
MSE2122 - Nonlinear Optimization Lecturer Notes I-XI

Fernando Dias (based on previous version by
Fabricio Oliveira)

October 26, 2023

Abstract

This is a compilation of the Lecture Notes between week I and week XI. Be aware that updates on the individual notes are made weekly while the semester is ongoing.

Contents

1	Week I	4
1.1	What is optimisation?	4
1.1.1	Mathematical programming and optimisation	4
1.1.2	Types of mathematical optimisation models	5
1.2	Examples of applications	6
1.2.1	Resource allocation and portfolio optimisation	6
1.3	The pooling problem: refinery operations planning	7
1.4	Robust optimisation	7
1.5	Combinatorial Optimization	9
1.6	Classification: support-vector machines	11
2	Week II	14
2.1	Convexity and optimisation	14
2.2	Identifying convexity of sets	14
2.2.1	Convexity-preserving set operations	15
2.2.2	Examples of convex sets	15
2.3	Convex hulls	17
2.4	Closure and interior of sets	18
2.4.1	Closure, interior and boundary of a set	18
2.4.2	The Weierstrass theorem	19
2.5	Separation and support of sets	19
2.5.1	Hyperplanes and closest points	20
2.6	Halfspaces and separation	20
2.6.1	Farkas' theorem	21
2.6.2	Supporting hyperplanes	22
3	Week III	24
3.1	Convexity in functions	24

3.1.1	Example of convex functions	24
3.1.2	Convex functions and their level sets	25
3.1.3	Convex functions and their epigraphs	25
3.2	Differentiability of functions	26
3.2.1	Subgradients and supporting hyperplanes	26
3.2.2	Differentiability and gradients for convex functions	27
3.2.3	Second-order differentiability	28
3.3	Quasiconvexity	29
4	Week IV	32
4.1	Recognising optimality	32
4.2	The role of convexity in optimality conditions	32
4.3	Optimality condition of convex problems	34
4.3.1	Optimality conditions for unconstrained problems	36
5	Week V	39
5.1	A prototype of an optimisation method	39
5.1.1	Line search methods	39
5.1.2	Exact line searches	40
5.1.3	Inexact line search	43
5.2	Unconstrained optimisation methods	44
5.2.1	Coordinate descent	44
5.2.2	Gradient (descent) method	45
5.2.3	Newton's method	46
6	Week VI	50
6.1	Unconstrained optimisation methods	50
6.1.1	Conjugate gradient method	50
6.1.2	Quasi Newton: BFGS method	54
6.2	Complexity, convergence and conditioning	57
6.2.1	Complexity	57
6.2.2	Convergence	58
6.2.3	Conditioning	59
7	Week VII	61
7.1	Optimality for constrained problems	61
7.1.1	Inequality constrained problems	62
7.2	Fritz-John conditions	62
7.3	Karush-Kuhn-Tucker conditions	63
7.4	Constraint qualification	65
8	Week VIII	68
8.1	The concept of relaxation	68
8.2	Lagrangian dual problems	69
8.2.1	Weak and strong duality	69
8.2.2	Employing Lagrangian duality for solving optimisation problems	76
8.2.3	Saddle point optimality and KKT conditions	76
8.3	Properties of Lagrangian functions	77
8.3.1	The subgradient method	78
9	Week IX	80
9.1	The concept of feasible directions	80
9.2	Conditional gradient - the Frank-Wolfe method	80
9.3	Sequential quadratic programming	81
9.4	Generalised reduced gradient	85
9.4.1	Wolfe's reduced gradient	85
9.4.2	Generalised reduced gradient method	86
10	Week X	88
10.1	Barrier functions	88
10.2	The barrier method	89
10.3	Interior point method for LP/QP problems	90

10.3.1 Primal/dual path-following interior point method	93
11 Week XI	96
11.1 Penalty functions	96
11.1.1 Geometric interpretation	97
11.1.2 Penalty function methods	98
11.2 Augmented Lagrangian method of multipliers	100
11.2.1 Augmented Lagrangian method of multipliers	100
11.2.2 Alternating direction method of multipliers - ADMM	102

1 Week I

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

1.1 What is optimisation?

Let us start with a simple definition, according to Oxford Dictionary:

Optimization (or optimisation, with British spelling):

- (*noun*) the action of making the **best** or **most effective** use of a **situation** or **resource**.

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**, a subfield of applied mathematics.

In **mathematical optimisation**, we build upon concepts and techniques from basic mathematical subfields. From **calculus**, **analysis**, **linear algebra**, and other, we try to model and develop **methods** that allow us to find 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:

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

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 **potential** values for x .

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

- **Analysing/visualising** properties of functions under specific domains and deriving the conditions that must be satisfied such that a point x is a candidate optimal point.
- **Applying** numerical methods that iteratively search 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. In addition, more and more fields, are recurring to operations research to find innovative solutions.

1.1.1 Mathematical programming and optimisation

Operations research and mathematical optimization are somewhat **intertwined**, as they were both born in similar circumstances. In the decades after the two world wars, **operations research** tools were more widely applied to problems in **business**, **industry**, and **engineering**. Since that time, operation research has expanded into a field widely used in industries ranging from petrochemicals to airlines, finance, logistics, and government, moving to a focus on the development of mathematical models that can be used to analyze and optimize sometimes complex systems. It has become an area of active academic and industrial research.

With the development of computers over the next three decades after WWI, **Operations Research** can now solve problems with hundreds of thousands of variables and constraints. Moreover, the large volumes of data required for such problems can be stored and manipulated very efficiently. Much of operations research relies upon stochastic variables and, therefore, access to truly random numbers. Fortunately, the cybernetics field also required the same level of randomness. More recently, the operations research approach, which dates back to the 1950s, has been criticized for being a collection of **mathematical models** but lacking an empirical basis for data collection for applications.

Let us separate **mathematical programming** from (mathematical) **optimization**. Mathematical

programming is a modelling paradigm in which we rely on (compelling) analogies to model *real-world* problems. In that, we look at a given decision problem considering that:

- **variables** represent *decisions* or *interest*, 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* and *limitations*, representing logic relations, design or engineering limitations, requirements, and such;
- function is an *objective function* that measures solution quality, profit or goal.

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 **optimization** methods. From now on, we will refer to this specific class of models as mathematical optimization models or optimization models for short. We will also use the term to **solve the problem** to refer to finding optimal solutions to optimization models.

This course mostly focuses on the **optimization techniques** employed to find **optimal solutions** for these models. As we will see, depending on the nature of the functions f and g 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 generalized consensus whether one method is generally better performing than another.

Therefore, in this course, we are focusing on **optimization** methods to find optimal solutions considering that either f and/or g are **nonlinear** functions, hence the name of the course.

1.1.2 Types of mathematical optimisation models

In general, **the rule of thumb** for solving a problem is :

The simpler the assumptions on the parts forming the optimisation model, the more efficient the methods for solving it.

Let us define some additional notation that we will use from now on. Consider a model in the **general (or standard) form**:

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

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, an **equality** constraint can represent every inequality 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 a very different nature, we will explicitly represent both whenever necessary.

Remark: be aware that some mathematicians and books might refer to both type of inequalities as a single one

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

- **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.
- **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, $X = \{x \in \mathbb{R}^n \mid x_j \geq 0, j = 1, \dots, n\}$ enforce that decision variables are constrained to be the nonnegative orthant.

- **Nonlinear programming (NLP):** some or all of the functions f , g , and h are nonlinear.
- **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 $i = 1, \dots, n - k$ and are meant to represent true-or-false or yes-or-no conditions.
- **Mixed-integer nonlinear programming (MINLP):** are the intersection of MIPs and NLPs.

See Diagram below:

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

1.2 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.

1.2.1 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 $i \in I = \{1, \dots, M\}$ be a collection of resources and $j \in J = \{1, \dots, N\}$ 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:

$$\max. \sum_{j \in J} c_j x_j \quad (2)$$

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

$$x_j \geq 0, \forall j \in J. \quad (4)$$

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 $j \in J = \{1, \dots, N\}$ 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:

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

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:

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

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

$$0 \leq x_j \leq 1, \quad \forall j \in J. \tag{7}$$

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. Constraint (7) defined 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.

1.3 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 $[q_j, \bar{q}_j]$ to be classified as product j . In this case, mass and property balances are calculated as:

$$x_j = \sum_{i \in I_j} x_i, \quad \forall j \in J \tag{8}$$

$$q_j = \frac{\sum_{i \in I_j} q_i x_i}{x_j}, \quad \forall j \in J. \tag{9}$$

These can then be incorporated into the resource allocation problem accordingly. One key aspect associated with pooling problem formulations is that the property balances represented by (9) 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.

1.4 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 1.2.1. 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:

$$\max. \quad c^\top x$$

$$\text{subject to: } \tilde{a}_i^\top x \leq b_i, \quad \forall i \in I$$

$$x_j \geq 0, \quad \forall j \in J.$$

Let us assume that the only information available are observations \hat{a}_i , from which we can estimate a

nominal value \bar{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 terms of feasibility

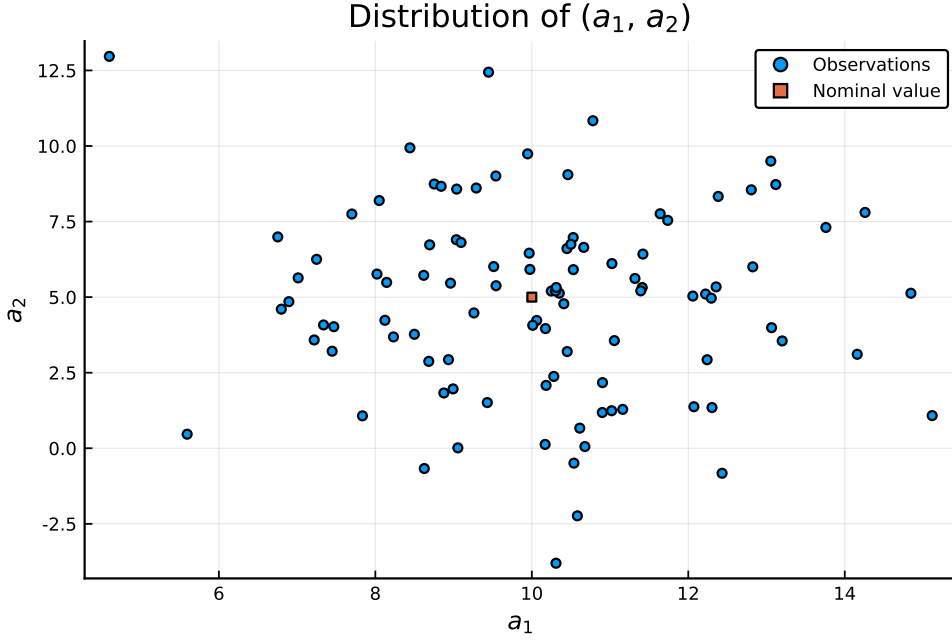


Figure 1: One hundred random realisations for \tilde{a}_i .

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:

$$\epsilon_i = \{\bar{a}_i + P_i u \mid \|u\|_2 \leq \Gamma_i\} \quad (10)$$

is an ellipsoid with the characteristic matrix P_i (i.e., its eigenvalues show how the ellipsoid extends in every direction from \bar{a}_i) and Γ_i employs a scaling of the ellipsoid size.

Remark: an alternative (perhaps more frequent) characterisation of an ellipsoid $\epsilon \subset \mathbb{R}^n$ centred at \bar{x} is given by $\epsilon = \{x \in \mathbb{R}^n \mid (x - \bar{x})^\top A (x - \bar{x}) = 1\}$. By making $A = P^{-2}$, we recover the representation in (10).

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.

$$\begin{aligned} & \max. \quad c^\top x \\ & \text{subject to:} \quad \max_{a_i \in \epsilon_i} \{a_i^\top x\} \leq b_i, \quad \forall i \in I \\ & \quad \quad \quad x_j \geq 0, \quad \forall j \in J. \end{aligned} \quad (11)$$

Notice how the constraint (11) 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.

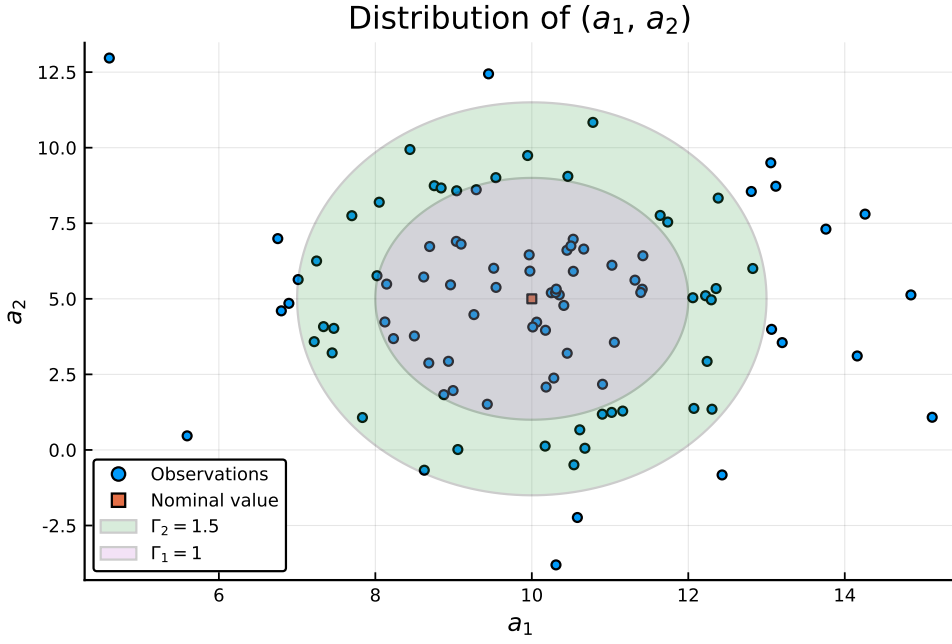


Figure 2: One hundred random realisations for \tilde{a}_i .

$$\max_{a_i \in \epsilon_i} \{a_i^\top x\} = \bar{a}_i^\top x + \max_u \{u^\top P_i x : \|u\|_2 \leq \Gamma_i\} \quad (12)$$

$$= \bar{a}_i^\top x + \Gamma_i \|P_i x\|_2. \quad (13)$$

In (12), 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 (13) we can derive a closed form for the inner optimisation problem.

With the closed form derived in (13), we can reformulate the original bi-level problem as a tractable single-level problem of the following form:

$$\begin{aligned} \max. \quad & c^\top x \\ \text{subject to:} \quad & \bar{a}_i^\top x + \Gamma_i \|P_i x\|_2 \leq b_i, \quad \forall i \in I \\ & x_j \geq 0, \quad \forall j \in J. \end{aligned} \quad (14)$$

Notice how the term $\Gamma_i \|P_i^\top x\|_2$ creates a buffer for constraint (14), 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*.

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

$$\epsilon_i = \left\{ \begin{bmatrix} 10 \\ 5 \end{bmatrix} + \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right\} \quad (15)$$

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.

1.5 Combinatorial Optimization

Combinatorial optimization is a sub-field of mathematical optimization that consists of finding an optimal solution from a finite **set** of possible solutions from a discrete **set**. Examples of **combinatorial problems** are the **travelling salesman problem** ("TSP"), the minimum spanning tree problem ("MST"), and the knapsack problem. The most trivial solution for such problems is exhaustive and enumerative search, although it is not tractable for larger solutions.

There is a large amount of literature on polynomial-time algorithms for certain special classes of

discrete optimization. The theory of linear programming unifies a considerable amount of it. Some examples of combinatorial optimization problems this framework covers are shortest paths and shortest-path trees, flows and circulations, spanning trees, matching, and matroid problems.

Many problems covered by linear optimization could also be described as combinatorial optimization problems (as part of discrete optimization). For example, shortest path, flow and circulations, spanning tree, matching, etc. In this theoretical framework, research on such problems focuses on the following topics: solution for special cases of discrete problems using specific parameters (fixed-parameter tractable (FPT) problems); approximation algorithms; heuristics and meta-heuristics; parameterized optimization (variation of FPT); NP-completeness.

Combinatorial optimization problems can be viewed as searching for the best solution amongst a set of viable solutions. Perhaps, as a search procedure. Many algorithms, such as branch-and-bound, branch-and-cut, dynamic programming, tabu search, can solve many variations of discrete problems. However, generic search algorithms are not guaranteed to find an optimal solution first, nor are they guaranteed to run quickly (in polynomial time). Since some discrete optimization problems are NP-complete, such as the travelling salesman (decision) problem (unless $P=NP$).

We can formally define a combinatorial optimization problem using the following notation:

- a set of **candidate** solutions I ;
- a subset of **feasible** solutions $f(x)$, such that $x \in I$;
- the performance **measure** of the feasible solutions $m(x, y)$;
- the **goal** function (generally minimization or maximization)

In this way, *an optimal solution can be found* when:

$$m(x, y) = g\{m(x, y') | y' \in f(x)\} \tag{16}$$

Each combinatorial optimization problem has a corresponding decision problem that asks whether there is a feasible solution for some particular measure m_0 .

The field of approximation algorithms deals with algorithms to find near-optimal solutions to challenging problems where exact and fast solutions are costly (and sometimes, impossible).

Classical examples of combinatorial problems are:

- **TSP (Travelling Salesman Problem):**
This problem can be defined as:

Given a set of cities and the distances between each pair of cities, what is the shortest possible route that simultaneously visits each city only once and returns to the origin city?

Variations of this problem, such as the travelling purchaser and vehicle routing problem (very present in logistics and transportation problems), are vastly researched.

- **Minimum spanning tree:**

A minimum spanning tree problem is a subset of edges from an undirected graph connecting all vertices without cycles, minimizing possible total edge weight.

There are many algorithms and solutions for this problem. However, they lack versatility when this problem's alterations and variations are brought into question.

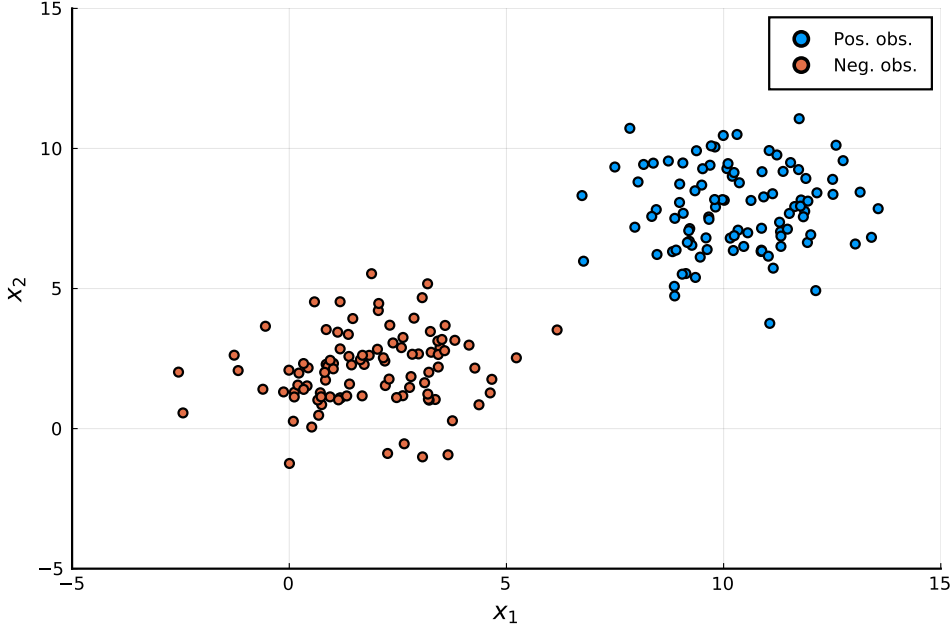


Figure 3: Two hundred observations for x_i classified to belong to I^- (orange) or I^+ (blue).

1.6 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 $D \in \mathbb{R}^n$ with $|D| = N + M$ that can be divided into two disjunct sets $I^- = \{x_1, \dots, x_N\}$ and $I^+ = \{x_1, \dots, x_M\}$.

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).

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:

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

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 $i = 1, \dots, N$ and $v_i \geq 0$ for $i = 1, \dots, M$ 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:

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

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

$$a^\top x_i - b + v_i \geq 0, i = 1, \dots, N \quad (20)$$

$$\|a\|_2 = 1 \quad (21)$$

$$u_i \geq 0, i = 1, \dots, M \quad (22)$$

$$v_i \geq 0, i = 1, \dots, N \quad (23)$$

$$a \in \mathbb{R}^n, b \in \mathbb{R}. \quad (24)$$

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.

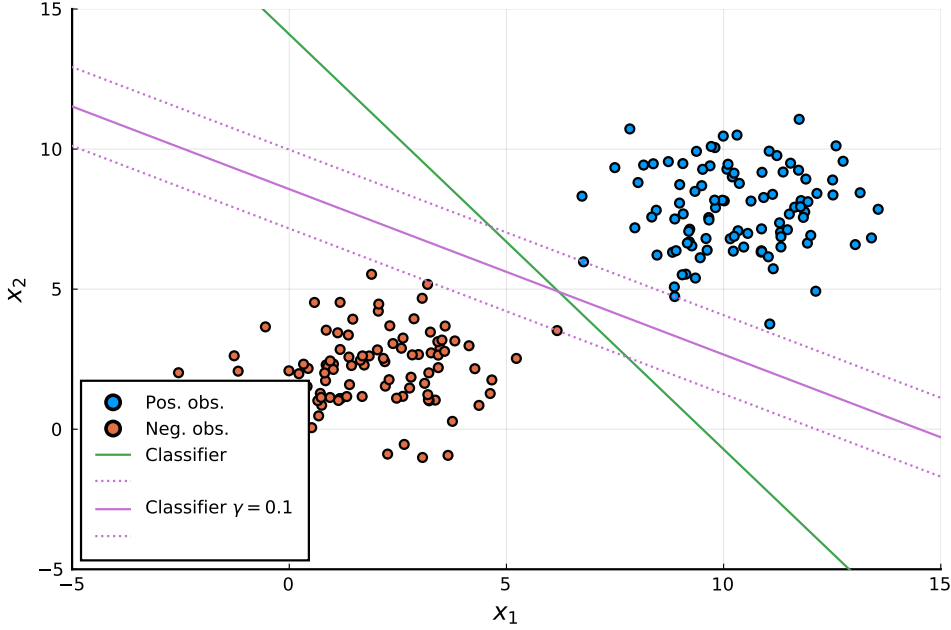


Figure 4: Two hundred observations for x_i classified to belong to I^- (orange) or I^+ (blue) with a classifier (green).

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.

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

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

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:

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

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

$$a^\top x_i - b + v_i \geq 1, \quad i = 1, \dots, N \quad (27)$$

$$u_i \geq 0, \quad i = 1, \dots, N \quad (28)$$

$$v_i \geq 0, \quad i = 1, \dots, M \quad (29)$$

$$a \in \mathbb{R}^n, b \in \mathbb{R}. \quad (30)$$

In objective function (25), the errors accumulated in variables $u_i, i = 1, \dots, N$ and $v_i, i = 1, \dots, M$ 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 .

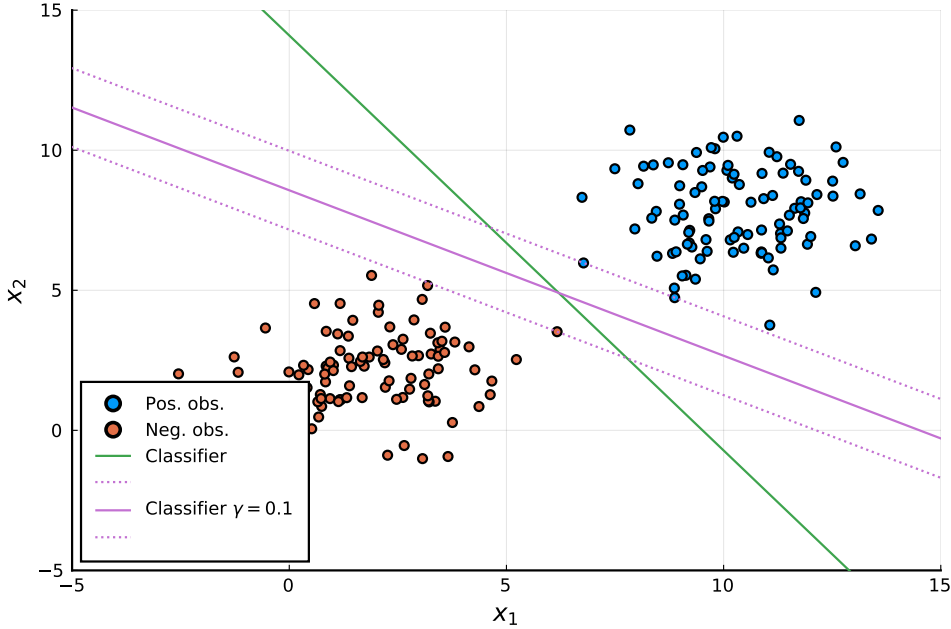


Figure 5: Two hundred observations for x_i classified to belong to I^- (orange) or I^+ (blue).

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.

2 Week II

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 practically when discussing optimality conditions later. We also discuss the concepts of separation and support, which are central to several results in optimisation theory, such as the Farkas theorem.

2.1 Convexity and optimisation

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

... in fact, the great watershed in optimization is not between linearity and nonlinearity but convexity and nonconvexity.

In a nutshell, convexity allows us to infer the **global properties** of a solution (i.e., that holds for all of its domain) by considering exclusively local information (such as gradients, for example). Such property is critical in **optimization** since most 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:

$$(P) : \min. f(x) \\ \text{subject to: } x \in X$$

to be convex, we need to verify whether f is a **convex function** and X is a **convex set**. If both statements hold, 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.

2.2 Identifying convexity of sets

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

- A **linear combination** of x_j for $j = 1, \dots, k$ is the set

$$\{x \in \mathbb{R}^n : \sum_{j=1}^k \lambda_j x_j, \lambda_j \in \mathbb{R} \text{ for } j = 1, \dots, k\}. \quad (31)$$

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

$$\{x \in \mathbb{R}^n : \sum_{j=1}^k \lambda_j x_j, \sum_{j=1}^k \lambda_j = 1, \lambda_j \in \mathbb{R} \text{ for } j = 1, \dots, k\}. \quad (32)$$

- A **conic combination** is a linear combination with the additional condition that $\lambda_j \geq 0$ for $j = 1, \dots, k$.

$$\{x \in \mathbb{R}^n : \sum_{j=1}^k \lambda_j x_j, \lambda_j \geq 0 \text{ for } j = 1, \dots, k\}. \quad (33)$$

- And finally, a **convex combination** is the intersection between an affine and a conic combination, implying that $\lambda_j \in [0, 1]$.

$$\{x \in \mathbb{R}^n : \sum_{j=1}^k \lambda_j x_j, \sum_{j=1}^k \lambda_j = 1, \lambda_j \geq 0 \text{ for } j = 1, \dots, k\}. \quad (34)$$

We say a set is convex if it contains all points formed by the convex combination of any pair of points in this set, which is equivalent to saying that the set contains the line segment between any two points

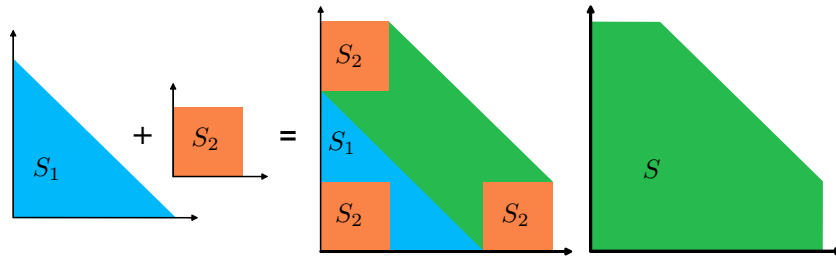


Figure 6: Minkowski sum of two convex sets.

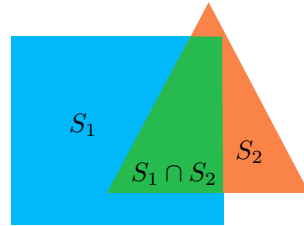


Figure 7: Intersection of two convex sets.

belonging to the set.

Definition 2.1. Convex sets

A set $S \subseteq \mathbb{R}^n$ is said to be convex if $\bar{x} = \sum_{j=1}^k \lambda_j x_j$ belongs to S , where $\sum_{j=1}^k \lambda_j = 1$, $\lambda_j \geq 0$ and $x_j \in S$ for $j = 1, \dots, k$.

Definition 2.1 is useful as it shows that some set operations preserve convexity.

2.2.1 Convexity-preserving set operations

Lemma 2.2. 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 6 and 7 illustrate the concept behind some of these set operations. Showing that the sets resulting from the operations in Lemma 2.2 are convex typically shows that convex combinations of elements in the resulting set S also belong to S_1 and S_2 .

2.2.2 Examples of convex sets

There are several familiar sets that are known to be convex. Knowing 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:

- The empty set \emptyset , any singleton $\{\bar{x}\}$ and the whole space \mathbb{R}^n ;
- **halfspaces**: $S = \{x : p^\top x \leq \alpha\} \subset \mathbb{R}^n$;
- **hyperplanes**: $H = \{x : p^\top x = \alpha\} \subset \mathbb{R}^n$, where $p \neq 0^n$ is a normal vector and $\alpha \in \mathbb{R}$ is a scalar. Notice that H can be equivalently represented as $H = \{x \in \mathbb{R}^n : p^\top (x - \bar{x}) = 0\}$ for $\bar{x} \in H$;
- **polyhedral sets**: $P = \{x : Ax \leq b\} \subset \mathbb{R}^n$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$;

- **norm-induced sets (balls):** $B = \{x : \|x - \bar{x}\| \leq \alpha\} \subseteq \mathbb{R}^n$, where $\|\cdot\|$ is any norm and α a scalar;
- **norm cones:** $C = \{(x, \alpha) \in \mathbb{R}^{n+1} : \|x\| \leq \alpha\}$;

For example, considering 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, $P = \bigcap_{i=1}^m H_i$ is also convex, as the intersection of sets is a convexity-preserving set operation.

1. Hyperplanes and halfspaces:

Hyperplanes and **half-spaces** will play a central role in future topics. Therefore, let us discuss some important aspects related to 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 \bar{x} determines the offset of the hyperplane from the origin. That is:

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

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

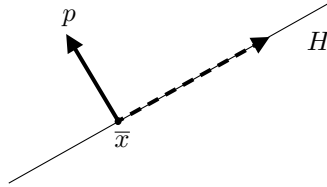


Figure 8: A hyperplane $H = \{x \in \mathbb{R}^n : p^\top(x - \bar{x}) = 0\}$ with normal vector p displaced to \bar{x} .

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

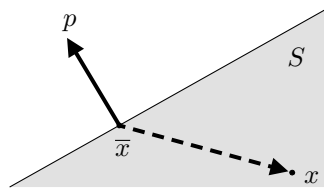


Figure 9: A halfspace $S = \{x \in \mathbb{R}^n : p^\top(x - \bar{x}) \leq 0\}$ defined by the same hyperplane H . Notice how the vectors p (or $p - \bar{x}$, which is fundamentally the same vector but translated to \bar{x}) and $x - \bar{x}$ form angles greater or equal than 90° .

2. Norm balls and norm cones:

An **Euclidean ball** (or **simply ball**) of radius ϵ in \mathbb{R}^n has the form:

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

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

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

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 $B(\bar{x}, \epsilon) = \{x \in \mathbb{R}^n : \|x - \bar{x}\| \leq$

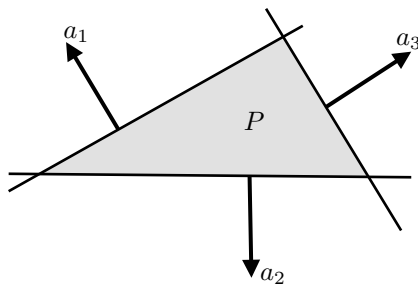


Figure 10: A polyhedron P formed by the intersection of three half-space. Each hyperplane $H_i = \{x \in \mathbb{R}^n : a_i^\top x \leq b_i\}$, for $i = 1, 2, 3$, has a normal vector a_i , and has an offset from the origin b_i (which cannot be seen once project on a 2-dimensional plane as in the picture).

$\epsilon\}$ 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 $p \geq 1$.

2.3 Convex hulls

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

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

Definition 2.3. 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 S . That is, for $x_j \in S$, with $j = 1, \dots, k$, $x \in \mathbf{conv}(S)$ if and only if

$$x = \sum_{j=1}^k \lambda_j x_j : \sum_{j=1}^k \lambda_j = 1, \lambda_j \geq 0, \text{ for } j = 1, \dots, k.$$

From Definition 2.3, 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 visualize this is to think of the infinitely many half-spaces containing S and their intersection, which can only be S . Figures 11 illustrates the convex hull $\mathbf{conv}(S)$ of a nonconvex set S .

The notion of convex hulls is a powerful tool in optimization. One important application is using

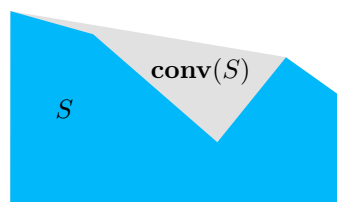


Figure 11: Example of an arbitrary set S (in solid blue) and its convex hull $\mathbf{conv}(S)$ (combined blue and grey areas).

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

Let us consider the convex hull of a finite collection of discrete points. Some of these sets are so important in optimization that they have their own names.

Definition 2.4

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

2.4 Closure and interior of sets

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

2.4.1 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 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 elements related to S .

1. **Closure of S** : the set defined by 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 not belonging to S . We say that a set is **closed** if $S = \mathbf{clo}(S)$; the set itself is its own closure.

2. **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, $\mathbf{int}(S) \subseteq S$.

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

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

We say that S is bounded if exists $N_\epsilon(x)$, $x \in \mathbb{R}^n$, for some $\epsilon > 0$ such that $S \subset N_\epsilon(x)$.

We say that a set is **compact** if it is both **closed** and **bounded**. Compact sets appear very frequently in real-world optimization applications since, typically, one can assume the existence of bounds for decision variables (such as nonnegativity or maximum physical bounds or, in an extreme case, smallest/largest computational constants). Another frequent example of a 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. $\mathbf{bou}(S) = \{(x_1, x_2) \in \mathbb{R}^n : x_1^2 + x_2^2 = 1\}$. Notice that it makes S bounded.
4. S is compact since it is closed and 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 2.5

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 2.5 helps infer the convexity of the elements related to S . We summarise the key results in the following corollary.

Corollary 2.6. Let S be a convex set with $\text{int}(S) \neq \emptyset$. Then

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

2.4.2 The Weierstrass theorem

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

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

be our optimization 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 **minimize**) and the operator \min . The first is meant to represent the problem of **minimizing** the function f in the domain S , while \min is shorthand for **minimum**, in this case z , assuming 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 maximization 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 $\bar{z} > z$ such that $\bar{z} \leq f(x)$ for all $x \in S$. We might sometimes use the notation:

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

to represent optimization 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 2.7. Weierstrass theorem

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

Figure 12 illustrates three examples. In the first (on the left), the domain $[a, b]$ is compact; thus, the minimum of f is attained at b . In the other two, $[a, b]$ is open; therefore, the 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.

