



ugr | Universidad
de **Granada**

TRABAJO FIN DE GRADO
INGENIERÍA INFORMÁTICA Y MATEMÁTICAS

**Teoremas de la alternativa, optimización convexa,
valoración de activos financieros
y
procesamiento de nubes de puntos generadas por
escáner láser.**

Autor
Pedro Manuel Flores Crespo

Directores
Manuel Ruiz Galán
Juan Carlos Torres Cantero



Facultad de Ciencias

—
Granada, junio de 2020

Índice general

1. Resumen y palabras clave	5
2. Resumen extendido y palabras clave en inglés	7
3. Introducción	11
4. Objetivos	15
5. Desarrollo	17
I Teoremas de la alternativa, optimización convexa y va-	
loración de activos financieros	19
6. Teoremas de la alternativa	21
6.1. Teorema de Mazur-Orlicz-König	21
6.2. Teorema de la alternativa de Gordan. Reformulaciones.	28
7. Alternativa y minimax	37
7.1. Una desigualdad minimax	37
7.2. Separación de convexos	40
8. Optimización	47
8.1. Lema de Farkas	47
8.2. Dualidad en programación lineal	49
8.3. Optimización con restricciones: Fritz John y Karush-Kuhn- Tucker	51
9. Valoración de activos financieros	57
9.1. Preliminares financieros	57
9.2. Primer teorema fundamental	60
9.3. Aplicación al modelo binomial	67

II Procesamiento de nubes de puntos generadas por escáner láser	75
10.Iterative Closest Point (ICP)	77
10.1. Cuaternios	77
10.1.1. Suma y producto	78
10.1.2. Conjugado, norma e inverso	79
10.1.3. Cuaternios y rotaciones	80
10.2. Cuaternios y optimización	83
10.3. Algoritmo ICP	86
10.4. Ejemplos prácticos	91
11.Cambios en el método: uso de normales	95
11.1. Variación de la normal	95
11.2. Pruebas	97
11.2.1. De puntos clave a todo el modelo	97
11.2.2. De puntos clave a puntos clave	100
11.3. Torre de las gallinas	102
11.4. Sensibilidad y uso en prealineado	106
11.5. Conclusiones	109
12.RANSAC: algoritmo de consenso	111
12.1. Algoritmo	111
12.2. Número de iteraciones	112
12.3. Ejemplos sencillos	114
12.3.1. Pie	115
12.3.2. Torre	115
12.4. Cálculo de los puntos de intersección	116
12.5. Ejemplo despacho	119
12.6. Ejemplo específico	124
13.Banco de pruebas	127
13.1. Estructura básica de la aplicación	127
13.1.1. Clase <i>_basic_object3D</i>	129
13.2. Picking	131
13.3. Selección mediante rectángulo	132
13.4. Código, instalación y uso	134
14.Conclusiones y vías futuras	135

Resumen y palabras clave

En el ámbito de las matemáticas, el trabajo fin de grado que presentamos supone la incursión a un campo dentro de la optimización, los teoremas de la alternativa, y sus aplicaciones, principalmente a la propia optimización y finanzas. El trabajo comienza con el teorema de Mazur-Orlicz-König, versión del conocido teorema de Hahn-Banach. Posteriormente, estudiamos el teorema de la alternativa de Gordan, esencial para el resto de resultados. Su primera aplicación se da en la teoría minimax, que nos conduce a resultados clásicos sobre separación de convexos. A continuación, deduciremos otro teorema de la alternativa, el de Farkas, y lo aplicaremos a la programación lineal. Para demostrar los resultados sobre optimización, volvemos al teorema de Gordan, que nos proporcionara los teoremas de Fritz John y Karush-Kuhn-Tucker. Finalmente, nos introducimos en el mundo de las matemáticas financieras obteniendo, gracias al teorema de separación, el primer teorema de asignación de precios para valorar opciones europeas. Concluimos realizando simulaciones de su valor en diferentes casos.

Por su parte, dentro del campo de la informática, se ha realizado un estudio acerca del procesamiento de nubes de puntos. En general, se tienen varias nubes que corresponden a un mismo objeto pero que están tomadas desde distintos puntos de referencia. Esto hace necesario un tratamiento de los conjuntos para conseguir que se encuentren en el mismo sistema de coordenadas y tener así un modelo digital del objeto. Este problema se denomina alineado y para resolverlo se aborda el estudio de dos algoritmos diferentes: ICP y RANSAC. El primero de ellos es un algoritmo genérico que veremos que tiene una gran desventaja, el tiempo de ejecución del mismo. Por ello, intentaremos disminuir todo lo posible este tiempo mediante el uso de descriptores para detectar puntos significativos del modelo y reducir, de ese modo, el conjunto de puntos a los que es necesario aplicar el algoritmo. Por su parte, el algoritmo RANSAC es genérico y sirve para estimar parámetros de un modelo matemático. En nuestro caso, lo usamos para la detección de planos que nos aportaran un número reducido de puntos clave a los que aplicar el proceso de alineado.

Palabras clave: teoremas de la alternativa, teorema de Hahn-Banach, minimax, optimización, matemáticas financieras, alineado, cuaternios, ICP, RANSAC.

Resumen extendido y palabras clave en inglés

In Mathematics, the end of degree project that we present is an incursion into a field included in optimization, the theorems of the alternative, and their applications, mainly to the own optimization and finances.

Many of the theorems of the alternative are just reformulations of convex separation's theorems in certain contexts, so we start this memory with a study on Hahn-Banach's theorem. There are several equivalent versions of this result, collected for example in [7]. That is the reason why many authors talk about Hahn-Banach's theorems. We will focus on one of them, Mazur-Orlicz-König's theorem. Despite its geometrical charge, it is mainly an algebraic result. We will provide a proof of that theorem – the results in this work are self-contained – and we will use it to give a convex version of Gordan's theorem of alternative. Actually, it is an equivalent result due to S. Simons. This version is more general than the original Gordan's result and it will be extremely useful in the next important results.

Then, we will apply the convex Gordan's theorem of the alternative in the minimax theory. A minimax inequality guarantees, under certain hypothesis, that in a two variables function we can substitute $\inf\sup$ for $\sup\inf$. The power of minimax inequalities is clear when we use it – in one that we will deduce from the convex Gordan's theorem – to give classical convex separation's results.

Next, we will deduce from one separation's theorem the Farkas's lemma, which is one of the most known theorem of the alternative. This will allow us to prove, almost immediately, one of the key points of linear programming: the duality theorem. Considering more general optimization theorems, those in which the objective function and the inequalities constraints are differentiables, we will establish, using Gordan's theorem of the alternative, the theorems of Fritz John and Karush-Kuhn-Tucker.

We will conclude with a little foray into the field of financial mathematics. After introducing some main concepts, like a derivative security; we will prove, as a consequence of one of the convex separation's theorem, the result known as “First Fundamental Theorem of Asset Pricing”. It can be used in the pricing of European's options in the binomial model. A European's option is a contract that gives the

owner the right, but not the obligation, to buy or sell certain asset. This situation is a clear advantage for the buyer because he can earn money without taking any risk, breaking the no-arbitrage principle. We solve this inconvenient imposing a price to get the option as we mentioned before. Finally, we have programmed using *Sagemath 2.8* some examples in order to know how their price varies according to its parameters. For instance, we can study its price depending on considerations whether the asset's price goes up or down or even the value of the money in the future, choosing the most profitable option. During this process, we have made some considerations about our market model. We can remark the positivity of prices which guarantees that all the prices are positive, divisibility which refers to the fact that one can hold a fraction of an asset or liquidity, meaning that any asset can be bought or sell on demand at the market. Some assumptions, like liquidity, are more mathematical than practical – in real markets there are bounds on the volume of trading.

Related to Computer Science, this project consists in a study of 3D digitalization process, which is considered part of Computer Graphics. First, in order to have a digital copy of some desired object, we must obtain some point clouds representing it. They are like three-dimesional photographs. Using this analogy, one photograph does not cover the whole object's surface. That is the reason why we need a few clouds for each one. Unfortunately, the point clouds are not in the same reference system because we must move either the object or the scanner to get all the information. We need to “modify” them, so they match and finally obtain a 3D copy of the object. This modification or treatment have three different processes: alignment, meshing and triangulation. The first one refers to the fact that we have mentioned before, the point clouds are not in the same reference system. This step solves this problem. To accomplish this target properly, our point clouds need some overlapping to know how to place them exactly. This situation leads us to the meshing step. Due to overlapping, once we have the shots expressed in the same coordinates, some parts have a higher level of points density. With meshing, we refer to omit “repeated” points in these zones that has been registered several times in order to get a good alignment. At this moment, we have the object represent by the points. The goal of triangulating is just getting the surface represented by the points. The three steps that we have just indicated, are very interesting and they deserve a complete study to solve them. However, in this project we will focus on the clouds' alignment, also called data registration.

The first algorithm we will study is the Iterative Closest Point (ICP). It is considered a generic algorithm because it can be used to align two points clouds independently of the characteristics of the object. We will talk about this topic lately in this work. However, it is important to highlight that there is not a suitable algorithm to align any shots from all type of objects. This section starts with a brief introduction to quaternions. Quaternios are similar to an extension of complex numbers. In our case, we will use them to represent rotations in three dimensions (a rotation and a translation are needed to align the clouds). There are other ways to represent them such as 3×3 arrays or Euler angles but quaternios have some advantages. After that, we will discuss how to obtain the matrix and the translation commented before by optimizing a distance function. This will give us the steps we must follow in the algorithm. We will also prove the local convergence of this

method. The fact that it converges locally forces us to have a pre-alignmet step. With pre-align we refer to place both point clouds close enough to ensure that the result is correct. Another disadvantage of ICP is the quadratic complexity in time which make it untreatable with large point clouds.

Our next goal will be to reduce the amount of time spent by the algorithm to complete. We will achieve it by reducing the elements in the clouds and just using the most relevant points of the model in our method. To detect these points, called key points, we will use the variation of the normal vector to distinguish between planar and sharp areas. It is possible thanks to the scanner used to get the shoots. The output format gives the points in a matrix so we can obtain neighborhood relations. This decrement of the points' quantity will cause a significant reduction in the execution time. Finally, we will study how affects the pre-alignment in the provided results.

The second algorithm we will discuss is the Random Sample Consensus (RANSAC). It is used to get the parameters that fit a mathematical model given by some observations. It is considered a robust algorithm even in the presence of outliers and is based on a hypothesis-test paradigm. In our case, we will use it to detect planes in our model. The steps to obtain the planes are very simple: select three random points that will represent a plane, count how many points fits that plane (with some tolerance), repeat this process N times and take the one that has the maximum number of points. We will give a formula to determine the number N depending on the probability to be an outlier, the desired probability to find the model and the number of parameters needed to define it. Once we have the planes, we will calculate their intersections in order to have significant corners. Then, we will use the angles between the planes to assign a descriptor to the intersection points and determine which of them match in different points clouds.

To test the proposed methods, we have different models: a clay feet property of the Faculty of Fine Arts of Granada, a part of *La torre de las Gallinas* in the Alhambra and an office. These models have been taken with a Faro Focus 130 laser scanner. It is has also been necessary to develop an application in which we can code and observe the results of the different algorithms. We will cover this part on the last chapter. This program is written in *C++*, we use *OpenGL* as the graphic library and *Qt* for the user interface. During its develeptment, we have applied several concepts related to Computer Graphics. For instance, methods to select or delete a bunch of points or pick one of them. We have also been concerned about the application performance, especially in our case where we use models with a lot of vertices.

Keywords: theorems of the alternative, Hahn-Banach's theorem, minimax, optimization, financial mathematics, data registration, quaternions, ICP, RANSAC.

Introducción

El estudio de los teoremas de la alternativa hunde sus raíces en el teorema de separación de convexos de Hahn-Banach. Es más, algunas de sus versiones más conocidas, como el teorema de la alternativa de Gordan o el lema de Farkas, son precursoras de ese resultado fundamental del análisis funcional o la optimización. Y es precisamente ahí, en un contexto de optimización, donde surgen hace casi siglo y medio. Desde entonces han aparecido en gran cantidad, vinculados a problemas de optimización convexa en muchas ocasiones y mediante el uso de técnicas de separación. Además, su aplicabilidad no se ha circunscristo exclusivamente al campo de la optimización sino que ha transcendido dicha área: análisis convexo, análisis funcional, problemas de equilibrio ...

En esta memoria se aborda el estudio de los dos resultados de la alternativa mencionados, el teorema de Gordan y el lema de Farkas, dando incluso una versión más general del primero. Para establecerlos usamos una versión muy simple del teorema de Hahn-Banach y una mejora del mismo. Además, se aplican para establecer una desigualdad minimax que deriva, en particular, en una serie de teoremas de separación de convexos. También se ilustra su aplicabilidad en establecer resultados centrales en la optimización, como es el teorema de dualidad en programación lineal o los teoremas de Fritz John y Karush-Kuhn-Tucker en un contexto diferenciable. Finalmente, dedicamos todo un capítulo de la memoria a usar los teoremas de la alternativa, en una de sus formas equivalentes, para demostrar un resultado importante en el ámbito de las matemáticas financieras, el primer teorema fundamental de valoración de activos financieros. Ello requiere un bagaje previo – conceptos y resultados – que también se recoge en la memoria. Dicho teorema se aplica al caso de ciertos derivados muy populares, las opciones europeas, bajo un modelo binomial y se presentan algunas simulaciones numéricas realizadas con *SageMath* en su versión 2.8.

Centrándonos en la parte informática, el trabajo que se expone a continuación puede considerarse como una introducción al mundo del digitalizado 3D. Es una técnica que se engloba dentro del campo de la Informática Gráfica y que se aplica a una gran cantidad de sectores: diseño industrial, arte, cine, medicina, etc. Por ejemplo, en el mundo artístico se emplea para la obtención de un modelo digital de la obra con el fin de tener una copia digital de la misma u obtener más información que puede ser de utilidad, como en los procesos de restauración, sin necesidad de tener el original. También podemos poner un ejemplo de su uso en diseño industrial donde el modelo obtenido se puede utilizar para realizar simulaciones con el

objetivo de detectar las partes más susceptibles a sufrir fallos.

La obtención de un modelo detallado y completo es un proceso complicado y ampliamente estudiado. En primer lugar, es importante el escáner que utilicemos para realizar dicho proceso. En la actualidad existen una gran cantidad de modelos que utilizan diferentes tecnologías para obtener cada una de las tomas necesarias aunque la mayoría de ellos suelen darnos como resultado una nube de puntos que define la muestra. Otras técnicas como la fotogrametría también tienen como objetivo la obtención de un modelo 3D pero a partir de fotografías. Nosotros estudiaremos mecanismos basados en nubes de puntos. En cuanto a las técnicas utilizadas para la obtención estas nubes de puntos hay dos tipos: por contacto o sin contacto. La más utilizada es la segunda de ellas que a su vez puede utilizar una tecnología láser o mediante luz estructurada. Entre otros aspectos a tener en cuenta encontramos la resolución del escáner, su precisión, las propiedades del objeto, etc. No es el objetivo de este trabajo ver las tecnologías disponibles en el digitalizado ni el propio proceso y sus consecuencias en el resultado final obtenido pero es importante conocer las alternativas disponibles. Si hay que tener en cuenta que algunos escáneres nos pueden aportar información adicional acerca de los puntos lo que puede ser de utilidad posteriormente. Una vez tenemos las nubes de punto, podemos distinguir tres etapas hasta llegar al modelo digital:

- Alinear: cuando se toman las muestras del objeto a digitalizar, es necesario mover el escáner con el que se realizan las medidas o el propio objeto. Esto conlleva que diferentes tomas van a estar en diferentes sistemas de coordenadas. Esto se podría solucionar, por ejemplo, mediante un sistema GPS que guardara las posiciones para posteriormente realizar las transformaciones correspondientes en las muestras. El error de GPS puede ser del orden de varios metros y no funciona bien en interiores. Por otra parte hay otro tipo de mecanismos que puede merecer la pena comentar: utilizar dianas en la escena; sensores iniciales en el escáner. En cualquier caso estos mecanismos no están siempre disponibles. Por ello, esta etapa, tiene como objetivo que todas las tomas estén respecto al mismo sistema de coordenadas. Otro aspecto a destacar es si estas tomas son rígidas o no, es decir, si no hay deformaciones en las mismas. Nosotros en este trabajo nos centraremos en el caso rígido.
- Fusionar: finalizada la etapa anterior debemos hacer que todas las nubes de puntos que tenemos alineadas se transformen en una sola. Más concretamente, durante la obtención de las muestras es necesario que estas se solapen, como veremos posteriormente, lo que hace que haya zonas con gran cantidad de puntos, muchos de ellos “repetidos”. De este modo, debemos detectar dichos puntos para que la densidad en todo el modelo sea uniforme. También hay que tener en cuenta que es posible que haya zonas que por falta de previsión o por imposibilidad física no se hayan podido obtener una muestra. En esta etapa también se deberían arreglar dichas faltas de información.
- Triangular: hasta ahora hemos trabajado con nubes de puntos pero el modelo es una superficie. Con esta etapa final se pretende obtener los triángulos que definen dicha superficie y así completar el proceso de digitalizado. Una alternativa sería obtener una función implícita que defina la superficie.

Las etapas que acabamos de mencionar no deben darse obligatoriamente en ese orden. Podríamos empezar calculando los triángulos que define cada nube de

puntos y posteriormente alinear y fusionar. Sin embargo, el orden descrito es el más habitual. Durante el proceso también se suele tener en cuenta la densidad de puntos, la posible existencia de ruido, de valores atípicos o del error que tiene el propio escáner durante la toma de medidas aunque no serán aspectos a los que nosotros les demos demasiada importancia.

Una idea importante a destacar es que en el digitalizado 3D no hay un algoritmo que sea capaz de reconstruir todo tipo de objetos. En este trabajo se estudiarán algunos algoritmos genéricos pero en la práctica existen métodos específicos para entornos urbanos, arquitectónicos, modelos curvilíneos, suaves o con aristas, etc. Más concretamente, nos centraremos en la primera de las etapas del proceso, el alineado. Dentro de este campo veremos dos de los algoritmos más usados: ICP y RANSAC. El primero de ellos es un algoritmo que se usa para cualquier modelo pero que veremos que tiene una gran desventaja: el tiempo de ejecución del mismo. Por ello, intentaremos disminuir todo lo posible este tiempo mediante el uso de descriptores para detectar puntos significativos del modelo y reducir de ese modo el conjunto de puntos a los que es necesario aplicar el algoritmo. Por su parte, el algoritmo RANSAC es más específico que el anterior y sirve para estimar parámetros de un modelo matemático. En nuestro caso, lo usaremos para la detección de planos que nos aportaran un número reducido de puntos clave a los que aplicar el proceso de alineado. Estos algoritmos se han programado dentro un banco de pruebas que nos permite realizar los algoritmos paso a paso con el fin de poder observar mejor el comportamiento de los mismos.

En definitiva, por un lado, en esta memoria se plasma tanto el carácter convexo-funcional de los teoremas de la alternativa como su aplicabilidad a campos tan diversos como la optimización, el análisis convexo y las matemáticas financieras. Por la parte informática, se ha realizado un análisis de dos procedimientos habituales para la alineación de dos nubes de puntos, proceso que se engloba dentro de la digitalización 3D. Señalamos finalmente que las referencias usadas en la elaboración de esta memoria aparecen recogidas en el capítulo de Bibliografía. No obstante, los textos de [1], [2] y [8] han sido los esenciales para la parte de matemáticas financieras, mientras que [13], [10], [11] y [14] han sido de gran utilidad en el tema de digitalizado.

Objetivos

Los objetivos inicialmente previstos en la propuesta de TFG en matemáticas fueron:

- Realizar una recopilación de algunos teoremas de la alternativa.
- Teorema de dualidad en programación lineal.
- Teoremas de Karush-Kuhn-Tucker y Fritz John para programación convexa.
- Aplicación a finanzas: teorema fundamental de valoración de activos financieros en mercados finitos.

Sin embargo, nuestro tratamiento final ha sido algo más ambicioso, pues hemos incluido todo un capítulo de aplicaciones de los teoremas de la alternativa a la teoría minimax y a la separación de convexos. Además, en lugar de considerar los teoremas de Karush-Kuhn-Tucker y Fritz John en un ambiente convexo, los hemos establecido en un contexto no lineal y diferenciable. La idea que nos ha llevado a ello ha sido aumentar el número y tipología de aplicaciones de los teoremas de la alternativa, mostrando su versatilidad en diversos campos.

Dentro de la parte informática se pretendía:

- Realizar una breve incursión en el mundo de la digitalización 3D, conociendo a rasgos generales el proceso empleado en la obtención de un modelo.
- Estudio de la información geométrica del modelo obtenido.
- Uso de dicha información para proponer una optimización de los algoritmos empleados en el proceso.
- Realización de un programa para evaluar los resultados obtenidos.

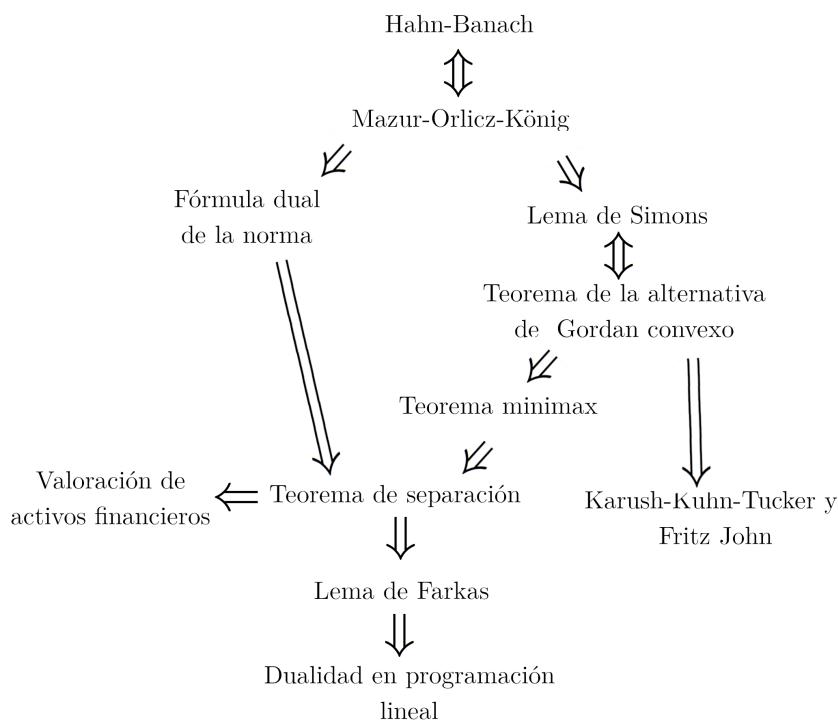
Destacamos que también se han alcanzado los objetivos propuestos en este caso. Inicialmente, la propuesta de TFG no contenía ninguna especificación acerca de los algoritmos o técnicas a usar debido a la amplia cantidad de métodos existentes así como la complejidad del proceso completo de digitalizado. Así, el primer punto ha servido como fase previa para el conocimiento del tema a trabajar para poder concretar un objetivo. Tras ello, se decidió centrar el estudio en la etapa de alineado tal, como ya se ha comentado anteriormente, y orientar de ese modo, los puntos segundo y tercero en ese ámbito. Finalmente, el cuarto punto se ha enfocado como un banco de pruebas para comprobar los resultados obtenidos en los puntos anteriores y no como una aplicación para uso de terceros. Esto ha conllevado el uso de

nuevas herramientas de desarrollo así como profundizar en conceptos clave de la Informática Gráfica. De este modo, se han alcanzado estos objetivos de una manera colateral.

Desarrollo

Matemáticas

El proceso seguido en el desarrollo del TFG ha sido, por un lado, recopilar material sobre el tema y analizarlo, y por otro, darle estructura totalmente auto-contenida, elaborando los diversos contenidos de forma jerarquizada en el sentido de que se deducen de los anteriores. A modo de esquema, los resultados se han estructurado atendiendo al siguiente esquema donde además se recoge la relación entre ellos:



Como puede observarse, las técnicas son de carácter convexo y analítico funcional.

Informática

Tras la concreción del tema a resolver dentro del ámbito del digitalizado 3D, se decidió empezar a trabajar en uno de los algoritmos más conocidos para ello, el *Iterative Closest Point* (ICP). También, al comenzar el trabajo, se decidió la manera de realizar las pruebas necesarias. Se optó por la realización de un banco de pruebas utilizando *OpenGL* (versión 4.6) como biblioteca gráfica, *C++* como lenguaje de programación así como *Qt* (versión 5.9.1) para la interfaz gráfica. El IDE, por lo tanto, ha sido *Qt Creator 4.3.1*. Este banco de pruebas debe proporcionar herramientas básicas para la manipulación de las nubes de puntos con las que estamos trabajando así como otras funcionalidades necesarias para el desarrollo de los distintos algoritmos. En este caso se hizo necesaria solventar problemas como: abrir y guardar archivos, manipulación de la matriz de vista, selección de un conjunto de puntos y borrado de los mismos, selección de puntos aislados, etc.

Volviendo al primer algoritmo a estudiar, se comprobó que era necesario un conocimiento previo acerca de los cuaternios, tanto para la demostración de la convergencia del método como para su implementación. Una vez completado, se procedió a la elaboración de distintas pruebas. Destacar que también se hizo necesario el cálculo de valores propios de una matriz por lo que se optó por usar la biblioteca *Eigen* (en su versión 3.3.7). Posteriormente, se planteó la posibilidad de incluir mejoras al método usando la variación de las normales para obtener información acerca de la geometría del modelo. Por ello, fue necesaria la inclusión de esta opción dentro del banco de pruebas así como un algoritmo de simplificado de la nube de puntos tal y como se explicará en el desarrollo del trabajo.

Llegados a este punto, se planteó la posibilidad de incluir otra serie de mejoras al procedimiento o de estudiar otros algoritmos diferentes. Se siguió la segunda opción con el fin de tener una visión general más general del problema. Por ello, en último lugar aparece un estudio acerca del algoritmo *Random Sample Consensus* (RANSAC), partiendo de la explicación de los pasos del propio algoritmo pasando por razonamientos probabilísticos para asegurar la obtención de buenos resultados y finalmente la realización de pruebas del mismo.

Parte I

Teoremas de la alternativa, optimización convexa y valoración de activos financieros

Teoremas de la alternativa

Los teoremas de la alternativa constituyen una potente herramienta en optimización. A pesar de tener un claro carácter convexo, se aplican incluso a problemas no convexos, tal y como se comprobará a lo largo de esta memoria. Nuestro punto de partida es el resultado del análisis convexo, más importante, el teorema de Hahn-Banach. Es más, daremos una versión equivalente debida a H. König, conocida como el teorema de Mazur-Orlicz-König. Como consecuencia, obtendremos el teorema de la alternativa de Gordan, tanto en su versión clásica como una más general.

6.1. Teorema de Mazur-Orlicz-König

El objetivo principal de esta sección es demostrar una versión equivalente no muy conocida del clásico teorema de Hanh-Banach, el teorema de Mazur-Orlicz-König. Iremos de una versión básica y algebraica del teorema de Hahn-Banach al teorema de Mazur-Orlicz-König siguiendo como aparece en el texto de S. Simons [8].

En primer lugar vamos a recordar la definición de funcional sublineal sobre un espacio vectorial V . Notar que todos los espacios vectoriales que vamos a usar son reales. Del mismo modo, los conjuntos que usaremos asumiremos que son no vacíos.

Definición 6.1. *Sea V un espacio vectorial. Decimos que el $P : V \rightarrow \mathbb{R}$ es sublineal si cumple las siguientes condiciones:*

- *P es subaditiva: $x_1, x_2 \in V \implies P(x_1 + x_2) \leq P(x_1) + P(x_2)$.*
- *P es positivamente homogénea: $x \in V$ y $\lambda > 0 \implies P(\lambda x) = \lambda P(x)$.*

Como consecuencia, podemos afirmar que $P(0) = 0$. En efecto:

$$P(0) = P(2 \times 0) = 2 \times P(0) \implies P(0) = 0.$$

Por ejemplo, toda norma o incluso toda seminorma sobre V es un funcional sublineal. Así, dados $a, b \in \mathbb{R}$ con $a < b$ tenemos que $P : H^1(a, b) \rightarrow \mathbb{R}$ definida como

$$P(x) = \|x'\|_{L^2(a,b)}$$

es una seminorma y por ello sublineal. También, si $V = \mathbb{R}$ y definimos la parte positiva

$$P(x) = [x]^+ = \max\{0, x\}, \quad \forall x \in \mathbb{R}$$

obtenemos un funcional sublineal sobre \mathbb{R} .

El lema que exponemos a continuación, y que generalizaremos posteriormente en el lema 6.2, nos servirá para demostrar el teorema de Hanh-Banach.

Lema 6.1. *Sea V un espacio vectorial y $P : V \rightarrow \mathbb{R}$ un funcional sublineal. Fijamos un elemento $y \in V$ y para todo $x \in V$ tomamos*

$$P_y(x) := \inf_{\lambda > 0} [P(x + \lambda y) - \lambda P(y)].$$

Entonces, $P_y(V) \subset \mathbb{R}$, $P_y : V \rightarrow \mathbb{R}$ es sublineal, $P_y \leq P$ y además $P_y(-y) \leq -P(y)$.

Demostración. Fijamos $y \in V$. Sea $x \in V$ y $\lambda > 0$. Como P es sublineal tenemos:

$$\lambda P(y) = P(\lambda y) = P(\lambda y + x - x) \leq P(x + \lambda y) + P(-x).$$

Por lo tanto, se obtiene que $P(x + \lambda y) - \lambda P(y) \geq -P(-x)$. Tomando el ínfimo sobre $\lambda > 0$ llegamos a $P_y(x) \geq -P(-x) > -\infty$. Por consiguiente, $P_y(V) \subset \mathbb{R}$, esto es $P_y : V \rightarrow \mathbb{R}$.

Probaremos ahora que P_y es sublineal. Empezamos viendo la subaditividad. Tomamos $x_1, x_2 \in V$ y sean $\lambda_1, \lambda_2 > 0$ arbitrarios. Entonces, la definición de P_y da

$$\begin{aligned} P(x_1 + \lambda_1 y) - \lambda_1 P(y) + P(x_2 + \lambda_2 y) - \lambda_2 P(y) \\ \geq P(x_1 + x_2 + (\lambda_1 + \lambda_2)y) - (\lambda_1 + \lambda_2)P(y) \\ \geq P_y(x_1 + x_2). \end{aligned}$$

Tomando ínfimos sobre λ_1 y λ_2 , $P_y(x_1) + P_y(x_2) \geq P_y(x_1 + x_2)$. Así, P_y es subaditiva. Para comprobar que es positivamente homogénea tomamos $x \in V$ y $\mu > 0$. Entonces:

$$\begin{aligned} P_y(\mu x) &= \inf_{\lambda > 0} [P(\mu x + \lambda y) - \lambda P(y)] \\ &= \mu \inf_{\lambda > 0} [P(x + (\lambda/\mu)y) - (\lambda/\mu)P(y)] \\ &= \mu \inf_{v > 0} [P(x + vy) - vP(y)] \\ &= \mu P_y(x). \end{aligned}$$

Obtenemos que P_y es positivamente homogénea y como consecuencia sublineal.

Para demostrar que $P_y \leq P$, sea $x \in V$ y tomando $\lambda = 1$ en la definición de P_y ,

$$P_y(x) \leq P(x + y) - P(y) \leq P(x) + P(y) - P(y) = P(x).$$

Como $x \in V$ es arbitrario, entonces $P_y \leq P$. Finalmente, razonando de manera similar,

$$P_y(-y) \leq P(-y + y) - P(y) = -P(y).$$

■

El teorema de Hahn-Banach que hemos mencionado antes (básico y algebraico) se enuncia en estos términos:

Teorema 6.1 (Hahn-Banach). *Sea V un espacio vectorial y $P : V \rightarrow \mathbb{R}$ un funcional sublineal. Entonces existe un funcional lineal $L : V \rightarrow \mathbb{R}$ tal que $L \leq P$.*

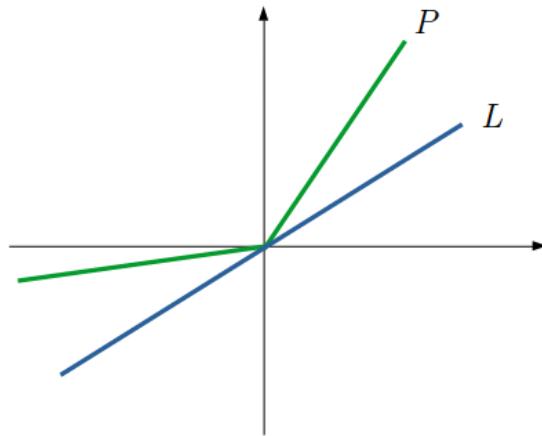


Figura 6.1: Teorema de Hahn-Banach.

Demuestra. Sea $SUB = \{Q : V \rightarrow \mathbb{R} : Q \leq P\}$, es decir, el conjunto no vacío de funcionales sublineales sobre V que minoren a P . Nuestro propósito es emplear el lema de Zorn con objeto de probar que tiene un elemento mínimo y tal elemento será el funcional L que buscamos. Para ello, dados $T_1, T_2 \in SUB$ consideramos la relación de orden usual, es decir:

$$T_1 \leq T_2 \iff T_1(x) \leq T_2(x) \quad \forall x \in V.$$

Primero probaremos que todo subconjunto \mathcal{T} totalmente ordenado de SUB tiene una cota inferior en SUB . Definimos $Q(x) := \inf\{T(x) : T \in \mathcal{T}\}$ y queremos ver que $Q(x) \in \mathbb{R}$. Si $x \in V$ y $T \in \mathcal{T}$, como T es subaditiva obtenemos la siguiente relación:

$$0 = T(0) = T(x - x) \leq T(x) + T(-x) \implies T(x) \geq -T(-x) \quad (6.1)$$

Por otro lado:

$$T \in \mathcal{Q} \implies T(x) \leq P(x) \implies -T(x) \geq -P(x) \quad (6.2)$$

Usando (6.1), (6.2) y tomando ínfimo sobre T llegamos a $Q(x) \geq -P(x) > -\infty$. Por lo tanto $Q : V \rightarrow \mathbb{R}$.

Ahora probaremos que Q es subaditiva. Para ello, tomamos $x_1, x_2 \in V$. Sean $T_1, T_2 \in \mathcal{T}$ arbitrarios. Si $T_1 \geq T_2$ (el caso de $T_2 \geq T_1$ es análogo.):

$$T_1(x_1) + T_2(x_2) \geq T_2(x_1) + T_2(x_2) \geq T_2(x_1 + x_2) \geq Q(x_1 + x_2).$$

Concluimos (en ambos casos) que $T_1(x_1) + T_2(x_2) \geq Q(x_1 + x_2)$. Tomando ínfimo en T_1 y T_2 obtenemos que $Q(x_1) + Q(x_2) \geq Q(x_1 + x_2)$. Así, Q es sublineal. Que sea positivamente homogénea es consecuencia de que T también lo es. Dado $\mu > 0$:

$$\begin{aligned} Q(\mu x) &= \inf\{T(\mu x) : T \in \mathcal{T}\} \\ &= \inf\{\mu T(x) : T \in \mathcal{T}\} \\ &= \mu \inf\{T(x) : T \in \mathcal{T}\} \\ &= \mu Q(x). \end{aligned}$$

De este modo, Q es sublineal y como es claro que $Q \leq P \implies Q \in \text{SUB}$. Así, es directo que Q es el elemento minimal de \mathcal{T} en SUB .

El lema de Zorn nos proporciona entonces un elemento minimal de SUB que llamaremos L . Vamos a comprobar que L es lineal y, por tanto, es el funcional buscado. Tomamos ahora $y \in V$. Con la notación del lema anterior, $L_y : V \rightarrow \mathbb{R}$ es sublineal, $L_y \leq L$ (como consecuencia $L_y \in \text{SUB}$) y $L_y(-y) \leq L(-y)$. De hecho, como L es minimal en SUB , $L_y = L$ y por ello $L(-y) \leq L(-y)$. Por otro lado, como L es subaditiva, $L(-y) \geq -L(y)$. Combinando ambas desigualdades, $L(-y) = -L(y)$. Tomamos $x \in V$ y $\lambda < 0$, usando la igualdad anterior llegamos a:

$$L(\lambda x) = L(-(-\lambda)x) = -L(-\lambda x) = -(-\lambda)L(x) = \lambda L(x)$$

obteniendo que L es homogénea. Si $x_1, x_2 \in V$, la subaditividad de L nos da $L(-x_1 - x_2) \leq L(-x_1) + L(-x_2)$. Usando la homogeneidad de L :

$$\begin{aligned} L(x_1 + x_2) &= L(-(-x_1 - x_2)) = -L(-x_1 - x_2) \\ &\geq -L(-x_1) - L(-x_2) = L(x_1) + L(x_2) \geq L(x_1 + x_2). \end{aligned}$$

Por ello, $L(x_1 + x_2) = L(x_1) + L(x_2)$ y por la arbitrariedad de $x_1, x_2 \in V$ concluimos que L es lineal. ■

Nos disponemos a probar, a partir del teorema de Hahn-Banach el de Mazur-Orlicz-König, que supone un refinamiento. Destacamos que el teorema de Hahn-Banach tiene gran cantidad de versiones equivalentes. Véase, por ejemplo, [7]. Antes, necesitamos un resultado técnico, que constituye una especie de versión global sobre un convexo del lema 6.1.

Lema 6.2. *Sea V un espacio vectorial y $P : V \rightarrow \mathbb{R}$ un funcional sublineal. Sea D un subconjunto convexo de V y sea $\beta := \inf_D P \in \mathbb{R}$. Para todo $x \in V$ tomamos*

$$Q(x) := \inf_{d \in D, \lambda > 0} [P(x + \lambda d) - \lambda \beta].$$

Entonces, $Q(V) \subset \mathbb{R}$, $Q : V \rightarrow \mathbb{R}$ es sublineal, $Q \leq P$ y además $\forall d \in D$ se cumple $-Q(-d) \geq \beta$.

Demuestração. Si $x \in V$, $d \in D$ y $\lambda > 0$, entonces

$$P(x + \lambda d) - \lambda\beta \geq -P(-x) + \lambda P(d) - \lambda\beta \geq -P(-x) > -\infty.$$

La primera desigualdad se deduce de la linealidad de P ya que:

$$\begin{aligned}\lambda P(d) &= P(\lambda d) \\ &= P(\lambda d + x - x) \\ &\leq P(x + \lambda d) + P(-x)\end{aligned}$$

por lo que $\lambda P(d) - P(-x) \leq P(x + \lambda d)$. La segunda se debe a que

$$\beta = \inf_D P \implies \lambda P(d) \geq \lambda\beta \implies \lambda P(d) - \lambda\beta \geq 0.$$

Tomando el ínfimo sobre $d \in D$ y $\lambda > 0$ llegamos a

$$Q(x) \geq -P(-x) > -\infty,$$

por lo que $Q(V) \subset \mathbb{R}$. Es relativamente fácil probar que Q es positivamente homogénea por lo que para ver que es sublineal solo queda comprobar la subaditividad. Para ello, tomamos $x_1, x_2 \in V$. Sean $d_1, d_2 \in D$ y $\lambda_1, \lambda_2 > 0$ arbitrarios. Para simplificar la notación llamamos $x := x_1 + x_2$, $\lambda := \lambda_1 + \lambda_2$ y $d := (\lambda_1/\lambda)d_1 + (\lambda_2/\lambda)d_2$. Notar que $d \in D$, al ser este convexo. Entonces:

$$\begin{aligned}P(x_1 + \lambda_1 d_1) - \lambda_1\beta + P(x_2 + \lambda_2 d_2) - \lambda_2\beta &\geq P(x + \lambda_1 d_1 + \lambda_2 d_2) - \lambda\beta \\ &= P(x + \lambda d) - \lambda\beta \\ &\geq Q(x) = Q(x_1 + x_2).\end{aligned}$$

Tomando ínfimo sobre d_1, d_2, λ_1 y λ_2 , $Q(x_1) + Q(x_2) \geq Q(x_1 + x_2)$. Así, Q es subaditiva y como consecuencia sublineal. Para concluir, fijamos $d \in D$. Sea $x \in V$ arbitrario. Entonces, $\forall \lambda > 0$, $Q(x) \leq P(x) + \lambda(P(d) - \beta)$. Tomando $\lambda \rightarrow 0$, $Q(x) \leq P(x)$ y como consecuencia $Q \leq P$. Finalmente, sea $d \in D$ arbitrario y tomando $\lambda = 1$:

$$Q(-d) \leq Q(-d + d) - \beta = -\beta \implies -Q(-d) \geq \beta.$$

■

Ya estamos preparados para probar el teorema de Mazur-Orlicz-König, debido a H. König [4, 5]. No solo es una versión equivalente del teorema de Hahn-Banach, también de un teorema de Mazur-Orlicz [6].

Teorema 6.2 (Mazur-Orlicz-König). *Sea V un espacio vectorial y $P : V \rightarrow \mathbb{R}$ un funcional sublineal. Sea D un subconjunto no vacío y convexo de V . Entonces existe un funcional $L : V \rightarrow \mathbb{R}$ lineal tal que $L \leq P$ e $\inf_D L = \inf_D P$.*

Antes de pasar a la demostración, observemos que, aunque es un resultado puramente algebraico, tiene una fuerte interpretación geométrica: el teorema de Hahn-Banach garantiza, dado un funcional sublineal $P : V \rightarrow \mathbb{R}$, la existencia de un funcional lineal $L : V \rightarrow \mathbb{R}$ con $L \leq P$. En la figura 6.2 mostramos unos ejemplos

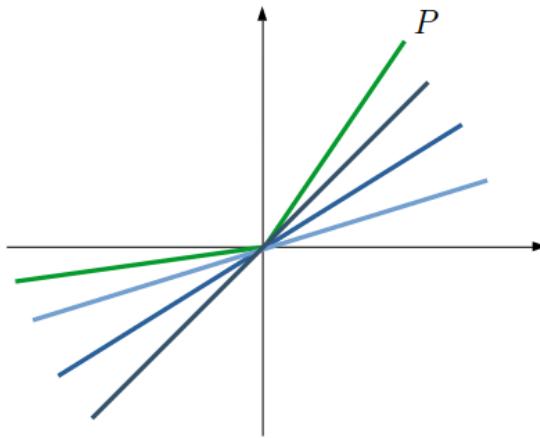


Figura 6.2: Ejemplos de funcionales aportados por el teorema de Hahn-Banach.

de funcionales que nos aporta el teorema de Hahn-Banach.

