it does not satisfy the positivity condition in (1). Figure 1 illustrates the behaviour of the log barrier function. Ideally, the log barrier function $B(x)$ has the role of an *indicator function*, which is zero for all feasible solutions $x \in \{x : g(x) < 0\}$ but assumes an infinite value if a solution is at the boundary $g(x) = 0$ or outside the feasible region. This is illustrated by the dashed line in Figure 1. The barrier functions for different values of barrier term μ illustrate how the log barrier mimics this behaviour, becoming more and more pronounced as μ decreases.

2 The barrier method

In a similar nature to what was developed for penalty methods, we can devise a solution method that successively evaluates $\theta(\mu)$ for decreasing values of the barrier term μ. We start by stating the result that guarantees convergence of the barrier method.

Theorem 1 (Convergence of barrier methods)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}$ be continuous functions and $X \subset \mathbb{R}^n$. Suppose $\{x \in X : g(x) < 0\}$ is nonempty. Let $\overline{x}$ be an optimal solution for P such that, for any neighbourhood $N_\delta(\overline{x}) = \{x : \|x - \overline{x}\| < \delta\}$ with $\delta > 0$, there exists $x \in X \cap N_\delta(\overline{x})$ for which $g(x) < 0$. Then,*

$$f(\overline{x}) = \min\{f(x) : g(x) \leq 0,\ x \in X\} = \lim_{\mu \to 0^+} \theta(\mu) = \inf_{\mu > 0} \theta(\mu),$$

where $\theta(\mu) = \inf_x \{f(x) + \mu B(x) : g(x) < 0,\ x \in X\}$ for a barrier function B satisfying (1)*.*

Proof. For $\mu \geq \lambda > 0$ and $x \in X$ such that $g(x) < 0$, we have

$$f(x) + \mu B(x) \geq f(x) + \lambda B(x)$$

due to the positivity condition in (1). Taking the infimum gives

$$\theta(\mu) = \inf_x \{f(x) + \mu B(x) : g(x) < 0,\ x \in X\} \geq \inf_x \{f(x) + \lambda B(x) : g(x) < 0,\ x \in X\} = \theta(\lambda),$$

which means that θ is nondecreasing. We thus conclude that $\lim_{\mu \to 0^+} \theta(\mu) = \inf\{\theta(\mu) : \mu > 0\}$.

Let $\epsilon > 0$. As f is continuous, by assumption there exists $x_\epsilon \in X$ with $g(x_\epsilon) < 0$ such that $f(\overline{x}) + \epsilon > f(x_\epsilon)$ for any $\epsilon > 0$. In consequence, it holds that

$$f(\overline{x}) + \epsilon + \mu B(x_\epsilon) > f(x_\epsilon) + \mu B(x_\epsilon) \geq \theta(\mu)$$

for any for $\mu > 0$. Letting $\mu \to 0^+$, it follows that $f(\overline{x}) + \epsilon \geq \lim_{\mu \to 0^+} \theta(\mu)$, and as this holds for any $\epsilon > 0$,

$$f(\overline{x}) \geq \lim_{\mu \to 0^+} \theta(\mu) = \inf\{\theta(\mu) : \mu > 0\}. \tag{2}$$

Conversely, since $B(x) \geq 0$ for such $x \in X$ that $g(x) < 0$,

$$\begin{aligned} \theta(\mu) &= \inf\{f(x) + \mu B(x) : g(x) < 0,\ x \in X\} \\ &\geq \inf\{f(x) : g(x) \leq 0,\ x \in X\} = f(\overline{x}) \end{aligned}$$

for any $\mu > 0$. Thus $f(\overline{x}) \leq \lim_{\mu \to 0^+} \theta(\mu) = \inf\{\theta(\mu) : \mu > 0\}$. Combining this with (2),

$$f(\overline{x}) = \lim_{\mu \to 0^+} \theta(\mu) = \inf\{\theta(\mu) : \mu > 0\},$$

which completes the proof. □

The proof has three main steps. First, we show that $\theta(\mu)$ is a nondecreasing function in μ, which implies that $\lim_{\mu \to 0^+} \theta(\mu) = \inf\{\theta(\mu) : \mu > 0\}$. This can be trivially shown as only feasible solutions x need to be considered. The other two steps prove the convergence of the barrier method by showing

that $\inf_{\mu>0} \theta(\mu) = f(\overline{x})$, where $\overline{x} = \arg\min \{f(x) : g(x) \leq 0,\ x \in X\}$ is assumed to exist.

In the second step, the continuity of f and the assumption on 'nonempty neighbourhoods' of $\overline{x}$ implies that for any $\epsilon > 0$, there exists x_ϵ such that $g(x_\epsilon) < 0$ and $f(x_\epsilon) - f(\overline{x}) < \epsilon$. From this, it straightforwardly follows that $f(\overline{x}) \geq \inf_{\mu>0} \theta(\mu)$. In the last step, we use the argument that including the boundary in the consideration can only improve the objective function value, leading to the converse inequality $f(\overline{x}) \leq \inf_{\mu>0} \theta(\mu)$.

It is worth highlighting that, to simplify the proof, we have assumed that the barrier function has the form described in (1), being, in particular, nonnegative. However, a proof in the veins of Theorem 1 can be still be developed for the Frisch log barrier (for which $B(x)$ is not necessarily nonnegative) since, essentially, (1) only needs to be investigated in a neighbourhood of points x satisfying $g(x) = 0$.

Corollary 2 (Convergence of barrier methods)**.** *Suppose the assumptions of Theorem 1 hold and X is closed. Furthermore, assume that there exists $x_\mu = \arg\min_x \{f(x) + \mu B(x) : g(x) < 0,\ x \in X\}$ for every $\mu > 0$ and that the set $\{x_\mu\}_{\mu \in \mathbb{R}_+}$ is bounded. Then any sequence $\{x_{\mu_k}\}_{k=1}^\infty$, with $\mathbb{R}_+ \ni \mu_k \to 0$ as $k \to \infty$, has a subsequence converging to a solution of P. Moreover, the limit of $\mu_k B(x_{\mu_k})$ for such a subsequence is 0.*

Proof. The existence of a infinite converging subsequence for $\{x_{\mu_k}\}_{k=1}^\infty$ follows from the fact that any bounded sequence in $\mathbb{R}^n$ has such a subseuqence. We abuse the notation by still denoting this subsequence by $\{x_{\mu_k}\}_{k=1}^\infty$ and its limit by $\hat{x}$. Our aim is to show that $\hat{x}$ is a solution of P.

First of all, $\hat{x} \in X$ as X is closed, and

$$g(\hat{x}) = \lim_{k\to\infty} g(x_{\mu_k}) \leq 0$$

since g is assumed to be continuous. Hence, $\hat{x}$ is feasible for P. Secondly,

$$\lim_{k\to\infty} \theta(\mu_k) = \lim_{k\to\infty} \big(f(x_{\mu_k}) + \mu_k B(x_{\mu_k})\big) \geq \lim_{k\to\infty} f(x_{\mu_k}) = f(\hat{x})$$

due to the nonnegativity of B and the continuity of f. By virtue of Theorem 1, $\hat{x}$ is thus a solution of P and $\mu_k B(x_{\mu_k}) \to 0$ as $k \to \infty$. □

The results in Theorem 1 and Corollary 2 allow to design optimisation methods that, starting from a strictly feasible (interior) solution, are based on successively reducing the barrier term until a solution with arbitrarily small barrier term is obtained. Algorithm 1 presents a pseudocode for such a method.

