LUISS

Department of Economics and Finance

Chair of Mathematical Finance

Optimization Techniques for models of Parimutuel Markets: Application to Finance

Prof.ssa Sara Biagini

Supervisor

Elisa Carucci, 254131

Candidate

Academic Year 2022/2023

Acknowledgements

This work was conceived and organized under the supervision of Prof. Giovanni A. Zanco, Chair of Mathematics 2 until 2022. As he moved to the University of Siena, Prof.ssa Sara Biagini, Chair of Mathematical Finance, undertook the supervisor role. Prof. Zanco continued to act as an advisor for the completion of the project.

Abstract

The aim of the thesis is to analyze the basics of optimization techniques from a theoretical viewpoint and understand how different algorithms can be used to optimize parimutuel models to be applied to real-world settings. The first part concerns an introduction to constrained optimization problems and the main theorems for equality, inequality and mixed constraints. We focus on the formulation with the Lagrangian function and the Kuhn Tucker variation for non-negativity constraints, as well as on the role of convexity in optimization. In the second part we analyze the main features of parimutuel models with particular attention to two methods of optimization: PMM and CPCAM. These two methods are employed in an options trading setting and are solved with different techniques. We focus on the CPCAM which is the convex formulation derived from the PMM and we are able to solve this constrained optimization problem with the SLSQP method from the Scipy library in Python. We believe that this algorithm is quite easy to implement, as opposed to the classical technique of Lagrange multipliers and through a few trials we can see how fundamental characteristics like number of iterations needed or objective function value vary as the dimensions of the input increase.

Numerical algorithms allow to analyze cases with higher dimensions, which would have been very difficult using the method of Lagrange multipliers. It is also interesting to note that the above-mentioned methods have been applied to financial markets but parimutuel models were actually introduced for bets to replace traditional fixed-odds betting settings. So, there is flexibility for the kinds of data to be used as inputs and the kinds of betting systems to implement.

The first part of the thesis is devoted to mathematical methods of optimization. In Chapters 2 and 3 we introduce the main theorems about necessary conditions for equality and inequality constraints; we also present some examples to describe different cases (Section 3.2). Then we deal with convexity (Chapter 4) and with convex optimization techniques (Chapter 5), studying how convexity can be beneficial to apply the algorithms presented in Chapter 6.

The second part of the thesis discusses parimutuel markets and the main models employed in betting systems and financial markets (Chapter 7). We then proceed in Chapter 8 with the mathematical formulation of two methods: PMM and CPCAM. Finally we run some simulations to test our algorithm on the CPCAM model and comment on the results (Chapter 9).

Contents

Part I

Chapter 1

Introduction

In the first part we introduce the main theorems concerning constrained optimization focusing mainly on multivariate cases, and we present a few examples. Then we shift our focus towards convexity, describing convex functions and some variations such as quasi-convex and pseudo-convex functions. After this analysis we study the basics of convex optimization which leads to an introduction on the main optimization algorithms used for constrained optimization problems. We dive deeper into the necessary conditions of the constraint qualifications and convexity to see how it is possible to have sufficient conditions for optimality in specific cases. All the information in this part come from two main references: [Simon and Blume, 1987] and [Boyd and Vandenberghe, 2004].
Pictures have been realized by Giovanni Zanco using TeXmacs.

We are dealing with constrained optimization problems with equality and inequality constraints to find the optimal values that maximize or minimize an objective function. In mathematical terms, we have a function f we want to maximize or minimize assuming its optimal values also have to satisfy constraints usually expressed by equations or inequalities. We introduce here a generic formulation of a basic constrained optimization problem:

$$\begin{cases} \max_{\mathbf{x}} f(\mathbf{x}) \\ h_1(\mathbf{x}) = h_2(\mathbf{x}) = \cdots = h_k(\mathbf{x}) = 0 \\ g_1(\mathbf{x}) \leq 0, g_2(\mathbf{x}) \leq 0, \ldots, g_m(\mathbf{x}) \leq 0 \end{cases} \tag{1.1}$$

f is the *objective function*, while the letter h refers to equality constraints and g to inequality constraints. Let C be the set of points that satisfy all equalities and inequalities; this restricts the values to be taken as solution, imposing conditions or limitations to the variables of the problem. We define the feasible region Ω as the set of points on which f is defined that satisfy every equality and inequality in the constraint set C:

$$\Omega = \{\mathbf{x} \in \text{Dom}(f) \text{ s.t. } h_i(\mathbf{x}) = 0 \forall i, g_j(\mathbf{x}) \leq 0 \forall j\} .$$

A point $\mathbf{x}$ is feasible if it satisfies the constraints. The problem is said to be feasible if there exists at least one feasible point, and infeasible otherwise. The feasible set is therefore the set of all feasible points.

Typical inequality constraints are also found in the form of non-negativity constraints, which just express the sign of a specific variable.

Sometimes it is easy to predict where we will find the maximum or minimum solution for a function. For example, if we take the function $f(x, y) = x^2 + y^2$ subject to the constraint $0 \leq x \leq 1$, we know that the maximum must lie at the corner of the domain where x and y are both at their maximum values of 1. However, we cannot always picture a graph in our mind and most of the times we will have to solve the problem following specific methods. In the next chapters we will explore basic concepts to understand how to find a solution for different kinds of constrained problems.

Chapter 2

Equality constraints

Let's consider a simple case of a two variable objective function f, differentiable everywhere, with one equality constraint:

$$\begin{cases} \max_x f(x_1, x_2) \\ h_1(x_1, x_2) = 0 \end{cases} \tag{2.1}$$

We call C the constraint, that is

$$C = \{x \in \mathbb{R} \mid h_1(x_1, x_2) = 0\}\,.$$

Imagine a xy plane where the objective function is visualized via its the level curves. To solve problem (2.1), given an equality in C, we want to find the highest level curve of f that intersects the constraint.

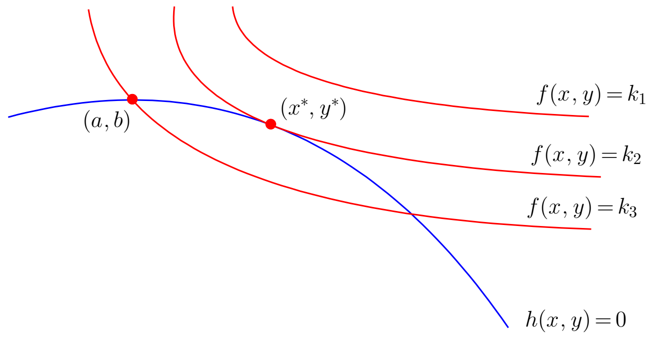


Figure 2.1: Optimality for equality constraints: if $k_1 > k_2 > k_3$, the point (a, b) is not optimal, while the point (x^*, y^*) is optimal.

We will have this condition at the optimal point x^*, where the tangent of f and h_1 will be the same.

When we discuss this topic we often refer to the slope of the curves, that is the slope of the tangent lines. The slope of a tangent line of the function refers to the rate of change of the function with respect to one variable; the gradient of a function is a vector that generalizes the concept of slope to multiple variables. The tangent of the objective function and the constraint are equal at the optimal solution and so they have equal slope. At the optimal point the gradient of f will be perpendicular to its tangent line and the gradient of h_1 will be perpendicular to its tangent line. Indeed, when we are looking for the optimal solution, if the tangent of the constraint is steeper than the one of f we can move and reach a higher point (Figure 2.1) and if it is less, again we need to move until we reach the point of the tangent lines having the same slope. We can think of a general multivariate function f where its gradient gives a vector perpendicular to the level curve and so perpendicular to the tangent plane. So, the gradient of the function in set C and f line up and point in the same or opposite direction (Figure 2.2). If all the partial derivatives are different from zero, we can say that the tangent lines of f and h_1 have the same slope and this is shown in (2.2). This means that we can multiply one by some scalar to get the other and in our case this multiplier is called μ and it is the result of the quotient in (2.3). The coefficient μ describes the the relationship of the two gradients in terms of proportionality between each other. We can think of the gradient as the direction the function is heading to. The gradient points at the direction of the maximum increase of the function.

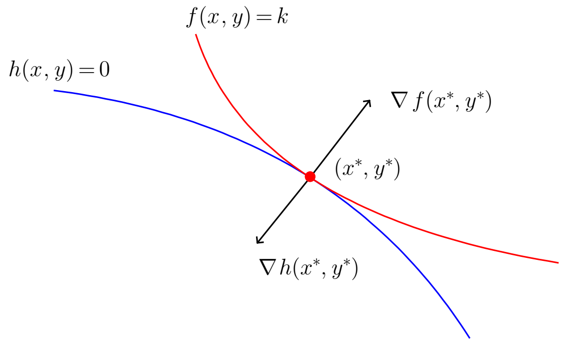


Figure 2.2: Optimality for equality constraints: parallel gradients.

If we express the image in mathematical terms, we write the gradient of the objective function f and h_1 equating each other. In equation (2.2), the slopes of the tangents of f and h are equal at $\mathbf{x}^*$. Then, we find that the terms can be exchanged and thus the division of the derivative of f with respect to x_1 over the derivative of h over x_1 is the same as dividing with respect to x_2 and we name the result of the division as μ. Here, the derivatives of h with respect to x_1 and x_2 need to be different from zero, in order for equation (2.3) to hold and we can define this condition as *constraint qualification* since it is a restriction we put on the constraint.

$$\frac{\frac{\partial f}{\partial x_1}(\mathbf{x}^*)}{\frac{\partial f}{\partial x_2}(\mathbf{x}^*)} = \frac{\frac{\partial h}{\partial x_1}(\mathbf{x}^*)}{\frac{\partial h}{\partial x_2}(\mathbf{x}^*)} \qquad (2.2)$$

$$\frac{\frac{\partial f}{\partial x_1}(\mathbf{x}^*)}{\frac{\partial h}{\partial x_1}(\mathbf{x}^*)} = \frac{\frac{\partial f}{\partial x_2}(\mathbf{x}^*)}{\frac{\partial h}{\partial x_2}(\mathbf{x}^*)} = \mu \qquad (2.3)$$

If we don't assume that all the derivatives of h are different from 0, we can rewrite (2.3) as two equations

$$\frac{\partial f}{\partial x_1}(\mathbf{x}^*) - \mu\frac{\partial h}{\partial x_1}(\mathbf{x}^*) = 0,$$
$$\frac{\partial f}{\partial x_2}(\mathbf{x}^*) - \mu\frac{\partial h}{\partial x_2}(\mathbf{x}^*) = 0.$$

We can then solve a system of three unknowns and three equations by including the constraint. The system is expressed as

$$\begin{cases} \frac{\partial f}{\partial x_1}(\mathbf{x}) - \mu \frac{\partial h}{\partial x_1}(\mathbf{x}) = 0 \\ \frac{\partial f}{\partial x_2}(\mathbf{x}) - \mu \frac{\partial h}{\partial x_2}(\mathbf{x}) = 0 \\ h\left(x_1, x_2\right) - c = 0. \end{cases} \tag{2.4}$$

Under some assumptions, any solution of problem (2.1) is a solution of system (2.4).

Assumption 1. *The constraint function h_1 is differentiable at $\mathbf{x}$ and the gradient of the constraint function at $\mathbf{x}$ is different from $\mathbf{0}$, that is, at least one partial derivative of h_1 is not zero at $\mathbf{x}$.*

We refer to this assumption as *non-degeneracy constraint qualifications* (or NDCQ) at $\mathbf{x}$.

Let us write equation (2.4) in the form of an unconstrained problem. We use the Lagrangian function.

$$L\left(x_1, x_2, \mu\right) \equiv f\left(x_1, x_2\right) - \mu\left(h\left(x_1, x_2\right) - c\right) \tag{2.5}$$

Theorem 2.0.1. *Suppose the NDCQ conditions are satisfied and $\mathbf{x}^*$ is a solution for problem (2.1). Let f be a differentiable function at $\mathbf{x}^*$. Then, there exists μ^* such that (x_1^*, x_2^*, μ^*) is a critical point of the Lagrangian function, that is,*

$$\frac{\partial L}{\partial x_1} = 0, \quad \frac{\partial L}{\partial x_2} = 0, \quad \textit{and} \quad \frac{\partial L}{\partial \mu} = 0.$$

The Lagrangian creates an unconstrained problem of three variables from a constrained problem of two variables. It has the same result of problem (2.4), in fact when we differentiate this function with respect to μ, we obtain the constraint. When the solution of our maximization problem is not one of the critical points of h we conclude that there is a number μ^* such that our solution can be found from the Lagrangian (x_1^*, x_2^*, μ^*).

2.1 Critical points of the Lagrangian

To solve optimization problems, we take the critical points of the constraints and include them in the candidates for the optimal solution. We then check that the $NDCQ$ conditions and we write the Lagrangian to find other candidate points in addition to the ones found with the gradient of h_1. Let's take an example where the gradient of h_1 is $(0, 0)$. Now we formulate the Lagrangian with an alternative method, useful to verify the constraint qualification conditions in a compact way. We insert a multiplier both in front of the objective function and of the constraint.

$$\begin{cases} \max_{x_1,x_2} -(x-1)^2 - y^2 \\ h_1(x_1,x_2) = \left(x^2+y^2-1\right)^2 = 0 \end{cases} \tag{2.6}$$

We compute the critical points of he constraint and we find that it is $(0,0)$ in all the point of the circumference $x^2+y^2=1$ and it lies in the constraint. Therefore, the NDCQ are not satisfied. The generic Lagrangian for this kinds of problems would be in this form, where we also include a multiplier μ_0 for f.

$$L\left(x_1,x_2,\mu_0,\mu_1\right) \equiv \mu_0 f\left(x_1,x_2\right) - \mu_1(h_1(x_1,x_2))$$

Theorem 2.1.1. *Suppose the objective function f and the equality constraint h are C^1 and $\mathbf{x}^*$ is the solution of the problem.*

Then, there exist multipliers such that

1. *μ_0 and μ_1 are not both zero*
2. *μ_0 is either 0 or 1*
3. *$x_1^*, x_2^*, \mu_0^*, \mu_1^*$ solves the system*

$$\begin{aligned} \frac{\partial L}{\partial x_1} &= \mu_1 \frac{\partial f}{\partial x_1}\left(x_1,x_2\right) - \mu_1 \frac{\partial h}{\partial x_1}\left(x_1,x_2\right) = 0 \\ \frac{\partial L}{\partial x_2} &= \mu_0 \frac{\partial f}{\partial x_2}\left(x_1,x_2\right) - \mu_1 \frac{\partial h}{\partial x_2}\left(x_1,x_2\right) = 0 \\ \frac{\partial L}{\partial \mu_1} &= c - h\left(x_1,x_2\right) = 0. \end{aligned}$$

Suppose we know the solution (x_1^*, x_2^*) and it satisfies the assumption of Theorem 2.1.1. If it is not a critical point of h_1 we can set $\mu_0^* = 1$ and deduce that (x_1^* , x_2^*, μ_1^*) satisfy system. If it is a critical point of h and all the partial derivatives are equal to zero, we can set μ_0^* equal to zero and deduce that (x_1^* , x_2^*, μ_1^*, μ_0^*) satisfy the constraint.

We apply what we have said to problem (2.6), Assumption 1 is not satisfied. We set $\mu_0 = 0$ and so we cannot find the solution of the problem using the derivatives of the

Lagrangian function.
If Assumption 1 is not satisfied at some point $\mathbf{x}$ that does not lie in h_1, we can still build the Lagrangian to go on and solve the problem. For example, consider the problem of optimizing a function subject to the constraint $h(x, y) = 0$. If you find the critical points by taking the partial derivatives of f with respect to x and y, and solving the resulting system of equations, you may obtain points (x^*, y^*) that satisfy the equations, but not the constraint. In such cases, those points may not be valid solutions to the original optimization problem. In general, if the constraint set contains a critical point, we include it in the candidates for the solution. This concept can be also explained geometrically by picturing the gradients of f and h. The gradients are perpendicular to the tangent and point in opposite or same direction. In either case, they are multiples of each other, where the scalar multiple is μ^*.

In the following example we cannot apply any of the theorems described above because the objective function is not differentiable everywhere.