El teorema de Mazir-Orlicz-König, fijado un subconjunto convexo D de V , nos da solo los funcionales lineales que minoran a P y que cumplen

$$\inf_D L = \inf_D P.$$

En la imagen 6.3 se muestra un ejemplo del teorema.

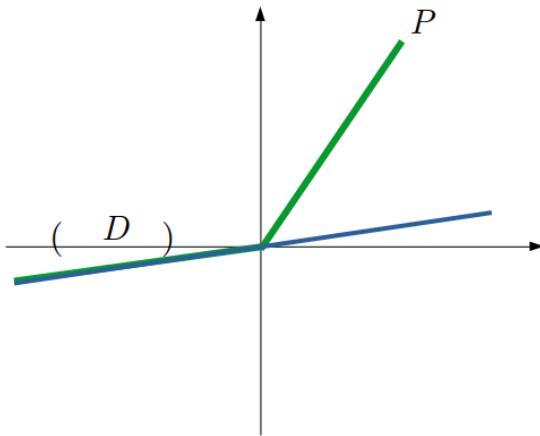


Figura 6.3: Teorema de Mazur-Orlicz-König.

*Demuestra*ción. Sea $\beta := \inf_D P$. En el caso de que $\beta = -\infty$ por el teorema de Hanh-Banach tenemos que $\exists L$ sobre V tal que es lineal y $L \leq P$. Así:

$$L \leq P \implies \inf_D L \leq \inf_D P = -\infty \implies \inf_D L = \inf_D P.$$

Supongamos entonces que $\beta \in \mathbb{R}$. Definimos el funcional auxiliar Q como en el lema 6.2. Del teorema de Hanh-Banach obtenemos que existe un funcional lineal L sobre V tal que $L \leq Q$ (como $Q \leq P$ tenemos que $L \leq P$). Sea $d \in D$, entonces:

$$L(d) = -L(-d) \geq -Q(-d) \geq \beta.$$

Tomando ínfimo sobre $d \in D$:

$$\inf_D L \geq \beta = \inf_D P.$$

Por otro lado, como $L \geq P$:

$$\inf_D L \leq \inf_D P.$$

Juntando ambas desigualdades obtenemos $\inf_D L = \inf_D P$. ■

Antes de probar la eficiencia del teorema de Mazur-Orlicz-König en el siguiente apartado, presentamos una consecuencia bien conocida. En particular, nos será de utilidad posteriormente para el teorema de separación. Recordemos dados E_1, E_2 dos espacios normados y $T : E_1 \rightarrow E_2$ un operador lineal, entonces T es continuo si, y solo si, verifica la siguiente condición:

$$\exists \alpha > 0 : \|T(x)\| \leq \alpha \|x\| \quad \forall x \in E_1,$$

Consideramos el espacio vectorial dado por:

$$E^* = \{T : E \rightarrow \mathbb{R} : T \text{ es lineal y continuo}\}.$$

conocido como el espacio dual (topológico) de E . Para todo $T \in E^*$ definimos su norma como:

$$\|T\| = \min\{\alpha > 0 : |T(x)| \leq \alpha \|x\| \quad \forall x \in E\}.$$

De este modo, podemos escribir:

$$|T(x)| \leq \|T\| \|x\|$$

siendo dicha desigualdad óptima. También podemos expresar su norma como el mínimo mayorante de un conjunto mayorado, es decir, el supremo:

$$\|T\| = \sup\{\|T(x)\| / \|x\| : \forall x \in E \setminus \{0\}\}.$$

Para $x \in E_1 \setminus \{0\}$ tenemos que $|T(x)| / \|x\| = \|T(x/\|x\|)\|$ y es claro que $\{x/\|x\| : x \in E_1 \setminus \{0\}\}$ es la esfera unidad de E que notamos como S_E . Si en vez de la esfera consideramos la bola unidad, B_E el supremo no varía. Efectivamente, si $x \in B_E$ se tiene que $x = \|x\| u$ con $u \in S_E$, y por ello $|T(x)| = \|x\| \|T(u)\| \leq \|T(u)\|$ ya que $\|x\| \leq 1$. De este modo, también tenemos que:

$$\|T\| = \sup_{x \in B_E} |T(x)|.$$

En este momento, estamos en disposición de enunciar y demostrar la igualdad que deseamos:

Corolario 6.1. *Dado un espacio normado E y $x \in E$, entonces se cumple que:*

$$\sup_{x^* \in B_{E^*}} x^*(x) = \|x\|. \quad (6.3)$$

Demuestra. Sea $x_0 \in E$ y sea $D := \{x_0\}$. Consideramos el funcional

$$\begin{aligned} P : E &\longrightarrow \mathbb{R} \\ x &\longmapsto \|x\|. \end{aligned}$$

Es claro que P es sublineal y que D es convexo. Podemos aplicar el Teorema de Mazur-Orlicz-König, teorema 6.2, y obtenemos que existe un funcional $L : E \rightarrow \mathbb{R}$ lineal tal que $L \leq P$ e $\inf_D L = \inf_D P$. Como $L \leq P$, entonces

$$\|L(x)\| \leq \|P(x)\| = \|x\| \quad \forall x \in E.$$

Concluimos que L es continua. Además, como tenemos que $L \in E^*$, llamamos $L = x^*$ y llegamos a que $\|L\| = \|x^*\| \leq 1$, es decir, $x^* \in B_{E^*}$. Por su parte, como $\inf_D L = \inf_D x^* = \inf_D P$ y $D = \{x_0\}$ entonces, $x^*(x_0) = \|x_0\|$. De este modo, llegamos a que existe $x^* \in B_{E^*}$ tal que $x^*(x_0) = \|x_0\|$. Si tomamos cualquier elemento $y^* \in B_{E^*}$, entonces

$$\|y^*(x_0)\| \leq \|y^*\| \|x_0\| \leq \|x_0\|$$

y podemos asegurar que:

$$\sup_{x^* \in B_{E^*}} x^*(x_0) = \|x_0\|.$$

Como $x_0 \in E$ es arbitrario, la desigualdad enunciada queda probada. ■

6.2. Teorema de la alternativa de Gordan. Reformulaciones.

Una vez que disponemos de la herramienta fundamental, el teorema de Mazur-Orlicz-König, nos disponemos a dar una versión convexa del teorema de la alternativa de Gordan. Antes de ello, y haciendo uso de este teorema tipo Hahn-Banach, probaremos el lema de Simons, sobre cierta condición de optimidad para funciones convexas.

Antes de comenzar, necesitamos hacer la siguiente definición. Para agilizar la lectura, cuando indicamos $N \in \mathbb{N}$ estamos suponiendo que $N \geq 1$. También, para $N \in \mathbb{N}$, notaremos con caracteres en negrita a los elementos de \mathbb{R}^N , por ejemplo, $\mathbf{t} = (t_1, \dots, t_N) \in \mathbb{R}^N$.

Definición 6.2. *Dado $N \in \mathbb{N}$ llamamos simplex unitario de \mathbb{R}^N al subconjunto de \mathbb{R}^N definido como:*

$$\Delta_N := \left\{ \mathbf{t} \in \mathbb{R}^N : \sum_{i=1}^N t_i = 1, \quad t_1, \dots, t_N \geq 0 \right\}.$$

Recordamos ahora la noción de envolvente convexa de un subconjunto cualquiera X de un espacio vectorial V , que notamos como $\text{co}(X)$ y que se define como la intersección de todos los conjuntos convexos que contienen a X . Nótese que $\Delta_N = \text{co}\{e_1, \dots, e_N\}$.

Antes de continuar veamos que Δ_N es convexo y compacto:

- Convexo: tenemos que comprobar que dados $\mathbf{t}, \mathbf{s} \in \Delta_N$ y $\lambda \in [0, 1]$ entonces $\lambda\mathbf{t} + (1 - \lambda)\mathbf{s} \in \Delta_N$. En efecto, las coordenadas $\lambda t_i + (1 - \lambda)s_i$ verifican:
 - i) $\lambda t_i + (1 - \lambda)s_i \geq 0$ para todo $i = 1, \dots, N$, ya que $t_i, s_i \geq 0$ y $0 \leq \lambda \leq 1$.
 - ii)

$$\begin{aligned} \sum_{i=1}^N (\lambda t_i + (1 - \lambda)s_i) &= \lambda \sum_{i=1}^N t_i + (1 - \lambda) \sum_{i=1}^N s_i \\ &= \lambda + (1 - \lambda) \\ &= 1, \end{aligned}$$

ya que $\sum_{i=1}^N t_i = \sum_{i=1}^N s_i = 1$.

Por lo tanto, $\lambda\mathbf{t} + (1 - \lambda)\mathbf{s} \in \Delta_N$ para todo $\lambda \in [0, 1]$.

- Compacto: al encontrarnos en \mathbb{R}^N y aplicando el conocido Teorema de Heine-Borel basta y sobra ver que Δ_N es cerrado y acotado. Pero claramente es acotado por lo que nos centraremos en probar que es cerrado. Sea $\{\mathbf{t}_n\}_{n \in \mathbb{N}}$ una sucesión de Δ_N y sea $\mathbf{t}_0 \in \mathbb{R}^N$ tal que $\{\mathbf{t}_n\}_{n \in \mathbb{N}} \rightarrow \mathbf{t}_0$. Tenemos que comprobar que $\mathbf{t}_0 \in \Delta_N$.
 - i) Como todas las coordenadas de cada \mathbf{t}_n para $n \in \mathbb{N}$ son no negativas las de \mathbf{t}_0 también lo son.
 - ii) La función $f : \mathbb{R}^N \rightarrow \mathbb{R}$ definida como $f(\mathbf{t}) = \sum_{i=1}^N t_i$ es continua. Claramente $f(\mathbf{t}) = 1$ para todo $\mathbf{t} \in \Delta_N$ y por ello $\{f(\mathbf{t}_n)\}_{n \in \mathbb{N}} \rightarrow 1$. Por continuidad de f y unicidad de límite tenemos que $f(\mathbf{t}_0) = 1$ pero eso equivale a que la suma de sus componentes vale 1.

Así, hemos demostrado que $\mathbf{t}_0 \in \Delta_N$ y por lo tanto Δ_N es compacto.

Antes de continuar, notamos que el espacio vectorial de todas las aplicaciones lineales de \mathbb{R}^N a \mathbb{R} , con $N \in \mathbb{N}$, se puede identificar con \mathbb{R}^N . Esto se debe a que si $L : \mathbb{R}^N \rightarrow \mathbb{R}$ es lineal, entonces es de la forma $L(\mathbf{x}) = a_1x_1 + \dots + a_Nx_N$, que se corresponde con $\mathbf{a} = (a_1, \dots, a_N) \in \mathbb{R}^N$. Del mismo modo, dado el vector tenemos la aplicación lineal asociada. En definitiva, el espacio dual de \mathbb{R}^N se identifica con \mathbb{R}^N .

También presentamos el producto escalar usual en \mathbb{R}^N que definimos como:

$$\begin{aligned} \langle \cdot, \cdot \rangle : \mathbb{R}^N \times \mathbb{R}^N &\longrightarrow \mathbb{R} \\ (\mathbf{x}, \mathbf{y}) &\longmapsto \langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^N x_i y_i. \end{aligned}$$

30 6.2. Teorema de la alternativa de Gordan. Reformulaciones.

Antes de enunciar el lema de Simons, hacemos otra identificación, esta vez, correspondiente a un simplex unitario:

Lema 6.3. *Sea $N \in \mathbb{N}$ y $S : \mathbb{R}^N \rightarrow \mathbb{R}$ definida por:*

$$S(\mathbf{x}) := \max\{x_1, \dots, x_N\}.$$

Entonces, S es sublineal. Además, si $L : \mathbb{R}^N \rightarrow \mathbb{R}$ es un funcional lineal tal que $L \leq S$ entonces L es de la forma

$$L(\mathbf{x}) = t_1 x_1 + \dots + t_N x_N$$

con $(t_1, \dots, t_N) \in \Delta_N$. De hecho, el recíproco también es cierto, es decir, si $L = \mathbf{t} = (t_1, \dots, t_N) \in \Delta_N$ entonces $L \leq S$.

*Demuestra*ón. Claramente, S es positivamente homogénea. También es subaditiva ya que dados $x, y \in V$:

$$\begin{aligned} S(\mathbf{x} + \mathbf{y}) &= \max\{x_1 + y_1, \dots, x_N + y_N\} \\ &\leq \max\{x_1, \dots, x_N\} + \max\{y_1, \dots, y_N\} = S(\mathbf{x}) + S(\mathbf{y}). \end{aligned}$$

Por ello, S es sublineal. Para terminar veamos que

$$\{L : \mathbb{R}^N \rightarrow \mathbb{R} : L \text{ lineal y } L \leq S\} = \Delta_N$$

a través de la correspondencia mostrada anteriormente entre \mathbb{R}^N y su dual.

\supseteq) Sea $\mathbf{t} \in \Delta_N$, definimos $L : \mathbb{R}^N \rightarrow \mathbb{R}$ como $L(\mathbf{x}) := \langle \mathbf{t}, \mathbf{x} \rangle$. Es evidente que L es lineal en \mathbf{x} al ser el producto escalar bilineal. Dado $\mathbf{x} \in \mathbb{R}^N$,

$$L(\mathbf{x}) = \sum_{i=1}^N t_i x_i \leq \sum_{i=1}^N t_i S(\mathbf{x}) = S(\mathbf{x}) \sum_{i=1}^N t_i = S(\mathbf{x}),$$

donde la primera desigualdad se debe a que $x_i \leq S(\mathbf{x})$ para todo x_i con $i = 1, \dots, N$ y a que $t_i \geq 0$ ya que $\mathbf{t} \in \Delta_N$. Esto también justifica la última igualdad ya que $\sum_{i=1}^N t_i = 1$.

\subseteq) Sea $L = \mathbf{t} = (t_1, \dots, t_N) \in \mathbb{R}^N$ lineal tal que para todo $\mathbf{x} \in \mathbb{R}^N$ cumple que $\sum_{i=1}^N t_i x_i \leq S(\mathbf{x})$. Así, si tomamos $e_i \in \mathbb{R}^N$ donde e_i representa el i-ésimo elemento de la base usual de \mathbb{R}^N con $i = 1, \dots, N$, entonces

$$L(-e_i) = -t_i \leq 0 \implies t_i \geq 0 \quad \forall i = 1, \dots, N.$$

Si ahora llamamos $e = \sum_{i=1}^N e_i$, obtenemos:

$$\left. \begin{aligned} L(e) &= \sum_{i=1}^N t_i \leq \max\{1, \dots, 1\} = 1 \\ L(-e) &= -\sum_{i=1}^N t_i \leq \max\{-1, \dots, -1\} = -1 \end{aligned} \right\} \implies \sum_{i=1}^N t_i = 1.$$

Concluimos entonces que $L = \mathbf{t} = (t_1, \dots, t_N) \in \Delta_N$.

■

Enunciamos ahora el lema de Simons [8]. Tal y como se ha mencionado antes, se trata de un resultado sobre funciones convexas y cierta condición de optimalidad: dos funciones distintas poseen el mismo ínfimo en un convexo. Esto sugiere el uso del teorema de Mazur-Orlicz-König.

Lema 6.4 (Simons). *Sea C un subconjunto convexo de un espacio vectorial y $N \in \mathbb{N}$. Dadas $f_1, \dots, f_N : C \rightarrow \mathbb{R}$, funciones convexas, entonces existe $\mathbf{t} \in \Delta_N$ que cumple*

$$\inf_{c \in C} \left[\max_{i=1, \dots, N} f_i(c) \right] = \inf_{c \in C} \left[\sum_{i=1}^N t_i f_i(c) \right].$$

Demostración. Sea $V = \mathbb{R}^N$. Definimos $S : V \rightarrow \mathbb{R}$ como

$$S(x_1, \dots, x_N) := \max\{x_1, \dots, x_N\}.$$

Por el lema 6.3, S es sublineal. Tomamos el subconjunto:

$$D = \{(x_1, \dots, x_N) \in V : \exists c \in C \text{ tal que } \forall i = 1, \dots, N, f_i(c) \leq x_i\}.$$

Comprobemos en primer lugar que D es un subconjunto convexo de V . Sean $\mathbf{x}, \mathbf{y} \in D$, por ello, existen $c_x, c_y \in C$ tales que $f_i(c_x) \leq x_i$ y $f_i(c_y) \leq y_i \quad \forall i = 1, \dots, N$. Dado $\lambda \in [0, 1]$, llamamos $c := (1-\lambda)c_x + \lambda c_y$ que pertenece a C por ser este convexo. Veamos que c es el elemento necesario de C para que cualquier combinación convexa de x e y esté en D . Así, para todo $i = 1, \dots, N$:

$$f_i(c) = f_i((1-\lambda)c_x + \lambda c_y) \leq (1-\lambda)f_i(c_x) + \lambda f_i(c_y) \leq (1-\lambda)x_i + \lambda y_i,$$

donde la primera desigualdad se debe a que las f_i son convexas y la segunda a que $\mathbf{x}, \mathbf{y} \in D$. Por ello, $(1-\lambda)x_i + \lambda y_i \in D, \quad \forall \lambda \in [0, 1]$ por lo que D es convexo. Aplicando el Teorema de Mazur-Orlicz-König, existe L funcional lineal en V tal que $L \leq S$ e $\inf_D L = \inf_D S$.

Nuevamente, por el lema 6.3 tenemos que $L = \mathbf{t} \in \Delta_N$. Finalmente:

$$\inf_D L = \inf_{c \in C} \left[\sum_{i=1}^N t_i f_i(c) \right]$$

e

$$\inf_D S = \inf_{c \in C} \left[\max_{i=1, \dots, N} f_i(c) \right]$$

por lo que

$$\inf_{c \in C} \left[\max_{i=1, \dots, N} f_i(c) \right] = \inf_{c \in C} \left[\sum_{i=1}^N t_i f_i(c) \right].$$

■

Enunciamos ahora el Teorema de la Alternativa de Gordan en su versión clásica.

32 6.2. Teorema de la alternativa de Gordan. Reformulaciones.

Teorema 6.3 (Teorema de la alternativa de Gordan). *Dados $N, M \in \mathbb{N}$, sean $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ con $\mathbf{x}_i \in \mathbb{R}^M$ para $i = 1, \dots, N$. Entonces una, y solo una, de las siguientes afirmaciones se cumple:*

$$i^*) \exists \mathbf{t} \in \Delta_N \text{ tal que } 0 = \sum_{i=1}^N t_i \mathbf{x}_i.$$

$$ii^*) \exists \mathbf{y} \in \mathbb{R}^M \text{ tal que cumple } \max_{i=1, \dots, N} \langle \mathbf{y}, \mathbf{x}_i \rangle < 0.$$

Omitimos esta demostración ya que a continuación mostramos la versión convexa del mismo y será la que probaremos. Después veremos que la versión convexa implica la clásica por lo que quedará probada. Antes de continuar, destacar que la interpretación geométrica de este resultado se observa en la figura 6.4.

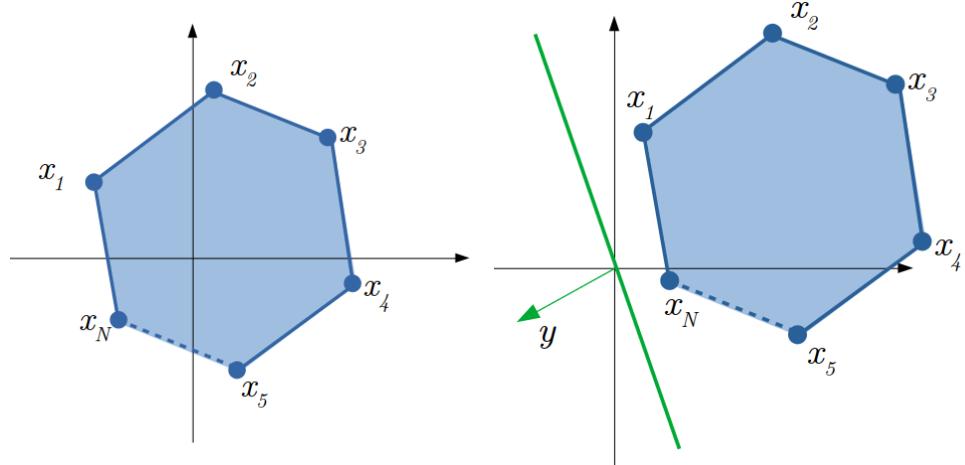


Figura 6.4: A la izquierda alternativa i^*) y a la derecha la alternativa ii^*).

Teorema 6.4 (Teorema de la Alternativa de Gordan-versión convexa). *Sean C un subconjunto convexo de un espacio vectorial, $N \in \mathbb{N}$ y $f_1, \dots, f_N : C \rightarrow \mathbb{R}$ funciones convexas. Entonces una, y solo una, de las siguientes afirmaciones se cumple:*

$$i) \exists \mathbf{t} \in \Delta_N \text{ tal que } 0 \leq \inf_{c \in C} \sum_{i=1}^N t_i f_i(c).$$

$$ii) \exists c \in C \text{ que cumple } \max_{i=1, \dots, N} f_i(c) < 0.$$

Demostración. Veamos que las alternativas $i)$ y $ii)$ son excluyentes y exhaustivas. Para ello, veamos que $\neg i) \iff ii)$. Suponemos en primer lugar que el valor de $\inf_{c \in C} [\max_{i=1, \dots, N} f_i(c)] \in \mathbb{R}$.

\Leftarrow) Si existe $c_0 \in C$ tal que $\max_{i=1,\dots,N} f_i(c) < 0$, entonces, dado $t \in \Delta_N$,

$$\inf_{c \in C} \sum_{i=0}^N t_i f_i(c) \leq \sum_{i=0}^N t_i f_i(c_0) < 0$$

donde la última desigualdad se debe a que $t \in \Delta_N$. Hemos probado entonces $\neg i$.

\Rightarrow) Si aplicamos el lema de Simons, lema 6.4, a las funciones f_1, \dots, f_N obtenemos:

$$\exists t \in \Delta_N : \inf_{c \in C} \left[\max_{i=1,\dots,N} f_i(c) \right] = \inf_{c \in C} \sum_{i=1}^N t_i f_i(c).$$

Entonces, de $\neg i$ sabemos que, para este $t \in \Delta_N$ (como para cualquier otro)

$$\inf_{c \in C} \sum_{i=0}^N t_i f_i(c) < 0$$

luego

$$\inf_{c \in C} \left[\max_{i=1,\dots,N} f_i(c) \right] < 0,$$

como queríamos demostrar.

Para finalizar, si $\inf_{c \in C} [\max_{i=1,\dots,N} f_i(c)] = -\infty$ es claro que estamos en el caso ii) y se prueba como en el caso de que dicho ínfimo sea un valor real. \blacksquare

Destacamos las siguientes observaciones:

Observación 6.1. Esta versión convexa del teorema implica la versión clásica del mismo.

Para ello, basta aplicar la versión convexa del teorema a $C := \mathbb{R}^M$ y a las funciones $f_1, \dots, f_N : C \rightarrow \mathbb{R}$ definidas por

$$f_i(\mathbf{c}) := \langle \mathbf{c}, \mathbf{x}_i \rangle, \forall i = 1, \dots, N.$$

Notar que las funciones f_1, \dots, f_N son lineales por la izquierda y como consecuencia son convexas. En este caso, la alternativa ii) implica ii^*) ya que:

$$\exists \mathbf{c} \in C = \mathbb{R}^M : \max_{i=1,\dots,N} \langle \mathbf{c}, \mathbf{x}_i \rangle = \max_{i=1,\dots,N} f_i(\mathbf{c}) < 0.$$

Por su parte, la alternativa i) nos da:

$$\exists \mathbf{t} \in \Delta_N : 0 \leq \inf_{\mathbf{c} \in \mathbb{R}^M} \sum_{i=1}^N t_i f_i(\mathbf{c}) = \inf_{\mathbf{c} \in \mathbb{R}^M} \sum_{i=1}^N t_i \langle \mathbf{c}, \mathbf{x}_i \rangle = \inf_{\mathbf{c} \in \mathbb{R}^M} \langle \mathbf{c}, \sum_{i=1}^N t_i \mathbf{x}_i \rangle.$$

Hemos obtenido por ello que $0 \leq \inf_{\mathbf{c} \in \mathbb{R}^M} \langle \mathbf{c}, \sum_{i=1}^N t_i \mathbf{x}_i \rangle$ lo que significa que $0 \leq \langle \mathbf{c}, \sum_{i=1}^N t_i \mathbf{x}_i \rangle$ para todo $\mathbf{c} \in \mathbb{R}^M$. Usando la bilinealidad del producto escalar:

$$0 \leq \langle -\mathbf{c}, \sum_{i=1}^N t_i \mathbf{x}_i \rangle \iff 0 \leq -\langle \mathbf{c}, \sum_{i=1}^N t_i \mathbf{x}_i \rangle \iff \langle \mathbf{c}, \sum_{i=1}^N t_i \mathbf{x}_i \rangle \leq 0, \quad \forall \mathbf{c} \in \mathbb{R}^M.$$

34 6.2. Teorema de la alternativa de Gordan. Reformulaciones.

Juntando ambas desigualdades obtenemos que $0 = \langle \mathbf{c}, \sum_{i=1}^N t_i \mathbf{x}_i \rangle, \forall \mathbf{c} \in \mathbb{R}^M$. Como la igualdad anterior se cumple para todo elemento de \mathbb{R}^M entonces podemos deducir que $\sum_{i=1}^N t_i \mathbf{x}_i = 0$ ya que $\sum_{i=1}^N t_i \mathbf{x}_i \in (\mathbb{R}^M)^\perp = \{0\}$. Así pues, tenemos que $i)$ equivale a i^*). ■

Observación 6.2. El lema de Simons (lema 6.4) y el teorema convexo de la Alternativa de Gordan (teorema 6.4) son equivalentes.

Ya hemos visto que el Lema de Simons implica el Teorema de la Alternativa de Gordan. Veamos que el recíproco también es cierto.

Llamamos $\alpha := \inf_{c \in C} [\max_{i=1,\dots,N} \{f_i(c)\}]$. Si $\alpha = -\infty$. Por el lema 6.3 sabemos que $\forall \mathbf{t} \in \Delta_N$ se cumple que $\sum_{i=1}^N t_i f_i(c) \leq \max_{i=1,\dots,N} \{f_i(c)\}$ para todo $c \in C$. Tomando ínfimos en C:

$$\inf_{c \in C} \left[\sum_{i=1}^N t_i f_i(c) \right] \leq \inf_{c \in C} \left[\max_{i=1,\dots,N} \{f_i(c)\} \right] = -\infty \implies \inf_{c \in C} \left[\sum_{i=1}^N t_i f_i(c) \right] = -\infty$$

y por ello $\forall \mathbf{t} \in \Delta_N$ (en particular para uno cualquiera) se cumple que

$$\inf_{c \in C} \left[\sum_{i=1}^N t_i f_i(c) \right] = \inf_{c \in C} \left[\max_{i=1,\dots,N} \{f_i(c)\} \right].$$

Supongamos ahora que $\alpha \in \mathbb{R}$. Sean las funciones $g_1, \dots, g_N : C \rightarrow \mathbb{R}$ definidas como $g_i = f_i - \alpha$ con $i = 1, \dots, N$. Veamos que las funciones g_1, \dots, g_N son convexas como consecuencia de que f_1, \dots, f_N lo son. Sean $i = 1, \dots, N, c_1, c_2 \in C$ y $\lambda \in [0, 1]$:

$$\begin{aligned} g_i(\lambda c_1 + (1 - \lambda)c_2) &= f_i(\lambda c_1 + (1 - \lambda)c_2) - \alpha \\ &\leq \lambda f_i(c_1) + (1 - \lambda)f_i(c_2) - \alpha \\ &= \lambda f_i(c_1) + (1 - \lambda)f_i(c_2) - \lambda\alpha + (1 - \lambda)\alpha \\ &= \lambda(f_i(c_1) - \alpha) + (1 - \lambda)(f_i(c_2) - \alpha) \\ &= \lambda g_i(c_1) + (1 - \lambda)g_i(c_2). \end{aligned}$$

Obtenemos así que g_i es convexa para todo $i = 1, \dots, N$. Si usamos el Teorema de la Alternativa de Gordan obtenemos que solo se puede dar una y solo de las siguientes posibilidades:

i) $\exists \mathbf{t} \in \Delta_N$ tal que $0 \leq \inf_{c \in C} \sum_{i=1}^N t_i g_i(c)$.

ii) $\exists c \in C$ que cumple $\max_{i=1,\dots,N} \{g_i(c)\} < 0$.

Razonemos que no se puede dar ii). Si fuese así, tendríamos que $\exists c \in C$ tal que $\max_{i=1,\dots,N} \{g_i(c)\} = \max_{i=1,\dots,N} \{f_i(c) - \alpha\} < 0$. En particular, se cumpliría

$$f_j(c) - \alpha < 0 \implies f_j(c) < \alpha = \inf_{c \in C} \left[\max_{i=1,\dots,N} \{f_i(c)\} \right], \quad \forall j \in 1, \dots, N.$$

Esto es imposible por la propia definición de α . Por ello, afirmamos que $\exists \mathbf{t} \in \Delta_N$ tal que $0 \leq \inf_C \sum_{i=1}^N t_i g_i$. Desarrollando el sumatorio:

$$\begin{aligned} 0 &\leq \inf_{c \in C} \sum_{i=1}^N t_i g_i(c) \\ &= \inf_{c \in C} \sum_{i=1}^N t_i (f_i(c) - \alpha) \\ &= \inf_{c \in C} \left[\sum_{i=1}^N t_i f_i(c) - \sum_{i=1}^N t_i \alpha \right] \\ &= \inf_{c \in C} \left[\sum_{i=1}^N t_i f_i(c) - \alpha \sum_{i=1}^N t_i \right] \\ &= \inf_{c \in C} \left[\sum_{i=1}^N t_i f_i(c) - \alpha \right] \\ &= \inf_{c \in C} \left[\sum_{i=1}^N t_i f_i(c) \right] - \alpha. \end{aligned}$$

Por lo tanto:

$$0 \leq \inf_{c \in C} \left[\sum_{i=1}^N t_i f_i(c) \right] - \alpha \iff \inf_{c \in C} \left[\max_{i=1,\dots,N} \{f_i(c)\} \right] = \alpha \leq \inf_{c \in C} \left[\sum_{i=1}^N t_i f_i(c) \right].$$

El lema 6.3 nos aporta la otra desigualdad y llegamos nuevamente a que $\exists \mathbf{t} \in \Delta_N$ que cumple:

$$\inf_{c \in C} \left[\max_{i=1,\dots,N} \{f_i(c)\} \right] = \inf_{c \in C} \left[\sum_{i=1}^N t_i f_i(c) \right].$$

■

Mencionemos antes de terminar que el teorema de la alternativa de Gordan no solo es equivalente al lema de Simons. Otro tipo de teorema de la alternativa equivalente es el lema de Farkas que estudiaremos en el capítulo sobre optimización. Son muchos los teoremas de la alternativa que se conocen, y siempre vinculados a la optimización: véase, por ejemplo, la amplia muestra que aparece recogida en [3]. A modo de ejemplo, y usando la notación clásica matricial, enunciaremos algunos de los resultados más conocidos:

Teorema 6.5 (Motzkin). *Dados $N, M \in \mathbb{N}$, tomamos las matrices $A, B, D \in \mathbb{R}^{M \times N}$. Entonces una, y solo una, de las siguientes condiciones se cumple:*

m1) $\exists \mathbf{x} \in \mathbb{R}^N$ tal que $A\mathbf{x} = 0$, $B\mathbf{x} \geq 0$ y $D\mathbf{x} > 0$.

m2) $\exists \mathbf{y}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^M$ tales que $\mathbf{y}^T A + \mathbf{v}^T B + \mathbf{w}^T D = 0$, $\mathbf{v} \geq 0$ y $\mathbf{w} \geq 0$.

36 6.2. Teorema de la alternativa de Gordan. Reformulaciones.

Teorema 6.6 (Tucker). *Dados $N, M \in \mathbb{N}$, tomamos $A, B, D \in \mathbb{R}^{M \times N}$. Entonces una, y solo una, de las siguientes condiciones se cumple:*

- m1) $\exists \mathbf{x} \in \mathbb{R}^N$ tal que $A\mathbf{x} = 0$, $B\mathbf{x} \geq 0$ y $D\mathbf{x} \geq 0$.*
- m2) $\exists \mathbf{y}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^M$ tales que $\mathbf{y}^T A + \mathbf{v}^T B + \mathbf{w}^T D = 0$, $\mathbf{v} \geq 0$ y $\mathbf{w} > 0$.*

Teorema 6.7 (Primer teorema de la alternativa de Fenchel). *Dados $N, M \in \mathbb{N}$, tomamos $A, B \in \mathbb{R}^{M \times N}$. Entonces una, y solo una, de las siguientes condiciones se cumple:*

- m1) $\exists \mathbf{x}, \mathbf{z} \in \mathbb{R}^N$ tales que $A\mathbf{x} + B\mathbf{z} = 0$, $\mathbf{x} \geq 0$ y $\mathbf{z} \geq 0$.*
- m2) $\exists \mathbf{y} \in \mathbb{R}^M$ tal que $\mathbf{y}^T A \geq 0$ e $\mathbf{y}^T B > 0$.*

Alternativa y minimax

Este capítulo se centra en el uso de los teoremas de la alternativa en la teoría minimax, surgida a finales de la década de 1920 de la mano de J. von Neumann en el seno de la teoría de juegos. Sin embargo, nosotros aplicaremos un teorema minimax deducido del teorema de la alternativa de Gordan para obtener resultados de separación de convexos

7.1. Una desigualdad minimax

En esta sección llegaremos a otro de los resultados clave del trabajo. Será uno de los denominados teoremas minimax. A rasgos generales y a modo introductorio, podemos decir que un teorema minimax es un resultado que afirma, bajo ciertas hipótesis, que:

$$\inf_{y \in Y} \sup_{x \in X} f(x, y) = \sup_{x \in X} \inf_{y \in Y} f(x, y),$$

donde X e Y son conjuntos y $f : X \times Y \rightarrow \mathbb{R}$. Obviamente, esta igualdad no es cierta en general tal y como mostramos en el siguiente ejemplo. Definimos $f : \{0, 1\} \times \{0, 1\} \rightarrow \mathbb{R}$ como:

$$f(x, y) = \begin{cases} 0, & \text{si } x = y \\ 1, & \text{si } x \neq y \end{cases}.$$

Por un lado tenemos

$$\inf_{y \in Y} \sup_{x \in X} f(x, y) = \min_{y \in Y} \max_{x \in X} f(x, y) = \min_{y \in Y} \{1\} = 1,$$

y por otro

$$\sup_{x \in X} \inf_{y \in Y} f(x, y) = \max_{x \in X} \min_{y \in Y} f(x, y) = \max_{x \in X} \{0\} = 0.$$

Notemos que la desigualdad

$$\inf_{y \in Y} \sup_{x \in X} f(x, y) \geq \sup_{x \in X} \inf_{y \in Y} f(x, y)$$

siempre se da ya que si $(x_0, y_0) \in X \times Y$, entonces:

$$\sup_{x \in X} f(x, y_0) \geq f(x_0, y_0) \geq \inf_{y \in Y} f(x_0, y).$$

Por ello, los teoremas minimax solo nos aportan la otra desigualdad necesaria, de ahí que este tipo de resultados se conozcan también como desigualdad minimax.

Antes de continuar, exponemos la siguiente definición que aparecerá posteriormente en el teorema. Se trata de una propiedad más débil que la continuidad para funciones reales.

Definición 7.1. *Sea X un espacio topológico. Decimos que $f : X \rightarrow \mathbb{R}$ es superiormente semicontinua en X si para todo $r \in \mathbb{R}$ se cumple que el conjunto $\{x \in X : f(x) \geq r\}$ es cerrado.*

Es claro que toda función continua en X es superiormente semicontinua, aunque el recíproco no es cierto, tal y como prueba la función $f : \mathbb{R} \rightarrow \mathbb{R}$ dada por

$$f(x) = \begin{cases} 0, & \text{si } x < 0 \\ 1, & \text{si } x \geq 0 \end{cases}.$$

Para funciones superiormente semicontinuas, podemos enunciar los siguientes resultados. En particular, el lema 7.2 es análogo al del caso continuo.

Lema 7.1. *El ínfimo de una familia de funciones reales $(f_\lambda)_{\lambda \in \Lambda}$ definidas en el mismo espacio topológico X , si es finito, determina una función superiormente semicontinua.*

Este hecho es inmediato de la definición pues dado $r \in \mathbb{R}$,

$$\begin{aligned} \{x \in X : \inf_{\lambda \in \Lambda} f_\lambda(x) \geq r\} &= \{x \in X : \lambda \in \Lambda \implies f_\lambda(x) \geq r\} \\ &= \bigcap_{\lambda \in \Lambda} \{x \in X : f_\lambda(x) \geq r\}, \end{aligned}$$

que es cerrado por ser intersección de cerrados.

Lema 7.2. *Si X es un espacio topológico compacto y $f : X \rightarrow \mathbb{R}$ es superiormente semicontinua, entonces f alcanza su supremo en X .*

La demostración de este resultado está recogida en [1].

En estos momentos nos encontramos en condiciones de enunciar y demostrar el siguiente teorema minimax, más general que el de von Neumann. Los ingredientes de la prueba son las condiciones topológicas que se suponen y el teorema de la alternativa de Gordan, versión convexa, en forma equivalente del lema de Simons. Seguimos la demostración dada por [8].

Teorema 7.1. Sean X, Y subconjuntos convexos de sendos espacios vectoriales (no tienen que ser el mismo) tal que X está dotado de una topología que lo hace compacto. Supongamos además que $f : X \times Y \rightarrow \mathbb{R}$ es:

- i) cóncava y superiormente semicontinua en X y
- ii) convexa en Y .

Entonces:

$$\inf_{y \in Y} \max_{x \in X} f(x, y) = \max_{x \in X} \inf_{y \in Y} f(x, y).$$

Demostración. En primer lugar, por lemas 7.1 y 7.2 podemos escribir máximo en ambos casos en vez de supremo ya que f es superiormente semicontinua en X , por ello $\inf_{y \in Y} f(x, y)$ también lo es y X es compacto. Como hemos explicado anteriormente, solo necesitamos la desigualdad

$$\inf_{y \in Y} \max_{x \in X} f(x, y) \leq \max_{x \in X} \inf_{y \in Y} f(x, y). \quad (7.1)$$

Definimos $\alpha := \inf_{y \in Y} \max_{x \in X} f(x, y)$. Si $\alpha = -\infty$ no hay nada que probar, por lo que supondremos que $\alpha \in \mathbb{R}$. Primero vamos a reescribir el resultado a probar. La desigualdad (7.1) es equivalente a

$$\exists x_0 \in X : \alpha \leq \inf_{y \in Y} f(x_0, y),$$

ya que si existe un elemento en X que lo cumpla el máximo también lo cumplirá y recíprocamente. O lo que es lo mismo:

$$\exists x_0 \in X : y \in Y \implies \alpha \leq f(x_0, y),$$

o lo que es igual,

$$\bigcap_{y \in Y} \{x \in X : \alpha \leq f(x, y)\} \neq \emptyset. \quad (7.2)$$

Como f es superiormente semicontinua en X estamos ante una intersección de cerrados. Usando la propiedad de intersección finita (X es compacto) obtenemos que (7.2) equivale a

$$\begin{aligned} N \in \mathbb{N}, y_1, \dots, y_N \in Y &\implies \bigcap_{i=1}^N \{x \in X : \alpha \leq f(x, y_i)\} \neq \emptyset. \\ &\Updownarrow \\ N \in \mathbb{N}, y_1, \dots, y_N \in Y &\implies \exists x_0 \in X : \alpha \leq \min_{i=1, \dots, N} f(x_0, y_i). \\ &\Updownarrow \\ N \in \mathbb{N}, y_1, \dots, y_N \in Y &\implies \alpha \leq \max_{x \in X} \min_{i=1, \dots, N} f(x, y_i). \end{aligned} \quad (7.3)$$

Esta última condición es cierta. En efecto, sean $N \in \mathbb{N}$ e $\{y_1, \dots, y_N\} \in Y$. Aplicamos el lema de Simons, lema 6.4, tomando $C := X$ y $f_i : X \rightarrow \mathbb{R}$ definidas como

$$f_i(x) := -f(x, y_i), \quad i = 1, \dots, N.$$

Como f es cóncava respecto a X tenemos que las f_i son convexas en X con $i = 1, \dots, N$. De este modo, existe $\mathbf{t} \in \Delta_N$ tal que

$$\inf_{x \in X} \max_{i=1, \dots, N} \{f_i(x)\} = \inf_{x \in X} \left[\sum_{i=1}^N t_i f_i(x) \right].$$

Si ponemos la igualdad en función de f y recordamos que alcanza el supremo en X ,

$$\begin{aligned} \inf_{x \in X} \max_{i=1, \dots, N} \{-f(x, y_i)\} &= \inf_{x \in X} \left[\sum_{i=1}^N t_i (-f(x, y_i)) \right] \\ &\Downarrow \\ \max_{x \in X} \min_{i=1, \dots, N} \{f(x, y_i)\} &= \max_{x \in X} \left[\sum_{i=1}^N t_i f(x, y_i) \right]. \end{aligned}$$

Al ser f convexa en Y :

$$\begin{aligned} \max_{x \in X} \min_{i=1, \dots, N} \{f(x, y_i)\} &= \max_{x \in X} \left[\sum_{i=1}^N t_i f(x, y_i) \right] \\ &\geq \max_{x \in X} \left[f\left(x, \sum_{i=1}^N t_i y_i\right) \right] \\ &\geq \inf_{y \in Y} \max_{x \in X} f(x, y) \\ &= \alpha. \end{aligned}$$

Hemos probado entonces la desigualdad (7.3) y con ello el teorema. ■

7.2. Separación de convexos

En esta sección introducimos algunos resultados sobre separación a partir del teorema minimax, teorema 7.1. En general, estos nos aportan herramientas para poder concluir cuándo dos subconjuntos convexos pueden ser separados mediante un hiperplano. En la siguiente figura 7.1 vemos un ejemplo sobre la situación en la que nos encontramos.

Exponemos un resultado sencillo en el que solo involucramos un conjunto.