Algorithm 1 Barrier method

1: **initialise.** $\epsilon > 0$, $x^0 \in X$ with $g(x^k) < 0$, μ_0, $\beta \in (0,1)$, $k = 0$.
2: **while** $\mu_k B(x^k) > \epsilon$ **do**
3: $\quad x^{k+1} = \arg\min \{f(x) + \mu_k B(x) : x \in X\}$
4: $\quad \mu_{k+1} = \beta \mu_k$
5: $\quad k = k + 1$
6: **end while**
7: **return** x^k.

One important aspect to notice is that starting the algorithm requires a strictly feasible point, which in some applications, might be challenging to be obtained. This characteristic is what renders the name *interior point methods* for this class of algorithms.

Consider the following example. Let $P = \left\{(x+1)^2 : x \geq 0\right\}$ and use the barrier function $B(x) = -\ln(x)$. Then, the unconstrained barrier problem becomes

$$(BP): \min_x f_\mu(x),$$
$$\text{subject to: } x > 0,$$

where $f_\mu(x) = (x+1)^2 - \mu \ln(x)$. Since f_μ is convex as a sum of two convex functions and its limit is obviously ∞ at both 0 and ∞, the first order condition $f'(x) = 0$ is necessary and sufficient for optimality. Thus, solving $2(x+1) - \frac{\mu}{x} = 0$, we obtain as the positive solution of a quadratic equation the unique solution $x_\mu = \frac{-1}{2} + \frac{\sqrt{4+8\mu}}{4}$ for BP. Figure 2 shows the behaviour of the problem as μ converges to zero. As can be seen, as $\mu \to 0^+$, the optimal solution x_μ converges to the constrained optimum $\overline{x} = 0$.

Let us also consider a more involved example:

$$P: \text{ min. } \left\{(x_1 - 2)^4 + (x_1 - 2x_2)^2 : x_1^2 - x_2 \leq 0\right\}$$

with the barrier function $B(x) = -\frac{1}{x_1^2 - x_2}$. We implemented Algorithm 1 and solved it with two distinct values for the penalty term μ and reduction term β. Figure 3 illustrate the trajectory of the algorithm with each parametrisation, exemplifying how these can affect the convergence of the method.

3 Interior point method for LP/QP problems

Perhaps ironically, the most successful applications of barrier methods in terms of efficient implementations are devoted to solving linear and quadratic programming (LP/QP) problems. The primal-dual interior point method has become in the last decade the algorithm of choice for many applications

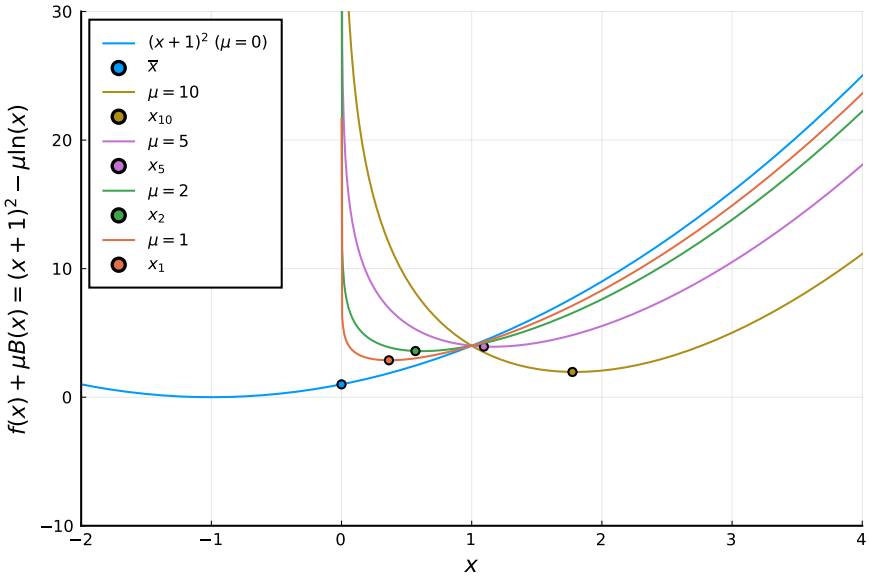


Figure 2: Example 1: solving a one-dimensional problem with the barrier method.

involving large-scale LP/QP problems.

To see how barrier methods can be applied to LP problems, consider the following primal/dual pair formed by a LP primal P

$$\begin{aligned} (P) \ : \ \text{min.} \ & c^\top x \\ \text{subject to:} \ & Ax = b, \\ & x \geq 0, \end{aligned}$$

and its respective dual problem

$$\begin{aligned} (D) \ : \ \text{max.} \ & b^\top v \\ \text{subject to:} \ & A^\top v \leq c, \end{aligned}$$

which can equivalently be written as

$$\begin{aligned} (D) \ : \ \text{max.} \ & b^\top v \\ \text{subject to:} \ & A^\top v + u = c, \\ & u \geq 0, \end{aligned}$$

with the help of a slack variable u. Moreover, the KKT conditions for P (or D for that matter) can

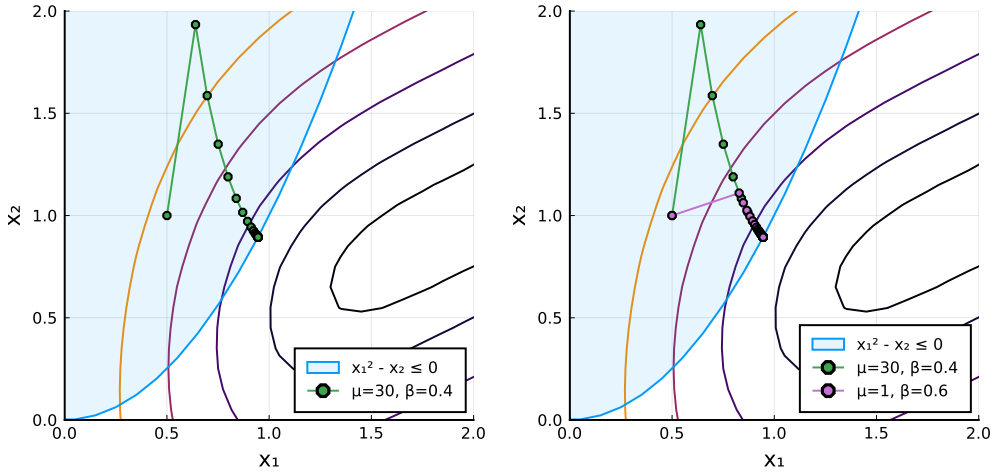


Figure 3: The trajectory of the barrier method for problem P. Notice how the parameters influence the trajectory and number of iterations. The parameters on the left require 27 iterations while those on the right require 40 iterations for convergence.

be written as

$$\begin{aligned} &A^\top v + u = c, \quad u \geq 0, && \text{(dual feasibility)}, \\ &Ax = b, \quad x \geq 0, && \text{(primal feasibility)} \\ &u^\top x = 0 && \text{(complementary slackness)} \end{aligned}$$

by choosing the "sign" of the unresricted Lagrange multiplier v appropriately. Note that $u^\top x = 0$ is equivalent to $u_i x_i = 0$ for $i = 1, \ldots, n$, due to the inequality constraints on x and u. These are going to be useful as a reference for the following developments.