Let us take a problem as example

$$\begin{cases} \max_{x_1,x_2} x_1 \\ h_1(x_1, x_2) = x_1^3 + x_2^2 \end{cases} \tag{2.7}$$

The function is not differentiable but we can see from Figure 2.3 that $(0, 0)$ is the solution. If we try to solve the problem following the setup of system (2.4), we will not find a solution even though we know by visualizing the graph that the maximum exists and it is in the origin.

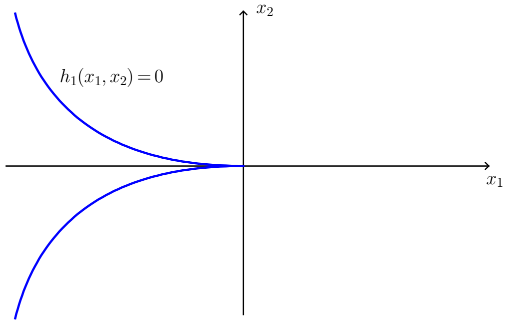


Figure 2.3: Non differentiable constraint.

2.2 Multiple constraints and variables

Let's consider this multivariate problem

$$\begin{cases} \max_{\mathbf{x}} f(\mathbf{x}) \\ h_1(\mathbf{x}) = 0 \\ h_2(\mathbf{x}) = 0 \\ \vdots \\ h_m(\mathbf{x}) = 0 \end{cases} \tag{2.8}$$

When we have m constraints, we can extend Assumption 1. We first define the Jacobian of $\mathbf{H} = (h_1, \ldots, h_m)$ at $\mathbf{x}$ as

$$D\mathbf{H}(\mathbf{x}) = \operatorname{rank} \begin{pmatrix} \frac{\partial h_1}{\partial x_1}(x) & \cdots & \frac{\partial h_1}{\partial x_n}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial x_1}(x) & \cdots & \frac{\partial h_m}{\partial x_n}(x) \end{pmatrix} \tag{2.9}$$

The rank of a $m \times n$ matrix A is the number of linearly independent columns of A or the number of nonzero rows in its row echelon form.

Assumption 2. *The function $\mathbf{H}$ is differentiable at $\mathbf{x}$ and the rank of $D\mathbf{H}(\mathbf{x})$ is m*

The rank needs to be m, to ensure that there is a tangent plane of the function at that point, with $n - m$ dimensions. Now, we look for candidate points in the constraint and we assume $\mathbf{x}^*$ maximizes our function and if the Jacobian matrix in $\mathbf{x}^*$ has maximal rank, then our constraint qualifications are satisfied. We write the Lagrangian

$$L(\mathbf{x}, \mu) \equiv f(\mathbf{x}) - \mu_1 h_1(\mathbf{x}) - \mu_2 h_2(\mathbf{x}) - \cdots - \mu_m h_m(\mathbf{x})$$

Theorem 2.2.1. *If at $\mathbf{x}^*$, Assumption 2 is satisfied and the objective function is C^1, then there are multipliers $\boldsymbol{\mu}^* = (\mu_1^*, \ldots, \mu_m^*)$ such that $\mathbf{x}^*$, μ^* is a solution of the system*

$$\frac{\partial L}{\partial x_1}(\mathbf{x}^*, \mu^*) = 0, \ldots, \frac{\partial L}{\partial x_n}(\mathbf{x}^*, \mu^*) = 0$$
$$\frac{\partial L}{\partial \mu_1}(\mathbf{x}^*, \mu^*) = 0, \ldots, \frac{\partial L}{\partial \mu_m}(\mathbf{x}^*, \mu^*) = 0$$

If the rank of the Jacobian of $\mathbf{H}$ at $\mathbf{x}^*$ is smaller than the number of constraint equations, the rows of the Jacobian are linearly dependent. In this case Assumption 2 is not satisfied for some rows, specifically for the functions in the constraint set that do not have at least one partial derivative different from 0.

Chapter 3

Inequality constraints

In optimization problems we often deal with inequality constraints where we take into consideration the region on one side of the constraint set, depending on the sign of the inequality. Let's take a two variable problem with one inequality constraint as reference:

$$\begin{cases} \max_x f(x_1, x_2) \\ g_1(x_1, x_2) \leq 0 \end{cases} \tag{3.1}$$

To find candidates for the maximization problem we still follow the same steps as with equality constraints but we distinguish the types of inequality constraints. We check the critical points of the constraint for our problem and we assume we know the solution of our problem x^*. If x^* satisfies the constraint as an equality, it is said to be binding. In this case we treat it as an equality constraint. If the highest level curve of f meets the constraint at a point $x^* = (x_1^*, x_2^*)$ laying on the boundary of the constraint, it means that $g(x_1^*, x_2^*) = 0$ and so the constraint is binding. Suppose that the maximum of f occurs not where $g(x_1, x_2) = 0$ but where $g(x_1, x_2) < 0$, in this case the constraint is said to be not binding. The highest level curve of f meets the constraint set at the highest point possible and so where it is binding. As for this, we only care for binding constraints to find the optimal solution. At the optimum, the level sets of f and g are tangent and so their gradients line up.

$$\nabla f(x_1, x_2) - \lambda \nabla g(x_1, x_2) = 0$$

Suppose the optimum is found where $g(x_1^*, x_2^*) = 0$, the gradients of f and g point in either the same direction since the gradient of f points where the function increase and the one of g points in the direction where $g(x_1, x_2) \geq 0$. If the gradient of f pointed in the interior region we could still keep $g(x_1, x_2) \leq 0$ but the optimal solution would not be in a point where the constraint is binding. So in this case, the gradients must point in the same direction at the point x^*, opposite to the feasible region of the constraint set (see Figure 3.1). The direction depends on the sign of the constraint. So, the multiplier $\lambda \geq 0$. If the maximum is in the interios region where $g(x_1^*, x_2^*) < 0$, in this point$g(x_1, x_2) = 0$ the gradients of f and g point in opposite directions. Here, the constraint is not binding and the maximum is unconstrained.

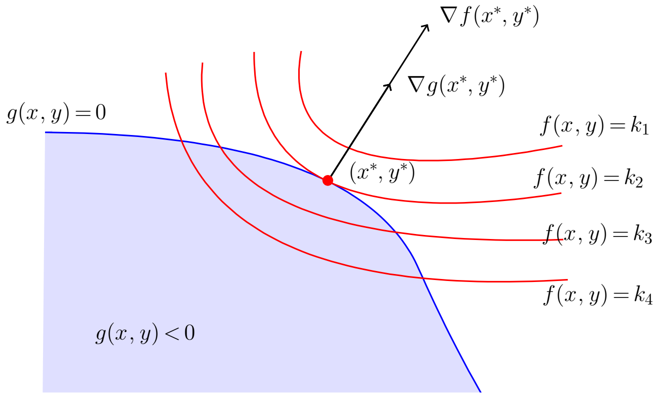


Figure 3.1: Optimality for inequality constraints: binding constraint.

we set up the Lagrangian function

$$L(x, y, \lambda) = f(x, y) - \lambda g(x, y)$$

and we proceed as we did in the previous section. We find the gradient of g and check if their critical points are candidates for the optimal solution of the problem following assumption 1 and set up the Lagrangian function with Lagrangian multipliers denoted with λ for inequality constraints. In the case of not binding constraint in (3.1), we can still set up the Lagrangian. Since the constraint is not binding at the maximum we don't take it into account and put the λ associated with the constraint equal to zero. In this case our problem (3.1) would turn out to be unconstrained since there is only one constraint that is not useful to find the solution. When solving the Lagrangian we don't consider those candidate solutions that have negative λ associated, where λ is the multiplier for inequality constraints. To explain this concept, we assume there are not internal solutions in the feasible region where the problem would be simply unconstrained and g would not be useful to solve. If lambda is negative, it means that the scalar multiplication leads the gradients of f and g to point in opposite directions (see Figure 3.2). So, the constraint is not binding as it will point towards the feasible region while the gradient of f will point outside of it.

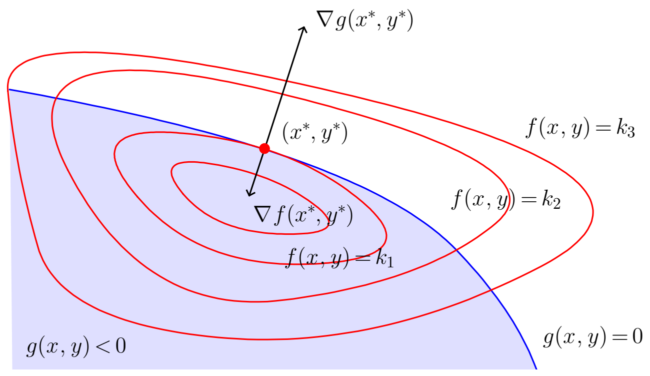


Figure 3.2: Optimality for inequality constraints: non binding constraint.

We can explain this concept graphically, why the constraint is not binding. The gradient of g is pointing in the shaded region representing the part where the constraint in (3.1) is less than 0. Since the gradient tells us the direction we have to follow to see where f increases the most, we want to reach the highest point possible while still satisfying the constraint and we will have this condition only on the border of the constraint since we cannot go any

3.1 Necessary conditions for inequality constraints

Suppose we want to solve the problem

$$\begin{cases} \max_{\mathbf{x}} f(\mathbf{x}) \\ g_1(\mathbf{x}) \leq 0 \\ g_2(\mathbf{x}) \leq 0 \\ \vdots \\ g_k(\mathbf{x}) \leq 0 \end{cases} \tag{3.2}$$

Suppose that $\mathbf{x}^*$ is a maximizer satisfying suitable constraint qualification conditions, that we discuss below.
We form the Lagrangian

$$L\left(x_1, \ldots, x_n, \lambda_1, \ldots, \lambda_k\right) \equiv f(\mathbf{x}) - \lambda_1\left[g_1(\mathbf{x}) - 0\right] - \cdots - \lambda_k\left[g_k(\mathbf{x}) - 0\right]$$

However, the formulation of the problem is more extended as there are more conditions to include. First of all if a constraint inequality $g_j(\mathbf{x}) \leq 0$ is binding at $\mathbf{x}^*$ we add the condition of λ_j being nonnegative, as we discussed before. In this case we have $g_j(\mathbf{x}^*) = 0$. Then we define the concept of *complementary slackness*. When the inequality constraint is not binding λ is equal to zero. Since we want to solve the Lagrangian system to find candidate maximizers, we don't know if the constraint will be binding at the maximizer or not before solving the system; so we cannot use the condition of the derivative of the Lagrangian function with respect to the multiplier of each constraint equal to zero as this means saying that each g_j is equal to zero, so binding. Thus, we insert two more conditions with respect to the formulation in (2.4): the positive sign for all λ_j and the condition that either g_j is binding or $\lambda_j = 0$.

Since for inequality constraints we only consider binding constraints in the Lagrangian, we proceed in the following way. We assume that $\mathbf{x}^*$ is the solution for our problem satisfying the constraints

$$g_j(\mathbf{x}) \leq 0, \quad j = 1, \ldots, k$$

and only $g_1, ..., g_e$ are binding. We set up the Jacobian matrix of the biding constraints

$$\begin{pmatrix} \frac{\partial g_1}{\partial x_1}(\mathbf{x}^*) & \cdots & \frac{\partial g_1}{\partial x_n}(\mathbf{x}^*) \\ \vdots & \ddots & \vdots \\ \frac{\partial g_e}{\partial x_1}(\mathbf{x}^*) & \cdots & \frac{\partial g_e}{\partial x_n}(\mathbf{x}^*) \end{pmatrix}$$

Assumption 3. *The constraint functions $g_1, ...g_e$ are differentiable at $\mathbf{x}$ and the rank is equal to e and so it is as large as possible. The constraint qualification conditions in this case are satisfied.*

Theorem 3.1.1. *Suppose Assumption 3 is satisfied and $\mathbf{x}^*$ is a solution for problem (3.2). Let f be a differentiable function at $\mathbf{x}^*$. Then, there exists $\lambda_1^*, \ldots, \lambda_k^*$ such that $(\mathbf{x}^*, \lambda_1^*, \ldots, \lambda_k^*)$ is a critical point of the Lagrangian function and*

(a) $\frac{\partial L}{\partial x_1}(\mathbf{x}^*, \lambda^*) = 0, \ldots, \frac{\partial L}{\partial x_n}(\mathbf{x}^*, \lambda^*) = 0,$

(b) $\lambda_1^* g_1(\mathbf{x}^*) = 0, \ldots, \lambda_k^* g_k(\mathbf{x}^*) = 0$

(c) $\lambda_1^* \geq 0, \ldots, \lambda_k^* \geq 0$

(d) $g_1(\mathbf{x}^*) \leq 0, \ldots, g_k(\mathbf{x}^*) \leq 0$

Suppose we know the solution of our problem exists and is equal to x^*. Also in the case of inequality constraints we can include the constraint qualification conditions in the formulation by inserting a multiplier λ_0 in front of the objective function.

$$L(\mathbf{x}, \lambda_0, \lambda_1, \ldots, \lambda_k) \equiv \lambda_0 f(\mathbf{x}) - \lambda_1 g_1(\mathbf{x}) - \cdots - \lambda_k g_k(\mathbf{x})$$

If $\mathbf{x}^*$ is the solution, $\mathbf{x}^*, \lambda_0^*, \lambda_1^* ... \lambda_k^*$ satisfy the system below.

(a) $\dfrac{\partial L}{\partial x_1}(\mathbf{x}^*, \lambda^*) = 0, \ldots, \dfrac{\partial L}{\partial x_n}(\mathbf{x}^*, \lambda^*) = 0$

(b) $\lambda_1^* g_1(\mathbf{x}^*) = 0, \ldots, \lambda_k^* g_k(\mathbf{x}^*) = 0$

(c) $\lambda_1^* \geq 0, \ldots \lambda_k^* \geq 0$

(d) $g_1(\mathbf{x}^*) \leq b_1, \ldots, g_k(\mathbf{x}^*) \leq b_k$

(e) $\lambda_0^* = 0$ or $1,$ and

(f) $(\lambda_0^*, \lambda_1^* \dots, \lambda_k^*) \neq (0, 0, \dots 0)$

We would like to have $\lambda_0 = 1$ otherwise the objective function will drop out of the first order conditions of the Lagrangian function and so we want some other qualification conditions that guarantee that $\lambda_0 \dots \lambda_k = 1$.

3.2 Examples

Let us solve a simple maximization problem

$$\begin{cases} \max\limits_{c_1, c_2} U(c_1, c_2) \\ p_1 c_1 + p_2 c_2 \leq k \\ c_1, c_2 \geq 0 \end{cases} \tag{3.3}$$

where U is a given function and $k > 0,\ p_1, p_2 \geq 0$ are fixed. This is the typical form of a utility maximization problem with two commodities.

We have three constraints and two variables but we don't know which ones are binding in $\mathbf{c}^* = (c_1^*, c_2^*)$, supposing that a solution to the problem exists. To check the constraint qualification conditions we compute the Jacobian of the constraint functions and so we will have a matrix with all the derivatives with respect to c_1 in the first column and with respect to c_2 in the second one. We switch all the signs of the variables in the inequality constraint, so that they read $-c_1 \leq 0$, $-c_2 \leq 0$. To satisfy Assumption 3 the rank needs to be as high as possible so in the case two constraints are binding we need to have rank 2; if only one is binding we need to have rank 1 and we check Assumption 1. In this example we have

$$\nabla g(c_1, c_2) = \begin{bmatrix} p_1 & p_2 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}$$

so that the rank is always 2.

Now we check all the possible combinations of binding constraints. If there are two binding constraints, we would need to have the rank equal to 2 and this would always be the case for this problem. If we check individually if these conditions are satisfied we may have some restrictions to make sure the constraint qualifications hold. For example if only the first row is binding then the rank would need to be 1 and so either p_1 or p_2 or both need to be nonzero but this will always be true since the constraint would not be binding otherwise. We conclude that NDCQ are satisfied at all points (c_1, c_2). If we assume that f is differentiable everywhere all the solutions of the maximization problem satisfy the system.

Let us analyze a simple one variable problem to understand what kinds of problems we might encounter when these conditions are not satisfied.

$$\begin{cases} \max x \\ x^3 + 2x^2 \leq 0 \end{cases} \tag{3.4}$$

We take the derivative of the constraint and see that it will be equal to zero when $x = 0$ and $x = \frac{-4}{3}$. We need the condition of the gradient to be different from zero. If we solve the Lagrangian we won't find a solution to our system because the real solution of this problem is actually $x = 0$ but earlier we saw that the constraint qualification conditions are not satisfied in this point. So, our optimal point does not satisfy the first order conditions of the Lagrangian function. When we find some points where the conditions are not satisfied and we solve the Lagrangian system we have the risk of having the actual solution of the problem in those points and so even though the solution exists we won't be always able to find it in this way.