Teorema 7.2. *Dado $N \in \mathbb{N}$, sea $C \subset \mathbb{R}^N$ convexo y definimos*

$$\delta := \inf\{\|\mathbf{c}\| : \mathbf{c} \in C\}.$$

Entonces, existe $\mathbf{x}_0 \in \mathbb{R}^N$ tal que

$$\mathbf{c} \in C \implies \delta \leq \langle \mathbf{x}_0, \mathbf{c} \rangle.$$

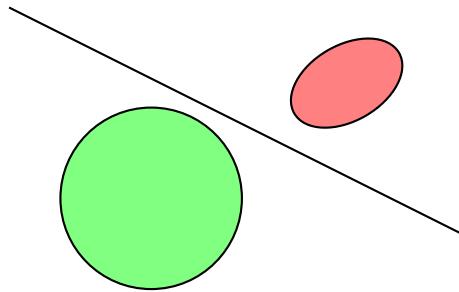


Figura 7.1: Situación de teoremas de separación

Demuestra. En primer lugar, vamos a reescribir la tesis del teorema. Queremos ver que:

$$\exists \mathbf{x}_0 \in \mathbb{R}^N : \mathbf{c} \in C \implies \delta \leq \langle \mathbf{x}_0, \mathbf{c} \rangle.$$

\Updownarrow

$$\exists \alpha > 0, \exists \mathbf{x}_0 \in \alpha B_{\mathbb{R}^N} : \mathbf{c} \in C \implies \delta \leq \langle \mathbf{x}_0, \mathbf{c} \rangle.$$

\Updownarrow

$$\exists \alpha > 0, \exists \mathbf{x}_0 \in \alpha B_{\mathbb{R}^N} : \delta \leq \inf_{\mathbf{c} \in C} \langle \mathbf{x}_0, \mathbf{c} \rangle.$$

\Updownarrow

$$\exists \alpha > 0 : \delta \leq \max_{\mathbf{x} \in \alpha B_{\mathbb{R}^N}} \inf_{\mathbf{c} \in C} \langle \mathbf{x}, \mathbf{c} \rangle.$$

Llamamos $X := \alpha B_{\mathbb{R}^N}$ que es compacto y convexo, $Y := C$ convexo y f a la función real y continua con valores en $X \times Y$ definida como

$$f(\mathbf{x}, \mathbf{y}) := \langle \mathbf{x}, \mathbf{y} \rangle$$

(al ser f continua, en particular es superiormente semicontinua). Aplicamos el teorema minimax, teorema 7.1, y obtenemos que probar la última desigualdad es equivalente a probar que

$$\exists \alpha > 0 : \delta \leq \inf_{\mathbf{c} \in C} \max_{\mathbf{x} \in \alpha B_{\mathbb{R}^N}} \langle \mathbf{x}_0, \mathbf{c} \rangle.$$

Tenemos que $\max_{\mathbf{x} \in \alpha B_{\mathbb{R}^N}} \langle \mathbf{x}, \mathbf{c} \rangle = \alpha \|\mathbf{c}\|$ por el corolario 6.1. Así, debemos demostrar que

$$\exists \alpha > 0 : \delta \leq \alpha \inf_{\mathbf{c} \in C} \|\mathbf{c}\|.$$

Pero esta desigualdad es cierta tomando, por ejemplo, $\alpha = 1$. ■

Ahora, vamos a hacer una generalización de este resultado.

Teorema 7.3. Sean $N \in \mathbb{N}$ y A, B subconjuntos convexos de \mathbb{R}^N tal que A es cerrado, B es compacto y $A \cap B = \emptyset$. Entonces existe $\mathbf{x}_0 \in \mathbb{R}^N$ tal que

$$\sup_{\mathbf{a} \in A} \langle \mathbf{x}_0, \mathbf{a} \rangle < \inf_{\mathbf{b} \in B} \langle \mathbf{x}_0, \mathbf{b} \rangle.$$

*Demuestra*ción. En primer lugar, veamos que $\text{dist}(A, B) > 0$ donde la distancia viene dada por

$$\text{dist}(A, B) = \inf\{\|\mathbf{a} - \mathbf{b}\| : \mathbf{a} \in A, \mathbf{b} \in B\}.$$

Para ello, razonemos por reducción al absurdo. Suponemos $\text{dist}(A, B) = 0$, entonces existe una sucesión $\{\mathbf{u}_n\}_{n \in \mathbb{N}} \subset A - B$ tal que $\{\mathbf{u}_n\}_{n \in \mathbb{N}} \rightarrow 0$. Para $n \in \mathbb{N}$ tenemos que $\mathbf{u}_n = \mathbf{a}_n - \mathbf{b}_n$ con $\mathbf{a}_n \in A$ y $\mathbf{b}_n \in B$. De este modo obtenemos las sucesiones $\{\mathbf{a}_n\}_{n \in \mathbb{N}} \subset A$ y $\{\mathbf{b}_n\}_{n \in \mathbb{N}} \subset B$. Como B es compacto, existe una sucesión parcial convergente, es decir, existe $\sigma : \mathbb{N} \rightarrow \mathbb{N}$ estrictamente creciente tal que $\{\mathbf{b}_{\sigma(n)}\}_{n \in \mathbb{N}} \rightarrow \mathbf{b}$ con $\mathbf{b} \in B$. Así,

$$\|\mathbf{a}_{\sigma(n)} - \mathbf{b}\| = \|\mathbf{a}_{\sigma(n)} - \mathbf{b}_{\sigma(n)} + \mathbf{b}_{\sigma(n)} - \mathbf{b}\| \leq \|\mathbf{a}_{\sigma(n)} - \mathbf{b}_{\sigma(n)}\| + \|\mathbf{b}_{\sigma(n)} - \mathbf{b}\|.$$

Entonces tenemos que $\|\mathbf{a}_{\sigma(n)} - \mathbf{b}\| \rightarrow 0$ ya que $\{\mathbf{b}_{\sigma(n)}\}_{n \in \mathbb{N}} \rightarrow \mathbf{b}$ y como se da $\|\mathbf{a}_n - \mathbf{b}_n\| \rightarrow 0$ también se cumple que $\|\mathbf{a}_{\sigma(n)} - \mathbf{b}_{\sigma(n)}\| \rightarrow 0$. Llegamos a que $\{\mathbf{a}_{\sigma(n)}\}_{n \in \mathbb{N}} \rightarrow \mathbf{b}$. Al ser A cerrado se debe cumplir que $\mathbf{b} \in A$ lo cual es imposible ya que $A \cap B = \emptyset$.

Ahora, aplicamos el teorema 7.2 a $C := B - A$. Notar que C es convexo por serlo A y B . Obtenemos entonces que existe $\mathbf{x}_0 \in \mathbb{R}^N$ tal que si $\mathbf{c} \in C$ se cumple que:

$$\delta = \inf_{\mathbf{c} \in C} \|\mathbf{c}\| \leq \langle \mathbf{x}_0, \mathbf{c} \rangle.$$

Por la definición de C , se tiene que:

$$\delta = \inf\{\|\mathbf{b} - \mathbf{a}\| : \mathbf{a} \in A, \mathbf{b} \in B\} = \text{dist}(A, B) > 0.$$

Del mismo modo, $\mathbf{c} = \mathbf{b} - \mathbf{a}$ para todo $\mathbf{c} \in C$ con $\mathbf{a} \in A$ y $\mathbf{b} \in B$. Así,

$$\begin{aligned} \exists \mathbf{x}_0 \in \mathbb{R}^N : \mathbf{a} \in A, \mathbf{b} \in B \implies 0 < \delta \leq \langle \mathbf{x}_0, \mathbf{b} - \mathbf{a} \rangle = \langle \mathbf{x}_0, \mathbf{b} \rangle - \langle \mathbf{x}_0, \mathbf{a} \rangle. \\ \Updownarrow \\ \exists \mathbf{x}_0 \in \mathbb{R}^N : \mathbf{a} \in A, \mathbf{b} \in B \implies \langle \mathbf{x}_0, \mathbf{a} \rangle + \delta \leq \langle \mathbf{x}_0, \mathbf{b} \rangle, \text{ con } \delta > 0. \\ \Updownarrow \\ \exists \mathbf{x}_0 \in \mathbb{R}^N \implies \sup_{\mathbf{a} \in A} \langle \mathbf{x}_0, \mathbf{a} \rangle + \delta \leq \inf_{\mathbf{b} \in B} \langle \mathbf{x}_0, \mathbf{b} \rangle, \text{ con } \delta > 0. \\ \Updownarrow \\ \exists \mathbf{x}_0 \in \mathbb{R}^N : \sup_{\mathbf{a} \in A} \langle \mathbf{x}_0, \mathbf{a} \rangle < \inf_{\mathbf{b} \in B} \langle \mathbf{x}_0, \mathbf{b} \rangle. \end{aligned}$$

Por ello, queda probado el teorema. ■

A partir de este resultado, llegamos al siguiente corolario que será necesario en la parte final del trabajo.

Corolario 7.1. Sean $N \in \mathbb{N}$, $L \leq \mathbb{R}^N$ subespacio vectorial y $K \subset \mathbb{R}^N$ un subconjunto compacto y convexo tales que $L \cap K = \emptyset$. Entonces, $\exists \mathbf{x}_0 \in \mathbb{R}^N$ tal que si $\mathbf{y} \in L$, entonces $\langle \mathbf{x}_0, \mathbf{y} \rangle = 0$ y

$$0 < \inf_{\mathbf{x} \in K} \langle \mathbf{x}_0, \mathbf{x} \rangle.$$

Demostación. Como $L \leq \mathbb{R}^N$ entonces L es convexo y además, como \mathbb{R}^N es finito dimensional, también es cerrado. Esto último se debe a que L es de la forma

$$L = \left\{ (x_1, \dots, x_N) \in \mathbb{R}^N : \begin{array}{c} a_{11}x_1 + \dots + a_{1N}x_N = 0 \\ \vdots \\ a_{N1}x_1 + \dots + a_{NN}x_N = 0 \end{array} \right\}$$

Para cada una de las ecuaciones anteriores definimos las funciones continuas para cada $i = 1, \dots, N$ como

$$\begin{aligned} f_i : \mathbb{R}^N &\longrightarrow \mathbb{R} \\ \mathbf{x} &\longmapsto f(\mathbf{x}) = a_{i1}x_1 + \dots + a_{iN}x_N. \end{aligned}$$

De este modo, el conjunto de los puntos que cumplen cada ecuación se puede escribir como $f_i^{-1}(\{0\})$, que es cerrado al ser la imagen inversa de un cerrado por una función continua. Por lo tanto, L es la intersección finita de cerrados y por ello cerrado. Estamos entonces bajo las condiciones del teorema 7.3, que nos aporta la existencia de un $\mathbf{x}_0 \in \mathbb{R}^N$ que cumple:

$$\sup_{\mathbf{y} \in L} \langle \mathbf{x}_0, \mathbf{y} \rangle < \inf_{\mathbf{x} \in K} \langle \mathbf{x}_0, \mathbf{x} \rangle. \quad (7.4)$$

Tomamos ahora $\mathbf{y} \in L$ con $\langle \mathbf{x}_0, \mathbf{y} \rangle \neq 0$ (si $\langle \mathbf{x}_0, \mathbf{y} \rangle = 0$ no hay nada que probar) y $\rho > 0$. Así,

$$\rho \langle \mathbf{x}_0, \mathbf{y} \rangle = \langle \mathbf{x}_0, \rho \mathbf{y} \rangle < \inf_{\mathbf{x} \in K} \langle \mathbf{x}_0, \mathbf{x} \rangle.$$

Como $\inf_{\mathbf{x} \in K} \langle \mathbf{x}_0, \mathbf{x} \rangle \in \mathbb{R}$, la arbitrariedad de $\rho > 0$ obliga a que $\langle \mathbf{x}_0, \mathbf{y} \rangle \leq 0$ ya que si no fuese así \mathbb{R} , estaría acotado. Como $L \leq \mathbb{R}^N$, cambiamos \mathbf{y} por $-\mathbf{y}$ y obtenemos que $-\langle \mathbf{x}_0, \mathbf{y} \rangle \leq 0$, por lo que $\langle \mathbf{x}_0, \mathbf{y} \rangle = 0$. Finalmente, el hecho de que

$$0 < \inf_{\mathbf{x} \in K} \langle \mathbf{x}_0, \mathbf{x} \rangle$$

se deduce de (7.4). ■

El teorema de separación anterior podemos escribirlo no solo en \mathbb{R}^N sino que se puede generalizar a cualquier espacio normado. A continuación, exponemos otro teorema de separación válido solo en el contexto finito dimensional, con tesis e hipótesis más débiles. Esta vez, resulta del teorema de la alternativa de Gordan.

Teorema 7.4. *Sea $N \in \mathbb{N}$ y A y B dos subconjuntos convexos y disjuntos de \mathbb{R}^N . Entonces existe $\mathbf{x}_0 \in \mathbb{R}^N \setminus \{0\}$ tal que*

$$\sup_{\mathbf{a} \in A} \langle \mathbf{x}_0, \mathbf{a} \rangle \leq \inf_{\mathbf{b} \in B} \langle \mathbf{x}_0, \mathbf{b} \rangle.$$

*Demuestra*ción. Al igual que en el teorema anterior, podemos reducir la prueba al caso en que C es un subconjunto de \mathbb{R}^N convexo de forma que $0 \notin C$, demostrando que:

$$\exists \mathbf{x}_0 \in \mathbb{R}^N \setminus \{0\} : \sup_{\mathbf{c} \in C} \langle \mathbf{x}_0, \mathbf{c} \rangle \leq 0,$$

equivalentemente,

$$\exists \mathbf{x}_0 \in S_{\mathbb{R}^N} \setminus \{0\} : \sup_{\mathbf{c} \in C} \langle \mathbf{x}_0, \mathbf{c} \rangle \leq 0.$$

Pero esta afirmación no es más que

$$\bigcap_{\mathbf{c} \in C} \{\mathbf{x} \in S_{\mathbb{R}^N} : \langle \mathbf{x}, \mathbf{c} \rangle \leq 0\} \neq \emptyset.$$

Usamos que $S_{\mathbb{R}^N}$ es compacta y por ello tenemos, gracias a la propiedad de intersección finita, que esto equivale a

$$\emptyset \neq C_0 \subset C \text{ finito} \implies \bigcap_{\mathbf{c} \in C_0} \{\mathbf{x} \in S_{\mathbb{R}^N} : \langle \mathbf{x}, \mathbf{c} \rangle \leq 0\} \neq \emptyset.$$

⇓

$$M \in \mathbb{N}, \{\mathbf{c}_1, \dots, \mathbf{c}_M\} \subset C \implies \exists \mathbf{x}_0 \in S_{\mathbb{R}^N} : \max_{i=1, \dots, M} \langle \mathbf{x}_0, \mathbf{c}_i \rangle \leq 0.$$

Esta condición se verifica, ya que si $M \geq 1$ y $\{\mathbf{c}_1, \dots, \mathbf{c}_M\} \subset C$, entonces, por ser C convexo se cumple que $\text{co}\{\mathbf{c}_1, \dots, \mathbf{c}_M\} \subset C$ y como $0 \notin C$, entonces tenemos que $0 \notin \text{co}\{\mathbf{c}_1, \dots, \mathbf{c}_M\}$. Por tanto, la versión clásica del Teorema de Gordan, teorema 6.3, garantiza

$$\exists \mathbf{z}_0 \in \mathbb{R}^N : \max_{i=1, \dots, M} \langle \mathbf{z}_0, \mathbf{c}_i \rangle < 0.$$

En particular, $\mathbf{z}_0 \neq 0$ y tomando $\mathbf{x}_0 = \frac{\mathbf{z}_0}{\|\mathbf{z}_0\|}$ queda probado el enunciado. ■

Destacamos ahora lo siguiente:

Observación 7.1. *El Teorema de separación 7.3 implica el Teorema de Gordan, teorema 6.3.*

En efecto, dados $\{\mathbf{x}_1, \dots, \mathbf{x}_M\} \subset \mathbb{R}^N$ con $M, N \in \mathbb{N}$ del enunciado de la alternativa de Gordan planteamos las siguientes alternativas excluyentes:

1. Si $0 \in \text{co}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ entonces es claro que se cumple la alternativa i*).

2. Si $0 \notin \text{co}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ entonces $\{0\} \cap \text{co}\{\mathbf{x}_1, \dots, \mathbf{x}_M\} = \emptyset$. Al ser ambos conjuntos son compactos usamos el teorema de separación y obtenemos que:

$$\exists \mathbf{y} \in \mathbb{R}^M : \sup_{x \in \text{co}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}} \langle \mathbf{y}, \mathbf{x} \rangle < 0,$$

luego, en particular ($\{\mathbf{x}_1, \dots, \mathbf{x}_M\} \subset \text{co}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$),

$$\exists \mathbf{y} \in \mathbb{R}^M : \max_{i=1, \dots, M} \langle \mathbf{y}, \mathbf{x}_i \rangle < 0.$$

■

Para finalizar este capítulo, recogemos las equivalencias que se han obtenido sobre diferentes resultados:

Teorema de Gordan clásico (teorema 6.3)

\Updownarrow

Teorema de Gordan convexo (teorema 6.4)

\Updownarrow

Teorema de separación de convexos en \mathbb{R}^N (teorema 7.3).

Optimización

Si en el capítulo anterior dábamos las primeras aplicaciones del teorema de Gordan en forma de teorema minimax y, consecuentemente, de teoremas de separación, ahora usamos los teoremas de la alternativa en su contexto original: la optimización. Tras presentar un nuevo resultado de la alternativa, el lema de Farkas, abordamos el estudio de dos problemas clásicos: uno de dualidad en programación lineal; otro de paso de un problema de optimización con restricciones (en este caso de tipo desigualdad) a otro sin restricciones, vía los teoremas de Fritz-John y Karush-Kuhn-Tucker. Para el primero de ellos usamos el de Farkas y para el segundo el teorema de Gordan en su versión clásica.

8.1. Lema de Farkas

En el capítulo anterior hemos visto como uno de los teoremas de separación implica el único teorema de la alternativa que hemos visto hasta el momento. Ahora, vamos a deducir de manera parecida otro teorema de la alternativa. Antes exponemos la siguiente definición.

Definición 8.1. Dados $M, N \in \mathbb{N}$ y $\{\mathbf{x}_1, \dots, \mathbf{x}_M\} \subset \mathbb{R}^N$, llamamos cono generado por $\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ al conjunto de \mathbb{R}^N dado por:

$$\text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\} := \left\{ \sum_{j=1}^N \mu_j \mathbf{x}_j : \mu_1, \dots, \mu_N \geq 0 \right\}.$$

Veamos que $\text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ es un subconjunto convexo y cerrado de \mathbb{R}^N :

- Convexo: tenemos que comprobar que dados $\mathbf{t}, \mathbf{s} \in \text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ y $\lambda \in [0, 1]$ entonces $\lambda \mathbf{t} + (1 - \lambda) \mathbf{s} \in \text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$. En efecto:

$$\begin{aligned} \lambda \mathbf{t} + (1 - \lambda) \mathbf{s} &= \lambda \sum_{j=1}^N \mu_j^t \mathbf{x}_j + (1 - \lambda) \sum_{j=1}^N \mu_j^s \mathbf{x}_j \\ &= \sum_{j=1}^N [\lambda \mu_j^t \mathbf{x}_j + (1 - \lambda) \mu_j^s \mathbf{x}_j] \\ &= \sum_{j=1}^N [\lambda \mu_j^t + (1 - \lambda) \mu_j^s] \mathbf{x}_j. \end{aligned}$$

Como μ_j^t y μ_j^s son no negativos, entonces $\lambda\mu_j^t + (1 - \lambda)\mu_j^s$ también es una cantidad no negativa para $j = 1, \dots, M$. Así, podemos concluir que $\lambda\mathbf{t} + (1 - \lambda)\mathbf{s} \in \text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ y por tanto es un subconjunto convexo.

- Cerrado: sea $\{\mathbf{t}_n\}_{n \in \mathbb{N}}$ una sucesión de $\text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ y sea $\mathbf{t}_0 \in \mathbb{R}^N$ tal que $\{\mathbf{t}_n\}_{n \in \mathbb{N}} \rightarrow \mathbf{t}_0$. Si notamos $\mathbf{t}_n = \sum_{j=1}^N \mu_j^n \mathbf{x}_j$, entonces:

$$\{\mathbf{t}_n\}_{n \in \mathbb{N}} = \{\sum_{j=1}^N \mu_j^n \mathbf{x}_j\}_{n \in \mathbb{N}} \rightarrow \sum_{j=1}^N \mu_j^0 \mathbf{x}_j = \mathbf{t}_0.$$

Como para cada \mathbf{t}_n cumple que $\mu_j^n \geq 0$ con $j = 1, \dots, N$ podemos asegurar que $\mu_j^0 \geq 0$ con $j = 1, \dots, N$. Hemos demostrado que \mathbf{t}_0 se expresa como combinación de $\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ con coeficientes no negativos. Por lo tanto, $\mathbf{t}_0 \in \text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ y por consiguiente es cerrado.

Enunciamos ahora otro de los teoremas de la alternativa más conocidos.

Lema 8.1 (Farkas). *Sean $M, N \in \mathbb{N}$ y $\{\mathbf{x}_1, \dots, \mathbf{x}_M\} \subset \mathbb{R}^N$ y $\mathbf{b} \in \mathbb{R}^N$. Entonces una, y solo una, de las siguientes alternativas se cumple:*

i') $\exists \mu_1, \dots, \mu_M \geq 0$ tal que $\mathbf{b} = \sum_{j=1}^M \mu_j \mathbf{x}_j$.

ii') $\exists \mathbf{z}_0 \in \mathbb{R}^N$ que cumple que:

1. $\max_{j=1, \dots, M} \langle \mathbf{z}_0, \mathbf{x}_j \rangle \leq 0$ y
2. $\langle \mathbf{z}_0, \mathbf{b} \rangle > 0$.

Demostración. Planteamos las siguientes alternativas, que obviamente son excluyentes, y que implican las de la tesis del lema:

a) $\mathbf{b} \in \text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$. Estamos en el caso i') ya que:

$$\mathbf{b} \in \left\{ \sum_{j=1}^N \mu_j \mathbf{x}_j : \mu_1, \dots, \mu_N \geq 0 \right\}.$$

b) $\mathbf{b} \notin \text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$. Por su parte, esta alternativa equivale a ii'). En efecto:

ii') \Rightarrow b) Razonamos por reducción al absurdo. Suponemos por ello que el elemento $\mathbf{b} \in \text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$, entonces, podemos expresar $\mathbf{b} = \sum_{j=1}^N \mu_j \mathbf{x}_j$ donde $\mu_1, \dots, \mu_N \geq 0$. Como se da ii'), en particular se cumple 1 y obtenemos, para el \mathbf{z}_0 cuya existencia se garantiza,

$$\begin{aligned} \langle \mathbf{z}_0, \mathbf{b} \rangle &= \langle \mathbf{z}_0, \sum_{j=1}^N \mu_j \mathbf{x}_j \rangle \\ &= \sum_{j=1}^N \mu_j \langle \mathbf{z}_0, \mathbf{x}_j \rangle \leq 0. \end{aligned}$$

Por otro lado, por 2 se tiene que $\langle \mathbf{z}_0, b \rangle > 0$. Así, obtenemos que

$$\langle \mathbf{z}_0, b \rangle \leq 0 < \langle \mathbf{z}_0, b \rangle,$$

lo cual es imposible.

- b) $\Rightarrow ii'$) Si $b \notin \text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ aplicamos el teorema de separación 7.3 a los conjuntos $\{b\}$ que es compacto y convexo y a $\text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ que es cerrado y convexo. Obtenemos que:

$$\exists \mathbf{z}_0 \in \mathbb{R}^N : \sup_{a \in \text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}} \langle \mathbf{z}_0, a \rangle < \langle \mathbf{z}_0, b \rangle.$$

Por un lado, es obvio que $0 \in \text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ y por ello:

$$0 = \langle \mathbf{z}_0, 0 \rangle \leq \sup_{a \in \text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}} \langle \mathbf{z}_0, a \rangle < \langle \mathbf{z}_0, b \rangle.$$

Hemos obtenido por tanto que es cierto 2. Para probar 1, fijamos $a_0 \in \text{co}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$. Entonces, dado $\rho > 0$,

$$\rho \langle \mathbf{z}_0, a_0 \rangle = \langle \mathbf{z}_0, \rho a_0 \rangle \leq \sup_{a \in \text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}} \langle \mathbf{z}_0, a \rangle.$$

Llegamos a que el conjunto $\{\rho \langle \mathbf{z}_0, a_0 \rangle : \rho > 0\}$ está acotado y eso solo es posible si $\langle \mathbf{z}_0, a_0 \rangle \leq 0$. La arbitrariedad de a_0 nos aporta que

$$\sup_{a \in \text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}} \langle \mathbf{z}_0, a \rangle = 0$$

de donde

$$\max_{j=1, \dots, M} \langle \mathbf{z}_0, \mathbf{x}_j \rangle \leq \sup_{a \in \text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}} \langle \mathbf{z}_0, a \rangle \leq 0$$

ya que obviamente $\{\mathbf{x}_1, \dots, \mathbf{x}_M\} \subset \text{cone}\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$. En particular, hemos probado 1. ■

8.2. Dualidad en programación lineal

Gracias al lema de Farkas podemos probar resultados como el teorema de dualidad en programación lineal que enunciamos a continuación. La prueba se recoge en [1].

Teorema 8.1. (*Teorema de dualidad en programación lineal*) Dados $N, M \in \mathbb{R}$ tomamos la matriz $A \in \mathbb{R}^{N \times M}$ y los vectores $\mathbf{b} \in \mathbb{R}^M$ y $\mathbf{c} \in \mathbb{R}^N$. Consideramos el problema de optimización primal:

$$\left. \begin{aligned} p := \inf_{\mathbf{x}} \mathbf{c}^T \mathbf{x} \\ s.a \quad A^T \mathbf{x} \leq \mathbf{b} \\ \mathbf{x} \in \mathbb{R}^N \end{aligned} \right\}, \quad (P)$$

donde $A^T \mathbf{x} \leq \mathbf{b}$ nos aporta un total de M restricciones. Sea el problema dual:

$$\left. \begin{array}{l} d := \sup_{\mathbf{y}} -\mathbf{b}^T \mathbf{y} \\ s.a \quad A\mathbf{y} = -\mathbf{c} \\ \mathbf{y} \in \mathbb{R}_+^M \end{array} \right\}. \quad (D)$$

Entonces, la solución del problema primal y el dual coincide, o lo que es lo mismo, $p = d$.

*Demuestra*ción. En primer lugar, veamos que $p \geq d$. Para ello, tomamos $\mathbf{x} \in \mathbb{R}^N$ e $\mathbf{y} \in \mathbb{R}^M$ factibles, es decir, cumplen las restricciones de sus respectivos problemas. Así:

$$\begin{aligned} & \left. \begin{array}{l} A^T \mathbf{x} - \mathbf{b} \leq 0 \\ \mathbf{y} \geq 0 \end{array} \right\} \\ & \Downarrow \\ & (A^T \mathbf{x} - \mathbf{b})^T \mathbf{y} = \mathbf{x}^T A\mathbf{y} - \mathbf{b}^T \mathbf{y} \leq 0 \\ & \Updownarrow \\ & -\mathbf{b}^T \mathbf{y} \leq \mathbf{c}^T \mathbf{x}, \end{aligned}$$

donde la última desigualdad se debe a que $A\mathbf{y} = -\mathbf{c}$. Tomando supremos en la izquierda e ínfimos en la derecha llegamos a $d \leq p$.

Supongamos ahora que $p \in \mathbb{R}$ ya que en caso contrario habríamos terminado. Consideramos el problema homogeneizado,

$$\left. \begin{array}{l} A^T \mathbf{x} - z\mathbf{b} \leq 0 \\ -\mathbf{c}^T \mathbf{x} + zp > 0 \\ \mathbf{x} \in \mathbb{R}^N, z \geq 0 \end{array} \right\}.$$

Este problema no tiene solución ya que en caso contrario, si $z > 0$, entonces $A^T \frac{\mathbf{x}}{z} < \mathbf{b}$ y, por la definición de p , $\mathbf{c}^T \frac{\mathbf{x}}{z} \geq p$ si, y solo si, $zp \leq \mathbf{c}^T \mathbf{x}$, lo que no es posible, pues debe ser $zp > \mathbf{c}^T \mathbf{x}$. En el caso $z = 0$, el problema queda

$$\left. \begin{array}{l} A^T \mathbf{x} \leq 0 \\ \mathbf{c}^T \mathbf{x} < 0 \end{array} \right\}. \quad (8.1)$$

Sea $\mathbf{u} \in \mathbb{R}^N$ un punto factible del problema primal, esto es,

$$A^T \mathbf{u} \leq \mathbf{b}.$$

Entonces, dado $\rho > 0$,

$$\begin{aligned} A^T(\rho \mathbf{x} + \mathbf{u}) &= \rho A^T \mathbf{x} + A^T \mathbf{u} \\ &\leq \mathbf{b} \end{aligned}$$

ya que $A^T \mathbf{x} \leq 0$, $\rho > 0$ y $A^T \mathbf{u} \leq \mathbf{b}$. Usando la definición de p ,

$$\begin{aligned} p &\leq \mathbf{c}^T(\rho \mathbf{x} + \mathbf{u}) \\ &= \rho \mathbf{c}^T \mathbf{x} + \rho \mathbf{u}. \end{aligned}$$

Como $p \in \mathbb{R}$, $\mathbf{c}^T \mathbf{x} < 0$ y $\rho > 0$ es arbitrario, llegamos a una contradicción (tómese límite cuando $\rho \rightarrow +\infty$). Aplicamos el lema de Farkas al problema homogeneizado, y al no tener el sistema solución observamos que no se puede dar la alternativa ii'). Entonces, tenemos que existen escalares $\mu_1, \dots, \mu_M, \beta \geq 0$ que cumplen

$$\sum_{j=1}^M \mu_j (A_j, -b_j) + \beta(0, -1) = (-\mathbf{c}, p),$$

donde para cada $j = 1, \dots, M$, A_j denota la fila j -ésima de A^T . Reescribiendo lo

anterior, llamamos $\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_M \end{pmatrix}$ y obtenemos

$$\left. \begin{aligned} \sum_{j=1}^M \mu_j A_j &= A\boldsymbol{\mu} = -\mathbf{c} \\ \boldsymbol{\mu} &\in \mathbb{R}_+^M \\ -\mathbf{b}^T \boldsymbol{\mu} - \beta &= p \end{aligned} \right\}.$$

El vector $\boldsymbol{\mu}$ es una solución factible para el problema dual (D) con valor objetivo al menos p obteniendo entonces que $p \leq d$, quedando así probado el resultado. ■

8.3. Optimización con restricciones: Fritz John y Karush-Kuhn-Tucker

Concluimos este capítulo con un apartado dedicado a la optimización con restricciones. A pesar de tratarse de resultados no lineales, el teorema de Gordan clásico basta para establecer los resultados fundamentales. Al igual que en el apartado anterior, seguimos el texto [1]. En primer lugar, empezamos recordando la definición de derivada direccional.

Definición 8.2. Sea $D \subset \mathbb{R}^M$ con $M \in \mathbb{N}$ y sea la función $g : D \rightarrow \mathbb{R}$, definimos la derivada direccional de g en $\mathbf{x} \in D$ en la dirección del vector $\mathbf{d} \in \mathbb{R}^M$ como

$$g'(\mathbf{x}; \mathbf{d}) = \lim_{t \rightarrow 0} \frac{g(\mathbf{x} + t\mathbf{d}) - g(\mathbf{x})}{t}$$

siempre y cuando el límite exista. Diremos que g es diferenciable en el sentido de Gâteaux en \mathbf{x} si $g'(\mathbf{x}; \cdot) : \mathbb{R}^M \rightarrow \mathbb{R}$ es lineal y en ese caso escribimos $\nabla g(\mathbf{x}) = g'(\mathbf{x}; \cdot)$, es decir, $g'(\mathbf{x}; \mathbf{d}) = \langle \nabla g(\mathbf{x}), \mathbf{d} \rangle$ con $\mathbf{d} \in \mathbb{R}^M$.

Dentro del contexto de este trabajo, cuando decimos que una función es diferenciable nos referimos a que lo es en el sentido de Gâteaux. A estas funciones también las llamaremos Gâteaux diferenciables. Destacamos que este concepto de diferenciabilidad es más débil que el de Fréchet. De hecho, si una función g es diferenciable en $\mathbf{x}_0 \in D$ en el sentido de Fréchet, y notamos su derivada como $Dg(\mathbf{x}_0)$ entonces g también es diferenciable en el sentido de Gâteaux en \mathbf{x}_0 y además

$Dg(\mathbf{x}_0) = \nabla g(\mathbf{x}_0)$. El recíproco no es cierto tal y como mostramos en el siguiente ejemplo. Sea $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ definida como:

$$f(x, y) = \begin{cases} \frac{xy^3}{x^2 + y^2}, & \text{si } (x, y) \neq (0, 0) \\ 0, & \text{si } (x, y) = (0, 0) \end{cases}.$$

Sabemos que si una función es Fréchet diferenciable en x_0 , entonces es continua en x_0 . Como f no es continua en $(0, 0)$, podemos afirmar que no es Fréchet diferenciable en dicho punto. Sin embargo, sí es Gâteaux diferenciable en $(0, 0)$. Para $\mathbf{d} = (d_1, d_2) \in \mathbb{R}^2$,

$$\begin{aligned} f'((0, 0); (d_1, d_2)) &= \lim_{t \rightarrow 0} \frac{f((0, 0) + t(d_1, d_2)) - f(0, 0)}{t} \\ &= \lim_{t \rightarrow 0} \frac{\frac{td_1(td_2)^3}{(td_1)^2 + (td_2)^2} - 0}{t} \\ &= \lim_{t \rightarrow 0} \frac{t^4 d_1 d_2^3}{t^3 d_1^2 + t^3 d_2^2} \\ &= \lim_{t \rightarrow 0} \frac{t d_1 d_2^3}{d_1^2 + d_2^2} \\ &= 0. \end{aligned}$$

Como el límite existe y es lineal, la función es diferenciable en el sentido Gâteaux en $(0, 0)$.

A continuación, demostremos cómo se calcula la derivada de la función máximo de un número finito de funciones diferenciables, lo que nos será útil en posteriores resultados.

Proposición 8.1. Sean $D \subset \mathbb{R}^M$ ($M \in \mathbb{N}$), $\bar{\mathbf{x}}$ un punto del interior de D y sean $g_1, \dots, g_N : D \rightarrow \mathbb{R}$ funciones continuas y diferenciables en $\bar{\mathbf{x}}$ donde $N \in \mathbb{N}$. Definimos $g : D \rightarrow \mathbb{R}$ como

$$g(\mathbf{x}) := \max_{i=1, \dots, N} \{g_i(\mathbf{x})\}$$

y el conjunto de índices $K = \{i : g_i(\bar{\mathbf{x}}) = g(\bar{\mathbf{x}})\}$. Entonces, para toda dirección $\mathbf{d} \in \mathbb{R}^M$ la derivada direccional de g existe en todo \mathbb{R}^M y viene dada por la siguiente expresión:

$$g'(\bar{\mathbf{x}}; \mathbf{d}) = \max_{i \in K} \langle \nabla g_i(\bar{\mathbf{x}}), \mathbf{d} \rangle, \quad \mathbf{d} \in \mathbb{R}^M. \quad (8.2)$$

Demuestração. Podemos suponer sin pérdida de generalidad que $K = \{1, \dots, N\}$, por facilitar la notación. Para cada $i \in K$ tenemos la siguiente desigualdad:

$$\liminf_{t \rightarrow 0} \frac{g(\bar{\mathbf{x}} + t\mathbf{d}) - g(\bar{\mathbf{x}})}{t} \geq \liminf_{t \rightarrow 0} \frac{g_i(\bar{\mathbf{x}} + t\mathbf{d}) - g_i(\bar{\mathbf{x}})}{t} = \langle \nabla g_i(\bar{\mathbf{x}}), \mathbf{d} \rangle.$$

La primera desigualdad se deduce de la definición de g ya que es el máximo de las g_i e $i \in K$ para $i = 1, \dots, N$ y la segunda igualdad de que todas las g_i son diferenciables en $\bar{\mathbf{x}}$ y por tanto existe el límite de la definición de derivada direccional y coincide con el límite inferior. Por lo tanto:

$$\liminf_{t \rightarrow 0} \frac{g(\bar{\mathbf{x}} + t\mathbf{d}) - g(\bar{\mathbf{x}})}{t} \geq \max_{i=1,\dots,N} \langle \nabla g_i(\bar{\mathbf{x}}), \mathbf{d} \rangle.$$

Por otro lado, afirmamos que

$$\limsup_{t \rightarrow 0} \frac{g(\bar{\mathbf{x}} + t\mathbf{d}) - g(\bar{\mathbf{x}})}{t} \leq \max_{i=1,\dots,N} \langle \nabla g_i(\bar{\mathbf{x}}), \mathbf{d} \rangle.$$

De lo contrario, existirían una sucesión $\{t_n\} \rightarrow 0$ y $\varepsilon > 0$ que cumplirían:

$$\frac{g(\bar{\mathbf{x}} + t_n \mathbf{d}) - g(\bar{\mathbf{x}})}{t_n} \geq \max_{i=1,\dots,N} \langle \nabla g_i(\bar{\mathbf{x}}), \mathbf{d} \rangle + \varepsilon \quad \forall n \in \mathbb{N}.$$

Tomamos ahora una sucesión parcial $\{t_{\sigma(n)}\}_{n \in \mathbb{N}}$ con $\sigma : \mathbb{N} \rightarrow \mathbb{N}$ estrictamente creciente y $j \in K$ un índice fijo tal que para todo $k \in \{\sigma(n) : n \in \mathbb{N}\}$ se cumple que $g(\bar{\mathbf{x}} + t_k \mathbf{d}) = g_j(\bar{\mathbf{x}} + t_k \mathbf{d})$. Tomando límite obtenemos que:

$$\begin{aligned} \limsup_{t \rightarrow 0} \frac{g(\bar{\mathbf{x}} + t\mathbf{d}) - g(\bar{\mathbf{x}})}{t} &= \limsup_{t \rightarrow 0} \frac{g_j(\bar{\mathbf{x}} + t\mathbf{d}) - g_j(\bar{\mathbf{x}})}{t} \\ &= \langle \nabla g_j(\bar{\mathbf{x}}), \mathbf{d} \rangle \\ &\geq \max_{i=1,\dots,N} \langle \nabla g_i(\bar{\mathbf{x}}), \mathbf{d} \rangle + \varepsilon, \end{aligned}$$

lo cual, es imposible. Finalmente, hemos obtenido que:

$$\limsup_{t \rightarrow 0} \frac{g(\bar{\mathbf{x}} + t\mathbf{d}) - g(\bar{\mathbf{x}})}{t} \leq \max_{i=1,\dots,N} \langle \nabla g_i(\bar{\mathbf{x}}), \mathbf{d} \rangle \leq \liminf_{t \rightarrow 0} \frac{g(\bar{\mathbf{x}} + t\mathbf{d}) - g(\bar{\mathbf{x}})}{t}.$$

Como el límite inferior es siempre menor o igual que el superior concluimos que ambos coinciden y por lo tanto existe el límite y además:

$$\lim_{t \rightarrow 0} \frac{g(\bar{\mathbf{x}} + t\mathbf{d}) - g(\bar{\mathbf{x}})}{t} = g'(\bar{\mathbf{x}}; \mathbf{d}) = \max_{i=1,\dots,N} \langle \nabla g_i(\bar{\mathbf{x}}), \mathbf{d} \rangle.$$

■

Nuestro objetivo ahora es encontrar soluciones a problemas del siguiente tipo:

$$\left\{ \begin{array}{l} \inf_{\mathbf{x} \in D} f(\mathbf{x}) \\ \text{s.a } g_1(\mathbf{x}) \leq 0 \\ \vdots \\ g_N(\mathbf{x}) \leq 0 \end{array} \right. \quad (8.3)$$

donde $D \subset \mathbb{R}^M$, f es la función objetivo y las restricciones g_i con $i = 1, \dots, N$ son funciones reales definidas en D y continuas. Si un punto satisface todas las restricciones diremos que es *factible* y como consecuencia llamamos *región factible*

al conjunto de todos los puntos factibles. Para un punto factible $\bar{\mathbf{x}}$ definimos el *conjunto activo* como $I(\bar{\mathbf{x}}) = \{i : g_i(\bar{\mathbf{x}}) = 0\}$. El papel que juega este conjunto de índices no es trivial, apareciendo en los teoremas de Fritz John y Karush-Kuhn-Tucker como los únicos necesarios. Este trabajo ya se ha realizado en la proposición 8.1. Para este problema y asumiendo que $\bar{\mathbf{x}} \in D$, llamamos *vector de multiplicadores de Lagrange para $\bar{\mathbf{x}}$* a $\boldsymbol{\lambda} \in (\mathbb{R}^N)^+$ si $\bar{\mathbf{x}}$ es un punto crítico de:

$$L(\mathbf{x}; \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^N \lambda_i g_i(\mathbf{x}),$$

es decir, se cumple que f, g_1, \dots, g_N son diferenciables y:

$$\nabla f(\bar{\mathbf{x}}) + \sum_{i=1}^N \lambda_i \nabla g_i(\bar{\mathbf{x}}) = 0.$$

Además, $\lambda_i = 0$ si $i \notin I(\bar{\mathbf{x}})$, por tanto

$$\nabla f(\bar{\mathbf{x}}) + \sum_{i \in I(\bar{\mathbf{x}})} \lambda_i \nabla g_i(\bar{\mathbf{x}}) = 0.$$

También, se dice que un punto $\bar{\mathbf{x}} \in D$ es un *mínimo global* del problema 8.3 si es un punto factible y además $f(\bar{\mathbf{x}}) \leq f(\mathbf{x})$ para todo $\mathbf{x} \in D$ factible. Por otro lado, diremos que $\bar{\mathbf{x}} \in D$ es un *mínimo local* del problema 8.3 si es un punto factible y existe U entorno de $\bar{\mathbf{x}}$ tal que $f(\bar{\mathbf{x}}) \leq f(\mathbf{x})$ para todo $\mathbf{x} \in D \cap U$ factible. Los resultados que exponemos a continuación se centran en mínimos locales.