We start by considering the *barrier problem* for P by using the logarithmic barrier function to represent the condition $x \geq 0$:

$$\begin{aligned} (BP) \ : \ \text{min.} \ & c^\top x - \mu \sum_{i=1}^{n} \ln(x_j) \\ \text{subject to:} \ & Ax = b, \end{aligned}$$

for $\mu > 0$ and $x > 0$. Notice that this problem is a strictly convex equality constrained problem that is suitable to be solved using the constrained variant of Newton's method (which simply consists of employing Newton's method to solve the KKT conditions of equality constrained problems). The

KKT conditions of BP are

$$\begin{aligned} &Ax = b, \quad x > 0, && \text{(primal feasibility)} \\ &A^\top v = c - \mu \left[\frac{1}{x_1}, \ldots, \frac{1}{x_n}\right]^\top. && \text{(dual feasibility)} \end{aligned}$$

Comparing these to the KKT conditions of P, the vector $\mu \left[\frac{1}{x_1}, \ldots, \frac{1}{x_n}\right]^\top > 0$ can be interpreted as an estimate for the Lagrangian dual variable $u \geq 0$.

To further understand the relationship between the KKT conditions of BP and P, let us define $X \in \mathbb{R}^{n \times n}$ and $U \in \mathbb{R}^{n \times n}$ as

$$X = \mathbf{diag}(x) = \begin{bmatrix} \ddots & & \\ & x_i & \\ & & \ddots \end{bmatrix} \quad \text{and} \quad U = \mathbf{diag}(u) = \begin{bmatrix} \ddots & & \\ & u_i & \\ & & \ddots \end{bmatrix},$$

and let $e = [1, \ldots, 1]^\top \in \mathbb{R}^n$. This allows us to rewrite the KKT conditions of BP as

$$Ax = b, \quad x > 0 \tag{3}$$

$$A^\top v + u = c \tag{4}$$

$$u = \mu X^{-1} e \iff XUe = \mu e. \tag{5}$$

Notice how the condition (5) resembles the complementary slackness condition for P: instead for requiring that

$$\mathbf{trace}(XU) = \sum_{i=1}^{n} u_i x_i = u^\top x = 0,$$

the condition is *relaxed* so that each diagonal element of XU, i.e., each product $u_i x_i$ for $i = 1, \ldots, n$, is required to equal μ. This is why this system is often referred to as the *perturbed KKT system*.

Let us denote the solution of (3)–(5) by $w_\mu = (x_\mu, v_\mu, u_\mu)$ — assuming such a solution exists. Theorem 1[1] guarantees that $w_\mu = (x_\mu, v_\mu, u_\mu)$ approaches the optimal primal-dual solution of P as $\mu \to 0^+$ (assuming the latter exists); recall that the KKT conditions are equivalent to the original minimization problem for a convex objetive function under Slater's CQ. The trajectory formed by successive solutions $\{w_\mu\}$ is called the *central path*, which is due to the interiority for x_μ enforced by the barrier function.

The term $u_\mu^\top x_\mu$ measures the duality gap for the solution of (3)–(5) at a given $\mu > 0$. To see this,

[1] In fact, we require a slight variant of Theorem 1 that allows for $B(x) \geq 0$ only being required in a neighbourhood of $g(x) = 0$.

notice first that

$$\begin{aligned} c^\top x_\mu &= (A^\top v_\mu + u_\mu)^\top x_\mu \\ &= (A^\top v_\mu)^\top x_\mu + u_\mu^\top x_\mu \\ &= v_\mu^\top (A x_\mu) + u_\mu^\top x_\mu. \end{aligned}$$

As $Ax_\mu = b$, we thus have

$$c^\top x_\mu - b^\top v_\mu = u_\mu^\top x_\mu = \sum_{i=1}^{n} (u_\mu)_i (x_\mu)_i = \sum_{i=1}^{n} \left(\frac{\mu}{(x_\mu)_i} \right) (x_\mu)_i = n\mu,$$

which gives the *total slack violation* that can be used to determine the convergence of the algorithm.

3.1 Primal/dual path-following interior point method

The primal/dual path following interior point method (IPM) is a specialised version of the setting described above for solving LP/QP problems. It consists in applying logarithmic barriers to LP/QP problems and solving the system (3)–(5) using Newton's method. However, instead of solving the problem accurately for each μ, only a *single* Newton step is taken before the barrier term μ is reduced.

Suppose that we start with a penalty term $\mu_k > 0$ and $w^k = (x^k, v^k, u^k)$ 'sufficiently close' to the corresponding solution w_{μ_k} of (3)–(5). The underlying idea is to choose a suitable $\beta \in (0, 1)$ and form a new iterate w^{k+1} that is 'sufficiently close' to the solution $w_{\mu_{k+1}}$ of (3)–(5) corresponding to the reduced penalty term $\mu_{k+1} = \beta\mu_k$. Figure 4 illustrates this procedure, showing how suboptimal solutions x^k (or w^k) do not necessarily need to lie in the central path (denoted by the dashed line) to guarantee convergence, as long as they remain within a suitable neighbourhood

$$N_{\mu_k}(\theta) = \left\{ w : \|XUe - \mu_k e\|_2 \leq \theta\mu_k \right\} = \left\{ w : \sum_{i=1}^{n} \left(x_i u_i - \mu_k \right)^2 \leq \theta^2 \mu_k^2 \right\}$$

of the central path. Observe that the centrality is here measured by the Euclidean norm of the discrepancy in (5), but other norms could also be employed. As an example, one may choose $\mu_0 = \frac{(x^0)^\top u^0}{n}$, $\beta = 1 - \frac{\sigma}{\sqrt{n}}$ and $\theta = \sigma = 0.1$ to guarantee that the successive Newton steps remain within $N_{\mu_k}(\theta)$.

To see how the procedure works in detail, denote the perturbed KKT system (3)–(5) for an arbitrary $\mu > 0$ as $H_\mu(w) = 0$, and let $J_\mu(w)$ be the Jacobian matrix of H_μ at w. Employing linearisation at w^k to approximately solve $H_{\mu_{k+1}}(w^{k+1}) = 0$, we obtain

$$J_{\mu_{k+1}}(w^k) d_w^{k+1} = -H_{\mu_{k+1}}(w^k), \tag{6}$$

where $d_w^{k+1} = w^{k+1} - w^k$. By rewriting $d_w^{k+1} = (d_x^{k+1}, d_v^{k+1}, d_u^{k+1})$, the linear system of equations (6)

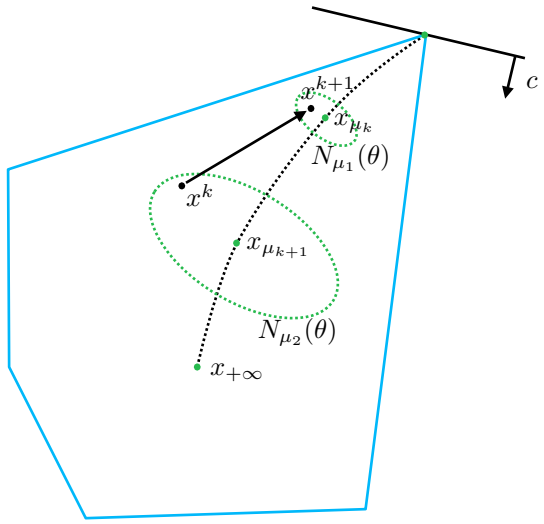


Figure 4: An illustrative representation of the central path and how the (x component in the) IPM follows it approximately.

can be equivalently written as

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^\top & I \\ U^k & 0 & X^k \end{bmatrix} \begin{bmatrix} d_x^{k+1} \\ d_v^{k+1} \\ d_u^{k+1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \mu_{k+1}e - X^k U^k e \end{bmatrix}, \tag{7}$$

if it is assumed that the previous iterate w^k is primal and dual feasible, that is, it satisfies (3)–(4). The system (7) is often referred to as the Newton's system.