3.3 Mixed constraints

When optimization problems have both inequality and equality constraints, we say they have mixed constraints. In the case of mixed constraints we follow the steps taken with problem (3.3) and so we check all the combinations of constraints being binding and in what points this condition is not satisfied. When we are done with this process and start solving the Lagrangian function we don't know what constraints are going to be binding at the solution $\mathbf{x}^*$ but we analyze all the cases as before.

We take a generic problem with the following constraint set where we have k inequalities and m equalities and the functions $f, g_1, ...g_k, h_1, ...h_m$ are C^1

$$\begin{aligned} g_1(x_1, \dots, x_n) &\leq b_1, \dots, g_k(x_1, \dots, x_n) \leq 0 \\ h_1(x_1, \dots, x_n) &= c_1, \dots, h_m(x_1, \dots, x_n) = 0 \end{aligned}$$

we write the Lagrangian function

$$\begin{aligned} L(x_1, \dots, x_n, \lambda_1, \dots, \lambda_k, \mu_1, \dots, \mu_m) &\equiv f(\mathbf{x}) - \lambda_1 g_1(\mathbf{x}) - \dots - \lambda_k g_k(\mathbf{x}) \\ &- \mu_1 h_1(\mathbf{x}) - \dots - \mu_m h_m(\mathbf{x}) \end{aligned}$$

We assume that the first k_0 inequality constraints are binding at $\mathbf{x}$ and we suppose that the NDCQ are satisfied at $\mathbf{x}$. So, the rank at $\mathbf{x}$ of the Jacobian matrix of the equality and binding constraints is as large as it can be.

$$\begin{pmatrix} \frac{\partial g_1}{\partial x_1}(\mathbf{x}^*) & \cdots & \frac{\partial g_1}{\partial x_n}(\mathbf{x}^*) \\ \vdots & \ddots & \vdots \\ \frac{\partial g_{k_0}}{\partial x_1}(\mathbf{x}^*) & \cdots & \frac{\partial g_{k_0}}{\partial x_n}(\mathbf{x}^*) \\ \frac{\partial h_1}{\partial x_1}(\mathbf{x}^*) & \cdots & \frac{\partial h_1}{\partial x_n}(\mathbf{x}^*) \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial x_1}(\mathbf{x}^*) & \cdots & \frac{\partial h_m}{\partial x_n}(\mathbf{x}^*) \end{pmatrix}$$

Theorem 3.3.1. *If $f, g_1, ...g_k, h_1, ...h_m$ are C^1 and the NDCQ are satisfied, there exist multipliers $\lambda_1, ...\lambda_k, \mu_1, ...\mu_k$ such that*

(a) $\frac{\partial l}{\partial x_1}(\mathbf{x}^*, \lambda^*) = 0, \dots, \frac{\partial L}{\partial x_n}(\mathbf{x}^*, \lambda^*) = 0,$

(b) $\lambda_1^*[g_1(\mathbf{x}^*)] = 0, \dots, \lambda_k^* \lfloor g_k(\mathbf{x}^*) \rfloor = 0,$

(c) $h_1(\mathbf{x}^*) = 0, \dots, h_m(\mathbf{x}^*) = 0$

(d) $\lambda_1 \geq 0, \dots, \lambda_k \geq 0$

(e) $g_1(\mathbf{x}^*) \leq 0, \dots, g_k(\mathbf{x}^*) \leq 0$

3.4 Kuhn-Tucker formulation

Some problems have inequality constraints and non-negativity constraints in the form

$$\begin{cases} \max_{\mathbf{x}} f(\mathbf{x}) \\ g_1(\mathbf{x}) \leq 0, \dots, g_k(\mathbf{x}) \leq 0 \\ h_1(\mathbf{x}) = 0, \dots, h_m(\mathbf{x}) = 0 \\ x_1, \dots, x_d \geq 0 \end{cases} \tag{3.5}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $d \leq n$.

Assuming that the constraint qualification conditions are satisfied, we then formulate the problem with the Kuhn Tucker method which is similar to the way we approached these problems before. Solving the problem following this formulation has some advantages such as the symmetry of the required conditions with respect to x_i and λ_i.
We write the Lagrangian for this problem

$$\widetilde{L}(\mathbf{x}, \lambda_1, \dots, \lambda_k, \mu_1, \dots, \mu_m) = f(\mathbf{x}) - \lambda_1 g_1(\mathbf{x}) - \cdots - \lambda_k g_k(\mathbf{x}) - \mu_1 h_1(\mathbf{x}) - \cdots - \mu_m h_m(\mathbf{x})$$

Theorem 3.4.1. *Suppose that $f, g_1, ...g_k, h_1, ...h_k$ are C^1 and the Jacobian matrix formed by equality and binding constraints has maximal rank at $\mathbf{x}^*$. Then, there exist non-negative multipliers $\lambda_1^*, ...\lambda_k^*, \mu_1^*, ...\mu_k^*$ such that the following system is satisfied*

(a) $\dfrac{\partial \widetilde{L}}{\partial x_1} \leq 0, \dots, \dfrac{\partial \widetilde{L}}{\partial x_d} \leq 0,$

(b) $\dfrac{\partial \widetilde{L}}{\partial x_{d+1}} = 0, \dots, \dfrac{\partial \widetilde{L}}{\partial x_n} = 0,$

(c) $\dfrac{\partial \widetilde{L}}{\partial \lambda_1} \geq 0, \dots, \dfrac{\partial \widetilde{L}}{\partial \lambda_k} \geq 0,$

(d) $\dfrac{\partial \widetilde{L}}{\partial \mu_1} = 0, \dots, \dfrac{\partial \widetilde{L}}{\partial \mu_m} = 0,$

(e) $x_1 \dfrac{\partial \widetilde{L}}{\partial x_1} = 0, \dots, x_d \dfrac{\partial \widetilde{L}}{\partial x_d} = 0,$

(f) $\lambda_1 \dfrac{\partial \widetilde{L}}{\partial \lambda_1} = 0, \dots, \lambda_k \dfrac{\partial \widetilde{L}}{\partial \lambda_k} = 0,$

(g) $\lambda_1 \geq 0, \dots, \lambda_k \geq 0.$

Conditions (a) and (e) are equivalent to the non-negativity constraints because either the non-negativity constraint is binding and so the variable is equal to zero or we can find the derivative of the Lagrangian with respect to that variable; condition (f) is complementary slackness. With reference to condition (a) and (e), for any variable $x_1, ...x_n$ we write the first order conditions derived from the generic Lagrangian function

$$\frac{\partial L}{\partial x_1} = \frac{\partial f}{\partial x_1} - \lambda_1 \frac{\partial g_1}{\partial x_1} - \cdots - \lambda_k \frac{\partial g_k}{\partial x_1} + \nu_1 = 0$$
$$\frac{\partial L}{\partial x_n} = \frac{\partial f}{\partial x_n} - \lambda_n \frac{\partial g_1}{\partial x_n} - \cdots - \lambda_k \frac{\partial g_k}{\partial x_n} + \nu_n = 0,$$

where v is a multiplier for non-negativity constraints. We rewrite them as

$$\frac{\partial L}{\partial x_j} = \frac{\partial \tilde{L}}{\partial x_j} + v_j = 0$$

or

$$\frac{\partial \tilde{L}}{\partial x_j} = -v_j$$

where j represents a generic index for the variables. From what we wrote just above and

$$v_1 x_1 = 0$$
$$\vdots$$
$$v_n x_n = 0$$

we derive that

$$\frac{\partial \tilde{L}}{\partial x_j} \leq 0 \quad \text{and} \quad x_j \frac{\partial \tilde{L}}{\partial x_j} = 0$$

If we think about a two variable maximization problem with two linear equality constraints $h_1 = 0$, $h_2 = 0$, intersecting at (x_1^*, x_2^*) we know the gradients of h_1 and h_2 have to be linearly independent to satisfy the NDCQ conditions. In this case, the feasible region, will be just one point, so it must be the maximization solution. On the other side, problems with more variables or inequality constraints or nonlinear constraints may be more complicated and the Lagrange formulation is useful to solve them.

These conditions include the stationarity condition, which states that the gradient of the objective function must be to the tangent plane of the equality and binding constraints. The stationarity condition can be expressed as a linear combination of the gradients of the constraints. Additionally, the feasibility conditions require that the constraints are satisfied, and the regularity condition, which ensures that the gradients of the binding

constraints are linearly independent at the optimal solution.
When we locate the intersection point that will maximize our objective function, we can set the λ corresponding to the constraint not in the solution, equal to zero as those constraints will not be binding. This means that for each constraint we have one equality and one inequality, representing the constraint and the λ corresponding to it. For example for the maximum solution, the constraints intersecting will be binding (equalities) and the corresponding λ will be an inequality (non-negative value) and the opposite for the constraints not included in the intersection point representing the solution. This refers to the concept of complementarity slackness. To summarize these conditions we include constraints in the formulation of the optimization problem referring to the concept of complementarity slackness, namely $\lambda(g(x) - b) = 0$.

Let's see how we can use the conditions of this formulation to solve some optimization problems. When we approach an optimization problems simply setting up and solving the Lagrangian system, depending on the number and type of the constraint we can have multiple solutions. These solutions are candidates for being the maximizers; then we choose which ones solve the maximization problem, if any. Then we will solve a problem with non-negativity constraints.

3.4.1 Examples

For example let us take this problem

$$\begin{cases} \max_{x_1,x_2} x_1^2 + x_2^2 - 14x_1 - 6x_2 \\ g(x_1, x_2) = x_1 + x_2 - 2 \leq 0 \\ g(x_1, x_2) = x_1 + 2x_2 - 3 \leq 0 \end{cases} \tag{3.6}$$

We proceed in the same way as the utility maximization example and check the constraint qualification conditions by checking the rank. In fact we don't know at the beginning what constraints are going to be active at the optimal solution. After we have explored all the cases we check for optimality. We set up the Lagrangian and differentiate with respect to x_1, x_2 and with respect to the λ. For example for one of the candidates the value of λ_2 is negative and so we cannot consider it as a solution.

The Lagrangian in this case is

$$L(x_1, x_2, \lambda_1, \lambda_2) = x_1^2 + x_2^2 - 14x_1 - 6x_2 - \lambda_1(x_1 + x_2 - 2) - \lambda_2(x_1 + 2x_2 - 3)$$

The Jacobian matrix of the constraints is

which is constant and always has rank 2, so the NDCQ are satisfied at every point. All the functions are differentiable, therefore all maximizers must be solutions of the system

$$\begin{cases} \dfrac{\partial L}{\partial x_1} = 2x_1 - 14 - \lambda_1 - \lambda_2 = 0 \\ \dfrac{\partial L}{\partial x_1} = 2x_2 - 6 - \lambda_1 - \lambda_2 = 0 \\ \lambda_1(x_1 + x_2 - 2) = 0 \\ \lambda_2(x_1 + 2x_2 - 3) = 0 \\ \lambda_1 \geq 0 \\ \lambda_2 \geq 0 \\ x_1 + x_2 - 2 \leq 0 \\ x_1 + 2x_2 - 3 \leq 0 \end{cases}$$

One can then easily check that this system has no solutions. For example if both constraints are binding the only feasible point is $(x_1, x_2) = (1, 1)$, to which correspond a negative value of λ_1, so that it is not an acceptable point. Indeed the objective function is unbounded above in the feasible region for this problem.

Let us now analyze a problem solved with the Kuhn-Tucker method. We present a problem with both inequality and non-negativity constraints.

$$\begin{cases} \max_{x_1,x_2} x_1^2 + x_1 + 4x_2^2 \\ g(x_1, x_2) = 2x_1 + 2x_2 \leq 1 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{cases} \tag{3.7}$$

In this case the feasible region is closed and bounded, therefore the problem has a solution

(see Theorem 5.1.1). We check the $NCDQ$ conditions with the Jacobian matrix and see that they are satisfied in every point. We write the Lagrangian and the complete formulation following the Kuhn-Tucker method; At the end we will have candidate solutions which can be substituted in the original function to maximize and select the candidate that gives a higher value.
We have

$$\widetilde{L}\left(x_1, x_2, \lambda_1\right) = x_1^2 + x_1 + 4x_2^2 - \lambda_1\left(2x_1 + 2x_2 - 1\right)$$

$$\begin{cases} \dfrac{\partial \widetilde{L}}{\partial x_1} = 2x_1 + 1 - 2\lambda_1 \leq 0 \\ \dfrac{\partial \widetilde{L}}{\partial x_2} = 8x_2 - 2\lambda_1 \leq 0 \\ \dfrac{\partial \widetilde{L}}{\partial \lambda_1} = -2x_1 - 2x_2 + 1 \geq 0 \\ x_1(2x_1 + 1 - 2\lambda_1) = 0 \\ x_2(8x_2 - 2\lambda_1) = 0 \\ \lambda_1(-2x_1 - 2x_2 + 1) = 0 \end{cases}$$

The solutions of the system are the three points

$$\left(0, \frac{1}{2}\right), \quad \left(\frac{1}{2}, 0\right), \quad \left(\frac{3}{10}, \frac{1}{5}\right) \;;$$

the maximum of f among them is 1.

3.5 Tangent cones and constraint qualification

Definition 3.5.1. *Let's take a set $A \subset \mathbb{R}^n$ and a point $\mathbf{x} \in A$; a vector $\mathbf{y}$ is* tangent *to A at $\mathbf{x}$ if $\mathbf{y} = 0$ or if $\exists \{\mathbf{x}_n\}_{n\in\mathbb{N}} \subset A$ such that $\mathbf{x}_n \neq \mathbf{x}$, $\mathbf{x}_n \to \mathbf{x}$ and*

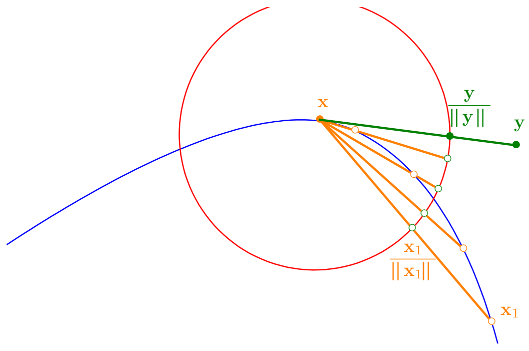
$$\frac{\mathbf{x}_n - \mathbf{x}}{\|\mathbf{x_n} - \mathbf{x_k}\|} \to \frac{\mathbf{y}}{\|\mathbf{y}\|}.$$

The set

$$T_A(\mathbf{x}) = \{\mathbf{y} \textit{ tangent to } A \textit{ at } \mathbf{x}\}$$

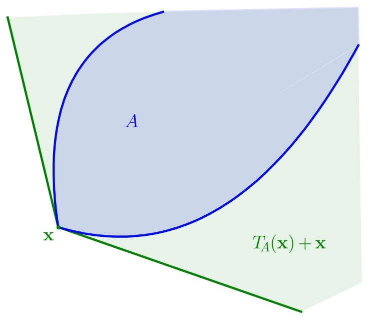
is called tangent cone *to A at $\mathbf{x}$.*

In the tangent cone $T_A(\mathbf{x})$ there are all the vectors $\mathbf{y}$ tangent to A in $\mathbf{x}$ where a vector $\mathbf{y}$ is tangent to A if $\mathbf{y} = 0$ or there exists a sequence of vectors $\mathbf{x}_k$ such that the directions of $\mathbf{x}_k - \mathbf{x}$ converge to the direction of $\mathbf{y}$. The definition is illustrated in Figure 3.3.



$\mathbf{x}$
$\frac{\mathbf{y}}{||\mathbf{y}||}$
$\mathbf{y}$
$\frac{\mathbf{x}_1}{||\mathbf{x}_1||}$
$\mathbf{x}_1$

(a) A tangent point $\mathbf{y}$ to A at a point $\mathbf{x}$.



(b) Tagent cone to a set A at a point $\mathbf{x}$.

Figure 3.3: Tangent point and cone.