Teorema 8.2 (Teorema de Fritz John). *Supongamos que el problema (8.3) tiene un mínimo local en $\bar{\mathbf{x}} \in D$. Si las funciones f, g_i con $i \in I(\bar{\mathbf{x}})$ son diferenciables en $\bar{\mathbf{x}}$ entonces existen $\lambda_0, \lambda_i \in \mathbb{R}^+$ para $i \in I(\bar{\mathbf{x}})$, no todas cero, que satisfacen:*

$$\lambda_0 \nabla f(\bar{\mathbf{x}}) + \sum_{i \in I(\bar{\mathbf{x}})} \lambda_i \nabla g_i(\bar{\mathbf{x}}) = 0.$$

Demostración. Consideramos la función

$$g(\mathbf{x}) = \max\{f(\mathbf{x}) - f(\bar{\mathbf{x}}), g_i(\mathbf{x}) : i \in I(\bar{\mathbf{x}})\}.$$

definida para todo punto $\mathbf{x} \in D$ y perteneciente a la región factible del problema (8.3). Como $\bar{\mathbf{x}}$ es un mínimo local de dicho problema también lo es de g . Supongamos que no fuera así, es decir, existe U entorno de $\bar{\mathbf{x}}$ existe un punto $\mathbf{x}_0 \in D \cap U$ factible tal que $g(\mathbf{x}_0) < g(\bar{\mathbf{x}})$. Como $\bar{\mathbf{x}}$ es un mínimo local del problema se tiene que $f(\mathbf{x}_0) - f(\bar{\mathbf{x}}) > 0$. En ese caso, como \mathbf{x} es factible (y por ello cumple las restricciones), se tiene que

$$\begin{aligned} g(\mathbf{x}_0) &= \max\{f(\mathbf{x}_0) - f(\bar{\mathbf{x}}), g_i(\mathbf{x}_0) : i \in I(\bar{\mathbf{x}})\} \\ &= f(\mathbf{x}_0) - f(\bar{\mathbf{x}}) \\ &> 0. \end{aligned}$$

Esto es imposible ya que se tendría que $0 < g(\mathbf{x}_0) < g(\mathbf{x}) = 0$. Por la proposición 8.1 tenemos que para toda dirección $\mathbf{d} \in \mathbb{R}^M$ se cumple:

$$g'(\bar{\mathbf{x}}; \mathbf{d}) = \max\{\langle \nabla f(\bar{\mathbf{x}}), \mathbf{d} \rangle, \langle \nabla g_i(\bar{\mathbf{x}}), \mathbf{d} \rangle : i \in I(\bar{\mathbf{x}})\} \geq 0,$$

ya que si $g'(\bar{\mathbf{x}}; \mathbf{d}) < 0$, para todo $t > 0$ suficientemente pequeño tendríamos que $g(\bar{\mathbf{x}} + t\mathbf{d}) < g(\bar{\mathbf{x}})$ lo que contradice que g alcanza un mínimo local en $\bar{\mathbf{x}}$.

Por lo tanto, el sistema

$$\begin{cases} \langle \nabla f(\bar{\mathbf{x}}), \mathbf{d} \rangle < 0 \\ \langle \nabla g_i(\bar{\mathbf{x}}), \mathbf{d} \rangle < 0 \text{ con } i \in I(\bar{\mathbf{x}}) \end{cases}$$

no tiene solución (para ninguna dirección) ya que al menos uno es no negativo. Si aplicamos el Teorema de la Alternativa de Gordan en su versión clásica, teorema 6.3, vemos que solo se puede dar la alternativa i*) y en ese caso obtenemos que:

$$\exists \mathbf{t} = (t_0, \dots, t_R) \in \Delta_{R+1} \text{ tal que } 0 = t_0 \nabla f(\bar{\mathbf{x}}) + \sum_{i \in I(\bar{\mathbf{x}})} t_i \nabla g_i(\bar{\mathbf{x}}),$$

con R el cardinal del conjunto $I(\bar{\mathbf{x}})$. La demostración concluye llamando $\lambda_0 = t_0$ y $\lambda_i = t_i$ con $i \in I(\bar{\mathbf{x}})$. ■

El teorema de Fritz John nos aporta una gran desventaja y es que es posible que $\lambda_0 = 0$ por lo que la función objetivo es independiente de las restricciones y no influye en la información que se da. Por ello, necesitamos imponer algunas condiciones de regularidad extra. En esta situación diremos que se cumple la *condición de Mangasarian-Fromovitz* si existe una dirección $\mathbf{d}_0 \in \mathbb{R}^M$ que satisface que $\langle \nabla g_i(\bar{\mathbf{x}}), \mathbf{d}_0 \rangle < 0$ para todo índice $i \in I(\bar{\mathbf{x}})$. Enunciamos ahora otro teorema clásico que soluciona el problema que acabamos de comentar.

Teorema 8.3 (Karush-Kuhn-Tucker). *Supongamos que el problema (8.3) tiene un mínimo local en $\bar{\mathbf{x}} \in D$. Si las funciones f, g_i con $i \in I(\bar{\mathbf{x}})$ son diferenciables en $\bar{\mathbf{x}}$ y se cumple la condición de Mangasarian-Fromovitz entonces existe un vector de multiplicadores de Lagrange para $\bar{\mathbf{x}}$.*

Demostración. Del teorema 8.2 de Fritz John obtenemos que:

$$\exists \mathbf{t} = (t_0, \dots, t_R) \in \Delta_{R+1} \text{ tal que } 0 = t_0 \nabla f(\bar{\mathbf{x}}) + \sum_{i \in I(\bar{\mathbf{x}})} t_i \nabla g_i(\bar{\mathbf{x}})$$

con R el cardinal de $I(\bar{\mathbf{x}})$. Si multiplicamos escalarmente la igualdad por \mathbf{d}_0 (dirección del espacio vectorial que nos aporta la condición de Mangasarian-Fromovitz), obtenemos:

$$0 = t_0 \langle \nabla f(\bar{\mathbf{x}}), \mathbf{d}_0 \rangle + \sum_{i \in I(\bar{\mathbf{x}})} t_i \langle \nabla g_i(\bar{\mathbf{x}}), \mathbf{d}_0 \rangle.$$

Entonces $t_0 \neq 0$. Razonemos por reducción al absurdo. Si no fuese así, tendríamos que

$$0 = \sum_{i \in I(\bar{\mathbf{x}})} t_i \langle \nabla g_i(\bar{\mathbf{x}}), \mathbf{d}_0 \rangle.$$

Al tener $\mathbf{t} \in \Delta_{R+1}$ se cumple que todas sus componentes son no negativas y $\sum_{i \in I(\bar{\mathbf{x}})} t_i = 1$ (estamos suponiendo que $t_0 = 0$ por lo que no influye en la suma de la definición de Δ_{R+1}) por lo que algún término es distinto de 0. Tenemos garantizado que $\langle g_i(\bar{\mathbf{x}}), \mathbf{d}_0 \rangle < 0 \quad \forall i \in I(\bar{\mathbf{x}})$. Así tendríamos que:

$$0 = \sum_{i \in I(\bar{\mathbf{x}})} t_i \langle \nabla g_i(\bar{\mathbf{x}}), \mathbf{d}_0 \rangle < 0,$$

lo cual es imposible. Por ello, concluimos que $t_0 \neq 0$. La demostración finaliza tomando $\lambda_i = t_i/t_0$ para $i \in I(\bar{\mathbf{x}})$. ■

La condición de Mangasarian-Fromovitz no es prescindible en el teorema de Karush-Kuhn-Tucker, tal y como pone de manifiesto este sencillo ejemplo.

$$\begin{cases} f(x, y) = 2x \\ \text{s.a } g_1(x, y) = 2y - 5x^3 \\ g_2(x, y) = -y \end{cases} .$$

Es claro que el mínimo se alcanza en $\bar{\mathbf{x}} = (0, 0)$ y tenemos que $\nabla f(0, 0) = (2, 0)$, $\nabla g_1(0, 0) = (0, 2)$, $\nabla g_2(0, 0) = (0, -1)$ e $I(\bar{\mathbf{x}}) = \{1, 2\}$. La condición de Mangasarian-Fromovitz no se cumple. Si existiese una dirección $d = (d_1, d_2) \in \mathbb{R}^2$ que la cumpliese se deberían satisfacer las siguientes condiciones simultáneamente

$$\begin{cases} \langle \nabla g_1(0, 0), (d_1, d_2) \rangle < 0 \iff \langle (0, 2), (d_1, d_2) \rangle < 0 \iff 2d_2 < 0 \\ \langle \nabla g_2(0, 0), (d_1, d_2) \rangle < 0 \iff \langle (0, -1), (d_1, d_2) \rangle < 0 \iff d_2 > 0 \end{cases}$$

lo que es imposible. Además, si $\lambda, \mu \geq 0$ fuesen multiplicadores de Lagrange para $(0, 0)$, entonces

$$\nabla f(0, 0) + \lambda \nabla g_1(0, 0) + \mu \nabla g_2(0, 0) = (0, 0)$$

\Updownarrow

$$(2, 2\lambda - \mu) = (0, 0),$$

que no se puede dar.

Finalmente, notamos que los teoremas de Fritz John y de Karush-Kuhn-Tucker que acabamos de demostrar también se pueden considerar consecuencia del lema de Farkas. Destacar también que en el caso convexo e imponiendo ciertas condiciones de regularidad podemos probar resultados análogos sin hipótesis de diferenciabilidad [1].

Valoración de activos financieros

En este último capítulo del trabajo no centramos en las *matemáticas financieras*, más concretamente, nuestro objetivo es obtener el precio de un activo financiero determinado. Para ello, será esencial el teorema de separación de convexos. De este modo, primero introduciremos los conceptos financieros que son necesarios para la comprensión de nuestro problema. Destacan el denominado *principio de arbitraje* y las *opciones compra*, que son los activos que queremos valorar. En la segunda sección, estableceremos el modelo matemático de nuestro mercado siguiendo el propuesto por [2]. Al acabar la misma, y gracias al teorema de separación, demostraremos el “primer teorema fundamental de asignación de precios”. Finalmente, se ha incluido una sección donde aplicaremos este último resultado para valorar las *opciones de compra* e ilustraremos cómo varía ese valor en función de sus parámetros.

9.1. Preliminares financieros

Nos introducimos ahora en el mundo de las denominadas *matemáticas financieras*. En las secciones anteriores hemos expuesto todas las herramientas matemáticas necesarias para el trabajo y en esta vamos a explicar los conceptos económicos o financieros. Una vez se haya completado este apartado estaremos dispuestos a enunciar y probar la aplicación de los teoremas de la alternativa culmen de este trabajo que nos servirá para la valoración de activos financieros en mercados finitos.

Nos preguntamos entonces: “¿qué es un *activo*?”. Un activo o título de valor se define como un recurso con valor que alguien posee con el fin de obtener un beneficio en el futuro. Podemos diferenciar entre activos seguros, como depósitos en el banco o bonos del estado, y activos con riesgo, como las acciones. Uno de los conceptos más importantes que tenemos es nuestro modelo del mercado financiero es el *principio de no arbitraje*. Este principio intuitivamente nos dice que no podemos obtener beneficio si no corremos algún riesgo. Puede resultar un poco confuso ya que acabamos de diferenciar entre activos seguros y con riesgo. Esto se debe a que un activo, aunque se llame seguro, no significa que tenga el beneficio asegurado. Por ejemplo, si tenemos nuestro dinero en una cuenta bancaria es posible que el banco quiebre y perdamos todos nuestros ahorros. Así, en la realidad estas oportunidades,

llamadas de arbitraje, son muy raras y cuando se dan suponen una ganancia muy pequeña en comparación con la cantidad de dinero que se está manejando globalmente.

Cuando nos movemos en el ámbito financiero también debemos de tener en cuenta el *valor del dinero*. Nuestro dinero se va devaluando con el paso del tiempo debido a muchas causas como, por ejemplo, la falta de riesgo absoluto antes mencionadas. Es preferible obtener una cantidad de dinero en este momento que en el futuro ya que no tendremos el mismo poder adquisitivo. Por eso, cuando alguien tiene una deuda debe devolver el dinero con cierto interés porque, de otro modo, sería injusto para la persona que presta el dinero. Además, dicho interés es en cierta medida una estimación, ya que no se puede saber con seguridad, el precio en un futuro. Lo mismo pasa con los activos con riesgo, solo sabemos el precio que tienen en este momento. Por tanto, es posible que el precio en el futuro sea mayor que el actual o menor. Matemáticamente, podemos representar su valor mediante una variable aleatoria que generalmente mide el retorno en vez del precio aunque se puede pasar de una a otra fácilmente. Podemos suponer una situación con gran número de posibles retornos intentando abarcar la mayor cantidad de situaciones que nos podríamos encontrar. Sin embargo, el caso binomial, en el que solo existen dos posibilidades, es el más habitual ya que es lo suficientemente simple de manejar y además refleja bastantes situaciones del mercado financiero real. Este modelo también supone que en cada paso el retorno tiene el mismo comportamiento. El retorno para un instante se suele definir como:

$$R_t = \frac{S_t}{S_{t-1}},$$

donde S_t es la variable aleatoria por la que se rige el precio del activo. También se puede definir el retorno como $K_t = R_t - 1 = (S_t - S_{t-1})/S_{t-1}$. El espacio de probabilidad Ω denota todos los posibles escenarios $\omega \in \Omega$ en los que varía el precio. Como nos hemos restringido al caso binomial, $\Omega = \{\omega_1, \omega_2\}$. Tenemos por tanto la variable aleatoria $R_t : \Omega \rightarrow (-1, \infty)$ definida como:

$$R_t = \begin{cases} 1 + u & \text{con probabilidad } p \\ 1 + d & \text{con probabilidad } 1 - p \end{cases}$$

cumpliendo $-1 < d < u$ y $0 < p < 1$. La primera condición es importante ya que garantiza que todos los precios van a ser positivos tal y como especificaremos más adelante. Si por ejemplo $d < 0$, significa que $R_t < 1$ y por tanto $S_t < S_{t-1}$, es decir, hemos perdido dinero. Si por el contrario es positivo, significa que hemos obtenido beneficio. No se tiene que dar que $d < 0 < u$ ya que podríamos estar en una situación en la que siempre perdamos dinero o lo ganemos. El valor d significa la menor ganancia y u la mayor. Deberíamos denotar como $R_t(\omega)$ al retorno obtenido en el paso t si el mercado sigue el escenario $\omega \in \Omega$. El árbol de probabilidad para dos pasos se observa en la figura 9.1.

Por lo tanto, el precio el instante 1 viene dado por

$$S_1 = \begin{cases} S_0(1 + u) & \text{con probabilidad } p \\ S_0(1 + d) & \text{con probabilidad } 1 - p \end{cases},$$

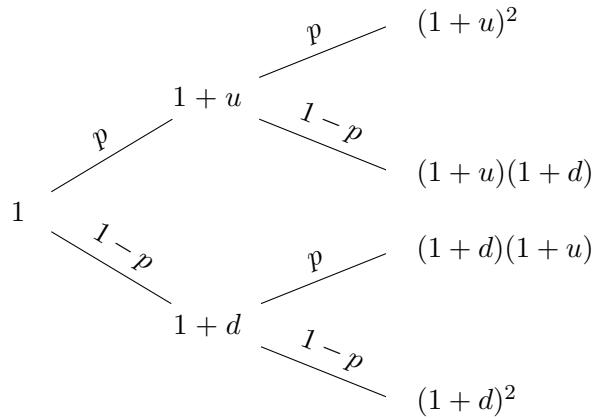


Figura 9.1: Ganancias en un árbol binomial de dos pasos.

donde S_0 es el precio actual del activo y que es conocido. Siguiendo un razonamiento “binomial”, el precio S_t de un activo se calcula de la siguiente forma:

$$S_t = S_0(1+u)^i(1+d)^{t-i} \text{ con probabilidad } \binom{t}{i} p^i (1-p)^{t-i}, \quad (9.1)$$

para $i = 1, \dots, t$, donde i es el número de escenarios en los que la ganancia es u (por ello hay $t - i$ escenarios en los que la ganancia es d).

Los activos que hasta ahora hemos presentado son denominados *primarios* porque son independientes de otros títulos de valor. Por otro lado tenemos los activos *derivados* que son aquellos cuyo valor cambia en función de otros activos denominados subyacentes que pueden ser primarios u otros derivados. Ejemplos de activos derivados son:

1. Contrato *forward* (a plazo): es un acuerdo entre dos partes para comprar o vender cierto activo con riesgo a un precio fijo en un momento determinado en el futuro.
2. Contrato de futuros: es un tipo de contrato *forward* pero que está estandarizado y negociado en un mercado organizado, como el de Chicago.
3. Opciones: es un contrato mediante el cual el comprador de la opción adquiere el derecho pero no la obligación de comprar o vender un activo subyacente al vendedor de la misma. El precio al que se puede ejercer el derecho de compra o de venta del activo se denomina precio de ejercicio o también *strike price*. Existen dos tipos de opciones:
 - a) Europeas: solo pueden ser ejercidas en la fecha de vencimiento.
 - b) Americanas: se puede ejercer en cualquier momento hasta la fecha de vencimiento.

A su vez, distinguimos entre:

- Opciones de compra (*call*): otorga al poseedor de la misma la posibilidad de comprar el activo.

- Opciones de venta (*put*): da al poseedor de la misma la posibilidad de vender el activo.

En esta memoria nos centraremos en las opciones europeas.

Si la situación es la que se ha explicado anteriormente, el propietario de una opción puede obtener un beneficio sin riesgo. Por ejemplo, supongamos que tenemos una opción que nos otorga el derecho de comprar un bien a un determinado precio. Si en el momento de ejercer dicha opción el precio de mercado es más bajo, no ejerzo el derecho y la compro a precio de mercado. Por otro lado, si es más alto puedo ejercer la opción y pagar menos dinero. De este modo, si S_t representa la variable aleatoria que modela el precio del activo y K es el precio del activo acordado en la opción, el *payoff* o ganancia obtenida en una opción *call* es $C_T = \max\{S_T - K, 0\} = [S_T - K]^+$ y de una opción *put* $P_T = \max\{K - S_T, 0\} = [K - S_T]^+$. Por eso, el comprador debe pagar una prima para obtener una opción. Este precio no puede ser ni demasiado bajo ni demasiado alto ya que, de ese modo, nadie compraría la opción. Si representamos por C_0 el precio de una opción y no tenemos en cuenta que el precio del dinero cambia con el tiempo, las ganancias o pérdidas de las opciones *call* y *put* se representan en las gráficas 9.2 y 9.3 respectivamente.

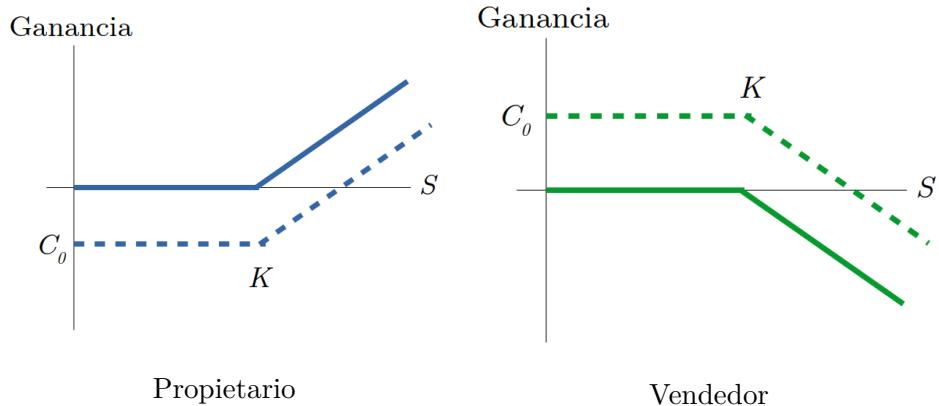
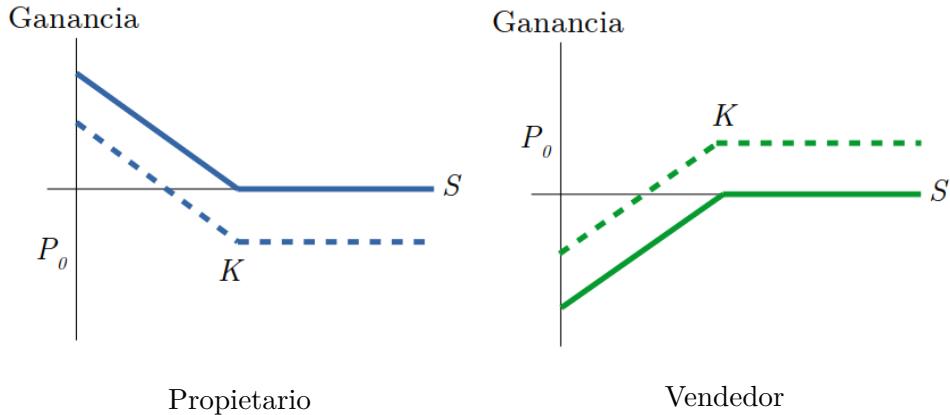


Figura 9.2: Ganancia opción *call*.

Uno de los problemas más importantes es determinar de manera única el precio “justo” de una opción en un momento determinado para que ambas partes estén de acuerdo. Para más información acerca de conceptos financieros, se pueden consultar [2] y el manual para universitarios publicado por la Comisión Nacional del Mercado de Valores que se puede consultar en <https://www.cnmv.es/DocPortal/Publicaciones/Guias/ManualUniversitarios.pdf>.

9.2. Primer teorema fundamental

Una vez explicada de manera simplificada el contexto financiero debemos formalizar matemáticamente el modelo del mercado. Para ello, seguimos [2]. Fijamos

Figura 9.3: Ganancia opción *put*.

el conjunto de tiempos $\mathbb{T} = \{0, 1, \dots, T\}$ con T el instante en el que finaliza el modelado de nuestra actividad económica. También fijamos el espacio de probabilidad (Ω, \mathcal{F}, P) . Este espacio contiene todos los posibles estados del mercado. La información de la que disponen los inversores acerca de la estructura en cada momento $t \in \mathbb{T}$ viene dada por una sucesión finita y creciente de sub- σ -álgebras de \mathcal{F} , $\mathcal{F}_0 \subset \dots \subset \mathcal{F}_T = \mathcal{F}$ con \mathcal{F}_0 trivial, es decir, contiene solamente conjuntos con medida 0 o 1. A este secuencia se le denomina filtración y se denota como $\mathbb{F} = (\mathcal{F}_t)_{t \in \mathbb{T}}$. El sentido financiero de esta filtración es indicar la información conocida hasta el momento acerca de la evolución de los precios. Por ejemplo, si comenzamos a modelar nuestro mercado hoy, \mathcal{F}_0 nos aporta únicamente el valor actual. Sin embargo, mañana tendremos la información de los precios en ese momento junto con la obtenida hoy, por lo que es claro que $\mathcal{F}_0 \subset \mathcal{F}_1$. También es claro que al llegar al instante final tenemos toda la información acerca del modelo. Llamamos l a la dimensión de nuestro mercado, es decir, el número de activos que manejamos. La evolución de los precios de los activos viene dada por el proceso estocástico $S = \{S_t^i : t \in \mathbb{T}, i = 0, \dots, l\}$ donde cada S_t^i es una variable aleatoria. Todos son activos con riesgos menos el marcado por 0. Asumimos que el proceso S_t^i es \mathcal{F}_t -medible para $i = 0, \dots, l$, es decir, se adapta a la filtración \mathbb{F} . Eso significa que para cada $t \in \mathbb{T}$ se conocen los precios de cada activo hasta ese instante como se ha explicado anteriormente. En general, primero se escoge S y se toma \mathbb{F} como la filtración que genera. La tupla $(\Omega, \mathcal{F}, P, \mathbb{F}, S)$ es la que modela nuestro mercado de valores.

El poder adquisitivo de una misma cantidad decrece con el tiempo, consideramos el factor de descuento (o de actualización) en el instante t dado por $\beta_t < 1$ y notamos al valor actual como $\bar{S}_t = \beta_t S_t$. Sin pérdida de generalidad se puede asumir que $S_0^0 = 1$ por lo que podemos expresar todas las unidades en función a dicho valor. En ese caso, el factor $\beta_t = 1/S_t^0$ es la cantidad de dinero que necesitamos para invertir en bonos en el instante 0 para tener una unidad en el instante t . Podríamos considerar que tomar $S_0^0 = 1$ es una simple normalización de los precios. El factor de descuento se puede ver como $\beta_t = (1+r)^{-t}$. Para entender esta fórmula podemos ver la situación contraria: calcular una cantidad con cierto intereses. De este modo, r representa el interés considerado o cómo aumentará el valor del dinero

en el futuro. Así, el valor de una cantidad S en el futuro es $(1+r)S$ considerando solo un paso o $(1+r)^t S$ si consideramos varios. Por eso, al actualizar pasamos dicho término dividiendo y llegamos a la fórmula de β_t expuesta. Observemos que en ocasiones, como en el caso continuo, el papel de $(1+r)^t$ es una función exponencial.

La cartera de inversión o portafolio para el instante $t \in \mathbb{T}$ viene dada por la variable aleatoria $l+1$ dimensional $\theta_t = (\theta_t^0, \dots, \theta_t^l)$. Cada θ_t^i indica el número de activos del tipo i que tiene un inversor en el instante t . El valor de la cartera viene determinado por $V_t(\theta)$ donde

$$V_0(\theta) = \langle \theta_1, S_0 \rangle, \quad V_t(\theta) = \langle \theta_t, S_t \rangle = \sum_{i=0}^l \theta_t^i S_t^i \text{ para } t \in \mathbb{T}, t \geq 1.$$

Cada inversor selecciona la cartera de inversión del instante t una vez se conocen los precios del momento anterior $t-1$. La estrategia de inversión $\theta = \{\theta_t : t = 1, \dots, T\}$ es el conjunto de todas las carteras de inversión *predecibles*. Decimos que una cartera θ_t es predecible si depende solamente de los precios de los activos hasta el instante $t-1$. Cuando cada cartera de inversión θ_{t+1} se puede financiar completamente con las ganancias o pérdidas actuales decimos que la estrategia es *autofinanciada*, es decir,

$$\langle \theta_{t+1}, S_t \rangle = \langle \theta_t, S_t \rangle,$$

Notamos $\Delta X_t = X_t - X_{t-1}$ para cualquier función sobre \mathbb{T} . Si una cartera es autofinanciada, escribimos la ganancia de una cartera de inversión entre los instantes $(t-1, t]$ como

$$\Delta V_t(\theta) = \langle \theta_t, S_t \rangle - \langle \theta_{t-1}, S_{t-1} \rangle = \langle \theta_t, S_t \rangle - \langle \theta_t, S_{t-1} \rangle = \langle \theta_t, \Delta S_t \rangle.$$

De este modo, la ganancia asociada a la estrategia θ hasta el instante t viene dada por

$$G_0(\theta) = 0, \quad G_t(\theta) = \langle \theta_1, \Delta S_1 \rangle + \dots + \langle \theta_t, \Delta S_t \rangle \text{ para } t \in \mathbb{T}, t \geq 1.$$

Vemos entonces que una estrategia es autofinanciada si, y solo si

$$V_t(\theta) = V_0(\theta) + G_t(\theta), \quad \forall t \in \mathbb{T}. \tag{9.2}$$

Destacamos que todas las definiciones anteriores se pueden definir también en base a los valores actualizados \bar{S}_t .

También es necesario imponer o aclarar una serie de suposiciones sobre nuestro modelo financiero:

1. Ninguna de las transacciones conlleva un coste extra.
2. Positividad: todos los precios son positivos, es decir,

$$S_t^i > 0 \text{ para } t \in \mathbb{T}, i = 0, \dots, l.$$

3. Divisibilidad, liquidez y *short selling* (venta en corto): el número de activos que posee un inversor puede ser cualquier valor real. Así,

$$\theta_t^i \in \mathbb{R} \text{ para } t \in \mathbb{T}, i = 0, \dots, l.$$

La divisibilidad hace referencia a que θ_t^i puede ser una fracción. Claramente, no podemos tener, por ejemplo, media acción. Sin embargo, cuando estamos trabajando con un gran número de acciones podemos considerar que tienen cifras decimales para trabajar con números menores. El hecho de que tome valores en \mathbb{R} también significa que podemos comprar o vender el número de acciones deseado, es decir, no imponemos ninguna restricción. Este hecho se conoce como liquidez. Claramente, en mercados reales sí existe dicha restricción en el volumen de las transacciones pero nosotros estamos trabajando con un modelo idealizado. Finalmente, cuando dichos valores son positivos decimos que el inversor tiene una posición larga o *long position*. Si por el contrario estos son negativos, tiene una posición corta o *short position*, por ejemplo, cuando vende algún tipo de bono. Estas acciones se suelen llevar a cabo, por ejemplo, en las denominadas ventas en corto o *short selling* que se usan cuando se prevé que un determinado activo va a bajar de valor. Así, suponemos que tenemos una acción cuyo precio actual es 100 y suponemos que va a disminuir. Antes de que baje más, la vendemos, es decir, tomamos una posición *short*. Pasado un periodo de tiempo, la acción bajan a 70 por lo que tomamos una posición *long* y las compramos de nuevo. De este modo, tenemos la misma cantidad de activos que al principio pero hemos obtenido un beneficio de 30. En estas operaciones también corremos el riesgo de perder dinero ya que es posible que el valor de los activos crezca en vez de disminuir.

4. Solvencia: el valor de todas las carteras de inversión debe ser siempre no negativo,

$$V_t(\theta) \geq 0 \text{ para todo } t \in \mathbb{T}.$$

En este caso, decimos que la cartera es *admisible*. A la clase de estrategias autofinanciadas y admisibles la denotaremos como Θ_a .

5. El espacio Ω es finito, es decir, las variables aleatorias S_t^i no pueden tomar infinitos valores. Así, tenemos que $\Omega = \{\omega_1, \dots, \omega_n\}$.

Ya tenemos el modelo financiero sobre el que vamos a trabajar. Exponemos ahora unos resultados que nos servirán posteriormente para probar el teorema principal del capítulo.

Lema 9.1. *Dada V_0 una función \mathcal{F}_0 -medible y para $l \in \mathbb{N}$ sean los procesos reales y predecibles $\theta^1, \dots, \theta^l$, el único proceso predecible θ^0 que convierte a $\theta = (\theta^0, \theta^1, \dots, \theta^l)$ en una estrategia autofinanciada con valor inicial $V_0(\theta) = V_0$ viene dado por*

$$\theta_t^0 = V_0 + \sum_{i=1}^{t-1} (\theta_i^1 \Delta \bar{S}_i^1 + \dots + \theta_i^l \Delta \bar{S}_i^l) - (\theta_t^1 \bar{S}_{t-1}^1 + \dots + \theta_t^l \bar{S}_{t-1}^l).$$

*Demuestra*ción. Claramente θ^0 es predecible. Para ver que la estrategia es autofinanciada, por (9.2) solo necesitamos ver que θ_t^0 es la única solución predecible de la ecuación

$$\begin{aligned} \bar{V}_t(\theta) &= \theta_t^0 + \theta_t^1 \bar{S}_t^1 + \dots + \theta_t^l \bar{S}_t^l \\ &= V_0 + \sum_{i=1}^t (\theta_i^1 \Delta \bar{S}_i^1 + \dots + \theta_i^l \Delta \bar{S}_i^l). \end{aligned}$$

■

Introducimos ahora el concepto de *viabilidad*, esencial para lo que sigue. Decimos que un mercado es viable si para toda estrategia admisible y autofinanciada no contiene ninguna oportunidad de arbitraje. La ausencia de arbitraje significa que si el valor inicial de una cartera de inversión es $V_0(\theta) = 0$ entonces $V_T(\theta) = 0$ con probabilidad 1 para toda $\theta \in \Theta_a$. La viabilidad del mercado impone la siguiente restricción.

Lema 9.2. *Si el modelo de mercado es viable, las ganancias actualizadas a cualquier proceso predecible $\hat{\theta} \in \mathbb{R}^l$ no pueden pertenecer a*

$$C = \{Y \in \mathbb{R}^n : Y_i \geq 0 \text{ para } i = 1, \dots, n \text{ y } \exists i \text{ tal que } Y_i > 0\}.$$

*Demuestra*ción. En primer lugar vemos que C es el octante positivo de \mathbb{R}^n sin el origen, que claramente es un cono y es convexo. La ausencia de arbitraje significa que para toda estrategia admisible $\theta \in \Theta_a$ tal que $V_0(\theta) = 0$ entonces

$$\bar{V}_t(\theta) = \bar{G}_t(\theta) \notin C.$$

Por el lema 9.1, dados los procesos predecibles $\hat{\theta} = (\theta^1, \dots, \theta^l)$, existe un único proceso real θ^0 tal que $\theta = (\theta^0, \theta^1, \dots, \theta^l)$ es autofinanciada y $V_0(\theta) = 0$. Las ganancias con los valores actualizados viene dada por

$$\bar{G}_t(\hat{\theta}) = \sum_{j=1}^t \langle \theta_j, \Delta \bar{S}_j \rangle = \sum_{j=1}^t \left(\sum_{i=1}^l \theta_j^i \Delta \bar{S}_j^i \right).$$

Supongamos que $\bar{G}_t(\hat{\theta}) \in C$; si β_T denota el factor de descuento en el instante T ,

$$V_T(\theta) = \beta_T^{-1} \bar{V}_T(\theta) = \beta_T^{-1} (V_0(\theta) + \bar{G}_T(\theta)) = \beta_T^{-1} \bar{G}_T(\hat{\theta}).$$

Vemos entonces que $V_T(\theta)$ es no negativa y estrictamente positiva con probabilidad no nula, lo que contradice la viabilidad al existir arbitraje. ■

Nuestro objetivo es caracterizar la viabilidad de un mercado en términos de los incrementos de \bar{S} . Para ello son necesarias las martingalas.

Definición 9.1. *Un proceso \mathbb{F} -adaptado $M = (M_t)_{t \in \mathbb{T}}$ es una (\mathbb{F}, P) -martingala si $E(|M_t|) < \infty$ para todo $t \in \mathbb{T}$ y*

$$E(M_{t+1} | \mathcal{F}_t) = M_t \text{ para todo } t \in \mathbb{T} \setminus \{T\}.$$

Si $M = (M_t)$ es una martingala y $\phi = (\phi_t)_{t \in \mathbb{T}}$ es un proceso predecible en $(\Omega, \mathcal{F}, P, \mathbb{F}, S)$ entonces al proceso $X = \phi \cdot M$ dado por

$$X_0 = 0, \quad X_t = \phi_1 \Delta M_1 + \cdots + \phi_t \Delta M_t \quad t \geq 1$$

se le denomina martingala transformada de M por ϕ .

Notamos que M es una martingala si, y solo, si

$$E(\Delta M_{t+1} | \mathcal{F}_t) = 0 \text{ para todo } t \in \mathbb{T} \setminus \{T\}.$$

También es importante destacar que, por la linealidad de la esperanza, cualquier combinación lineal de martingalas es una martingala.

Que los precios en el mercado sigan una martingala, tal y como veremos, no debe de ser extraño. Recordemos que para $t \in \mathbb{T}$, \mathcal{F}_t indica la cantidad de información acerca de los precios de los activos hasta dicho momento. Por lo tanto, la esperanza condicionada solo nos indica que estamos calculando el valor esperado a partir de lo que conocemos hasta ahora. Además, para que el mercado sea “justo”, dicho valor en el futuro debería ser, en media, el que tenemos ahora.

El siguiente resultado es meramente técnico y nos servirá posteriormente.

Teorema 9.1. *Un proceso real M es una martingala si, y solo si,*

$$E((\phi \cdot M)_t) = E\left(\sum_{i=1}^t \phi_i \Delta M_i\right) = 0, \quad \forall t \in \mathbb{T} \setminus \{0\}$$

para todo proceso ϕ predecible y acotado.

Demostración. Si M es un martingala, también lo es la transformada $X = \phi \cdot M$ y $X_0 = 0$ ya que

$$E(\Delta X_{t+1} | \mathcal{F}_t) = E(\phi_{t+1} \Delta M_{t+1} | \mathcal{F}_t) = \phi_{t+1} E(\Delta M_{t+1} | \mathcal{F}_t) = 0.$$

Por ello, $E((\phi \cdot M)_t) = 0$ para todo $t \geq 1$ en \mathbb{T} . Demostremos ahora la otra implicación. Para $s > 0$, sea $A \in \mathcal{F}_s$ y definimos el proceso predecible ϕ como $\phi_{s+1} = 1_A$ y $\phi_t = 0$ para el resto de $t \in \mathbb{T}$. Entonces, para $t > s$ se tiene que

$$0 = E((\phi \cdot M)_t) = E(1_A(M_{s+1} - M_s)).$$

Como es cierto para todo $A \in \mathcal{F}_s$, se cumple que $E(\Delta M_{s+1} | \mathcal{F}_s) = 0$ por lo que M es una martingala. ■

Nos encontramos ahora un contexto general donde no asumimos que el modelo sea finito o que \mathbb{F} sea generada por S . Supongamos que el proceso de los precios actualizados \bar{S} es una martingala bajo una probabilidad Q , esto es:

$$E_Q(\Delta \bar{S}_t^i | \mathcal{F}_{t-1}) = 0, \quad \text{para } t \in \mathbb{T} \setminus \{0\} \text{ e } i = 0, \dots, l,$$

donde E_Q significa la esperanza respecto de la probabilidad (medida) Q . Sea $\theta \in \Theta_a$ una estrategia admisible tal que los procesos de los precios actualizados son integrables respecto a Q . Por (9.2) tenemos que:

$$\begin{aligned} \bar{V}_t(\theta) &= V_0(\theta) + \bar{G}_t(\theta) \\ &= \langle \theta_1, S_0 \rangle + \sum_{u=1}^t \langle \theta_u, \Delta \bar{S}_u \rangle \\ &= \sum_{i=1}^l (\theta_1^i S_0^i + \sum_{u=1}^t \theta_u^i \Delta \bar{S}_u^i). \end{aligned}$$

Vemos entonces que $\bar{V}_t(\theta)$ es una constante más una suma finita de martingalas transformadas, por lo que también es una martingala con valor inicial $V_0(\theta)$. Entonces, tenemos que:

$$E(\bar{V}_t(\theta)) = E(V_0(\theta)) = V_0(\theta).$$

Esta situación imposibilita la existencia de arbitraje. Si sabemos de antemano que los procesos de los precios actualizados son integrables respecto a Q , supongamos que $V_0(\theta) = 0$ y $V_T(\theta) \geq 0$ casi seguramente (respecto a Q). Como $E_Q = (\bar{V}_t(\theta)) = 0$ se sigue que $V_T(\theta) = 0$ casi seguramente (respecto a Q). Esto sigue siendo verdadero casi seguramente para P , demostrando que P y Q tienen los mismos conjuntos vacíos. Llegamos entonces a la siguiente definición:

Definición 9.2. Una probabilidad Q que sea equivalente a P como media ($Q \sim P$) es una medida martingala equivalente (EMM) para S si el proceso de los precios actualizados \bar{S} es una martingala bajo Q para la filtración \mathbb{F} . Es decir, para cada $i = 0, \dots, d$ tenemos que \bar{S}^i es una (\mathbb{F}, Q) -martingala.

Todo lo presentado anteriormente, nos aporta el siguiente resultado que acabamos de demostrar:

Proposición 9.1. Si existe una medida martingala equivalente para S , entonces el modelo de mercado discreto es viable, es decir, no contiene ninguna oportunidad de arbitraje.

El recíproco de esta afirmación es cierto, tal y como exponemos en el siguiente teorema que es el objetivo final de este capítulo y en el que usaremos el teorema de la alternativa de Gordan en forma equivalente del teorema de separación establecido en el corolario 7.1. Recibe el nombre de “primer teorema fundamental de asignación de precios”.

Teorema 9.2. (Primer teorema fundamental de asignación de precios) Un modelo de mercado discreto es viable si, y solo si, existe una medida de martingala equivalente para S .

Demostración. Ya sabemos por la proposición 9.1 que la existencia de una medida de martingala equivalente garantiza la viabilidad del modelo por lo que solo tenemos que probar la otra implicación.

Suponemos entonces que el modelo es viable. Necesitamos construir una medida $Q \sim P$ en la que los precios son martingalas relativas a la filtración \mathbb{F} . Sea C es el cono convexo de todas las variables aleatorias reales ϕ en (Ω, \mathcal{F}) tales que $\phi(\omega) \geq 0$ casi seguramente y $\phi(\omega_i) > 0$ para al menos un $\omega_i = \Omega = \{\omega_1, \dots, \omega_n\}$. Asumimos $p_i = P(\{\omega_i\}) > 0$. Por el lema 9.2, hemos visto que para un mercado

viable debemos tener $\bar{G}_t(\hat{\theta}) \notin C$ para todos los procesos predecibles $\hat{\theta} \in \mathbb{R}^l$. Por otro lado, el conjunto definido por tales ganancias

$$L = \{\bar{G}_t(\hat{\theta}) : \hat{\theta} = (\theta^1, \dots, \theta^l), \text{ con } \theta^i \text{ predecible para } i = 1, \dots, l\},$$

es un subespacio vectorial del espacio de todas las funciones reales y \mathcal{F} -medibles en Ω . Como L y C son disjuntos, podemos separar L y el subconjunto compacto de C definido como $K = \{X \in C : E_P(X) = 1\}$ gracias al corolario 7.1. Sea $\mathbf{x}^0 \in \mathbb{R}^n$ el elemento que proporciona dicho resultado. Tomamos $\xi_i = (0, \dots, \frac{1}{p_i}, \dots, 0)$ para $i \leq n$. Vemos que $E_P(\xi_i) = \frac{p_i}{p_i} = 1$ por lo que $\xi_i \in K$ y $\langle \mathbf{x}^0, \xi_i \rangle = \frac{x_i^0}{p_i} > 0$. Por ello, $x_i^0 > 0$ para todo $i = 1, \dots, n$.