In practice, the iterate w^k may not be primal and dual feasible, in which case (6) is of the form

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^\top & I \\ U^k & 0 & X^k \end{bmatrix} \begin{bmatrix} d_x^{k+1} \\ d_v^{k+1} \\ d_u^{k+1} \end{bmatrix} = - \begin{bmatrix} Ax^k - b \\ A^\top v^k + u^k - c \\ X^k U^k e - \mu_{k+1} e \end{bmatrix}. \tag{8}$$

To see how the update rule (8) still eventually leads to primal and dual feasible solutions, consider the primal residual $r_p(x,u,v) = Ax - b$ and the dual residual $r_d(x,u,v) = A^\top v + u - c$, measuring the extent to which (3) and (4), respectively, are not satisfied. Let $r(w) = r(x,u,v) = (r_p(x,u,v), r_d(x,u,v))$ be the total residual, where we again use the notation $w = (x,v,u)$. The first two conditions in (3)–(5) can be expressed as $r(w) = 0$.

Consider the linearisation of r around w^k,

$$r(w^k + d_w) \approx r(w^k) + Dr(w^k)d_w,$$

where $Dr(w^k)$ denotes the Jacobian matrix of r evaluated at w^k. Observe that

$$Dr(w^k) = \begin{bmatrix} A & 0 & 0 \\ 0 & A^\top & I \end{bmatrix}$$

i.e., it is defined by the first two rows of the Newton system (7) and is independent of w^k. A step d_w that makes the residual vanish in the linearisation satisfies

$$Dr(w^k)d_w = -r(w^k), \tag{9}$$

which coincides with the first two rows of (8). In particular, a solution d_w^{k+1} of (8) satisfies (9). If we thus consider the directional derivative of the square of the norm of r at w^k in the direction d_w^{k+1} that solves (8), we obtain

$$\frac{d}{dt}\|r(w^k + td_w^{k+1})\|_2^2\Big|_{t=0} = 2r(w^k)^\top Dr(w^k)d_w^{k+1} = -2r(w^k)^\top r(w^k) = -2\|r(w^k)\|^2 \tag{10}$$

which is negative unless the residual at w^{k+1} vanishes. That is, the solution d_w^{k+1} of (8) is a descent direction for the norm of the residual r at w^k, and thus it does not seem unreasonable to expect that $r(w^k)$ converges to zero in the iteration defined by (8).

The algorithm proceeds by iteratively solving the system (8) with $\mu_{k+1} = \beta\mu_k$ with $\beta \in (0, 1)$ until $n\mu_k$ is less than a specified tolerance. Algorithm 2 summarises a simplified form of the IPM.

Algorithm 2 Interior point method (IPM) for LP

1: **initialise.** (primal-dual feasible) w^0, $\epsilon > 0$, $\mu_0 = \mu_1 > 0$, $\beta \in (0, 1)$, $k = 0$.
2: **while** $n\mu_k > \epsilon$ **do**
3: compute $d_w^{k+1} = (d_x^{k+1}, d_v^{k+1}, d_u^{k+1})$ using (8)
4: $w^{k+1} = w^k + d_w^{k+1}$
5: $k = k + 1$
6: $\mu_{k+1} = \beta\mu_k$
7: **end while**
8: **return** w^k.

Figure 5 illustrates the behaviour of the IPM when employed to solve the linear problem

$$\begin{aligned} \text{min. } & x_1 + x_2 \\ \text{subject to: } & 2x_1 + x_2 \geq 8, \\ & x_1 + 2x_2 \geq 10, \\ & x_1, x_2 \geq 0, \end{aligned}$$

considering two distinct initial penalties μ. Notice that instead of mere positivity constraints on the components of x, here more general inequality constraints are considered; although not analysed

above, the IPM algorithm can be generalized to such a setting. Notice how higher penalty values enforce a more central convergence for the method.

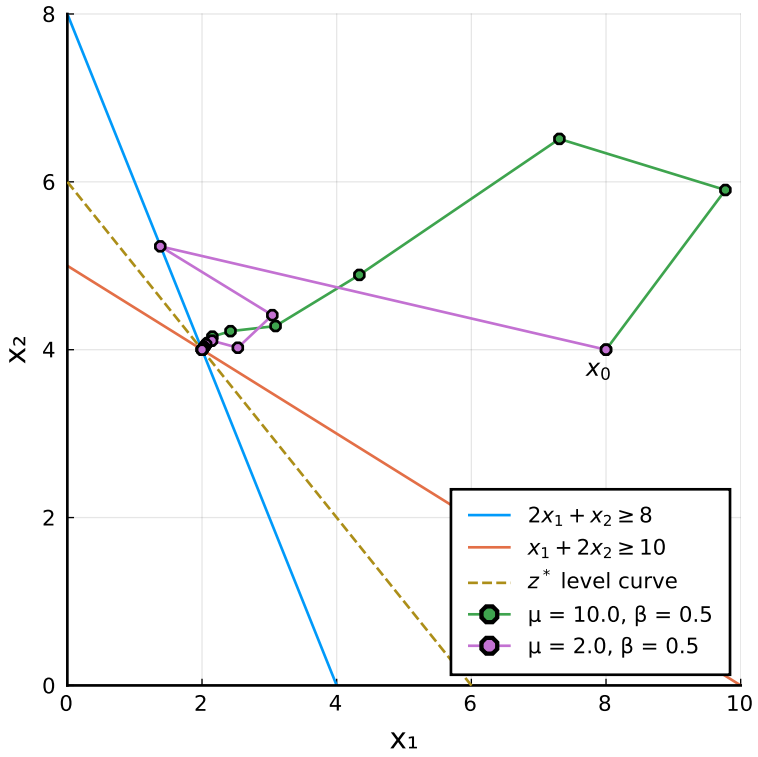


Figure 5: IPM applied to a LP problem with two different barrier terms.

Some points are worth emphasising concerning Algorithm 2. First, notice that in Line 4, a fixed step size is considered. A line search can be incorporated to prevent infeasibility and improve numerical stability. Typically, $\lambda_i^k = \min\left\{\alpha, -\frac{x_i^k}{d_i^k}\right\}$, with $\alpha < 1$ but close to 1, is used. Moreover, even though the algorithm is initialised with a feasible solution w^0, this may in practice be unnecessary. Implementations of the infeasible IPM method can efficiently handle primal and dual infeasibility.

Under specific conditions, the IPM can be shown to have complexity of $O(\sqrt{n}\ln(1/\epsilon))$, which is polynomial and of much better worst-case performance than the simplex method, which makes it the algorithm of choice for solving large-scale LPs. Another important advantage is that IPM can be modified with little effort to solve a wider class of problems under the class of *conic optimisation problems*.