Let's call $P_A(\mathbf{x})$ the set of directions that starting from $\mathbf{x} \in A$ can be followed remaining inside A:

$$P_A(\mathbf{x}) = \{\mathbf{y}\colon \exists\alpha > 0 \text{ such that } \mathbf{x} + t\mathbf{y} \in A \ \forall t \in [0, \alpha]\}$$

We list here some fact about T_A and P_A:

Proposition 3.5.2. *(i) if a vector is in T_A all its multiples are also in T_A;*

(ii) if $\mathbf{x}$ is in the interior of A then $T_A(\mathbf{x})=\mathbb{R}^n$;

(iii) if $f\colon A \to \mathbb{R}$ is C^1 and $\mathbf{x}^*$ is a point of local minimum for f then $\nabla f(\mathbf{x}^*) \cdot \mathbf{d} \geq 0$, $\forall d \in T_A(\mathbf{x}^*)$.

This last property extends the necessary condition for unconstrained local minima to the case where f is restricted to some set A, so that the minimum could be on the boundary of A.

Consider a generic problem with inequality and equality constraints

$$\begin{cases} \max_{\mathbf{x}} f(\mathbf{x}) \\ g_1(\mathbf{x} \leq 0, \ldots, g_k(\mathbf{x}) \leq 0 \\ h_1(\mathbf{x}) = 0, \ldots, h_m(\mathbf{x}) = 0 \end{cases} \tag{3.8}$$

and assume all functions are differentiable. As before we call Ω the feasible set,

$$\Omega = \mathrm{Dom}(f) \cap \{\mathbf{x}\colon g_1(\mathbf{x}) \leq 0, \ldots, h_k(\mathbf{x}) \leq 0, h_1(\mathbf{x} = 0, \ldots, h_m(\mathbf{x}) = 0\}\,.$$

We define the *feasible direction cone* at a point $\mathbf{x} \in \Omega$ as

$$D(\mathbf{x}) = \{\mathbf{d} \in \mathbb{R}^n \colon \nabla h_j(\mathbf{x}) \cdot \mathbf{d} = 0 \ \forall j = 1, \ldots, m, \nabla g_i(\mathbf{x}) \cdot \mathbf{d} \leq 0 \ \forall i \text{ s.t. } g_i \text{ is binding at } \mathbf{x}\}\,.$$

Then for every $\mathbf{x} \in \Omega$ one has

$$T_\Omega(\mathbf{x}) \subset D(\mathbf{x}).$$

Assumption 4. *We have $T_\Omega(\mathbf{x}) = D(\mathbf{x})$.*

When Assumption 4 is satisfied we say that Abadie Constrain Qualification (ACQ) holds at $\mathbf{x}$.
The ACQ condition is one of the weakest conditions under which we can formulate results about necessary conditions for constrained optimization with Lagrange multipliers:

Theorem 3.5.3. *If x^* is a point of minimum for f in Ω and Assumption 4 is valid at x^* then there exist $\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*$ such that*

$$\begin{cases} \nabla f\left(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*\right) + \boldsymbol{\lambda}^* \cdot \nabla G\left(\underline{x}^*\right) + \mu^* \cdot \nabla \phi\left(\mathbf{x}^*\right) = 0 \\ \lambda^* \cdot G\left(\mathbf{x}^*\right) = 0, \lambda^* \geq 0, G\left(\mathbf{x}^*\right) \leq 0, H\left(\mathbf{x}^*\right) = 0. \end{cases}$$

Assumption 4 is difficult to check on examples; the usual conditions that guarantee that it is satisfied are the following. Note that they include the usual NDCQ conditions of Assumptions 1,2,3.

Theorem 3.5.4. *1. If g_j and h_i are affine functions, ACQ is valid everywhere.*

2. If $\nabla g_j(\mathbf{x})$ for binding constraints and $\nabla h_i(\mathbf{x})$ are all linearly independent, ACQ is valid at $\mathbf{x}$.

Chapter 4

Convexity

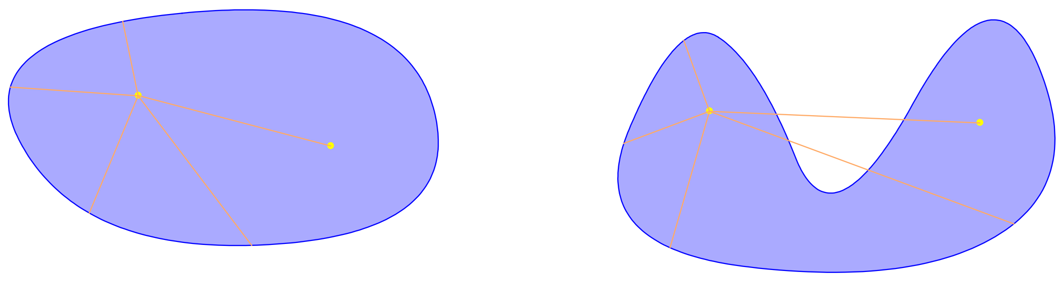
4.1 Convex sets

We call a subset X of a vector space V *convex* if for every two points x and y in X all the points on the line segments that connects them are in X. Formally,

$$\forall x, y \in X \text{ and } \forall t \in [0,1], (1-t)x + ty \in X \ . \tag{4.1}$$

X is a subspace, naming a set of points closed under addition and scaling. It can be formally defined as a span $V(v_1, \ldots, v_n) = \{\sum_{i=1}^{n} \alpha_i v_i \mid \alpha_i \in \mathbb{R}\}$ where we can find all the possible linear combinations. In matrix form, we can represent the same concept by setting up matrix A with all the vectors $v_1 ... v_k$ as columns and the range of A is the span.

Working in a convex set can be very useful in optimization problems as we will see later on. If we take a point in a convex bounded set set and move along a direction, we will reach the boundary at some point. In that case, we know we have explored all the points in the set in the direction we chose. If instead we take a non-convex bounded set there may exist straight lines moving along which we cross the boundary of the set multiple times (see Figure 4.1).



(a) A convex set in $\mathbb{R}^2$. (b) A non-convex set in $\mathbb{R}^2$.

Figure 4.1: Convex and nonconvex sets.

Convex sets have the following property:

Proposition 4.1.1. *If we have $X_1, X_2, \ldots, X_n$ convex sets, then their intersection $\bigcap_{i=1}^n X_i$ is also convex.*

Proof. If we take two points x and y in the intersection $\bigcap_{i=1}^n X_i$ and t between 0 and 1, we will have $x, y \in X_i \ \forall \ i$. Since X_i is convex $\forall i$, $(1-t)x + ty \in X_i \ \forall i$ and so $(1-t)x + ty \in \bigcap_{i=1}^n X_i$. □

If we have $X_1, X_2, \ldots, X_n$ convex sets, then their union is not necessarily a convex set; for example this is not the case for different sets that have an empty intersection.

Now that we have seen some concepts related to convex sets we can look at two important properties of functions of convex sets.

Proposition 4.1.2.

1. *If X is a convex set and $f : X \to \mathbb{R}$ is a linear function, then the image of X through f is a convex set.*
2. *If $D \subset \mathbb{R}$ is convex and $f : X \to \mathbb{R}$ is a linear function, also $f^{-1}(D)$ is convex.*

Proof.

1. Let's take two points x and $y \in f(X)$ and t between 0 and 1. This means they can be written as f of some elements in X. So we take x' and $y' \in X$ such that $x = f(x')$ and $y = f(y')$. Now we know that $(1-t)x' + ty'$ is also an element of X and $(1-t)x + ty = (1-t)f(x') + tf(y') = f((1-t)x' + ty')$, which is in $f(X)$.
2. To show that $f^{-1}(D)$ is convex, it has to hold that for any $x, y \in f^{-1}(D)$ and any $t \in [0, 1]$, the point $tx + (1-t)y$ is also in $f^{-1}(D)$. Since $x, y \in f^{-1}(D)$, we have

$f(x), f(y) \in D$, and since D is convex, for any $t \in [0,1]$, we have $tf(x)+(1-t)f(y) \in D$.

Now, let's consider the point $tx + (1-t)y$. Since f is a linear function, we have $f(tx + (1-t)y) = tf(x) + (1-t)f(y)$ and we know that $tf(x) + (1-t)f(y) \in D$. Therefore, $tx + (1-t)y \in f^{-1}(D)$.

□

4.2 Convex functions

Let's consider a vector space V, a subset $U \subset V$ and a function $f : U \to \mathbb{R}$.

Definition 4.2.1. *We say that f is* convex *if*

1. *U is convex;*
2. *For every $x, y \in U$*

$$f((1-t)x + ty) \leq (1-t)f(x) + tf(y). \tag{4.2}$$

We say that f is concave *if the inequality in (4.2) is reversed.*
We say that f is strictly convex *if the inequality in (4.2) is strict, meaning that*

$$f((1-t)x + ty) < (1-t)f(x) + tf(y), \forall x, y \in U.$$

This means that the graph of a convex function is below the segment connecting any two points $(x, f(x))$ and $(y, f(y))$. Indeed we can take two points in the domain of f and take a convex combination of them; the value of f at any convex combination will be smaller or equal than the combination of the values of f at the two points. Equivalently, if we connect the values of the image of those two points and we draw a segment, it will lie above the graph and this will be true for every two points taken and every value of t between 0 and 1.
We can also say that a function is concave if $-f$ is convex.
In a non-formal way we can say that convex functions curve up or do not curve (like affine functions). In the case of affine functions, equality is attained in (4.2). An equivalent way of defining convex functions is through the concept of epigraph:

$$\text{epi}(f) = \{(x,y) \in U \times \mathbb{R} \mid y \geq f(x)\}. \tag{4.3}$$

Here, we are expanding the formula to consider the pairs of x and y where the image of x through y is is smaller of equal than the y coordinate.

Theorem 4.2.2. *f is convex if and only if the epigraph of f is convex.*

A second characterization of convex functions is given restricting to one-variable functions.

Theorem 4.2.3. *A function $f\colon U \to \mathbb{R}$ is convex if and only if, restricted to every segment in its domain, is convex as a function of a single variable. Formally: $f\colon U \to \mathbb{R}$ is convex if and only if $\forall x, y \in U$ the function $g(t) = f(tx + (1-t)y)$ is convex.*

Another useful concept is that of sub-level set: we call *set of sub-level α for f* the set

$$S_\alpha = \{x \in U : f(x) \leq \alpha\} \ . \tag{4.4}$$

S_α is the set of points in the domain of f where $f(x)$ is less than α. This concept can be useful for optimization problems.

Proposition 4.2.4. *If f is a convex function then all its sub-level sets are convex.*

The reverse implication, however, is not true. Consider the function $f(x) = \log(x)$. For any constant c, the sub-level set is given by

$$S_c = \{x > 0 \mid \log_{10}(x) \leq c\} = (0, 10^c)\,. \tag{4.5}$$

These sub-level sets are convex. Consider two points $x_1 = 2$ and $x_2 = 4$, and let's take the midpoint $x_m = \frac{1}{2}(x_1 + x_2) = 3$. We have $f(x_m) = f(3) = \log_{10}(3) \simeq 0.48$. On the other hand, the midpoint of the function values at x_1 and x_2 is $\frac{1}{2}(f(x_1) + f(x_2)) = \frac{1}{2}(\log_{10}(2) + \log_{10}(4)) \simeq 0.45$. the function violates the convexity property.

We consider here only the case $V = \mathbb{R}^n$ with $n \geq 1$.
The most important link between convexity and optimization is the following result.

Theorem 4.2.5. *Let $f\colon U \subset \mathbb{R}^n \to \mathbb{R}$ be a convex function. Then every point of local minimum is a point of global minimum.*

Proof. Suppose that $\mathbf{x}^*$ is a point of local minimum for f; this means that there exists $R > 0$ such that

$$B_R(\mathbf{x}^*) := \{\mathbf{y} \in X : \|\mathbf{x}^* - \mathbf{y}\| < R\} \subset U$$

and $\forall\ \mathbf{y} \in B_R(\mathbf{x}^*)\ f(\mathbf{y}) \geq f(\mathbf{x}^*)$. Now let $\mathbf{z} \in U$ be a point outside $B_R(\mathbf{x}^*)$ and suppose that the function in $\mathbf{z}$ gives a lower value than in $\mathbf{x}^*$. Consider the segment between the two points $\mathbf{x}^*$ and $\mathbf{z}$; part of this segment is surely contained in $B_R(\mathbf{x}^*)$, therefore there exists $\bar{t} \in (0,1)$ such that $\bar{t}\mathbf{x}^* + (1-\bar{t})\mathbf{z} \in B_R(\mathbf{x}^*)$. Then we have

$$f(\mathbf{x}^*) \leq f(\bar{t}\mathbf{x}^* + (1-\bar{t})\mathbf{z}) \leq \bar{t}f(\mathbf{x}^*) + (1-\bar{t})f(\mathbf{z}) < f(\mathbf{x}^*)$$

which is impossible. □

We have not assumed differentiability of f until now, so we can investigate convexity properties based on how many times a function is differentiable. If a one-variable function $f\colon U \to \mathbb{R}$ is C^1 (continuously differentiable) then it is convex if and only if its derivative f' is increasing. Equivalently, a C^1 one-variable function $f\colon U \to \mathbb{R}$ is convex if and only if the graph of f is above its tangent lines at all points. Regarding secants, a one-variable function f is convex if and only if given a point $(x, f(x))$ in the graph of f, the segments that link it to any other point $(y, f(y))$ in the graph of f lies above the tangent line at $(x, f(x))$. We can express this concept in mathematical terms:

$$f(y) - f(x) \geq f'(x)(y-x)\ \forall x, y \in U. \tag{4.6}$$

In dimension n the concept is the same but the correct mathematical formulation is

Theorem 4.2.6. *A differentiable function f on a convex and open set $U \subset \mathbb{R}^n$ is convex if and only if*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot (\mathbf{y} - \mathbf{x})\ \forall \mathbf{x}, \mathbf{y} \in U. \tag{4.7}$$

It is strictly convex if and only if the inequality above is strict.

The right-hand side of (4.7) above is the first-order Taylor approximation of f at $\mathbf{x}$.

We have that for a global point of minimum $\mathbf{x}^*$, $\nabla f(\mathbf{x}^*) = \mathbf{0}$.
If for every point $\mathbf{y}$ in the domain of a convex function f we have

$$\nabla f(\mathbf{x}) \cdot (\mathbf{y} - \mathbf{x}) \geq 0$$

then

$$f(\mathbf{y}) - f(\mathbf{x}) \geq 0\ \forall \mathbf{y} \in U$$

so that $\mathbf{x}$ is a point of minimum for f. Therefore we have proved the following theorem.

Theorem 4.2.7. *If f is C^1 a convex function on an open convex set U, $\mathbf{x}^*$ is a point of global minimum if and only if $\nabla f(\mathbf{x}^*) = \mathbf{0}$.*

The necessary condition of Fermat's theorem is also sufficient if the function is convex.

If a one-variable functions f has second derivative then it is convex if and only if $f''(x) \geq 0$ for every x. In dimension n if f has all second derivatives then its Hessian is

$$H_f(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & & \vdots & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}.$$

Theorem 4.2.8. *A twice differentiable function f in a convex and open set $U \subset \mathbb{R}^n$ is convex if and only if its Hessian is positive semi-definite.*

If the Hessian is positive definite f is strictly convex but the opposite is not true.

Consider the function $f(x, y) = x^4 + y^4$, which is strictly convex. Taking the second derivatives, we have:

$$\frac{\partial^2 f}{\partial x^2}(x, y) = 12x^2, \quad \frac{\partial^2 f}{\partial y^2}(x, y) = 12y^2, \quad \frac{\partial^2 f}{\partial x \partial y}(x, y) = \frac{\partial^2 f}{\partial y \partial x}(x, y) = 0$$

and the Hessian matrix becomes

$$H_f(x, y) = \begin{bmatrix} 12x^2 & 0 \\ 0 & 12y^2 \end{bmatrix}$$

However, at the minimum point $(0, 0)$, the Hessian matrix is

$$H_f(0, 0) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Note that the signs of the formulas in the theorems above are reversed for concave functions.

Another special case of convex functions are the strongly convex functions.