Definimos ahora el funcional lineal $g(\mathbf{x}) = \frac{\langle \mathbf{x}^0, \mathbf{x} \rangle}{\alpha}$ donde $\alpha = \sum_{i=1}^n x_i^0$. Sea $p^* \in \mathbb{R}^n$ un vector con $p_i^* = \frac{x_i^0}{\alpha}$ por lo que $\sum_{i=1}^n p_i^* = 1$. Usamos el vector p^* para inducir una probabilidad P^* en $\Omega = \{\omega_1, \dots, \omega_n\}$ haciendo $P^*(\{\omega_i\}) = p_i^* > 0$. Veamos que P^* es la martingala equivalente deseada. En efecto, sea $E^*(\cdot)$ que denota la esperanza relativa a P^* . Nuevamente, por el corolario 7.1, tenemos que $g(\mathbf{x}) = \frac{1}{\alpha} \langle \mathbf{x}^0, \mathbf{x} \rangle = 0$ para todo $\mathbf{x} \in L$. En particular, esta situación se da para para $\bar{G}_T(\hat{\theta})$ con $\hat{\theta} = (\theta^1, \dots, \theta^l)$ un vector de procesos predecibles por lo que.

$$E^*(\bar{G}_T(\hat{\theta})) = 0.$$

Hemos conseguido una estrategia auto-financiada θ con $V_0(\theta) = 0$. Como $\bar{V}_T(\theta) = V_0(\theta) + \bar{G}_T(\theta)$, implica que $E^*(\bar{V}_T(\theta)) = 0$ para ese θ . Por el lema 9.1 podemos generar tal θ para cada proceso predecible de n dimensiones. En particular, lo podemos hacer para $(0, \dots, \theta^i, \dots, 0)$ donde $i \leq n$. Por lo tanto

$$E^*\left(\sum_{t=1}^T \theta_t^i \Delta \bar{S}_t^i\right) = 0$$

se da para cada proceso predecible y acotado $(\theta_i)_{i=1, \dots, T}$. El teorema 9.1 implica que cada S^i es una martingala bajo P^* . Por ello, $P^* \sim P$. ■

9.3. Aplicación al modelo binomial

Usemos toda la información obtenida para llegar a una fórmula que nos permitirá establecer el precio de una opción europea. Para ello, utilizaremos un modelo binomial con más de un paso en el que suponemos que $l = 1$. De este modo, tenemos un activo sin riesgo con precios S^0 y uno con riesgo con precios S^1 los cuales denotaremos por S . También, como mencionamos anteriormente, sin pérdida de generalidad asumimos que $S_0^0 = 1$ por lo que $S_t^0 = (1+r)^t$ para $t \in \mathbb{T}$. Los precios de nuestro activo con riesgo vienen determinados por $S_{t-1}(1+u)$ o $S_{t-1}(1+d)$ cumpliendo $-1 < d < u$, es decir, siguen una distribución de Bernoulli. El valor inicial S_0 es constante y conocido. Tomamos entonces como espacio de probabilidad $\Omega = \{1+u, 1+d\}^T$ con la filtración \mathbb{F} dada por $\mathcal{F}_0 = \{\emptyset, \Omega\}$ y $\mathcal{F}_t = \sigma(S_u^1 : u \leq t)$ para $t > 0$. Para $\omega = \{\omega_1, \dots, \omega_T\}$ en Ω definimos

$$P(\{\omega\}) = P\left(\frac{S_t}{S_{t-1}} = \omega_t, t = 1, \dots, T\right).$$

Notamos a $\frac{S_t}{S_{t-1}}$ como R_t tal y como ya hemos hecho anteriormente. Veamos la relación necesaria entre d, u y r para que tengamos una medida martingala equivalente. En el libro de referencia aparece una errata en el razonamiento de este resultado debido a una confusión en la relación entre β_t y β_{t-1} .

Lema 9.3. *Para tener una EMM en el modelo binomial se debe cumplir que $d < r < u$.*

Demostración. Sea Q una probabilidad en (Ω, \mathcal{F}) . Para que sea una martingala equivalente se debe dar:

$$E_Q(\bar{S}_t | \mathcal{F}_{t-1}) = \bar{S}_{t-1}.$$

⇓

$$E_Q\left(\frac{\bar{S}_t}{\bar{S}_{t-1}} | \mathcal{F}_{t-1}\right) = E_Q\left(\frac{\beta_t S_t}{\beta_{t-1} S_{t-1}} | \mathcal{F}_{t-1}\right) = \frac{(1+r)^{-t}}{(1+r)^{-(t-1)}} E_Q(R_t | \mathcal{F}_{t-1}) = 1.$$

Entonces, $E_Q(\bar{S}_t | \mathcal{F}_{t-1}) = \bar{S}_{t-1}$ es equivalente a que $E_Q(R_t | \mathcal{F}_{t-1}) = \frac{\beta_{t-1}}{\beta_t} = 1 + r$. Como solo puede tomar valores $1+u$ y $1+d$, su valor medio solo puede $1+r$ si, y solo si, $d < r < u$. ■

A continuación, vamos a construir dicha medida Q mediante el siguiente lema:

Lema 9.4. *El proceso \bar{S} es una (\mathbb{F}, Q) -martingala si, y solo si, las variables aleatorias R_t son independientes e idénticamente distribuidas con $Q(R_1 = 1+u) = \frac{r-d}{u-d}$ y $Q(R_1 = 1+d) = 1 - \frac{r-d}{u-d}$. Además, si el modelo es viable, Q es única.*

Demostración. Para simplificar la notación, llamamos $q = \frac{r-d}{u-d}$.

⟺) Como R_t son independientes:

$$E_Q(R_t | \mathcal{F}_{t-1}) = E_Q(R_t) = q(1+u) + (1-q)(1+d) = q(u-d) + 1 + d = 1 + r.$$

Siguiendo un proceso similar a la demostración del lema 9.3, vemos que efectivamente es una (\mathbb{F}, Q) -martingala.

⇒) En este caso, $E_Q(R_t | \mathcal{F}_{t-1}) = 1 + r$. Como solo puede tomar valores $1+u$ y $1+d$, se debe cumplir que

$$(1+d)Q(R_1 = 1+d | \mathcal{F}_{t-1}) + (1+u)Q(R_1 = 1+u | \mathcal{F}_{t-1}) = 1 + r$$

y

$$Q(R_1 = 1+d | \mathcal{F}_{t-1}) + Q(R_1 = 1+u | \mathcal{F}_{t-1}) = 1.$$

Llamando $q = Q(R_1 = 1+u | \mathcal{F}_{t-1})$ obtenemos

$$(1+d)(1-q) + (1+u)q = 1 + r \implies q = \frac{r-d}{u-d}.$$

Con un argumento inductivo, para $\omega = (\omega_1, \dots, \omega_T) \in \Omega$ vemos que

$$Q(R_1 = \omega_1, \dots, R_t = \omega_t) = \prod_{i=1}^t q_i$$

donde

$$q_i = \begin{cases} q & \text{si } \omega_i = 1 + u \\ 1 - q & \text{si } \omega_i = 1 + d \end{cases}.$$

Por lo tanto, las variables R_t son independientes e idénticamente distribuidas.

Finalmente veamos la unicidad. Es claro que $q \in (0, 1)$ si, y solo si, $d < r < u$. Por ello, usando el teorema 9.2 y el lema 9.3, obtenemos que el modelo admite una única EMM que además es Q . ■

Al valor $q = (r - d)/(u - d)$ obtenido en el resultado anterior se le suele denominar *probabilidad de riesgo neutral*. Es un objeto matemático abstracto que no tiene que coincidir con la probabilidad real del mercado o estar relacionada con la misma y que de hecho, se conoce *a priori*. Sin embargo, para la valoración de opciones se utiliza q ya que es la que hace que los precios actualizados se comporten según una martingala tal y como acabamos de probar. De este modo, la ganancia $C_T = [S_T - K]^+$ de una opción *call* en el instante $t \in \mathbb{T}$ viene dada por

$$V_t(C_T) = \frac{1}{\beta_t} E_Q(\beta_T C_T | \mathcal{F}_t).$$

Usando la definición de R_t , vemos que se cumple la siguiente relación:

$$S_T = S_t \prod_{u=t+1}^T R_u.$$

Podemos calcular su media ya que S_t es \mathcal{F}_t -medible y R_u son independientes de \mathcal{F}_t si $u > t$. De ese modo,

$$\begin{aligned} V_t(C_T) &= \frac{\beta_T}{\beta_t} E_Q \left(\left[S_t \prod_{u=t+1}^T R_u - K \right]^+ | \mathcal{F}_t \right) \\ &= (1+r)^{t-T} E_Q \left(\left[S_t \prod_{u=t+1}^T R_u - K \right]^+ | \mathcal{F}_t \right) \\ &= (1+r)^{t-T} \sum_{v=0}^{T-t} \binom{T-t}{v} q^v (1-q)^{T-t-v} [S_t (1+u)^v (1+d)^{T-t-v} - K]^+, \end{aligned}$$

donde en el último paso se ha usado la fórmula (9.1). Finalmente, en el instante 0, el valor de la opción es:

$$V_0(C_T) = (1+r)^{-T} \sum_{v=A}^T \binom{T}{v} q^v (1-q)^{T-v} [S_0 (1+u)^v (1+d)^{T-v} - K],$$

donde A denota el primer entero k que cumple $S_0(1+u)^k(1+d)^{T-k} > K$. Esto se debe a que

$$[S_0(1+u)^j(1+d)^{T-j} - K]^+ = 0, \text{ para } j < k.$$

Esta fórmula nos aporta el valor de una opción *call* o de compra. En la explicación de estos activos derivados dijimos que también existían las denominadas opciones *put* o de venta. Gracias a la ausencia de arbitraje podemos calcular el precio de una en función del precio de la otra. A este hecho se le denomina *paridad put-call*. Si denominamos por C_t al precio de la opción *call* y como P_t al de la opción *put* y tenemos en cuenta las ganancias como muestran las gráficas 9.2 y 9.3, se da la siguiente relación.

$$C_T - P_T = (S_T - K)^+ - (K - S_T)^+ = S_T - K.$$

Esta relación se debe dar para todo $t \in \mathbb{T}$ y si consideramos la actualización del valor de K en cada instante mediante el interés r llegamos a:

$$C_t - P_t = S_t - (1+r)^{-(T-t)}K,$$

donde el $(1+r)^{-(T-t)}$ es el factor de actualización entre los instantes t y T . Este hecho es bastante conocido y aparece recogido, por ejemplo, en [2].

Una vez obtenida la fórmula y hecha la aclaración sobre la relación de precios entre los dos tipos de opciones, podemos realizar una serie de simulaciones para ver cómo varía el precio de una opción de compra según los distintos parámetros (K, T, S, d, r, u) . Estas simulaciones se han realizado en *SageMath 2.8*. Para ello se han implementado las siguientes funciones como se observa en 9.1: una para obtener el valor de A llamada *getA* y otro el de la propia opción, *getV*.

Código 9.1: Definición en Sage de las funciones

```

1 sage: def getA(K, T, S, d, u):
2     k = 0
3     while S*(1+u)**(k)*(1+d)**(T-k) <= K:
4         k+=1
5     return k
6 sage: def getV(K, T, S, d, r, u):
7     A = getA(K, T, S, d, u)
8     q = (r-d)/(u-d)
9     if r <= d or r >= u:
10        return 0
11    else:
12        return (1+r)**(-T)*sum(binomial(T,v)*q**v*(1-q)**(T-v)*(S*(1+u)
13            **(v)*(1+d)**(T-v)- K) for v in range(A,T))

```

Para ello, partiremos del siguiente problema general del que iremos variando los diferentes parámetros en el que tomamos una unidad monetaria (u.m.) cualquiera.

“Las acciones de una empresa tienen valor actual 25 u.m. cada una. Queremos comprar acciones de dicha empresa pero no estamos seguros de obtener una gran ganancia con ellas en el futuro. Por eso, decidimos realizar una opción de compra de aquí a un año en el que se nos ofrece un precio de 27 u.m. Los tiempos se mueven en el intervalo de meses. Según las estimaciones suponemos que en cada mes el retorno podría ser de 1.2 (u = 0.2) pero también podemos perder dinero llegando

este a tomar el valor 0.72 ($d = -0.28$). Consideramos que el precio del dinero se incrementa de manera constante cada mes un 0.1 (de este modo el mercado es viable)."

El precio en esta situación es aproximadamente de 12.9325 u.m. Veamos cómo varía en función de cada parámetro manteniendo el resto como en el enunciado. En primer lugar, consideremos el valor como función de T y tomamos valores desde un mes hasta dos años, es decir, 24 meses. Los resultado se obtienen en la figura 9.4. También se incluye un ejemplo del código en 9.2 para dibujar la gráfica en este caso junto con la definición de los parámetros generales. Para el resto de parámetros se hace de manera análoga.

Código 9.2: Código en Sage para dibujar una gráfica

```

1 sage:K = 27
2 T = 12
3 S = 25
4 d = -0.28
5 r = 0.1
6 u = 0.2
7 sage: point([(T, getV(K, T, S, d, r, u)) for T in range(1, 25,1)],
    rgbcolor = (1,0,0), size = 35, axes_labels=['$T$', '$V_0(C_T)$'])

```

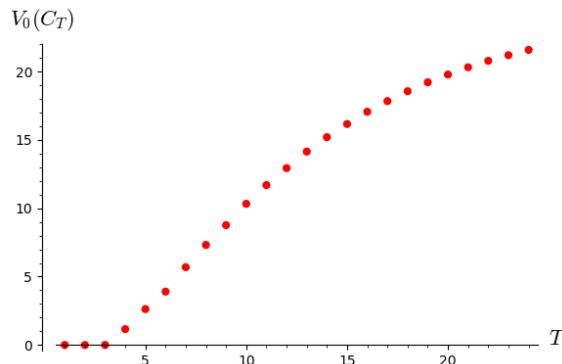


Figura 9.4: Valor de la opción en función del número de pasos.

Vemos que tendríamos que poner una fecha de compra superior a los 4 meses ya que durante los tres primeros $A = T$, lo que significa que lo más seguro es que no ejerciéramos nuestro derecho a la opción. Observamos que los valores crecen en función de este parámetro. Veamos qué pasa si variamos el precio de la acción fijado en la opción de compra, es decir, K . Se ha obtenido des este modo la gráfica 9.5 en la que K varía de 23 a 33. Vemos que en este caso es decreciente. Esta situación podemos considerar que es la esperada. Cuanto menor sea el precio más posibilidades tiene el propietario de ejercerla ya que es raro que en el futuro la acción tenga ese valor tan bajo. Por ello, el vendedor tiene que exigir un mayor. Por otro lado, cuanto mayor sea el precio acordado es el comprador el que corre un mayor riesgo de no ejercerla en el futuro por lo que tiene que pagar menos por ella ahora.

Obtengamos ahora el precio en función del valor actual S_0 . Hemos variado dicho valor entre 23 y 33. Los resultados se muestran en la gráfica 9.6.

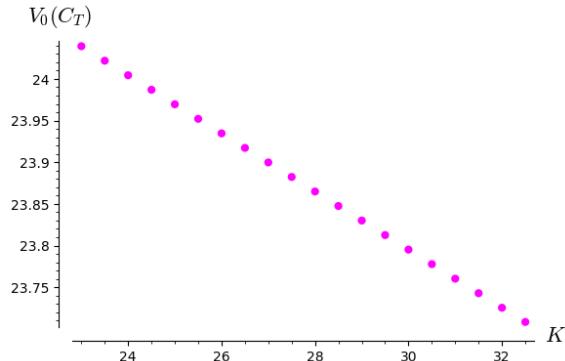


Figura 9.5: Valor de la opción en función del precio acordado.

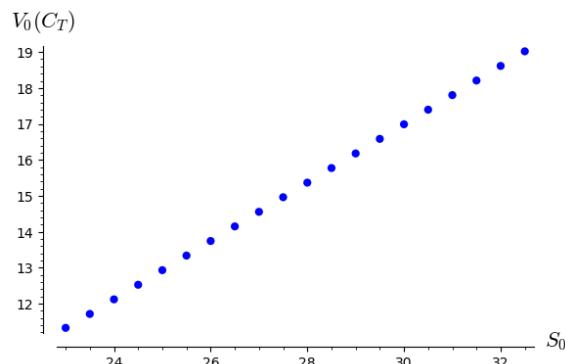


Figura 9.6: Valor de la opción en función del valor actual.

En este caso vuelven a ser crecientes, lo que era de esperar, ya que en la fórmula vemos que el precio de la opción es proporcional al valor actual de la misma. Finalmente, variemos los valores de u , d y r teniendo en cuenta las restricciones del lema 9.3. En la gráfica 9.7 se muestran los resultados en función de r . La variación

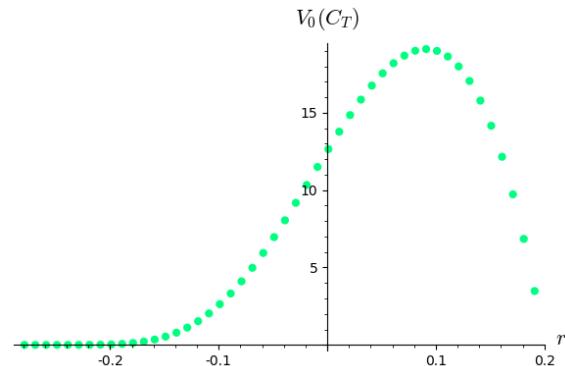


Figura 9.7: Valor de la opción en función del interés.

de este valor es quizás la más interesante desde el punto de vista del vendedor de la opción. El parámetro r es en cierto modo subjetivo ya que es una estimación del valor que tendrá el dinero en el futuro. Por ello, según los valores fijados de d y u puede ver con qué interés obtiene el mayor beneficio. Es claro que si $r < 0$ significa que el dinero en el futuro tendrá menos valor y es una situación que en principio no la contemplamos ya que suponemos que el valor del dinero aumentará en un futuro. De todos modos, se han usado valores negativos para ver su evolución. Para $r \geq 0$ vemos que en este caso la función alcanza un máximo que como hemos dicho es el valor más favorable al vendedor. Si consideramos un interés mayor al mismo le estamos exigiendo demasiada rentabilidad y eso hace bajar el precio de la opción. Veamos cómo varía ahora para valores de u entre $-0,4$ y $0,1$ mostrados en la gráfica 9.8. En este caso también es creciente lo que es normal ya que cuanto mayor sea

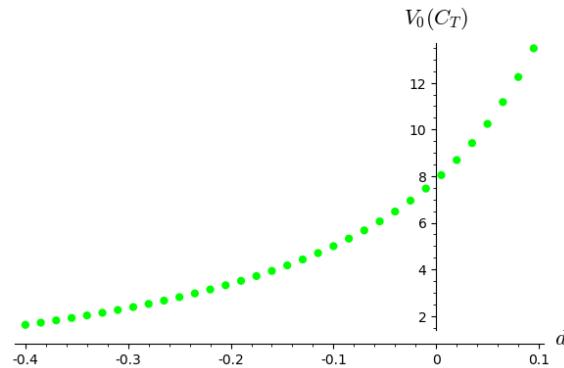


Figura 9.8: Valor de la opción en función de d .

d mayor será la posible ganancia del comprador. Por último veamos cómo varía en función de u . Para ello se han tomado valores entre $0,1$ y $0,55$. Se muestran en la gráfica 9.9.

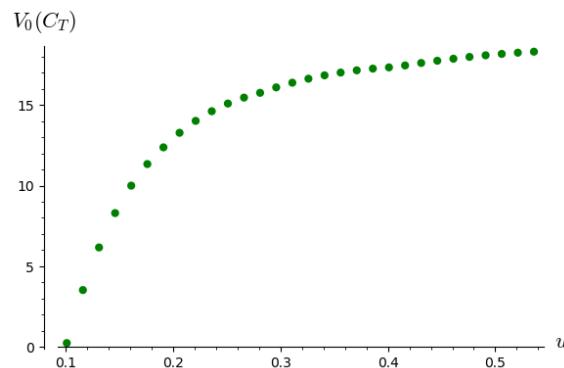


Figura 9.9: Valor de la opción en función de u .

Observamos que para valores cercanos al interés el valor de la opción es relativamente bajo ya que la ganancia que obtendremos sería menor. El precio va

aumentando aunque llega un punto en el que dicho crecimiento se reduce.

Parte II

Procesamiento de nubes de puntos generadas por escáner láser

Iterative Closest Point (ICP)

En esta sección explicaremos el primer algoritmo denominado *Iterative Closest Point* (ICP), o Iteración del Punto más Cercano. Este algoritmo se utiliza en la etapa de alineación y su finalidad es que todas las tomas de datos estén referenciadas respecto al mismo sistemas de coordenadas. Este proceso también se suele denominar *registro de datos* o *data registration*. Veremos el método inicial propuesto en 1992 por Paul J.Besl and Neil D.MacKay [10]. Como veremos, este es un algoritmo complejo en tiempo y por ello, se tarda bastante en obtener los resultados deseados. Por ello, una vez estudiado el algoritmo se propondrán unas pequeñas mejoras del mismo que consistirán, básicamente, en el uso de las normales en cada punto para acelerar el proceso.

Antes de explicar el algoritmo necesitamos unas nociones previas acerca de los *cuaternios*. Los utilizaremos para la representación de las rotaciones en tres dimensiones, frente a la forma tradicional de representación matricial, y serán de gran ayuda a la hora de demostrar la convergencia del algoritmo. Tanto para la explicación de los cuaternios como su uso en optimización se ha tomado como referencia [13].

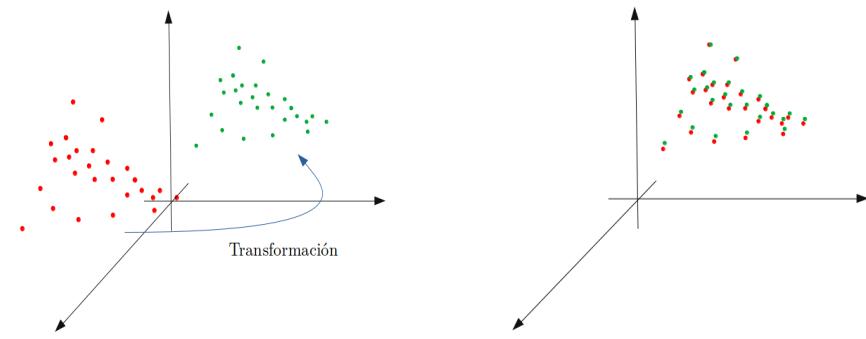


Figura 10.1: Antes y después de alinear dos nubes de puntos.

10.1. Cuaternios

Su desarrollo se atribuye a W.R Hamilton en 1843. Son una extensión de los números complejos. Mientras que estos últimos solo incluyen la unidad imaginaria

\mathbf{i} , los cuaternios añaden las unidades imaginarias $\mathbf{i}, \mathbf{j}, \mathbf{k}$ tales que:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1.$$

Así, el conjunto de los cuaternios se define como:

$$\{q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} : q_0, q_1, q_2, q_3 \in \mathbb{R}\}.$$

Podemos ver a los cuaternios como la suma de un escalar $q_0 \in \mathbb{R}$ y de un vector de \mathbb{R}^3 que llamamos $\mathbf{q} = (q_1, q_2, q_3)$ donde la base usual de \mathbb{R}^3 viene dada por los vectores unitarios $\mathbf{i} = (1, 0, 0)$, $\mathbf{j} = (0, 1, 0)$ y $\mathbf{k} = (0, 0, 1)$. Notaremos $q = q_0 + \mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$. Si la parte escalar es 0, es decir, $q_0 = 0$ hablamos de un *cuaterniono puro*. Por otro lado, a modo de notación podemos expresar q como vectores de cuatro dimensiones, es decir, si q es de la forma mencionada podemos pensar que $q = (q_0, q_1, q_2, q_3) \in \mathbb{R}^4$.

10.1.1. Suma y producto

La suma de dos cuaternios se realiza componente a componente. Por ello, dados los cuaternios

$$\begin{aligned} q &= q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} = q_0 + \mathbf{q} \\ p &= p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k} = p_0 + \mathbf{p}, \end{aligned}$$

definimos su suma de la siguiente manera:

$$q + p = (q_0 + p_0) + (q_1 + p_1)\mathbf{i} + (q_2 + p_2)\mathbf{j} + (q_3 + p_3)\mathbf{k}.$$

Es directo comprobar que la suma es asociativa y commutativa.

El producto, por su parte, sigue una serie de reglas introducidas por Hamilton y que las presentamos a continuación:

$$\begin{aligned} \mathbf{i}^2 &= \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1. \\ \mathbf{ij} &= \mathbf{k} = -\mathbf{ji}. \\ \mathbf{jk} &= \mathbf{i} = -\mathbf{kj}. \\ \mathbf{ki} &= \mathbf{j} = -\mathbf{ik}. \end{aligned}$$

Por ello, el producto de dos cuaternios viene dado por:

$$\begin{aligned} pq &= (p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k})(q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) \\ &= p_0q_0 - (p_1q_1 + p_2q_2 + p_3q_3) + p_0(q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) \\ &\quad + q_0(p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}) + (p_2q_3 - p_3q_2)\mathbf{i} + (p_3q_1 - p_1q_3)\mathbf{j} \\ &\quad + (p_1q_2 - p_2q_1)\mathbf{k}. \end{aligned}$$

Podemos simplificar esta expresión usando el producto escalar y vectorial de \mathbb{R}^3 quedando la expresión más concisa:

$$pq = p_0q_0 - \langle \mathbf{p}, \mathbf{q} \rangle + p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p} \times \mathbf{q}.$$

No es difícil comprobar que el producto es asociativo y distributivo respecto a la suma pero no commutativo. La no commutatividad se puede observar del hecho de que

el producto vectorial no es commutativo. Es importante notar que los cuaternios son cerrados bajo la suma y el producto definidos anteriormente. En el caso del producto, obtenemos otro cuaternion con parte escalar $p_0 q_0 - \langle \mathbf{p}, \mathbf{q} \rangle$ y vector $p_0 \mathbf{q} + q_0 \mathbf{p} + \mathbf{p} \times \mathbf{q}$.

Este conjunto con las operaciones de suma y producto forman un anillo no commutativo. El elemento neutro para la suma sería un cuaternion donde $q_0 = q_1 = q_2 = q_3 = 0$ y dado un elemento $q = q_0 + \mathbf{q}$ su opuesto sería $-q = -q_0 - \mathbf{q}$. Por su parte, el elemento neutro para el producto sería un cuaternion con parte real 1 y vector $\mathbf{0}$.

10.1.2. Conjugado, norma e inverso

Sea el cuaternion dado por $q = q_0 + q_1 \mathbf{i} + q_2 \mathbf{j} + q_3 \mathbf{k}$. Notamos a su *conjugado* como q^* y que se define como:

$$q^* = q_0 - \mathbf{q} = q_0 - q_1 \mathbf{i} - q_2 \mathbf{j} - q_3 \mathbf{k}.$$

De la definición obtenemos las siguientes propiedades básicas del manejo de cuaternios:

$$\begin{aligned} (q^*)^* &= q - (-\mathbf{q}) = q. \\ q + q^* &= q_0. \\ qq^* &= q^*q = (q_0 + \mathbf{q})(q_0 + \mathbf{q}) \\ &= q_0 q_0 - \langle -\mathbf{q}, \mathbf{q} \rangle + q_0 \mathbf{q} + (-\mathbf{q}) q_0 + (-\mathbf{q}) \times \mathbf{q} \\ &= q_0^2 + \langle \mathbf{q}, \mathbf{q} \rangle \\ &= q_0^2 + q_1^2 + q_2^2 + q_3^2 \\ &= q_0^2 + \|\mathbf{q}\|. \end{aligned}$$

En la última expresión $\|\cdot\|$ representa la norma usual de \mathbb{R}^3 . Podemos probar que dados dos cuaternion p, q se cumple que $(pq)^* = q^* p^*$. Efectivamente:

$$\begin{aligned} (pq)^* &= p_0 q_0 - \langle \mathbf{p}, \mathbf{q} \rangle - p_0 \mathbf{q} - q_0 \mathbf{p} - \mathbf{p} \times \mathbf{q} \\ &= p_0 q_0 - \langle \mathbf{p}, \mathbf{q} \rangle - p_0 \mathbf{q} - q_0 \mathbf{p} + \mathbf{q} \times \mathbf{p} \\ &= q_0 p_0 - \langle -\mathbf{q}, -\mathbf{p} \rangle + q_0 (-\mathbf{p}) + p_0 (-\mathbf{q}) + (-\mathbf{q}) \times (-\mathbf{p}) \\ &= (q_0 - \mathbf{q})(p_0 - \mathbf{p}) \\ &= q^* p^*. \end{aligned}$$

La *norma* de un cuaternion se nota como $|q|$ y se define como $|q| = \sqrt{q^* q}$. Usando la propiedad anterior de que el conjugado del producto es el producto de los conjugados cambiados de orden llegamos a:

$$\begin{aligned} |pq|^2 &= (pq)^*(pq) \\ &= q^* p^* pq \\ &= q^* |p|^2 q \\ &= |p|^2 q^* q \\ &= |p|^2 |q|^2. \end{aligned}$$

Por su parte, el *inverso* de un cuaternion se define como:

$$q^{-1} = \frac{q^*}{|q|^2}.$$

Se cumple que:

$$qq^{-1} = q^{-1}q = \frac{q^*}{|q|^2}q = \frac{|q|^2}{|q|^2} = 1.$$

En el caso de que q sea unitario (tenga norma uno) tenemos que $q^{-1} = q^*$. Notar que los cuaternios unitarios forman un grupo con operación interna el producto.

10.1.3. Cuaternios y rotaciones

El grupo de rotaciones en \mathbb{R}^3 alrededor de un eje que pasa por el origen, y que lo notamos como $SO(3)$, está formado por las matrices reales 3×3 que son ortogonales (su inversa coincide con la traspuesta) y tienen determinante igual a 1.

Nos surge entonces la siguiente pregunta: ¿por qué usar los cuaternios para representar dichas rotaciones? En primer lugar, el uso de matrices resulta redundante. Las rotaciones en \mathbb{R}^3 tienen básicamente tres grados de libertad: dos para la representación del eje (dado en coordenadas esféricas, por ejemplo) y uno para el ángulo de giro. Por ello, al usar matrices estamos usando demasiados elementos. Con el uso de los cuaternios seguimos teniendo redundancia pero solo por un valor más. Además, a partir de un eje de rotación y un ángulo de giro se puede construir fácilmente el cuaternion asociado, y del mismo modo, obtener el eje y ángulo a partir del cuaternion, mientras que en matrices dicha relación no es tan clara. También reducimos el número de operaciones necesarias a la hora de componer dos rotaciones. En la multiplicación de matrices se necesitan 27 multiplicaciones y 18 sumas y en el producto de cuaternios (veremos que esta es la operación de composición) 16 multiplicaciones y 20 sumas. Por otro lado, podemos usar los *ángulos de Euler* para identificar las rotaciones que utiliza el número de valores mínimo. Sin embargo, con este sistema podemos llegar a una situación no deseada que se conoce como *gimbal lock* o *bloqueo del cardán* [9]. Este fenómeno se produce cuando los tres ejes están en el mismo plano y nos produce la pérdida de uno de los ejes de giro. Por estas razones, entre otras, los cuaternios se conocen generalmente como la “mejor” manera para representar las rotaciones en el espacio. Más específicamente, la representación de las rotaciones en tres dimensiones se hace mediante cuaternios unitarios. Veamos el proceso:

Sea q un cuaternion unitario de la forma $q = q_0 + \mathbf{q}$. Al tener norma uno tenemos que $|q|^2 = q_0^2 + \|\mathbf{q}\|^2 = 1$. Así, como $(q_0, \|\mathbf{q}\|) \in \mathbb{S}^1$, existe un único ángulo $\theta \in [0, 2\pi]$ que cumple:

$$\begin{cases} \cos^2 \theta = q_0^2 \\ \sin^2 \theta = \|\mathbf{q}\|^2, \end{cases}$$

y podemos escribir ahora q en términos de θ y \mathbf{u} con $\mathbf{u} = \mathbf{q}/\|\mathbf{q}\|$ de la siguiente manera: $q = \cos \theta + \mathbf{u} \sin \theta$.

Dado el cuaternion unitario q , definimos la aplicación L_q dada por:

$$\begin{aligned} L_q : \mathbb{R}^3 &\longrightarrow \mathbb{R}^3 \\ \mathbf{v} &\longmapsto L_q(\mathbf{v}) = q\mathbf{v}q^* = (q_0^2 - \|\mathbf{q}\|^2)\mathbf{v} + 2\langle \mathbf{q}, \mathbf{v} \rangle \mathbf{q} + 2q_0(\mathbf{q} \times \mathbf{v}) \end{aligned}$$

La aplicación L_q está bien definida y no tenemos problema en multiplicar un vector por un cuaternion, ya que podemos ver un vector como un cuaternion puro. Hacemos ahora dos observaciones interesantes que nos hacen pensar, en principio, que la aplicación L_q actúa como una rotación con eje \mathbf{q} .

Observación 10.1. *La aplicación L_q no cambia la norma de $\mathbf{v} \in \mathbb{R}^3$.*

En efecto:

$$\|L_q(\mathbf{v})\| = \|q\mathbf{v}q^*\| = |q| \|\mathbf{v}\| |q^*| = \|\mathbf{v}\|.$$

Observación 10.2. *Si $\mathbf{v} = k\mathbf{q}$ con $k \in \mathbb{R}$ entonces la dirección de \mathbf{v} no cambia al aplicarle L_q .*

Usando la definición de L_q y al ser q unitario llegamos a:

$$\begin{aligned} L_q(\mathbf{v}) &= q\mathbf{v}q^* \\ &= (q_0^2 - \|\mathbf{q}\|^2)(k\mathbf{q}) + 2\langle \mathbf{q}, k\mathbf{q} \rangle \mathbf{q} + 2q_0(\mathbf{q} \times k\mathbf{q}) \\ &= k(q_0^2 - \|\mathbf{q}\|^2)\mathbf{q} + 2k\langle \mathbf{q}, \mathbf{q} \rangle \mathbf{q} + 2kq_0(\mathbf{q} \times \mathbf{q}) \\ &= k(q_0^2 - \|\mathbf{q}\|^2)\mathbf{q} + 2k\|\mathbf{q}\|^2 \mathbf{q} \\ &= k(q_0^2 + \|\mathbf{q}\|^2)\mathbf{q} \\ &= k\mathbf{q}. \end{aligned}$$

Antes de formalizar la “intuición” que nos han dado las observaciones anteriores necesitamos otra que es un poco más técnica.

Observación 10.3. *La aplicación L_q es lineal, es decir, dados $a_1, a_2 \in \mathbb{R}$ y $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^3$ se tiene que $L_q(a_1\mathbf{v}_1 + a_2\mathbf{v}_2) = a_1L_q(\mathbf{v}_1) + a_2L_q(\mathbf{v}_2)$.*

Se obtiene por la distributividad del producto:

$$\begin{aligned} L_q(a_1\mathbf{v}_1 + a_2\mathbf{v}_2) &= q(a_1\mathbf{v}_1 + a_2\mathbf{v}_2)q^* \\ &= q(a_1\mathbf{v}_1q^* + a_2\mathbf{v}_2q^*) \\ &= qa_1\mathbf{v}_1q^* + qa_2\mathbf{v}_2q^* \\ &= a_1q\mathbf{v}_1q^* + a_2q\mathbf{v}_2q^* \\ &= a_1L_q(\mathbf{v}_1) + a_2L_q(\mathbf{v}_2). \end{aligned}$$

El teorema que exponemos a continuación es clave y nos indica que dado un vector y ángulo podemos construir un cuaternion q tal que L_q representa la rotación deseada.

Teorema 10.1. Para cualquier cuaternionio $q = q_0 + \mathbf{q} = \cos \theta/2 + \mathbf{u} \operatorname{sen} \theta/2$ y para cualquier vector $\mathbf{v} \in \mathbb{R}^3$ tenemos que $L_q(\mathbf{v})$ es equivalente a la rotación con eje de giro \mathbf{u} y ángulo θ .

Demuestração. Tomamos $\mathbf{v} \in \mathbb{R}^3$ y lo expresamos en términos de \mathbf{q} y $\{\mathbf{q}\}^\perp$. De este modo, $\mathbf{v} = \mathbf{a} + \mathbf{n}$ donde \mathbf{a} es la componente respectiva a \mathbf{q} y \mathbf{n} la de $\{\mathbf{q}\}^\perp$. Bajo L_q , \mathbf{a} se queda invariante por la observación 10.2 mientras que \mathbf{n} rota un ángulo θ alrededor de \mathbf{q} . En efecto:

$$\begin{aligned} L_q(\mathbf{n}) &= (q_0^2 - \|\mathbf{q}\|^2)\mathbf{n} + 2\langle \mathbf{q}, \mathbf{n} \rangle \mathbf{q} + 2q_0(\mathbf{q} \times \mathbf{n}) \\ &= (q_0^2 - \|\mathbf{q}\|^2)\mathbf{n} + 2q_0(\mathbf{q} \times \mathbf{n}) \\ &= (q_0^2 - \|\mathbf{q}\|^2)\mathbf{n} + 2q_0 \|\mathbf{q}\| (\mathbf{u} \times \mathbf{n}), \end{aligned}$$

donde la última igualdad se debe a que $\mathbf{u} = \mathbf{q}/\|\mathbf{q}\|$. Notamos $\mathbf{n}_0 = \mathbf{u} \times \mathbf{n}$. Es claro por ello que \mathbf{n}_0 es perpendicular tanto a \mathbf{u} (y por ello también a \mathbf{q}) y a \mathbf{n} . Obtenemos:

$$L_q(\mathbf{n}) = (q_0^2 - \|\mathbf{q}\|^2)\mathbf{n} + 2q_0 \|\mathbf{q}\| \mathbf{n}_0. \quad (10.1)$$

Tanto \mathbf{n}_0 como \mathbf{n} tienen la misma longitud:

$$\|\mathbf{n}_0\| = \|\mathbf{n} \times \mathbf{u}\| = \|\mathbf{u} \times \mathbf{n}\| = \|\mathbf{n}\| \|\mathbf{u}\| \operatorname{sen}(\pi/2) = \|\mathbf{n}\|.$$

Escribimos 10.1 es términos de θ :

$$\begin{aligned} L_q(\mathbf{n}) &= (\cos^2 \frac{\theta}{2} - \operatorname{sen}^2 \frac{\theta}{2})\mathbf{n} + (2 \cos^2 \frac{\theta}{2} \operatorname{sen}^2 \frac{\theta}{2})\mathbf{n}_0 \\ &= \cos \theta \mathbf{n} + \operatorname{sen} \theta \mathbf{n}_0. \end{aligned}$$

Las últimas igualdades se han obtenido mediante las fórmulas del seno y coseno del ángulo doble. Vemos que el vector resultante es la rotación de \mathbf{n} un ángulo θ en el plano definido por \mathbf{n} y \mathbf{n}_0 . Claramente el vector resultante es perpendicular al eje de rotación, ya que \mathbf{n} y \mathbf{n}_0 pertenecen a $\{\mathbf{q}\}^\perp$. La demostración termina usando la linealidad que nos aporta la observación 3. \blacksquare

En resumen, dados $\mathbf{q} \in \mathbb{R}^3$ y $\theta \in \mathbb{R}$ representamos la rotación de ángulo de giro θ alrededor del eje \mathbf{q} por $q = \cos \frac{\theta}{2} + \frac{\mathbf{q}}{\|\mathbf{q}\|} \operatorname{sen} \frac{\theta}{2}$ y la aplicamos a un vector $\mathbf{u} \in \mathbb{R}^3$ mediante $L_q(\mathbf{u}) = \mathbf{q} \mathbf{u} q^*$.

Finalmente, exponemos la siguiente observación que nos indica la relación de la composición de rotaciones con su representación mediante cuaternios.

Observación 10.4. Sean p, q dos cuaternios unitarios, entonces $L_q \circ L_p = L_{qp}$.

Como p y q son unitarios entonces pq también lo es. Notamos $L_p(\mathbf{u}) = \mathbf{v}$ y $L_q(\mathbf{v}) = \mathbf{w}$. Obtenemos:

$$\begin{aligned} \mathbf{w} &= L_q(\mathbf{v}) = q \mathbf{v} q^* \\ &= q(p u p^*) q^* \\ &= (qp) \mathbf{u} (qp)^* \\ &= L_{qp}(\mathbf{u}). \end{aligned}$$

De este modo, hemos obtenido que la composición de rotaciones representadas en cuaternios se obtiene mediante la multiplicación de los cuaternios que representa cada una de las rotaciones por separado.