Predictor-corrector methods are variants of IPM that incorporate a two-phase direction calculation using a *predicted* direction d_w^{pred}, calculated by setting $\mu = 0$ and a *correcting* direction, which is computed considering the impact that d_w^{cor} would have in the term $\overline{X}\,\overline{U}e$. To be more precise, let $\Delta X = \mathbf{diag}(d_x^{\text{pred}})$ and $\Delta U = \mathbf{diag}(d_u^{\text{pred}})$. Then

$$\begin{aligned}(X+\Delta X)(U+\Delta U)e &= XUe + (U\Delta X + X\Delta U)e + \Delta X\Delta U e \\ &= XUe + (0 - XUe) + \Delta X\Delta U e. \\ &= \Delta X\Delta U e. \end{aligned} \tag{11}$$

Using the last equation (11), the corrector Newton step becomes $\overline{U}d_x + \overline{X}d_u = \hat{\mu}e - \Delta X\Delta U e$. Finally, d_w^k is set to be a combination of d_w^{pred} and d_w^{cor}.

Lecture 11 - Feasible direction methods

Fabricio Oliveira (with modifications by Nuutti Hyvönen)

Last update: November 24, 2022

Abstract. In this lecture, we look into the class of feasible direction methods, also known as primal methods.

Outline of this lecture

1 The concept of feasible directions

Feasible direction methods are a class of methods that incorporate both improvement and feasibility requirements when devising search directions. As feasibility is also observed throughout the solution process, they are also referred to as *primal methods*. However, depending on the geometry of the feasible region, it might be so that the methods allow for some infeasibility in the course of the algorithm, as we will see later.

As a reminder, an *improving feasible direction* can be defined as follows:

Definition 1. *Consider the problem* $\min. \{f(x) : x \in S\}$ *with* $f : \mathbb{R}^n \to \mathbb{R}$ *and* $\emptyset \neq S \subseteq \mathbb{R}^n$. *A vector* $d \neq 0$ *is a feasible direction at* $x \in S$ *if there exists* $\delta > 0$ *such that* $x + \lambda d \in S$ *for all* $\lambda \in (0, \delta)$. *Moreover,* d *is an improving feasible direction at* $x \in S$ *if there exists* $\delta > 0$ *such that* $f(x + \lambda d) < f(x)$ *and* $x + \lambda d \in S$ *for* $\lambda \in (0, \delta)$.

The key feature of the feasible direction methods is the process of deriving such directions and associated step sizes that retain feasibility, even if only approximately. Similarly to the other methods we have discussed in the past lectures, these methods progress following two basic steps:

1. Obtain an *improving feasible direction* d^k and a step size λ^k;
2. Define $x^{k+1} = x^k + \lambda^k d^k$.

As a general rule, for a feasible direction method to perform satisfactorily, the computation of the directions d^k and step sizes λ^k must be simple enough. Often, these steps can be reduced to closed forms or, more frequently, to solving linear or quadratic programming problems, or to posing modified Newton systems.

2 Conditional gradient – the Frank–Wolfe method

The conditional gradient method is named as such due to the direction definition step, in which the next direction $d = x - x^k$, $x \in S$, is selected so that the inner product between the direction and the gradient at x^k is minimised; as usual, $x^k \in S$ denotes here the current iterate produced by the minimisation algorithm. Note that this process accounts for both the angle between the gradient and the next search direction as well as the length of the step one can take to the considered direction so that feasibility is retained.

Recall that, $x - x^k$, with $x \in S$, is a *descent direction* for a differentiable objective function f at x^k if

$$\nabla f(x^k)^\top (x - x^k) < 0.$$

A straightforward way of trying to obtain an improving feasible direction $d = x - x^k$ is by solving a *direction search problem* of the form

$$(DS): \quad \min_x \ \left\{\nabla f(x^k)^\top (x - x^k) : x \in S\right\}.$$

Problem DS consists of finding the point $x \in S$ such that the scalar projection of the proposed direction $x - x^k$ onto the direction defined by the gradient $\nabla f(x^k)$ is minimised, i.e., the scalar projection should be as negative as possible. This is precisely what gives the name *conditional gradient*. Observe that DS can be equivalently stated as

$$(DS): \quad \min_x \ \left\{\nabla f(x^k)^\top x : x \in S\right\}$$

since the constant term $-f(x^k)^\top x^k$ does not affect the solution of the considered minimisation problem.

The solvability of DS is guaranteed by, e.g., assuming that S is compact, i.e., closed and bounded. By setting $\overline{x}^k = \arg\min_{x \in S} \left\{\nabla f(x^k)^\top x\right\}$ and determining $\lambda^k \in (0, 1]$ via a line search, the update rule becomes

$$x^{k+1} = x^k + \lambda^k (\overline{x}^k - x^k).$$

Assuming that S is convex, the feasibility of x^{k+1} is guaranteed by constraining λ^k to lie in the interval $(0, 1]$; note that $\overline{x}^k$ is itself "borderline feasible" by definition.

In accordance with the above considerations, assume that $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable and S is convex and compact. Furthermore, assume for simplicity that f is convex. If the unconstrained optimum lies outside the feasible region S, the standard stopping condition $\nabla f(x^k) \approx 0$ may never be achieved. However, the sequence of estimates $\{x^k\}_{k=0}^\infty$ is guaranteed to converge and, furthermore, the sequence of directions $d^k = \overline{x}^k - x^k$ is guaranteed to converge to zero. Moreover, the point of convergence of the algorithm satisfies the first-order (constrained) optimality conditions. Therefore, the term $\nabla f(x^k)^\top d^k$ converges to zero regardless of whether the global minimiser of the objective function belongs to S or not, and hence monitoring this term is used as the stopping condition for the algorithm. Algorithm 1 summarises the Frank–Wolfe method.

Algorithm 1 Franke–Wolfe method

1: **initialise.** $\epsilon > 0$, $x^0 \in S$, $k = 0$.
2: **while** $|\nabla f(x^k)^\top d^k| > \epsilon$ **do**
3: $\quad \overline{x}^k = \arg\min \left\{\nabla f(x^k)^\top x : x \in S\right\}$
4: $\quad d^k = \overline{x}^k - x^k$
5: $\quad \lambda^k = \arg\min_\lambda \left\{f(x^k + \lambda d^k) : 0 \le \lambda \le 1\right\}$
6: $\quad x^{k+1} = x^k + \lambda^k d^k$
7: $\quad k = k + 1$
8: **end while**
9: **return** x^k

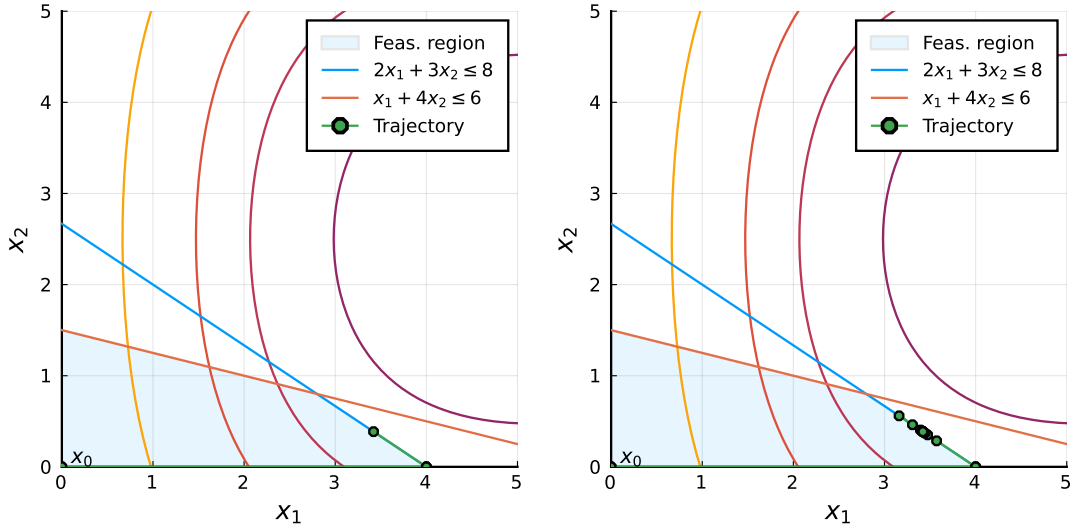


Figure 1: The Frank-Wolfe method applied to a problem with linear constraints. The algorithm takes two steps using an exact line search (left) and 15 with Armijo line search (right).