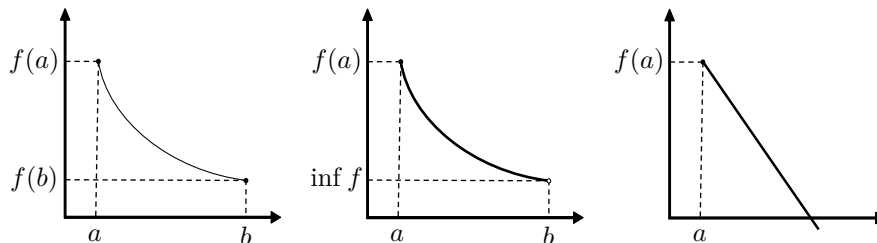


Figure 12: Examples of attainable minimum (left) and infimum (centre) and an example where neither are attainable (right).

2.5 Separation and support of sets

The concepts of **separation** and **support** of sets are vital 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 point not belonging to S .

2.5.1 Hyperplanes and closest points

We start with how to identify the closest points to sets.

Theorem 2.8. 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 $\bar{x} \in S$ with minimum distance from y . In addition, \bar{x} is the minimising point if and only if

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

Simply put, if S is a closed convex set, then $\bar{x} \in S$ will be the closest point to $y \notin S$ if the vector $y - \bar{x}$ is such that it forms an angle that is greater or equal than 90° with all other vectors $x - \bar{x}$ for $x \in S$. Figure 13 illustrates this logic.

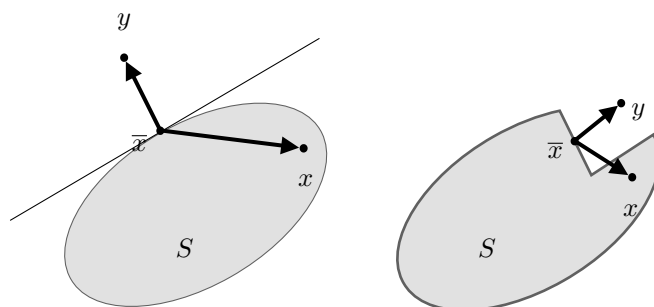


Figure 13: Closest-point theorem for a closed convex set (on the left). On the right is an illustration of how the absence of convexity invalidates the result.

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

2.6 Halfspaces and separation

We can use half-spaces to build the concept of separation. Let us start 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 half-spaces $H^+ = \{x : p^\top x \geq \alpha\}$ and $H^- = \{x : p^\top x \leq \alpha\}$. Figure 14 illustrates the concept. Notice how the vector p lies in the half-space H^+ .

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

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

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

We can now define the separation of convex sets.

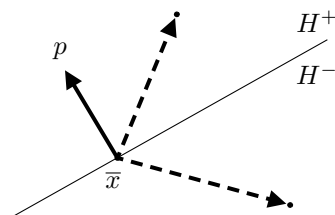


Figure 14: Normal vectors, hyperplane and halfspaces

Definition 2.9

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 15 illustrates the three types of separation in Definition 2.9. On the left, proper separation is illustrated, obtained by any hyperplane that does not contain both S_1 and S_2 but 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. A powerful yet simple result that we will use later is that, for a closed convex set S , there always exists

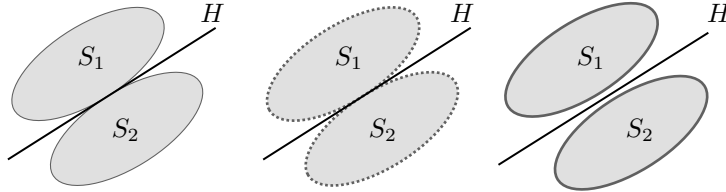


Figure 15: Three types of separation between S_1 and S_2 .

a hyperplane separating S and a point y that does not belong to S .

Theorem 2.10. 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 2.8 guarantees the existence of a unique minimising $\bar{x} \in S$ such that $(y - \bar{x})^\top (x - \bar{x}) \leq 0$ for each $x \in S$. Let $p = (y - \bar{x}) \neq 0$ and $\alpha = \bar{x}^\top (y - \bar{x}) = p^\top \bar{x}$. Then we get $p^\top x \leq \alpha$ for each $x \in S$, while $p^\top y - \alpha = (y - \bar{x})^\top (y - \bar{x}) = \|y - \bar{x}\|^2 > 0$. 🤔

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 optimization problem and uses the Weierstrass theorem (our Theorem 2.8 is a consequence of the Weierstrass theorem stated in Theorem 2.7) to guarantee that such a \bar{x} exists. Being a minimum distance point, we know from Theorem 2.8 that $(y - \bar{x})^\top (x - \bar{x}) \leq 0$ holds. Now, by defining p and α as in the proof, one might notice that:

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

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

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

Theorem 2.10 has interesting consequences. For example, one can apply it to every point in the boundary $\text{bou}(S)$ 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 $y \notin \text{clo}(\text{conv}(S))$, then one can show that a strong separation between y and S exists since there will surely be a distance $\epsilon > 0$ between y and S .

2.6.1 Farkas' theorem

Farkas' theorem plays a central role in deriving optimality conditions. It can assume several alternative forms, typically referred to as Farkas' lemmas. In essence, Farkas' theorem demonstrates 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 2.11

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:

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

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 2.10, 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 $p^\top c > 0$. Also, $\alpha \geq p^\top A^\top y = y^\top Ap$ for $y \geq 0$. 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 2.10) 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 , for $i = 1, \dots, m$. Using the $0 \in S$, one can show that $\alpha \geq 0$. The last part uses the identity $p^\top A^\top = (Ap)^\top$ and the fact that $(Ap)^\top y = y^\top Ap$. Notice that, since y can be arbitrarily large and α is a constant, $y^\top Ap \leq \alpha$ can only hold if $y^\top Ap \leq 0$, requiring that $p \leq 0$ since $y \geq 0$ from the definition of S .

Farkas' theorem has an interesting geometrical interpretation from this proof, as illustrated in Figures 16. Consider the cone C formed by the rows of A :

$$C = \{c \in \mathbb{R}^n : c_j = \sum_{i=1}^m a_{ij}y_i, j = 1, \dots, n, y_i \geq 0, i = 1, \dots, m\}$$

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

$$C^0 = \{x : Ax \leq 0\}.$$

Notice that (1) has a solution if the intersection between the polar cone C^0 and the positive (H^+ as defined earlier) half-space $H^+ = \{x \in \mathbb{R}^n : c^\top x > 0\}$ is not empty. If (2) has a solution, as at 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.

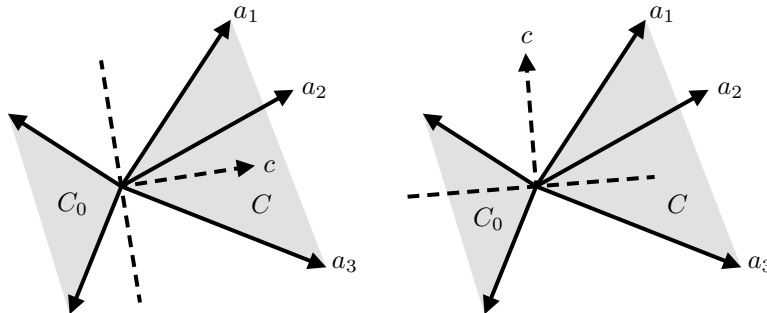


Figure 16: Geometrical illustration of the Farkas' theorem. On the left, system (2) has a solution, while on the right, system (1) has a solution

2.6.2 Supporting hyperplanes

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

Definition 2.12. Supporting hyperplane

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

Figure 17 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.

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

$$\bar{x} = \operatorname{argmax}_{x \in S} f(x)$$

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 toward p . The constraint $x \in S$ will eventually prevent the movement further from S , and this last contact point is precisely \bar{x} . This is a useful concept for optimizing problems using gradients of functions, as we will discuss later in the course.

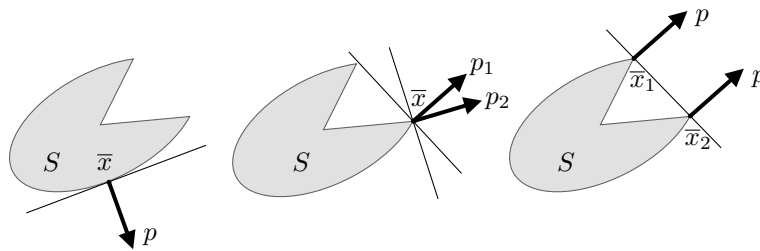


Figure 17: Supporting hyperplanes for an arbitrary set. Notice how a single point might have multiple supporting planes (middle) or different points might have the same supporting hyperplane (right)

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

Theorem 2.13

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

The proof follows immediately from Theorem 2.10, without explicitly considering a point $y \notin S$ and by noticing that $\mathbf{bou}(S) \subset \mathbf{clo}(S)$. Figure 18 illustrates the theorem.

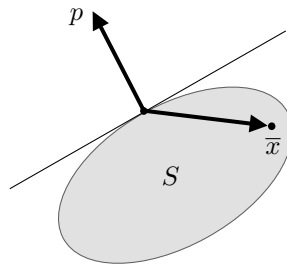


Figure 18: Supporting hyperplanes for convex sets. Notice how every boundary point has at least one supporting hyperplane

3 Week III

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 unique subgradients for differentiable functions. Second-order differentiability is also considered a mechanism to determine the convexity of functions. We finalise discussing the notion of quasiconvexity and how some nonconvex functions can still be present in convex (sub)level sets, which can be exploited for developing optimisation methods.

3.1 Convexity in functions

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

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

with $f : \mathbb{R}^n \mapsto \mathbb{R}$, $g : \mathbb{R}^n \mapsto \mathbb{R}^m$ 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 textbfasise (perhaps redundantly at this point) how crucial it is for us to attest to 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 3.1. Convexity of a function I

Let $f : S \mapsto \mathbb{R}$ 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 $\lambda \in [0, 1]$.

We often use the term convex to loosely refer to concave functions, which must be done cautiously. In fact, if f is convex, then $-f$ 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 3.1 for each $\lambda \in (0, 1)$ (notice the open interval instead). In practice, the function is guaranteed to not present flatness around its minimum (or maximum, for concave functions).

3.1.1 Example of convex functions

Some examples of convex functions 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 -norm);
5. $f(x) = -\log x$ and negative entropy $f(x) = -x \log x$ are concave;
6. $f(x) = \max\{x_1, \dots, x_n\}$.

Knowing these common convex functions helps identify convexity in more complex functions formed by **composition**. 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 $f_1, \dots, f_k : \mathbb{R}^n \mapsto \mathbb{R}$ be convex. Then these are convex:
 - $f(x) = \sum_{j=1}^k \alpha_j f_j(x)$ where $\alpha_j > 0$ for $j = 1, \dots, k$;
 - $f(x) = \max\{f_1(x), \dots, f_k(x)\}$;
2. $f(x) = \frac{1}{g(x)}$ on S , where $g : \mathbb{R}^n \mapsto \mathbb{R}$ is concave and $S = \{x : g(x) > 0\}$;
3. $f(x) = g(h(x))$, where $g : \mathbb{R} \mapsto \mathbb{R}$ is a nondecreasing convex function and $h : \mathbb{R}^n \mapsto \mathbb{R}$ is convex.
4. $f(x) = g(h(x))$, where $g : \mathbb{R}^m \mapsto \mathbb{R}$ is convex and $h : \mathbb{R}^n \mapsto \mathbb{R}^m$ is affine: $h(x) = Ax + b$ with

$A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

3.1.2 Convex functions and their level sets

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

Definition 3.2. Lower level set

Let $S \subseteq \mathbb{R}^n$ be a nonempty set. The lower level set of $f : \mathbb{R}^n \mapsto \mathbb{R}$ for $\alpha \in \mathbb{R}$ is given by

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

Figure 19 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 .

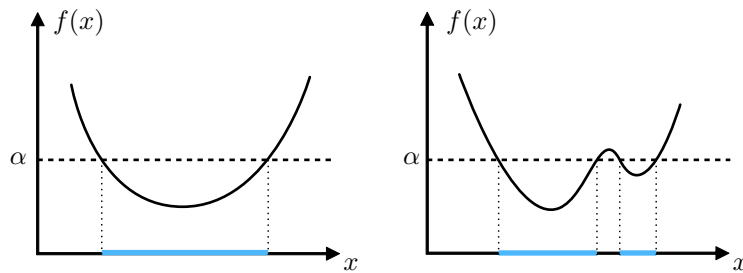


Figure 19: The lower level sets S_α (in blue) of two functions, given a value of α . Notice the nonconvexity of the level set of the nonconvex function (on the right)

Notice that, for convex functions, no discontinuity can be observed, making S_α convex. Lemma 3.3 states this property.

Lemma 3.3 Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set and $f : S \mapsto \mathbb{R}$ 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 the function is convex. As we will see later, there are nonconvex functions that have convex level sets (the so-called quasiconvex functions).

3.1.3 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 3.4. Epigraph

Let $S \subseteq \mathbb{R}^n$ be a nonempty set and $f : S \mapsto \mathbb{R}$. The epigraph of f is

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

Figure 20 illustrates the epigraphs of two functions. Notice that the second function (on the right) is neither convex nor epigraph. We can use the convexity of epigraphs (and the technical results associated with the convexity of sets) to show the convexity of functions.

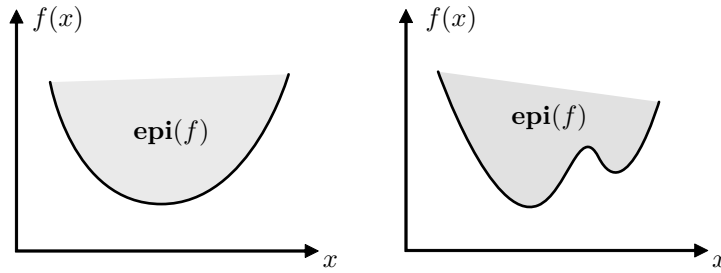


Figure 20: The epigraph $\text{epi}(f)$ of a convex function is a convex set (in grey on the left).

Theorem 3.5. Convex epigraphs

Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set and $f : S \mapsto \mathbb{R}$. Then f is convex if and only if $\text{epi}(f)$ is a convex set.

Proof. First, suppose f is convex and let $(x_1, y_1), (x_2, y_2) \in \text{epi}(f)$ for $\lambda \in (0, 1)$. 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 $\lambda x_1 + (1 - \lambda)x_2 \in S$, $(\lambda x_1 + (1 - \lambda)x_2, \lambda y_1 + (1 - \lambda)y_2) \in \text{epi}(f)$.

Conversely, suppose $\text{epi}(f)$ is convex. Therefore $x_1, x_2 \in S$: $(x_1, f(x_1)) \in \text{epi}(f)$, $(x_2, f(x_2)) \in \text{epi}(f)$ and $(\lambda x_1 + (1 - \lambda)x_2, \lambda f(x_1) + (1 - \lambda)f(x_2)) \in \text{epi}(f)$ for $\lambda \in (0, 1)$, implying that $\lambda f(x_1) + (1 - \lambda)f(x_2) \geq f(\lambda x_1 + (1 - \lambda)x_2)$. 🤔

The proof starts with the implication “if f is convex, then $\text{epi}(f)$ is convex”. For that, it assumes that f is convex and uses the convexity of f to show that any convex combination of x_1, x_2 in S will also be in the $\text{epi}(f)$, which is the definition of a convex set.

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

3.2 Differentiability of functions

3.2.1 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 3.6. Subgradients

Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set and $f : S \mapsto \mathbb{R}$ a convex function. Then $\xi \in \mathbb{R}^n$ is a **subgradient** of f at $\bar{x} \in S$ if

$$f(x) \geq f(\bar{x}) + \xi^\top (x - \bar{x}). \quad (35)$$

Inequality (35) is called the **subgradient inequality** and will be useful in several contexts later in this course. The set of subgradients ξ of f at \bar{x} is the **subdifferential**

$$\partial_f(\bar{x}) = \{\xi \in \mathbb{R}^n : f(x) \geq f(\bar{x}) + \xi^\top (x - \bar{x})\}.$$

Every convex function $f : S \mapsto \mathbb{R}$ has at least one subgradient at any point \bar{x} in the interior of the convex set S . Requiring that $\bar{x} \in \text{int}(S)$ allows us to disregard boundary points of f where $\partial(\bar{x})$ might be empty. Theorem 3.7 presents this result.

Theorem 3.7

Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set and $f : S \mapsto \mathbb{R}$ a convex function. Then for all $\bar{x} \in \text{int}(S)$, there exists $\xi \in \mathbb{R}^n$ such that

$$H = \{(x, y) : y = f(\bar{x}) + \xi^\top (x - \bar{x})\}$$

supports $\text{epi}(f)$ at $(\bar{x}, f(\bar{x}))$. In particular,

$$f(x) \geq f(\bar{x}) + \xi^\top(x - \bar{x}), \forall x \in S.$$

The proof consists of directly applying Theorem 3.5 and then using the support of the convex sets theorem (Theorem 14 in Lecture 2) to show that the subgradient inequality holds.

3.2.2 Differentiability and gradients for convex functions

Let us first define the differentiability of a function.

Definition 3.8

Let $S \subseteq \mathbb{R}^n$ be a nonempty set. The function $f : S \mapsto \mathbb{R}$ is differentiable at $\bar{x} \in \text{int}(S)$ if there exists a vector $\nabla f(\bar{x})$, called a gradient vector, and a function $\alpha : \mathbb{R}^n \mapsto \mathbb{R}$ such that

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

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

Notice that this definition is based on the existence of first-order (Taylor series) expansion, with an error term α . This definition is helpful as it highlights the requirement that $\nabla f(\bar{x})$ exists and is unique at \bar{x} since the gradient is given by $\nabla f(x) = \left[\frac{\partial f(x)}{\partial x_i} \right]_i = 1, \dots, n$.

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 3.9:

Lemma 3.9 Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set and $f : S \mapsto \mathbb{R}$ a convex function. Suppose that f is differentiable at $\bar{x} \in \text{int}(S)$. Then $\partial_f(\bar{x}) = \{\nabla f(\bar{x})\}$, i.e., the subdifferential $\partial_f(\bar{x})$ is a singleton with $\nabla f(\bar{x})$ as its unique element.

Proof. From Theorem 3.7, $\partial f(\bar{x}) \neq \emptyset$. Moreover, combining the existence of a subgradient ξ and differentiability of f at \bar{x} , we obtain:

$$f(\bar{x} + \lambda d) \geq f(\bar{x}) + \lambda \xi^\top d \tag{36}$$

$$f(\bar{x} + \lambda d) = f(\bar{x}) + \lambda \nabla f(\bar{x})^\top d + \lambda \|d\| \alpha(\bar{x}; \lambda d) \tag{37}$$

Subtracting (37) from (36), we get $0 \geq \lambda(\xi - \nabla f(\bar{x}))^\top d - \lambda \|d\| \alpha(\bar{x}; \lambda d)$. Dividing by $\lambda > 0$ and letting $\lambda \mapsto 0^+$, we obtain $(\xi - \nabla f(\bar{x}))^\top d \leq 0$. Now, by setting $d = \xi - \nabla f(\bar{x})$, it becomes clear that $\xi = \nabla f(\bar{x})$. 🤔

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

Figure 21 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 a section in one of the x dimensions). The subdifferential set contains an infinite number of subgradients for the points in which the function is not differentiable. At points where the function is differentiable (any mid-segment point), the subgradient is unique (a gradient), and the subdifferential is a singleton.

If $f : S \mapsto \mathbb{R}$ is a convex differentiable function, then Theorem 3.7 can be combined with Lemma 3.9 to express one of the most powerful results relating f and its affine (first-order) approximation at \bar{x} .

Theorem 3.10. Convexity of a function II

Let $S \subseteq \mathbb{R}^n$ be a nonempty convex open set, and let $f : S \mapsto \mathbb{R}$ be differentiable on S . The function f is convex if and only if for any $\bar{x} \in S$, we have

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

The proof for this theorem follows from the proof for Theorem 3.7 to obtain the subgradient inequality and then use Lemma 3.9 to replace the subgradient with the gradient. To see how the opposite direction

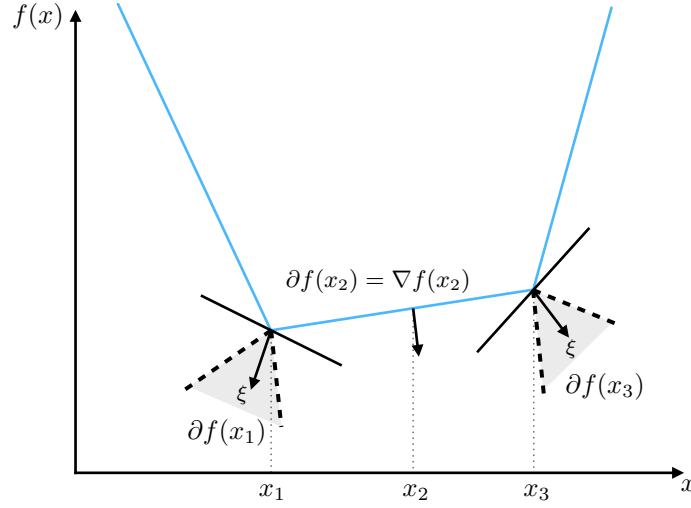


Figure 21: A representation of the subdifferential (in grey) for nondifferentiable (x_1 and x_3) and differentiable (x_2) points

(subgradient inequality holding implying the convexity of f), one should proceed as follows.

1. Take x_1 and x_2 from S . The convexity of S implies that $\lambda x_1 + (1 - \lambda)x_2$ is also in S .
2. Assume that the subgradient exists, and therefore, the two relations hold:

$$f(x_1) \geq f(\lambda x_1 + (1 - \lambda)x_2) + (1 - \lambda)\xi^\top(x_1 - x_2) \quad (38)$$

$$f(x_2) \geq f(\lambda x_1 + (1 - \lambda)x_2) + \lambda\xi^\top(x_2 - x_1) \quad (39)$$

3. Multiply (38) by λ , (39) by $(1 - \lambda)$, and add them together. One will then obtain

$$\lambda f(x_1) + (1 - \lambda)f(x_2) \geq f(\lambda x_1 + (1 - \lambda)x_2),$$

which implies convexity.

3.2.3 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. 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 semi-definiteness (positive definiteness).

Let $f_{ij}(\bar{x}) = \frac{\partial^2 f(\bar{x})}{\partial x_i \partial x_j}$. Recall that the Hessian matrix $H(\bar{x})$ at \bar{x} is given by

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

Second-order differentiability can be defined as follows.

Definition 3.11. Second-order differentiability

Let $S \subseteq \mathbb{R}^n$ be a nonempty set, and let $f : S \mapsto \mathbb{R}$. Then f is twice differentiable at $\bar{x} \in \mathbf{int}(S)$ if there exists a vector $\nabla f(\bar{x}) \in \mathbb{R}^n$, an $n \times n$ symmetric matrix $H(\bar{x})$ (the Hessian), and a function $\alpha : \mathbb{R}^n \mapsto \mathbb{R}$ such that

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

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

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

Theorem 3.12. L

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

Proof. Suppose f is convex and let $\bar{x} \in S$. Since S is open, $\bar{x} + \lambda x \in S$ for a small enough $|\lambda| \neq 0$. From Theorem 3.10 and twice differentiability of f , we have:

$$f(\bar{x} + \lambda x) \geq f(\bar{x}) + \lambda \nabla f(\bar{x})^\top x \quad (40)$$

$$f(\bar{x} + \lambda x) = f(\bar{x}) + \lambda \nabla f(\bar{x})^\top x + \frac{1}{2} \lambda^2 x^\top H(\bar{x}) x + \lambda^2 \|x\|^2 \alpha(\bar{x}; \lambda x) \quad (41)$$

Subtracting (40) from (41), we get $\frac{1}{2} \lambda^2 x^\top H(\bar{x}) x + \lambda^2 \|x\|^2 \alpha(\bar{x}; \lambda x) \geq 0$. Dividing by $\lambda^2 > 0$ and letting $\lambda \rightarrow 0$, it follows that $x^\top H(\bar{x}) x \geq 0$.

Conversely, assume that $H(\bar{x})$ is PSD for all $\bar{x} \in S$. Using the mean value theorem and second-order expansion, one can show that:

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

where $\hat{x} = \lambda \bar{x} + (1 - \lambda)x$ for $\lambda \in (0, 1)$. Note that $\hat{x} \in S$ and $H(\hat{x})$ are positive semidefinite by assumption. Thus $(x - \bar{x})^\top H(\hat{x}) (x - \bar{x}) \geq 0$, implying $f(x) = f(\bar{x}) + \nabla f(\bar{x})^\top (x - \bar{x}) \geq 0$. 😊

The proof uses a trick we have seen before. First, we assume convexity and use the definition of convexity provided by Theorem 3.10 combined with an alternative definition for $(x - \bar{x})$ to show that $x^\top H(\bar{x}) x \geq 0$. Instead of using the reference points x and \bar{x} , we incorporate a step size λ from \bar{x} in the direction of x .

To show the other direction of implication, that is, that $x^\top H(\bar{x}) x \geq 0$ implies convexity, we use the **mean value theorem**. The mean value theorem states that a point \hat{x} must exist between x and \bar{x} for which the second-order approximation is exact. We can derive the definition of convexity from these, as in Theorem 3.10.

Checking for positive semi-definiteness can be done efficiently using the appropriate computational algebra method, though it can be computationally expensive. It involves calculating the eigenvalues of $H(\bar{x})$ and testing whether they are all nonnegative (positive), which implies the positive semi-definiteness (definiteness) of $H(\bar{x})$. Some nonlinear solvers can return warning messages (or errors even), pointing out a lack of convexity by testing (under a certain threshold) for positive semi-definiteness.

3.3 Quasiconvexity

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

Definition 3.13

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

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \max\{f(x_1), f(x_2)\}. \quad (42)$$

If f is quasiconvex, then $-f$ is quasiconcave. Also, functions that are both quasiconvex and quasiconcave are called **quasilinear**. Quasiconvex functions are also called **unimodal**.

Figure 22 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. On the other hand, the epigraph $\text{epi}(f)$ is not a convex set.

Examples of quasiconvex functions include:

1. $f(x) = \sqrt{|x|}$ in \mathbb{R}

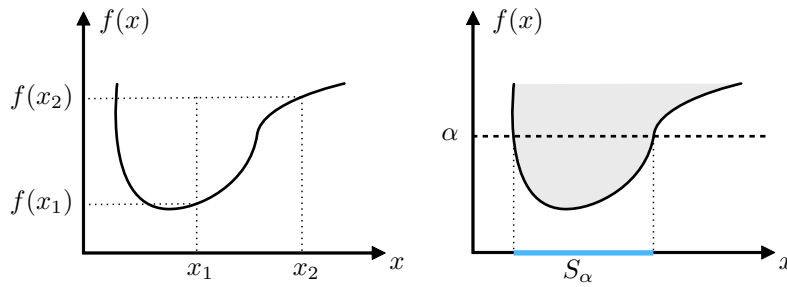


Figure 22: A quasiconvex function with its epigraph (in grey) and lower level set (in blue).

2. $f(x) = \log x$ is quasilinear for $x > 0$
 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 $S = \{(x_1, x_2) \in \mathbb{R}^2 : x_1, x_2 > 0\}$
- An important property of quasiconvex functions is that their level sets are convex.

Theorem 3.14

Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set and $f : S \mapsto \mathbb{R}$. The 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 $\alpha \in \mathbb{R}$. 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 the 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 3.15

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

Surface plot and level curves for $f(x) = \sqrt{\|x\|_1}$ in Fig. 23 and Fig. 24.

The condition in Theorem 3.15 is, in fact, sufficient for global optimality if one can show that f is in fact **strictly quasiconvex**, that is when (42) holds strictly. Figure 23 and 24 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 are a subset of a more general class of functions named **pseudoconvex**, for which the conditions in Theorem 3.15 is sufficient for global optimality.

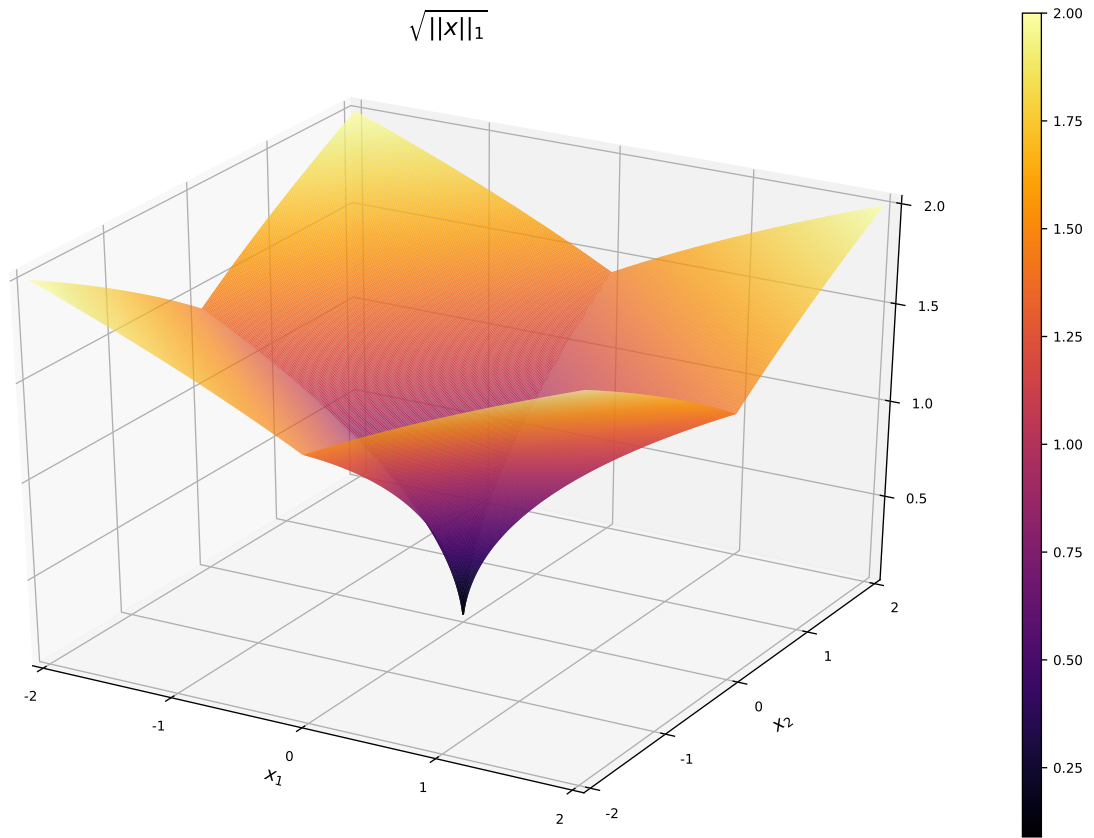


Figure 23: surface plot

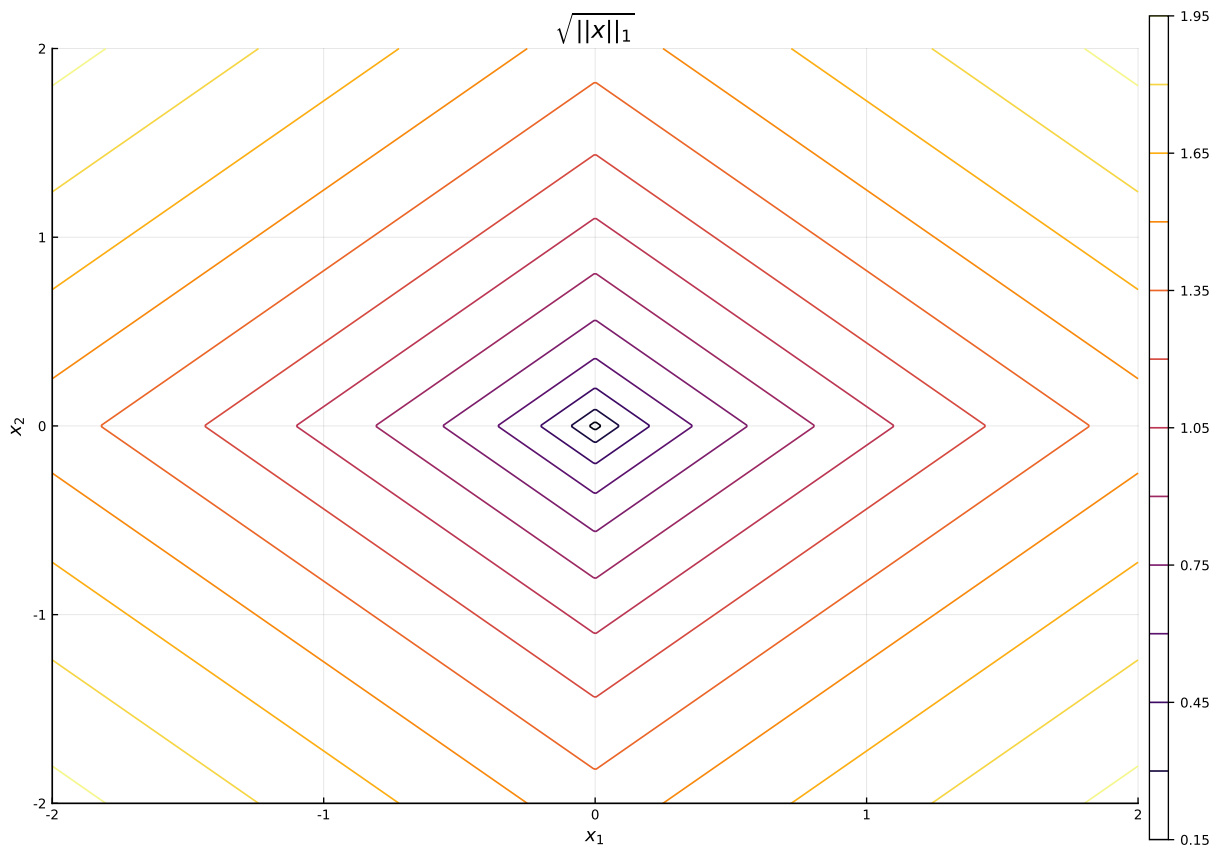


Figure 24: level curves

4 Week IV

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.

4.1 Recognising optimality

We now turn our focus to recognising whether a given point satisfy 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 $f : \mathbb{R}^n \mapsto \mathbb{R}$. Consider the problem $(P) : \min. \{f(x) : x \in S\}$.

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

Figure 25 illustrates the concepts above. Solution x_1 is an unconstrained global minimum, but is not a feasible solution considering the feasibility set S . Solution x_2 is a local optima, for any neighbourhood $N_\epsilon(x_2)$ only encompassing the points within the same plateau. Solution x_3 is a local optimum, while x_4 is neither a local or a global optimum in the unconstrained case, but is a global maximum in the constrained case. Finally, x_5 is the global minimum in the constrained case.

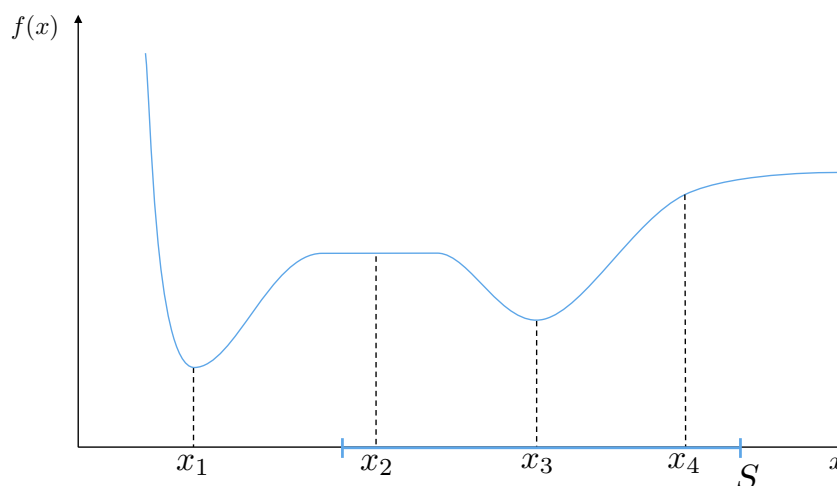


Figure 25: Examples of local and global optima

In the figures below, we observed the 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.