10.2. Cuaternios y optimización

Sea $P = \{\mathbf{p}_i\}$ un conjunto con N_p puntos que queremos alinear tomando como sistema de referencia otro conjunto $X = \{\mathbf{x}_i\}$ de tamaño N_x . Suponemos que $N_p = N_x$ y que cada $\mathbf{p}_i \in P$ corresponde con $\mathbf{x}_j \in X$ siempre y cuando $i = j$. Esta correspondencia hace referencia a que conocemos cómo se relacionan los puntos de ambos conjuntos y por ello p_i es el punto x_i pero en un sistema de referencia diferente. Aclararemos esta relación en apartados siguientes. Suponer que ambos conjuntos de puntos tienen el mismo tamaño puede parecer bastante restrictivo en un primer momento pero posteriormente veremos que no es el caso. De este modo, el problema consiste en encontrar una rotación R y una traslación \mathbf{b} que nos proporcionen las distancias más pequeñas entre los puntos. Así, la función objetivo a ser minimizada es:

$$F(R, \mathbf{b}) = \sum_{i=1}^{N_p} \|R\mathbf{p}_i + \mathbf{b} - \mathbf{x}_i\|^2. \quad (10.2)$$

En primer lugar, obtenemos los centroides de ambos conjuntos de puntos. Recorriendo que $N_p = N_x$, los notamos como:

$$\bar{\mathbf{p}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \mathbf{p}_i, \quad \bar{\mathbf{x}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \mathbf{x}_i.$$

A continuación, obtenemos las coordenadas relativas a los centroides:

$$\mathbf{p}'_i = \mathbf{p}_i - \bar{\mathbf{p}}, \quad \mathbf{x}'_i = \mathbf{x}_i - \bar{\mathbf{x}}.$$

Notamos que se cumple que:

$$\begin{aligned} \sum_{i=1}^{N_p} \mathbf{p}'_i &= \sum_{i=1}^{N_p} (\mathbf{p}_i - \bar{\mathbf{p}}) = \sum_{i=1}^{N_p} \mathbf{p}_i - N_p \bar{\mathbf{p}} = \sum_{i=1}^{N_p} \mathbf{p}_i - N_p \frac{1}{N_p} \sum_{i=1}^{N_p} \mathbf{p}_i = 0, \\ \sum_{i=1}^{N_p} \mathbf{x}'_i &= \sum_{i=1}^{N_p} (\mathbf{x}_i - \bar{\mathbf{x}}) = \sum_{i=1}^{N_p} \mathbf{x}_i - N_p \bar{\mathbf{x}} = \sum_{i=1}^{N_p} \mathbf{x}_i - N_p \frac{1}{N_p} \sum_{i=1}^{N_p} \mathbf{x}_i = 0. \end{aligned} \quad (10.3)$$

Expresando ahora (10.2) en términos de \mathbf{p}_i , \mathbf{x}_i , $\bar{\mathbf{p}}$ y $\bar{\mathbf{x}}$:

$$\begin{aligned} \sum_{i=1}^{N_p} \|R\mathbf{p}_i + \mathbf{b} - \mathbf{x}_i\|^2 &= \sum_{i=1}^{N_p} \|R(\mathbf{p}'_i + \bar{\mathbf{p}}) + \mathbf{b} - (\mathbf{x}'_i + \bar{\mathbf{x}})\|^2 \\ &= \sum_{i=1}^{N_p} \|R\mathbf{p}'_i - \mathbf{x}'_i + R\bar{\mathbf{p}} + \mathbf{b} - \bar{\mathbf{x}}\|^2 \\ &= \sum_{i=1}^{N_p} \langle R\mathbf{p}'_i - \mathbf{x}'_i + R\bar{\mathbf{p}} + \mathbf{b} - \bar{\mathbf{x}}, R\mathbf{p}'_i - \mathbf{x}'_i + R\bar{\mathbf{p}} + \mathbf{b} - \bar{\mathbf{x}} \rangle. \end{aligned}$$

Para simplificar la escritura, llamamos $a_i = R\mathbf{p}'_i - \mathbf{x}'_i$ y $c = R\bar{\mathbf{p}} + \mathbf{b} - \bar{\mathbf{x}}$. Como el producto escalar es bilineal obtenemos:

$$\begin{aligned} \sum_{i=1}^{N_p} \langle a_i + c, a_i + c \rangle &= \sum_{i=1}^{N_p} \langle a_i + c, a_i \rangle + \sum_{i=1}^{N_p} \langle a_i + c, c \rangle \\ &= \sum_{i=1}^{N_p} \langle a_i, a_i \rangle + \sum_{i=1}^{N_p} \langle c, a_i \rangle + \sum_{i=1}^{N_p} \langle a_i, c \rangle + \sum_{i=1}^{N_p} \langle c, c \rangle \\ &= \sum_{i=1}^{N_p} \|a_i\|^2 + 2 \sum_{i=1}^{N_p} \langle a_i, c \rangle + \sum_{i=1}^{N_p} \|c\|^2 \\ &= \sum_{i=1}^{N_p} \|a_i\|^2 + 2 \sum_{i=1}^{N_p} \langle a_i, c \rangle + N_p \|c\|^2 \\ &= \sum_{i=1}^{N_p} \|a_i\|^2 + 2 \langle \sum_{i=1}^{N_p} a_i, c \rangle + N_p \|c\|^2. \end{aligned}$$

Si sustituimos a_i por su valor en el término central y usamos (10.3):

$$\sum_{i=1}^{N_p} a_i = \sum_{i=1}^{N_p} (R\mathbf{p}'_i - \mathbf{x}'_i) = \sum_{i=1}^{N_p} R\mathbf{p}'_i - \sum_{i=1}^{N_p} \mathbf{x}'_i = 0.$$

Por lo tanto:

$$\sum_{i=1}^{N_p} \|R\mathbf{p}_i + \mathbf{b} - \mathbf{x}_i\|^2 = \sum_{i=1}^{N_p} \|R\mathbf{p}'_i - \mathbf{x}'_i\|^2 + N_p \|R\bar{\mathbf{p}} + \mathbf{b} - \bar{\mathbf{x}}\|^2.$$

La translación \mathbf{b} buscada debería hacer 0 el segundo término obteniendo entonces:

$$\mathbf{b} = -R\bar{\mathbf{p}} + \bar{\mathbf{x}}.$$

Ahora, para obtener la mejor rotación utilizamos el primero término. Reescribiendo la misma llegamos a:

$$\begin{aligned}
\sum_{i=1}^{N_p} \|R\mathbf{p}'_i - \mathbf{x}'_i\|^2 &= \sum_{i=1}^{N_p} \langle R\mathbf{p}'_i - \mathbf{x}'_i, R\mathbf{p}'_i - \mathbf{x}'_i \rangle \\
&= \sum_{i=1}^{N_p} \langle R\mathbf{p}'_i, R\mathbf{p}'_i \rangle - 2 \sum_{i=1}^{N_p} \langle R\mathbf{p}'_i, \mathbf{x}'_i \rangle + \sum_{i=1}^{N_p} \langle \mathbf{x}'_i, \mathbf{x}'_i \rangle \\
&= \sum_{i=1}^{N_p} RR^T \|\mathbf{p}'_i\|^2 - 2 \sum_{i=1}^{N_p} \langle R\mathbf{p}'_i, \mathbf{x}'_i \rangle + \sum_{i=1}^{N_p} \|\mathbf{x}'_i\|^2 \\
&= \sum_{i=1}^{N_p} (\|\mathbf{p}'_i\|^2 + \|\mathbf{x}'_i\|^2) - 2 \sum_{i=1}^{N_p} \langle R\mathbf{p}'_i, \mathbf{x}'_i \rangle.
\end{aligned}$$

El primer término no depende de la rotación por lo que no influirá en el cálculo. Y el segundo, que sí depende de R , hará que $\sum_{i=1}^{N_p} \|R\mathbf{p}'_i - \mathbf{x}'_i\|^2$ sea mínimo cuando tome su valor máximo al estar restando. Nos encontramos ahora en un problema de maximización. Usamos ahora la representación en cuaternios de las rotaciones y que hemos introducido en la sección 10.1.3. Sea q el cuaternion (unitario) que representa la misma rotación que la matriz R . Tenemos que maximizar:

$$\sum_{i=1}^{N_p} \langle q\mathbf{p}'_i q^*, \mathbf{x}'_i \rangle.$$

Aplicando la definición del producto de cuaternios no es difícil probar que:

$$\sum_{i=1}^{N_p} \langle q\mathbf{p}'_i q^*, \mathbf{x}'_i \rangle = \sum_{i=1}^{N_p} \langle q\mathbf{p}'_i, \mathbf{x}'_i q \rangle. \quad (10.4)$$

Visto q como un vector de \mathbb{R}^4 tenemos que $q = (q_0, q_1, q_2, q_3)^T$. También vemos los puntos \mathbf{p}'_i y \mathbf{x}'_i como cuaternios puros notando a sus componentes como $\mathbf{p}'_i = (0, p'_{i1}, p'_{i2}, p'_{i3})$ y $\mathbf{x}'_i = (0, x'_{i1}, x'_{i2}, x'_{i3})$ para $i = 1, \dots, N_p$.

Nuestro propósito ahora es escribir los sumandos de (10.4) como producto matricial. Construimos las matrices para cada $i = 1, \dots, N_p$:

$$P_i = \begin{pmatrix} 0 & -p'_{i1} & -p'_{i2} & -p'_{i3} \\ p'_{i1} & 0 & p'_{i3} & -p'_{i2} \\ p'_{i2} & -p'_{i3} & 0 & p'_{i1} \\ p'_{i3} & p'_{i2} & -p'_{i1} & 0 \end{pmatrix}, X_i = \begin{pmatrix} 0 & -x'_{i1} & -x'_{i2} & -x'_{i3} \\ x'_{i1} & 0 & -x'_{i3} & x'_{i2} \\ x'_{i2} & x'_{i3} & 0 & -x'_{i1} \\ x'_{i3} & -x'_{i2} & x'_{i1} & 0 \end{pmatrix}. \quad (10.5)$$

Observamos que $q\mathbf{p}'_i$ coincide con $P_i q$ y que $\mathbf{x}'_i q$ hace lo mismo con $X_i q$, sustituimos

en (10.4):

$$\begin{aligned} \sum_{i=1}^{N_p} \langle q p'_i, x'_i q \rangle &= \sum_{i=1}^{N_p} \langle P_i q, X_i x \rangle \\ &= \sum_{i=1}^{N_p} \langle q, P_i^T X_i q \rangle \\ &= \langle q, \left(\sum_{i=1}^{N_p} P_i^T X_i \right) q \rangle. \end{aligned}$$

Podemos comprobar sin dificultad que la matriz $P_i^T X_i$ es simétrica por lo que la suma de ellas también lo es. Llameamos a su suma

$$M = \sum_{i=1}^{N_p} P_i^T X_i. \quad (10.6)$$

Como M es simétrica podemos asegurar que todos sus valores propios son reales. Sean $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4$ dichos valores propios (tenemos en cuenta las multiplicidades) y $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4$ sus correspondientes vectores propios unitarios asociados. Sabemos que vectores propios asociados a diferentes valores propios son perpendiculares entre sí. Si corresponden al mismo podemos elegirlos de tal manera para que también sean ortogonales. Por lo tanto, podemos escribir el cuaternio q como combinación lineal de vectores propios:

$$q = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3 + \alpha_4 \mathbf{v}_4,$$

tenemos entonces que la matriz M expresada en la base $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4\}$ de sus vectores propios es diagonal con $M = \text{diag}(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$. Así:

$$\begin{aligned} \langle q, Mq \rangle &= \langle \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3 + \alpha_4 \mathbf{v}_4, M(\alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3 + \alpha_4 \mathbf{v}_4) \rangle \\ &= \langle \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3 + \alpha_4 \mathbf{v}_4, \lambda_1 \alpha_1 \mathbf{v}_1 + \lambda_2 \alpha_2 \mathbf{v}_2 + \lambda_3 \alpha_3 \mathbf{v}_3 + \lambda_4 \alpha_4 \mathbf{v}_4 \rangle \\ &= \lambda_1 \alpha_1^2 + \lambda_2 \alpha_2^2 + \lambda_3 \alpha_3^2 + \lambda_4 \alpha_4^2. \end{aligned}$$

De este modo, $\langle q, Mq \rangle$ es máximo cuando $\alpha_1^2 = 1$ y $\alpha_2^2 = \alpha_3^2 = \alpha_4^2 = 0$ ya que al ser q unitario debe cumplir $\alpha_1^2 + \alpha_2^2 + \alpha_3^2 + \alpha_4^2 = 1$. Así, el cuaternio que maximiza (10.4) es el vector propio asociado al máximo valor propio de M .

10.3. Algoritmo ICP

Una vez explicadas las herramientas que intervienen en el proceso de minimización de la función (10.2), explicamos el algoritmo tal y como fue propuesto por Paul J.Besl and Neil D.MacKay. Destacamos una sutil diferencia entre la construcción original y la que nosotros hemos presentado. Besl and MacKay identifican la matriz M que hemos definido en (10.6) como:

$$M = Q(\Sigma_{px}) = \begin{pmatrix} \text{tr}(\Sigma_{px}) & \Delta^t \\ \Delta & \Sigma_{px} + \Sigma_{px}^t + \text{tr}(\Sigma_{px})I_3 \end{pmatrix}, \quad (10.7)$$

donde

$$\Sigma_{px} = \frac{1}{N_p} \sum_{i=1}^{N_p} [(\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{x}_i - \bar{\mathbf{x}})^t],$$

$$A_{ij} = (\Sigma_{px} - \Sigma_{px}^t)_{ij} \quad i, j \in \{0, 1, 2\},$$

y

$$\Delta = \begin{pmatrix} A_{12} & A_{20} & A_{01} \end{pmatrix}.$$

No es difícil comprobar que ambas definiciones son equivalentes haciendo la construcción elemento a elemento. Destacar que se han presentado estas dos definiciones de la misma matriz, ya que la dada por (10.6) es más fácil de comprender en el proceso de minimización. Por su parte, (10.7) será la que se implementará en el código del algoritmo al ser la propuesta original.

Así, con todo los mecanismos a nuestra disposición, la solución propuesta para alinear ambas nubes de puntos es la siguiente:

1. Sea el conjunto de puntos P a alinear con el modelo X . Notar que no tienen porqué tener el mismo tamaño.
2. Inicializaremos el conjunto $P_0 = P$. Para la iteración $k \geq 0$, repetir a), b) y c) hasta que haya convergencia con una tolerancia τ .
 - a) Calcular el conjunto de Y_k de los puntos más cercanos de P_k respecto a X . Notamos como e_k a la distancia media entre Y_k y P_k obtenida mediante (10.2).
 - b) Aplicar el método de optimización de la distancia media (sección 10.2) para obtener los cuaternios qT_k y qR_k .
 - c) Obtener el conjunto $P_{k+1} = R(qR_k)P_0 + qT_k$. Calcular d_k que nota la distancia media entre Y_k y $P_k + 1$ obtenida mediante (10.2).
 - d) Terminar cuando $d_{k-1} - d_k < \tau$.

En el pseudocódigo hemos notado como $R(qR_k)$ a la rotación asociada al cuaternion qR_k de la iteración k-ésima. Notar que cada $y_{i,k}$ es el punto más cercano asociado a $p_{i,k}$ para $i = 1, \dots, N_p$. De este modo, queda clara la nota que apuntamos al inicio de la sección y es que podemos suponer que los conjuntos tienen el mismo número de puntos, ya que el proceso no se realiza con el modelo X sino con el conjunto de puntos más cercanos.

Exponemos ahora el teorema que nos asegura la convergencia del método sugerido, que en este caso es convergencia local. Esto nos impondrá una serie de restricciones que explicaremos más adelante.

Teorema 10.2. *El algoritmo ICP siempre converge de manera monótona a un mínimo local respecto de la función objetivo distancia media.*

Demostración. Tenemos el conjunto P_0 que queremos ajustar al modelo X . Sea $P_k = R(qR_{k-1})P_0 + qT_{k-1}$ el conjunto de puntos transformados al principio de la iteración k y sea Y_k el conjunto de puntos más cercanos a cada punto con $y_{i,k}$

el punto más cercano asociado a $p_{i,k}$ para $i = 1, \dots, N_p$ que hemos calculado. Definimos:

$$e_k = \frac{1}{N_p} \sum_{i=1}^{N_p} \|y_{i,k} - p_{i,k}\|.$$

Aplicamos el proceso explicado en la sección 10.2 y obtenemos qT_k y qR_k . Calculamos:

$$d_k = \frac{1}{N_p} \sum_{i=1}^{N_p} \|y_{i,k} - R(qR_k)p_{i,k} - qT_k\|.$$

Afirmamos que $d_k \leq e_k$ para la iteración k -ésima. De lo contrario, si $d_k > e_k$ tendríamos que la distancia media del conjunto inicial es menor que la distancia media tras aplicar el proceso descrito de optimización lo cual es imposible por la propia construcción del mismo.

En la siguiente iteración, calculamos $P_{k+1} = R(qR_k)P_k + qT_k$. Obtenemos ahora el Y_{k+1} y e_{k+1} igual que en la anterior iteración. Es evidente que $e_{k+1} \leq d_k$ por la propia definición de Y_{k+1} . Notar que para toda iteración k se cumple $e_k, d_k \geq 0$ al ser la suma de números no negativos por lo que hemos conseguido la siguiente relación:

$$0 \leq d_{k+1} \leq e_{k+1} \leq d_k \leq e_k$$

Finalmente como la sucesión $\{d_k\}_{k \geq 1}$ es decreciente y acotada podemos afirmar que es convergente y como consecuencia que el método converge de manera monótona a un mínimo local. ■

Una vez demostrada la convergencia del método, es necesario recalcar una serie de observaciones acerca del mismo.

Observación 10.5. *Al converger a un mínimo local es necesario una fase de prealineado*

La demostración del método asegura que el método es convergente pero no nos aporta el elemento al que converge. Por ello, al aplicar el método podemos llegar a una situación donde el procedimiento converja a una mejora en la transformación pero no sea la que nosotros deseamos. Necesitamos entonces la fase de *prealineado*. Este proceso se puede llevar a cabo de diferentes maneras. La más habitual es dar manualmente una serie de puntos clave que se correspondan en los dos conjuntos y aplicar una transformación de acuerdo a los mismos. Tenemos así una primera aproximación de ambos conjuntos de puntos para asegurar que la convergencia se produce a la situación que deseamos. En nuestro caso, dicho procedimiento se ha implementado del siguiente modo:

1. Escoger de cada conjunto tres puntos que se encuentren en ambos y sean fácilmente identificables al pertenecer a una esquina, tener el mismo color, etc. Estos puntos deben de estar ordenados de la misma forma en ambos casos, es decir, el primer punto seleccionado de un conjunto debe “coincidir” con el primero del otro. Igual con los dos restantes. Este proceso se recoge en la figura 10.2.

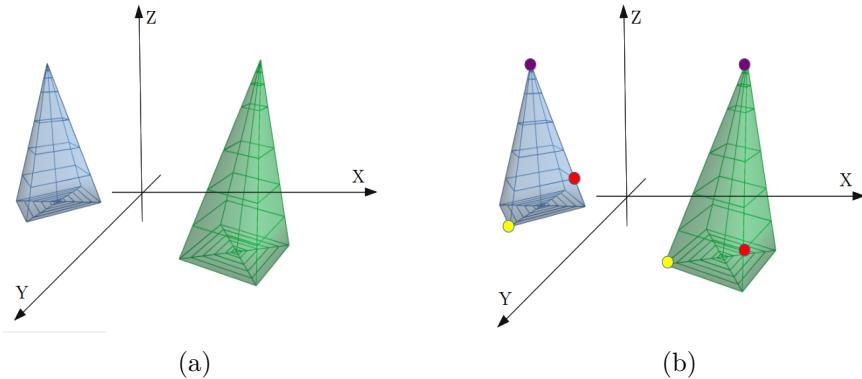


Figura 10.2: Selección de puntos en ambos conjuntos.

2. Llevar los primeros puntos seleccionados al origen de coordenadas.
3. Llevar los segundos a la parte positiva del eje X manteniendo fijos los anteriores.
4. Intentar aproximar lo máximo posible los otros puntos mediante una rotación alrededor del eje X .

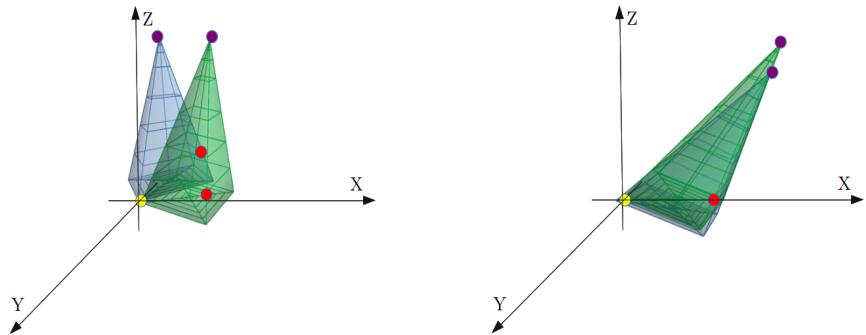
La figura 10.3 muestra el proceso a seguir en estos últimos pasos. En secciones posteriores veremos un ejemplo práctico de este procedimiento.

Observación 10.6. *El cálculo de los puntos más cercanos entre las nubes de puntos es un proceso muy costoso en tiempo.*

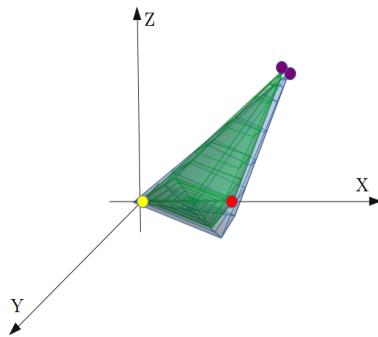
Este paso del algoritmo es el que más tiempo ocupa de todo el proceso al tener una complejidad $O(N_p^2)$ en el peor de los casos. También tenemos que tener en cuenta el tamaño de los conjuntos de los puntos. La finalidad de este proceso es tener un modelo digital detallado y fiel a la realidad de un objeto real. Ello conlleva que las tomas se realicen con cierta calidad lo que hace que, incluso en modelos relativamente pequeños, cada una ronde los 2 millones de puntos. Si es un modelo más complejo podríamos llegar a los 20 millones. En estas situaciones, aplicar el método tradicional podría durar horas en completarse.

En esta situación, nos planteamos la posibilidad de una variante para no tener que comprobar todos y cada uno de los puntos en el proceso. En una primera aproximación, podríamos pensar en coger un subconjunto aleatorio de puntos de cada toma y aplicar el proceso descrito. Sin embargo, comprobamos que esta solución no es válida. Esto se debe a la convergencia local del método lo que nos conduce a las situaciones indeseadas que queremos evitar con el prealineado. Por ello, nos tenemos que plantear mecanismos más complejos que aseguren que se tengan en cuenta las tomas completas.

Un primer método sería usar índices espaciales como los *octrees*. Los *octrees* son estructuras jerárquicas que dividen el espacio en cubos denominados *voxels*. Cada



(a) Paso 2: llevar los primeros puntos al (b) Paso 3: llevar los segundos puntos al
origen. eje X .



(c) Paso 4: aproximar lo máximo posible el punto restante.

Figura 10.3: Mecanismo de prealineado a partir de tres puntos.

voxel puede contener un máximo número de puntos por lo que si se supera dicho número el *voxel* se dividiría en otros *voxels* más pequeños. También podríamos imponer una profundidad máxima del árbol. En esta situación, solo deberíamos comprobar la distancia con un punto de cada *voxel* y una vez se tiene, filtrar los puntos del modelo por esa distancia.

Otro mecanismo sería identificar alguna característica que nos permita clasificar los puntos. Como las tomas corresponden al mismo modelo, podemos suponer que la característica que buscamos se mantienen en cierta medida en ambos casos y solo tenemos que buscar el punto más cercano con aquellos que tienen características similares. Esta será la opción que nosotros usaremos para proponer alternativas basadas en este procedimiento y que veremos en la sección 11.1 .

10.4. Ejemplos prácticos

En esta sección procedemos a probar el método implementado. Empezamos con un ejemplo parecido al artículo original. Los conjuntos de puntos para la prueba son:

x_1	x_2	x_3
0.4389	-0.0588	1.0699
0.4202	0.2052	1.1252
0.4201	0.2539	1.1325
0.4495	0.0469	1.1260
0.4412	0.1796	1.1515
0.4826	-0.0137	1.1359
0.4628	0.0703	1.1458
0.4700	0.1852	1.1765

Tabla 10.1: Conjunto 1.

y_1	y_2	y_3
0.7278	0.0712	1.4610
0.7019	0.2480	1.4867
0.7621	0.1828	1.4720
0.7271	0.1769	1.4809
0.7067	0.0462	1.495
0.6438	-0.0540	1.4359
0.7247	-0.0116	1.4385
0.6982	0.1981	1.4832
0.7700	0.2500	1.5000
0.8000	-0.100	1.400
0.8300	0.3000	1.4500

Tabla 10.2: Conjunto 2.

En la imagen 10.4 se muestran los puntos antes y después de aplicar una iteración de ICP. Notar que los del primer conjunto se han dibujado de color rojo y los del segundo de color verde. En esta ocasión no ha sido necesaria la etapa de prealineado al no haber una situación deseada tras la transformación.

Según los datos calculados, la distancia media en milímetros ha pasado de 403.3571 a 61.0297. El tiempo que ha tardado en realizar el proceso es de 0.00226548 segundos. La traslación y rotación expresada como cuaternion vienen dadas por

$$q_T = (0.166271, 0.165485, 0.349303)$$

y

$$q_R = (0.0234481, 0.0238614, -0.154144, 0.987482).$$

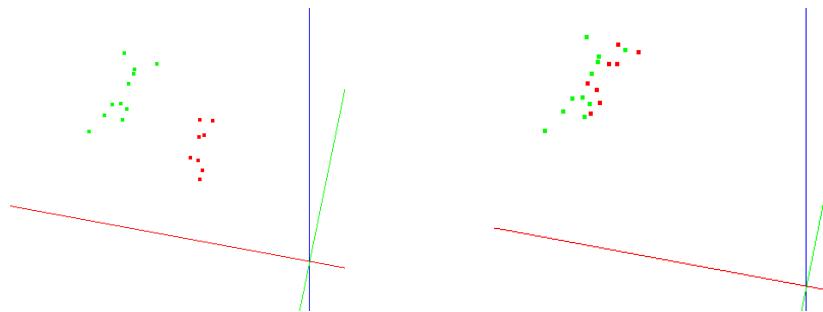
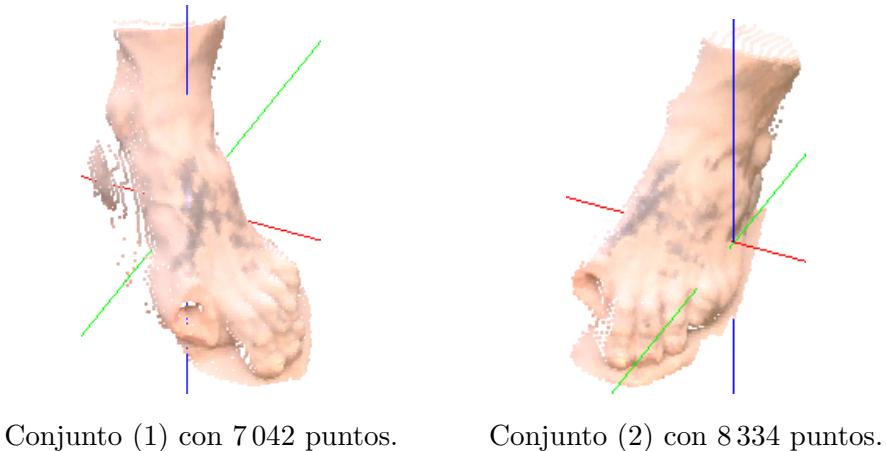


Figura 10.4: Conjuntos de puntos antes y después de una iteración de ICP.

El segundo ejemplo que vamos a probar consiste en un caso real, concretamente dos tomas de una escultura de un pie perteneciente a la Facultad de Bellas Artes de la Universidad de Granada tomadas con un escáner láser Faro Focus 130. Queremos alinear dos tomas obtenidas desde diferentes ángulos:



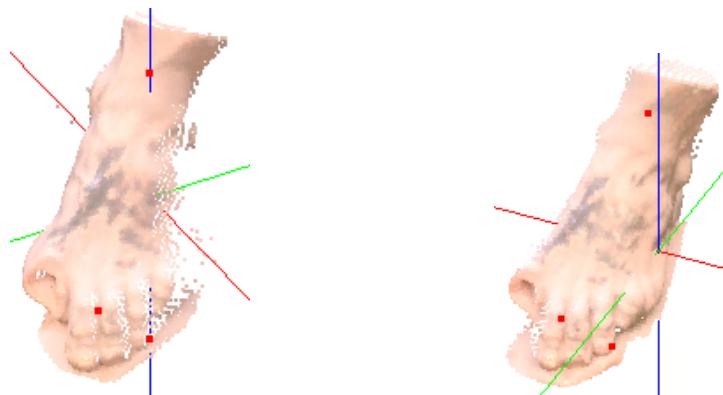
Conjunto (1) con 7 042 puntos. Conjunto (2) con 8 334 puntos.

Figura 10.5: Tomas a alinear.

Tras la etapa de prealineado (los puntos seleccionados aparecen en la figura 10.6) se ha obtenido el resultado que muestran la figura 10.7.

A continuación, comenzamos usamos el algoritmo ICP. La tabla 10.3 recoge los datos tras cada iteración. Destacar que con distancia nos referimos a la distancia media entre los puntos y los más cercanos, es decir, lo que hemos notado como e_k y d_k en la sección 10.3. Por su parte, la figura 10.8, muestra el resultado final tras las cinco iteraciones llevadas a cabo.

En conclusión, vemos cómo el programa propuesto ha sido capaz de realizar un buen ajuste y por ello hemos conseguido un modelo más completo al original uniendo ambas tomas. Destacamos nuevamente la importancia que tiene en el proceso el prealineado. No solo la necesidad de hacerlo sino también la manera en la



Puntos clave del primer conjunto. Puntos clave del segundo conjunto.

Figura 10.6: Etapa de prealineado.

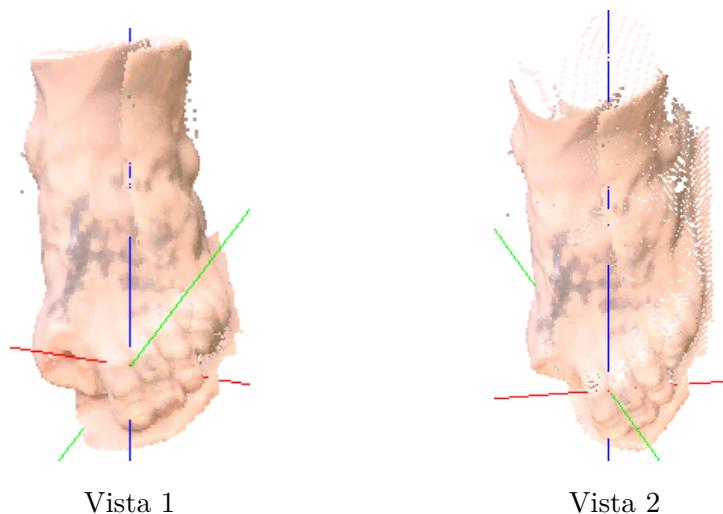


Figura 10.7: Vistas tras prealinear.

Iteración	Dist. antes (mm)	Dist. después (mm)	Segundos
1	21.2000	18.1299	62.8672
2	16.7174	15.8707	62.7888
3	15.1625	14.4617	62.8636
4	13.9753	13.7418	63.059
5	13.6518	13.6071	62.8259

Tabla 10.3: Resultados ajuste mediante ICP.

que sea lleva a cabo este proceso. Es posible que incluso incluyendo esta etapa el resultado no sea el deseado. Este comportamiento se debe como ya hemos indicado a la convergencia local del método ICP. Las dos nubes de puntos tienen que estar

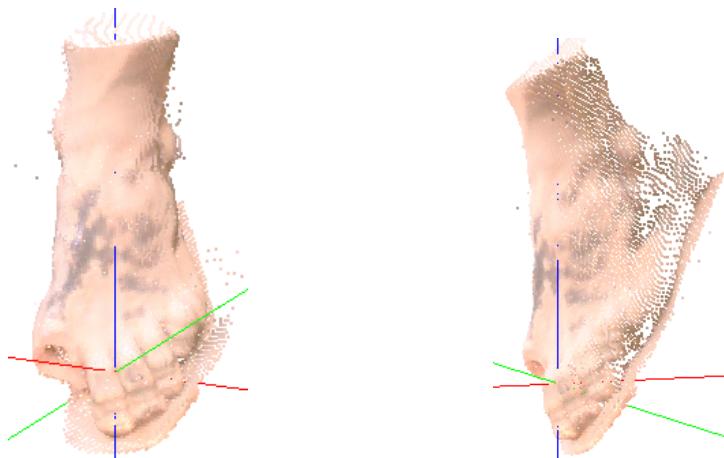


Figura 10.8: Vistas tras cinco iteraciones del ICP.

los no solo lo suficientemente cerca sino también colocadas de tal manera que el algoritmo propuesto solo deba realizar un pequeño ajuste.

Por otro lado, también vemos que el tiempo empleado es elevado tal y como se indicó en la observación 10.6: cada iteración ha durado algo más de un minuto. Hay que tener en cuenta que las nubes de puntos tomadas como modelo tienen un tamaño muy pequeño. En situaciones reales las tomas pueden tener millones de puntos lo que conllevaría mucho tiempo a la hora de aplicar el algoritmo e incluso sin la certeza de si el resultado es el que deseamos. En la siguiente sección planteamos unos métodos para acelerar este proceso.

Cambios en el método: uso de normales

En la sección anterior nos encontramos con el siguiente problema: el método ICP funciona correctamente pero es demasiado lento, ya que tiene un orden cuadrático y manejamos gran cantidad de información. Nuestro objetivo es entonces reducir el conjunto de puntos con los que trabajamos sin perder información. Este proceso no puede ser aleatorio y debe ser lo más significativo posible, ya que, de lo contrario, podríamos obtener un resultado no deseado. Así usaremos en estos procedimientos los denominados *descriptores*. Un descriptor lo podríamos definir como una medida de un punto basándonos en cierta característica que escojamos. De este modo obtenemos los llamados *puntos significativos*, *puntos clave* o *key points* de cada una de las tomas. Como en nuestro marco de trabajo los modelos que tomamos como base son rígidos, es decir, no hay movimiento, cambios de escala, etc. podemos suponer que los puntos clave detectados en una toma deben de ser los mismos que en la otra. Por lo tanto, en una primera aproximación, si realizamos el proceso para buscar la mejor transformación solo con los puntos clave deberíamos conseguir nuestro objetivo.

Como observación, aclarar que del mismo modo que no existe un método que funcione correctamente en el alineado de cualquier modelo, no existe un descriptor que funcione de manera adecuada con todos los modelos. Por ejemplo, podríamos detectar puntos clave en relación con el color. Si nos encontramos ante una escultura con un ropaje colorido y detallado es un descriptor que podríamos considerar válido. Sin embargo, si nuestro modelo presenta un color plano o una variación poco significativa del mismo no conseguiremos obtener puntos significativos. También debemos de tener en cuenta el tiempo necesario para la detección de puntos clave. El proceso debe ser lo suficientemente rápido para que reduzcamos el tiempo de alineación de las tomas y lo suficientemente bueno para que el conjunto de puntos calculado sea pequeño y descriptivo dentro de toda la nube.

11.1. Variación de la normal

El método para la obtención de puntos consistirá en la detección de zonas donde se produzca un cambio brusco de la normal calculando de este modo esquinas, hendiduras, etc. Este procedimiento no es válido en superficies suaves, ya que la

variación de la normal en un entorno sería parecida en todo el modelo. Situación de este tipo nos la podemos encontrar, por ejemplo, en esferas. Una vez explicada la idea principal procedemos a explicar el proceso llevado a cabo.

En primer lugar, debemos calcular las normales de cada punto de la nube. Generalmente, es un proceso costoso aunque en nuestra situación tenemos una gran ventaja. La información aportada por escáner con el que tomamos cada una de las muestras nos aporta, en cierta medida, una malla implícita, ya que que puede entregar los datos de los puntos registrados siguiendo una cuadrícula en un archivo con formato PTX, que nos aporta el resultado por columnas por lo que podemos reconstruir la “matriz” espacial calculada previamente. Así con esta información para cada punto calculamos la normal mediante el producto vectorial de los triángulos adyacentes y luego promediamos obteniendo la de cada punto. En la figura 11.1, vemos a la izquierda la malla implícita que obtenemos con el formato que estamos trabajando. A la derecha aparece un ejemplo de cómo se han calculado las normales. En amarillo aparecen los vectores que se calculan (siempre y cuando existan dichos puntos) y en rojo los productos vectoriales necesarios. Destacamos que es importante que todas las normales tengan un convenio general de apuntar hacia dentro o hacia fuera del modelo. El nuestro ha sido de que apunten hacia afuera, lo que se ha tenido en cuenta durante la obtención de los productos vectoriales.

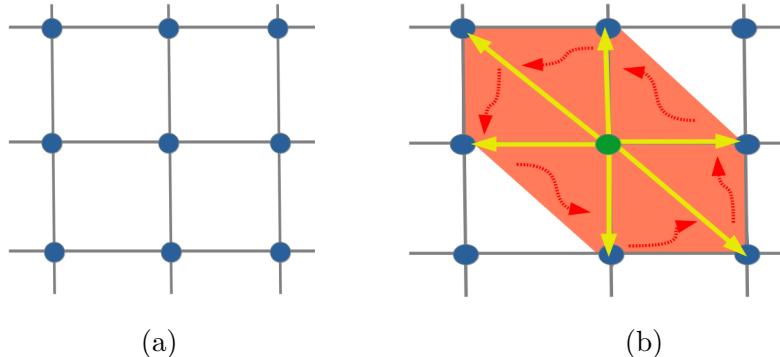


Figura 11.1: (a) Malla implícita, (b) Cálculo de la normal.

La idea del algoritmo propuesto se basa en que en zonas relativamente planas las normales son muy parecidas por lo que la suma los productos escalares de la normal en un punto con aquellos contenidos en un entorno suyo es prácticamente 0. Razonemos lo que pasa en los lugares en los que hay esquinas. Tomamos un punto y sus vecinos, que en general son un total de 8. Suponemos una situación ideal en la que el punto que hemos escogido está en un borde. Si es así, consideraremos que dos de los puntos adyacentes también están en dicho borde. Así, tenemos que de sus vecinos adyacentes, dos tienen una normal parecida y los seis restantes perpendicular. Si calculamos el producto escalar medio en esta situación ideal obtenemos que es de 0.25. Por ello, el algoritmo propuesto tiene en cuenta este valor y solo se queda con los puntos cuyo descriptor de normales esté en un rango próximo al valor comentado, que pertenecería a la situación ideal. En nuestros experimentos hemos usado como criterio que el valor del producto escalar medio esté entre 0.15

y 0.3.

A continuación mostramos un ejemplo de los resultados obtenidos. Para ello, hemos usado diferentes niveles de precisión. Esto se ha conseguido simplificando la malla implícita que tenemos del modelo e indicando el número de filas y columnas que queremos conservar. Por ejemplo, si tenemos un nivel de precisión de 5, toma como puntos válidos aquellos que están cada 5 filas y 5 columnas. De este modo, de cada 25 puntos nos quedamos con uno. Si el nivel de precisión es 8, nos quedamos con un punto de 64. En la figura 11.2 se muestran varios ejemplos de cómo varía la densidad de la malla en función del nivel de simplificado.

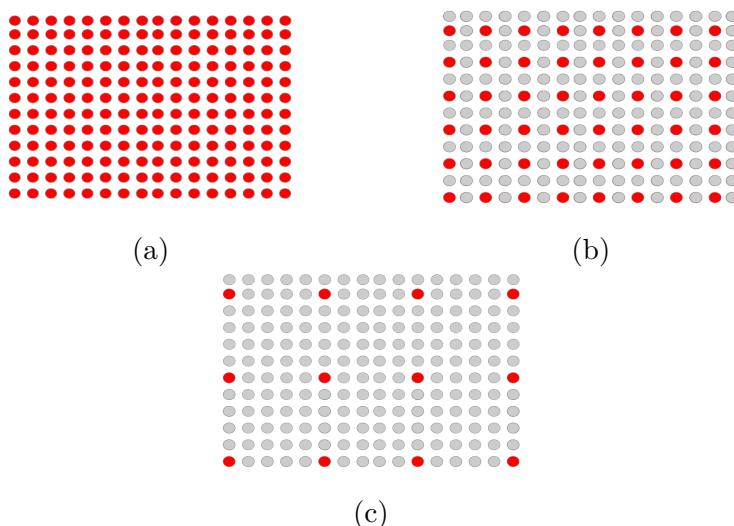


Figura 11.2: Ejemplos de simplificado de la malla: (a) sin simplificado, (b) valor 2 y (c) valor 5. En rojo aparecen marcados los puntos con los que nos quedamos en cada caso.

Los resultados obtenidos tras aplicar el algoritmo de detección de puntos claves se muestran en la tabla 11.1 y figura 11.3. Vemos que uno de los problemas que tiene este método es que muy sensible al ruido. Esto, unido a la alta densidad de puntos con la que estamos trabajando hace que detectemos una gran cantidad de puntos clave. Por lo tanto, si queremos que los puntos obtenidos tras el proceso sean significativos tenemos que calcularlos a partir del modelo simplificado. Esto no debe de ser un problema ya que esto solo lo usaremos en las primeras iteraciones del método ICP, ya que para obtener la solución deseada debemos tener en cuenta posteriormente el modelo entero debido a la sensibilidad del procedimiento.

11.2. Pruebas

11.2.1. De puntos clave a todo el modelo

La primera prueba que se va a realizar es aplicar el procedimiento a los puntos clave pero tomando dolo de parte de una toma que queremos alinear con otra toma

Nivel de simplificado	Puntos totales	Tiempo cálculo normales (seg.)	Tiempo cálculo ptos. clave (seg.)	Puntos clave
1	2 275 886	25.1373	13.0592	592 611
2	568 830	6.39203	3.2690	192 031
3	252 881	2.82000	1.4716	46 610
4	142 186	1.72909	0.8370	11 866
5	90 983	1.00018	0.5496	4490
8	35 519	0.394472	0.2151	1 885

Tabla 11.1: Resultados del cálculo de puntos clave. En la primera columna aparece el nivel de simplificado utilizado, en la segunda el número total de puntos de la malla (simplificada), en la tercera y cuarta se incluye el tiempo en segundos para el cálculo de las normales y el cálculo de los valores del descriptor respectivamente. En la última columna se aporta el número de puntos clave obtenidos en cada caso.

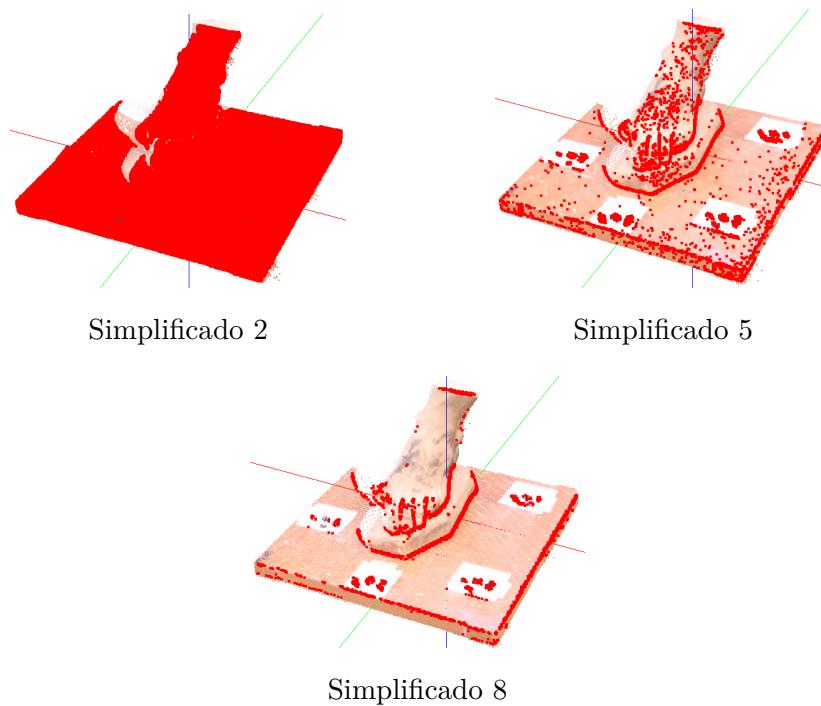
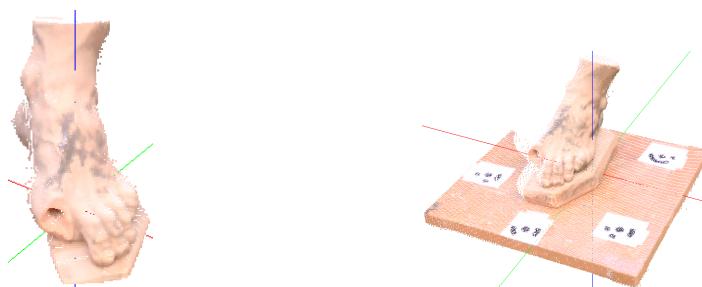


Figura 11.3: Descriptor normales según el nivel de simplificado.

completa que tomamos como modelo. Lo explicamos con nuestro ejemplo base del pie. Si probamos con las mismas tomas de la sección 10.4 en las que solo aparece el pie los puntos que nos aportará el descriptor mediante normales es posible que no tengan una correspondencia clara con la otra toma, sobre todo, los que se encuentran en el borde. Por eso, debemos tomar como modelo la toma completa

para asegurar en todo lo que posible que haya correspondencia entre los puntos clave aportados. Del mismo modo, también habría problemas si tomamos las dos tomas completas, ya que nuevamente es probable que algunos puntos no estén en el modelo, y al intentar ajustarlos, nos aporte una solución errónea. Destacamos que aplicar el procedimiento propuesto a una parte de las tomas no debe de ser restrictivo porque nos interesa conocer la transformación. Podríamos almacenar los datos de las transformaciones que se van realizando y posteriormente aplicarlo a la toma completa. Otra solución sería, por ejemplo, imponer una distancia máxima entre dos puntos para asegurarse de que realmente se corresponden. Esta solución será la que usaremos en la siguiente sección.