Notice that for a polyhedral feasibility set, the direction-defining subproblems are linear programming problems.

Figure 1 shows the employment of the FW method for optimising a nonlinear function within a polyhedral feasibility set. We consider the problem

$$\begin{aligned} \text{min. } & e^{-(x_1-3)/2} + e^{(4x_2+x_1-20)/10} + e^{(-4x_2+x_1)/10} \\ \text{subject to: } & 2x_1 + 3x_2 \leq 8, \\ & x_1 + 4x_2 \leq 6, \\ & x_1, x_2 \geq 0, \end{aligned}$$

starting from $(0, 0)$ and using an exact line search to set the step sizes $\lambda^k \in [0, 1]$. Notice that the method can be utilised with an inexact line search as well.

3 Sequential quadratic programming

Sequential quadratic programming (SQP) is a method inspired by the idea that the KKT system of a nonlinear problem can be solved by using Newton's method. It forms perhaps the most general family of methods in terms of considering both nonlinear constraints and a nonlinear objective function.

To understand how SQP works, let us first consider an equality constraint problem

$$P: \text{ min. } \{f(x) : h_i(x) = 0,\ i = 1, \ldots, l\}.$$

The KKT conditions for P are given by the system $W(x, v) = 0$, where $W : \mathbb{R}^{n+l} \to \mathbb{R}^{n+l}$ is defined by

$$W(x, v) = \begin{bmatrix} \nabla f(x) + \sum_{i=1}^{l} v_i \nabla h_i(x) \\ h(x) \end{bmatrix} = \begin{bmatrix} \nabla f(x) + J_h(x)^\top v \\ h(x) \end{bmatrix},$$

where $J_h : \mathbb{R}^n \to \mathbb{R}^{l \times n}$ denotes the Jacobian matrix of $h : \mathbb{R}^n \to \mathbb{R}^l$. One can (try to) solve for (x, v) in $W(x, v) = 0$ using the Newton–Raphson method.

To be more precise, starting from an initial guess (x^0, v^0), we employ successive linearizations of the form

$$W(x^k, v^k) + J_W(x^k, v^k) \begin{bmatrix} x - x^k \\ v - v^k \end{bmatrix} = 0 \tag{1}$$

to solve for (x^{k+1}, v^{k+1}), with $J_W : \mathbb{R}^{n+l} \to \mathbb{R}^{(n+l)\times(n+l)}$ denoting the Jacobian matrix of W. Upon a closer inspection, the Jacobian term in (1) is given by

$$J_W(x^k, v^k) = \begin{bmatrix} \nabla^2 L(x^k, v^k) & J_h(x^k)^\top \\ J_h(x^k) & 0 \end{bmatrix},$$

where

$$\nabla^2 L(x^k, v^k) = \nabla^2 f(x^k) + \sum_{i=1}^{l} v_i^k \nabla^2 h_i(x^k)$$

is the *Hessian of the Lagrangian* function

$$L(x, v) = f(x) + v^\top h(x)$$

at (x^k, v^k) with respect to x.

Setting $d = x - x^k$, we can thus rewrite (1) as

$$\nabla^2 L(x^k, v^k) d + J_h(x^k)^\top v = -\nabla f(x^k), \tag{2}$$

$$J_h(x^k) d = -h(x^k), \tag{3}$$

which can be repeatedly solved until

$$\left\| (x^k, v^k) - (x^{k-1}, v^{k-1}) \right\|_2 \leq \epsilon,$$

i.e., until convergence is observed. Assuming that the iteration converges to a solution of $H(x, v) = 0$, we can thus find an estimate of a KKT point for P.

This is the underlying idea of SQP, that is, one approximates the solution of the original optimisation problem by solving a sequence of approximating problems. However, instead of explicitly taking Newton steps for the original KKT system, we interpret our task as solving a sequence of approximating quadratic problems of the form

$$QP_k: \min_d \; f(x^k) + \nabla f(x^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k, v^k) d \tag{4}$$

$$\text{subject to: } h_i(x^k) + \nabla h_i(x^k)^\top d = 0, \; i = 1, \ldots, l, \tag{5}$$

where d denotes the sought-for additive update to x^k. Notice that QP_k is a linearly constrained quadratic programming problem, for which we have seen several solution approaches. Moreover, the KKT optimality conditions for QP_k are given by (2) and (3), with v being the dual variable associated with the equality constraints in (5), which, in turn, represent first-order approximations for the original constraints. The objective function in QP_k can be interpreted as a second-order approximation for $f(x)$ enhanced with the term $(1/2)\sum_{i=1}^{l} v_i^k d^\top \nabla^2 h_i(x^k) d$ that captures second-order information on the constraints at x^k.

An alternative interpretation for the objective function of QP is a second order approximation for the Lagrangian function $L(x, v) = f(x) + v^\top h(x)$ with respect to x around (x^k, v^k), assuming the validity of the constraints in (5):

$$\begin{aligned}
L(x, v^k) &\approx L(x^k, v^k) + \nabla_x L(x^k, v^k)^\top d + \frac{1}{2} d^\top \nabla_x^2 L(x^k, v^k) d \\
&= f(x^k) + (v^k)^\top h(x^k) + \Big(\nabla f(x^k) + \sum_{i=1}^{l} v_i^k \nabla h_i(x^k)\Big)^\top d + \frac{1}{2} d^\top \nabla_x^2 L(x^k, v^k) d \\
&= f(x^k) + \nabla f(x^k)^\top d + \sum_{i=1}^{l} v_i^k \big(\nabla h_i(x^k)^\top d + h_i(x^k)\big) + \frac{1}{2} d^\top \nabla_x^2 L(x^k, v^k) d \\
&= f(x^k) + \nabla f(x^k)^\top d + \frac{1}{2} d^\top \nabla_x^2 L(x^k, v^k) d.
\end{aligned}$$

Let us then consider a more general setting with inequality constraints defined by $g : \mathbb{R}^n \to \mathbb{R}^m$ included, which leads to the Lagrangian

$$L(x, u, v) = f(x) + u^\top g(x) + v^\top h(x), \qquad x \in \mathbb{R}^n, \; u \in \mathbb{R}^m, \; v \in \mathbb{R}^l.$$

In this case, a quadratic subproblem in the SQP method is (after dropping the constant term $f(x^k)$ from the objective function)

$$\begin{aligned}
QP_k: \; &\min. \; \nabla f(x^k)^\top d + \frac{1}{2} d^\top \nabla_x^2 L(x^k, u^k, v^k) d \\
&\text{subject to: } g_i(x^k) + \nabla g_i(x^k)^\top d \le 0, \; i = 1, \ldots, m, \\
&\qquad\qquad h_i(x^k) + \nabla h_i(x^k)^\top d = 0, \; i = 1, \ldots, l,
\end{aligned}$$

which includes the inequality constraints $g_i(x) \leq 0$ for $i = 1, \ldots, m$ in a linearised form. This generalisation is possible since we are resorting to an optimisation setting rather than a Newton system that only allows for equality constraints, even though the inequality constraints could also be handled in the Newton approach by introducing suitable slack variables. Clearly, there are several options that could be considered to solve this quadratic problem, including employing a primal/dual interior point method.