4.2 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.

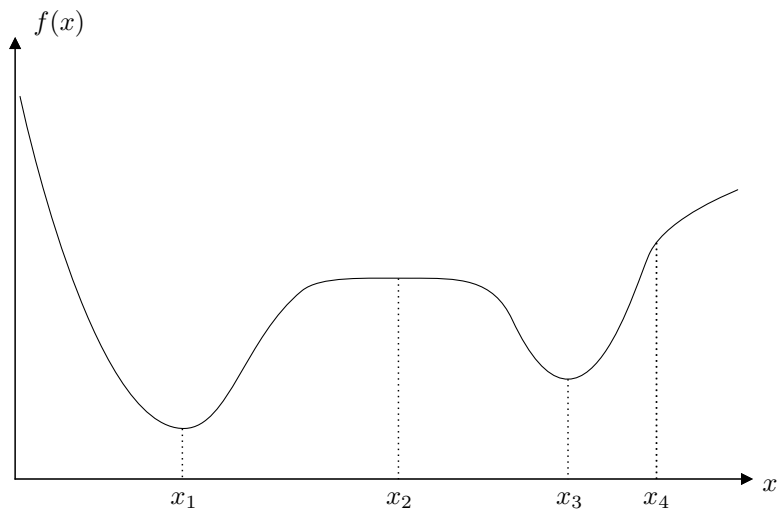


Figure 26: Unconstrained optimisation problem

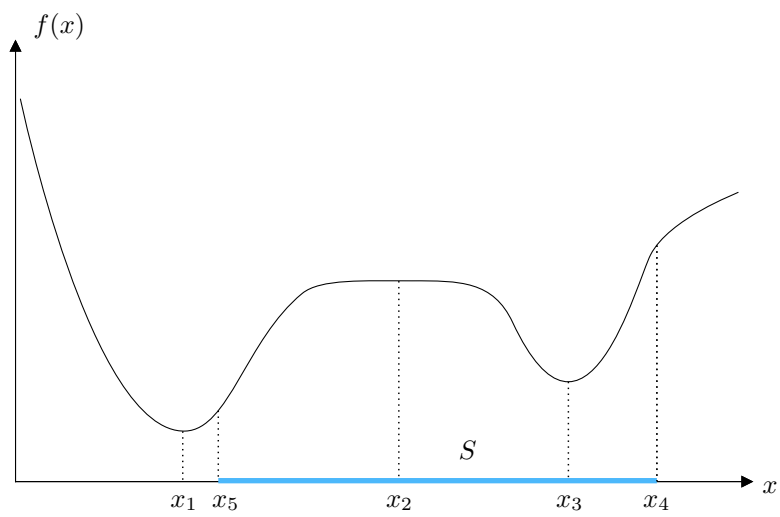


Figure 27: Constrained optimisation problem

Theorem 4.1. Global optimality of convex problems

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

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

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

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

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 the convex combination between hypothetical better solution \hat{x} and \bar{x} would have to be both in $N_\epsilon(\bar{x})$ and better than \bar{x} , contradicting the local optimality of \bar{x} .

4.3 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 4.2. Optimality condition for convex problems

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

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

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

and hence \bar{x} is optimal.

Conversely, suppose that \bar{x} is a global optimal for P . Construct the sets:

$$\Lambda_1 = \{(x - \bar{x}, y) : x \in \mathbb{R}^n, y > f(x) - f(\bar{x})\}$$

$$\Lambda_2 = \{(x - \bar{x}, y) : x \in S, y \leq 0\}$$

Note that Λ_1 and Λ_2 are convex. By optimality of \bar{x} , $\Lambda_1 \cap \Lambda_2 = \emptyset$. Using the *separation theorem*, there exists a hyperplane defined by $(\xi_0, \mu) \neq 0$ and α that separates Λ_1 and Λ_2 :

$$\xi_0^\top(x - \bar{x}) + \mu y \leq \alpha, \quad \forall x \in \mathbb{R}^n, y > f(x) - f(\bar{x}) \quad (43)$$

$$\xi_0^\top(x - \bar{x}) + \mu y \geq \alpha, \quad \forall x \in S, y \leq 0. \quad (44)$$

$$(45)$$

Letting $x = \bar{x}$ and $y = 0$ in (44), we get $\alpha \leq 0$. Next, letting $x = \bar{x}$ and $y = \epsilon > 0$ in (43), we obtain $\alpha \geq \mu\epsilon$. As this holds for any $\epsilon > 0$, we must have $\mu \leq 0$ and $\alpha \geq 0$, the latter implying $\alpha = 0$.

If $\mu = 0$, we get from (43) that $\xi_0^\top(x - \bar{x}) \leq 0$ for all $x \in \mathbb{R}^n$. Now, by letting $x = \bar{x} + \xi_0$, it follows that $\xi_0^\top(x - \bar{x}) = \|\xi_0\|^2 \leq 0$, and thus $\xi_0 = 0$. Since $(\xi_0, \mu) \neq 0$, we must have $\mu < 0$.

Dividing (43) and (44) by $-\mu$ and denoting $\xi = \frac{-\xi_0}{\mu}$, we obtain:

$$\xi^\top(x - \bar{x}) \leq y, \quad \forall x \in \mathbb{R}^n, \quad y > f(x) - f(\bar{x}) \quad (46)$$

$$\xi^\top(x - \bar{x}) \geq y, \quad \forall x \in S, \quad y \leq 0 \quad (47)$$

Letting $y = 0$ in (47), we get $\xi^\top(x - \bar{x}) \geq 0$ for all $x \in S$. From (46), we can see that $y > f(x) - f(\bar{x})$ and $y \geq \xi^\top(x - \bar{x})$. Thus, $f(x) - f(\bar{x}) \geq \xi^\top(x - \bar{x})$, which is the *subgradient inequality*. Thus, ξ is a subgradient at \bar{x} with $\xi^\top(x - \bar{x}) \geq 0$ for all $x \in S$. 😊

In the first part of the proof, we use the definition of convexity based on the subgradient inequality to show that $\xi^\top(x - \bar{x}) \geq 0$ implies that $f(\bar{x}) \leq f(x)$ for all $x \in S$. The second part of the proof uses the separation theorem in a creative way to show that the subgradient inequality must hold if \bar{x} is optimal. This is achieved by using the two sets Λ_1 and Λ_2 . Notice that, \bar{x} being optimal implies that $y > f(x) - f(\bar{x}) \geq 0$, which leads to the conclusion that $\Lambda_1 \cap \Lambda_2 = \emptyset$, demonstrating the existence of a separating hyperplane between them, as shown in (43) and (44). We can show that α in those has to be 0 by noticing that $\mu\epsilon \leq 0$ must hold for $\epsilon > 0$ to be a bounded constant.

The second part is dedicated to show that $\mu < 0$, so we can divide (43) and (44) by μ to obtain the subgradient inequality as we have seen. We show that by contradiction, since $\mu = 0$ would imply $\xi_0 = 0$, which disagrees with existence of a $(\xi, \mu) \neq 0$ in the separation theorem. Finally, as $y > f(x) - f(\bar{x})$ and $y \geq \xi^\top(x - \bar{x})$, for any given y , we have that $f(x) - f(\bar{x}) \geq \xi^\top(x - \bar{x})$ ¹, which leads to the subgradient inequality.

Notice that this result provides necessary and sufficient conditions for optimality for convex problems. These conditions can be extended to the unconstrained case as well, which is presented in Corollary 4.3.

Corollary 4.3 (Optimality in open sets). Under the conditions of Theorem 4.2, if S is open, \bar{x} is an optimal solution to P if and only if $0 \in \partial f(\bar{x})$.

Proof. From Theorem 4.2, \bar{x} is optimal if and only if ξ is a subgradient at \bar{x} with $\xi^\top(x - \bar{x}) \geq 0$ for all $x \in S$. Since S is open, $x = \bar{x} - \lambda\xi \in S$ for some $\lambda > 0$, and thus $-\lambda\|\xi\|^2 \geq 0$, implying $\xi = 0$. 😊

Notice that, if S is open, then the only way to attain the condition $\xi^\top(x - \bar{x}) \geq 0$ 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 $\nabla f(\bar{x}) = 0$, which we state below for completeness.

Corollary 4.4 (Optimality for differentiable functions). Suppose that $S \subseteq \mathbb{R}^n$ is a nonempty convex set and $f : S \rightarrow \mathbb{R}$ a differentiable convex function on S . Then $\bar{x} \in S$ is optimal if and only if $\nabla f(\bar{x})^\top(x - \bar{x}) \geq 0$ for all $x \in S$. Moreover, if S is open, then \bar{x} is optimal if and only if $\nabla f(\bar{x}) = 0$.

The proof for Corollary 4.4 is the same as Theorem 4.2 under a setting where $\partial(x) = \{\nabla f(x)\}$ due to the differentiability of f .

Let us consider two examples. First, consider the problem:

¹Notice that, on the line of nonnegative reals, for a same y , $f(x) - f(\bar{x})$ is always on the 'right side' of $\xi^\top(x - \bar{x})$ because it is an open interval.

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

Figure 28 presents a plot of the feasible region S , which is formed by the intersection of the two half-spaces, and the level curves of the objective function, with some of the values indicated in the curves. Notice that this is a convex problem.

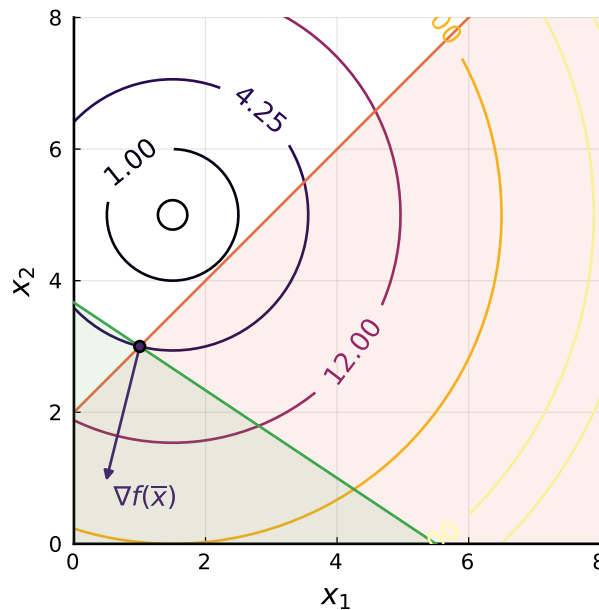


Figure 28: Example 1

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

Figure 29 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 $\bar{x} = (1, 2)$ the vector $x - \bar{x}$ forms an angle greater than 90° with the gradient of f at \bar{x} , $\nabla f(\bar{x})$, and thus the condition $\nabla f(\bar{x})^\top (x - \bar{x}) \geq 0$ does not hold for all S . The condition does hold for $\bar{x} = (2, 1)$, as can be seen in Figure 29.

A geometrical interpretation of the optimality condition $\xi^\top (x - \bar{x}) \geq 0$ is as follows. If there exists a subgradient ξ (or a gradient $\nabla f(\bar{x})$ if f is differentiable) that serves as a separating hyperplane between the level curve of f at \bar{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.

4.3.1 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 4.3 and 4.4. 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

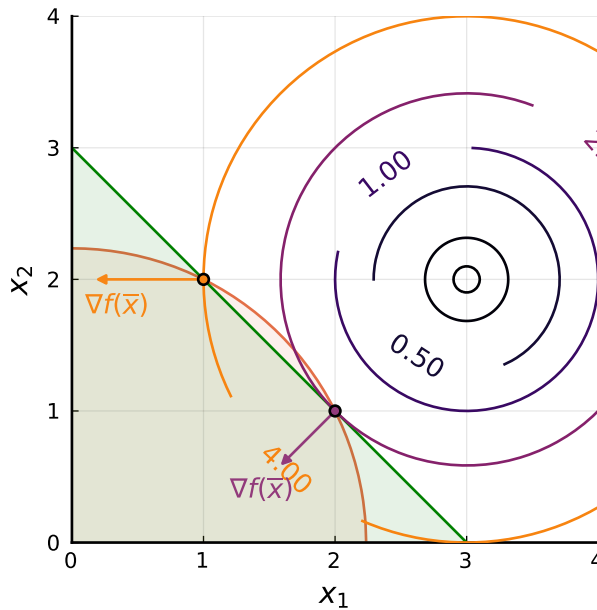


Figure 29: Example 2

state Theorem 4.2.

1. First-order optimality conditions:

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

Theorem 4.5. Descent direction

Suppose $f : \mathbb{R}^n \mapsto \mathbb{R}$ is differentiable at \bar{x} . If there is d such that $\nabla f(\bar{x})^\top d < 0$, there exists $\delta > 0$ such that $f(\bar{x} + \lambda d) < f(\bar{x})$ for each $\lambda \in (0, \delta)$, so that d is a descent direction of f at \bar{x} .

Proof. By differentiability of f at \bar{x} , we have that:

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

Since $\nabla f(\bar{x})^\top d < 0$ and $\alpha(\bar{x}; \lambda d) \rightarrow 0$ when $\lambda \rightarrow 0$ for some $\lambda \in (0, \delta)$, we must have $f(\bar{x} + \lambda d) - f(\bar{x}) < 0$. 🤔

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

We can derive the first-order optimality condition in Corollary 4.4 as a consequence from Theorem 4.5. 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 4.6 (First-order necessary condition). Suppose $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is differentiable at \bar{x} . If \bar{x} is a local minimum, then $\nabla f(\bar{x}) = 0$.

Proof. By contradiction, suppose that $\nabla f(\bar{x}) \neq 0$. Letting $d = -\nabla f(\bar{x})$, we have that $\nabla f(\bar{x})^\top d = -\|\nabla f(\bar{x})\|^2 < 0$. By Theorem 4.5, there exists a $\delta > 0$ such that $f(\bar{x} + \lambda d) < f(\bar{x})$ for all $\lambda \in (0, \delta)$, thus contradicting the local optimality of \bar{x} . 🤔

Notice that Corollary 4.6 only holds in one direction. The proof uses contradiction once again, where we assume local optimality of \bar{x} and show that having $\nabla f(\bar{x}) \neq 0$ contradicts the local optimality of \bar{x} , our initial assumption. To do that, we simply show that having any descent direction

d (we use $-\nabla f(\bar{x})$ since in this setting it is guaranteed to exist as $\nabla f(\bar{x}) \neq 0$) would mean that small step λ can reduce the objective function value, contradicting the local optimality of \bar{x} .

2. Second-order optimality conditions:

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

Theorem 4.7. Second-order necessary condition

Suppose $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is twice differentiable at \bar{x} . If \bar{x} is a local minimum, then $H(\bar{x})$ is positive semidefinite.


Proof. Take an arbitrary direction d . As f is twice differentiable, we have:

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

since \bar{x} is a local minimum, Corollary 4.6 implies that $\nabla f(\bar{x}) = 0$ and $f(\bar{x} + \lambda d) \geq f(\bar{x})$.

Rearranging terms and dividing by $\lambda^2 > 0$ we obtain

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

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

The second-order conditions can be used to attest local optimality of \bar{x} . In the case where $H(\bar{x})$ 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 $N_\epsilon(\bar{x})$.


In case f is convex, then the first-order condition $\nabla f(x) = 0$ becomes also sufficient for attesting the global optimality of \bar{x} . Recall that 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 \bar{x} .

Theorem 4.8

Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be convex. Then \bar{x} is a global minimum if and only if $\nabla f(\bar{x}) = 0$.

Proof. From Corollary 4.6, if \bar{x} is a global minimum, then $\nabla f(\bar{x}) = 0$. Now, since f is convex, we have that:

$$f(x) \geq f(\bar{x}) + \nabla f(\bar{x})^\top (x - \bar{x})$$

Notice that $\nabla f(\bar{x}) = 0$ implies that $\nabla f(\bar{x})^\top (x - \bar{x}) = 0$ for each $x \in \mathbb{R}^n$, thus implying that $f(\bar{x}) \leq f(x)$ for all $x \in \mathbb{R}^n$. 

5 Week V

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.

5.1 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  $\lambda_k$ 
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*.

5.1.1 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 $f : \mathbb{R}^n \mapsto \mathbb{R}$ is differentiable. We define the unidimensional function $\theta : \mathbb{R} \mapsto \mathbb{R}$ 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 5.1 establishes the mechanism underpinning line search methods. In that, we use the as-

sumption that the function has a unique minimum (a consequence of being strictly quasiconvex) to successively reduce the search space until the optimal is contained in a sufficiently small interval l within an acceptable tolerance.

Theorem 5.1. Line search reduction

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

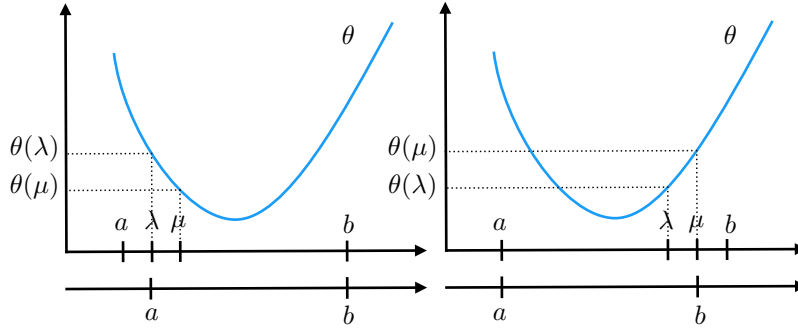


Figure 30: Applying Theorem 5.1 allows to iteratively reduce the search space.

Figure 30 provides an illustration of Theorem 5.1. 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* when optimal step sizes λ_k^* are calculated at each iteration k , and *inexact* when 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 that must be taken into account when choosing between exact and inexact line searches.

5.1.2 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 $b_k - a_k \leq l$.

1. Uniform search

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

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

From Theorem 5.1, we know that the optimal step size $\lambda^* \in [\hat{\lambda} - \delta, \hat{\lambda} + \delta]$. The process can then be repeated, by making $a = \hat{\lambda} - \delta$ and $b = \hat{\lambda} + \delta$ (see Figure 31). until $|a - b|$ is less than a prespecified tolerance l . Without enough repetition of the search, the uniform search becomes an inexact search.

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) of 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.

2. Dichotomous search

The *dichotomous search* is an example of a sequential line search method, in which evaluations of

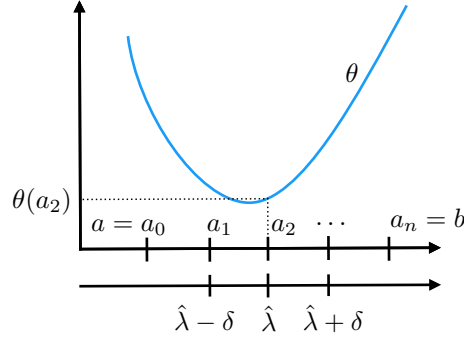


Figure 31: Grid search with 5 points; Note that $\theta(a_2) = \min_{i=0,\dots,n} \theta(a_i)$.

the function θ at a current iteration k are reused in the next iteration $k+1$ to minimise the number of function evaluations and thus improve performance.

The word dichotomous refer to the mutually exclusive parts that the search interval $[a, b]$ is divided at each iteration. We start by defining a distance margin ϵ and defining two reference points $\lambda = \frac{a+b}{2} - \epsilon$ and $\mu = \frac{a+b}{2} + \epsilon$. Using the function values $\theta(\lambda)$ and $\theta(\mu)$, we proceed as follows.

- (a) If $\theta(\lambda) < \theta(\mu)$, then *move to the left* by making $a_{k+1} = a_k$ and $b_{k+1} = \mu_k$;
- (b) Otherwise, if $\theta(\lambda) > \theta(\mu)$, then *move to the right* by making $a_{k+1} = \lambda_k$ and $b_{k+1} = b_k$.

Notice that, the assumption of strict quasiconvexity implies that $\theta(\lambda) = \theta(\mu)$ cannot occur, but in a more general setting one must make sure a criterion for resolving the tie. 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. Notice that, at any given iteration k , one can calculate what will be the size $|a_{k+1} - b_{k+1}|$, given by:

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

This is useful in that it allows predicting the number of iterations Algorithm 2 will require before convergence. Figure 32 illustrates the process for two distinct functions. Notice that the employment 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  $\bar{\lambda} = \frac{a_k + b_k}{2}$ 

```

3. Golden section search*

The *golden section search* is named after the *golden ratio* $\varphi = \frac{1+\sqrt{5}}{2}$, of which the inverse is used as the ratio of reduction for the search interval $[a, b]$ at each iteration.

Consider that, once again, we rely on two reference points λ_k and μ_k . The method is a consequence of imposing two requirements for the line search:

- (a) the reduction in the search interval should not depend on whether $\theta(\lambda_k) > \theta(\mu_k)$ or vice-versa.

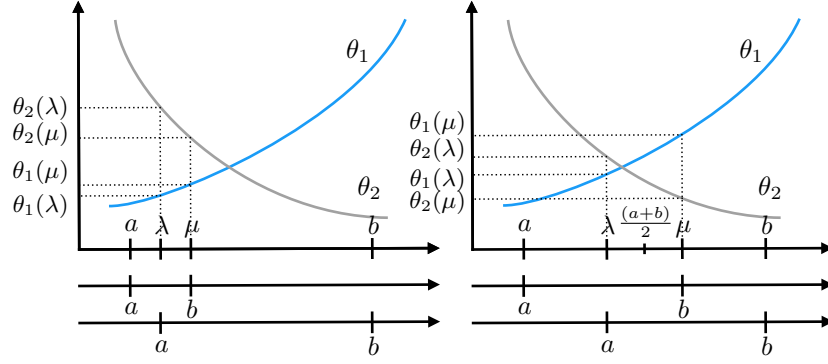


Figure 32: Using the midpoint $(a + b)/2$ and Theorem 5.1 to reduce the search space.

- (b) at each iteration, we perform a single function evaluation, thus making $\lambda_{k+1} = \mu_k$ if $\theta(\lambda_k) > \theta(\mu_k)$ or vice-versa.

From requirement 1, we can infer that $b_{k+1} - a_{k+1} = b_k - \lambda_k = \mu_k - a_k$ is required. To find the interval reduction rate $\alpha \in (0, 1)$ that would allow so, we define $\mu_k = a_k + \alpha(b_k - a_k)$ and, consequently, $\lambda_k = a_k + (1 - \alpha)(b_k - a_k)$. Notice that this makes $b_{k+1} - a_{k+1} = \alpha(b_k - a_k)$.

Notice the following. Suppose that $\theta(\lambda_k) > \theta(\mu_k)$ at iteration k . We then make $a_{k+1} = \lambda_k$ and $b_{k+1} = b_k$, a "movement to the right". From requirement 2, we also make $\lambda_{k+1} = \mu_k$ so that $\theta(\lambda_{k+1}) = \theta(\mu_k)$, avoiding a function evaluation.

From the above, we can calculate the ratio α that would allow the method to work. Notice that:

$$\begin{aligned}
 \lambda_{k+1} &= \mu_k \\
 a_{k+1} + (1 - \alpha)(b_{k+1} - a_{k+1}) &= \mu_k \\
 (1 - \alpha)[\alpha(b_k - a_k)] &= \mu_k - \lambda_k \\
 (\alpha - \alpha^2)(b_k - a_k) &= a_k + \alpha(b_k - a_k) - [a_k + (1 - \alpha)(b_k - a_k)] \\
 \alpha^2 + \alpha - 1 &= 0
 \end{aligned}$$

to which $\alpha = \frac{2}{1 + \sqrt{5}} = 0.618\dots = \frac{1}{\varphi}$ is the positive solution. Clearly, the same result is obtained if one consider $\theta(\lambda_k) < \theta(\mu_k)$. 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

```

1: initialise. tolerance  $l > 0$ ,  $[a_0, b_0] = [a, b]$ ,  $\alpha = 0.618$ ,  $k = 0$ 
2:  $\lambda_k = a_k + (1 - \alpha)(b_k - a_k)$ ,  $\mu_k = a_k + \alpha(b_k - a_k)$ 
3: while  $b_k - a_k > l$  do
4:   if  $\theta(\lambda_k) > \theta(\mu_k)$  then
5:      $a_{k+1} = \lambda_k$ ,  $b_{k+1} = b_k$ ,  $\lambda_{k+1} = \mu_k$ , and
6:      $\mu_{k+1} = a_{k+1} + \alpha(b_{k+1} - a_{k+1})$ . Calculate  $\theta(\mu_{k+1})$ 
7:   else
8:      $a_{k+1} = a_k$ ,  $b_{k+1} = \mu_k$ ,  $\mu_{k+1} = \lambda_k$ , and
9:      $\lambda_{k+1} = a_{k+1} + (1 - \alpha)(b_{k+1} - a_{k+1})$ . Calculate  $\theta(\lambda_{k+1})$ 
10:  end if
11:   $k \leftarrow k + 1$ 
12: end while
13: return  $\bar{\lambda} = \frac{a_k + b_k}{2}$ 

```

Comparing the above method for a given accuracy l , the required number of function evaluations is:

$$\min \left\{ \begin{array}{l} \text{uniform: } n \geq \frac{b_1 - a_1}{l/2} - 1 \\ \text{dichotomous: } (1/2)^{n/2} \leq \frac{l}{b_1 - a_1} \\ \text{golden section: } (0.618)^{n-1} \leq \frac{l}{b_1 - a_1} \end{array} \right\}$$

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.

4. Bisection search

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

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

- (a) 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 $[a_{k+1}, b_{k+1}] = [a_k, \lambda_k]$.
- (b) 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 $[a_{k+1}, b_{k+1}] = [\lambda_k, b_k]$.

As in the dichotomous search, we set $\lambda_k = \frac{1}{2}(b_k + a_k)$, which provides robust guarantees of 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

```

1: initialise. tolerance  $l > 0$ ,  $[a_0, b_0] = [a, b]$ ,  $k = 0$ 
2: while  $b_k - a_k > l$  do
3:    $\lambda_k = \frac{(b_k + a_k)}{2}$  and evaluate  $\theta'(\lambda_k)$ 
4:   if  $\theta'(\lambda_k) = 0$  then return  $\lambda_k$ 
5:   else if  $\theta'(\lambda_k) > 0$  then
6:      $a_{k+1} = a_k$ ,  $b_{k+1} = \lambda_k$ 
7:   else
8:      $a_{k+1} = \lambda_k$ ,  $b_{k+1} = b_k$ 
9:   end if
10:   $k \leftarrow k + 1$ 
11: end while
12: return  $\bar{\lambda} = \frac{a_k + b_k}{2}$ 

```

5.1.3 Inexact line search

Often, it is worth sacrificing 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 are related to how the directions d_k are defined in Algorithm 1. Next, we present the *Armijo rule*, arguably the most used technique to obtain step sizes in efficient implementations of optimisation methods.

1. Armijo rule

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

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

One way of understanding the Armijo rule is to look at what it means in terms of the function $\theta(\lambda) = f(x + \lambda d)$. Notice that, at $\lambda = 0$, the Armijo rule becomes:

$$\theta(\bar{\lambda}) - \theta(0) \leq \alpha \bar{\lambda} \theta'(0)$$

$$\theta(\bar{\lambda}) \leq \theta(0) + \alpha \bar{\lambda} \theta'(0). \tag{48}$$

$$\tag{49}$$

That is, $\theta(\bar{\lambda})$ has to be less than the deflected linear extrapolation of θ at $\lambda = 0$. The deflection is given by the pre-specified parameter α . In case $\bar{\lambda}$ does not satisfy the test in (48), $\bar{\lambda}$ is reduced by a factor $\beta \in (0, 1)$ until the test in (48) is satisfied.

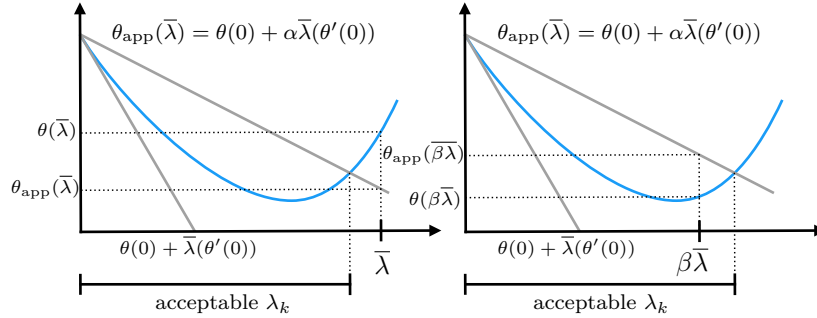


Figure 33: At first $\lambda_0 = \bar{\lambda}$ is not acceptable; after reducing the step size to $\lambda_1 = \beta\bar{\lambda}$, it enters the acceptable range where $\theta(\lambda_k) \leq \theta_{\text{app}}(\lambda_k) = \theta(0) + \alpha\lambda_k\theta'(0)$.

In Figure 33, we can see the acceptable region for the Armijo test. At first, $\bar{\lambda}$ does not satisfy the condition (48), being then reduced to $\beta\bar{\lambda}$, which, in turn, satisfies (48). In this case, λ_k would have been set $\beta\bar{\lambda}$. Suitable values for α are within $(0, 0.5]$ and for β are within $(0, 1)$, trading off 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 $\theta(\delta\bar{\lambda}) \geq \theta(0) + \alpha\delta\bar{\lambda}\theta'(0)$, with $\delta > 1$.

A possible general pseudocode for the Armijo's rule heuristic is given in 5.

Algorithm 5 Armijo's rule heuristic

- 1: **initialise.** $\lambda = 1, \alpha, \beta, k = 0$
 - 2: **while** $\theta(\lambda) > \theta(0) + \alpha\lambda\theta'(0)$ **do**
 - 3: $\lambda = \lambda * \beta$
 - 4: $k \leftarrow k + 1$
 - 5: **end while**
 - 6: **return** λ
-

5.2 Unconstrained optimisation methods

We now focus on developing methods that can be employed to optimise $f : \mathbb{R}^n \mapsto \mathbb{R}$. We start with coordinate descent method, which is derivative free, to 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 searches to define optimal step sizes, which can be any of the methods seen before or any other unidimensional optimisation method.

5.2.1 Coordinate descent

The *coordinate descent method* relies on a simple yet powerful idea. By focusing on one coordinate at the time, the method trivially derives directions d having $d_i = 1$ for coordinate i and $d_{j \neq i} = 0$ otherwise. 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 order $1, \dots, n$;

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 6 summarises the general structure of the coordinate descent method. Notice that the for-loop starting in Line 3 uses the cyclic variant of the coordinate descent method.

Algorithm 6 Coordinate descent method (cyclic)

```

1: initialise. tolerance  $\epsilon > 0$ , initial point  $x^0$ , iteration count  $k = 0$ 
2: while  $\|x^{k+1} - x^k\| > \epsilon$  do
3:   for  $j = 1, \dots, n$  do
4:      $d = \{d_i = 1, \text{ if } i = j; d_i = 0, \text{ if } i \neq j\}$ 
5:      $\bar{\lambda}_j = \operatorname{argmin}_{\lambda \in \mathbb{R}} \{f(x_j^k + \lambda d_j)\}$ 
6:      $x_j^{k+1} = x_j^k + \bar{\lambda}_j d_j$ 
7:   end for
8:    $k = k + 1$ 
9: end while
10: return  $x^k$ 

```

Figure 34 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.

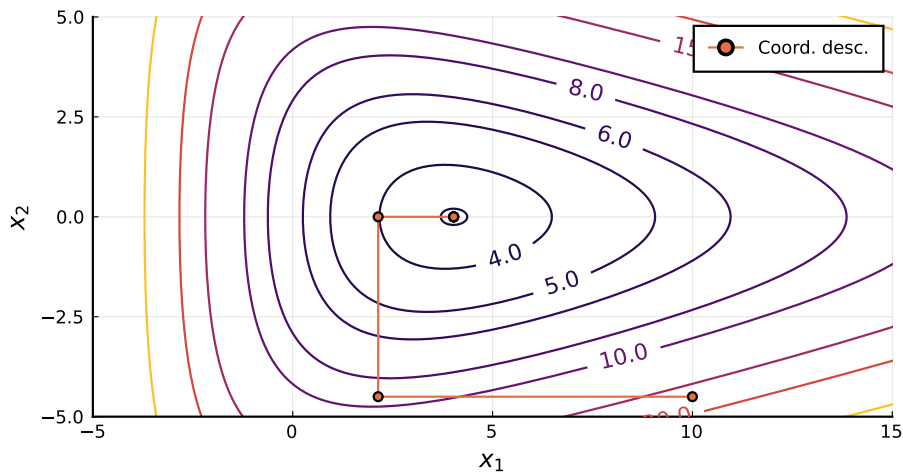


Figure 34: Coordinate descent method applied to f . Convergence is observed in 4 steps for a tolerance $\epsilon = 10^{-5}$

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.

5.2.2 Gradient (descent) method

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

Lemma 5.2. S Suppose that $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is differentiable at $x \in \mathbb{R}^n$ and $\nabla f(x) \neq 0$. Then $\bar{d} = -\frac{\nabla f(x)}{\|\nabla f(x)\|}$ is the direction of steepest descent of f at x .

Proof. From differentiability of f , we have:

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

Thus, $\bar{d} = \operatorname{argmin}_{\|d\| \leq 1} \{\nabla f(x)^\top d\} = -\frac{\nabla f(x)}{\|\nabla f(x)\|}$



In the proof, we use the differentiability to define a directional derivative for f at direction d , that is, the change in the value of f by a move of size $\lambda > 0$ in the direction d , which is given by $\nabla f(x)^\top d$. If we minimise this term in d for $\|d\|_2 \leq 1$, we observe that d is a vector of length one that has the opposite direction of $\nabla f(x)$, thus $d = -\frac{\nabla f(x)}{\|\nabla f(x)\|}$.

That provides us with the insight that we can use $\nabla f(\bar{x})$ to derive (potentially good) directions for optimising f . Notice that the direction employed is the opposite direction of the gradient for minimisation problems, being the opposite in case of maximisation. That is the reason why the gradient method is called the *steepest descent* method in some references, though gradient and steepest descent might refer to different methods in specific contexts.

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

The gradient method has many known variants that try to mitigate issues associated with the poor convergence caused by the natural 'zigzagging' behaviour of the algorithm (see, for example 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 $i = 1, \dots, n$ forming *blocks* of coordinates at each iteration. If these blocks are randomly formed, these methods are known as *stochastic gradient* methods.

In Algorithm 7 we provide a pseudocode for the gradient method. In Line 2, the stopping condition for the while-loop is equivalent of testing $\nabla f(\bar{x}) = 0$ for a tolerance ϵ .

Algorithm 7 Gradient method