Definition 4.2.9. *We say that function $f\colon U \subset \mathbb{R}^n \to \mathbb{R}$ is* strongly convex *if U is convex and there exists $\tau > 0$ such that $f(\mathbf{x}) - \frac{\tau}{2}\|\mathbf{x}\|^2$ is convex.*

Every strongly convex function is strictly convex.

Theorem 4.2.10. *A differentiable function f on a convex and open set $U \subset \mathbb{R}^n$ is strongly convex if and only if*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot (\mathbf{y} - \mathbf{x}) + \frac{\alpha}{2}\|\mathbf{x} - \mathbf{y}\|^2 \ \forall \mathbf{x}, \mathbf{y} \in U \ . \tag{4.8}$$

In dimension 1 This means that the graph of the function is above the tangent line the gap between the tangent and the graph is quadratic. In particular the graph of a strongly convex function cannot be flat anywhere.

4.3 Quasi-convex and pseudo-convex functions

Definition 4.3.1. *A function $f\colon U \to \mathbb{R}$ is* quasi-convex *if the sub-level set $S_\alpha(g)$ is convex for every $\alpha \in \mathbb{R}$.*

For quasi-concave functions, we change the sign of S or consider super-level set.
A convex function is also quasi-convex but the opposite is not true.
To show this we can refer to the same example given after Proposition 4.2.4.

Theorem 4.3.2. *$f \in C^1$ is quasi-convex if and only if*

$$f(\mathbf{y}) - f(\mathbf{x}) \leq 0 \Rightarrow \nabla f(\mathbf{x}) \cdot (\mathbf{y} - \mathbf{x}) \leq 0.$$

If f is quasi-convex, we cannot directly say that a local minimum is also a global minimum. Let's explore a second definition to find the condition we need to say that a local minimzer is also global.

Definition 4.3.3. *A C^1 function $f : U \to \mathbb{R}$ is* pseudo-convex *if*

$$f(\mathbf{y}) - f(\mathbf{x}) < 0 \Rightarrow \nabla f(\mathbf{x}) \cdot (\mathbf{y} - \mathbf{x}) < 0 \quad \forall x, y \in U. \tag{4.9}$$

Every convex function is pseudo-convex and every pseudo-convex function is quasi-convex. Pseudo-convex functions are those for which the condition of having zero derivative is sufficient for having a minimum.

Theorem 4.3.4. *If f is pseudo-convex and $\mathbf{x}^*$ is s.t. $\nabla f(x^*) = \mathbf{0}$ then x^* is a point of global minimum.*

If the set U is open and $\nabla f(x) \neq 0, \forall x \in U$, then f is pseudo-convex if and only if it is quasi-convex.

Chapter 5

Convex optimization

Now that we have analyzed some important properties of convex functions, let us use them for optimization problems.

5.1 Existence of a global minimum

We recall that a set Ω is closed if the boundary is included in Ω. Also, it is bounded if the distance between points is finite and so no distance goes to infinity. With two variables this is usually easy to understand.

Theorem 5.1.1. *If a function f is continuous and the set Ω is closed and bounded, then f has global maximum and minimum on Ω.*

As a corollary, if f is continuous and Ω is closed and there exists α such that

$$S_\alpha(f) \cap \Omega \neq \emptyset$$

is bounded, then f has a global minimum on Ω.
Two sufficient conditions for existence and uniqueness of a global minimum follow:

Theorem 5.1.2.
1. *If f is strongly convex and Ω is closed, there exists a global minimum of f on Ω.*
2. *If f is strongly convex and Ω is closed and convex the global minimum of f on Ω is unique.*

Theorem 5.1.3. *If f is continuous and coercive and Ω is closed the global minimum of f on Ω exists, where $f : \mathbb{R}^h \to \mathbb{R}$ is coercive if*

$$\lim_{\|\mathbf{x}\| \to +\infty} f(\mathbf{x}) = +\infty$$

Since the level set of a coercive function is bounded, we only say that the region Ω needs to be closed. So, this first condition imposed recalls Theorem 5.1.1. More specifically, we can choose a point $\mathbf{x_0}$ in the closed set ω; given $\mathbf{x_0}$ any minimizer must lie in

$$\Omega \cap S_{f(\mathbf{x_0})}(f) = \{\mathbf{x} \in \Omega : f(\mathbf{x}) \leq f(\mathbf{x_0})\}.$$

This set is closed and bounded, so the minimizer exists.

5.2 Convex Constrained Optimization

When we talk about a convex optimization problem, both the objective function f we want to minimize and the constraint set have to be convex. The typical problem we want to solve is

$$\begin{cases} \min f(\mathbf{x}) \\ \mathbf{G}(\mathbf{x}) \leq \mathbf{0} \\ \mathbf{H}(\mathbf{x}) = \mathbf{0} \end{cases} \tag{5.1}$$

where we set $\mathbf{G}(\mathbf{x}) = (g_1(x), \ldots, g_n(x))$, $\mathbf{H}(\mathbf{x}) = (h_1(\mathbf{x}, \ldots, h_k(\mathbf{x})$ and equalities and inequalities involving $\mathbf{G}$ and $\mathbf{H}$ are to be intended component-wise (meaning that they are applied to each element in the vectors). For example, linear programming problems are convex because their objective functions are linear and linear functions are convex.

We define the Lagrangian function $L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) - \boldsymbol{\lambda} \cdot \mathbf{G}(\mathbf{x}) - \boldsymbol{\mu} \cdot \mathbf{H}(\mathbf{x})$ and set up the corresponding system:

$$\begin{cases} \nabla L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = 0 \\ \boldsymbol{\lambda} \geq 0 \\ \mathbf{G}(\mathbf{x}) \leq \mathbf{0} \\ \mathbf{H}(\mathbf{x}) = \mathbf{0} \\ \boldsymbol{\lambda} \cdot \mathbf{G}(\mathbf{x}) = 0 \end{cases} \tag{5.2}$$

As with unconstrained optimization (Theorem 4.2.7), the necessary conditions of Theorem 2.1 becomes also sufficient in the convex case, that is, if $(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ solve the above system, we can say that $\mathbf{x}^*$ is a point of global minimum. We have indeed:

Theorem 5.2.1. *Suppose the objective function f and the feasible set Ω are convex. If $(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ solves system (5.2) and satisfies the NDCQ, then $\mathbf{x}^*$ solves problem (5.1).*

Let's take an example more related to the problem we are going to analyze later on. Consider

$$\begin{cases} \min \mathbf{x} \cdot Q\mathbf{x} + \mathbf{c} \cdot \mathbf{x} \\ A\mathbf{x} \leq \mathbf{b} \\ \mathbf{x} \geq 0 \end{cases} \tag{5.3}$$

where Q is positive definite, A is any matrix and $\mathbf{b}, \mathbf{c}$ are fixed vectors. As all constraints involve affine functions, the constraint set is the intersection of translations of vector subspaces of $\mathbb{R}^n$, which are convex; therefore the feasible set is convex and closed, being the intersection of convex closed sets. As, the matrix Q is positive definite, then the function $f(\mathbf{x}) = \mathbf{x} \cdot Q\mathbf{x} + \mathbf{c} \cdot \mathbf{x}$ is strongly convex (second derivative test) and so problem (5.3) has a unique solution.

Consider again the generic problem (5.1); its feasible set is

$$\Omega = \{\mathbf{x} \in \mathrm{Dom}(f) : g_1(\mathbf{x}) \leq 0, \ldots, g_m(\mathbf{x}) \leq 0, h_1(x) = 0, \ldots, h_k(\mathbf{x}) = 0\}$$

We assume that f is convex; by Proposition 4.2.4, if all the g_j are convex the sub-level set $\{\mathbf{x} \colon g_j(\mathbf{x}) \leq 0\}$ is also convex $\forall i$ and so their intersection $\bigcap_{i=1}^m S_0(g_i)$ is convex as well. Having all the functions h_i convex, however, is not enough to say that $\{\mathbf{x} : \mathbf{H}(\mathbf{x}) = 0\}$ is convex. For example the function $g(x) = \log x$ is not convex but the sub-levels are convex while if $h(x) = x^2 - 1$ then $\{x \colon h(x) = 0\} = \{-1, 1\}$ which is not a convex set.
Therefore we must require something stronger, and the easiest condition that ensures convexity of $\{\mathbf{x} : \mathbf{H}(\mathbf{x}) = 0\}$; a typical condition is that all functions h_i be affine, that is, $h_i(\mathbf{x}) = \mathbf{c}_i \cdot \mathbf{x} + b_i$.
We state this facts as a corollary to Theorem 5.2.1:

Corollary 5.2.2. *Assume that in problem (5.1) f is convex, the g_i's are convex and the h_j's are affine. If $(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ solves system (5.2) and satisfies the NDCQ, then $\mathbf{x}^*$ solves problem (5.1).*

We can weaken the assumptions as we did in Theorem 4.3.4.

Corollary 5.2.3. *Assume that in problem (5.1) f is pseudo-convex, the g_j are quasi-convex and the h_i are affine. If $(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ solves system (5.2) and satisfies the NDCQ, then $\mathbf{x}^*$ solves problem (5.1).*

5.3 Tangent cones and convexity

For convex sets we can say something more about convex cones and ACQ. We have:

Proposition 5.3.1. *(i) if A is convex then also $T_A(\mathbf{x})$ is convex for every $\mathbf{x}$;*

(ii) if A is convex, then $P_A(\mathbf{x})$ is the set of all vectors with direction $\mathbf{y} - \mathbf{x}$ for all $\mathbf{y} \in A$;

(iii) if A is convex, then $T_A(\mathbf{x})=\overline{P_A(\mathbf{x})}$(the closure of $P_A(\mathbf{x})$) for every $\mathbf{x}$;

(iv) if A is convex then $A \subset T_A(\mathbf{x}) + \mathbf{x}$ for every $\mathbf{x}$.

(v) if in problem (5.1) the g_j are quasi-convex and the h_i are affine, ACQ is valid everywhere.

As it happens with general unconstrained optimization, if A is convex and also f is convex then there is also a sufficient condition for optimality that extends Theorem 4.2.7:

Theorem 5.3.2. *If a C^1 function f is convex on a convex set A then $\mathbf{x}^*$ is a point of global minimum for f if and only if*

$$\nabla f\,(\mathbf{x}^*) \cdot \mathbf{d} \geq 0 \quad \forall \mathbf{d} \in P_A(\mathbf{x}^*).$$

This is useful when f is convex and the border of A is polygonal; in fact in this case we have $P_A(x) = T_A(x)$.

Chapter 6

Optimization algorithms

6.1 Gradient Descent

This general method is an algorithm used to solve unconstrained optimization problems. We will explain it referring to maximization problems and we will go deeper in more specific algorithms in subsequent paragraphs. By properties of directional derivatives, the gradient of a function in a point indicates the direction in which the graph of the function grows the most; therefore that should be the direction to follow in order to reach the maximum in a faster way. When we want to find the minimum, we want to compute the negative of the gradient. In a schematic way, the algorithm for the gradient method can be described as:

- choose a random point x_0
- calculate $\nabla\ f(x_0)$ and stop if it is equal to zero and if not choose another point $x_1 = x_0 + \eta\nabla f\ (x_0)$ finding a η that maximizes $f(x_1)$
- compute $\nabla\ f(x_1)$
- repeat the scheme.

A general formula to summarize these steps can be written as

$$x_{t+1} = x_t - \eta\nabla f(x_t) \tag{6.1}$$

where η is the step size and x_t is where we are after t iterations. The step size corresponds to the rate at which new information is accumulated by the algorithm. If it is too small, it will be computationally expensive and if it is too big there is the risk to diverge from the solution. As we mentioned above, this method is usually used for unconstrained problems; main examples include ordinary least squares or lasso models. This method can be adapted to deal with a constrained problem with a convex function f to maximize over a convex set.

To apply the traditional method without variations and adjustments that we will explore later on, we need two sufficient conditions to make sure the algorithm will find the optimal global minimum

- the function is differentiable so there exist a derivative everywhere
- the function is convex (4.2)

Let's see an application with a single variable function $f(x) = x^2 - x + 3$. We take first and second derivative respectively equal to $2x - 1$ and 2 and we notice that the second derivative is always bigger than 0 so the function is strictly convex and with univariate functions this condition is sufficient. If we take a quasi-convex function $f(x) = x^4 - 2x^3 + 2$ and take first and second derivative we will have $x^2(4x - 6)$ and $12x(x - 1)$. From the first derivative we derive two candidate points: 0 and 1.5. If we substitute those points back in the derivatives we will see that for the point equal to 0, those derivatives will be 0 and so this is a saddle point while 1.5 is a minimum. This means that if the algorithm finds the saddle point before the actual global minimum, it will not work. We can also summarize infletion points with the following so since 1.5 is bigger than 0 we know that at that point the function is convex.

- $x < 0$ convex
- $0 < x < 1$ concave
- $x > 1$ convex

For multivariate functions we will test this using the hessian. If at an inflection point the hessian is positive definite than the point is a minimum and if it is negative definite the point is a maximum. The algorithm calculates the next point to reach with the gradient at the current position and substracts in the case of minimization problems, to take a step. A saddle point imposes a challenge so that is the reason why we need convex functions. When we apply this algorithm we need to know if the function is differentiable.

While a necessary conditions to find a solution for a differentiable convex function is that the gradient at our candidate solution is zero $\nabla f(x^*) = 0$, this does not work when f is not differentiable. In this case one can introduce the concept of subdifferential and modify the algorithm accordingly. Let's take the function $f(x) = |x|$ as example and we want to minimize it. We know that the function is not differentiable at the minimum point.
Let us examine a few important theorems that connect the concepts we have seen on convexity and gradient descent algorithm.

Theorem 6.1.1. *If f is coercive and convex the sequence x_t has a subsequence that converges to a minimum point for f.*

Theorem 6.1.2. *If f is strongly convex the sequence x_t converges to the global minimum point for f, which is unique.*

Let's examine the case of convex constraints g_i with a convex function f; therefore the feasible set Ω is convex. We can transform the constrained problem in an unconstrained one. To do this we choose a new function $\tilde{f}$ such that $\tilde{f} = f$ in Ω e $\tilde{f} > f$ in $\Omega^{\complement}$. We then define a penalization function, for example

$$p(\mathbf{x}) = \sum_{i=1}^{m} \left(\max\{0, g_i(x)\}\right)^2 ;$$

if $\mathbf{x}$ is in Ω then $p(\mathbf{x})$ is zero, if $\mathbf{x}$ is not in Ω then $p(\mathbf{x})$ is positive, meaning that it penalizes when we are outside the feasible set: if we have a point outside, at least one constraint will be violated. We will then solve the unconstrained for

$$\tilde{f}_\varepsilon(x) = f(\mathbf{x}) + \frac{1}{\varepsilon} p(\mathbf{x})$$

to find the minimum in $\mathbb{R}^n$. We have:

- $\tilde{f}_\varepsilon(x)$ is convex.
- If $\mathbf{x}^*_\varepsilon$ is the solution for $\tilde{f}_\varepsilon(x)$ and it is in Ω, then we also have the solution for the constrained problem.
- If $\mathbf{x}^*_\varepsilon$ is not in Ω we take $\varepsilon' < \varepsilon$ having that in $\mathbb{R}^n$, $\tilde{f}_{\varepsilon'}(\mathbf{x}) > \tilde{f}_\varepsilon(\mathbf{x})$; we then choose a sequence $\varepsilon_k \to 0$ and $\forall k$ we find the corresponding sequence $\mathbf{x}^*_{\epsilon_k}$ that, at some point, we be inside Ω.

6.2 Projected Gradient Descent

Gradient descent improves strictly at every iteration and has a self tuning property. We can expand the use of this algorithm using projection operators; given a set X if we have a constrained minimization problem and need to stay in the set, we can use those operators to iterate while finding points in X. More specifically, those points are defined as

$$\mathbf{x}_t + 1 = proj_X(\mathbf{x}_t - \eta \nabla f(\mathbf{x}_t)) \tag{6.2}$$

where the projection on X, $proj_X(\mathbf{x})$, is defined as the point $\mathbf{y} \in X$ closest to $\mathbf{x}$, that is, the one among all points of X that minimizes the distance $\|\mathbf{y} - \mathbf{x}\|$. In the projected gradient descent algorithm, we have a feasible set where the projection of a point $\mathbf{x}$ is a point $\mathbf{y}$ belonging to the feasible region. When we iterate we follow the direction of the gradient and land in point $\mathbf{x}$ and then we need to project that point in our set finding the closest point $\mathbf{y}$ to $\mathbf{x}$. We keep doing this until we find our solution. Projected points are closer than original points:

$$\|\mathbf{y}_1 - \mathbf{y}_2\| \geq \|\mathrm{proj}_X(\mathbf{y}_1) - \mathrm{proj}_X(\mathbf{y}_2)\|. \tag{6.3}$$

If X is convex then the projection on X is unique.