A pseudocode for the standard SQP method is presented in Algorithm 2.

Algorithm 2 SQP method

1: **initialise.** $\epsilon > 0$, $x^0 \in S$, $u^0 \geq 0$, v^0, $k = 0$.
2: **while** $\|d^k\| > \epsilon$ **do**
3: Solve d^k as well as the Lagrange multipliers u^{k+1} and v^{k+1} from QP_k.
4: $x^{k+1} = x^k + d^k$
5: $k = k + 1$
6: **end while**
7: **return** x^k.

Notice that on Line 3, one also needs to solve for the dual variables. Therefore, QP_k needs to be tackled by an algorithm that also considers dual variables.

Figure 2 illustrates the behaviour of the SQP method when applied to

$$\min_{x \in \mathbb{R}^2} \left\{ 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 : x_1^2 - x_2 \leq 0,\ x_1 + 5x_2 \leq 5,\ x_1 \geq 0,\ x_2 \geq 0 \right\}$$

Notice how the trajectory of approximate solutions may become infeasible since only linear approximations of the constraints are considered.

One important feature for the SQP method is that it closely mimics the convergence properties of Newton's method and therefore, under appropriate conditions, superlinear convergence can be observed. Moreover, a quasi-Newton method, e.g. BFGS, can be used to approximate $\nabla^2 L(x^k, u^k, v^k)$, which enables considering only first order derivatives of the objective and constraint functions.

Because the constraints are considered implicitly when solving the subproblem QP_k, one cannot devise a line search for the method, which in turn, being based on successive quadratic approximations, presents a risk for divergence. The l_1-SQP is a modern variant of SQP that addresses divergence issues arising in the SQP method when considering iterates that are far away from the optimum, while presenting superior computational performance.

In essence, l_1-SQP relies on a similar principle as penalty methods, encoding penalisation for infeasibility in the objective function of the quadratic subproblem. In the context of SQP algorithms, these functions are called *merit* functions. This not only allows for considering line search methods (since feasibility becomes encoded in the objective function with feasibility guaranteed at a minimum, cf. penalty methods) but also, alternatively, relying on a trust region approach, ultimately preventing

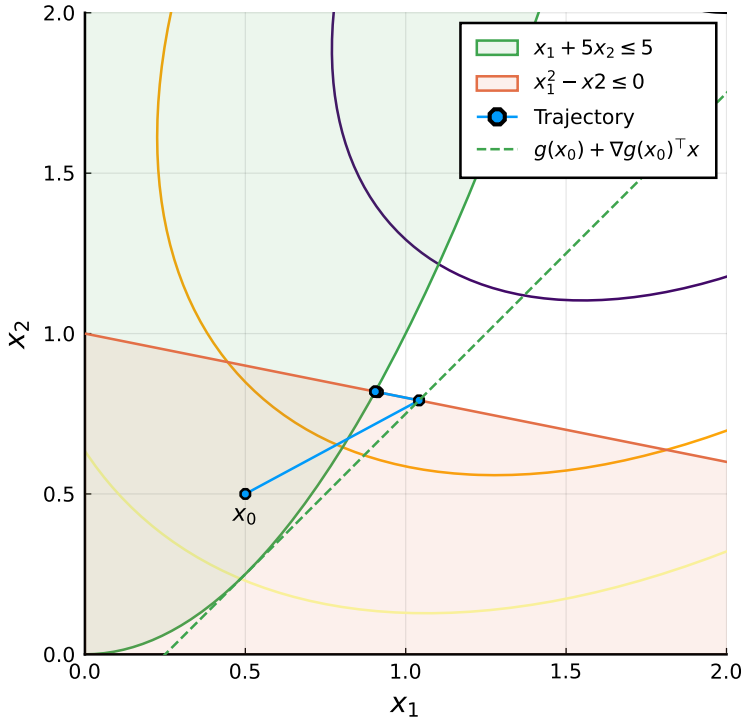


Figure 2: The SQP method converges in 6 iterations with $\epsilon = 10^{-6}$

divergence issues.

Let us consider the trust-region based l_1-penalty QP subproblem, which can be formulated as

$$
\begin{aligned}
l_1\text{-}QP_k: \min_d \; & \nabla f(x^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k, v^k) d \\
& + \mu \left(\sum_{i=1}^{m} \left[g_i(x^k) + \nabla g_i(x^k)^\top d \right]^+ + \sum_{i=1}^{l} \left| h_i(x^k) + \nabla h_i(x^k)^\top d \right| \right) \\
\text{subject to: } & -\Delta^k \leq d \leq \Delta^k,
\end{aligned}
$$

where $\mu > 0$ is a penalty term, $[\,\cdot\,]^+ = \max\{0, \cdot\}$, and $\Delta^k > 0$ is a trust region term. This variant is of particular interest because the subproblem l_1-QP_k can be recast as a quadratic programming

problem with linear constraints:

$$
\begin{aligned}
l_1\text{-}QP_k: \min_d\ & \nabla f(x^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k, v^k) d + \mu \left(\sum_{i=1}^{m} y_i + \sum_{i=1}^{l} (z_i^+ + z_i^-) \right) \\
\text{subject to: } & -\Delta^k \leq d \leq \Delta^k, \\
& y_i \geq g_i(x^k) + \nabla g_i(x^k)^\top d,\ i = 1, \ldots, m, \\
& z_i^+ - z_i^- = h_i(x^k) + \nabla h_i(x^k)^\top d,\ i = 1, \ldots, l, \\
& y,\, z^+,\, z^- \geq 0.
\end{aligned}
$$

The subproblem l_1-QP_k enjoys the same benefits as the original form of QP_k, meaning that it can be solved with efficient simplex-method based solvers.

The trust-region variant of l_1-SQP is globally convergent (does not diverge) and enjoys superlinear convergence rate, as does the original SQP. The l_1-penalty term is what is often called a *merit function* in the literature. Alternatively, one can disregard the trust region and employ a line search (either exact or inexact) which would also enjoy globally convergent properties.

4 Generalised reduced gradient*

The generalised reduced gradient method resembles in many aspects the simplex method for linear optimisation problems. It derives from the Wolfe's reduced gradient. The term "reduced" refers to the consideration of a reduced variable space, formed by a subset of the decision variables.

4.1 Wolfe's reduced gradient

Let us consider the linearly constrained problem:

$$
\begin{aligned}
(P)\ :\ \text{min. } & f(x) \\
\text{subject to: } & Ax = b \\
& Ax \geq 0,
\end{aligned}
$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, A is $m \times n$ and $b \in \mathbb{R}^m$.

To ease the illustration, we assume linear programming *nondegeneracy*, i.e., that any m columns of A are linearly independent and every extreme point of feasible region has at least m positive components and at most $n - m$ zero components.

Being so, let us define a partition of A as $A = (B, N)$, $x^\top = (x_B^\top, x_N^\top)$, where B is an invertible

$m \times m$ matrix with $x_B > 0$ as a basic vector and $x_N \geq 0$ as a nonbasic vector. This implies that $\nabla f(x)^\top$ can also be partitioned as $\nabla f(x)^\top = (\nabla_B f(x)^\top, \nabla_N f(x)^\top)$.