```

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

```

Figure 35 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 overshoots slightly some of the steps, taking a few more iterations to converge.

5.2.3 Newton's method

One can think of gradient methods as using first-order information to derive directions of improvement, while *Newton's method* consists of a step forward also incorporating 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 idea of the Newton's method is the following. Consider the second-order approximation of f at x_k , which is given by:

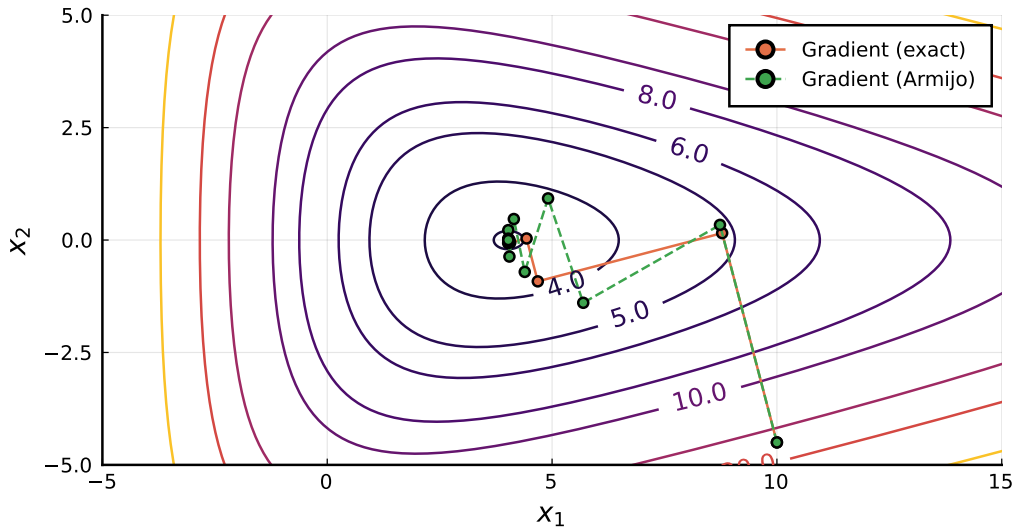


Figure 35: Gradient method applied to f . Convergence is observed in 10 steps using exact line search and 19 using Armijo’s rule (for $\epsilon = 10^{-5}$)

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

The method uses as direction d that of the extremum of the quadratic approximation at x_k , which can be obtained from the first-order condition $\nabla q(x) = 0$. This renders:

$$\nabla q(x) = \nabla f(x_k) + H(x_k)(x - x_k) = 0. \quad (50)$$

Assuming that $H^{-1}(x_k)$ exists, we can use (50) to obtain the following update rule, which is known as the *Newton step*:

$$x_{k+1} = x_k - H^{-1}(x_k) \nabla f(x_k) \quad (52)$$

Notice that the “pure” Newton’s method has embedded in the direction of the step, its length (i.e., the step size) as well. In practice, the method uses $d = -H^{-1}(x_k) \nabla f(x_k)$ as a direction combined with a line search to obtain optimal step sizes and prevent divergence (that is, converge to $-\infty$) in cases where the second-order approximation might lead to divergence. Fixing $\lambda = 1$ renders the natural Newton’s method, as derived in (52). The Newton’s method can also be seen as employing Newton-Raphson method to solve the system of equations that describe the first order conditions of the quadratic approximation at x_k .

Figure 36 shows the calculation of 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 the that of the minimum of the quadratic approximation $q(x)$ at x_k . The employment of a line search allows for overshooting the exact minimum, making the search more efficient.

The Newton’s method might diverge if the initial point is too far from the optimal and fixed step sizes (such as $\lambda = 1$) are used, since the quadratic approximation minimum and the actual function minimum 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 of a criteria for step-size acceptance that require improvement (such as employing the Armijo rule for defining the step sizes) is often sufficient for guaranteed convergence. Figure 37 compares the convergence of the pure Newton’s method and the method employing an exact line search.

Algorithm 8 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 $H(x_k)$ tends to become singular, potentially causing numerical instabilities.

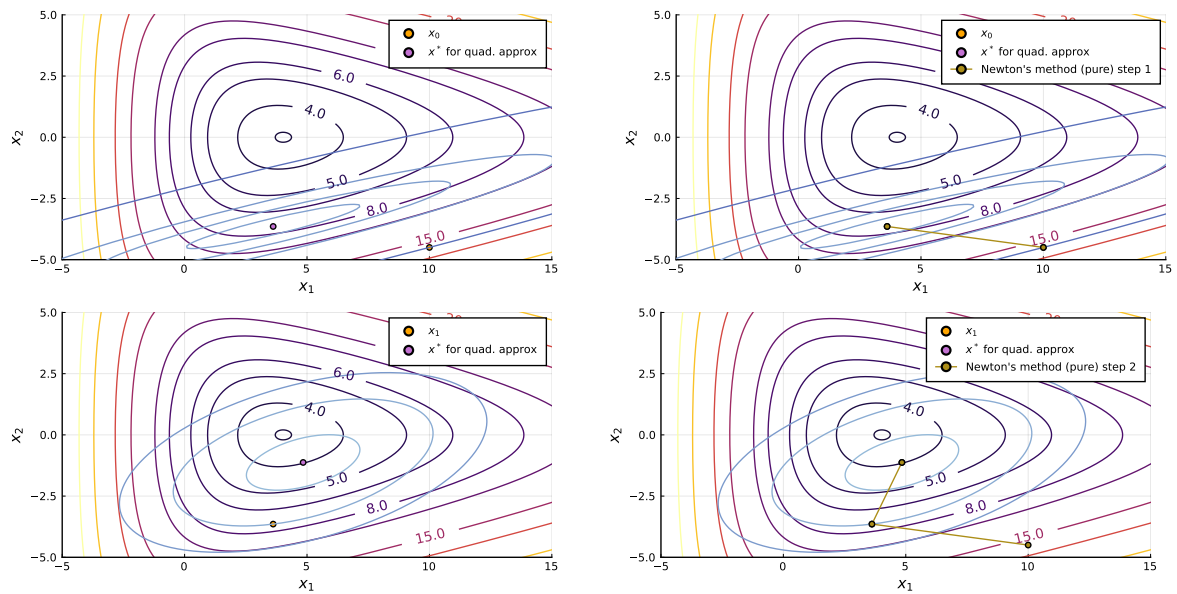


Figure 36: The calculation of the direction $d = x^* - x_0$ in the first two iterations of the Newton's method with step size λ fixed to 1 (the pure Newton's method, in left to right, top to bottom order). Notice in blue the level curves of the quadratic approximation of the function at the current point x_k and how it improves from one iteration to the next.

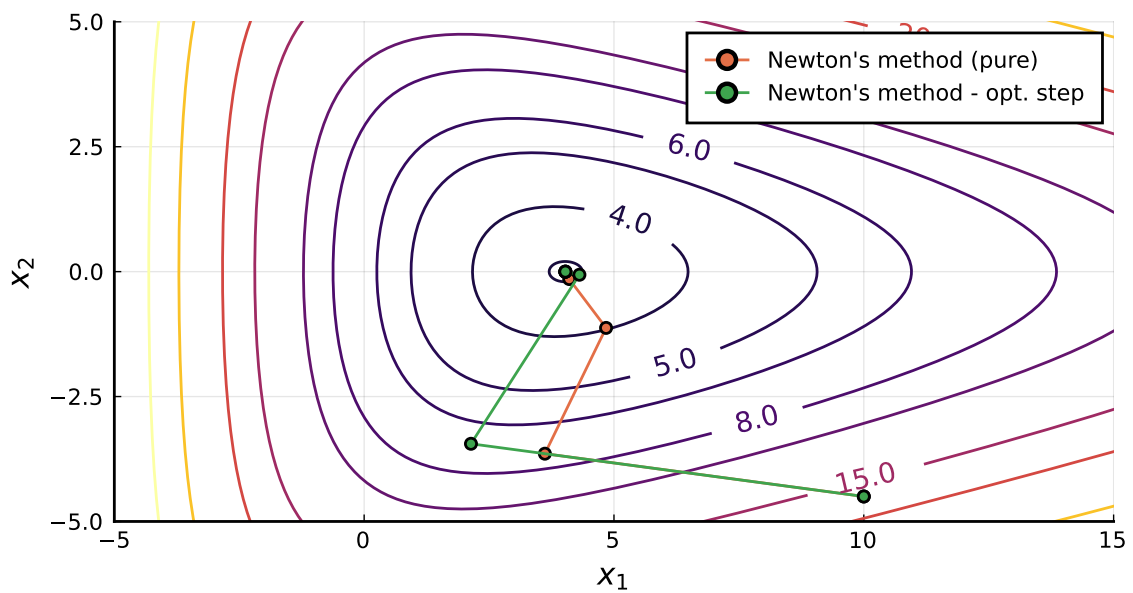


Figure 37: A comparison of the trajectory of both Newton's method variants. Notice that in the method using the exact line search, while the direction $d = x^* - x_0$ is utilised, the step size is larger in the first iteration.

Algorithm 8 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: $d = -H^{-1}(x_k)\nabla f(x_k)$
 - 4: $\bar{\lambda} = \operatorname{argmin}_{\lambda \in \mathbb{R}} \{f(x_k + \lambda d)\}$
 - 5: $x_{k+1} = x_k + \bar{\lambda}d$
 - 6: $k = k + 1$
 - 7: **end while**
 - 8: **return** x_k
-

Figure 38 shows the progression of the Newton's method for f with exact and inexact line searches.

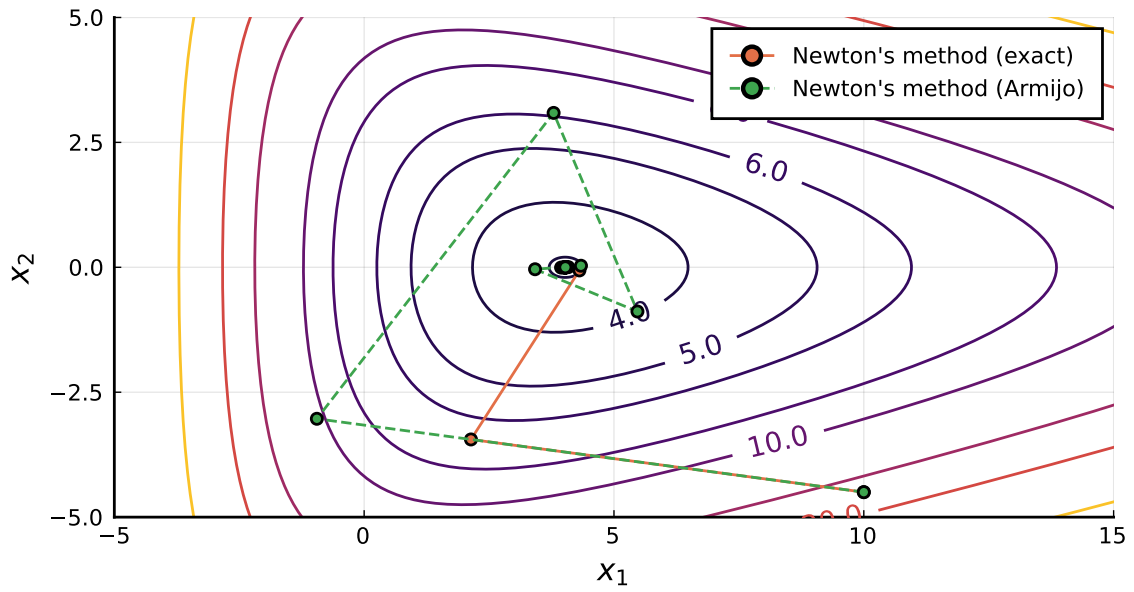


Figure 38: Newton's method applied to f . Convergence is observed in 4 steps using exact line search and 27 using Armijo's rule ($\epsilon = 10^{-5}$)

6 Week VI

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.

6.1 Unconstrained optimisation methods

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

6.1.1 Conjugate gradient method

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

1. The concept of conjugacy

Let us first define the concept of conjugacy.

Definition 6.1

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

Notice that H -conjugacy (or simply conjugacy) is a generalisation of orthogonality under the linear transformation imposed by the matrix H . Notice that orthogonal vectors are H -conjugate for $H = I$. Figure 39 illustrate the notion of conjugacy between two vectors d_1 and d_2 that are H -conjugate, being H the Hessian of the underlying quadratic function. Notice how it allows one to generate, from direction d_1 , a direction d_2 that, if used in combination with an exact line search, would take us to the centre of the curve.

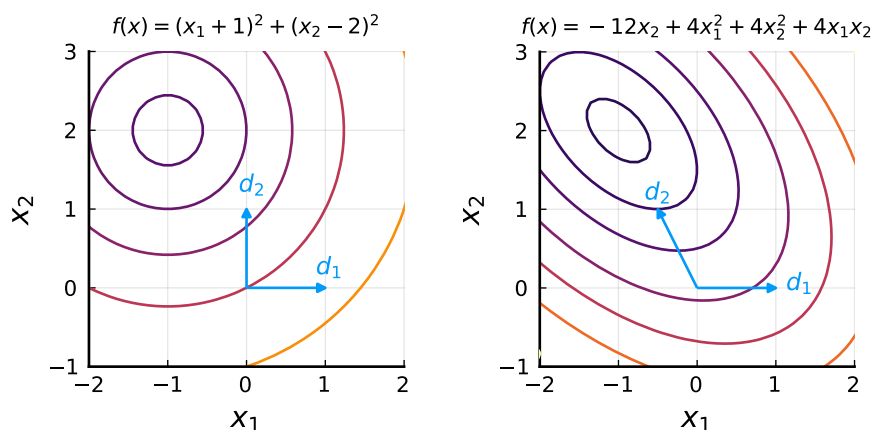


Figure 39: d_1 and d_2 are H -conjugates; on the left, $H = I$.

One can use H -conjugate directions to find optimal solutions for the quadratic function $f(x) = c^\top x + \frac{1}{2}x^\top Hx$, where H is a symmetric matrix. Suppose we know directions d_1, \dots, d_n that are H -conjugate. Then, given an initial point x_0 , any point x can be described as $x = x_0 + \sum_{j=1}^n \lambda_j d_j$.

We can then reformulate $f(x)$ as a function of the step size λ , i.e.:

$$\begin{aligned} f(x) &= F(\lambda) = c^\top(x_0 + \sum_{j=1}^n \lambda_j d_j) + \frac{1}{2}(x_0 + \sum_{j=1}^n \lambda_j d_j)^\top H(x_0 + \sum_{j=1}^n \lambda_j d_j) \\ &= \sum_{j=1}^n [c^\top(x_0 + \lambda_j d_j) + \frac{1}{2}(x_0 + \lambda_j d_j)^\top H(x_0 + \lambda_j d_j)]. \end{aligned}$$

This reformulation exposes an important properties that having conjugate directions d_1, \dots, d_n allows us to explore: separability. Notice that $F(\lambda) = \sum_{j=1}^n F_j(\lambda_j)$, where $F_j(\lambda_j)$ is given by:

$$F_j(\lambda_j) = c^\top(x_0 + \lambda_j d_j) + \frac{1}{2}(x_0 + \lambda_j d_j)^\top H(x_0 + \lambda_j d_j),$$

and is, ultimately, a consequence of the linear independence of the conjugate directions. Assuming that H is positive definite, and thus that first-order conditions are necessary and sufficient for optimality, we can then calculate optimal $\bar{\lambda}_j$ for $j = 1, \dots, n$ as:

$$\begin{aligned} F'_j(\lambda_j) &= 0 \\ c^\top d_j + x_0^\top H d_j + \lambda_j d_j^\top H d_j &= 0 \\ \bar{\lambda}_j &= -\frac{c^\top d_j + x_0^\top H d_j}{d_j^\top H d_j}, \text{ for all } j = 1, \dots, n. \end{aligned}$$

This result can be used to devise an iterative method that can obtain optimal solution for quadratic functions in exactly n iterations. From an initial point x_0 and a collection of H -conjugate directions d_1, \dots, d_n , the method consists of the successively executing the following step:

$$x_k = x_{k-1} + \lambda_k d_k, \text{ where } \lambda_k = -\frac{c^\top d_k + x_{k-1}^\top H d_k}{d_k^\top H d_k}$$

Notice the resemblance this method hold with the coordinate descent method. In case $H = I$, then the coordinate directions given by $d_i = 1$ and $d_{j \neq i} = 0$ are H -conjugate and thus, the coordinate descent method converges in two iterations. Figure 40 illustrates this behaviour. 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, both methods become equivalent, since, when $H = I$, the coordinate directions become also conjugate to each other.

2. Generating conjugate directions

The missing part at this point is how one can generate H -conjugate directions. This can be done efficiently using an adaptation of the *Gram-Schmidt* procedure, typically employed to generate orthonormal bases.

We intend to build a collection of conjugate directions d_0, \dots, d_{n-1} , which can be achieved provided that we have a collection of linearly independent vectors ξ_0, \dots, ξ_{n-1} .

The method proceed as follows.

- (a) First, set $d_0 = \xi_0$ as a starting step.
- (b) At a given iteration $k + 1$, we need to set the coefficients α_{k+1}^l such that d_{k+1} is H -conjugate to d_0, \dots, d_k and formed by adding ξ_{k+1} to a linear combination of d_0, \dots, d_k , that is:

$$d_{k+1} = \xi_{k+1} + \sum_{l=0}^k \alpha_{k+1}^l d_l.$$

- (c) To obtain H -conjugacy one must observe that, for each $i = 0, \dots, k$,

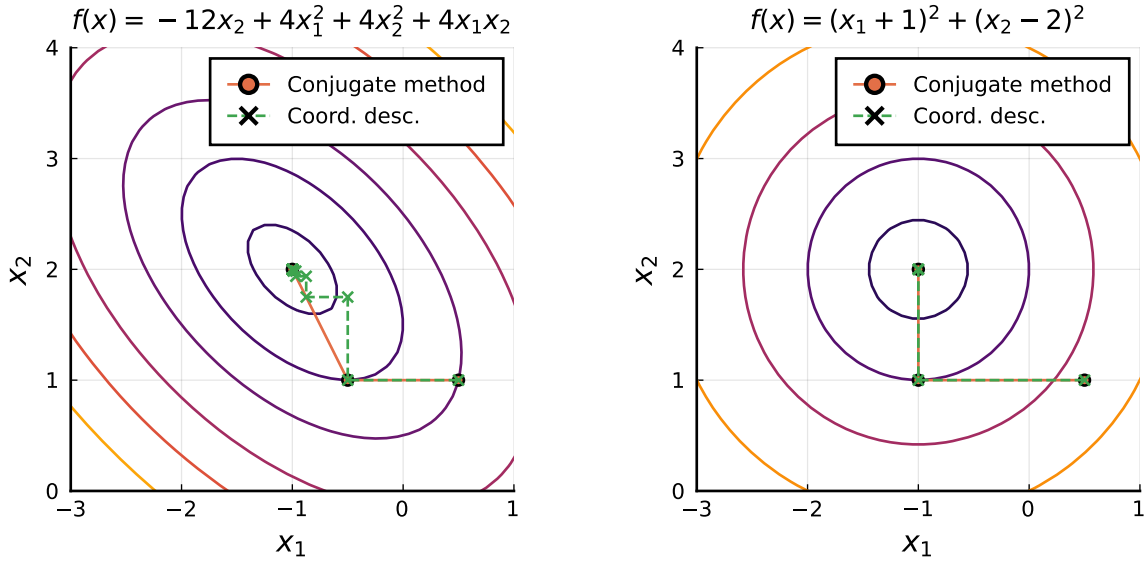


Figure 40: Optimising f with the conjugate method and coordinate descent (left). For $H = I$, both methods coincide (right)

$$d_{k+1}^\top H d_i = \xi_{k+1}^\top H d_i + \left(\sum_{l=0}^k \alpha_{k+1}^l d_l \right)^\top H d_i = 0.$$

Due to the H -conjugacy, $d_l^\top H d_k = 0$ for all $l \neq k$. Thus the value of α_{k+1} is:

$$\alpha_{k+1}^i = \frac{-\xi_{k+1}^\top H d_i}{d_i^\top H d_i}, \text{ for } i = 0, \dots, k. \quad (54)$$

$$(55)$$

3. Gradients and conjugate directions

The next piece required for developing a method that could exploit conjugacy is the definition of what collection of linearly independent vectors ξ_0, \dots, ξ_{n-1} could be used to generate conjugate directions. In the setting of developing an unconstrained optimisation method, the gradients $\nabla f(x_k)$ can play this part, which is the key result in Theorem 6.2.

Theorem 6.2

Let $f(x) = c^\top x + \frac{1}{2} x^\top H x$, where H is an $n \times n$ symmetric matrix. Let d_1, \dots, d_n be H -conjugate, and let x_0 be an arbitrary starting point. Let λ_j be the optimal solution to $F_j(\lambda_j) = f(x_0 + \lambda_j d_j)$ for all $j = 1, \dots, n$. Then, for $k = 1, \dots, n$ we must have:

- x_{k+1} is optimal to $\min. \{f(x) : x - x_0 \in L(d_1, \dots, d_k)\}$ where $L(d_1, \dots, d_k) = \{\sum_{j=1}^k \mu_j d_j : \mu_j \in \mathbb{R}, j = 1, \dots, k\}$;
- $\nabla f(x_{k+1})^\top d_j = 0$, for all $j = 1, \dots, k$;
- $\nabla f(x_0)^\top d_k = \nabla f(x_k)^\top d_k$.

The proof of this theorem is based on the idea that, for a given collection of conjugate directions d_0, \dots, d_k , x_k will be optimal in the space spanned by the conjugate directions d_0, \dots, d_k , meaning that the partial derivatives of $F(\lambda)$ for these directions is zero. This phenomena is sometimes called *the expanding manifold property*, since at each iteration $L(d_0, \dots, d_k)$ expands in one independent (conjugate) direction at the time. To verify the second point, notice that the optimality condition for $\lambda_j \in \arg \min \{F_j(\lambda_j)\}$ is $d_j^\top \nabla f(x_0 + \lambda_j d_j) = 0$.

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 .

In specific, the method operates generating a sequence of iterates:

$$x_{k+1} = x_k + \lambda_k d_k,$$

where $d_0 = -\nabla f(x_0)$. Given a current iterate x_{k+1} with $-\nabla f(x_{k+1}) \neq 0$, we use Gram-Schmidt procedure, in particular (55), to generate a conjugate direction d_{k+1} by making the linearly independent vector $\xi_{k+1} = \nabla f(x_{k+1})$. Thus, we obtain:

$$d_{k+1} = -\nabla f(x_{k+1}) + \alpha_k d_k, \text{ with } \alpha_k = \frac{\nabla f(x_{k+1})^\top H d_k}{d_k^\top H d_k}. \quad (56)$$

Notice that, since $\nabla f(x_{k+1}) - \nabla f(x_k) = H(x_{k+1} - x_k) = \lambda_k H d_k$ and $d_k = -\nabla f(x_k) + \alpha_{k-1} d_{k-1}$, α_k can be simplified to be:

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

where the last relation follows from Theorem 6.2. Algorithm 9 summarises the conjugate gradient method.

Algorithm 9 Conjugate gradient method