As an example let us take a multivariate function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathbf{T}}\mathbf{Q}\mathbf{x}$ with a matrix Q positive definite. The constraint is $\|\mathbf{x}\| = 1$. The projection operator is this case is defined as

$$\mathrm{proj}_{\|\mathbf{x}\|=1}\,\mathbf{x} = \frac{\mathbf{x}}{\|\mathbf{x}\|}.$$

We fix a step size η and compute the gradient $\nabla f(\mathbf{x}) = \mathbf{Q}\mathbf{x}$ and take a step

$$\begin{aligned}
\mathbf{x}_{\mathbf{t+1}} &\leftarrow \mathbf{x}_t - \eta\mathbf{Q}\mathbf{x}_t = (\mathbf{I} - \eta\mathbf{Q})\mathbf{x}_t \\
\mathbf{x}_{t+1} &\leftarrow \frac{(\mathbf{I} - \eta\mathbf{Q})\mathbf{x}^t}{\|(\mathbf{I} - \eta\mathbf{Q})\mathbf{x}^t\|}.
\end{aligned}$$

Notice however that the constraint is not convex, and indeed some points (like the origin) cannot be projected on the circle in a unique way. This may lead to computational errors.

6.3 Other methods

6.3.1 Mirror descent

The direction used to chose the next point can also not be the gradient. Mirror descent algorithms use other functions in place of the gradient.

6.3.2 Frank Wolfe

This algorithm is an alternative to projected gradient descent. Let us take the formula of projected gradient descent for reference

$$\mathbf{x}_t + 1 = proj_X(\mathbf{x}_t - \eta \nabla f(\mathbf{x}_t)) \tag{6.4}$$

Frank Wolfe algorithm (FW) always stays inside the set and follows the steepest direction indicated until the border of the set to reach the solution. If we think of a typical LP problem in two dimensions, when we define a convex set made of linear constraints we look for the maximum or minimum is the corners. Here the concept is similar as FW explores the corners but without always following the direction of the gradient. On the contrary of projected descent, FW replaces projection with linear minimization or maximization depending on the problem.

6.3.3 Newton

This algorithm employs quadratic convergence meaning that one iteration doubles digits of accuracy. We need to assume that f has first and second derivatives while with gradient descent we had a linear approximation of f. There are two phases, when we are close to the solution there is no step size and so it is said to be undamped while when we are not very close we use damped. We give the formulas for single variable and multivariate functions:

$$x_{t+1} = x_t - f'(x_t)/f''(x_t)$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t - H(\mathbf{x}_t)^{-1} \nabla f(\mathbf{x}_t)$$

Let's see an example with a multivariate function $f(x, y) = xy + 4y - 3x^2 - y^2$ with a random point to start equal to $x_0 = (1, -1)$. We then define the gradient of f evaluated at point $x - 0$ with the partial derivatives with respect to all the variables and the Hessian with all the second partial derivatives evaluated at x_0.

$$\nabla f = \begin{bmatrix} y - 6x \\ x + 4 - 2y \end{bmatrix}$$

$$\nabla f\left(x_0\right) = \begin{bmatrix} -7 \\ 7 \end{bmatrix}$$

$$H\left(x_0\right) = \begin{bmatrix} -6 & -1 \\ 1 & -2 \end{bmatrix}$$

The we evaluate the first iteration following the formula and go on until we reach the maximum number of iterations decided in advance.

Part II

Chapter 7

Betting systems

Models for parimutuel markets have been introduced as a different betting model, opposed to the popular fixed odds ones that we usually see for bets in sports. In fixed odds betting, there are fixed quotes (odds) available on returns on wagers and when bets are placed the odd associated is the current one which will be fixed for that specific bet. This means that when the odd changes depending on what other players do, the fixed odd a player saw at the time of the bet does not change for his bet. On the other side, the odds in parimutuel markets fluctuate as bets come in and the final odds are calculated at the end and those will be the finalized ones. Parimutuel markets are modeled as specific betting settings where the organizer does not incur any loss as all the money paid to bet are pooled together and redistributed to players who bet on the event realized. This will result in a proportional redistribution of money corresponding to the ratio of money bet initially. There are many variations, especially regarding the payout rules but in any case, the condition of protecting the organizer from losses is respected. The setting of these markets takes inspiration from auctions. The two can be organized in a very similar manner but this always depends on the specific rules and structure of the mechanisms. For example, in an auction, participants typically bid on a single item or lot, and the highest bidder wins that item. In contrast, in a parimutuel model, participants bet on a specific outcome of an event, such as a horse winning a race or a particular team winning a game. However, both parimutuel models and auctions involve participants placing bets on an outcome, with the total amount wagered being distributed among the winners. Moreover, the setting of parimutuel models is flexible in the sense that it is possible to place bets on more complex events rather than just a single outcome. In any case, the strategy depends on what the other participants do, as opposed to the case of fixed odds betting.

7.1 Fixed odds bettings

In fixed odds bettings, odds are calculated with the ratio of the probability of an event happening by the probability of not happening and are showed with fractions, decimal numbers or integers (money-line) number depending on the system adopted in each country. Fractional odds (e.g. 2:1) show the amount the player could win with respect to the stake to be bet, to win that amount. Decimal odds (also called European odds) show the return on a unit stake and the last form, money-line odds, show either the amount a player can win with 100 dollars (plus odds) or the amount a player has to put at stake to win 100 dollars (minus odds). If for example I bet with the fixed odd of 2:1 and my stake is 10, I get 20 and my total return is 30 since i will get also my stake back. If I convert this example in the other ways odds are showed, it would correspond to 3.0 for European odds and +200 for money-line(plus odds).
From the odds it is possible to calculate the implied probability of an event happening. We will show the formula to find implied probability for decimal odds, the system commonly used in Europe. It is very simple as the only calculation is dividing 1 by the odds. If we have odds equal to 3.0, the implied probability will be equal to 1/3 and so 0.33. For the UK system, where fractional odds are used, this would correspond to the denominator divided by numerator and denominator added together so 1/(1+2). For the American plus odds we would need to take 100/(100+ 200). For American minus odds we would need to convert the negative number in positive and suppose we have the same odds of the positive case, we would take 200/(100+200). This means that for higher odds, the probability decreases.

It is very important to understand how odds change, knowing how money influence movements in the market. At the beginning odds are calculated by the organizer of the betting system taking into consideration the strenght of participants, specific information and expectation on where money will be placed. As the betting goes on, odds are adjusted on new information, market confidence (liquidity injected) and flow of money. When a high amount of money is bet on at outcome, it means bettors are more confident and that outcome will have a relatively high implied probability. In this case odds will be adjusted to a lower value. This reflects the fact that when there is a higher probability for that outcome to realize, the risk is lower and so the odds will be lower as for any unit of money bet, the money gained will be proportionally less. At the beginning of the betting, there won't be much money in the system and the odds will not reflect the expectation of the players much. As more information is reflected in the odds, confidence in betting increases and the inflow of money will increase as well. When the game is about to start and the betting period is closing, the odds will reflect more closely the sentiment of the players and so it makes more sense to place bets at this moment.

From the point of view of the organizer, it is better to update the odds taking into account where the money are bet as this will prevent arbitrage opportunities to arise. This can happen when the probability of an event happening is different from the one reflected by the odds. In this form of betting the organizer takes risks and so it is better not to create riskier settings where he could lose more.

Suppose the players know the current fixed odds for a specific setting, let's say 4:1. Then, certain information come out and the probability for an event to happen increases, in this case the odd (European method) will be a lower number let's say 2:1. Suppose the organizer sets the odd at 3:1 (probability equals 0.33) and does not move it for some time. If players understand the difference in the probability of an event happening and the one reflected by the odds, they will start to bet more for that outcome meaning that for every unit bet they will receive 3 instead of the one closer to the appropriate which would be 2 in this example. In this case the organizer will be worse off since the players are getting paid more than they should, and the money collected with the bets may be not sufficient to pay the players out.

7.2 Parimutuel markets

In traditional parimutuel markets, the odds are continuously changing depending on what the participants do. Also, the odds are not set at the beginning and are only available at the end, when the betting ends and the game starts. The final odds will be used to calculate the implied probability. All the money bet is put together and for every outcome it is calculated the payoff to be divided among the participants who bet on that outcome. It is found by dividing the money bet for that outcome by the total money bet. Even though the odds fluctuate while the betting is open, the effective odds will be the ones calculated at the end. In this case, it is like players are playing against each other and not against the organizer as odds reflect only the relative volume of money in the system and it is not something set by the organizer. This means that the final odds fully reflect the market sentiment without considering direct information aggregation. In financial terms, the implied probabilities are called prices and so we can say that in parimutuel markets the relative prices equal the relative aggregate amounts wagered.

The implied probability is the probability distribution implied by (dollar) betting volume. In financial terms odds and prices reflect the relationship between supply and demand. We can think of a random variable as a financial asset that has associated outcomes with corresponding probabilities. The payoff can be expressed in probability as it will be the same as the share of money bet with respect to the money in the pool to be distributed among winners. Thus, the state price where the state is a specific outcome, is just the probability of that outcome happening, that is, of the random variable taking a particular value, and so the sum of prices over all the states is equal to 1. The worth of the financial asset is equal to the expectation of the random variable. So, we can say that prices reflect the distribution of the random variable.

7.3 Parimutuel Mechanisms in Financial Markets

In financial markets, the usage of centrally managed markets is very common. As opposed to decentralized markets, there is a unique dealer that manages the market. In this form it is possible to place different kinds of orders through brokers or the exchange. A broker trades for others for commissions and a dealer trades for himself and profits for a markup. The stock market in general is a dealer and broker market. For example, the NYSE is a broker market in the form of a continuous double auction, where brokers represent their clients' orders and try to match them with orders from other clients or with orders in the exchange's central order book. The NASDAQ is a dealer exchange. The main difference is how trades are transacted. At the NYSE the auction system is used to set prices at the market open and close and participants transact directly between each other. Before the official opening it is possible to submit orders that are then matched with the highest bidding price paired with the lowest asking price. The market is maintained by designated market makers that are human points of contact on the trading floor and run the opening and closing auctions even though nowadays most of it is computerized. On the other side, in the NASDAQ system participants transact through dealers. Market makers maintain inventories of stock to buy and sell from their accounts and to transact they give two sided quotes with bid and ask prices. Given the similarity of some characteristics of the parimutuel markets and some settings already present in trading systems in financial markets, it had been possible to directly implement those models for financial transactions.

There are a variety of models for parimutuel markets implemented for different kinds of financial settings. We will illustrate the most popular ones. As we have already mentioned, the NYSE sets prices with a continuous double auction system (CDA). For this system various models have been employed to set prices starting from the standard parimutuel model and applying innovative variations as well. Recently, these models have been also applied to different real-world markets like Yahoo, TheWSX and InklingMarkets. In the typical CDA system after the opening and before the closing, offers are placed continuously and whenever there is a cross-over of orders there will be a transaction. In reality, the mechanism is much more complex than this, depending on the asset traded and the kinds of orders available. In finance, we refer to the organizer of the market as the market maker. An important characteristic is that the organizer does not encounter any risk as the purpose is to find bilateral agreements. In fixed odds models, the bookmaker sets the odds and adjusts for demand and information. Again liquidity is provided but this time risk is involved. In parimutuel models for some specific settings like sports, the events are mutually exclusive and traders put their money wagering on events. When the winning event is declared the money is given proportionally to the people who bet on the winning event. All the money put by the people who lost go to the people who won. It is possible to think of the payoff in two ways, either the refund bet plus the money taken from the people who lost or the share of money bet over the total amount in the pool of money. The first to study a CDA applying a variation of the standard parimutuel model was Bossaerts [Bossaerts et al., 2002], who solved a contingent claim call auction market with a linear programming formulation. Then, these kinds of models have also been applied to prediction markets with additional variations like the Logarithmic Market Scoring Rule (LMSR) by Hanson[Hanson, 2002] and the Dynamic Pari-Mutuel Market (DPM) by Pennock [Pennock, 2004], see also [Agrawal et al., 2009]. In this kind of market it is easier to see similarities with other types of bets, in sports and horse racing. Both mechanisms operate as automated market makers. Moreover, in the DPM, bets don't have to be placed before the game starts, as the prices (probabilities) will change so it has the property of updating dynamic information. On the other side there is the problem of liquidity, as when there is nobody willing to sell due to scarce interest there will be also nobody willing to buy, so the DPM considers the amount of people willing to trade in the market. This can happen mainly when the model is applied to sports events as traders will be more willing to sell and buy despite real events. The concept is the same and it is based on a call auction mechanism. This mechanism has been also employed by many banks like Goldman Sachs which was based on a model developed by Lange and Economides (PDCA)[Lange and Economides, 2003]. The organizer collects the orders and closes the market to determine which orders to accept so traders cannot know if the order is accepted until the market closes. While the other two models mentioned above, LMSR and DPM, run as an automated posted price market maker, the PDCA is implemented as a call auction where traders need to reveal prices. An interesting implementation of an auction parimutuel model is the Sequential Convex Pari-Mutuel Mechanism(SCPM) where the organizer takes immediate decisions on the orders. This model derives from the PDCA which has also been solved with a convex formulation called CPCAM (see Chapter 8). These kinds of models are based on the auction principle and are created to generate liquidity and avoid thin market problems that can arise in CDA systems.

Chapter 8

Mathematical structure

We have n traders and i possible outcomes (states). Trader j provides an order made of a bet, that is a vector a_{ij}, $i = 1, \ldots, S$, with $a_{ij} = 1$ if he wants to bet on outcome i occuring and $a_{ij} = 0$ otherwise. He also provides a limit quantity q_j corresponding to the maximum number of identical bets he is willing to submit and the maximum price π_j he is willing to pay for each bet. The market organizer has to find the order fill x_j for each trader, that is, how many of trader j's bets are accepted, the price p_i for each state and the price c_j requested to trader j for his bet. We denote by A the matrix formed by row vectors a_{ij}. Notation is recalled in table 8.1.

Variable	Name	Description
$a_{i,j}$	State Order	Trader j's order on state i
q_j	Limit Quantity	Trader j's maximum number of orders requested
π_j	Limit Price	Trader j's maximum price for order
c_j	Equilibrium price	Trader j's equilibrium bid price
p_i	Price	Organizer's price level for state i
x_j	Order Fill	Number of trader j's orders accepted

Table 8.1: The variables for the PMM and CPCAM models.

When applying these models to a setting specific to options, each a_ij represents the potential payout for the order j if the state i is realized. In the following examples we will concentrate on digital options as the setting is much easier and we have a fixed payoff of 1 for each order in the realized states. In this case matrix A is filled with binary elements with 1 if the trader wants to bet in that state and 0 in the other case.