Conjunto (1) con 7 527 puntos.

Conjunto (2) con 35 519 puntos.

Figura 11.4: Tomas a alinear.

El número de puntos característicos detectados en la toma de la izquierda mediante el descriptor de las normales en el intervalo $(0.15, 0.3)$ es de 506. Aplicamos el procedimiento y los resultados obtenidos se muestran en la figura 11.5 y en la tabla 11.2.

Iteración	Dist. antes (mm)	Dist. después (mm)	Segundos
1	23.3617	16.9620	20.0778
2	16.1331	15.4993	19.7434
3	15.2290	15.0944	19.4636
4	14.9740	14.9382	19.4807

Tabla 11.2: Resultados ajuste mediante ICP.

Vemos que el procedimiento funciona correctamente y además que el tiempo empleado se reduce considerablemente respecto a los tiempos que habríamos necesitado considerando los conjuntos completos. Para hacernos una idea de ese tiempo basta tomar el ejemplo del pie de la sección 10.4. En ese caso, los conjuntos tenían unos 7 000 y 8 000 puntos cada uno y tardaba poco más de un minuto. Ahora unos de ellos tiene más de 35 500 (ya simplificada), por lo que teniendo en cuenta el orden cuadrático del método, incrementaría bastante el tiempo necesario para completar cada iteración. Destacar que al igual que el procedimiento original, es posible que el resultado obtenido no sea el deseado y tenga que aplicarse desde el principio.

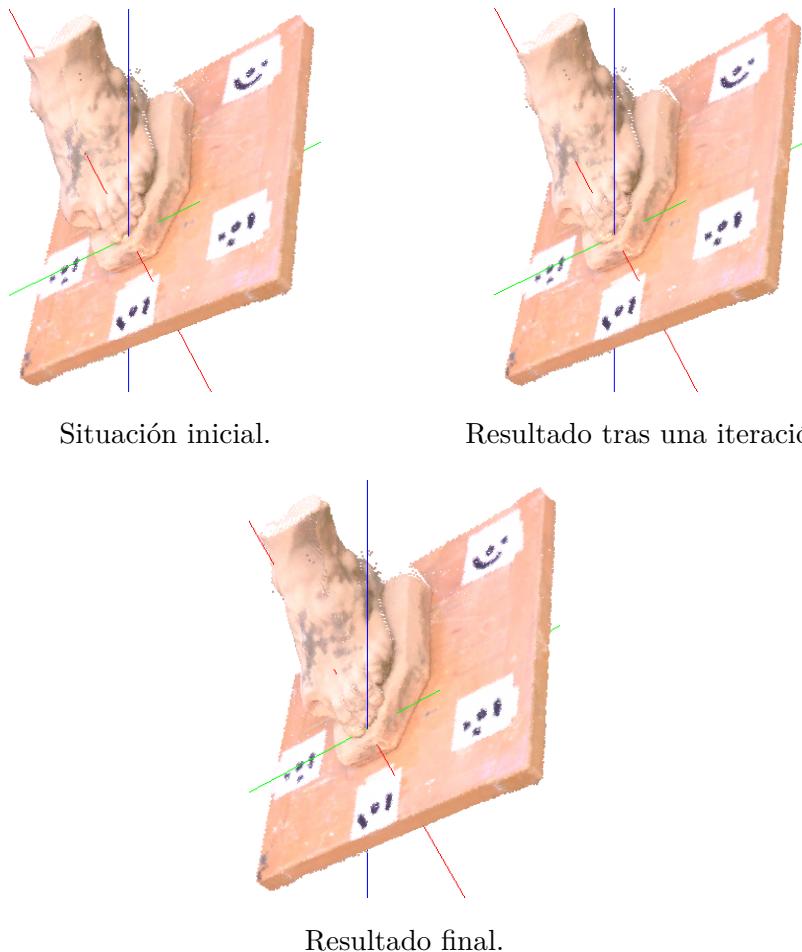


Figura 11.5: Algoritmo ICP usando un conjunto de puntos clave.

11.2.2. De puntos clave a puntos clave

Nuestro propósito ahora es intentar reducir aún más el conjunto de puntos con los que realizar el proceso ICP. Para ello, vamos a probar qué pasa si aplicamos el algoritmo solamente a los conjuntos de puntos clave calculados. Este paso hay que hacerlo con cuidado, ya que, incluso en mayor medida que tomando en una muestra la toma entera, nos arriesgamos a que muchos puntos no tengan una correspondencia. Así, para el cálculo del punto más cercano se ha tenido en cuenta:

1. Solo se comprueban puntos clave con un valor de descriptor parecido. Así, se ha fijado una diferencia máxima permitida y solo se comprueban los puntos que tengan un valor dentro de ese intervalo. Esto se debe a que al ser movimientos rígidos no hay ningún tipo de deformación por lo que los planos serán los “mismos” entre una toma y otra. Sin embargo, ponemos un intervalo debido a los errores del escáner y a las pérdidas de precisión debido al simplificado del modelo para la obtención de puntos clave.

2. Se ha acotado la distancia máxima permitida para un punto. Así, una vez calculada la distancia mínima de un punto al otro conjunto comprobamos que esa distancia no supere un umbral que hemos prefijado. Si lo supera significa que el punto más cercano está demasiado lejos y por ello no se corresponden realmente. De este modo, nos aseguramos que zonas que no se solapen se intenten ajustar y acabemos teniendo una solución errónea.

En la figura 11.6 mostramos un ejemplo del resultado obtenido imponiendo que los valores de los descriptores entre los puntos clavén difieran en 0.01 y que el cuadrado de la distancia (en metros) entre ellos no sea mayor a 0.1. Las dos tomas simplificadas contienen tanto la figura del pie como el tablero en el que se apoya y contienen unos 35 000 puntos cada una.

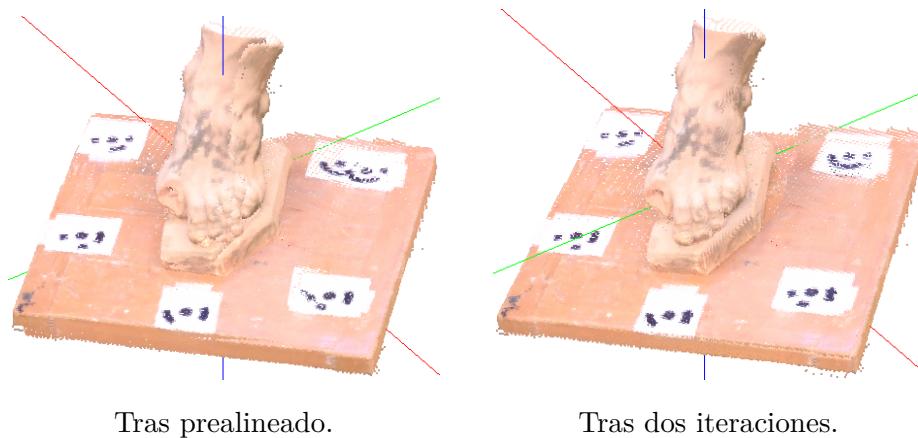


Figura 11.6: Proceso ICP solo con puntos clave.

Se han realizado dos iteraciones. En ambas ha tardado aproximadamente 0.7 segundos y la distancia media (en milímetros) ha variado de 27.9743 a 26.8752. Los puntos clave calculados en cada caso son en la primera toma un total de 1 142 y en la segunda 2 273. Observamos cómo tras las dos iteraciones se han conseguido ajustar mejor las dos tomas. Nuevamente, hacemos hincapié en que el algoritmo ha tardado menos de lo que habría necesitado en caso de haber tomado las nubes de puntos completas gracias a la reducción del número de elementos de ambos conjuntos.

Una segunda aproximación es contar el número de productos escalares que dan “cero” para saber el número de planos perpendiculares que tiene cada punto. De este modo, solo comparamos con puntos cuyo número de ceros es el mismo. Se ha tomado perpendicularidad cuando el producto escalar sea menor que 0.01 debido a errores en la toma de muestras y simplificado de la malla. Notar que seguimos teniendo en cuenta la cota de la distancia máxima. Esto puede hacer que en un paso del método la distancia pueda ser mayor que el anterior al existir un mayor o menor número de puntos que se han tomado como referencia pero no debería ser ningún problema *a priori*. En el ejemplo que hemos usado ese número no ha variado entre iteraciones. En la tabla 11.3 se recogen los resultados para cada iteración.

Iteración	Dist. antes (mm)	Dist. después (mm)	Segundos
1	35.2816	34.8378	2.33132
2	34.6116	34.4906	2.32062
3	34.4138	34.3591	2.30985
4	34.3277	34.3057	2.32283

Tabla 11.3: Resultados ajuste mediante ICP del modelo del con puntos clave agrupándolos por el número de ceros de sus productos escalares.

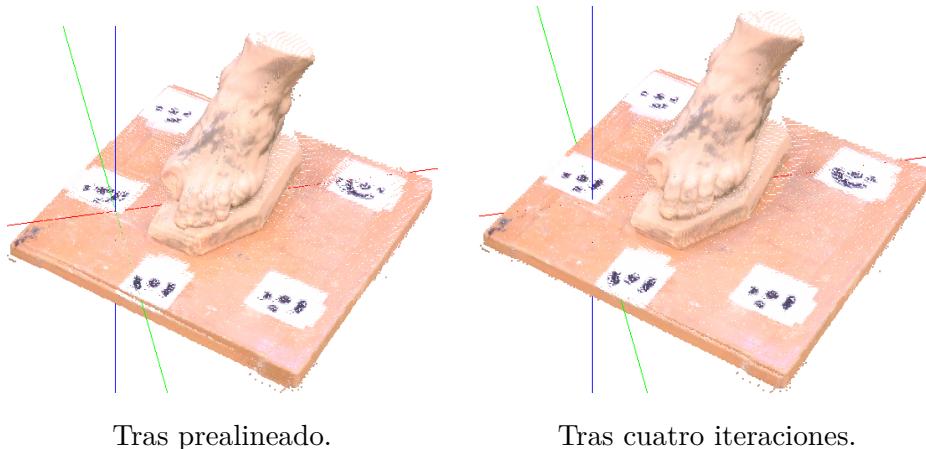


Figura 11.7: Proceso ICP solo con puntos clave agrupándolos por el número de ceros de sus productos escalares.

Con este procedimiento, en la figura 11.7 vemos que nuevamente se ha obtenido un proceso satisfactorio en la zona de los dedos del pie aunque ha tardado más tiempo en cada iteración que en el caso anterior.

11.3. Torre de las gallinas

En secciones anteriores, hemos realizado diversas pruebas con un modelo de un pie sobre un tablero. En esta probaremos los métodos ya implementados con otro modelo que representa una estancia dentro de una torre. Más concretamente, se trata de una parte de la torre de las gallinas, situada en la Alhambra escaneada usando un escáner láser Faro Focus 130. En las figuras 11.8 y 11.9 se muestran las dos tomas que se quieren alinear desde dos puntos de vista diferentes.

Destacamos la gran cantidad de puntos que tiene cada una de las nubes: la primera de ellas consta de 10 557 654 puntos y la segunda de 10 801 863 puntos. Si aplicáramos el procedimiento ICP original necesitaríamos bastantes horas para realizar el proceso. Por ello, lo que vamos a hacer es calcular los puntos clave de las tomas y luego aplicamos el algoritmo de alineado a los puntos obtenidos.

En primer lugar, vamos a comprobar cómo se comparten las tomas para la

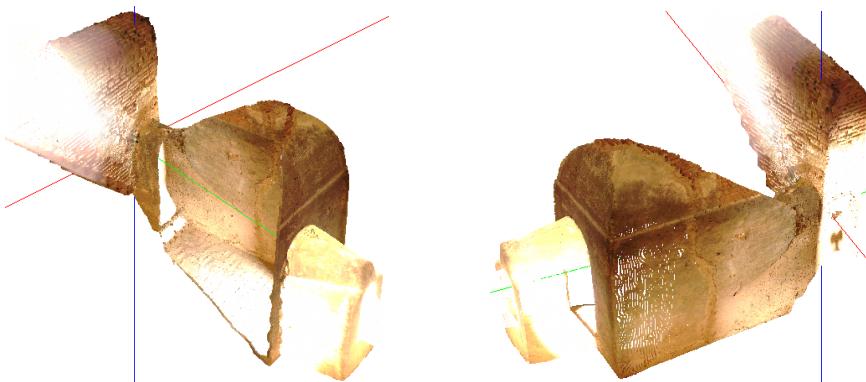


Figura 11.8: Nube de puntos de la torre de las gallinas usada como toma 1.

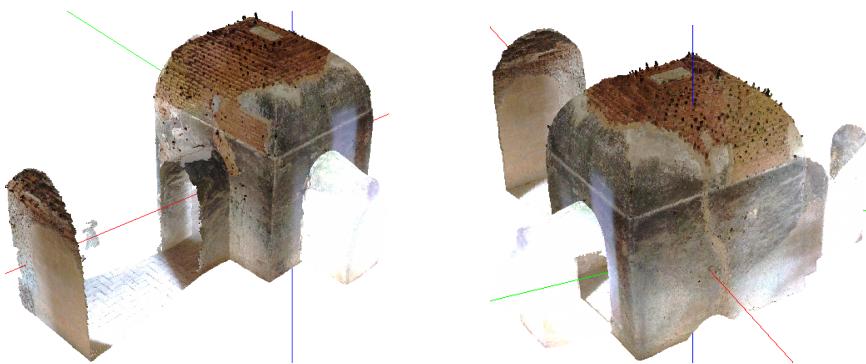


Figura 11.9: Nube de puntos de la torre de las gallinas usada como toma 2.

obtención de puntos clave sin simplificado y con simplificado tal y como hemos hecho en la sección 11.1. También tendremos en cuenta el número de puntos que se detectan en cada una de ellas. El nivel de simplificado que se ha escogido ha sido 5, ya que se ha considerado que puede ser un buen valor para tener la suficiente densidad en la malla y a su vez tener un número de puntos más tratable. Veamos cómo varían los datos de tiempo y tamaño mostrados en la tabla 11.4. Notamos que se han usado las mismas cotas para el cálculo de los puntos clave que anteriormente.

Las imágenes con los puntos clave obtenidos se muestran en 11.10 y 11.11. Vemos que a pesar de que en la toma simplificada la densidad de puntos es menor, el algoritmo nuevamente ha sido capaz de identificar mejor las aristas que en la original. Además, se observa que los tiempos empleados para el cálculo de las normales y obtención de puntos clave es considerablemente mejor. Finalmente, en la toma original se han obtenido un total de 862 225 puntos lo que sigue siendo un valor demasiado alto para obtener resultados con el algoritmo ICP en un tiempo razonable. Por otro lado, con la versión simplificada hemos obtenido 16 412. Mostramos también los puntos clave de la otra nube de puntos en la figura 11.12. Se observa que en la zona superior de la cúpula se han detectado una gran cantidad de los mismos al estar hecha de ladrillo visto. Ahora realizaremos las pruebas del algoritmo comparando los valores del descriptor para la obtención de los puntos clave,

	Original	Simp. 5
Calc. normales (seg.)	115.886	4.62641
Ptos. clave (seg.)	62.9924	2.41946
Ptos. clave (num.)	862 225	16 412

Tabla 11.4: Comparación resultados toma 1 original y simplificada (nivel de precisión 5). En la primera columna aparecen los valores que se han medido: tiempo de cálculo en segundos, tiempo en el cálculo de puntos clave y número de puntos clave detectados. En las columnas segunda y tercera aparecen los resultados obtenidos para la toma original y para la simplificada respectivamente.

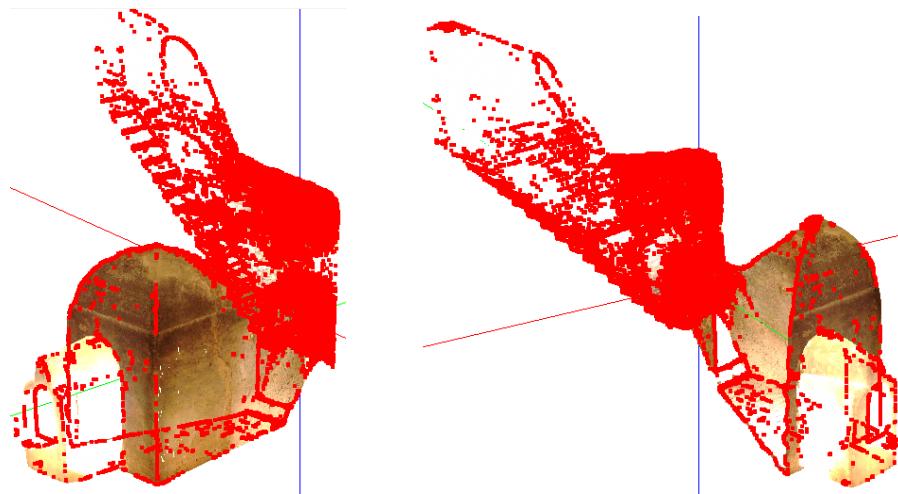


Figura 11.10: Puntos clave detectados en la toma 1 de la torre de las gallinas. Se han detectado 862 225 puntos.

ya que, en el ejemplo del pie anterior hemos observado que tardaba menos en la ejecución de cada iteración que viendo el número de ceros de los productos escalares.

En la figura 11.13 aparecen las tomas tras el prealineado y en la 11.14 mostramos los pasos del proceso aplicando el algoritmo entre puntos clave. Vemos cómo el método ha funcionado correctamente a la hora de hacer corresponder el hueco en la pared que se encuentra a la izquierda y cuadrar las paredes de la habitación. Recordamos que el alineado obtenido no es el más refinado, ya que en caso de quererlo deberíamos usar todo el modelo. Finalmente, destacamos que en el último paso, la distancia ha aumentado tras el procedimiento lo que puede parecer contradictorio. Sin embargo, en ocasiones se ha visto en la práctica que el método cuando está cerca de la solución óptima empieza a oscilar. Además, en las pruebas que estamos realizando no tenemos en cuenta un criterio de parada en función de la variación de la distancia media obtenida. Por último, también vemos que el número de puntos clave a los que se le aplica el método no es el mismo en cada caso. Por ello, entre una iteración y otra no es raro que aumente la distancia media. En lo relativo al

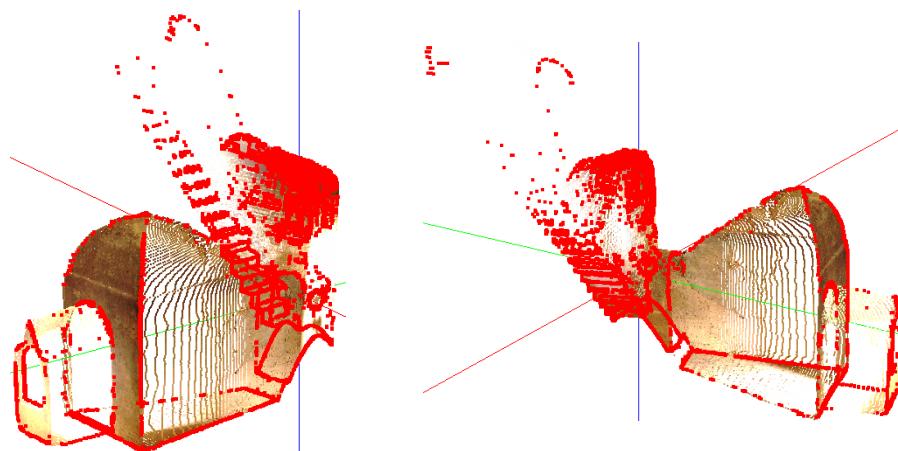


Figura 11.11: Puntos clave detectados en la toma 1 simplificada (nivel 5) de la torre de las gallinas. Se han detectado 16 412 puntos.

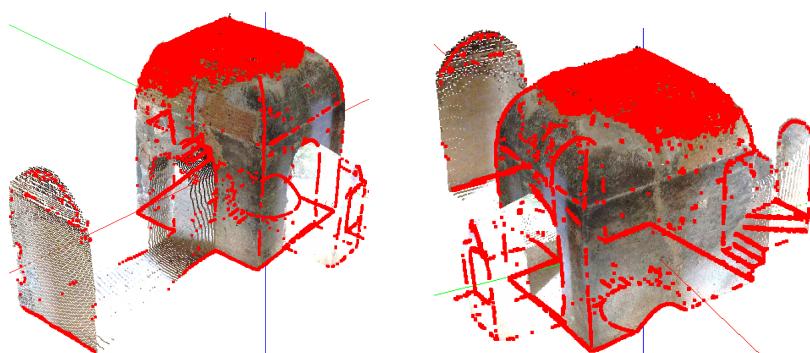


Figura 11.12: Puntos clave detectados toma 2 simplificada de la torre de las gallinas. Se han detectado un total de 33 187 puntos.

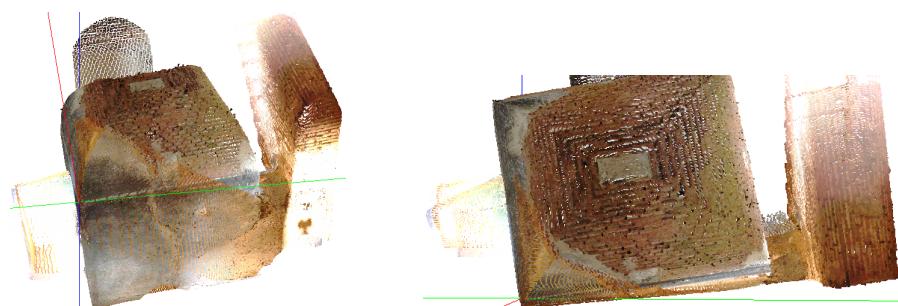


Figura 11.13: Vista general y detalle del prealineado de las tomas de la torre de las gallinas.

número de puntos clave, vemos que ha ido aumentando en cada paso por lo que las tomas están cada vez “más cerca”.

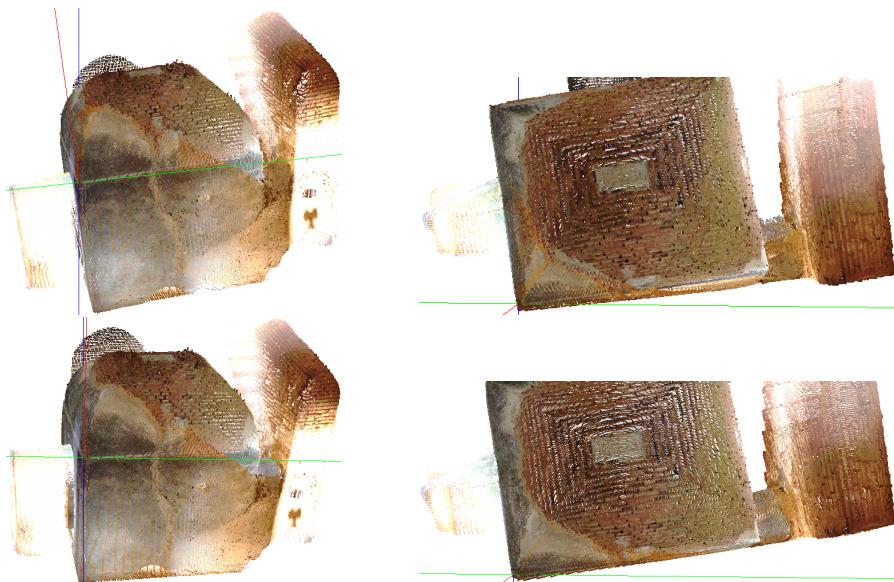


Figura 11.14: Dos vistas con el resultado de la primera iteración (fila superior) y de la cuarta iteración (fila inferior).

It.	Dist. antes (mm)	Dist. después (mm)	Segundos	Num. puntos
1	206.0735	203.3742	40.4705	1 997
2	185.7078	179.6215	41.7012	2 219
3	184.4570	182.8958	41.8810	2 385
4	182.9830	184.7062	35.8616	2 473

Tabla 11.5: Resultados ajuste mediante ICP de la torre de las gallinas usando puntos clave. En la primera columna aparece el número de cada iteración en la segunda y tercera las distancias medias antes y después de cada iteración respectivamente. En la cuarta se incluye el tiempo en segundos que ha tardado cada iteración y en última columna el número de puntos clave de la primera toma con “correspondencia” en la segunda.

11.4. Sensibilidad y uso en prealineado

En la sección anterior, hemos usado los puntos clave para alinear dos tomas. En el ejemplo, tras el prealineado dichas tomas estaban “bien” posicionadas, es decir, lo suficientemente cerca la una de la otra para asegurar en todo lo posible que el algoritmo funcione correctamente. Nos hacemos ahora las siguientes preguntas: ¿cómo de sensible es el método propuesto a la alineación inicial? o ¿se podría usar para el propio prealineado?. La primera no es una cuestión baladí, ya que, como se ha indicado anteriormente, es un paso crucial. Si a dos nubes de puntos, se le aplica el procedimiento general ICP no podemos asegurar que si repetimos el proceso con otro prealineado también funcione. Con esta sección, pretendemos determinar cómo de bien posicionadas deben de estar las tomas. Cuanto menos sensible sea,

menos nos debemos de preocupar del mismo e incluso si le afecta poco, podríamos usarlo sin necesidad de prealineado. Destacar que el objetivo de esta sección no es el análisis del tiempo que tarda en ejecutarse el algoritmo por lo que no se incluirán dichas tablas como en anteriores ejemplos.

De este modo, volvemos a las tomas que usamos anteriormente y se muestran en las figuras 11.8 y 11.9. Si aplicamos el procedimiento a las muestras tomadas directamente desde el escáner sin ningún prealineado obtenemos la secuencia de la figura 11.15. Destacar que se ha quitado el límite de longitud máxima para considerar el mínimo. De otro modo, los puntos clave más alejados no se tendrán en cuenta.

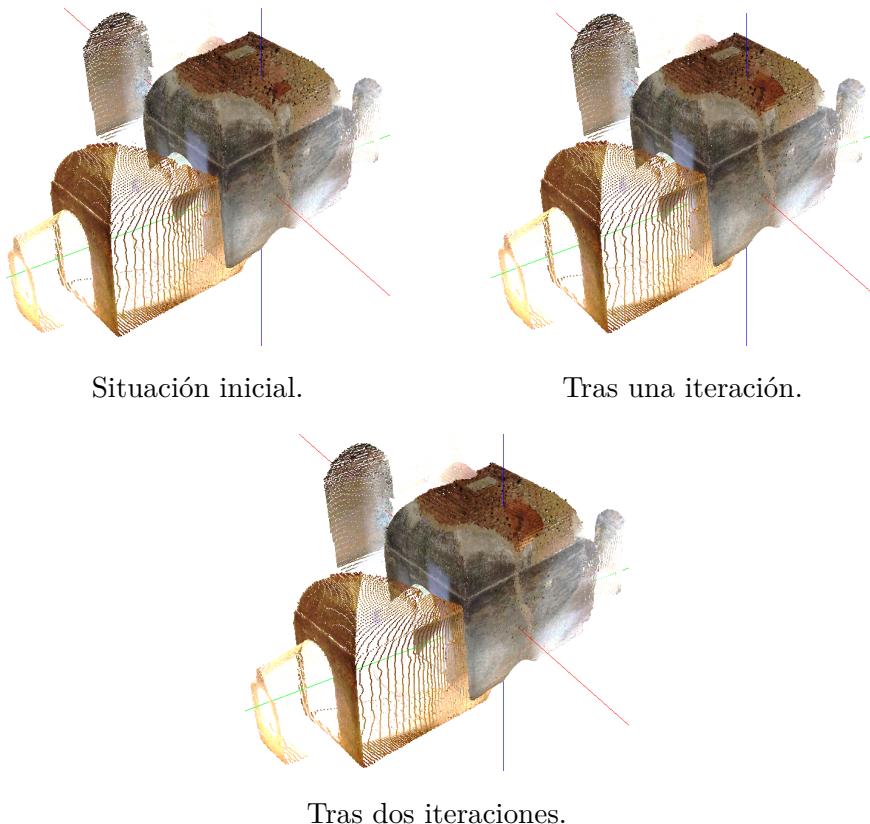


Figura 11.15: Ejemplo de sensibilidad. Aplicación del algoritmo ICP sin prealineado.

Vemos que el resultado no ha sido satisfactorio al no haber prácticamente cambios en el modelo. Esta situación puede deberse en cierta medida a que en la primera nube, se centran los puntos clave en la zona de la escalera y en la segunda, en la cúpula. Observamos que dichas zonas están juntas por lo que intenta ajustar esas zonas. Utilizamos ahora un prealineado no demasiado exacto para ver cómo se comporta el método. A partir de ahora, sí utilizaremos la cota para aceptar el mínimo. Esta se basará en un valor aproximado a la distancia entre los terceros vértices del

prealineado ya, que de ese modo obtenemos cierta referencia de las distancia, que manejamos. Claramente, este valor debe ser mayor, ya que las tomas están más separadas lo que también conllevará que se incremente el tiempo de cómputo. Los resultados aparecen en las figuras 11.16, 11.17 y 11.18.



Figura 11.16: Situación inicial. Usamos un prealineado no exacto.



Figura 11.17: Tras una iteración. Usamos un prealineado no exacto.



Figura 11.18: Tras dos iteraciones. Usamos un prealineado no exacto.

Vemos, que como en el anterior, el modelo apenas cambia tras aplicarle el algoritmo. Finalmente, realizamos el proceso con un prealineado un poco más fino que

el anterior. Este proceso aparece reflejado en las figuras 11.19, 11.20 y 11.21.

En este caso, hemos tardado 6 iteraciones en conseguir un resultado que podríamos considerar satisfactorio. No es tan preciso como en la sección anterior cuando el prealineado era mejor, pero teniendo en cuenta la situación inicial lo podemos considerar como aceptable. Por ejemplo, si nos fijamos en la evolución de las fotos de la izquierda, observamos cómo la esquina inferior se va acercando cada vez más entre ambas tomas hasta estar bastante cercana en ambos casos

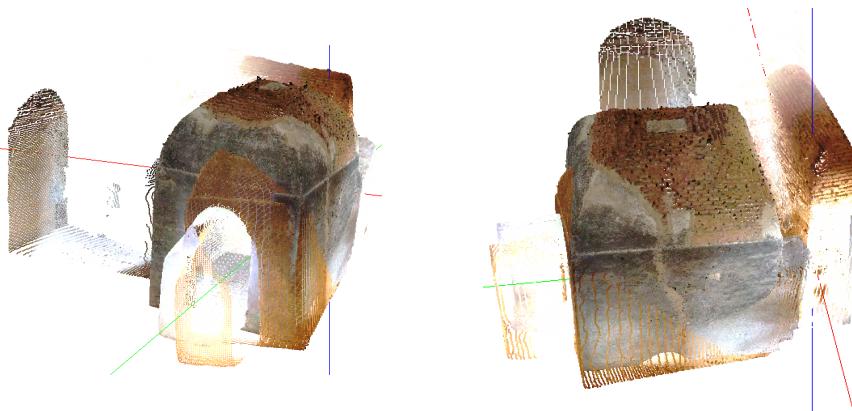


Figura 11.19: Situación inicial. Las tomas están mejor prealineadas.

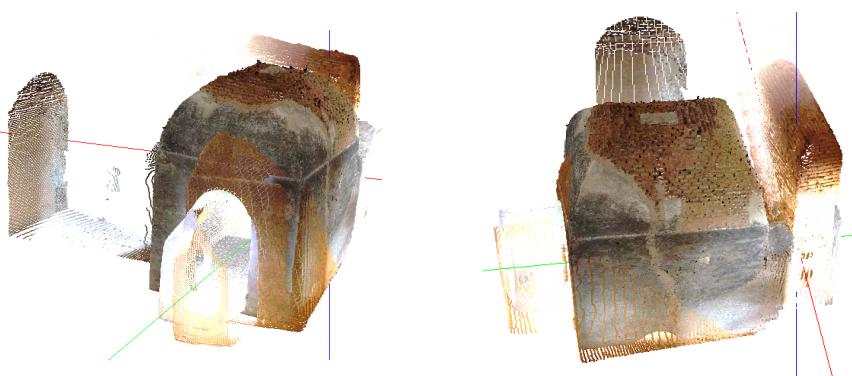


Figura 11.20: Tras una iteración. Las tomas están mejor prealineadas.

11.5. Conclusiones

Concluimos el estudio del algoritmo ICP con una sección de conclusiones acerca de todo lo que hemos visto hasta ahora. En primer lugar, gracias a este procedimiento hemos conseguido un mecanismo para alinear dos nubes correspondientes a un mismo objeto. Para poder comprenderlo hemos tenido que realizar un estudio previo acerca de los cuaternios. Sin embargo, este algoritmo tiene una convergencia local lo que significa que ambas tomas deben estar lo suficientemente “cerca” una

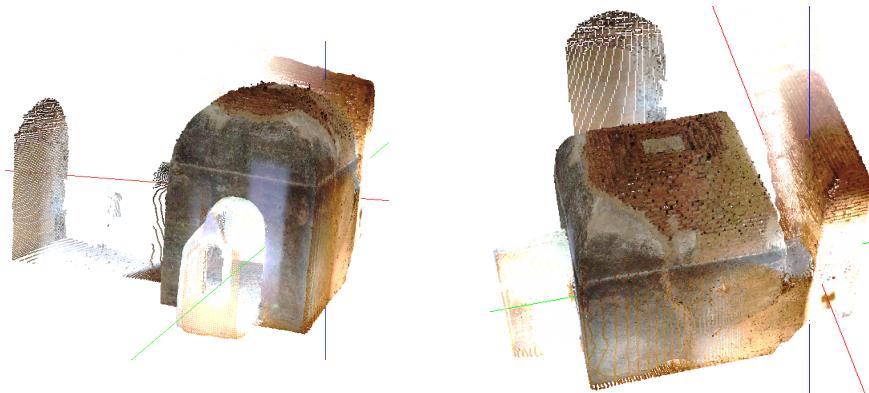


Figura 11.21: Tras seis iteraciones. Las tomas están mejor prealineadas.

de otra para obtener una solución adecuada. Esta situación hace necesario el uso de un prealineado inicial e incluso haciendo uso de él puede que no consigamos un resultado satisfactorio. Otro inconveniente que presenta es la complejidad cuadrática en tiempo. Esto implica que hasta en modelos simples, tarde bastante tiempo en completarse cada una de las iteraciones. Por ello, para intentar acelerar el proceso hemos introducido unos descriptores para cada uno de los puntos basándonos en la variación de la normal. Esto ha sido posible gracias a la forma que tiene el escáner usado para la realización de tomas de darnos los resultados. De esta forma hemos detectado zonas significativas en ambas tomas y realizado el mecanismo de alineación entre ellas. Hemos podido comprobar que los resultados han sido satisfactorios teniendo en cuenta tanto la simplificación del modelo para la obtención de las normales como el hecho de usar una parte de cada una de las nubes de puntos en el procedimiento.

RANSAC: algoritmo de consenso

En esta sección estudiaremos otro de los algoritmos más utilizados para la alineación de dos nubes de puntos. Se trata del algoritmo *Random Sample Consensus* (RANSAC). El método ICP que acabamos de ver podríamos decir que es específico para la solución de nuestro problema, es decir, no se utiliza más allá de la situación en la que lo hemos estudiado. Por su parte, el algoritmo RANSAC es más genérico y utilizado en otras tareas diferentes. Su finalidad es la de ajustar parámetros de un modelo matemático que viene determinado por un conjunto de observaciones. Nosotros lo utilizaremos para la detección de superficies planas en nuestros modelos que a su vez las utilizaremos para la obtención de sus puntos intersección. Esos puntos que podríamos considerarlos como clave, siguiendo la nomenclatura utilizada, serán la base para el alineado. Destacar que es considerado un algoritmo robusto incluso bajo la existencia de gran cantidad de valores atípicos y que sigue un paradigma de hipótesis-y-prueba.

12.1. Algoritmo

Este algoritmo fue propuesto por A. Fischler y Robert C. Bolles en [11]. Este será el artículo que seguiremos como referencia en lo que respecta a la explicación matemática de la veracidad del método. Como hemos dicho en la introducción, podemos considerar que es un algoritmo del tipo hipótesis-y-prueba. A diferencia de otros algoritmos, no tiene en cuenta todo el conjunto, sino que toma el menor número posible de puntos necesarios para ajustar el modelo y comprueba si efectivamente esos puntos determinan el modelo que queremos. En nuestro caso, el modelo que deseamos obtener es un plano.

El algoritmo consta de las siguientes etapas:

1. Tomar de manera aleatoria el mínimo número de puntos necesarios que define el modelo.
2. Con una tolerancia ε , contar el número de puntos que distan del modelo definido en el punto anterior.
3. Repetir 1 y 2 un total de N veces y quedarse con el que contenga una mayor cantidad de puntos.

De este modo, podemos considerar que el algoritmo necesita, por ahora, tres parámetros. El primero de ellos, el mínimo número de puntos para definir el modelo. Por ejemplo, si queremos obtener una recta debemos tomar 2 puntos mientras que si queremos obtener un plano debemos coger 3. También tenemos que decidir una tolerancia ε que depende, en cierto modo, de la nube de puntos que tenemos entre manos y del resultado final que queremos. Finalmente, hay que decidir un N que determina el número de veces que repetimos el proceso. Su cálculo se explicará en siguientes secciones pero destacamos ahora que debe de ser lo suficientemente grande para asegurar que este proceso aleatorio conseguirá nuestro objetivo.

Antes de continuar, veamos un ejemplo sencillo en dos dimensiones donde explicaremos cómo funciona este procedimiento. Supongamos que tenemos la nube de la figura 12.1.

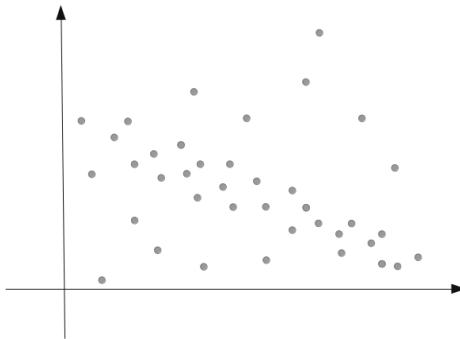


Figura 12.1: Nube de puntos.

Queremos aplicarle el algoritmo RANSAC para obtener una recta dentro de ella. Para ello, primero cogemos dos puntos aleatorios del modelo, calculamos la recta que pasa por ellos, y obtenemos los puntos que distan de la misma una distancia menor que ε . Supongamos que la situación es la que se muestra en la figura 12.2.

Si contamos el número de puntos que se ajustan al modelo, vemos que hay un total de 8. Repetimos el proceso y, en una iteración determinada, se obtiene el modelo de la figura 12.3.

En esta ocasión, un total de 27 puntos se ajustan al modelo. Si no se detectara en el resto de iteraciones un ajuste mejor, este sería el modelo que nos devolvería al algoritmo.

12.2. Número de iteraciones

Uno de los puntos clave del algoritmo es asegurarse de que el número de iteraciones N sea lo suficientemente grande para obtener puntos que ajusten correctamente

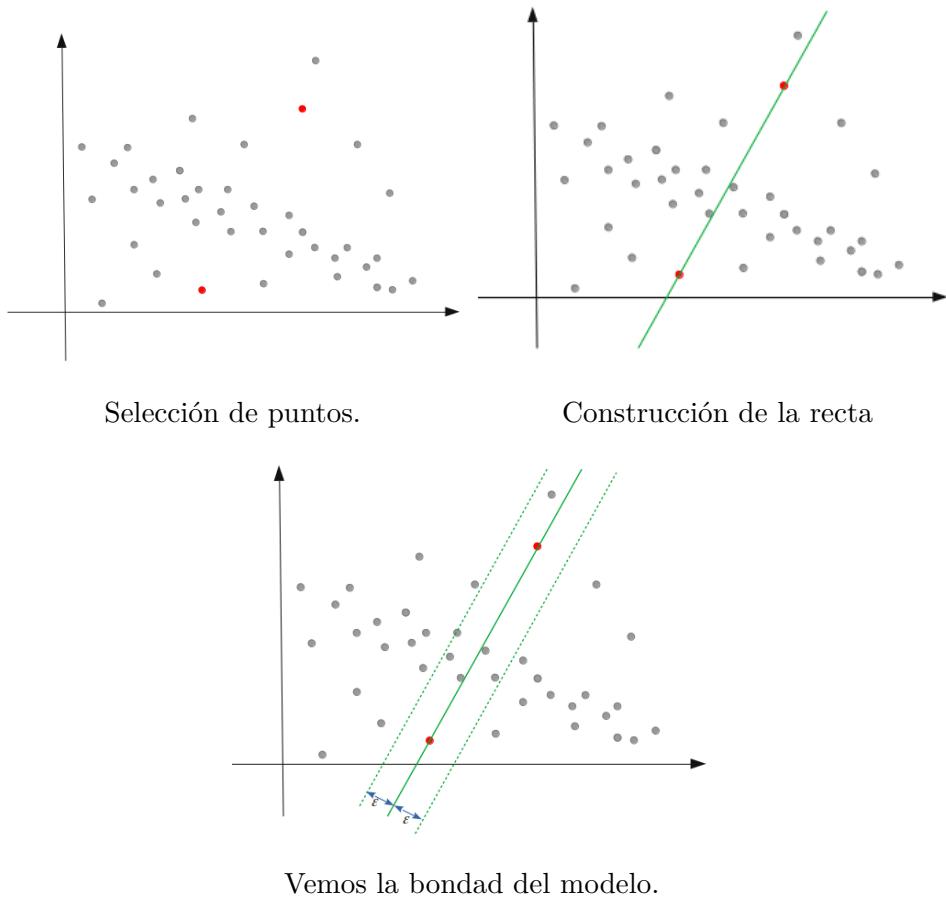


Figura 12.2: Primer paso del algoritmo RANSAC.

el modelo que queremos. De este modo, sea p la probabilidad que al menos uno de los conjuntos tomados no contiene valores atípicos y $1 - p$ de que todos los conjuntos contienen al menos un valor atípico. Notamos por u , a la probabilidad de que un valor sea correcto y, por lo tanto, $v = 1 - u$ es la probabilidad de ser un valor atípico. Sea m el número de puntos mínimo para definir el modelo: dos para una recta, tres para un plano, Así, la probabilidad de escoger todos los puntos correctos en una iteración es u^m y la de escoger alguno incorrecto es $1 - u^m$. Si hacemos un total de N iteraciones, la probabilidad de escoger siempre un valor atípico es $(1 - u^m)^N$. Pero esto no es más que

$$1 - p = (1 - u^m)^N.$$

Si despejamos el valor de N ,

$$N = \frac{\log(1 - p)}{\log(1 - u^m)} = \frac{\log(1 - p)}{\log(1 - (1 - v)^m)}. \quad (12.1)$$

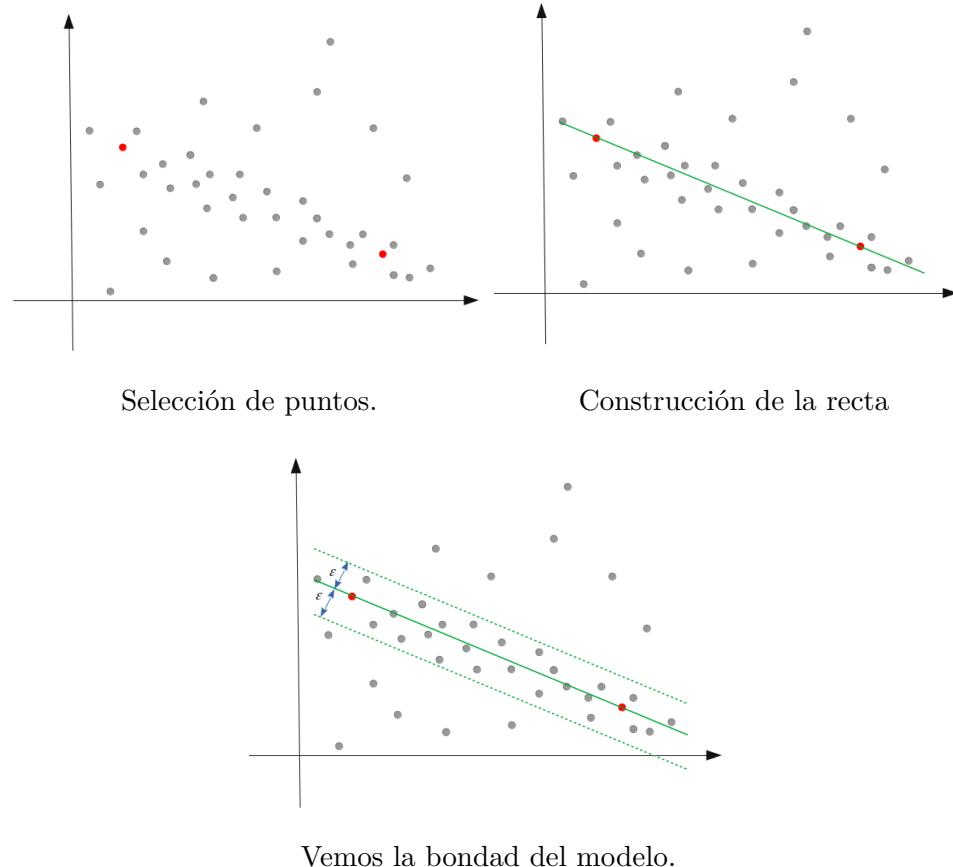


Figura 12.3: Paso del algoritmo RANSAC.