```

1: initialise. tolerance  $\epsilon > 0$ , initial point  $x_0$ , direction  $d_0 = -\nabla f(x_0)$ ,  $k = 1$ 
2: while  $\|\nabla f(x_k)\| > \epsilon$  do
3:    $y_0 = x_{k-1}$ 
4:    $d_0 = -\nabla f(y_0)$ 
5:   for  $j = 1, \dots, n$  do
6:      $\bar{\lambda}_j = \operatorname{argmin}_{\lambda \geq 0} f(y_{j-1} + \lambda d_{j-1})$ 
7:      $y_j = y_{j-1} + \bar{\lambda}_j d_{j-1}$ 
8:      $d_j = -\nabla f(y_j) + \alpha_j d_{j-1}$ , where  $\alpha_j = \frac{\|\nabla f(y_j)\|^2}{\|\nabla f(y_{j-1})\|^2}$ .
9:   end for
10:   $x_k = y_n$ ,  $k = k + 1$ 
11: end while
12: return  $x_k$ .

```

The conjugate gradient method using $\alpha_k = \frac{\|\nabla f(x_{k+1})\|^2}{\|\nabla f(x_k)\|^2}$ is due to Fletcher and Reeves ².

An alternative version of the method uses:

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

which is known for having better numerical properties for solving problems that are not quadratic. Figure 41 illustrates the behaviour of the conjugate gradient 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.

If $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a quadratic function, then the method is guaranteed to converge in exactly n iterations. However, the method can be applied to any differentiable function f , in which setting the method behaves as successively solving quadratic approximations of f , in a similar 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

²(Fletcher, Reeves, and Colin M. Reeves. "Function minimization by conjugate gradients." The computer journal 7.2 (1964): 149-154.)

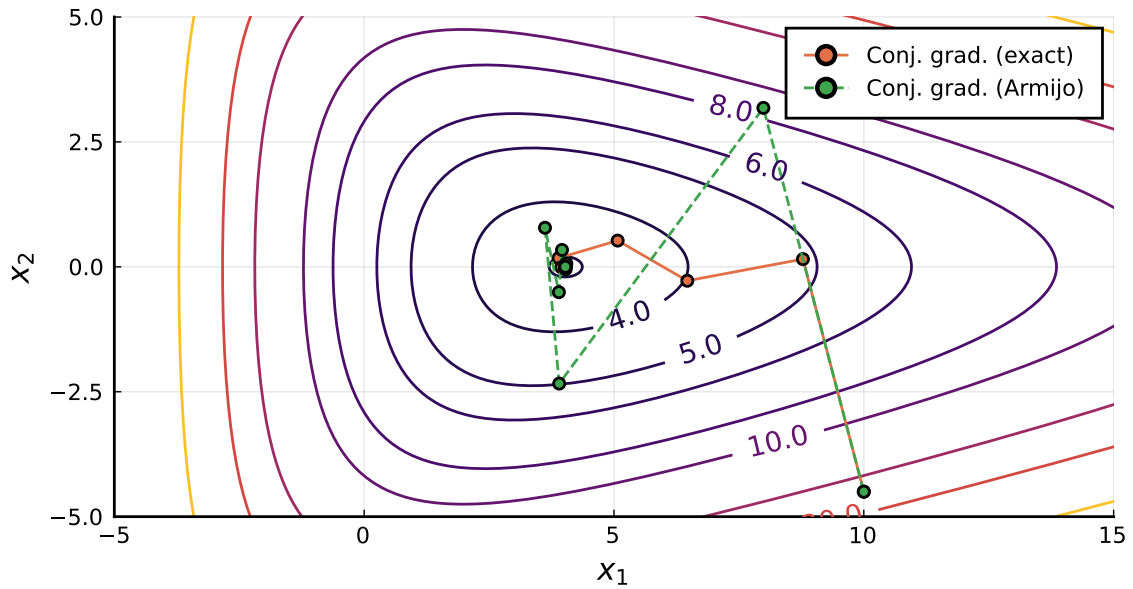


Figure 41: Conjugate gradient method applied to f . Convergence is observed in 24 steps using exact line search and 28 using Armijo's rule ($\epsilon = 10^{-6}$)

functions, the process of obtaining conjugate directions is restarted at the current point x_k after n steps (represented in the loop starting in Line 5 in Algorithm 2).

Equation (56) 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 weighting the current direction (given by the gradient) $-\nabla f(x_{k+1})$ and the previous direction $\alpha_k d_k$, which naturally compensates for the curvature encoded in the original matrix H (which is the Hessian of the quadratic approximation).

6.1.2 Quasi Newton: BFGS method

Quasi-Newton methods is a term referring to methods that use approximations for the inverse of the Hessian of f at \bar{x} , $H^{-1}(\bar{x})$, that do not explicitly require second-order information (i.e., Hessians) neither expensive inversion operations.

In quasi-Newton methods, we consider the search direction $d_k = -D_k \nabla f(x_k)$, where D_k acts as the approximation for the inverse Hessian $H^{-1}(\bar{x})$. To compute D_k , we use local curvature information, in the attempt to approximate second-order derivatives. For that, let us define the terms:

$$\begin{aligned} p_k &= \lambda_k d_k = x_{k+1} - x_k \\ q_k &= \nabla f(x_{k+1}) - \nabla f(x_k) = H(x_{k+1} - x_k) = H p_k. \end{aligned}$$

Starting from an initial guess D_0 , quasi-Newton methods progress by successively updating $D_{k+1} = D_k + C_k$, with C_k being such that it only uses the information in p_k and q_k and that, after n updates, D_n converges to H^{-1} .

For that to be the case, we require that p_j , $j = 1, \dots, k$ are eigenvectors of $D_{k+1}H$ with unit eigenvalue, that is:

$$D_{k+1}H p_j = p_j, \text{ for } j = 1, \dots, k. \quad (57)$$

This condition guarantees that, at the last iteration, $D_n = H^{-1}$. To see that, first, notice the following from (57).

$$\begin{aligned}
D_{k+1}Hp_j &= p_j, \quad j = 1, \dots, k \\
D_{k+1}q_j &= p_j, \quad j = 1, \dots, k \\
D_kq_j + C_kq_j &= p_j, \quad j = 1, \dots, k \\
p_j &= D_kHp_j + C_kq_j = p_j + C_kq_j, \quad j = 1, \dots, k-1,
\end{aligned}$$

which implies that $C_kq_j = 0$ for $j = 1, \dots, k-1$.

Now, for $j = k$, we require that:

$$\begin{aligned}
D_{k+1}q_k &= p_k \\
D_kq_k + C_kq_k &= p_k \\
(D_k + C_k)q_k &= p_k
\end{aligned}$$

This last condition allows, after n iterations, to recover:

$$D_n = [p_0, \dots, p_{n-1}][q_0, \dots, q_{n-1}]^{-1} = H(x_n) \quad (58)$$

Condition (58) is called the *secant condition* as a reference to the approximation to the second-order derivative. Another way of understanding the role this condition has is by noticing the following.

$$\begin{aligned}
D_{k+1}q_k &= p_k \\
D_{k+1}(\nabla f(x_{k+1}) - \nabla f(x_k)) &= x_{k+1} - x_k \\
\nabla f(x_{k+1}) &= \nabla f(x_k) + D_{k+1}^{-1}(x_{k+1} - x_k),
\end{aligned} \quad (59)$$

where D_{k+1}^{-1} can be seen as an approximation to the Hessian H , just as D_{k+1} is an approximation to H^{-1} . Now, consider the second-order approximation of f at x_k :

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

We can now notice the resemblance the condition (59) holds with:

$$\nabla q(x) = \nabla f(x_k) + H(x_k)^\top (x - x_k) = 0.$$

In other words, at each iteration, the updates are made such that the optimality conditions in terms of the quadratic expansion remains valid.

The *Davidon-Fletcher-Powell* (DFP) is one classical quasi-Newton method available. It employs updates of the form:

$$D_{k+1} = D_k + C^{DFP} = D_k + \frac{p_k p_k^\top}{p_k^\top q_k} - \frac{D_k q_k q_k^\top D_k}{q_k^\top D_k q_k}$$

We can verify that C^{DFP} satisfies conditions (57) and (58). For that, notice that:

$$\begin{aligned}
(1) \quad C^{DFP}q_j &= C^{DFP}Hp_j \\
&= \frac{p_k p_k^\top Hp_j}{p_k^\top q_k} - \frac{D_k q_k q_k^\top H D_k Hp_j}{q_k^\top D_k q_k} = 0, \quad \text{for } j = 1, \dots, k-1; \\
(2) \quad C^{DFP}q_k &= \frac{p_k p_k^\top q_k}{p_k^\top q_k} - \frac{D_k q_k q_k^\top D_k q_k}{q_k^\top D_k q_k} = p_k - D_k q_k.
\end{aligned}$$

In a pseudocode form, the update proposed by DFP can be summarized in the algorithm below (see 10.

Algorithm 10 Quasi-Newton (DFP) method

1: **initialise.** tolerance $\epsilon > 0$, initial point x_0 , $D_0 = I$, iteration count $k = 0$.
2: **while** $\|\nabla f(x_k)\| > \epsilon$ **do**
3: $d = -D_k \nabla f(x_k)$
4: $\bar{\lambda} = \operatorname{argmin}_{\lambda \in \mathbb{R}} \{f(x_k + \lambda d)\}$
5: $x_{k+1} = x_k + \bar{\lambda} d$
6: $p_k = \bar{\lambda} d$
7: $q_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
8: $D_{k+1} = D_k + \frac{p_k p_k^\top}{p_k^\top q_k} - \frac{D_k q_k q_k^\top D_k}{q_k^\top D_k q_k}$
9: $k = k + 1$
10: **end while**
11: **return** x_k .

The main difference between available quasi-Newton methods is the nature of the matrix C employed in the updates. Over the years, several ideas emerged in terms of generating updates that satisfied the above properties. The most widely used quasi-Newton method is the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS), which has been widely shown to have remarkable practical performance. BFGS is part of the Broyden family of updates, given by:

$$C^B = C^{DFP} + \phi \frac{\tau_j v_k v_k^\top}{p_k^\top q_k},$$

where $v_k = p_k - \left(\frac{1}{\tau_k}\right) D_k q_k$, $\tau_k = \frac{q_k^\top D_k q_k}{p_k^\top q_k}$, and $\phi \in (0, 1)$. 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 updates from the Broyden family also satisfy the quasi-Newton conditions (57) and (58). The BFGS update is obtained for $\phi = 1$, which renders:

$$C_k^{BFGS} = \frac{p_k p_k^\top}{p_k^\top q_k} \left(1 + \frac{q_k^\top D_k q_k}{p_k^\top q_k} \right) - \frac{D_k q_k p_k^\top + p_k q_k^\top D_k}{p_k^\top q_k}.$$

The BFGS method is often presented explicitly approximating the Hessian H instead of its inverse, which is useful when using specialised linear algebra packages that rely on the “backslash” operator to solve linear systems of equations. Let B_k be the current approximation of H . Then $D_{k+1} = B_{k+1}^{-1} = (B_k + \bar{C}_k^{BFGS})^{-1}$, with:

$$\bar{C}_k^{BFGS} = \frac{q_k q_k^\top}{q_k^\top p_k} - \frac{B_k p_k p_k^\top B_k}{p_k^\top B_k p_k}.$$

The update for the inverse Hessian H^{-1} can then be obtained using the *Sherman-Morrison formula*.

The Quasi-Newton algorithm with BFGS update is below (as seen in 11).

Algorithm 11 Quasi-Newton (BFGS) method

1: **initialise.** tolerance $\epsilon > 0$, initial point x_0 , $D_0 = I$, iteration count $k = 0$.
2: **while** $\|\nabla f(x_k)\| > \epsilon$ **do**
3: $d = -D_k \nabla f(x_k)$
4: $\bar{\lambda} = \operatorname{argmin}_{\lambda \in \mathbb{R}} \{f(x_k + \lambda d)\}$
5: $x_{k+1} = x_k + \bar{\lambda} d$
6: $p_k = \bar{\lambda} d$
7: $q_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
8: $D_{k+1} = D_k \frac{p_k p_k^\top}{p_k^\top q_k} \left(1 + \frac{q_k^\top D_k q_k}{p_k^\top q_k} \right) - \frac{D_k q_k p_k^\top + p_k q_k^\top D_k}{p_k^\top q_k}$
9: $k = k + 1$
10: **end while**
11: **return** x_k .

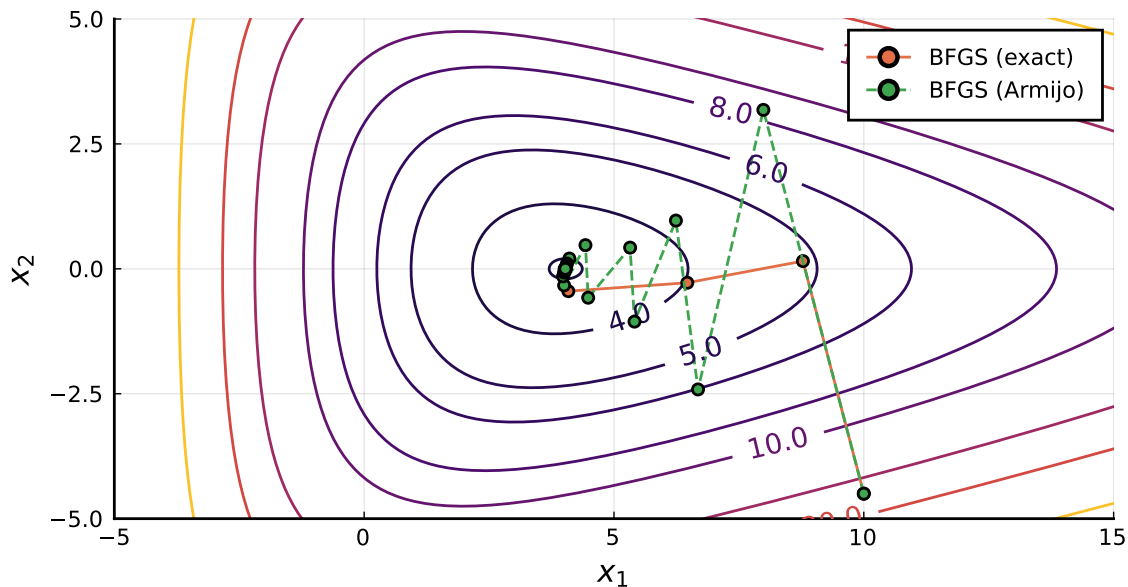


Figure 42: BFGS method applied to f . Convergence is observed in 11 steps using exact line search and 36 using Armijo's rule ($\epsilon = 10^{-6}$)

Figure 42 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 both in the calculation of H^{-1} and in the line search turns the search noisy. This combination (BFGS combined with 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.

6.2 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.

6.2.1 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 6.3. 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.

6.2.2 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 \bar{x} .

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

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

The sequence $\{e(x_k)\}$ is then compared to the geometric progression β^k , with $k = 1, 2, \dots$, 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:

$$\limsup_{k \rightarrow \infty} \frac{e(x_{k+1})}{e(x_k)} \leq \beta.$$

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:

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

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 6.4. 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 $\underline{\lambda} = \min_{i=1, \dots, n} \lambda_i$ and $\bar{\lambda} = \max_{i=1, \dots, n} \lambda_i$, where λ_i are eigenvalues of H . Then, for all k ,

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

Theorem 6.4 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 6.5 summarises these conditions.

Theorem 6.5. Convergence of Newton's method - general case

Let $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be differentiable, \bar{x} such that $g(\bar{x}) = 0$, and let $\{e(x_k)\} = \{\|x_k - \bar{x}\|\}$. Moreover, let $N_\delta(\bar{x}) = \{x : \|x - \bar{x}\| \leq \delta\}$ for some $\delta > 0$. Then

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

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

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

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

If $\frac{LM\delta}{2} < 1$ and $x_0 \in N_\delta(\bar{x})$, $\{e(x_k)\}$ 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 $N_\delta(\bar{x})$ 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 43 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 $e(x) = \|x_k - \bar{x}\|$. 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.

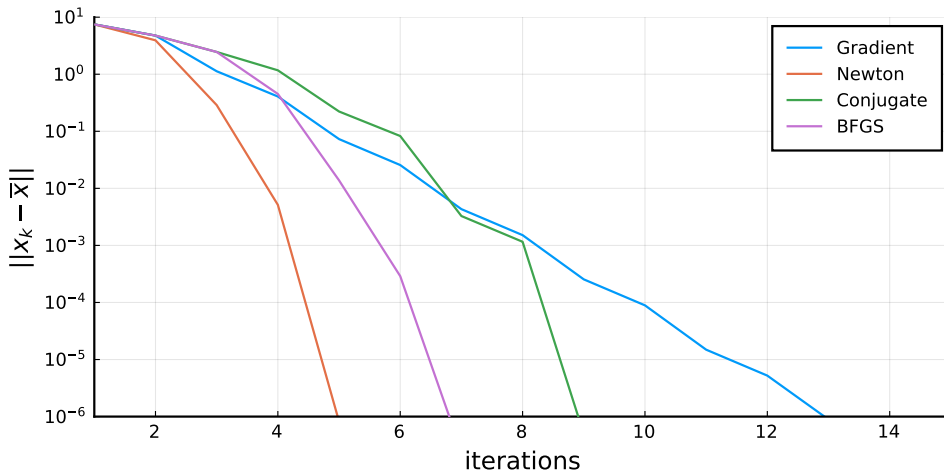


Figure 43: Convergence comparison for the four methods

6.2.3 Conditioning

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

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

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.

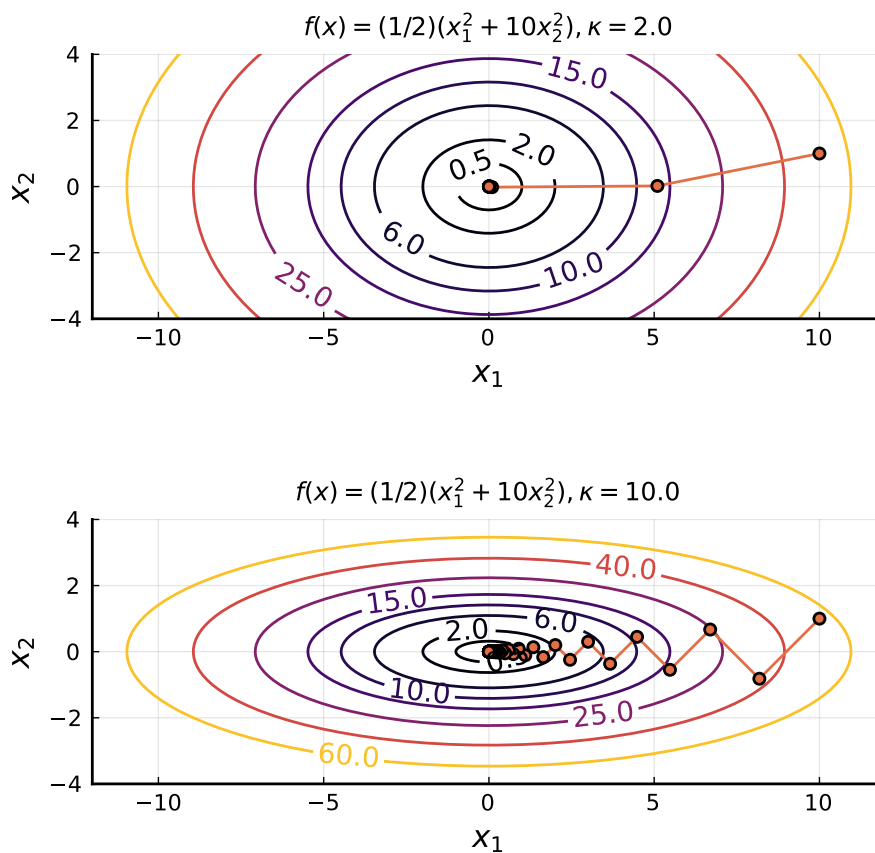
Roughly speaking, having $\kappa \geq 10^k$ means that at each iteration, k digits of accuracy are lost. As general rule, one would prefer smaller κ numbers, but good values are entirely problem dependent.

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 $f : \mathbb{R}^n \mapsto \mathbb{R}$. 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.

he gradient method with exact line search for different κ :



In the figure above, it is illustrated 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 6.4.

7 Week VII

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.

7.1 Optimality for constrained problems

We now investigate how to derive optimality conditions for the problem:

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

In particular, we are interested in understanding the role of the feasibility set S 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 7.1. Cone of feasible directions

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

$$D = \{d : d \neq 0, \text{ and } \bar{x} + \lambda d \in S \text{ for all } \lambda \in (0, \delta) \text{ for some } \delta > 0\}.$$

Definition 7.2. Cone of descent directions

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

$$F = \{d : f(\bar{x} + \lambda d) < f(\bar{x}) \text{ for all } \lambda \in (0, \delta) \text{ for some } \delta > 0\}.$$

These cones are geometrical descriptions of the regions that, from a given point \bar{x} , one can obtain feasible (D) and improving (F) solutions. This is useful because it allows us to express the optimality conditions for \bar{x} as it observes that $F \cap D = \emptyset$ holds. In other words, \bar{x} is optimal if no feasible direction can improve 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. Let us define an algebraic representation for F to reach that objective. Let us assume that $f : S \subset \mathbb{R}^n \mapsto \mathbb{R}$ is differentiable. Recall that d is a descent direction at \bar{x} if $\nabla f(\bar{x})^\top d < 0$. Thus, we can define the set F_0 .

$$F_0 = \{d : \nabla f(\bar{x})^\top d < 0\}$$

As an algebraic representation for F , notice that F_0 is an open half-space formed by the hyperplane with normal $\nabla f(\bar{x})$. Figure 44 illustrates the condition $F_0 \cap D = \emptyset$.

Theorem 7.3 establishes that the condition $F_0 \cap D = \emptyset$ is necessary for optimality in constrained optimisation problems.

Theorem 7.3. Geometric necessary condition

Let $S \subseteq \mathbb{R}^n$ be a nonempty set, and let $f : S \rightarrow \mathbb{R}$ be differentiable at $\bar{x} \in S$. If \bar{x} is a local optimal solution to

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

then $F_0 \cap D = \emptyset$, where $F_0 = \{d : \nabla f(\bar{x})^\top d < 0\}$ and D is the cone of feasible directions.

The proof for this theorem consists of using the separation theorem to show that $F_0 \cap D = \emptyset$ implies that the first-order optimality condition $\nabla f(\bar{x})^\top d \geq 0$ holds.

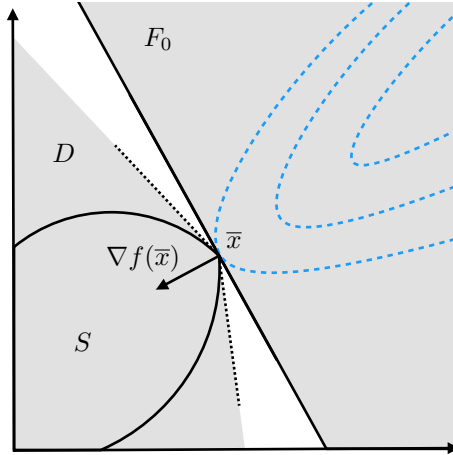


Figure 44: Illustration of the cones F_0 and D for the optimal point \bar{x} . Notice that D is an open set.

As discussed earlier (in Lecture 4), these conditions become sufficient for optimality in the presence of convex. Moreover, if f is strictly convex, then $F = F_0$. If f is linear, it might be worth considering $F'_0 = \{d \neq 0 : \nabla f(\bar{x})^\top d \leq 0\}$ to allow for considering orthogonal directions.

7.1.1 Inequality constrained problems

In mathematical programming applications, a set of inequalities typically expresses the feasibility set S . Let us redefine P as:

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

where $g_i : \mathbb{R}^n \mapsto \mathbb{R}$, are differentiable functions for $i = 1, \dots, m$ and $X \subset \mathbb{R}^n$ is a nonempty open set. The differentiability of g_i , $i = 1, \dots, m$, allows for the definition of a proxy for D using the gradients of the binding constraints $i \in I = \{i : g_i(\bar{x}) = 0\}$ at \bar{x} . This set, denoted by G_0 , is defined as:

$$G_0 = \{d : \nabla g_i(\bar{x})^\top d < 0, i \in I\}.$$

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 7.4. As $F_0 \cap D = \emptyset$ must hold for a locally optimal solution $\bar{x} \in S$, $F_0 \cap G_0 = \emptyset$ must also hold.

Lemma 7.4 Let $S = \{x \in X : g_i(x) \leq 0 \text{ for all } i = 1, \dots, m\}$, where $X \subset \mathbb{R}^n$ is a nonempty open set and $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$ a differentiable function for all $i = 1, \dots, m$. For a feasible point $\bar{x} \in S$, let $I = \{i : g_i(\bar{x}) = 0\}$ be the index set of the binding (or active) constraints. Let

$$G_0 = \{d : \nabla g_i(\bar{x})^\top d < 0, i \in I\}$$

Then $G_0 \subseteq D$, where D is the cone of feasible directions.

In settings in which g_i is affine for some $i \in I$, it might be worth considering $IG'_0 = \{d \neq 0 : \nabla g_i(\bar{x})^\top d \leq 0, i \in I\}$ so those orthogonal feasible directions can also be represented. Notice that in this case, $D \subseteq G'_0$.

7.2 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 algebraic conditions are convenient as they only involve the gradients of the binding constraints,

and they can be verified computationally.

Theorem 7.5. Fritz-John necessary conditions

Let $X \subseteq \mathbb{R}^n$ be a nonempty open set, and let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$ be differentiable for all $i = 1, \dots, m$. Additionally, let \bar{x} be feasible and $I = \{i : g_i(\bar{x}) = 0\}$. If \bar{x} solves P locally, there exist scalars $u_i, i \in \{0\} \cup I$, such that:

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

Proof. Since \bar{x} solves P locally, Theorem 7.3 guarantees that there is no d such that $\nabla f(\bar{x})^\top d < 0$ and $\nabla g_i(\bar{x})^\top d < 0$ for each $i \in I$. Let A be the matrix whose rows are $\nabla f(\bar{x})^\top$ and $\nabla g_i(\bar{x})^\top$ for $i \in I$.

We can use Farkas' theorem to show that if $Ad < 0$ is inconsistent, there is nonzero $p \geq 0$ such that $A^\top p = 0$. Letting $p = (u_0, u_{i_1}, \dots, u_{i_{|I|}})$ for $I = \{i_1, \dots, i_{|I|}\}$ and making $u_i = 0$ for $i \notin I$, the result follows. 😊

The proof considers that, if \bar{x} is optimal, then $f(\bar{x})^\top d \geq 0$ holds and a matrix A formed by:

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

with $I = \{i_1, \dots, i_{|I|}\}$, will violate $Ad < 0$. This is used with a variant of Farkas' theorem (known as Gordan's theorem) to show that the alternative system $A^\top p = 0$, with $p \geq 0$ holds, which, by setting $p = [u_0, u_{i_1}, \dots, u_{i_{|I|}}]$ and enforcing that the remainder of the gradients $\nabla g_i(\bar{x})$, for $i \notin I$, are removed by setting $u_i = 0$, which leads precisely to the Fritz-John conditions.

The multipliers u_i , for $i = 0, \dots, m$, are named Lagrangian multipliers due to the connection with Lagrangian duality, as we will see later. Also, notice that for nonbinding constraints ($g_i(\bar{x}) < 0$ for $i \notin I$), u_i must be zero to form the Fritz-John conditions. This condition is called complementary slackness.

Unfortunately, The Fritz-John conditions are too weak, which is problematic in some rather common settings. A point \bar{x} satisfies the Fritz-John conditions only if $F_0 \cap G_0 = \emptyset$, which is trivially satisfied when $G_0 = \emptyset$.

For example, the Fritz-John conditions are trivially satisfied for points where some of the gradient vanishes (i.e., $\nabla f(\bar{x}) = 0$ or $\nabla g_i(\bar{x}) = 0$ for some $i = 1, \dots, m$). Sets with no relative interior near \bar{x} also satisfy Fritz-John conditions.

An interesting case is for problems with equality constraints, as illustrated in Figure 45. In general, if the additional regularity condition that the gradients $\nabla g_i(\bar{x})$ are linearly independent does not hold, \bar{x} trivially satisfies the Fritz-John conditions.

7.3 Karush-Kuhn-Tucker conditions

The Karush-Kuhn-Tucker (KKT) conditions are the Fritz-John conditions with an extra regularity requirement for $\bar{x} \in S$. This regularity requirement is called *constraint qualification* and, in a general sense, is meant to prevent the trivial case $G_0 = \emptyset$, thus making the optimality conditions stronger (i.e., more stringent).

This is achieved by making $u_0 = 1$ in Theorem 7.5, which ultimately implies that the gradients $\nabla g_i(\bar{x})$ for $i \in I$ must be linearly independent. This condition is called *linearly independent constraint*

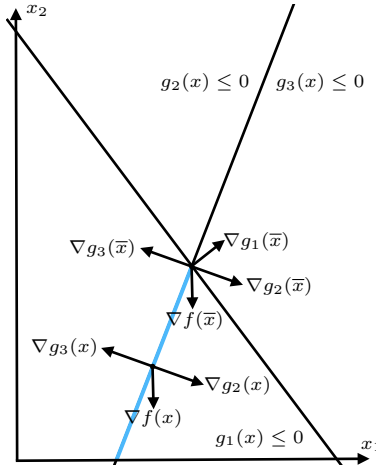


Figure 45: All points in the blue segment satisfy FJ conditions, including the minimum \bar{x} .

qualification (LICQ) and is one of several known constraint qualifications that can be used to guarantee regularity of $\bar{x} \in S$.

Theorem 7.6 establishes the KKT conditions as necessary for local optimality of \bar{x} assuming that LICQ holds. For notational simplicity, let us assume for now that:

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

Theorem 7.6. Karush-Kuhn-Tucker necessary conditions

Let $X \subseteq \mathbb{R}^n$ be a nonempty open set, and let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$ be differentiable for all $i = 1, \dots, m$. Additionally, for a feasible \bar{x} , let $I = \{i : g_i(\bar{x}) = 0\}$ and suppose that $\nabla g_i(\bar{x})$ are linearly independent for all $i \in I$. If \bar{x} solves P locally, there exist scalars u_i for $i \in I$ such that:

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

Proof. By Theorem 7.5, there exists nonzero (\hat{u}_i) for $i \in \{0\} \cup I$ such that:

$$\begin{aligned} \hat{u}_0 \nabla f(\bar{x}) + \sum_{i=1}^m \hat{u}_i \nabla g_i(\bar{x}) &= 0 \\ \hat{u}_i &\geq 0, \quad i = 0, \dots, m \end{aligned}$$

Note that $\hat{u}_0 > 0$, as the linear independence of $\nabla g_i(\bar{x})$ for all $i \in I$ implies that $\sum_{i=1}^m \hat{u}_i \nabla g_i(\bar{x}) \neq 0$. Now, let $u_i = \hat{u}_i / \hat{u}_0$ for each $i \in I$ and $u_i = 0$ for all $i \notin I$. 😊

The proof builds upon the Fritz-John conditions, which under the assumption that the gradients of the active constraints $\nabla g_i(\bar{x})$ for $i \in I$ are independent, the multipliers \hat{u}_i can be rescaled so that $u_0 = 1$.

The general conditions, including inequality and equality constraints, are posed as follows. Notice that the Lagrange multipliers v_i associated with the equality constraints $h(\bar{x}) = 0$ for $i = 1, \dots, l$ are not restricted in sign, and the complementary slackness condition is not explicitly stated since it holds redundantly. These can be obtained by replacing equality constraints $h(x) = 0$ with two equivalent inequalities $h_-(x) \leq 0$ and $-h_+(x) \leq 0$ and writing the conditions in Theorem 7.6. Also, notice that, without constraints, the KKT conditions reduce to the unconstrained first-order condition $\nabla f(\bar{x}) = 0$.

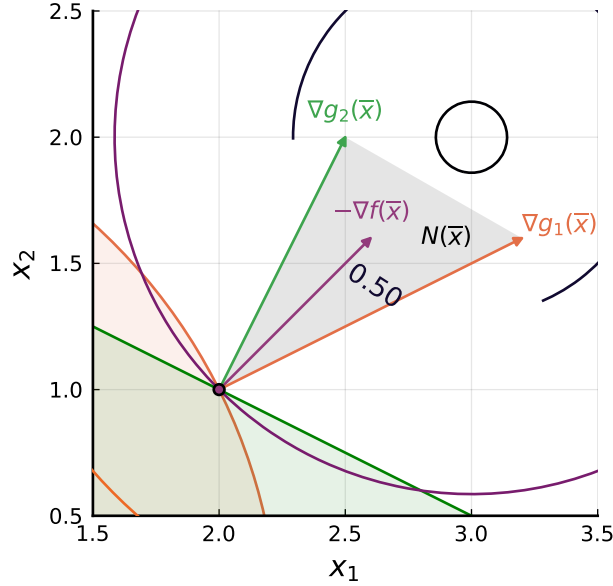


Figure 46: Graphical illustration of the KKT conditions at the optimal point \bar{x}

$$\begin{aligned} \nabla f(\bar{x}) + \sum_{i=1}^m u_i \nabla g_i(\bar{x}) + \sum_{i=1}^l v_i \nabla h_i(\bar{x}) &= 0 && \text{(dual feasibility 1)} \\ u_i g_i(\bar{x}) &= 0, \quad i = 1, \dots, m && \text{(complementary slackness)} \\ \bar{x} \in X, \quad g_i(\bar{x}) &\leq 0, \quad i = 1, \dots, m && \text{(primal feasibility)} \\ h_i(x) &= 0, \quad i = 1, \dots, l \\ u_i &\geq 0, \quad i = 1, \dots, m && \text{(dual feasibility 2)} \end{aligned}$$

The KKT conditions can be interpreted geometrically as follows. Consider the cone spanned by the active constraints at \bar{x} , defined as $N(\bar{x}) = \{\sum_{i \in I} u_i \nabla g_i(\bar{x}) : u_i \geq 0\}$. A solution \bar{x} will then satisfy the KKT conditions if $-\nabla f(\bar{x}) \in N(\bar{x})$, which is equivalent to $-\nabla f(\bar{x}) = \sum_{i=1}^m u_i \nabla g_i(\bar{x})$. Figure 46 illustrates this condition.

7.4 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 a proxy for D , it is important to be sure that the former is a reliable description of the latter.

Specifically, 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 to an optimal solution. Remind that gradients can only provide a *first-order* approximation of the feasible region, which might lead to mismatches. This is typically when the feasible region has cusps or a single feasible point.

Constraint qualification can be seen as certificates for proper relationships between the set of feasible directions:

$$G'_0 = \{d \neq 0 : \nabla g_i(\bar{x})^\top d \leq 0, i \in I\}$$

and the cone of tangents (or tangent cone):

$$T = \{d : d = \lim_{k \rightarrow \infty} \lambda_k (x_k - \bar{x}), \lim_{k \rightarrow \infty} x_k = \bar{x}, x_k \in S, \lambda_k > 0, \forall k\} \quad (60)$$

with $S = \{g_i(x) \leq 0, i = 1, \dots, m; h(x) = 0, i = 1, \dots, l; x \in X\}$.

The cone of tangents represents all directions in which the feasible region allows for an arbitrarily

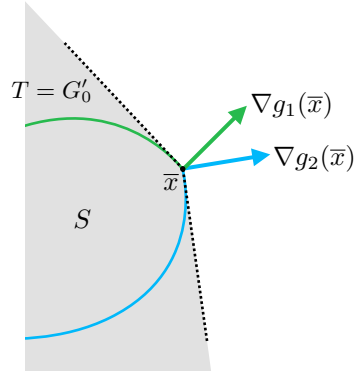


Figure 47: Case I

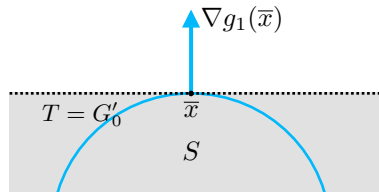


Figure 48: Case II

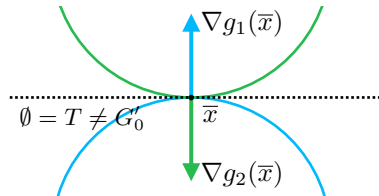


Figure 49: Case III

small movement from the point \bar{x} while retaining feasibility. As the name suggests, it is normally formed by the tangent lines to S at \bar{x} . However, if the point 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 (60) is the following: consider a sequence of feasible points $x \in S$ in any trajectory you like, but in a way that the sequence converges to \bar{x} . Then, take the last (in a limit sense, since $k \rightarrow \infty$) x_k and consider this direction from which x_k came onto \bar{x} . The collection of all these directions from all possible trajectories is what forms the cone of tangents.

Constraint qualification holds when $T = G'_0$ holds for \bar{x} , a condition named *Abadie's constraint qualification*. In the presence of equality constraints, the condition becomes $T = G'_0 \cap H_0$, with:

$$H_0 = \{d : \nabla h_i(\bar{x})^\top d = 0, i = 1, \dots, l\}.$$

In the figures below, it illustrates the tangent cone T and the cone of feasible directions (G'_0) for cases when constraint qualification holds (Figures 47 and 48) for which case $T = G'_0$, and a case for when it does not (Figure 49, where $T = \emptyset$ and G'_0 is given by the dashed black line).

In the figure above, CQ holds for 47 and 48, since the tangent cone T and the cone of feasible directions G'_0 (denoted by the dashed black lines and a grey area) match; for 49, they do not match, as $T = \emptyset$.

The importance of Abadie constraint qualification is that it allows for generalising the KKT conditions by replacing the condition with the linear independence of the gradients $\nabla g_i(\bar{x})$ for $i \in I$. This allows us to state the KKT conditions as presented in Theorem 7.7.

Theorem 7.7. Karush-Kuhn-Tucker necessary conditions II

Consider the problem

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

Let $X \subseteq \mathbb{R}^n$ be a nonempty open set, and let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$ be differentiable for all $i = 1, \dots, m$. Additionally, for a feasible \bar{x} , let $I = \{i : g_i(\bar{x}) = 0\}$ and suppose that Abadie CQ holds at \bar{x} . If \bar{x} solves P locally, there exist scalars u_i for $i \in I$ such that:

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

Despite being a more general result, Theorem 7.7 is of little use, as 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 include:

1. **Linear independence (LI)CQ:** holds at \bar{x} if $\nabla g_i(\bar{x})$, for $i \in I$, as well as $\nabla h_i(\bar{x})$, $i = 1, \dots, l$ are linearly independent.
2. **Affine CQ:** holds for all $x \in S$ if g_i , for all $i = 1, \dots, m$, and h_i , for all $i = 1, \dots, l$, are affine.
3. **Slater's CQ:** holds for all $x \in S$ if g_i is a convex function for all $i = 1, \dots, m$, h_i is an affine function for all $i = 1, \dots, l$, and there exists $x \in S$ such that $g_i(x) < 0$ for all $i = 1, \dots, m$.

Slater's constraint qualification is the most frequently used, particularly in 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: $P = \{\min. x_1 : x_1^2 + x_2 \leq 0, x_2 \geq 0\}$. Notice that P is convex and therefore the KKT system for P is:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = 0; u_1, u_2 \geq 0,$$

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 that KKT conditions are only sufficient (not necessary) in the presence of convexity.

Corollary 7.8 summarises the setting in which one should expect the KKT conditions to be necessary and sufficient conditions for global optimality, i.e., convex optimisation.

Corollary 7.8 (Necessary and sufficient KKT conditions). Suppose that Slater's CQ holds. Then, if f is convex, the conditions of Theorem 7.7 are *necessary and sufficient* for \bar{x} to be a globally optimal solution.

8 Week VIII

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 connect 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.

8.1 The concept of relaxation

The idea of using relaxations is central in several constrained optimisation methods. Generally, 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 classic 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 also integers. However, there are several other settings in which relaxations are purposely derived to lead to problems with a convenient structure that can be exploited.

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $S \subseteq \mathbb{R}^n$. Consider the following problem:

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

Definition 8.1 provides the conditions for P_R to be a *relaxation* of P , where:

$$(P_R) : \min. \{f_R(x) : x \in S_R\}$$

with $f_R : \mathbb{R}^n \rightarrow \mathbb{R}$, $S_R \subseteq \mathbb{R}^n$.

Definition 8.1. 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 . 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 8.2 presents the technical result that allows for using relaxations for solving P .

Theorem 8.2. Relaxation theorem

Let us define:

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

If P_R is a relaxation of P , then the following hold:

1. if P_R is infeasible, so is P ;
2. if \bar{x}_R is an optimal solution to P_R such that $\bar{x}_R \in S$ and $f_R(\bar{x}_R) = f(\bar{x}_R)$, then \bar{x}_R is optimal to P as well.

Proof. Result (1) follows since $S \subseteq S_R$. To show (2), notice that $f(\bar{x}_R) = f_R(\bar{x}_R) \leq f_R(x) \leq f(x)$ 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 \bar{x}_R (first inequality) and the definition of a relaxation (second inequality) to derive the optimality condition of \bar{x}_R for P , which is $f(\bar{x}_R) \leq f(\bar{x})$ for all $x \in S$.

8.2 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 $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $h : \mathbb{R}^n \rightarrow \mathbb{R}^l$, and assume that $X \subseteq \mathbb{R}^n$ is an open set. Then, consider P defined as:

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

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*) of P is:

$$(D) : \theta(u, v) = \inf_{x \in X} \phi(x, u, v)$$

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 \bar{x} for $\phi(x, u, v)$, given (u, v) .