Models in parimutuel markets are similar in terms of the constraints as they all endow the defining features of those markets, listed below:

- riskless funding of claim payouts
- equilibrium pricing conditions requiring the relative prices of contingent claims equal the relative aggregate amounts wagered on such claims
- endogenous determination of unique state prices
- call auction, non-continuous trading

8.1 PMM

One of the most famous modifications of the traditional implementation is the model developed by Lange and Economides in [Lange and Economides, 2003] (PMM) which will be explored more in the detail in the next chapter.

$$\begin{cases} \max_{\mathbf{x},\mathbf{p}} M = \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{s} \theta_i \\ \sum_i p_i = 1 \\ c_j = \sum_i a_{i,j} p_i \ \forall j \\ M = \sum_j x_j a_{i,j} + \frac{\theta_i}{p_i} \ \forall i \\ c_j - \pi_j + y_j \geq 0 \\ x_j \left(c_j - \pi_j + y_j\right) = 0 \\ y_j \left(q_j - x_j\right) = 0 \\ 0 \leq x \leq q \\ p > 0 \\ y \geq 0 \end{cases}$$

Here we are maximizing the variables $\mathbf{p}$ and $\mathbf{x}$. c_j represents the equilibrium price of the option in order j so we can say that

$$c_j \equiv \sum_{s=1}^{S} a_{j,s} p_s$$

Now let's define the total premium paid in the auction

$$M \equiv \left(\sum_{j=1}^{J} x_j c_j\right) + \sum_{s=1}^{S} \theta_s$$

and y_s is the payout order trader j receives if the state occurs represented as aggregated customer payout.

$$y_s \equiv \sum_{j=1}^{J} a_{j,s} x_j$$

8.2 CPCAM

We now introduce the CPCAM model which is a convex formulation similar to the previous model.

$$\begin{cases} \max_{\mathbf{x},M,\mathbf{s}} \sum_{j=1}^{n} \pi_j x_j - M + \sum_{i=1}^{S} \theta_i \log(s_i) \\ \sum_j a_{i,j} x_j + s_i = M \\ 0 \leq x \leq q \\ s \geq 0 \end{cases}$$

Here we are solving for M, $\mathbf{x}$ and $\mathbf{s}$. We will show in the next chapter that $\mathbf{s}$ is equal to $\frac{\boldsymbol{\theta}}{\mathbf{p}}$. θ is the vector of starting orders and $\pi^T x - M$ is the profit for the organizer.

The second term can be also expressed in the following way, representing a disutility function

$$\sum_i \theta_i \log(s_i) = \sum_i \theta_i \log\left(M - \sum_j a_{i,j} x_j\right)$$

So, the value of the objective function will increase when the organizer is able to maximize the profit while maintaining the parimutuel properties.

8.3 Parimutuel markets in options trading

In finance, parimutuel models are mainly applied to prediction settings but this is flexible depending on the purpose of the application. The main purpose for the creation of these markets in finance is to help generate liquidity. This is usually done through a centralized organizer who facilitates trades between participants. In OTC markets, for example, parties trade directly between each other and for specialized or unique financial assets like customized options, it may be hard to find a counter-part.

We are interested in studying a similar setup but for contingent claims and limit orders. Note that all trading strategies are implemented with orders in the form of notional amounts since derivatives contracts are based upon the notional amount to be bought or sold. However, in the specific examples we won't introduce details on more complex structures of contracts. In reality, the purchase of an option will correspond to a desired size of the position in notional terms.

Let us define the setup of a PDCA parimutuel contingent claim microstructure which can be also extended to other setups. The strategies can be implemented with a buy or sell order and as result we will have a vector of payout ratios corresponding to a range of states and a limit price. We denote the value of an underlying variable as U, for example referring to an economic index. Before the start, the exchange or market will typically set a range of strike prices based on various factors, such as the current market conditions, the underlying asset being traded, and the expiration date of the option. They are set across the range of likely outcomes of the asset and denoted as $k_1, ...k_s - 1$. S states are formed, paying out only if that state occurs. We can imagine the underlying asset to have a pricing range and each strike price represents an interval in that range. We have $S - 1$ states because the strike prices are placed at the boundaries between the trading states. Before opening the auction the opening orders denoted as θ are entered for each of the S state contingent claims to ensure parimutuel prices are unique. Then customers denoted with $j_1, ...j_n$ submit orders and request a specific amount of contracts denoted with q_j. Contracts have different type and each type has a different payout, for example digital options pay 1 if the option expires in the money. For vanilla options the payout is 1 per point of the option being in the money. In reality, it is possible for a trading post to trade different kinds of options that refer to the same underlying variable. Also in parimutuel markets different betting options with different payout structures can be offered even though the majority of models developed in academia treat only digital options as the structure is easier to implement. In this model both digital and vanilla options are allowed. Customers can specify limit prices for each order, representing the maximum price when they are trying to purchase and the minimum price when they are trying to sell. It will be denoted as π_j for customer order j. In our notation, a_j, s will represent the order of trader j for state s in terms of the notional payout amount which depends on the type of option. For example, if the type of option for the contingent claim for trader j is a digital put option and the outcome is that $U < K_1$, $a_j, 1$ will be 1, in the first entry of a vector a_j, s with all the replication weights of the whole order. If trader j has a buy order, then the claim pays out if the strike price at a particular state is k_v and U is greater of equal. If the order is vanilla, the claim pays depending on the spread k_v and k_w. The vector p_s represents the implied probability that the state occurs and that the claim expires in the money.

8.4 Discussion

We can think of solving the following models by first identifying the optimal vector **p** and then solve for the optimal x_j or viceversa. In the CPCAM the market organizer collects the limit prices for the accepted orders and in the PMM, the optimal prices are taken. It has been showed that if we collect the parimutuel prices also in the CPCAM model the optimal prices don't chnage. In general we denote the parimutuel price of participant j with the variable c_j in the case this is the price charges and not the limit price. Here are the two ways to approach this

$$\begin{aligned}
&\pi_j < c_j \rightarrow x_j = 0 \\
&\pi_j = c_j \rightarrow 0 \leq x_j \leq q_j \\
&\pi_j > c_j \rightarrow x_j = q_j
\end{aligned}$$

$$\begin{aligned}
&x_j = 0 \rightarrow p^T a_j \geq \pi_j \\
&0 \leq x_j \leq q_j \rightarrow p^T a_j = \pi_j \\
&x_j = q_j \rightarrow p^T a_j \leq \pi_j
\end{aligned}$$

or written to solve the PMM model which charges the parimutuel prices

$$\begin{aligned}
&x_j = 0 \rightarrow p^T a_j = c_j \geq \pi_j \\
&0 \leq x_j \leq q_j \rightarrow p^T a_j = c_j = \pi_j \\
&x_j = q_j \rightarrow p^T a_j = c_j \leq \pi_j
\end{aligned}$$

One of the variables of interest is the vector of prices for each state. Prices represent implied probabilities and reflect the pricing of the specific assets in the model. Let as define the vector y as the vector of elements representing the aggregate payout of each state, so the total amount paid for the claim in each state. One important equality is that relative prices are equal to the aggregate relative amount paid. Also unique prices are determined endogeneously. In the usual setup orders are expressed with notional amounts (explain) and orders are limit orders. If we imagine this call auction mechanism in a financial setting, we will have a non continuous trading situation and a risk free funding of payouts using the amounts paid for all the claims during the auction. Moreover, the total amount paid for the claims, is exactly sufficient to pay for contingent claims having positive return. So, the mechanism is self-funded and risk-neutral since the total premium paid for the claims is equal to the state contingent payouts. After the betting is closed, we will have a vector of prices as outcome which will represent an elemental state outcome and will be S dimentional. In the models we will describe for parimutuel markets, the odd refers to the payout system. Let as remind that in the traditional game, the organizer quotes the odds on specified wagers while in parimutuel games, the prices fluctuate until the betting is closed. The prices are set depending on relative amounts. So, for every state i, p_i will tell us how much the investors whose claim is in the money has the probability of earning.

In these kinds of event-based settings, liquidity may be a problem especially when the set of traders is small and the kinds of tradeable securities is diverse. That is why these markets are usually centrally organized. In this way it will be easier to create specific claim payouts. Moreover, we can specify some features like the kinds of orders and assets traded. In this way the organizer knows how to manage trades and can decide what to accept or reject. In the case of limit orders traders express limit prices to buy or sell the security in question. This is usually a feature that is included since it results in less trades and more efficiency consequently. Another important feature we would like to have in a balanced book for the organizer, so that there can never be a liquidity risk for availability of assets to trade. In fact, in a usual setting traders will be able to buy and sell. An important specification for this model is the objective function which can be either in terms of number of orders accepted or value of orders. The trader will know if the order is being considered at the end, when prices will be announced. At this point for every order the the trader will see what the order state price is, namely the sum of prices of individual states. Then, the order fill will be communicated as well. Orders can be accepted fully, in part or not accepted depending on the optimal price calculated and the limit price of the order. This way to solve the model is different from the traditional setting, where all orders are accepted and participants are charged a fixed amount of money. The problem is that the odds are continuously changing while in the varied settings, the acceptance of orders depends on the optimal prices and the odds will not vary when the auction is open because the optimal order fill is still not determined. Both models we are going to analyze will represent auctions with limit orders.

8.4.1 PMM

The model will be run in a call auction setting. Now we will explain the main variables and some conditions. First of all, the vector of optimal prices obtained from the model follows to normalization condition of

$$\sum_{i=1}^{s} p_i = 1, \mathbf{p} > 0.$$

Then, the total amounts paid for all contingent claims are equal to the total contingent payouts.

$$\frac{p_s}{p_k} = \frac{y_s p_s}{y_k p_k} = \frac{\left(\mathbf{y}^T \mathbf{p}\right) p_s}{\left(\mathbf{y}^T \mathbf{p}\right) p_k} s, k = 1, 2, \ldots, S$$

where p_s and y_s are the s^{th} elements of vectors p and y. This means that the relative prices of each fundamental state contingent claim is equal to the aggregate relative amounts paid for the respective claims as we said before. In general, the parimutuel markets are designed with contingent claim prices derived endogenously and so, the demand of the investors has an effect on the price. This is what connects these models with the so-called market games.

With respect to the optimization problem introduced above, in order for the contracts to be accepted by the market organizer, some conditions have to hold

$$\pi_j < c_j \rightarrow x_j = 0$$
$$\pi_j = c_j \rightarrow 0 \leq x_j \leq q_j$$
$$\pi_j > c_j \rightarrow x_j = q_j$$

and this relates to one of the most important properties of parimutuel markets, which is to bet truthfully with respect to the valuation of the option.

One important condition is that there is sufficient premium to fund any state

$$y_s + \frac{\theta_s}{p_s} = M \quad s = 1, 2, \dots, S$$

where on the left-hand side we have the total amount of customer payout plus the notional payout amount of the opening order. This side represents the the total payout to be made if state s occurs, so the amount collected is equal to the amount needed to settle the total filled requests. Also, the lower the payouts y_s, the lower the price of the state contingent claim p_s. Thus the aggregate demand for a particular state can be explained as follows

$$m_s = p_s y_s + \theta_s, s = 1, 2, \dots, S$$

which implies

$$\frac{m_s}{m_k} = \frac{p_s}{p_k} \quad s, k = 1, 2, \dots, S$$

Given the demands for the orders, there exists a unique parimutuel equilibrium. We now proceed to setup a more specific model with limit orders. To reflect the reality of how centralized markets work, we conceptually aggregate liquidity in the same state space allowing bets for any type of contingent claim. This is not usually the case for non-financial bets. For example in horse races, pools of money will be separate for ”win” bets and ”place” bets. The model is formulated as an optimization problem with an objective function that aims at maximizing the volume of orders, so we want to have as much liquidity as possible. We can write the same problem in more ways, depending on how we modify the payout system. For example if we want to solve the PMM introducing the traditional payout of parimutuel models, we will introduce the variable y_s representing the optimal payout of each state but if we want to work with a simpler case we can set a fixed payout equal to

1, as in the second problem down below where y_j is a slack varibale.

8.4.2 CPCAM

The CPCAM model is a convex modification of the PMM model. It is employed in call auction settings, as the PMM model. It has been introduced mainly in order to overcome two issues in the original PMM formulation: the non-convexity of the optimization problem, which leads to difficulties in solving it (one of the main drawbacks is that there is no guarantee that solving algorithms converge in polynomial time), and the difficulty in quantifying the influence of the starting orders θ_i on the solution of the system. The CPCAM model has been proposed and studied in [Peters et al., 2006]. It has been shown that it models lead to the same solutions as the PMM and since we have two variables of interest we can also think of solving this model with respect to prices first and use the solution of the price vector for the PMM model to then go on and solve for the other variable representing the optimal order fill. If we do this, the PMM model will be solved as a linear program. This interchangeability of solution with respect to prices is possible because of the property of price uniqueness of both models. This means that if we input the same seed of money to fund the auction at the beginning, the solution is unique. So, we would need to start the two models with the same seed vector of starting orders θ. In the original paper where the CPCAM method is presented, it is shown how the constraints present in the PMM formulation can be analytically derived following the steps needed to solve the constrained problem with Lagrange. After the optimal values are communicated are the realized outcomes are clear, traders get a fixed payout if the order is accepted, as in the PMM. This is like saying that from the beginning traders know what the odds will be and the actual payout will depend on the optimal values. We can view the fixed payout as a scaling factor for the results to adjust the proportionality of bets on the total amount bet. In the traditional model this is equal to 1 but we can easily modify the payout mechanism and adapt it to the level of risks of specific settings in the financial markets. In the application of this model we assume that there are limit orders so that traders know the payout if the order is accepted and the states are realized. There us a requirement of nonzero starting orders. There is a difference with the PMM model where traders will pay $p^T a_j$ for the order if it is accepted, not the limit price. We also have the price consistency constraint of $p^T a_j \leq \pi_j$ for accepted orders, so the organizer could change a higher price than the optimal and keep profits. These are the price consistency constraints for the PMM model

$$x_j = 0 \longrightarrow p^T a_j = c_j \geq \pi_j$$

$$0 < x_j < q_j \longrightarrow p^T a_j = c_j = \pi_j$$

$$x_j = q_j \longrightarrow p^T a_j = c_j \leq \pi_j$$

Traders will be charged their limit price but it is also demonstrated that the vector of unique optimal prices does not change in case we decide to charge the optimal state prices instead. The optimization problem is convex and can be solved deriving the Lagrangian and the associated KKT conditions.

$$\begin{aligned} L(x, M, s) =& \pi^T x - M + \sum_i \theta_i \log(s_i) \\ & - \sum_i \mu_i \left(\sum_j a_{i,j} x_j + s_i - M \right) \\ & + \sum_j \lambda_j (x_j - q_j) \end{aligned}$$

and we derive the KT conditions which include some of the constraints of the PMM model.

$$\begin{aligned} \pi_j - \sum_i p_i a_{i,j} + \gamma_j &\leq 0 \quad \text{for } 1 \leq j \leq n \\ x_j \left(\pi_j - \sum_i p_i a_{i,j} + \gamma_j \right) &= 0 \quad \text{for } 1 \leq j \leq n \\ \sum_i p_i &= 1 \\ \frac{\theta_i}{s_i} - p_i &\geq 0 \quad \text{for } 1 \leq i \leq S \\ s_i \left(\frac{\theta_i}{s_i} - p_i \right) &= 0 \quad \text{for } 1 \leq i \leq S \\ \gamma_j (x_j - q_j) &= 0 \quad \text{for } 1 \leq j \leq n \\ \gamma &\leq 0 \end{aligned}$$

One important condition we derive is the relationship of optimal variables of interest which can be derived from the complementarity slackness property of KT. It has the application as in the PMM model but following this conditions we derive the optimal variables in the opposite way as before. However, as we have specified above this does not change anything in the solution. Moreover, the organizer collects the limit prices for the accepted orders but it has been shown that collecting the optimal prices $\mathbf{p}$ does not change the optimal solution.

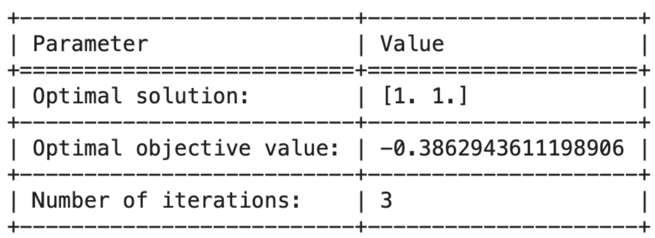
Chapter 9

Simulations

We have solved the CPCAM optimization problem using the SLSQP method in the Scipy library of Python. We have seen how in the paper of [Peters et al., 2006] the main properties of the parimutuel model are explicitely shown by solving the problem with Lagrange. With the KT formulation, some of the constraints of the PMM have been derived through this formulation while in the PMM they had been manually introduced. The aim of trying build in algorithms from Scipy is to analyze how these properties hold with other methods even though they don't appear in the mathematical formulation. To check this, we collected results from three trials, all with the same algorithm but with different dimensions. In this way we were also able to monitor the increasing number of iterations with increasing dimension of inputs. SLSQP stands for sequential least square quadratic programming and it minimizes a function of several variables with any combination of bounds, equality and inequality constraints. SLSQP is ideal for mathematical problems for which the objective function and the constraints are twice continuously differentiable. Note that the wrapper handles infinite values in bounds by converting them into large floating values. It is a gradient based algorithm which converges to the solution starting from initial values.