In this context, for d to be an improving feasible direction, we must observe that

1. $\nabla f(x)^\top d < 0$
2. $Ad = 0$, with $d_j \geq 0$ if $x_j = 0$ to retain feasibility.

We will show how to obtain a direction d that satisfies conditions 1 and 2. For that, let $d^\top = (d_B^\top, d_N^\top)$. We have that $0 = Ad = Bd_B + Nd_n$ for any d_N, implying that $d_B = -B^{-1}Nd_N$. Moreover,

$$\begin{aligned} \nabla f(x)^\top d &= \nabla_B f(x)^\top d_B + \nabla_N f(x)^\top d_N \\ &= (\nabla_N f(x)^\top - \nabla_B f(x)^\top B^{-1}N)d_N \end{aligned} \tag{6}$$

The term $r_N^\top = (\nabla_N f(x)^\top - \nabla_B f(x)^\top B^{-1}N)$ is referred to as the reduced gradient as it express the gradient of the function in terms of the nonbasic directions only. Notice that the reduced gradient r holds a resemblance to the reduced costs from the simplex method. In fact

$$\begin{aligned} r^\top = (r_B^\top, r_N^\top) &= \nabla f(x) - \nabla_B f(x)^\top B^{-1}A \\ &= (\nabla_B f(x) - \nabla_B f(x)^\top B^{-1}B, \nabla_N f(x) - \nabla_B f(x)^\top B^{-1}N) \\ &= (0, \nabla_N f(x) - \nabla_B f(x)^\top B^{-1}N), \end{aligned}$$

and thus $\nabla f(x) = r^\top d$.

Therefore, d_N must be chosen such that $r_N^\top d_N < 0$ and $d_j \geq 0$ if $x_j = 0$. One way of achieving so is employing the classic *Wolfe's rule*, which states that

$$d_j = \begin{cases} -r_j, \text{ if } r_j \leq 0, \\ -x_j r_j, \text{ if } r_j > 0. \end{cases}$$

Notice that the rule is related with the direction of the optimisation. For $r_j \leq 0$, one wants to increase the value of x_j in that coordinate direction, making d_j non-negative. On the other hand, if the reduced gradient is positive ($r_j > 0$), one wants to reduce the value of x_j, unless it is already zero, a safeguard created by multiplying x_j in the definition of the direction d.

The following result guarantee the convergence of Wolfe's reduced gradient to a KKT point.

Theorem 2. *Let $\overline{x}$ be a feasible solution to P such that $\overline{x} = (\overline{x}_B^\top, \overline{x}_N^\top)$ and $x_B > 0$. Consider that A is decomposed accordingly into (B, N). Let $r^\top = \nabla f(\overline{x})^\top - \nabla_B f(\overline{x})^\top B^{-1}A$ and let d be formed using Wolfe's rule. Then*

1. *if $d \neq 0$, then d is an improving direction;*

2. if $d = 0$, then $\overline{x}$ is a KKT point.

Proof. d is a feasible direction by construction. Now, notice that

$$\begin{aligned}\nabla f(\overline{x})^\top d &= \nabla_B f(\overline{x})^\top d_B + \nabla_N f(\overline{x})^\top d_N \\ &= [\nabla_N f(\overline{x})^\top - \nabla_B f(\overline{x})^\top B^{-1} N] d_N = \sum_{j \notin I_B} r_j d_j\end{aligned}$$

where I_B is the index set of basic variables. By construction (using Wolfe's rule), either $d = 0$ or $\nabla f(\overline{x})^\top d < 0$, being thus an *improvement direction*.

$\overline{x}$ is a KKT point if and only if there exists $(u_B^\top, u_N^\top) \geq (0, 0)$ and v such that

$$(\nabla_B f(x)^\top, \nabla_N f(x)^\top) + v^\top (B, N) - (u_B^\top, u_N^\top) = (0, 0) \tag{7}$$

$$u_B^\top x_B = 0, u_N^\top x_N = 0. \tag{8}$$

Since $x_B > 0$ and $u_B \geq 0$, $u_B^\top x_B = 0$ if and only if $u_B = 0$. From top row in (7), $v^\top = -\nabla_B f(x) B^{-1}$. Substituting in the bottom row of (7), we obtain $u_N^\top = \nabla_N f(x)^\top - \nabla_B f(x)^\top B^{-1} N = r_N$.

Thus, the KKT conditions reduce to $r_N \geq 0$ and $r_N^\top x_N = 0$, only observed when $d = 0$ by definition. □

Algorithm 3 presents a pseudocode for the Wolfe's reduced gradient. A few implementation details stand out. First, notice that the basis is selected choosing the largest components in value, which differs from the simplex method by allowing for nonbasic variables to assume nonzero values. Moreover, notice that a line search is employed conditioned on bounds on the step size λ to guarantee that feasibility $x \geq 0$ is retained.

Algorithm 3 Wolfe's reduced gradient method

initialise. $\epsilon > 0, x^0$ with $Ax^k = b, k = 0$, columns a_j, $j = 1, \ldots, n$ of A
while $||d^k|| > \epsilon$ **do**
  $I^k =$ index set for m largest components of x^k
  Let $A = (B, N)$, where $B = \left\{a_j : j \in I^k\right\}$, and $N = \left\{a_j : j \notin I^k\right\}$
  $r^{k\top} = \nabla f(x^k)^\top - \nabla_B f(x^k)^\top B^{-1} A$
  $d_j^k = \begin{cases} -r_j^k, \text{ if } j \notin I^k \text{ and } r_j \leq 0; \\ -r_j x_j, \text{ if } j \notin I^k and r_j > 0. \end{cases}$
  $d_B = -B^{-1} N d_N$
  if $d = 0$ **then**
    return x^k
  end if
  $\overline{\lambda} = \begin{cases} +\infty, \text{ if } d^k \geq 0; \\ \min\left\{x_j^k / d_j^k : d_j^k < 0\right\}, \text{ if } d^k < 0. \end{cases}$
  $\lambda^k = \arg\min\left\{f(x^k + \lambda d^k) : 0 \leq \lambda \leq \overline{\lambda}\right\}$.
  $x^{k+1} = x^k + \lambda^k d^k; k = k + 1.$
end while
return x^k.

4.2 Generalised reduced gradient method

The *generalised reduced gradient* extends Wolfe's method by:

1. Nonlinear constraints are considered via first-order approximation at x^k

$$h(x^k) + \nabla h(x^k)^\top (x - x^k) = 0 \Rightarrow h(x^k)^\top x = h(x^k)^\top x^k.$$

   with an additional *restoration phase* that has the purpose of recovering feasibility via projection or using Newton's method.

2. Consideration of *superbasic variables*. In that, x_N is further partitioned into $x_N^\top = (x_S^\top, x_{N'}^\top)$.

The superbasic variables x_S (with index set J_S, $0 \leq |J_S| \leq n - m$), are allowed change value, while $x_{N'}$ are kept at their current value. Hence, $d^\top = (d_B^\top, d_S^\top, d_{N'}^\top)$, with $d_{N'} = 0$. From $Ad = 0$, we obtain $d_B = -B^{-1} S d_S$. Thus d becomes

$$d = \begin{bmatrix} d_B \\ d_S \\ d_{N'} \end{bmatrix} = \begin{bmatrix} -B^{-1}S \\ I \\ 0 \end{bmatrix} d_S.$$