8.2.1 Weak and strong duality

Weak and robust duality are, to some extent, consequences of Theorem 8.2 and the fact that the Lagrangian relaxation is indeed a relaxation of P . We start with the equivalent to Definition 8.1, referred to as *weak duality*.

Theorem 8.3. Weak Lagrangian duality

Let x be a feasible solution to P , and let (u, v) be such that $u \geq 0$, i.e., feasible for D . 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

$$\theta(u, v) = \inf_{x \in X} \{f(x) + u^\top g(x) + v^\top h(x)\} \leq f(x) + u^\top g(x) + v^\top h(x) \leq f(x).$$

Which completes the proof.



The proof uses the fact that the infimal of the Lagrangian function $\phi(x, u, v)$, and in fact, any value for $\phi(x, u, v)$ for all primal feasible x and dual feasible $u \geq 0$ (a condition for the Lagrangian relaxation to be indeed a relaxation) bounds to $f(x)$. This arises from observing $g(x) \leq 0$ 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 8.3. This can be achieved by optimising $\theta(u, v)$ in the space of the dual variables (u, v) , that is:

$$(D) : \theta(u, v) = \inf_{x \in X} \phi(x, u, v).$$

The use of Lagrangian dual problems is an alternative for dealing with constrained optimisation prob-

lems, as they allow to convert of 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 8.3.

Corollary 8.4 (Weak Lagrangian duality II).

$$\sup_{u,v} \{\theta(u,v) : u \geq 0\} \leq \inf_x \{f(x) : g(x) \leq 0, h(x) = 0, x \in X\}.$$

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 8.5 (Strong Lagrangian duality). If $f(\bar{x}) = \theta(\bar{u}, \bar{v})$, $\bar{u} \geq 0$, and $\bar{x} \in \{x \in X : g(x) \leq 0, h(x) = 0\}$, then \bar{x} and (\bar{u}, \bar{v}) are optimal solutions to P and D , respectively.

Proof. Use part (2) of Theorem 8.2 with D being a Lagrangian relaxation. 😎

Notice that Corollary 8.5 implies that if the optimal solution value of the primal and the dual problems match, the respective primal and dual solutions are optimal. However, to use Lagrangian relaxations to solve constrained optimisation problems, we need the opposite clause also to hold, which is called *strong duality* and, unfortunately, does not always hold.

1. Geometric interpretation of Lagrangian duality

Let us focus on a graphical interpretation of Lagrangian dual problems to investigate the cases in which strong duality can hold. 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 $f : \mathbb{R}^n \mapsto \mathbb{R}$, a single constraint $g : \mathbb{R}^n \mapsto \mathbb{R}$ and $X \subseteq \mathbb{R}^n$ an open set.

Let us define the mapping $G = \{(y, z) : y = g(x), z = f(x), x \in X\}$, which consists of a mapping of points $x \in X$ to the (y, z) -space obtained using $(f(x), g(x))$. In this setting, solving P means finding a point with minimum ordinate z for which $y \leq 0$. Figure 50 illustrates this setting.

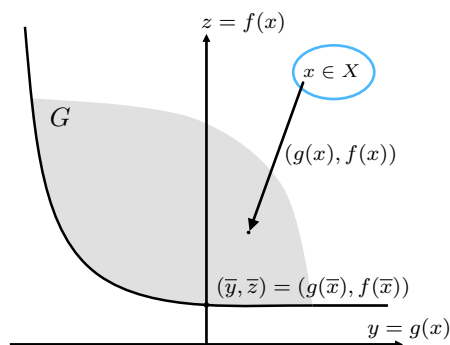


Figure 50: Illustration of the mapping G , in which solving P amounts to finding the lowermost point on the vertical axis (the ordinate) still contained within G .

Now, assume that $u \geq 0$ is given. The Lagrangian function is given by:

$$\theta(u) = \min_x \{f(x) + ug(x) : x \in X\},$$

, which can be represented by a hyperplane of the form $z = \alpha - uy$. Therefore, optimising the Lagrangian dual problem $(D) : \sup_u \{\theta(u)\}$ consists of finding the slope $-u$ that would achieve the maximum intercept on the ordinate z while being a supporting hyperplane for G . Figure 51 illustrates this effect. Notice that, in this case, the optimal values of the primal and dual problems

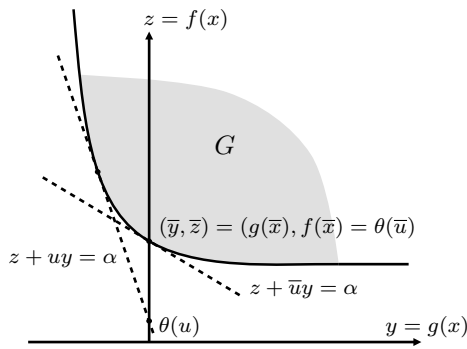


Figure 51: Solving the Lagrangian dual problem is the same as finding the coefficient u such that $z = \alpha - uy$ is a supporting hyperplane of G with the uppermost intercept α . Notice that, for \bar{u} , the hyperplane supports G at the same point that solves P .

coincide.

The *perturbation function* $v(y) = \min. \{f(x) : g(x) \leq y, x \in X\}$ is an analytical tool that plays an important role in understanding when strong duality holds, which, in essence, is the underlying reason why the optimal values of the primal and dual problems coincide.

Specifically, notice that $v(y)$ is the greatest monotone nonincreasing lower envelope of G . Moreover, the reason why $f(\bar{x}) = \theta(\bar{u})$ is related to the convexity of $v(y)$, which implies that:

$$v(y) \geq v(0) - \bar{u}y \text{ for all } y \in \mathbb{R}.$$

Notice that this is a consequence of Theorem 12 from Lecture 2 (that states that convex sets have supporting hyperplanes for all points on their boundary) and Theorem 5 in Lecture 3 (that convex functions have convex epigraphs)

A *duality gap* exists when the perturbation function $v(y)$ does not have supporting hyperplanes within all its domains, which is otherwise the case when $v(y)$ is convex. Figure 1 illustrates a case in which $v(y)$ is not convex and therefore $\theta(\bar{u}) < f(\bar{x})$.

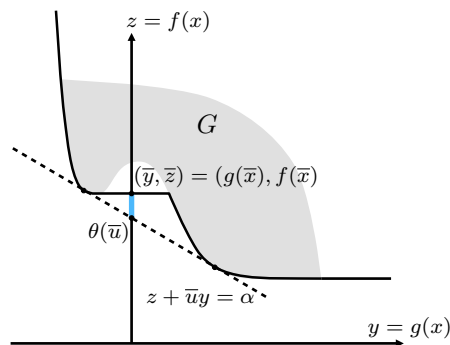


Figure 52: An example in which the perturbation function $v(y)$ is not convex. Notice the consequent mismatch between the intercept of the supporting hyperplane and the lowermost point on the ordinate still contained in G .

Let us illustrate the above with two numerical examples. First, consider the following problem:

$$(P) : \begin{aligned} \min. \quad & x_1^2 + x_2^2 \\ & x_1 + x_2 \geq 4 \\ & x_1, x_2 \geq 0. \end{aligned}$$

The Lagrangian dual function is given by:

$$\begin{aligned}
(D) : \theta(u) &= \inf\{x_1^2 + x_2^2 + u(-x_1 - x_2 + 4) : x_1, x_2 \geq 0\} \\
&= \inf\{x_1^2 - ux_1 : x_1 \geq 0\} + \inf\{x_2^2 - ux_2 : x_2 \geq 0\} + 4u \\
&= \begin{cases} -1/2u^2 + 4u, & \text{if } u \geq 0 \\ -4u, & \text{if } u < 0. \end{cases}
\end{aligned}$$

The primal problem P as a constrained optimisation problem, and the dual problem D , as an unconstrained optimisation problem. Notice how the Lagrangian dual function is discontinuous due to the implicit minimisation in x of $\theta(u) = \inf_{x \in X} \phi(x, u)$:

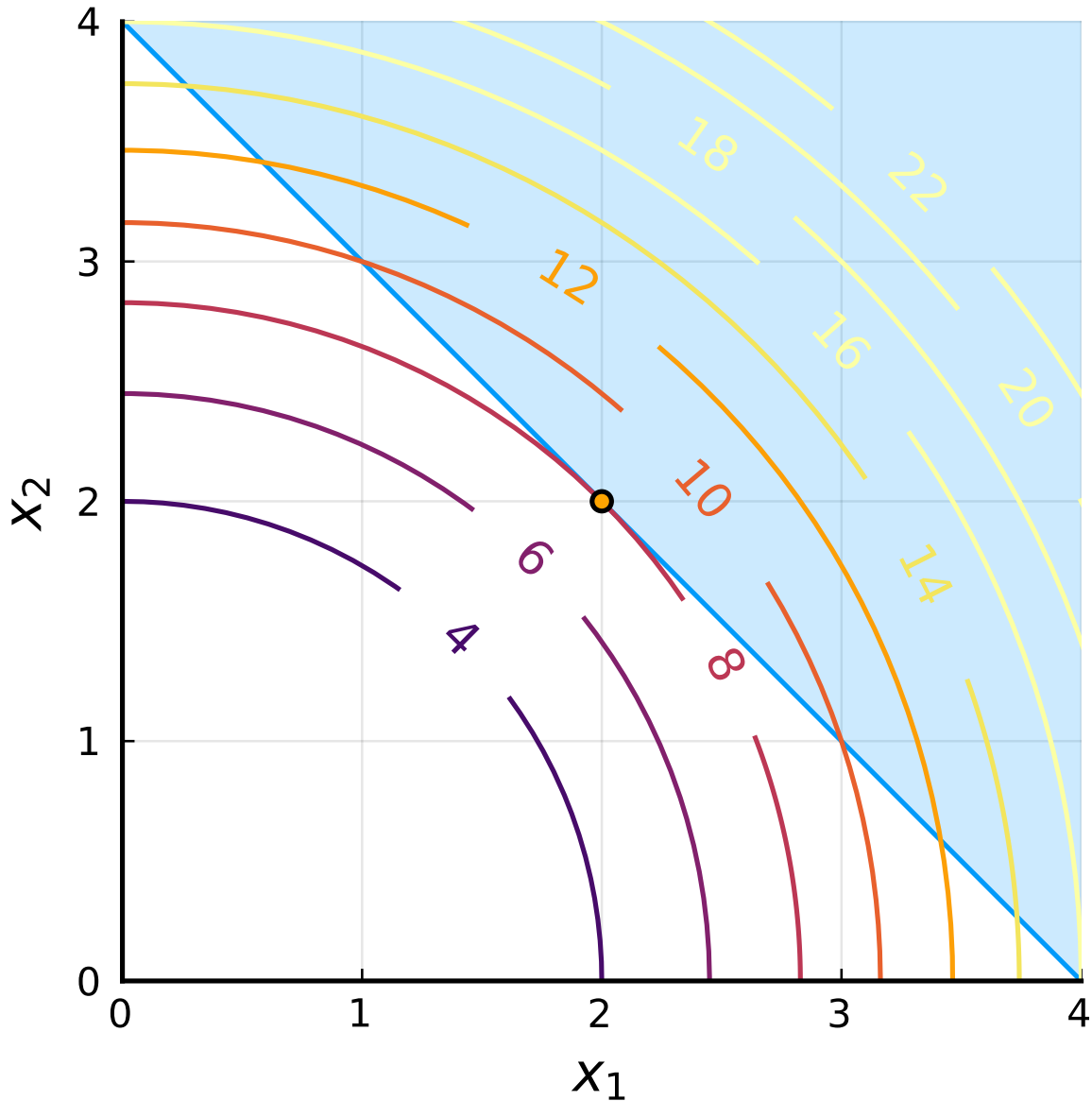


Figure 53: P

Figures 53 and 54 provide a graphical representation of the primal problem P and dual problem D . As can be seen, both problems have as optimal value $f(\bar{x}_1, \bar{x}_2) = \theta(\bar{u}) = 8$, with the optimal solution $\bar{x} = (2, 2)$ for P and $\bar{u} = 4$ for D .

To draw the (g, f) map of X , we proceed as follows. First, notice that:

$$v(y) = \min. \{x_1^2 + x_2^2 : -x_1 - x_2 + 4 \geq y\}$$

which shows that $(x_1, x_2) = (0, 0)$ if $y > 4$. For $y \leq 4$, $v(y)$ can be equivalently rewritten as:

$$v(y) = \min. \{x_1^2 + x_2^2 : -x_1 - x_2 + 4 = y\}.$$

Let $h(x) = -x_1 - x_2 + 4$ and $f(x) = x_1^2 + x_2^2$. Now, the optimality conditions for \bar{x} to be an optimum for v are such that:

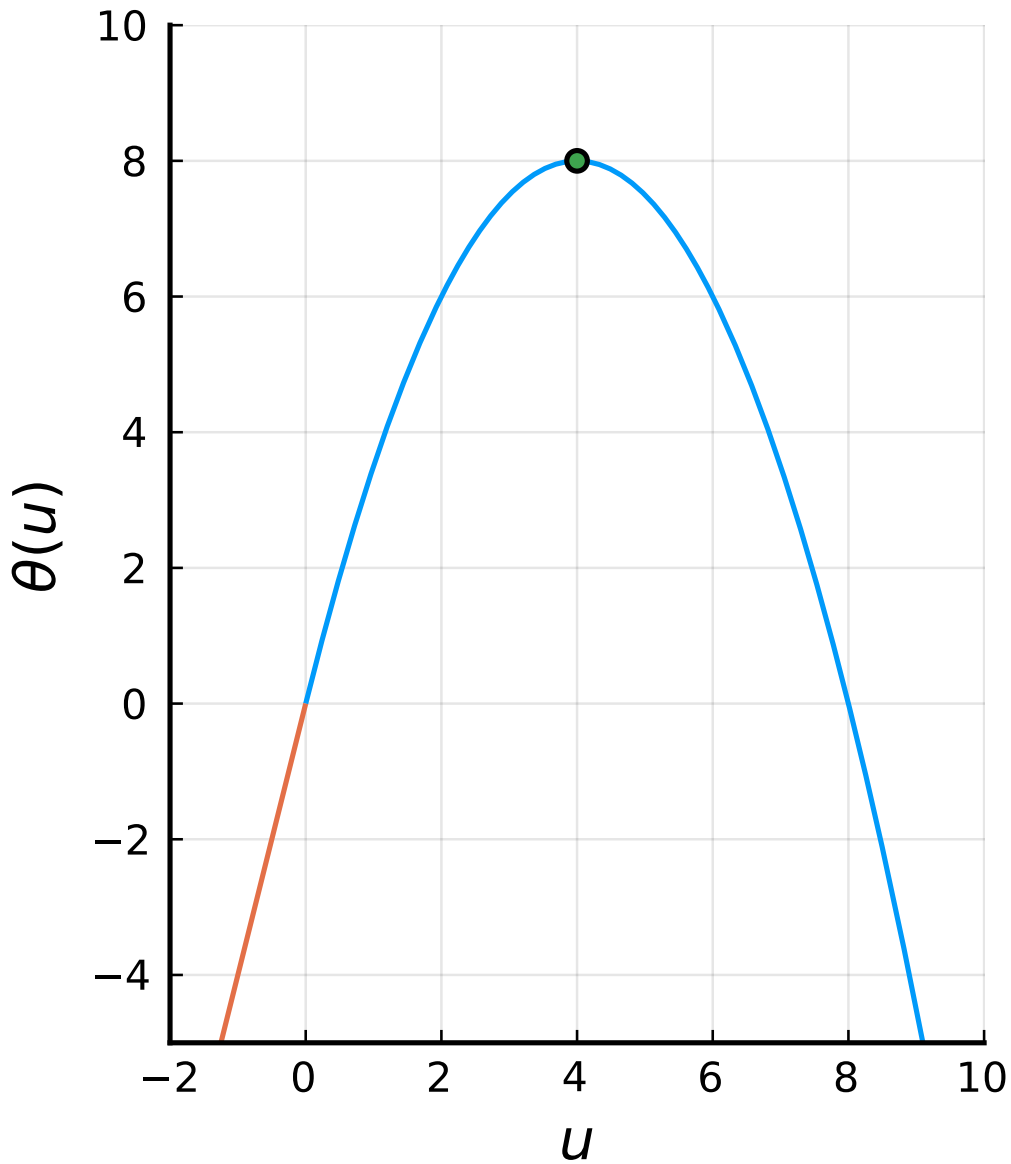


Figure 54: D

$$\nabla f(\bar{x}) + u\nabla h(\bar{x}) = 0 \Rightarrow \begin{cases} 2x_1 - u = 0 \\ 2x_2 - u = 0 \end{cases} \Rightarrow \bar{x}_1 = \bar{x}_2 = u/2.$$

From the definition of $h(x)$, we see that $u = 4 - y$, and thus $\bar{x} = (\frac{4-y}{2}, \frac{4-y}{2})$, which, substituting in $f(x)$ gives $v(y) = (4 - y)^2/2$. Note that $v(y) \geq v(0) - \bar{u}y$ holds for all $y \in \mathbb{R}$, that is, $v(y)$ is convex. Also, notice that the supporting hyperplane is exactly $z = 8 - 4y$.

Now, let us consider a second example: the feasible set is not convex, so the mapping G will not be convex either. For that, consider the problem:

$$(P) : \begin{aligned} \min. \quad & -2x_1 + x_2 \\ \text{subject to: } & x_1 + x_2 = 3 \\ & x_1, x_2 \in X. \end{aligned}$$

Where $X = \{(0, 0), (0, 4), (4, 4), (4, 0), (1, 2), (2, 1)\}$. The optimal point $\bar{x} = (2, 1)$. The Lagrangian dual function is given by:

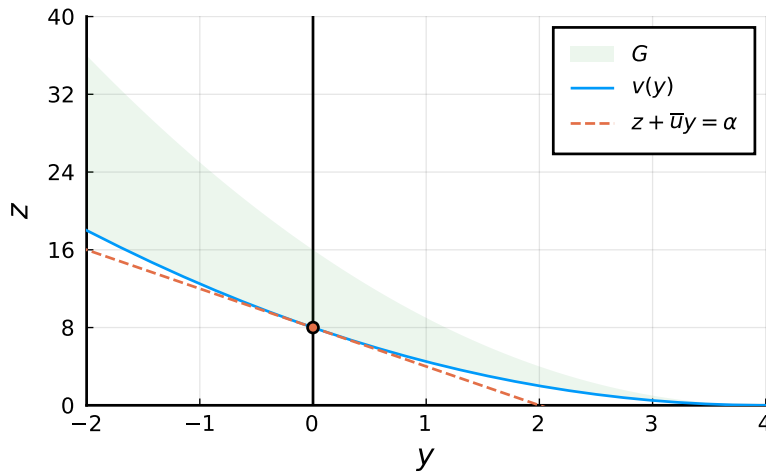


Figure 55: The G mapping for the first example.

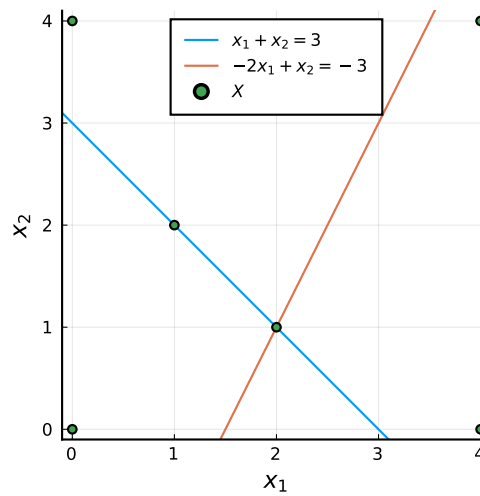


Figure 56: P

$$\begin{aligned} \theta(v) &= \min\{(-2x_1 + x_2) + v(x_1 + x_2 - 3) : (x_1, x_2) \in X\} \\ &= \begin{cases} -4 + 5v, & \text{if } v \leq -1 \\ -8 + v, & \text{if } -1 \leq v \leq 2 \\ -3v, & \text{if } v \geq 2. \end{cases} \end{aligned}$$

Figure 56 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 57.

The primal problem P as a constrained optimisation problem and the dual problem D . Notice how the Lagrangian dual function is concave and piecewise linear, despite the nonconvex nature of P : Similarly to the previous example, we can plot the G mapping, 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 $v(y)$ in this case is discontinuous, represented by the three lowermost points. $v(y)$ does not have a supporting hyperplane at the minimum of P , which illustrates the existence of a duality gap, as stated by the fact that $-3 = f(\bar{x}) > \theta(\bar{v}) = -6$.

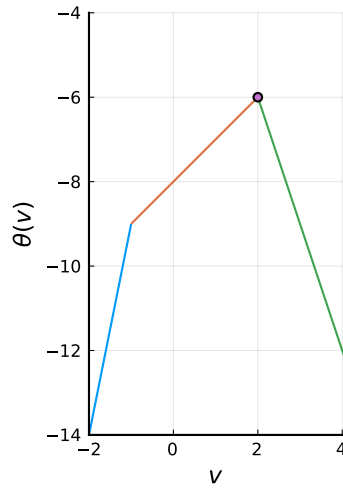


Figure 57: (D) : max. $\theta(v)$

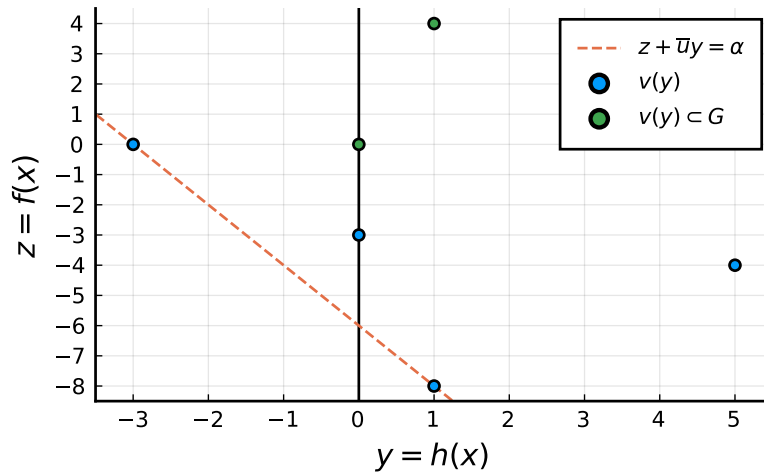


Figure 58: The G mapping for the second example. The blue dots represent the perturbation function $v(y)$, which is not convex and thus cannot be supported everywhere. Notice the duality gap represented by the difference between the intercept of $z = -6 - 2y$ and the optimal value of P at $(0, -3)$.

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 8.6.

Theorem 8.6. Strong duality

Let $X \subseteq \mathbb{R}^n$ be a nonempty convex set. Moreover, let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be convex functions, and let $h : \mathbb{R}^n \rightarrow \mathbb{R}^l$ be an affine function: $h(x) = Ax - b$. Suppose that Slater's constraint qualification holds. 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 $\theta(u, v) = \inf_{x \in X} \{f(x) + u^\top g(x) + v^\top h(x)\}$ 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 \bar{x} , then $\sup_{u,v} \{\theta(u, v) : u \geq 0\}$ is achieved at (\bar{u}, \bar{v}) with $\bar{u} \geq 0$ and $\bar{u}^\top g(\bar{x}) = 0$.

The proof for the strong duality theorem follows the following outline:

- (a) 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 8.3; the right-hand side holds by assumption of the existence of a feasible point from Slater's

constraint qualification).

(b) Formulate the inconsistent system:

$$f(x) - \gamma < 0, \quad g(x) \leq 0, \quad h(x) = 0, \quad x \in X.$$

(c) Use the separation theorem (or a variant form of Farkas theorem) to show that $(\bar{u}_0, \bar{u}, \bar{v})$ with $\bar{u}_0 > 0$ and $\bar{u} \geq 0$ exists such that, after scaling using \bar{u}_0 one obtains $\theta(\bar{u}, \bar{v}) := f(x) + \bar{u}^\top g(x) + \bar{v}^\top h(x) \geq \gamma$, $x \in X$, which requires the assumption of Slater's constraint qualification.

(d) From weak duality (Theorem 8.3), we have that $\theta(\bar{u}, \bar{v}) \leq \gamma$, which combined with the above, yields $\theta(\bar{u}, \bar{v}) = \gamma$.

(e) Finally, an optimal \bar{x} solving the primal problem implies that $g(\bar{x}) \leq 0$, $h(\bar{x}) = 0$, $\bar{x} \in X$, and $f(\bar{x}) = \gamma$. From 3, we have $\bar{u}^\top g(\bar{x}) \geq 0$. As $g(\bar{x}) \leq 0$ and $\bar{u} \geq 0$, $\bar{u}^\top g(\bar{x}) \geq 0 = 0$.

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 8.3, yields strong duality.

8.2.2 Employing Lagrangian duality for solving optimisation problems

Weak duality can be used to derive a stopping criterion for solution methods that generate both primal and dual feasible solutions, also known as primal-dual pairs. Such methods are typically referred to as primal-dual methods, the primal-dual interior point method (which we will discuss in detail in an upcoming lecture) and perhaps the most widely known.

For feasible x and (u, v) , one can bound how suboptimal $f(x)$ is, by noticing that:

$$f(x) - f(\bar{x}) \leq f(x) - \theta(u, v),$$

which is a consequence of $f(\bar{x}) \geq \theta(u, v)$ (i.e., weak duality). We say that x is ϵ -optimal, with $\epsilon = f(x) - \theta(u, v)$.

In essence, (u, v) is a certificate of (sub-)optimality of x , as (u, v) proves that x is ϵ -optimal. Moreover, in case strong duality holds, under the conditions of Theorem 6, one can expect ϵ to converge to zero.

To see how this is the case, observe the following. First, as can be seen in Theorem 8.6, a consequence of strong duality is that complementarity conditions $\bar{u}^\top g(\bar{x}) \geq 0 = 0$ hold for an optimal primal-dual pair $(\bar{x}, (\bar{u}, \bar{v}))$. Secondly, notice that, by definition, \bar{x} and (\bar{u}, \bar{v}) are primal and dual feasible, respectively.

The last component missing is to notice that, if \bar{x} is a minimiser for $\phi(x, \bar{u}, \bar{v}) = f(x) + \bar{u}^\top g(x) + \bar{v}^\top h(x)$, then we must have:

$$\nabla f(\bar{x}) + \sum_{i=1}^m u_i \nabla g_i(\bar{x}) + \sum_{i=1}^l v_i \nabla h_i(\bar{x}) = 0.$$

Combining the above, one can see that we have listed all of the KKT optimality conditions, which under the assumptions of Theorem 8.6 are known to be necessary and sufficient for global optimality. 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.

8.2.3 Saddle point optimality and KKT conditions

An alternative perspective for establishing necessary and sufficient conditions for strong duality involves identifying saddle points for the Lagrangian dual problem.

Let us first define saddle points in the context of Lagrangian duality. Let:

$$(P) : \min. \{f(x) : g(x) \leq 0, h(x) = 0, x \in X\}.$$

Let us define the Lagrangian function $\phi(x, u, v) = f(x) + u^\top g(x) + v^\top h(x)$. A solution $(\bar{x}, \bar{u}, \bar{v})$ is called a *saddle point* if $\bar{x} \in X$, $\bar{u} \geq 0$, and:

$$\phi(\bar{x}, u, v) \leq \phi(\bar{x}, \bar{u}, \bar{v}) \leq \phi(x, \bar{u}, \bar{v})$$

for all $x \in X$ and (u, v) such that $u \geq 0$.

Notice that this definition implies that:

- \bar{x} minimises $\phi(x, u, v)$ when (u, v) is fixed at (\bar{u}, \bar{v}) ;
- (\bar{u}, \bar{v}) maximises $\phi(x, u, v)$ when x is fixed at \bar{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 8.7 establishes the relationship between the existence of saddle points for Lagrangian dual problems and zero duality gaps.

Theorem 8.7. Saddle point optimality and zero duality gap

A solution $(\bar{x}, \bar{u}, \bar{v})$ with $\bar{x} \in X$ and $\bar{u} \geq 0$ 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. $\phi(\bar{x}, \bar{u}, \bar{v}) = \min. \{ \phi(x, \bar{u}, \bar{v}) : x \in X \}$
2. $g(\bar{x}) \leq 0, h(\bar{x}) = 0$, and
3. $\bar{u}^\top g(\bar{x}) = 0$

Moreover, $(\bar{x}, \bar{u}, \bar{v})$ is a saddle point if and only if \bar{x} and (\bar{u}, \bar{v}) are optimal solutions for the primal (P) and dual (D) problems, respectively, with $f(\bar{x}) = \theta(\bar{u}, \bar{v})$.

From Theorem 8.7, it becomes clear that there is a strong connection between the existence of saddle points and the KKT conditions for optimality. Figure 59 illustrates the existence of a saddle point and the related zero optimality gap.

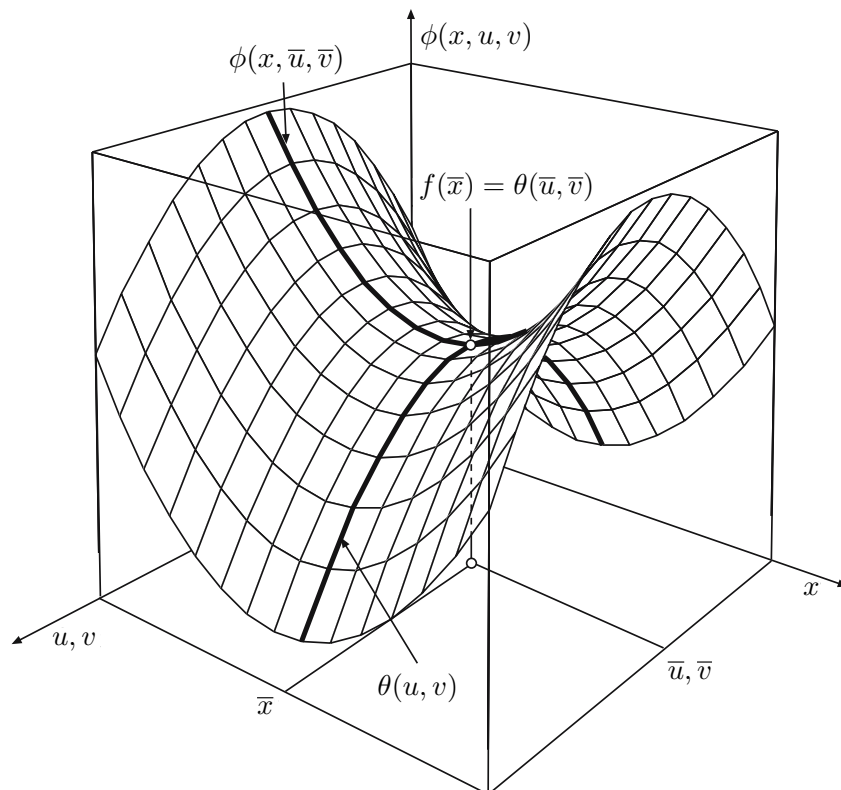


Figure 59: Illustration of a saddle point for the Lagrangian dual problem

8.3 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 exposes 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. However, they are typically not differentiable, requiring the employment of a *nonsmooth optimisation* method to be appropriately solved. Theorem 8.8 establishes the concavity of the Lagrangian dual function.

Theorem 8.8. Concavity of Lagrangian dual functions

Let $X \subseteq \mathbb{R}^n$ be a nonempty compact set, and let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $\beta : \mathbb{R}^n \rightarrow \mathbb{R}^{m+l}$, with $w^\top \beta(x) = \begin{pmatrix} u \\ v \end{pmatrix}^\top \begin{pmatrix} g(x) \\ h(x) \end{pmatrix}$ be continuous. Then $\theta(w) = \inf_x \{f(x) + w^\top \beta(x) : x \in X\}$ is concave in \mathbb{R}^{m+l} .

Proof. Since f and β are continuous and X is compact, θ is finite on \mathbb{R}^{m+l} . Let $w_1, w_2 \in \mathbb{R}^{m+l}$, and let $\lambda \in (0, 1)$. We have:

$$\begin{aligned} \theta[\lambda w_1 + (1 - \lambda)w_2] &= \inf_x \{f(x) + [\lambda w_1 + (1 - \lambda)w_2]^\top \beta(x) : x \in X\} \\ &= \inf_x \{\lambda [f(x) + w_1^\top \beta(x)] + (1 - \lambda)[f(x) + w_2^\top \beta(x)] : x \in X\} \\ &\geq \lambda \inf_x \{f(x) + w_1^\top \beta(x) : x \in X\} + (1 - \lambda) \inf_x \{f(x) + w_2^\top \beta(x) : x \in X\} \\ &= \lambda \theta(w_1) + (1 - \lambda)\theta(w_2). \end{aligned}$$



The proof uses the fact that the Lagrangian function $\theta(w)$ is the infimum of affine functions in w and therefore concave. An alternative approach to show the concavity of the Lagrangian function is to show that it has subgradients everywhere. This is established in Theorem 8.9.

Theorem 8.9. Lagrangian Dual Subgradient

Let $X \subset \mathbb{R}^n$ be a nonempty compact set, and let $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $\beta : \mathbb{R}^n \mapsto \mathbb{R}^{m+l}$, with $w^\top \beta(x) = \begin{pmatrix} u \\ v \end{pmatrix}^\top \begin{pmatrix} g(x) \\ h(x) \end{pmatrix}$ be continuous. If $\bar{x} \in X(\bar{w}) = \{x \in X : x = \operatorname{argmin}\{f(x) + w^\top \beta(x)\}\}$, then $\beta(\bar{x})$ is a subgradient of $\theta(\bar{w})$.

Proof. Since f and β are continuous and X is compact, $X(\bar{w}) \neq \emptyset$ for any $w \in \mathbb{R}^{m+l}$. Now, let $\bar{w} \in \mathbb{R}^{m+l}$ and $\bar{x} \in X(\bar{w})$. Then:

$$\begin{aligned} \theta(w) &= \inf \{f(x) + w^\top \beta(x) : x \in X\} \\ &\leq f(\bar{x}) + w^\top \beta(\bar{x}) \\ &= f(\bar{x}) + (w - \bar{w})^\top \beta(\bar{x}) + \bar{w}^\top \beta(\bar{x}) \\ &= \theta(\bar{w}) + (w - \bar{w})^\top \beta(\bar{x}). \end{aligned}$$



Theorem 8.9 can be used to derive a simple optimisation method for Lagrangian functions using subgradient information that is easily available from the term $\beta(w)$.

8.3.1 The subgradient method

One challenging aspect concerning the solution of Lagrangian dual functions is that they are often not differentiable. This requires an adaptation of the gradient method to consider subgradient information instead.

The challenge with using subgradients (instead of gradients) is that subgradients are not guaranteed to be descent directions (as opposed to gradients being the steepest descent direction under adequate norms). Nevertheless, for suitable step size choices, convergence can be observed. Figure 60 illustrates that subgradients are not necessarily descent directions.

Algorithm 12 summarises the subgradient method. Notice that the stopping criterion emulates the optimality condition $0 \in \partial\theta(w_k)$. However, in practice, one also enforces more heuristically driven criteria, such as a maximum number of iterations or a given number without observable improvement on the $\theta(w)$ value.

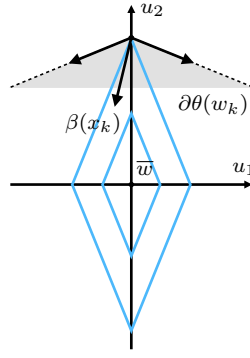


Figure 60: One possible subgradient $\beta(x_k)$ that is a descent direction for suitable step size. Notice that other subgradients without descent direction are available within the subdifferential $\partial\theta(w_k)$.

Algorithm 12 Subgradient method

- 1: **initialise.** tolerance $\epsilon > 0$, initial point w_0 , iteration count $k = 0$.
 - 2: **while** $\|\beta(x_k)\|_2 > \epsilon$ **do**
 - 3: $x_k \leftarrow \operatorname{argmin}_x \{\theta(w_k) = \inf_x \{f(x) + w_k^\top \beta(x)\}\}$
 - 4: $LB_k = \max\{LB_k, \theta(w_k)\}$
 - 5: update λ_k
 - 6: $w_{k+1} = w_k + \lambda_k \beta(x_k)$.
 - 7: $k \leftarrow k + 1$.
 - 8: **end while**
 - 9: **return** $LB_k = \theta(w_k)$.
-

One critical aspect of the subgradient method is the step size update described in Step 5 of Algorithm 12. Theoretical convergence is guaranteed if Step 5 generates a sequence $\{\lambda_k\}$ such that: $\sum_{k=0}^{\infty} \lambda_k \rightarrow \infty$ and $\lim_{k \rightarrow \infty} \lambda_k = 0$. However, discrepant performance can be observed for distinct parametrisation of the method.

The classical step update rule employed for the subgradient method is known as the Polyak rule, which gives

$$\lambda_{k+1} = \frac{\alpha_k (LB_k - \theta(w_k))}{\|\beta(x_k)\|^2}$$

with $\alpha_k \in (0, 2)$ and LB_k being the best-available lower-estimate of $\theta(\bar{w})$. The following result inspires this rule.

Proposition 8.10. Improving step size

If w_k is not optimal, then, for all optimal dual solutions \bar{w} , we have

$$\|w_{k+1} - \bar{w}\| < \|w_k - \bar{w}\|$$

for all step sizes λ_k such that

$$0 < \lambda_k < \frac{2(\theta(\bar{w}) - \theta(w_k))}{\|\beta(x_k)\|^2}.$$

Proof. We have that $\|w_{k+1} - \bar{w}\|^2 = \|w_k + \lambda_k \beta(x_k) - \bar{w}\|^2 =$

$$\|w_k - \bar{w}\|^2 - 2\lambda_k (\bar{w} - w_k)^\top \beta(x_k) + (\lambda_k)^2 \|\beta(x_k)\|^2.$$

By the subgradient inequality: $\theta(\bar{w}) - \theta(w_k) \leq (\bar{w} - w_k)^\top \beta(x_k)$. Thus

$$\|w_{k+1} - \bar{w}\|^2 \leq \|w_k - \bar{w}\|^2 - 2\lambda_k (\theta(\bar{w}) - \theta(w_k))^\top \beta(x_k) + (\lambda_k)^2 \|\beta(x_k)\|^2.$$

Parameterising the last two terms by $\gamma_k = \frac{\lambda_k \|\beta(x_k)\|^2}{\theta(\bar{w}) - \theta(w_k)}$ leads to

$$\|w_{k+1} - \bar{w}\|^2 \leq \|w_k - \bar{w}\|^2 - \frac{\gamma_k (2 - \gamma_k) (\theta(\bar{w}) - \theta(w_k))^2}{\|\beta(x_k)\|^2}.$$

Notice that if $0 < \lambda_k < \frac{2(\theta(\bar{w}) - \theta(w_k))}{\|\beta(x_k)\|^2}$ then $0 < \gamma_k < 2$ and, thus, $\|w_{k+1} - \bar{w}\| < \|w_k - \bar{w}\|$. 😊

In practice, since $\theta(\bar{w})$ is not known, it must be replaced by a proxy LB_k , which is chosen to be a lower bound on $\theta(\bar{w})$ to satisfy the subgradient inequality still. The α_k is then reduced from the nominal value 2 to correct for the estimation error of term $\theta(\bar{w}) - \theta(w_k)$.

9 Week IX

In this lecture, we look into the class of feasible direction methods, also known as primal methods.

9.1 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 observed throughout the solution process, they are referred to as *primal methods*. However, it depends on the geometry of the feasible region, and it might be so that the method allows for some infeasibility in the algorithm, as we will see later.

An *improving feasible direction* can be defined as follows.

Definition 9.1. Improving Feasible Direction

Consider the problem $\min. \{f(x) : x \in S\}$ with $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $\emptyset \neq S \subseteq \mathbb{R}^n$. A vector d is a feasible direction at $x \in S$ if 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 a $\delta > 0$ such that $f(x + \lambda d) < f(x)$ and $x + \lambda d \in S$ for $\lambda \in (0, \delta)$.

The key feature of feasible direction methods is deriving such directions and associated step sizes that retain feasibility, even if 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. Make $x^{k+1} = x^k + \lambda^k d^k$.

As a general rule, for a feasible direction method to perform satisfactorily, it must be that the calculation of the directions d^k and step sizes λ^k are simple enough. Often, these steps can be reduced to closed forms or, more frequently, solving linear or quadratic programming problems or even posing modified Newton systems.

9.2 Conditional gradient - the Frank-Wolfe method

The conditional gradient method is named as such due to the direction definition step, in which the direction d is selected such that the angle between the gradient $\nabla f(x)$ and d is as close to 180° degrees as the feasible region S allows.

Recall that, if $\nabla f(x^k)$ is a *descent direction*, then:

$$\nabla f(x^k)^\top (x - x^k) < 0 \text{ for } x \in S.$$

A straightforward way to obtain improving feasible directions $d = (x - x^k)$ is by solving the *direction search problem DS* of the form:

$$(DS) : \min. \{\nabla f(x^k)^\top (x - x^k) : x \in S\}.$$

Problem *DS* consists of finding the furthest feasible point in the direction of the gradient, that is, we move in the direction of the gradient, under the condition that we stop if the line search mandates so or that the search reaches the boundary of the feasible region. This is precisely what gives the name *conditional gradient*.

By letting $\bar{x}^k = \operatorname{argmin}_{x \in S} \{\nabla f(x^k)^\top (x - x^k)\}$ and obtaining $\lambda^k \in (0, 1]$ employing a line search, the method iterates making:

$$x^{k+1} = x^k + \lambda^k (\bar{x}^k - x^k).$$

One important condition to observe is that λ^k has to be constrained such that $\lambda \in (0, 1]$ to guarantee feasibility, as \bar{x}^k is feasible by definition. Also, notice that the condition $\nabla f(x^k) = 0$ might never be achieved since it might be so that the unconstrained optimum is outside the feasible region S . After two successive iterations, we will observe that $x^k = x^{k-1}$ and thus that $d^k = 0$. This eventual stall of the

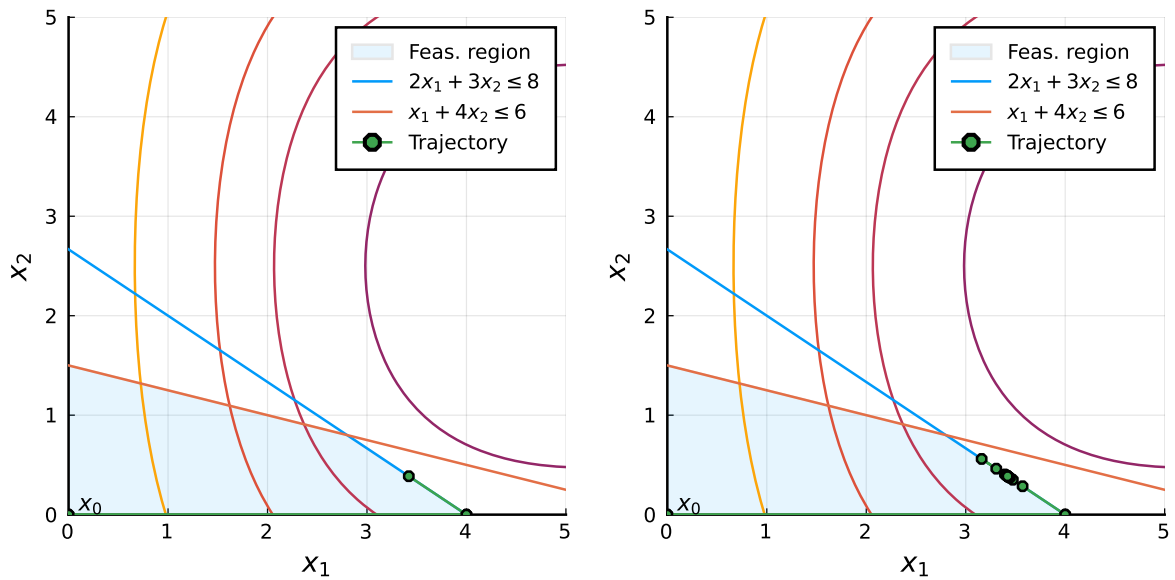


Figure 61: The Frank-Wolfe method applied to a problem with linear constraints. The algorithm takes 2 steps using an exact line search (left) and 15 with an Armijo line search (right).

algorithm will happen at a point x^k satisfying first-order (constrained) optimality conditions. Therefore, the term $\nabla f(x)^\top d^k$ will become zero regardless of whether the minimum of them function belongs to S and is hence used as the stopping condition of the algorithm. Algorithm 13 summarises the Frank-Wolfe method.

Algorithm 13 Frank-Wolfe method

- 1: **initialise.** $\epsilon > 0, x^0 \in S, k = 0$.
 - 2: **while** $|\nabla f(x)^\top d^k| > \epsilon$ **do**
 - 3: $\bar{x}^k = \operatorname{argmin}\{\nabla f(x^k)^\top d : x \in S\}$
 - 4: $d^k = \bar{x}^k - x^k$
 - 5: $\lambda^k = \operatorname{argmin}_\lambda \{f(x^k + \lambda d^k) : 0 \leq \lambda \leq \bar{\lambda}\}$
 - 6: $x^{k+1} = x^k + \lambda^k d^k$
 - 7: $k = k + 1$
 - 8: **end while**
 - 9: **return** x^k
-

Notice that the subproblems for a polyhedral feasibility set are linear programming problems, meaning that the Frank-Wolfe method can be restarted fairly efficiently using dual simplex at each iteration.

Figure 61 shows the employment of the FW method for optimising a nonlinear function within a polyhedral feasibility set. We consider the problem:

$$\begin{aligned} \min. & \quad e^{-(x_1-3)/2} + e^{(4x_2+x_1-20)/10} + e^{(-4x_2+x_1)/10} \\ \text{subject to:} & \quad 2x_1 + 3x_2 \leq 8 \\ & \quad x_1 + 4x_2 \leq 6, & \quad x_1, x_2 \geq 0 \end{aligned}$$

starting from $(0,0)$ and using an exact line search to set step sizes $\lambda \in [0,1]$. Notice that the method can be utilised with inexact line searches as well.

9.3 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 using Newton's method. It consists perhaps of the most general method for considering both nonlinear constraints and objective functions.

To see how that works, let us first consider an equality constraint problem P as:

$$P = \min. \{f(x) : h(x) = 0, i = 1, \dots, l\}.$$

The KKT conditions for P are given by the system $W(x, v)$ where:

$$W(x, v) = \begin{cases} \nabla f(x) + \sum_{i=1}^l v_i \nabla h_i(x) = 0 \\ h_i(x) = 0, i = 1, \dots, l \end{cases}$$

Using the Newton(-Raphson) method, we can solve $W(x, v)$. Starting from (x^k, v^k) , we can solve $W(x, v)$ by successively employing Newton steps of the form:

$$W(x^k, v^k) + \nabla W(x^k, v^k) \begin{bmatrix} x - x^k \\ v - v^k \end{bmatrix} = 0. \quad (61)$$

Upon closer inspection, one can notice that the term $\nabla W(x, v)$ is given by:

$$\nabla W(x^k, v^k) = \begin{bmatrix} \nabla^2 L(x^k, v^k) & \nabla h(x^k)^\top \\ \nabla h(x^k) & 0 \end{bmatrix},$$

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 . Now, setting $d = (x - x^k)$, we can rewrite (61) as:

$$\nabla^2 L(x^k, v^k) d + \nabla h(x^k)^\top v = -\nabla f(x^k) \quad (62)$$

$$\nabla h(x^k) d = -h(x^k), \quad (63)$$

which can be repeatedly solved until:

$$\|(x^k, v^k)^\top - (x^{k-1}, v^{k-1})^\top\| = 0,$$

i.e., convergence is observed. Then, (x^k, v^k) is a KKT point by definition.

This is fundamentally the underlying idea of SQP. However, the approach is taken under a more specialised setting. Instead of relying on Newton steps, we resort to successively solving quadratic sub-problems of the form:

$$QP(x^k, v^k) : \min. f(x^k) + \nabla f(x^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k, v^k) d \quad (64)$$

$$\text{subject to: } h_i(x^k) + \nabla h_i(x^k)^\top d = 0, i = 1, \dots, l. \quad (65)$$

Notice that QP is a linearly constrained quadratic programming problem, for which we have seen several solution approaches. Moreover, notice that the optimality conditions of QP are given by (62) and (63), where v is the dual variable associated with the constraints in (65), which, in turn, represent first-order approximations of the original constraints.

The objective function in QP can be interpreted as being a second-order approximation of $f(x)$ enhanced with the term $(1/2) \sum_{i=1}^l v_i^k d^\top \nabla^2 h_i(x^k) d$ that captures constraint curvature information.

An alternative interpretation for the objective function of QP is to notice that it consists of the second-order approximation of the Lagrangian function $L(x, v) = f(x) + \sum_{i=1}^l v_i h_i(x)$ at (x^k, v^k) , which is given by:

$$\begin{aligned} L(x, v) &\approx L(x^k, v^k) + \nabla_x L(x^k, v^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k, v^k) d \\ &= f(x_k) + v^{k^\top} h(x^k) + (\nabla f(x^k) + v^{k^\top} \nabla h(x^k))^\top d + \frac{1}{2} d^\top (\nabla^2 f(x^k) + \sum_{i=1}^l v_i^k \nabla^2 h_i(x^k)) d \end{aligned}$$

To see this, notice that terms $f(x^k)$, $v^{k^\top} h(x^k)$ are constants and that $\nabla h(x^k)^\top (x - x^k) = 0$ (from (65), as $h(x^k) = 0$).

The general subproblem in the SQP method can be stated as:

$$\begin{aligned}
 QP(x^k, u^k, v^k) : \min. \quad & \nabla f(x^k)^\top d + \frac{1}{2}d^\top \nabla^2 L(x^k, u^k, v^k)d \\
 \text{subject to: } & g_i(x^k) + \nabla g_i(x^k)^\top d \leq 0, i = 1, \dots, m \\
 & h_i(x^k) + \nabla h_i(x^k)^\top d = 0, i = 1, \dots, l,
 \end{aligned}$$

which includes inequality constraints $g_i(x) \leq 0$ for $i = 1, \dots, m$ in a linearised form and their respective associated Lagrangian multipliers u_i , for $i = 1, \dots, m$. This is possible since we use an optimisation setting rather than a Newton system that only allows for equality constraints, even though the latter can be obtained by introducing slack variables. Several options could be considered to handle this quadratic problem, including employing a primal/dual interior point method.

A pseudocode for the standard SQP method is presented in Algorithm 14.

Algorithm 14 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:    $d^k = \operatorname{argmin} QP(x^k, u^k, v^k)$ 
4:   obtain  $u^{k+1}, v^{k+1}$  from  $QP(x^k, u^k, v^k)$ 
5:    $x^{k+1} = x^k + d^k, k = k + 1.$ 
6: end while
7: return  $x^k.$ 