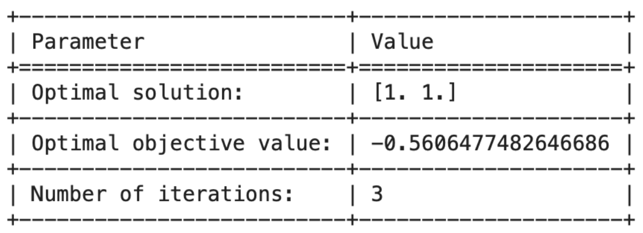
We have defined the specific problems being consistent with the kinds of data used both in the PMM and CPCAM model without specifying a specific betting setting as the focus of this thesis is to solve those models with a diffrent algorithm. However, it is interesting to note that the rules of the settings have unlimited flexibility as long as the fundamental rules of parimutuel models are respected. The kinds of data we used are more consistent with a setting for digital options but the same model can be applied to more complex types of options.

Now we will show the results of the first trials and discuss some interesting findings. Specifically we share the first case where we only optimize for the optimal orders and specify manually the optimal price and the liquidity $(\mathbf{p}, M)$. This is the only case where we don't solve for s but we write it directly as $s_i = \frac{\theta_i}{p_i}$. Note that this was derived by the KT conditions for solving the CPCAM model with Lagrange. We present a few results with the aim of monitoring how optimal values change as the inputs change, in particular the specified values of $\mathbf{p}$. Note that we decide the values of $\mathbf{p}$ making sure their sum is 1, as it is specified in the fundamental conditions derived though Lagrange in the CPCAM formulation. We share results from three trials where we start by specifying the prices equal to each other (0.5, 0.5), then with some variation (0.3, 0.7) and finally with a lot of variation between each entry (0.01, 0.99). We notice that a higher optimal value for our objective function is reached where prices are set equal to each other, with the lowest possible variation between each entry.



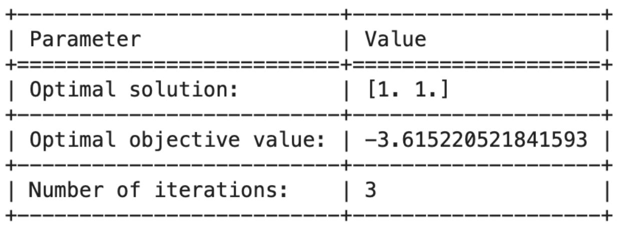
```
+---------------------------+----------------------+
| Parameter                 | Value                |
+===========================+======================+
| Optimal solution:         | [1. 1.]              |
+---------------------------+----------------------+
| Optimal objective value:  | -0.3862943611198906  |
+---------------------------+----------------------+
| Number of iterations:     | 3                    |
+---------------------------+----------------------+
```

Figure 9.1



```
+---------------------------+----------------------+
| Parameter                 | Value                |
+===========================+======================+
| Optimal solution:         | [1. 1.]              |
+---------------------------+----------------------+
| Optimal objective value:  | -0.5606477482646686  |
+---------------------------+----------------------+
| Number of iterations:     | 3                    |
+---------------------------+----------------------+
```

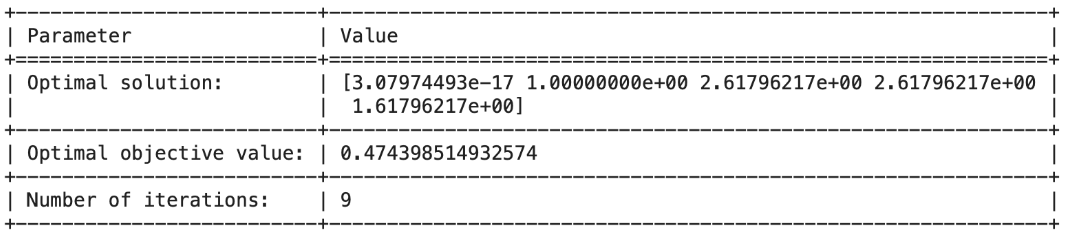
Figure 9.2



```
+--------------------------+--------------------+
| Parameter                | Value              |
+==========================+====================+
| Optimal solution:        | [1. 1.]            |
+--------------------------+--------------------+
| Optimal objective value: | -3.615220521841593 |
+--------------------------+--------------------+
| Number of iterations:    | 3                  |
+--------------------------+--------------------+
```

Figure 9.3

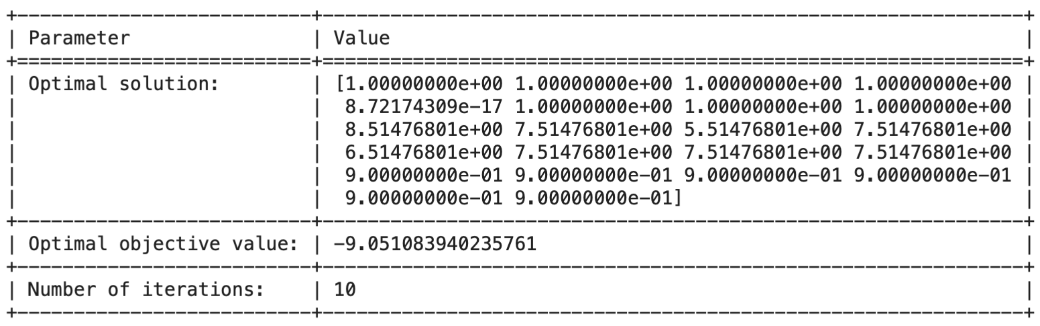
In the following two trials we are solving for $\mathbf{s}, M, x$. The only bounded variable is x as this represents the optimal number of contract accepted and it is set following the preferences declared by the traders ($\mathbf{q}$). In this setting every trader n makes one order j and for everyone the optimal order size is 1 so $\mathbf{q}$ is a vector filled of ones. In the results showed in the following table we have a vector of five entries representing the optimal solutions. The first two entries of the optimal solution vector are the optimal orders accepted ($\mathbf{x}$), then M and the last two are the entries of the vector $\mathbf{s}$. Since every trader asks for one order accepted and the first entry of this vector is out of bound, we interpret it like that order was not accepted while the second order was, since the bounds were clearly specified in the code.



```
+--------------------------+----------------------------------------------------------------+
| Parameter                | Value                                                          |
+==========================+================================================================+
| Optimal solution:        | [3.07974493e-17 1.00000000e+00 2.61796217e+00 2.61796217e+00 |
|                          |  1.61796217e+00]                                               |
+--------------------------+----------------------------------------------------------------+
| Optimal objective value: | 0.474398514932574                                              |
+--------------------------+----------------------------------------------------------------+
| Number of iterations:    | 9                                                              |
+--------------------------+----------------------------------------------------------------+
```

Figure 9.4

In the following table we ran the same experiment with a 7×8 matrix as input for a. Since the columns represent the numbers of orders we count the first eight entries in the optimal solution vector and see that almost all the orders were accepted.



```
+--------------------------+--------------------------------------------------------------+
| Parameter                | Value                                                        |
+==========================+==============================================================+
| Optimal solution:        | [1.00000000e+00 1.00000000e+00 1.00000000e+00 1.00000000e+00 |
|                          |  8.72174309e-17 1.00000000e+00 1.00000000e+00 1.00000000e+00 |
|                          |  8.51476801e+00 7.51476801e+00 5.51476801e+00 7.51476801e+00 |
|                          |  6.51476801e+00 7.51476801e+00 7.51476801e+00 7.51476801e+00 |
|                          |  9.00000000e-01 9.00000000e-01 9.00000000e-01 9.00000000e-01 |
|                          |  9.00000000e-01 9.00000000e-01]                              |
+--------------------------+--------------------------------------------------------------+
| Optimal objective value: | -9.051083940235761                                           |
+--------------------------+--------------------------------------------------------------+
| Number of iterations:    | 10                                                           |
+--------------------------+--------------------------------------------------------------+
```

Figure 9.5

Now we summarize some findings and share result about number of iterations, orders accepted and optimal objective value all varying with the dimension of inputs. In particular we plot on the x-axis the number of rows of the matrix a, as a proxy for dimensions in the setting. The number of columns in these trials will be all 50 percent more than the rows. On the y-axis there will be our dependent variable.

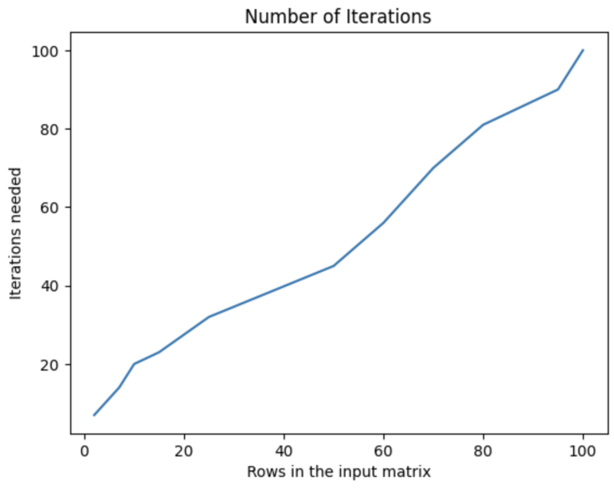


Figure 9.6

The curve represents how the number of iterations change as the dimensions in the setting increase. On the other side Figure 12 below shows hoe the maximization problem returns a lower value for the objective function as the dimensions increase. This is because of how the objective function is, since the value of the liquidity M is subtracted. Finally for the orders accepted in Figure 13, we have not found a particular trend. This is probably due to the fact that entries are set randomly in the inputs.

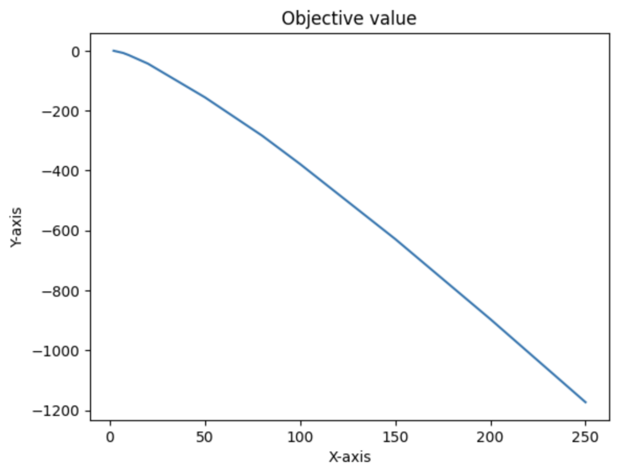


Figure 9.7

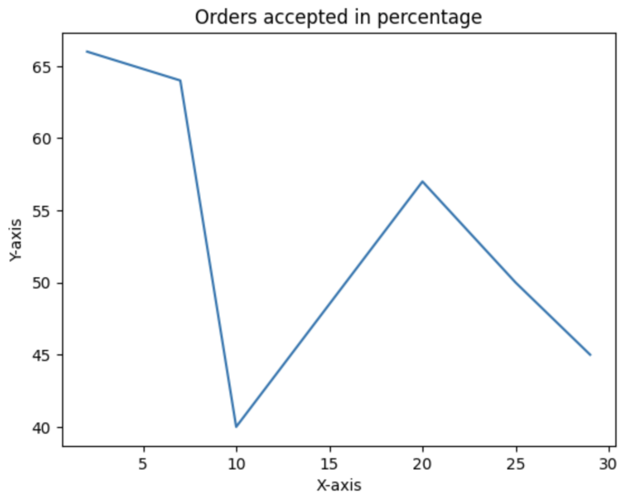


Figure 9.8

Appendix A

Simulation codes

We will share three Python codes, the first one is the one used to optimize only the variable $\mathbf{x}$ while the second and third ones are used to optimize $\mathbf{x}, M, \mathbf{p}$. The last one is used for high dimensions problems.

```
from scipy.optimize import minimize
import numpy as np

# define the objective function
def objective(x, a, theta, pi, M):
    s = np.array([theta[0]/p[0], theta[1]/p[1]]) # calculate s
    obj = np.dot(pi, x) - M + np.sum(theta*np.log(s))
    return -obj # we need to minimize the negative of the objective

# define the constraints
def constraint(x, a, theta, pi, M):
    s = np.array([theta[0]/p[0], theta[1]/p[1]]) # calculate s
    constraints = []
    for i in range(len(a)):
        constraints.append(np.dot(a[i], x) + s[i] - M)
    return constraints

# define the bounds and initial guess
q = np.array([1, 1, 1])
bounds = [(0, q[0]), (0, q[1])]
x0 = np.array([0.5, 0.5])
```

```
# define the other parameters - set M and p
a = np.array([[1, 0], [0, 1]])
theta = np.array([1, 1])
pi = np.array([0.3, 0.7])
M = 2
p = np.array([0.5, 0.5])

# solve the problem
sol = minimize(objective, x0, method='SLSQP', args=(a, theta, pi, M),
    bounds=bounds,
constraints={'type': 'ineq', 'fun': constraint, 'args': (a, theta, pi, M)})
```

```
from scipy.optimize import minimize
import numpy as np

# define parameters
a = np.array([[1, 0], [0, 1]])
theta = np.array([1, 1])
pi = np.array([0.3, 0.7])

lp=len(pi)
lth=len(theta)
la=len(a)

# define the objective function
def objective(x, a, theta, pi):
    obj = np.dot(pi, x[:lp]) - x[lp] + np.sum(theta*(np.log(x[lp+1:lp+lth+1])))
    return -obj


# define the constraints
def constraint(x, a, theta, pi):
    constraints = []
    for i in range(len(a)):
        constraints.append(np.dot(a[i], x[0:lp]) + x[i+lp+1] - x[lp])
    return constraints
```

```
# define the bounds and initial guess
q = np.array([1, 1])
bounds = [(0, q[0]), (0, q[1]),(0,np.inf),(0,np.inf),(0,np.inf)]
x0 = np.array([0.9, 0.9, 0.9, 0.9, 0.9])

# solve the problem
sol = minimize(objective, x0, method='SLSQP', args=(a, theta, pi), bounds=bounds,
constraints={'type': 'eq', 'fun': constraint, 'args' : (a, theta, pi)})
```

```
from scipy.optimize import minimize
import numpy as np

# Define the dimensions
m = 95
n = 143

# Generate random inputs
a = np.random.randint(2, size=(m, n))
theta = np.ones(m)
pi = np.random.rand(n)
q = np.ones(n)

# Define the lengths
lp = len(pi)
lth = len(theta)
la = a.shape[0]

# Define the objective function
def objective(x, a, theta, pi):
    obj = np.dot(pi, x[:lp]) - x[lp] + np.sum(theta * np.log(x[lp + 1 : lp + lth
        + 1]))
    return -obj

# Define the constraints
def constraint(x, a, theta, pi):
    constraints = []
    for i in range(la):
        constraints.append(np.dot(a[i], x[0 : lp]) + x[i + lp + 1] - x[lp])
    return constraints

# Define the bounds and initial guess
bounds = [(0, q[i]) for i in range(lp)] + [(0, np.inf)] * (la + lth)
x0 = np.array([0.9] * lp + [0.9] * (la + lth))

# Solve the problem
sol = minimize(
    objective,
    x0,
    method="SLSQP",
    args=(a, theta, pi),
    bounds=bounds,
    constraints={"type": "eq", "fun": constraint, "args": (a, theta, pi)},
)
```

Bibliography

[Agrawal et al., 2009] Agrawal, S., Delage, E., and Peters, M. (2009). A unified framework for dynamic pari-mutuel information market design.

[Bossaerts et al., 2002] Bossaerts, P., Fine, L., and Ledyard, J. (2002). Inducing liquidity in thin financial markets through combined-value trading mechanisms. *European Economic Review*, 46:1671—-1695.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). Convex optimization.

[Hanson, 2002] Hanson, R. (2002). Logarithmic market scoring rules for modular combinatorial information aggregation.

[Lange and Economides, 2003] Lange, J. and Economides, N. (2003). A parimutuel market microstructure for contingent claims. *European Financial Management*, 11(1):25–49.

[Pennock, 2004] Pennock, D. (2004). A dynamic pari-mutuel market for hedging, wagering, and information aggregation.

[Peters et al., 2006] Peters, M., So, A. M., and Ye, Y. (2006). A convex parimutuel formulation for contingent claim markets. 1(1):1–10.

[Simon and Blume, 1987] Simon and Blume (1987). Mathematics for economists. 14(3):342–351.