```

Notice that in Line 4, dual variable values are retrieved from the constraints in $QP(x^k, u^k, v^k)$. Therefore, $QP(x^k, u^k, v^k)$ needs to be solved by an algorithm that can return these dual variables, such as the (dual) simplex method.

Figure 62 illustrates the behaviour of the SQP method on the problem. Notice how the trajectory might eventually become infeasible due to the consideration of linear approximations of the nonlinear constraint.

$$\min. \{2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 : x_1^2 - x_2 \leq 0, x_1 + 5x_2 \leq 5, x_1 \geq 0, x_2 \geq 0\}$$

One important feature of the SQP method is that it closely mimics Newton's method's convergence properties; therefore, under appropriate conditions, superlinear convergence can be observed. Moreover, the BFGS method can be used to approximate $\nabla^2 L(x^k, v^k)$, which can turn the method dependent only on first-order derivatives.

Notice that because the constraints are considered implicitly in the subproblem $QP(x^k, u^k, v^k)$, 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 solutions that are far away from the optimum while presenting superior computational performance.

In essence, l_1 -SQP relies on a similar principle to 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 searches (since feasibility becomes encoded in the objective function with feasibility guaranteed at a minimum. cf. penalty methods) or relying on a trust region approach, ultimately preventing divergence issues.

Let us consider the trust-region based l_1 -penalty QP subproblem, which can be formulated as:

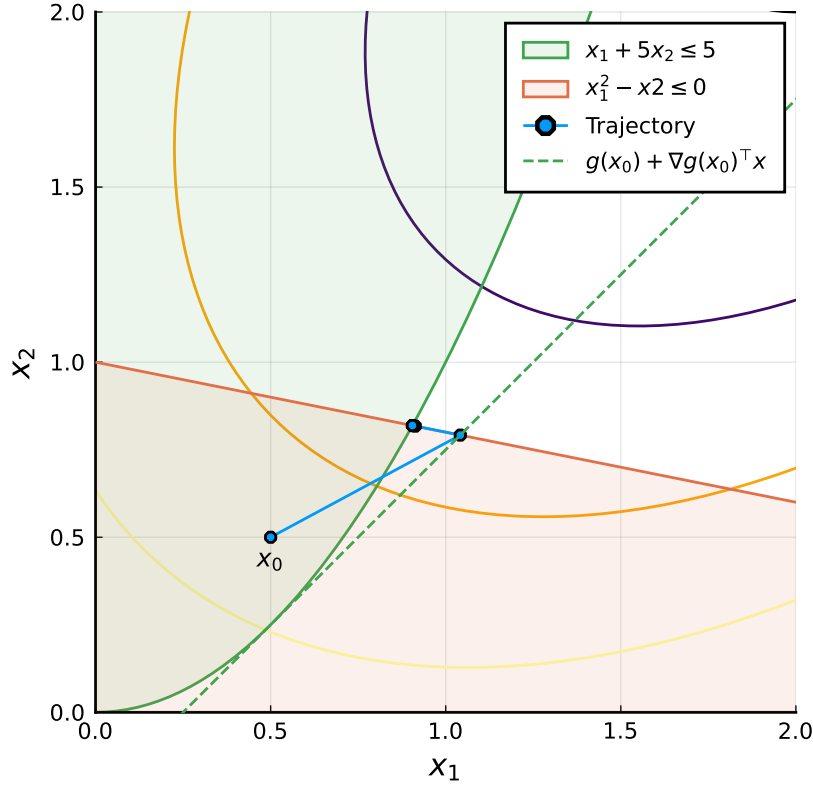


Figure 62: The SQP method converges in 6 iterations with $\epsilon = 10^{-6}$

$$\begin{aligned}
 & l_1 - QP(x^k, v^k) : \\
 & \min. \quad \nabla f(x^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k, v^k) d \\
 & \quad + \mu \left[\sum_{i=1}^m [g_i(x^k) + \nabla g_i(x^k)^\top d]^+ + \sum_{i=1}^l |h_i(x^k) + \nabla h_i(x^k)^\top d| \right] \\
 & \text{subject to: } -\Delta^k \leq d \leq \Delta^k,
 \end{aligned}$$

where μ is a penalty term, $[\cdot] = \max\{0, \cdot\}$, and Δ^k is a trust region term. This variant is of particular interest because the subproblem $l_1 - QP(x^k, v^k)$ can be recast as a QP problem with linear constraints:

$$\begin{aligned}
 & l_1 - QP(x^k, v^k) : \\
 & \min. \quad \nabla f(x^k)^\top d + \frac{1}{2} d^\top \nabla^2 L(x^k, v^k) d + \mu \left[\sum_{i=1}^m y_i + \sum_{i=1}^l (z_i^+ - z_i^-) \right] \\
 & \text{subject to: } -\Delta^k \leq d \leq \Delta^k \\
 & \quad y_i \geq g_i(x^k) + \nabla g_i(x^k)^\top d, i = 1, \dots, m \\
 & \quad z_i^+ - z_i^- = h_i(x^k) + \nabla h_i(x^k)^\top d, i = 1, \dots, l \\
 & \quad y, z^+, z^- \geq 0.
 \end{aligned}$$

The subproblem $l_1 - QP(x^k, v^k)$ enjoys the same benefits as the original form, meaning 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 a superlinear convergence rate, as the original SQP. The l_1 -penalty term is often called a *merit function* in the literature. Alternatively, one can disregard the trust region and employ a line search (exact or inexact), which would also enjoy globally convergent properties.

9.4 Generalised reduced gradient

The generalised reduced gradient method resembles, in many aspects, the simplex method for linear optimisation problems. It derives from Wolfe's reduced gradient. The term "reduced" refers to considering a reduced variable space formed by a subset of the decision variables.

9.4.1 Wolfe's reduced gradient

Let us consider the linearly constrained problem:

$$(P) : \min. f(x) \\ \text{subject to: } Ax = b \\ Ax \geq 0,$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ 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 the 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:

$$\begin{aligned} \nabla f(x)^\top d &= \nabla_B f(x)^\top d_B + \nabla_N f(x)^\top d_N \\ &= (\nabla_N f(x)^\top - \nabla_B f(x)^\top B^{-1}N) d_N \end{aligned} \quad (66)$$

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 expresses the gradient of the function in terms of the nonbasic directions only. Notice that the reduced gradient r resembles the reduced costs from the simplex method. In fact:

$$\begin{aligned} r^\top &= (r_B^\top, r_N^\top) = \nabla f(x)^\top - \nabla_B f(x)^\top B^{-1}A \\ &= (\nabla_B f(x)^\top - \nabla_B f(x)^\top B^{-1}B, \nabla_N f(x)^\top - \nabla_B f(x)^\top B^{-1}N) \\ &= (0, \nabla_N f(x)^\top - \nabla_B f(x)^\top B^{-1}N), \end{aligned}$$

and thus $\nabla f(x)^\top = r^\top d$.

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 to 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 guarantees the convergence of Wolfe's reduced gradient to a KKT point.

Theorem 9.2

Let \bar{x} be a feasible solution to P such that $\bar{x} = (\bar{x}_B, \bar{x}_N)$ and $x_B > 0$. Consider that A is decomposed into (B, N) . Let $r^\top = \nabla f(\bar{x})^\top - \nabla_B f(\bar{x})^\top B^{-1}A$ 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 \bar{x} is a KKT point.

Proof. d is a feasible direction by construction. Now, notice that:

$$\begin{aligned}\nabla f(\bar{x})^\top d &= \nabla_B f(\bar{x})^\top d_B + \nabla_N f(\bar{x})^\top d_N \\ &= [\nabla_N f(\bar{x})^\top - \nabla_B f(\bar{x})^\top B^{-1}N]d_N = \sum_{j \notin I_B} r_j d_j\end{aligned}$$

where I_B is the index set of basic variables. By construction (using Wolfe's rule), either $d = 0$ or $\nabla f(\bar{x})^\top d < 0$, being thus an *improvement direction*.

\bar{x} is a KKT point if and only if there exists $(u_B^\top, u_N^\top) \geq (0, 0)$ and v such that:

$$(\nabla_B f(x)^\top, \nabla_N f(x)^\top) + v^\top(B, N) - (u_B^\top, u_N^\top) = (0, 0) \quad (67)$$

$$u_B^\top x_B = 0, u_N^\top x_N = 0. \quad (68)$$

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 (67), $v^\top = -\nabla_B f(x)B^{-1}$. Substituting in the bottom row of (67), 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 15 presents a pseudocode for Wolfe's reduced gradient. A few implementation details stand out. First, notice that the basis is selected by choosing the largest components in value, which differs from the simplex method by allowing 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 15 Wolfe's reduced gradient method

- 1: **initialise.** $\epsilon > 0, x^0$ with $Ax^k = b, k = 0$, columns $a_j, j = 1, \dots, n$ of A
 - 2: **while** $\|d^k\| > \epsilon$ **do**
 - 3: $I^k =$ index set for m largest components of x^k
 - 4: Let $A = (B, N)$, where $B = \{a_j : j \in I^k\}$, and $N = \{a_j : j \notin I^k\}$
 - 5: $r^{k\top} = \nabla f(x^k)^\top - \nabla_B f(x^k)^\top B^{-1}A$
 - 6: $d_j^k = \begin{cases} -r_j^k, & \text{if } j \notin I^k \text{ and } r_j \leq 0; \\ -r_j x_j, & \text{if } j \notin I^k \text{ and } r_j > 0. \end{cases}$
 - 7: $d_B = -B^{-1}N d_N$
 - 8: **if** $d = 0$ **then**
 - 9: **return** x^k
 - 10: **end if**
 - 11: $\bar{\lambda} = \begin{cases} +\infty, & \text{if } d^k \geq 0; \\ \min\{x_j^k/d_j^k : d_j^k < 0\}, & \text{if } d^k < 0. \end{cases}$
 - 12: $\lambda^k = \operatorname{argmin}\{f(x^k + \lambda d^k) : 0 \leq \lambda \leq \bar{\lambda}\}$.
 - 13: $x^{k+1} = x^k + \lambda^k d^k; k = k + 1$.
 - 14: **end while**
 - 15: **return** x^k .
-

9.4.2 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 aims to recover feasibility via projection or 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 $d_B = -B^{-1}Sd_S$. Thus d becomes:

$$d = \begin{bmatrix} d_B \\ d_S \\ d_{N'} \end{bmatrix} = \begin{bmatrix} -B^{-1}S \\ I \\ 0 \end{bmatrix} d_S.$$

10 Week X

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.

10.1 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 differs from penalty functions in that they 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:

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

We define the *barrier problem* BP as:

$$(BP) : \inf_{\mu} \theta(\mu) \\ \text{subject to: } \mu > 0$$

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. In practice, the constraint $g(x) < 0$ can be dropped, as the barrier function automatically enforces them.

The barrier function $B : \mathbb{R}^m \rightarrow \mathbb{R}$ is such that:

$$B(x) = \sum_{i=1}^m \phi(g_i(x)), \text{ where } \begin{cases} \phi(y) \geq 0, & \text{if } y < 0; \\ \phi(y) = +\infty, & \text{when } y \rightarrow 0^-. \end{cases} \quad (69)$$

Examples of barrier functions that impose 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 defined as:

$$B(x) = -\sum_{i=1}^m \ln(-g_i(x)).$$

Figure 63 illustrates the behaviour of the barrier function. Ideally, the 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 assume infinite value if a solution is at the boundary $g(x) = 0$ or outside the feasible region. This is illustrated in the dashed line in Figure 63. 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.

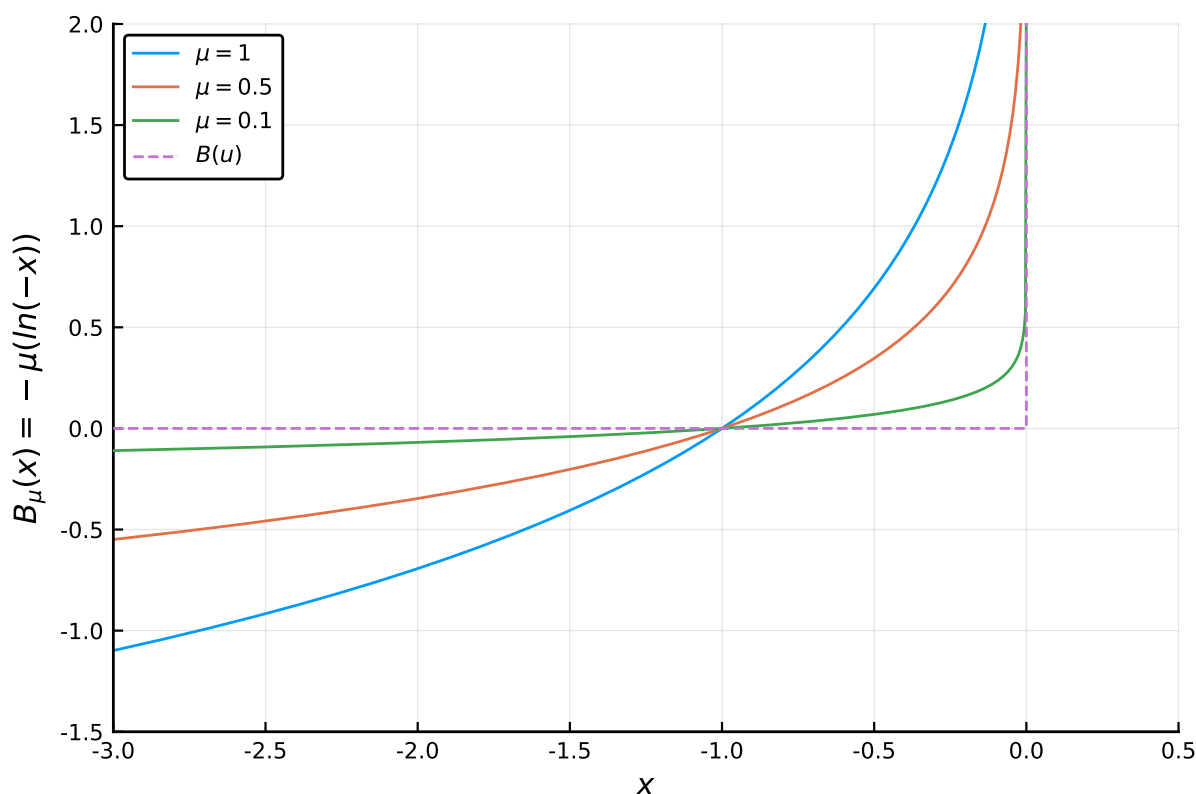


Figure 63: The barrier function for different values of μ

10.2 The barrier method

Similar to what was developed for penalty methods, we can devise a solution method that successively solves the barrier problem $\theta(\mu)$ for decreasing the barrier term μ .

We start by stating the result that guarantees convergence of the barrier method.

Theorem 10.1. Convergence of barrier methods

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $g : \mathbb{R}^n \rightarrow \mathbb{R}$ be continuous functions and $X \in \mathbb{R}^n$ a nonempty closed set in problem P . Suppose $\{x \in \mathbb{R}^n : g(x) < 0, x \in X\}$ is not empty. Let \bar{x} be the optimal solution of P such that, for any neighbourhood $N_\epsilon(\bar{x}) = \{x : \|x - \bar{x}\| \leq \epsilon\}$, there exists $x \in X \cap N_\epsilon$ for which $g(x) < 0$. Then:

$$\min\{f(x) : g(x) \leq 0, x \in X\} = \lim_{\mu \rightarrow 0^+} \theta(\mu) = \inf_{\mu > 0} \theta(\mu).$$

Letting $\theta(\mu) = f(x_\mu) + \mu B(x_\mu)$, where $B(x)$ is a barrier function as described in (69), $x_\mu \in X$ and $g(x_\mu) < 0$, the limit of $\{x_\mu\}$ is optimal to P and $\mu B(x_\mu) \rightarrow 0$ as $\mu \rightarrow 0^+$.

Proof. First, we show that $\theta(\mu)$ is a nondecreasing function in μ . For $\mu > \lambda > 0$ and x such that $g(x) < 0$ and $x \in X$, we have:

$$\begin{aligned} f(x) + \mu B(x) &\geq f(x) + \lambda B(x) \\ \inf_x \{f(x) + \mu B(x)\} &\geq \inf_x \{f(x) + \lambda B(x)\} \\ \theta(\mu) &\geq \theta(\lambda). \end{aligned}$$


From these, we conclude that $\lim_{\mu \rightarrow 0^+} \theta(\mu) = \inf\{\theta(\mu) : \mu > 0\}$. Now, let $\epsilon > 0$. As \bar{x} is optimal, by assumption there exists some $\hat{x} \in X$ with $g(\hat{x}) < 0$ such that $f(\bar{x}) + \epsilon > f(\hat{x})$. Then, for $\mu > 0$ we have:

$$f(\bar{x}) + \epsilon + \mu B(\hat{x}) > f(\hat{x}) + \mu B(\hat{x}) \geq \theta(\mu).$$

Letting $\mu \rightarrow 0^+$, it follows that $f(\bar{x}) + \epsilon \geq \lim_{\mu \rightarrow 0^+} \theta(\mu)$, which implies $f(\bar{x}) \geq \lim_{\mu \rightarrow 0^+} \theta(\mu) =$

$\inf\{\theta(\mu) : \mu > 0\}$. Conversely, since $B(x) \geq 0$ and $g(x) < 0$ for some $\mu > 0$, we have:

$$\begin{aligned}\theta(\mu) &= \inf\{f(x) + \mu B(x) : g(x) < 0, x \in X\} \\ &\geq \inf\{f(x) : g(x) < 0, x \in X\} \\ &\geq \inf\{f(x) : g(x) \leq 0, x \in X\} = f(\bar{x}).\end{aligned}$$

Thus $f(\bar{x}) \leq \lim_{\mu \rightarrow 0^+} \theta(\mu) = \inf\{\theta(\mu) : \mu > 0\}$. Therefore, $f(\bar{x}) = \lim_{\mu \rightarrow 0^+} \theta(\mu) = \inf\{\theta(\mu) : \mu > 0 \geq\}$. 

The proof has three main steps. First, we show that $\theta(\mu)$ is a nondecreasing function in μ , which implies that $\lim_{\mu \rightarrow 0^+} \theta(\mu) = \inf\{\theta(\mu) : \mu > 0\}$. This can be trivially shown as only feasible solutions x must be considered.

Next, we show the convergence of the barrier method by showing that $\inf_{\mu > 0} \theta(\mu) = f(\bar{x})$, where $\bar{x} = \operatorname{argmin}\{f(x) : g(x) \leq 0, x \in X\} = \lim_{\mu \rightarrow 0^+} \theta(\mu) = \inf_{\mu > 0} \theta(\mu)$. The optimality of \bar{x} implies that $f(\hat{x}) - f(\bar{x}) < \epsilon$ for feasible \hat{x} and $\epsilon > 0$. Moreover, $B(\hat{x}) \geq 0$ by definition. In the last part, we argue that including the boundary can only improve the objective function value, leading to the last inequality. It is worth highlighting that, to simplify the proof, we have assumed that the barrier function has the form described in (69). However, a proof in the veins of Theorem 10.1 can still be developed for the Frisch log barrier (for which $B(x)$ is not necessarily nonnegative) since, essentially, (69) only needs to be observed in a neighbourhood of $g(x) = 0$.

The result in Theorem 10.1 allows the design of an optimisation method that, starting from a strictly feasible (interior) solution, is based on successively reducing the barrier term until a solution with an arbitrarily small barrier term is obtained. Algorithm 16 present a pseudo code for such a method.

Algorithm 16 Barrier method

- 1: **initialise.** $\epsilon > 0, x^0 \in X$ with $g(x^k) < 0, \mu^k, \beta \in (0, 1), k = 0$.
 - 2: **while** $\mu^k B(x^k) > \epsilon$ **do**
 - 3: $\bar{x}^{k+1} = \operatorname{argmin}\{f(x) + \mu^k B(x) : x \in X\}$
 - 4: $\mu^{k+1} = \beta \mu^k, k = k + 1$
 - 5: **end while**
 - 6: **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 obtain. This characteristic renders the name *interior point methods* for this class of algorithms.

Consider the following example. Let $P = \{(x + 1)^2 : x \geq 0\}$. Let us assume we use the barrier function $B(x) = -\ln(x)$. Then, the unconstrained barrier problem becomes:

$$(BP) : \min_x (x + 1)^2 - \mu \ln(x). \tag{70}$$

Since this is a convex function, the first-order condition $f'(x) = 0$ is necessary and sufficient for optimality. Thus, solving $2(x + 1) - \frac{\mu}{x} = 0$ we obtain the positive root and unique solution $x_\mu = \frac{-1}{2} + \frac{\sqrt{4+8\mu}}{4}$. Figure X shows the behaviour of the problem as μ converges to zero. As can be seen, as $\mu \rightarrow 0$, the optimal solution x_μ converges to the constrained optimum $\bar{x} = 0$.

We now consider a more involved example. Let us consider the problem:

$$P = \min. \{(x_1 - 2)^4 + (x_1 - 2x_2)^2 : x_1^2 - x_2 \leq 0\}$$

with $B(x) = -\frac{1}{x_1^2 - x_2}$. We implemented Algorithm 16 and solved it with two distinct values for the penalty term μ and reduction term β . Figure 64 illustrates the trajectory of the algorithm with each parameterisation, exemplifying how these can affect the convergence of the method.

10.3 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. In the last

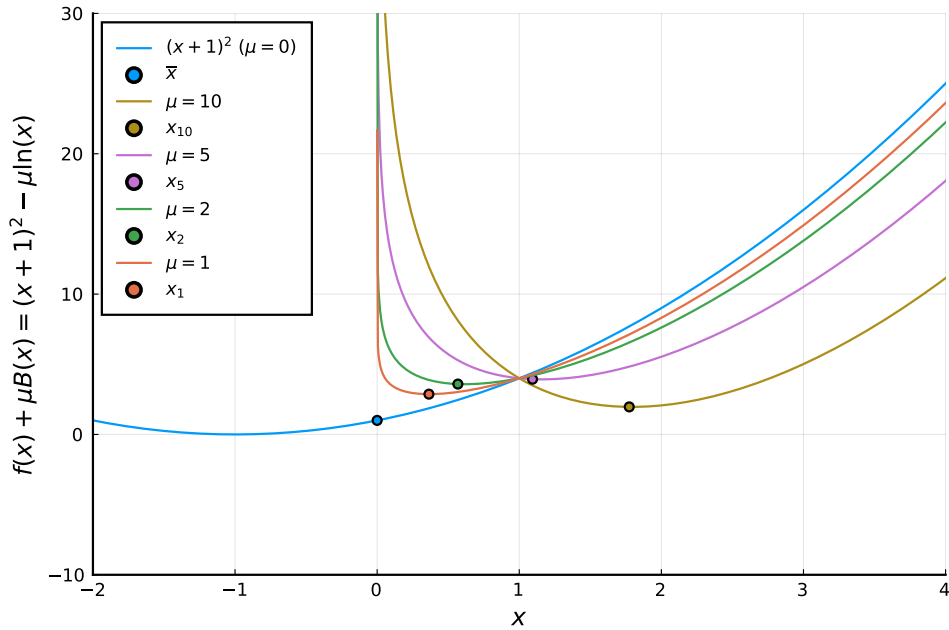


Figure 64: Example 1: solving a one-dimensional problem with the barrier method

decade, the primal-dual interior point method has become the algorithm of choice for many applications 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 an LP primal P :

$$(P) : \min. \quad c^\top x$$

$$\text{subject to: } Ax = b \quad : v$$

$$x \geq 0 \quad : u$$

and its respective dual formulation D :

$$(D) : \max. \quad b^\top v$$

$$\text{subject to: } A^\top v + u = c$$

$$u \geq 0, v \in \mathbb{R}^m.$$

The optimal solution $(\bar{x}, \bar{v}, \bar{u}) = \bar{w}$ is such that it satisfies KKT conditions of P , given by:

$$Ax = b, \quad x \geq 0$$

$$A^\top v + u = c, \quad u \geq 0, \quad v \in \mathbb{R}^m$$

$$u^\top x = 0.$$

These are going to be useful as a reference for the next developments. We start by considering the *barrier problem* for P by using the logarithmic barrier function to represent the condition $x \geq 0$. Thus, the barrier problem BP can be defined as:

$$(BP) : \min. \quad c^\top x - \mu \sum_{i=1}^n \ln(x_i)$$

$$\text{subject to: } Ax = b.$$

Notice that this problem is a strictly equality-constrained problem that can 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). Moreover, observe that the KKT conditions of BP are:

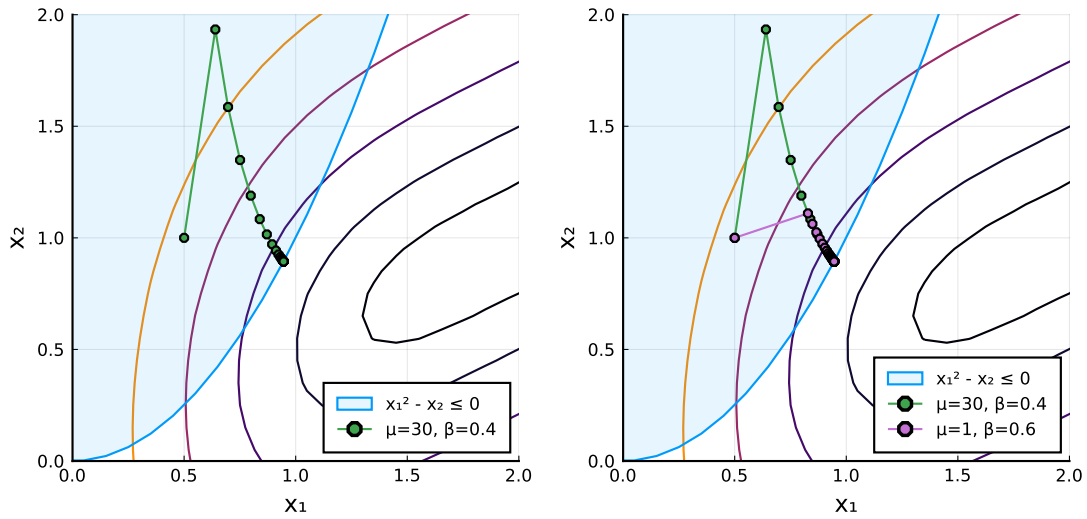


Figure 65: The trajectory of the barrier method for problem P . Notice how the parameters influence the trajectory and number of iterations. The parameters on the left require 27 iterations, while those on the right require 40 for convergence.

$$Ax = b, \quad x > 0$$

$$A^\top v = c - \mu \left(\frac{1}{x_1}, \dots, \frac{1}{x_n} \right)$$

Notice that, since $\mu > 0$ and $x > 0$, $u = \mu \left(\frac{1}{x_1}, \dots, \frac{1}{x_n} \right)$ serves as an estimate for the Lagrangian dual variables. To further understand the relationship between the optimality 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 $e = [1, \dots, 1]^\top$ be a vector of ones of suitable dimension. We can rewrite the KKT conditions of BP as:

$$Ax = b, \quad x > 0 \tag{71}$$

$$A^\top v + u = c \tag{72}$$

$$u = \mu X^{-1} e \Rightarrow X U e = \mu e. \tag{73}$$

Notice how the condition (73) resembles the complementary slackness from P , but *relaxed* to be μ instead of zero. This system is often called the *perturbed KKT system*.

Theorem 10.1³ guarantees that $w_\mu = (x_\mu, v_\mu, u_\mu)$ approaches the optimal primal-dual solution of P as $\mu \rightarrow 0^+$. The trajectory formed by successive solutions $\{w_\mu\}$ is called the *central path* due to the interiority enforced by the barrier function. When the barrier term μ is large enough, the solution of the barrier problem is close to the analytic centre of the feasibility set. The analytic centre of a polyhedral set $S = \{x \in \mathbb{R}^n : Ax \leq b\}$ is given by:

$$\begin{aligned} & \max_x \prod_{i=1}^m (b_i - a_i^\top x) \\ & \text{subject to: } x \in X, \end{aligned}$$

Which corresponds to finding the point of maximum distance to each of the hyperplanes forming the

³In fact, we require a slight variant of Theorem 1 that allows for $B(x) \geq 0$ only being required in a neighbourhood of $g(x) = 0$.

polyhedral set. This is equivalent to the convex problem:

$$\begin{aligned} \min_x \quad & \sum_{i=1}^m -\ln(b_i - a_i^\top x) \\ \text{subject to: } & x \in X, \end{aligned}$$

And thus justifying the nomenclature.

One aspect should be observed for defining the stopping criterion. Notice that the term $u^\top x$ is such that it measures the duality gap at a given solution. That is, notice that:

$$\begin{aligned} c^\top x &= (A^\top v + u)^\top x \\ &= (A^\top v)^\top x + u^\top x \\ &= v^\top (Ax) + u^\top x \\ c^\top x - b^\top v &= u^\top x = \sum_{i=1}^n u_i x_i = \sum_{i=1}^n \left(\frac{\mu}{x_i}\right) x_i = n\mu. \end{aligned}$$

This gives the *total slack violation* that can be used to determine the algorithm's convergence.

10.3.1 Primal/dual path-following interior point method

The primal/dual path-following interior point method (IPM) is the specialised version of the setting described earlier for solving LP/QP problems.

It consists of building upon employing logarithmic barriers to LP/QP problems and solving the system (71) - (73) using Newton's method. However, instead of solving the problem to optimality for each μ , only a *single* Newton step is taken before the barrier term μ is reduced.

Suppose we start with a $\bar{\mu} > 0$ and a $w^k = (x^k, v^k, u^k)$ sufficiently close to $w_{\bar{\mu}}$. Then, for a sufficiently small $\beta \in (0, 1)$, $\beta\bar{\mu}$ will lead to a w^{k+1} sufficiently close to $w_{\beta\bar{\mu}}$. Figure 66 illustrates this effect, showing how a suboptimal solution x^k does not necessarily need to be in the central path (denoted by the dashed line) to guarantee convergence, as long as they are guaranteed to remain within the same neighbourhood $N_\mu(\theta)$ of the central path.

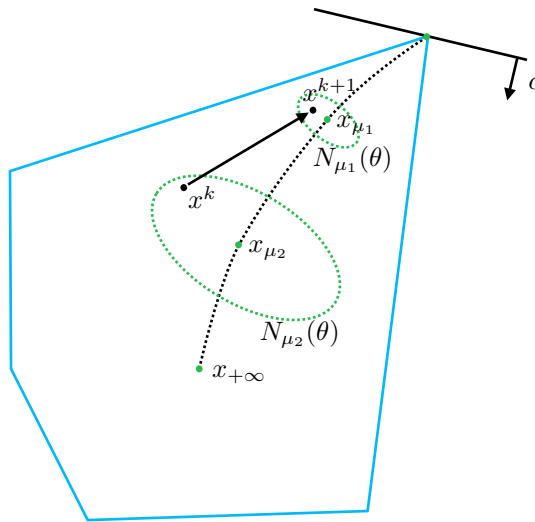


Figure 66: an illustrative representation of the central path and approximately how the IPM follows it.

For example, let $N_\mu(\theta) = \{x \mid \|X_\mu U_\mu e - \mu e\| \leq \theta\mu\}$. Then, by selecting $\beta = 1 - \frac{\sigma}{\sqrt{n}}$, $\sigma = \theta = 0.1$, and $\mu^0 = (x^\top u)/n$, successive Newton steps are guaranteed to remain within $N_\mu(\theta)$.

To see how the setting works, let the perturbed KKT system (71) - (73) for each $\hat{\mu}$ be denoted as

$H(w) = 0$. Let $J(\bar{w})$ be the Jacobian of $H(w)$ at \bar{w} .

Applying Newton's method to solve $H(w) = 0$ for \bar{w} , we obtain:

$$J(\bar{w})d_w = -H(\bar{w}) \quad (74)$$

where $d_w = (w - \bar{w})$. By rewriting $d_w = (d_x, d_v, d_u)$, (74) can be equivalently stated as:

$$\begin{bmatrix} A & 0^\top & 0 \\ 0 & A^\top & I \\ \bar{U} & 0^\top & \bar{X} \end{bmatrix} \begin{bmatrix} d_x \\ d_v \\ d_u \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \hat{\mu}e - \bar{X}\bar{U}e \end{bmatrix}. \quad (75)$$

The system (75) is often called the Newton's system.

In practice, the updates incorporate primal and dual infeasibility, which precludes the need for additional mechanisms to guarantee primal and dual feasibility throughout the algorithm. This can be achieved with a simple modification in the Newton system, rendering the direction update step:

$$\begin{bmatrix} A & 0^\top & 0 \\ 0 & A^\top & I \\ U^k & 0^\top & X^k \end{bmatrix} \begin{bmatrix} d_x^{k+1} \\ d_v^{k+1} \\ d_u^{k+1} \end{bmatrix} = - \begin{bmatrix} Ax^k - b \\ A^\top v^k + u^k - c \\ X^k U^k e - \mu^{k+1} e \end{bmatrix}, \quad (76)$$

To see how this still leads to primal and dual feasible solutions, consider the primal residuals (i.e., the amount of infeasibility) as $r_p(x, u, v) = Ax - b$ and the dual residuals $r_d(x, u, v) = A^\top v + u - c$. Now, let $r(w) = r(x, u, v) = (r_p(x, u, v), r_d(x, u, v))$, recalling that $w^k = (x, v, u)$. The optimality conditions can be expressed as requiring that the residuals vanish, that is $r(\bar{w}) = 0$.

Now, consider the first-order Taylor approximation for r at w for a step d_w :

$$r(w + d_w) \approx r(w) + Dr(w)d_w,$$

Where $Dr(w)$ is the derivative of r evaluated at w , given by the two first rows of the Newton system (75). The step d_w for which the residue vanishes is:

$$Dr(w)d_w = -r(w), \quad (77)$$

Which is the same as (61) without the bottom equation. Now, if we consider the directional derivative of the square of the norm of r in the direction d_w , we obtain:

$$\left. \frac{d}{dt} \|r(w + td_w)\|_2^2 \right|_{t \rightarrow 0^+} = 2r(w)^\top Dr(w)d_w = -2r(w)^\top r(w), \quad (78)$$

Which is strictly decreasing. That is, the step d_w is such that it will make the residual decrease and eventually become zero. From that point onwards, the Newton system will take the form of (75).

The algorithm proceeds by iteratively solving the system (76) with $\mu^{k+1} = \beta\mu^k$ with $\beta \in (0, 1)$ until $n\mu^k$ is less than a specified tolerance. Algorithm 17 summarises a simplified form of the IPM.

Algorithm 17 Interior point method (IPM) for LP

- 1: **initialise.** primal-dual feasible w^k , $\epsilon > 0$, μ^k , $\beta \in (0, 1)$, $k = 0$.
 - 2: **while** $n\mu = c^\top x^k - b^\top v^k > \epsilon$ **do**
 - 3: compute $d_{w^{k+1}} = (d_{x^{k+1}}, d_{v^{k+1}}, d_{u^{k+1}})$ using (76) and w^k .
 - 4: $w^{k+1} = w^k + d_{w^{k+1}}$
 - 5: $\mu^{k+1} = \beta\mu^k$, $k = k + 1$
 - 6: **end while**
 - 7: **return** w^k .
-

Figure 67 illustrates the behaviour of the IPM when employed to solve the linear problem:

$$\begin{aligned} & \min. x_1 + x_2 \\ & \text{subject to: } 2x_1 + x_2 \geq 8 \\ & \quad \quad \quad x_1 + 2x_2 \geq 10, \\ & \quad \quad \quad x_1, x_2 \geq 0 \end{aligned}$$

Considering two distinct initial penalties μ . Notice how higher penalty values enforce a more central convergence of the method.

Some points are worth noticing concerning Algorithm 17. First, notice that a fixed step size is consid-

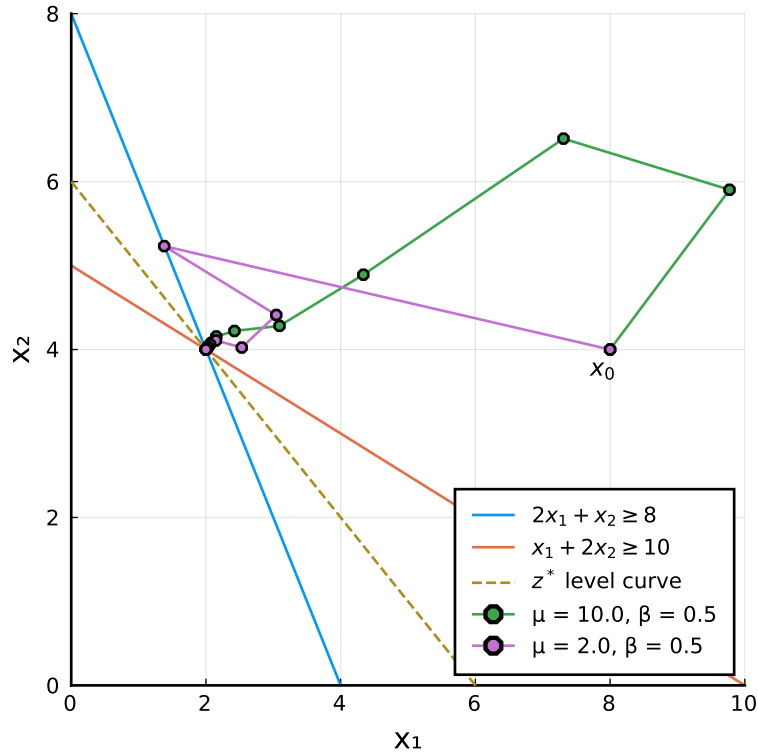


Figure 67: IPM applied to an LP problem with two different barrier terms

ered in Line 4. A line search can be incorporated to prevent infeasibility and improve numerical stability. Typically, it is used $\lambda_i^k = \min\{\alpha, -\frac{x_i^k}{d_i^k}\}$ with $\alpha < 1$ but close to 1.

Also, even though the algorithm is initialised with a feasible solution w^k , this might, in practice, not be necessary. 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 $\bar{X}\bar{U}e$.

Let $\Delta X = \text{diag}(d_x^{\text{pred}})$ and $\Delta U = \text{diag}(d_u^{\text{pred}})$.
Then:

$$\begin{aligned} (X + \Delta X)(U + \Delta U)e &= XUe + (U\Delta X + X\Delta U)e + \Delta X\Delta Ue \\ &= XUe + (0 - XUe) + \Delta X\Delta Ue \\ &= \Delta X\Delta Ue \end{aligned} \tag{79}$$

Using the last equation (79), the corrector Newton step becomes $\bar{U}d_x + \bar{X}d_u = \hat{\mu}e - \Delta X\Delta Ue$. Finally, d_w^k is set to be a combination of d_w^{pred} and d_w^{cor} .

11 Week XI

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.

11.1 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 solutions.

Consider the problem $(P) : \min. \{f(x) : g(x) \leq 0, h(x) = 0, x \in X\}$. A *penalised version* of P is given by:

$$(P_\mu) : \min. \{f(x) + \mu\alpha(x) : x \in X\},$$

where $\mu > 0$ is a *penalty term* and $\alpha(x) : \mathbb{R}^n \mapsto \mathbb{R}$ is a *penalty function* of the form:

$$\alpha(x) = \sum_{i=1}^m \phi(g_i(x)) + \sum_{i=1}^l \psi(h_i(x)). \quad (80)$$

For $\alpha(x)$ to be a suitable penalty function, one must observe that $\phi : \mathbb{R} \mapsto \mathbb{R}$ and $\psi : \mathbb{R} \mapsto \mathbb{R}$ are continuous and satisfy:

$$\begin{aligned} \phi(y) &= 0 \text{ if } y \leq 0 \text{ and } \phi(y) > 0 \text{ if } y > 0 \\ \psi(y) &= 0 \text{ if } y = 0 \text{ and } \psi(y) > 0 \text{ if } y \neq 0. \end{aligned}$$

Typical options are $\phi(y) = ([y]^+)^p$ with $p \in \mathbb{Z}_+$ and $\psi(y) = |y|^p$ with $p = 1$ or $p = 2$.

Figure 68 illustrates the solution of $(P) : \min. \{x_1^2 + x_2^2 : x_1 + x_2 = 1, x \in \mathbb{R}^2\}$ using a penalty-based approach. Using $\alpha(x_1, x_2) = (x_1 + x_2 - 1)^2$, the penalised auxiliary problem P_μ becomes $(P_\mu) : \min. \{x_1^2 + x_2^2 + \mu(x_1 + x_2 - 1)^2 : x \in \mathbb{R}^2\}$. Since f_μ is convex and differentiable, necessary and sufficient optimality conditions $\nabla f_\mu(x) = 0$ imply:

$$\begin{aligned} x_1 + \mu(x_1 + x_2 - 1) &= 0 \\ x_2 + \mu(x_1 + x_2 - 1) &= 0, \end{aligned}$$

which gives $x_1 = x_2 = \frac{\mu}{2\mu+1}$.

One can notice that, as μ increases, the solution of the unconstrained penalised problem, represented by the level curves, becomes closer to the optimal of the original constrained problem P , represented by the dot on the hyperplane defined by $x_1 + x_2 = 1$.

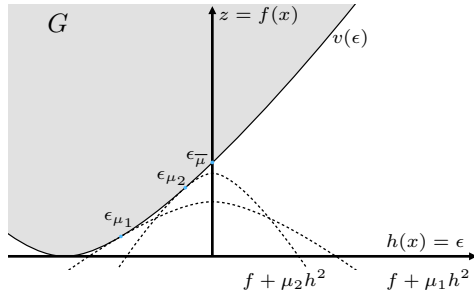


Figure 69: Geometric representation of penalised problems in the mapping $G = [h(x), f(x)]$

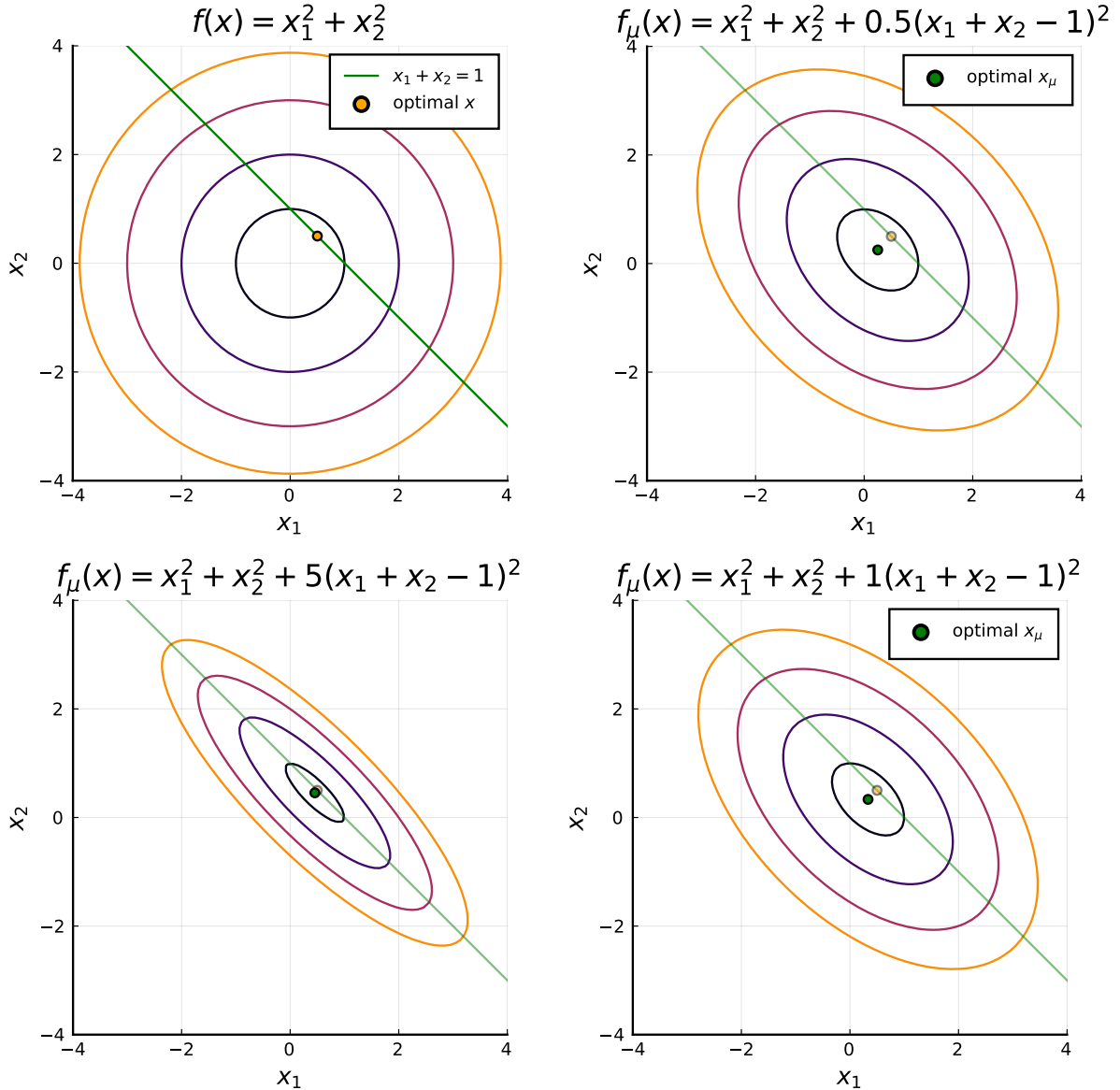


Figure 68: Solving the constrained problem P (top left) by gradually increasing the penalty term μ (0.5, 1, and 5, in clockwise order)

11.1.1 Geometric interpretation

A similar geometrical analysis to that performed with the Lagrangian duals can be employed for understanding how penalised problems can obtain optimal solutions. For that, let us the problem from the previous example (P) : $\min. \{x_1^2 + x_2^2 : x_1 + x_2 = 1, x \in \mathbb{R}^2\}$. Let $G : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ be a mapping $\{[h(x), f(x)] : x \in \mathbb{R}^2\}$, and let $v(\epsilon) = \min. \{x_1^2 + x_2^2 : x_1 + x_2 - 1 = \epsilon, x \in \mathbb{R}^2\}$. The optimal solution is $x_1 = x_2 = \frac{1+\epsilon}{2}$ with $v(\epsilon) = \frac{(1+\epsilon)^2}{2}$.

Minimising $f(x) + \mu(h(x)^2)$ consists of moving the curve downwards until a single contact point ϵ_μ remains. One can notice that, as $\mu \rightarrow \infty$, $f + \mu h$ becomes sharper ($\mu_2 > \mu_1$), and ϵ_μ converges to the optimum ϵ_μ . Figure 69 illustrates this behaviour.

The shape of the penalised problem curve is due to the following. First, notice that:

$$\begin{aligned} & \min_x \{f(x) + \mu \sum_{i=1}^l (h_i(x))^2\} \\ &= \min_{x, \epsilon} \{f(x) + \mu \|\epsilon\|^2 : h_i(x) = \epsilon, i = 1, \dots, l\} \\ &= \min_\epsilon \{\mu \|\epsilon\|^2 + \min_x \{f(x) : h_i(x) = \epsilon, i = 1, \dots, l\}\} \\ &= \min_\epsilon \{\mu \|\epsilon\|^2 + v(\epsilon)\}. \end{aligned}$$

Consider $l = 1$, and let $x_\mu = \mathbf{argmin}_x \{f(x) + \mu \sum_{i=1}^l (h_i(x))^2\}$ with $h(x_\mu) = \epsilon_\mu$, implying that $\epsilon_\mu = \mathbf{argmin}_\epsilon \{\mu \|\epsilon\|^2 + v(\epsilon)\}$. Then, the following holds:

1. $f(x_\mu) + \mu(h(x_\mu))^2 = \mu\epsilon_\mu^2 + v(\epsilon_\mu) \Rightarrow f(x_\mu) = v(\epsilon_\mu)$, since $h(x_\mu) = \epsilon_\mu$;
2. and $v'(\epsilon_\mu) = \frac{\partial}{\partial \epsilon} (f(x_\mu) + \mu(h(x_\mu))^2 - \mu\epsilon_\mu^2) = -2\mu\epsilon_\mu$.

Therefore, $(h(x_\mu), f(x_\mu)) = (\epsilon_\mu, v(\epsilon_\mu))$. Denoting $f(x_\mu) + \mu h(x_\mu)^2 = k_\mu$, we see the parabolic function $f = k_\mu - \mu\epsilon^2$ matching $v(\epsilon_\mu)$ for $\epsilon = \epsilon_\mu$ and has the slope $-2\mu\epsilon$, matching that of $v(\epsilon)$ at that point.

11.1.2 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:

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

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 (80). For that to be possible, we need first to state a convergence result guaranteeing that:

$$\min \{f(x) : g(x) \leq 0, h(x) = 0, x \in X\} = \sup_{\mu \geq 0} \theta(\mu) = \lim_{\mu \rightarrow \infty} \theta(\mu).$$

In practice, that would mean that μ_k can be increased at each iteration k until a suitable tolerance is achieved. Theorem 11.1 states the convergence of penalty based methods.

Theorem 11.1. Convergence of penalty-based methods

Consider the problem P , where f, g_i for $i = 1, \dots, m$, and h_i for $i = 1, \dots, l$ are continuous, and $X \subset \mathbb{R}^n$ a compact set. Suppose that, for each μ , there exists $x_\mu = \mathbf{argmin} \{f(x) + \mu\alpha(x) : x \in X\}$, where α is a suitable penalty function and $\{x_\mu\}$ is contained within X . Then:

$$\min_x \{f(x) : g(x) \leq 0, h(x) = 0, x \in X\} = \sup_{\mu \geq 0} \{f(x_\mu) + \mu\alpha(x_\mu)\} = \lim_{\mu \rightarrow \infty} \theta(\mu),$$

where $\theta(\mu) = \min_x \{f(x) + \mu\alpha(x) : x \in X\} = f(x_\mu) + \mu\alpha(x_\mu)$. Also, the limit of any convergent subsequence of $\{x_\mu\}$ is optimal to the original problem and $\mu\alpha(x_\mu) \rightarrow 0$ as $\mu \rightarrow \infty$.

Proof. We first show that $\theta(\mu)$ are nondecreasing function of μ . Let $0 < \lambda < \mu$. From the definition of $\theta(\mu)$, we have that:

$$f(x_\mu) + \lambda\alpha(x_\mu) \geq f(x_\lambda) + \lambda\alpha(x_\lambda) \quad (81)$$

Adding and subtracting $\mu\alpha(x_\mu)$ in the left side of (81), we conclude that $\theta(\mu) \geq \theta(\lambda)$. Now, for $x \in X$ with $g(x) \leq 0$ and $h(x) = 0$, notice that $\alpha(x) = 0$. This implies that:


$$f(x) = f(x) + \mu\alpha(x) \geq \inf_x \{f(x) + \mu\alpha(x) : x \in X\} = \theta(\mu) \quad (82)$$

and, therefore, $\theta(\mu)$ is bounded above, and thus $\sup_{\mu \geq 0} \theta(\mu) = \lim_{\mu \rightarrow \infty} \theta(\mu)$. For that to be the case, we must have that $\mu\alpha(x_\mu) \rightarrow 0$ as $\mu \rightarrow \infty$. Moreover, we notice from (82) that:

$$\min_x \{f(x) : g(x) \leq 0, h(x) = 0, x \in X\} \geq \lim_{\mu \rightarrow \infty} \theta(\mu). \quad (83)$$

On the other hand, take any convergent subsequence $\{x_{\mu_k}\}$ of $\{x_\mu\}_{\mu \rightarrow \infty}$ with limit \bar{x} . Then:

$$\sup_{\mu \geq 0} \theta(\mu) \geq \theta(\mu_k) = f(x_{\mu_k}) + \mu\alpha(x_{\mu_k}) \geq f(x_{\mu_k}).$$

Since $x_{\mu_k} \rightarrow \bar{x}$ as $\mu \rightarrow \infty$ and f is continuous, this implies that $\sup_{\mu \geq 0} \theta(\mu) \geq f(\bar{x})$. Combined with (83), we have that $f(\bar{x}) = \sup_{\mu \geq 0} \{\theta(\mu)\}$ and thus the result follows. 

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 infeasibility $\alpha(x)$ behave as we increase the penalty coefficient μ . For that, notice that using the same trick 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. $f(x_\lambda) + \mu\alpha(x_\lambda) \geq f(x_\mu) + \mu\alpha(x_\mu)$.

Notice that in 1, we use the fact that $x_\lambda = \mathbf{argmin}_x \theta(\lambda) = \mathbf{argmin}_x \{f(x) + \lambda\alpha(x)\}$ and therefore, must be less or equal then $f(x_\mu) + \lambda\alpha(x_\mu)$ for an arbitrary $x_\mu \in X$. The same logic is employed in 2, but reversed in 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 at 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 $\mu \rightarrow \infty$. Though this is a strong assumption, it tends to not be so restrictive in practical cases, since variables typically lie within finite lower and upper bounds. Finally, notice that $\alpha(\bar{x}) = 0$ implies that \bar{x} is feasible for g_i for $i = 1, \dots, m$, and h_i for $i = 1, \dots, l$, and thus optimal for P . This is stated in the following corollary.

Corollary 11.2. If $\alpha(x_\mu) = 0$ for some μ , then x_μ is optimal for P .

Proof. If $\alpha(x_\mu) = 0$, then x_μ is feasible. Moreover, x_μ is optimal, since

$$\begin{aligned} \theta(\mu) &= f(x_\mu) + \mu\alpha(x_\mu) \\ &= f(x_\mu) \leq \inf \{f(x) : g(x) \leq 0, h(x) = 0, x \in X\}. \end{aligned} \quad \text{smiley face}$$

A technical detail of the proof of Theorem 11.1 is that the convergence of such approach is asymptotically, i.e., by making μ arbitrarily large, x_μ can be made arbitrarily close to the true optimal \bar{x} and $\theta(\mu)$ can be made arbitrarily close to the optimal value $f(\bar{x})$. 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 $\kappa = \frac{\max_{i=1, \dots, n} \lambda_i}{\min_{i=1, \dots, n} \lambda_i}$, where $\{\lambda_i\}_{i=1, \dots, n}$ are the eigenvalues of $H_\mu(x_\mu)$. Since the influence is only on some of the eigenvalues, this affects the conditioning of the problem and might lead to numerical instabilities. An indication of that can be seen in Figure 68, where

one can notice the elongated profile of the function as the penalty term μ increases.

Consider the following example. Let the penalised function $f_\mu(x) = x_1^2 + x_2^2 + \mu(x_1 + x_2 - 1)^2$.

The Hessian of $f_\mu(x)$ is:

$$\nabla^2 f_\mu(x) = \begin{bmatrix} 2(1+\mu) & 2\mu \\ 2\mu & 2(1+\mu) \end{bmatrix}.$$

Solving $\det(\nabla^2 f_\mu(x) - \lambda I) = 0$, we obtain $\lambda_1 = 2$, $\lambda_2 = 2(1+2\mu)$, with eigenvectors $(1, -1)$ and $(1, 1)$, which gives $\kappa = (1+2\mu)$. This illustrates that the eigenvalues, and consequently the conditioning number, is proportional to the penalty term.

11.2 Augmented Lagrangian method of multipliers

For simplicity, consider the (primal) problem P as $(P) : \min. \{f(x) : h_i(x) = 0, i = 1, \dots, l\}$. The augmented Lagrangian method of multipliers arises from the idea of seeking for a penalty term that would allow for exact convergence for a finite penalty.

Considering the geometrical interpretation in Figure 69, one might notice that a horizontal shift in the penalty curve would allow for the extreme point of the curve to match the optimum on the z ordinate.

Therefore, we consider a modified penalised problem of the form:

$$f_\mu(x) = f(x) + \mu \sum_{i=1}^l (h_i(x) - \theta_i)^2$$

where θ_i is the shift term. One can notice that:

$$\begin{aligned} f_\mu(x) &= f(x) + \mu \sum_{i=1}^l (h_i(x) - \theta_i)^2 \\ &= f(x) + \mu \sum_{i=1}^l h_i(x)^2 - \sum_{i=1}^l 2\mu\theta_i h_i(x) + \mu \sum_{i=1}^l \theta_i^2 \\ &= f(x) + \sum_{i=1}^l v_i h_i(x) + \mu \sum_{i=1}^l h_i(x)^2, \end{aligned}$$

with $v_i = -2\mu\theta_i$. The last term is a constant and can be dropped.

The term *augmented Lagrangian* refers to the fact that $f_\mu(x)$ is equivalent to the Lagrangian function of problem P , augmented by the penalty term.

This allows for noticing important properties associated with the augmented Lagrangian $f_\mu(x)$. For example, assume that (\bar{x}, \bar{v}) is a KKT solution to P . Then the optimality condition:

$$\nabla_x f_\mu(x) = \nabla f(x) + \sum_{i=1}^l \bar{v}_i \nabla h_i(x) + 2\mu \sum_{i=1}^l h_i(x) \nabla h_i(x) = 0,$$

implies that the optimal solution \bar{x} can be recovered using a finite penalty term μ , unlike with the previous penalty-based method. The existence of finite penalty terms $\mu > 0$ that can recover optimality has an interesting geometrical interpretation, in light of what was previously discussed. Consider the same setting from Figure 69, but now we consider curves of the form $f + \bar{v}h + \mu h^2 = k$. This is illustrated in Figure 70.

Optimising the augmented Lagrangian function amounts to finding the curve $f + \bar{v}h + \mu h^2 = k$ in which $v(\epsilon) = k$. The expression for k can be conveniently rewritten as $f = -\mu [h + (\bar{v}/2\mu)]^2 + [k + (\bar{v}^2/4\mu)]$, exposing that f is a parabola shifted by $h = -\bar{v}/2\mu$.

11.2.1 Augmented Lagrangian method of multipliers

We can employ an unconstrained optimisation method to solve the augmented Lagrangian function:

$$L_\mu(x, v) = f(x) + \sum_{i=1}^l v_i h_i(x) + \mu \sum_{i=1}^l h_i(x)^2,$$

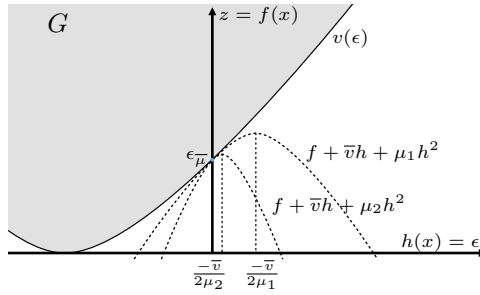


Figure 70: Geometric representation of augmented Lagrangians in the mapping $G = [h(x), f(x)]$

which amount to rely on strong duality and search for KKT points (or primal-dual pairs) (\bar{x}, \bar{v}) by iteratively operating in the primal (x) and dual (v) spaces. In particular, the strategy consists of:

1. *Primal space*: optimise $L_\mu(x, v^k)$ using an unconstrained optimisation method;
2. *Dual space*: perform a dual variable update step retaining the optimality condition $\nabla_x L(x^{k+1}, v^k) = \nabla_x L(x^{k+1}, v^{k+1}) = 0$.

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:

$$\bar{v}^{k+1} = \bar{v}^k + 2\mu h(\bar{x}^{k+1}).$$

The motivation for the dual step update stems from the following observation:

1. $h(\bar{x}^k)$ is a subgradient of $L_\mu(x, v)$ at \bar{x}^k for any v .
2. The step size is devised such that the optimality condition of the Lagrangian function is retained, i.e., $\nabla_x L(\bar{x}^k, \bar{v}^{k+1}) = 0$.

Part 2 refers to the following:

$$\begin{aligned} \nabla_x L(\bar{x}^k, \bar{v}^{k+1}) &= \nabla f(\bar{x}^k) + \sum_{i=1}^l \bar{v}_i^{k+1} \nabla h_i(\bar{x}^k) = 0 \\ &= \nabla f(\bar{x}^k) + \sum_{i=1}^l (\bar{v}_i^k + 2\mu h_i(\bar{x}^k)) \nabla h_i(\bar{x}^k) = 0 \\ &= \nabla f(\bar{x}^k) + \sum_{i=1}^l \bar{v}_i^k \nabla h_i(\bar{x}^k) + \sum_{i=1}^l 2\mu h_i(\bar{x}^k) \nabla h_i(\bar{x}^k) = \nabla_x L_\mu(\bar{x}^k, \bar{v}^k) = 0. \end{aligned}$$

That is, by employing $\bar{v}^{k+1} = \bar{v}^k + 2\mu h(\bar{x}^{k+1})$ one can retain optimality in the dual variable space for the Lagrangian function from the optimality conditions of the penalised functions, which is a condition for \bar{x} to be a KKT point.

Algorithm 18 summarises the augmented Lagrangian method of multipliers (ALMM).

Algorithm 18 Augmented Lagrangian method of multipliers (ALMM)

- 1: **initialise.** tolerance $\epsilon > 0$, initial dual solution v^0 , iteration count $k = 0$
 - 2: **while** $|h(\bar{x}^k)| > \epsilon$ **do**
 - 3: $\bar{x}^{k+1} = \mathbf{argmin} L_\mu(x, \bar{v}^k)$
 - 4: $\bar{v}^{k+1} = \bar{v}^k + 2\mu h(\bar{x}^{k+1})$
 - 5: $k = k + 1$
 - 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 $h_i(x)$. Such a procedure is still guaranteed to converge, as the requirement in Theorem 11.1 that $\mu \rightarrow \infty$ is still trivially satisfied.

One 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.

11.2.2 Alternating direction method of multipliers - ADMM

ADMM is a distributed version of the augmented Lagrangian method of multipliers, and is more suited to large problems with a decomposable structure.

Consider the problem P to be of the following form:

$$(P) : \min. \quad f(x) + g(y)$$

$$\text{subject to: } Ax + By = c.$$

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 consideration of the penalty term prevents the problem from being completely separable. To see that, let

$$\phi(x, y, v) = f(x) + g(y) + v^\top(c - Ax - By) + \mu(c - Ax - By)^2$$

be the augmented Lagrangian function. One can notice that 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 are considered: x , y , and v . The ADMM is summarised in Algorithm 19.

Algorithm 19 ADMM

- 1: **initialise.** tolerance $\epsilon > 0$, initial dual and primal solutions v^0 and y^0 , $k = 0$
 - 2: **while** $|c - A\bar{x}^k - B\bar{y}^k|$ and $\|y^{k+1} - y^k\| > \epsilon$ **do**
 - 3: $\bar{x}^{k+1} = \mathbf{argmin}\phi_\mu(x, \bar{y}^k, \bar{v}^k)$
 - 4: $\bar{y}^{k+1} = \mathbf{argmin}\phi_\mu(\bar{x}^{k+1}, y, \bar{v}^k)$
 - 5: $\bar{v}^{k+1} = \bar{v}^k + 2\mu(c - A\bar{x}^{k+1} - B\bar{y}^{k+1})$
 - 6: $k = k + 1$
 - 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 having in some of the optimisation problems (in Lines 3 and 4 in Algorithm 2) that might have solutions in closed forms.