El valor obtenido no tiene que ser obligatoriamente un número natural por lo que se toma el primero entero que supere o iguale dicho valor. La probabilidad p es entrada del algoritmo y debe ser elevado. También u (o v) son parámetros que se eligen y que dependerán en cierto modo, del modelo con el que se está trabajando. La tabla 12.1 recoge diferentes valores de N en función de p , v y m .

12.3. Ejemplos sencillos

En esta sección, veremos cómo actúa al algoritmo en dos ejemplos concretos: el del pie y en el de la torre. En ambos casos usaremos los modelos simplificados tal y como hemos visto en la sección 11.1. Esto no debería de ser una restricción, ya que la densidad total del modelo no varía y los planos se mantienen. Para cada, utilizaremos diferente valores de los parámetros y veremos cómo de buenos son los resultados obtenidos. Al ser modelos en los que buscamos planos el valor de m es 3. Destacamos que para el cálculo de la distancia entre el modelo y cada punto se ha hecho mediante la función distancia con signo (en valor absoluto).

	m	v			
		0.05	0.2	0.5	0.8
$p = 0.99$	2	2	5	17	113
	3	3	7	35	574
$p = 0.9$	2	1	3	9	57
	3	2	4	18	287
$p = 0.8$	2	1	2	6	40
	3	1	3	13	201

Tabla 12.1: Número de iteraciones para diferentes valores de p (probabilidad), m (número de puntos para definir el modelo: 2 recta y 3 plano) y v probabilidad de ser un valor atípico.

12.3.1. Pie

Se ha tomado como nivel de tolerancia $\varepsilon = 0.05$, ya que ese valor depende sobre todo del modelo que estamos trabajando. El modelo consta de un total de 34 120 puntos. En la tabla 12.2 y en la figura 12.4 están los resultados que nos proporciona.

Prueba	p	v	Tiempo (seg.)	Iteraciones
(a)	0.5	0.4	0.052172	3
(b)	0.5	0.3	0.026094	2
(c)	0.7	0.3	0.055964	3
(d)	0.9	0.3	0.116656	6
(e)	0.9	0.5	0.356769	18
(f)	0.9	0.7	1.712500	85

Tabla 12.2: Resultados de aplicar el algoritmo RANSAC una vez a una toma del pie. La primera columna identifica la prueba (ver figura 12.4), la segunda y tercera los valores de p y v escogidos y las dos últimas el tiempo empleado en el cálculo del plano y las iteraciones que se han realizado.

Vemos que al ser un proceso aleatorio los resultados que se han obtenido son dispares pero en este caso, en un número “bajo” de iteraciones hemos visto que es capaz de detectar el plano que contiene el modelo.

12.3.2. Torre

Realizamos diferentes pruebas con nuestro modelo de la torre que consta de 431 254 puntos. En esta ocasión se ha optado por una tolerancia de $\varepsilon = 0.03$. Los resultados se recogen en la tabla 12.3 y en la figura 12.5.

Vemos que nuevamente se han podido obtener resultados buenos en relativamente pocas iteraciones.

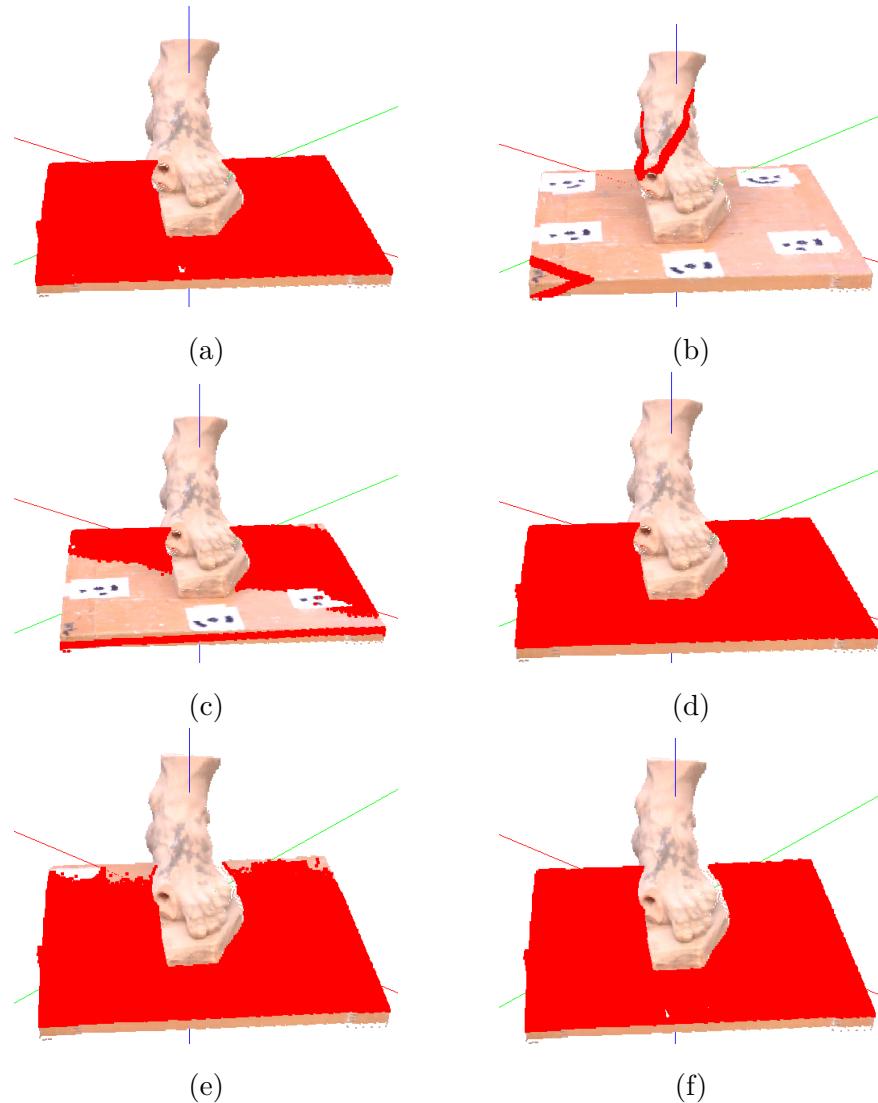


Figura 12.4: Resultados de aplicar RANSAC en el ejemplo del pie para diferentes parámetros (ver tabla 12.2).

12.4. Cálculo de los puntos de intersección

En esta sección, vamos a obtener un punto que sea intersección de tres planos obtenidos mediante el algoritmo RANSAC. Para ello, tomamos el ejemplo de la torre. En primer lugar, veamos cómo calculamos dicha intersección. Lo único que tenemos que hacer, dado un conjunto de puntos que hemos obtenido como plano, es tomar una normal del mismo. Esta normal puede ser, por ejemplo, la misma que hemos usado para el cálculo de las distancias mediante la función distancia con signo, la cual, se va almacenando para evitar su cálculo posterior. Notamos a la normal obtenida para el paso i como $N_i = (N_{i1}, N_{i2}, N_{i3}) \in \mathbb{R}^3$. De este modo,

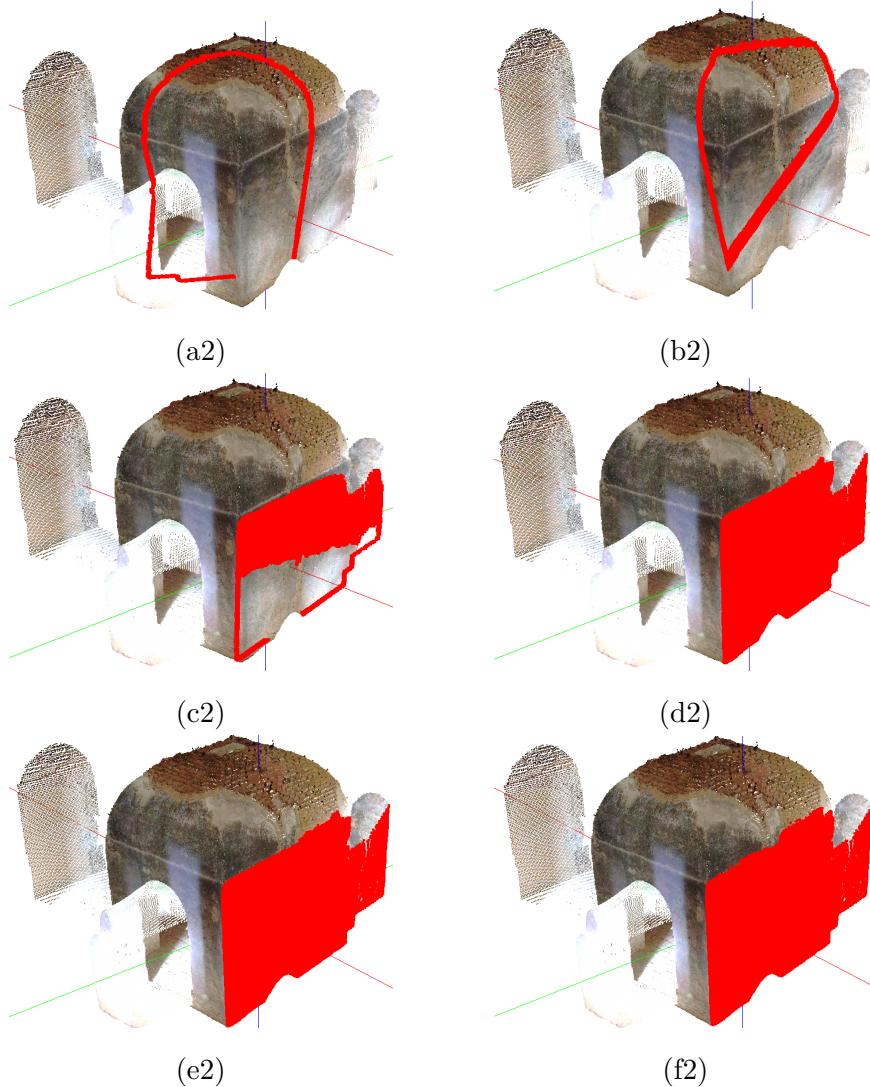


Figura 12.5: Resultados de aplicar RANSAC en el ejemplo de la torre para diferentes parámetros (ver tabla 12.3).

cada punto (x, y, z) del conjunto debe cumplir la ecuación:

$$N_{i1}x + N_{i2}y + N_{i3}z = d_i.$$

Finalmente, solo nos queda calcular d_i , que se puede hacer simplemente sustituyendo el valor de (x, y, z) por un punto conocido del conjunto obtenido. Hemos calculado la ecuación del plano de una manera sencilla. Ahora solo nos queda calcular la intersección de tres planos. Para ello, tenemos que resolver el sistema para los planos obtenidos en las iteraciones i, j, k :

Prueba	p	v	Tiempo (seg.)	Iteraciones
(a2)	0.5	0.4	0.484033	3
(b2)	0.7	0.4	0.961606	5
(c2)	0.7	0.6	4.271831	19
(d2)	0.8	0.6	5.749162	25
(e2)	0.8	0.7	14.00424	59
(f2)	0.85	0.8	55.98620	237

Tabla 12.3: Resultados de aplicar el algoritmo RANSAC una vez a una toma de la torre. La primera columna identifica la prueba (ver figura 12.5), la segunda y tercera los valores de p y v escogidos y las dos últimas el tiempo empleado en el cálculo del plano y las iteraciones que se han realizado.

$$\begin{aligned} \left. \begin{array}{l} N_{i1}x + N_{i2}y + N_{i3}z = d_i \\ N_{j1}x + N_{j2}y + N_{j3}z = d_j \\ N_{k1}x + N_{k2}y + N_{k3}z = d_k \end{array} \right\} \\ \Updownarrow \\ \begin{pmatrix} N_{i1} & N_{i2} & N_{i3} \\ N_{j1} & N_{j2} & N_{j3} \\ N_{k1} & N_{k2} & N_{k3} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} d_i \\ d_j \\ d_k \end{pmatrix} \\ \Updownarrow \\ A \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{d}. \end{aligned}$$

Si el sistema es compatible determinado, lo que significa que la intersección está formada solo por un punto, podemos calcular A^{-1} y obtenemos el punto de intersección como:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = A^{-1} \mathbf{d}.$$

Veamos este caso en el modelo real. Primero, obtenemos tres planos de la nube de puntos. Usamos el algoritmo con valor $p = 0.8$, $v = 0.8$ y $\varepsilon = 0.03$. Los planos que se han obtenido se muestran en la siguiente figura 12.6.

Los valores que se han obtenido han sido los siguientes:

- $N_1 = (0.987981, 0.154204, 0.0107157)$, $d_1 = -0.50992$.
- $N_2 = (-0.0192558, 0.0458887, 0.998761)$, $d_2 = -1.13957$.
- $N_3 = (0.147295, -0.987795, 0.0506549)$, $d_3 = -1.34888$.

Claramente, el sistema que forman es compatible determinado. Si lo resolvemos tal y como se ha explicado anteriormente obtenemos el punto cuyas coordenadas son $(-0.690391, 1.20058, -1.20946)$, que se corresponde con el marcado en la figura 12.7.

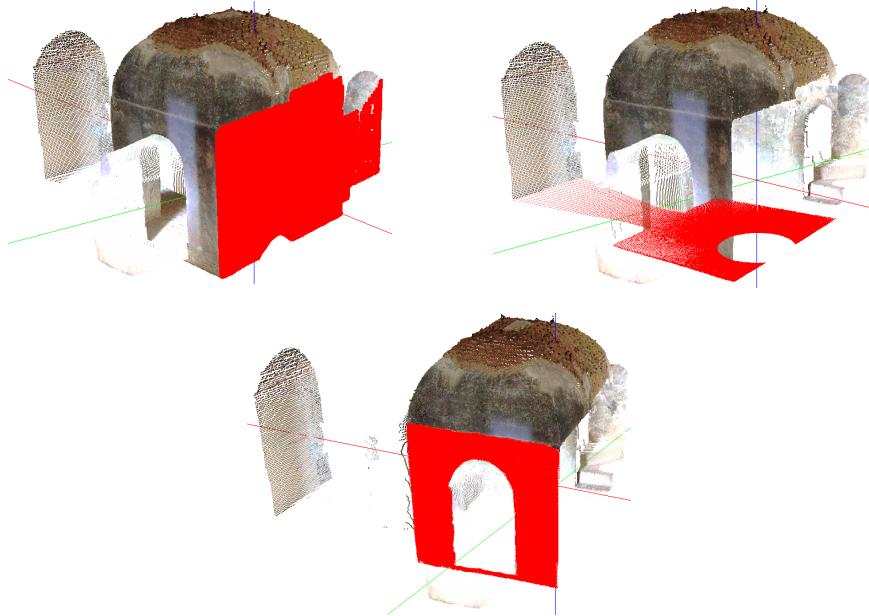


Figura 12.6: Planos detectados iterativamente en el ejemplo de la torre de las gallinas de la Alhambra. Los parámetros son $p = 0.8$, $v = 0.8$ y $\varepsilon = 0.03$.

12.5. Ejemplo despacho

El objetivo de esta sección es probar el comportamiento del algoritmo en un ejemplo más complejo y del que podemos obtener un mayor número de planos. En concreto, se trata de dos tomas de un despacho. Destacamos que nuevamente se ha realizado el proceso de simplificado de las mismas con un nivel de 4 y tras este proceso se han obtenido aproximadamente 650 000 puntos en cada toma. En la figura 12.8 se muestran ambas nubes de puntos.

En primer lugar, veamos los planos que se obtienen al ejecutar el algoritmo RANSAC en la primera de ellas. Los planos obtenidos se muestran en la figura

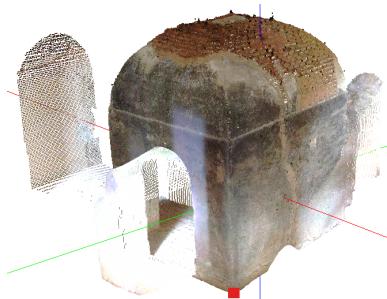


Figura 12.7: Punto de intersección.

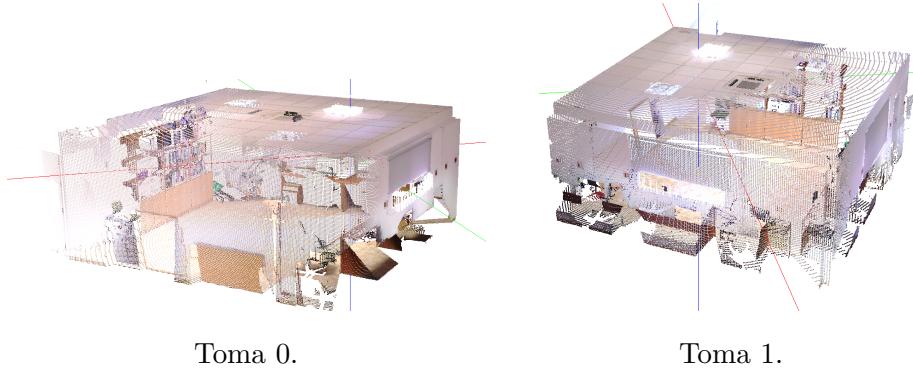


Figura 12.8: Tomas del laboratorio.

12.9. Posteriormente, en la tabla 12.4 aparece el tiempo empleado y el número de puntos detectados en cada plano. Finalmente, en la figura 12.10 están los puntos de intersección obtenidos.

Iteración	Tiempo (seg.)	Num. puntos
1	273.455	239 099
2	169.570	98 342
3	113.872	27 342
4	100.320	56 325
5	72.0483	46 067
6	64.7628	17 130

Tabla 12.4: Resultados de aplicar RANSAC en la toma 0 del laboratorio. En la primera columna aparece el número de la iteración, en la segunda el tiempo empleado en segundos y en la tercera el número de puntos que contiene cada plano detectado.

Repitamos el proceso con la otra toma. Los resultados se encuentran en las figuras 12.11 y 12.12 y en la tabla 12.5.

Una vez que hemos obtenido los puntos clave, lo que tenemos que hacer es agruparlos, es decir, ver la correspondencia entre ellos y calcular la transformación que lleva un conjunto en otro. Así, la idea es calcular tres correspondencias y aplicar un prealineado al igual que hicimos en la fase previa del algoritmo ICP. Nuevamente, utilizaremos los descriptores para obtener características significativas de los puntos y poder asignar unos con otros. Para ello, tomamos como referencia el artículo [16]. En él, propone varios parámetros posibles para formar parte del descriptor aunque el mejor de todos y el que usaremos es el vector formado por los ángulos de intersección de los planos en dicho punto. La explicación de usar este descriptor es que estos ángulos solo dependen de la geometría del modelo y no de otros factores como pueden ser el punto de vista. Una vez los hemos calculado, se obtiene una matriz con las “distancias” entre los ángulos de los puntos de ambas tomas. Ahora,



Figura 12.9: De izquierda a derecha y de arriba a abajo los planos que se han ido detectando en la toma 0.

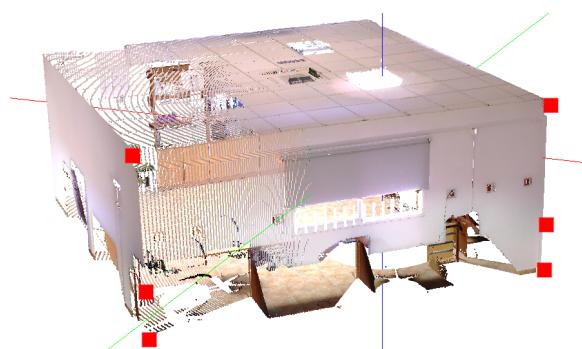


Figura 12.10: Intersecciones entre los planos en la toma 0.

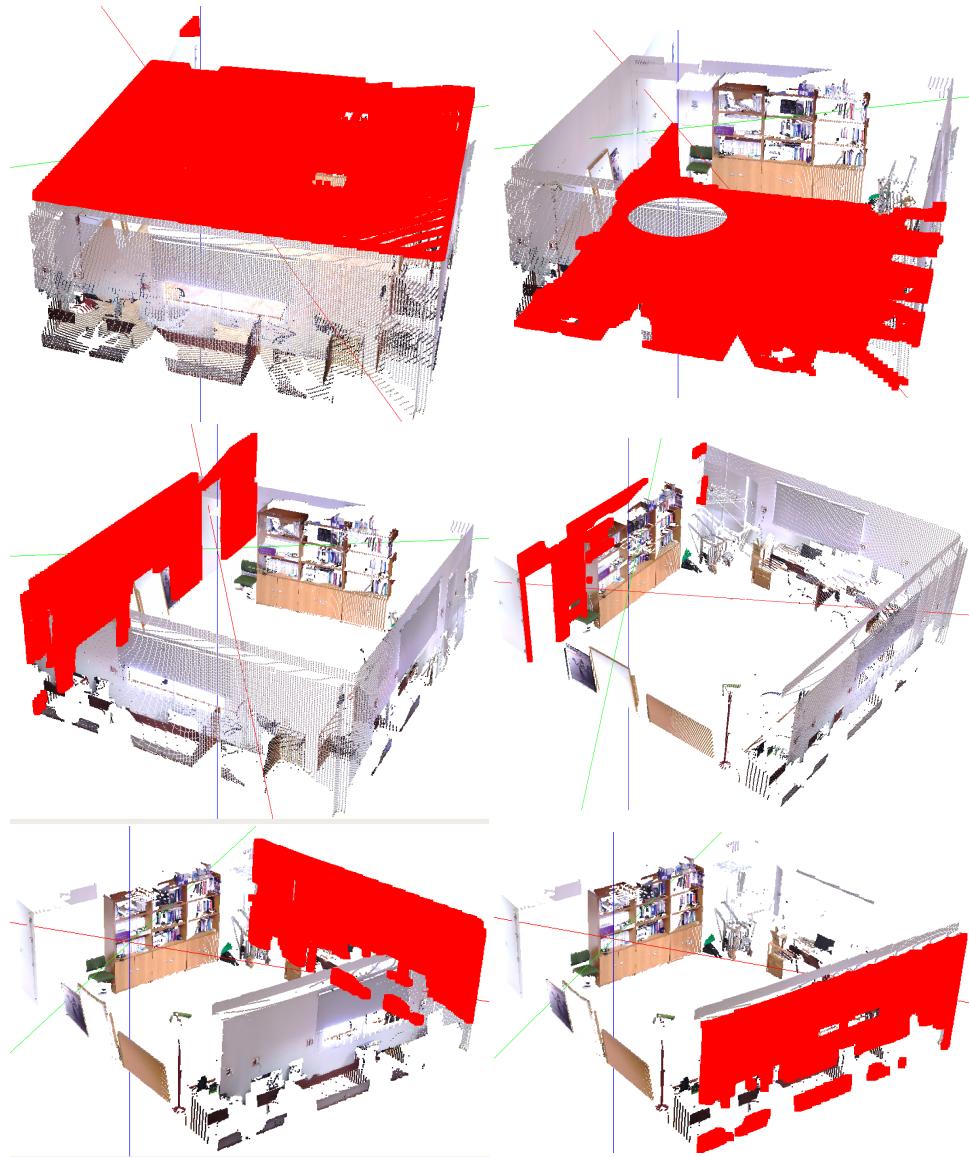


Figura 12.11: De izquierda a derecha y de arriba a abajo los planos que se han ido detectando en la toma 1.

aplicamos un algoritmo *greedy* o voraz para obtener las tres mejores correspondencias: vamos tomando en cada paso la menor de las distancias siempre y cuando no se repitan puntos ya obtenidos. La razón de que no se puedan repetir los puntos es bastante simple, ya que, de otro modo no se podría realizar el prealineado. Por ello, el resultado obtenido se muestra en la figura 12.13.

Tal y como se aprecia en la figura, los resultados no han sido los esperados. ¿A qué se puede deber este resultado? En primer lugar, a la propia geometría del mode-

Iteración	Tiempo (seg.)	Num. puntos
1	342.291	269 194
2	203.528	103 483
3	151.065	83 988
4	90.8403	31 849
5	87.0826	15 167
6	75.1206	12 961

Tabla 12.5: Resultados de aplicar RANSAC en la toma 1 del laboratorio. En la primera columna aparece el número de la iteración, en la segunda el tiempo empleado en segundos y en la tercera el número de puntos que contiene cada plano detectado.

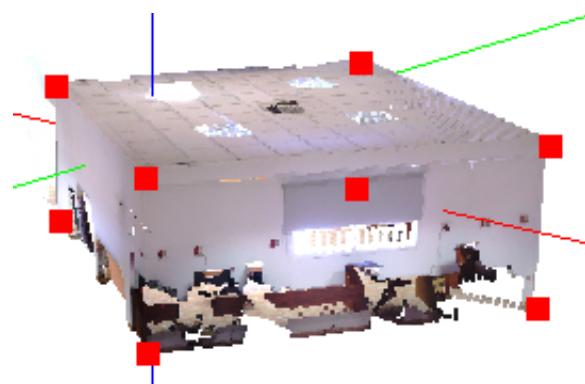


Figura 12.12: Intersecciones entre los planos en la toma 0.

lo. La habitación tiene sus paredes formando un ángulo recto entre ellas lo que hace que todos los descriptores sean bastante parecidos. Sin embargo, como ya hemos dicho: no existe un algoritmo que funcione con todos los modelos. En segundo lugar, a los propios errores de medición a los que se suman los del cálculo de los planos. Esto hace que las pequeñas variaciones que pueda haber se disipen. También tenemos que tener en cuenta las propias apreciaciones realizadas en el artículo tomado como referencia. Indica que la única manera para obtener la solución óptima sería hacer un algoritmo exhaustivo, es decir, probar todas las posibilidades. Ello conllevaría un gasto de tiempo enorme para el cálculo de la alineación al tener que calcular en cada combinación posible la distancia mínima entre los modelos (que ya hemos visto que es muy costoso). Otras posibles soluciones serían, por ejemplo, a partir de esos puntos que el usuario los seleccione manualmente como en el prealineado de ICP. La ventaja frente a esa situación es, que al ser calculados los puntos automáticamente, se obtendría un mejor resultado que si el usuario lo hiciera de una manera visual. Otra opción sería, siempre y cuando se hubiesen obtenido los mismos puntos en ambos casos, calcular la distancia media solo entre los puntos de intersección.

También influye en gran medida el número de planos obtenido durante el proceso. Según el artículo de referencia, el porcentaje de acierto con 15 planos no llega ni al 10 % mientras que unos 30 ya ronda el 80 %. Esto hace pensar que el algoritmo



Figura 12.13: Resultado de la alineación.

será más indicado con modelos más “complejos” que el usado en nuestro caso.

12.6. Ejemplo específico

En la sección anterior acabamos de ver que el algoritmo propuesto no funcionaba correctamente en nuestro modelo. Como adelantamos, esto no significa que no se pueda utilizar en otras ocasiones en las que podamos disponer de un modelo adecuado. Incluso en [16], dice que en el ejemplo que propone no se realizan todas las correspondencias adecuadamente. Por ello, se ha creado un ejemplo *ad hoc* para demostrar que realmente funciona. En nuestro ejemplo ficticio tenemos los planos

$$\left\{ \begin{array}{l} x = 2 \\ y = 2 \\ z = 2 \\ -0.707106x + 0.707106z = -2 \end{array} \right.$$

que conforman una “toma”. La otra está formada por la rotación de $\frac{\pi}{2}$ alrededor del eje X, es decir,

$$\left\{ \begin{array}{l} x = 2 \\ z = 2 \\ -y = 2 \\ -0.707106x - 0.707106y = -2 \end{array} \right.$$

En la figura 12.14 vemos cómo quedan ambos conjuntos.

Los puntos de intersección existentes entre los planos del primer conjunto son: (2,2,2), (4.82843, 2, 2) y (2, 2, -0.82843). En el segundo caso, son estos puntos tras

aplicar la rotación indicada anteriormente. Los puntos calculados se pueden observar en la figura 12.15

Si calculamos los ángulos que forman los planos en cada uno de los puntos obtenemos que en un caso es $(\frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2})$, en otro $(\frac{\pi}{4}, \frac{\pi}{2}, \frac{\pi}{2})$ y en el último $(\frac{\pi}{2}, \frac{\pi}{2}, \frac{3\pi}{4})$. En la otra toma estos valores no varían en los correspondientes puntos. Como vemos, ahora los valores son bastante diferentes entre sí y por ello el algoritmo los asigna correctamente y observamos el resultado en la figura 12.16.

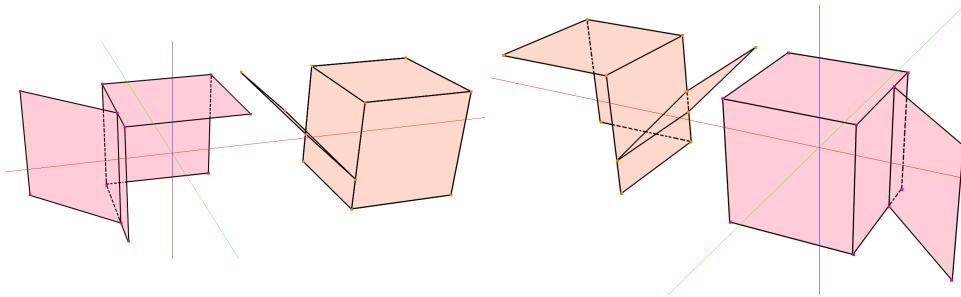


Figura 12.14: Planos de ejemplo. Notamos que en una toma se ha modificado la matriz de modelado¹ para poder visualizarlos por separado.

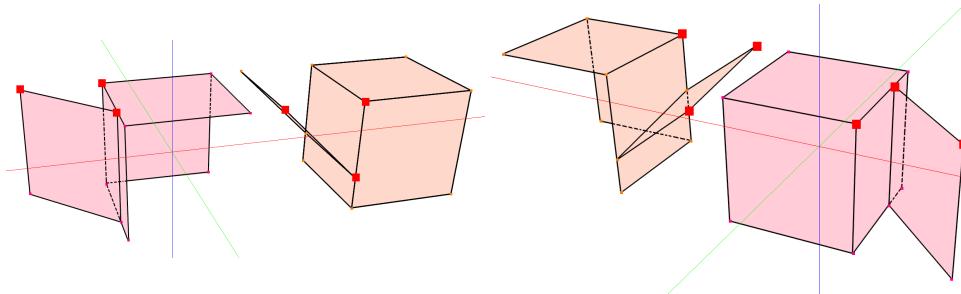


Figura 12.15: Puntos de intersección calculados en el ejemplo.

Concluimos que los resultados desfavorables obtenidos en la sección anterior se deben en gran medida a la simetría del modelo disponible y que en otros modelos sí es capaz de funcionar correctamente.

¹Esta matriz posiciona los puntos en su lugar en coordenadas de mundo. Se explica con algo más en detalle en el próximo capítulo.

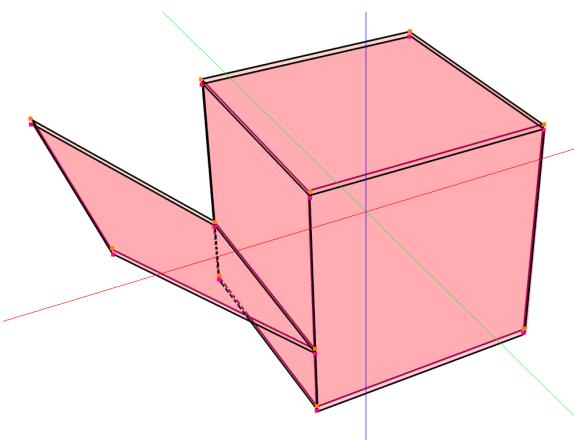


Figura 12.16: Resultado de la alineación. Se ha aplicado un leve cambio en una matriz de modelado para poder apreciar la superposición de ambos conjuntos.

Banco de pruebas

En esta última sección nos dedicaremos a explicar algunos aspectos relevantes acerca del banco de pruebas que se ha desarrollado. Como se ha indicado anteriormente, la finalidad de este trabajo fin de grado no era la del desarrollo de una aplicación que se pudiera usar para el proceso digitalizado 3D sino del estudio de las ventajas e inconvenientes de cada uno de los métodos que intervienen. Por ello, esta aplicación ha sido una herramienta más usada durante el proceso. Sin embargo, se ha considerado que merece tener cierta atención ya que se han implementado algunas características interesantes, especialmente las relacionadas con las nubes de puntos, así como aspectos importantes de la Informática Gráfica. Por eso, no se va a hacer un estudio detallado acerca de la aplicación pero sí se van a explicar aspectos generales de la misma.

Para el desarrollo del programa ha sido de gran ayuda la referencia [14]. En él se explica cómo desarrollar aplicaciones haciendo uso de la biblioteca *OpenGL* (*Open Graphic Library*) para el ámbito gráfico y *C++* como lenguaje de programación. Sin embargo, la ventaja la encontramos en que utiliza la herramienta *Qt*. La biblioteca gráfica solo nos permite producir los gráficos. Por ello, necesitamos una interfaz de usuario que nos permita la interacción y esa es la razón de usar *Qt*. A modo aclaratorio, también es necesario el uso de *GLEW* (*OpenGL Extension Wrangler Library*) que nos permite una gestión más fácil de *OpenGL*. También es conveniente destacar que el sistema operativo en el que se ha llevado a cabo todo este trabajo ha sido *Ubuntu 18.04.4* sobre un ordenador con procesador *Intel Core i7-3537U* y tarjeta gráfica *GeForce GT 720M*.

Finalmente, notamos que la aplicación que se propone en [14] ha servido como base para nuestro estudio. Principalmente, para el manejo de los diferentes objetos y algunos mecanismos como la selección de un punto (*picking*). El código se ha ido completando y modificando según las necesidades que teníamos en cada etapa del desarrollo.

13.1. Estructura básica de la aplicación

Una vez se han explicado las herramientas con las que trabajamos, la aplicación base para la visualización consta de las siguientes clases:

- *main*: es el programa principal. Su única finalidad es mostrar la ventana de la aplicación.

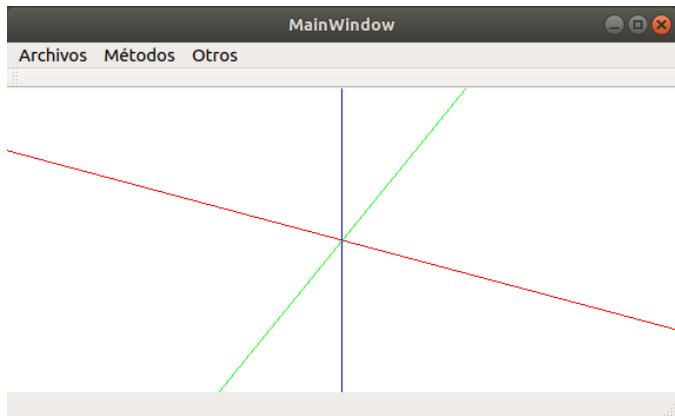


Figura 13.1: Interfaz de usuario del banco de pruebas programado.

- *mainwindow*: es la encargada de crear la interfaz de usuario. Deriva a su vez de la clase *QMainWindow*. En el archivo *mainwindow.ui* encontramos almacenada la información relativa a su configuración, principalmente, los menús con los que iremos activando cada una de las funciones necesarias.
- *_gl_widget*: clase derivada de *QOpenGLWidget* y que implementa las funciones para dibujar con *OpenGL*. Destacamos las funciones *initializeGL*, *resizeGL* y *paintGL* que sirven para iniciar *OpenGL*, actualizar el tamaño de la ventana y dibujar respectivamente. El código está preparado para visualizar los vértices, aristas o con relleno. Sin embargo, al trabajar nosotros con nubes de puntos, solo se visualizarán los mismos.

También se han programado los *shaders*. A modo aclaratorio, estos programas se ejecutan en la tarjeta gráfica del juego (no en la CPU) y que se aplican para transformar vértices, aplicar iluminación o crear algún tipo de efecto. Existen dos diferentes: *vertex shader* que se aplica para cada vértice y que transforma vértices, normales, calcula iluminación, etc. y *fragment shader* para cálculo de color de un fragmento (información para generar un píxel). En nuestro caso se han programado, por un lado, los dos *shaders* para la visualización normal de los objetos, y otro para la selección de puntos o *picking* que en la sección 13.2.

En cuanto al manejo de la cámara, se hace mediante teclado siendo posible tanto acercar o alejar la cámara y rotarla para observar mejor la imagen. La cámara se mueve alrededor de una esfera con centro el origen y no se ha añadido la posibilidad de cambiarlo. Por ello, es posible que haya objetos que podamos visualizar bien debido a su situación en el espacio. Esto se ha solucionado con la posibilidad de modificar la matriz de modelado de los mismos (mediante combinación de teclas). Esta matriz, como se ha indicado al final del capítulo anterior, se aplica a cada uno de los vértices del modelo y sirve para posicionarlos. Por ejemplo, cuando es la identidad, los puntos se mantienen en las mismas coordenadas que indican sus vértices. Sin embargo, si componemos transformaciones (en nuestro caso traslaciones pero también se puede hacer con cualquiera) a la misma conseguimos desplazar o escalar el objeto sin añadir carga extra. Esto se debe en el propio *vertex shader* se utiliza para calcular la posición de los puntos, junto las matrices de proyección y

vista (ver sección 13.3).

También relacionada con evitar demasiada carga de trabajo a la aplicación, nos encontramos con la manera de enviar los vértices a la tarjeta gráfica. Nos encontramos que estamos trabajando con una gran cantidad de puntos por lo que necesitamos que este trabajo se haga de manera rápida. Para solventar este inconveniente se usa el *envío en diferido* de los datos mediante los llamados *Vertex Buffer Object* (VBO). Estos son vectores que solo pueden ser usados por las tarjetas gráficas. Esto implica que los datos que necesitamos, vértices y colores, se encuentren en la GPU (*Graphics Processing Unit*) en vez de en memoria principal, acelerando el proceso de lectura de los mismos. Estos se han agrupado en un *Vertex Array Object* (VAO), que básicamente son estructuras para almacenar VBOs y que los activa automáticamente.

Dentro del mismo VBO almacenamos la información relativa a diversos objetos. Por ello, necesitamos saber en cada momento qué posición ocupa cada uno de esos objetos para poder, por ejemplo, visualizar algunos de ellos. Es ahora cuando entra en juego la clase *_object_management*. Esta clase contiene todos los objetos de la escena y se encarga del manejo de las posiciones de inicio dentro del vector y del tamaño de cada uno de los mismos.

13.1.1. Clase *_basic_object3D*

La clase que representa un objeto genérico es *_basic_object3D*. Encontramos por ejemplo los vértices, colores, normales, puntos claves, etc. Se ha ido modificando según las necesidades que teníamos en cada momento. Destacamos que de esta clase deriva otra, *_malla_ptx* que es la encargada de guardar y leer objetos que vienen en un archivo con formato PTX.

Destacamos que los puntos se mantienen en dos vectores y en una matriz. Un primer vector, que podríamos considerar auxiliar, con solo los vértices y que es el que se utiliza para llenar el VBO. El segundo y la matriz contienen elementos de una estructura que hemos definido y que se denomina *Punto*. Contiene tanto las coordenadas del vértice así como información relevante del mismo: posición dentro de la matriz (en el caso del vector) o del vector (en el caso de la matriz) y si el punto está activo o no. El hecho de hacerlo se debe a que cada uno de los casos es beneficioso para unas operaciones u otras. La matriz es la que nos indica la relación de vecindad entre cada uno de los puntos. Se usa, por ejemplo, para acelerar el cálculo de las normales, ya que solo tenemos que acceder a las posiciones de la matriz que rodean al vértice actual y no es necesario buscarlas en el vector de vértices. Por otro lado, el vector es más intuitivo a la hora del manejo de vértices en general. En cuanto a si al punto se encuentra activo o no se refiere a si se ha borrado o no. Al realizar una toma, generalmente no se capta solamente el objeto deseado en sí, sino también parte de su entorno. Así, necesitamos indicar los puntos que no nos interesan (ver sección 13.3) y borrarlos. En vez de ir modificando el vector de vértices dejando solo los activos, lo que conllevaría una gran carga de trabajo al escribir constantemente en memoria, solo se marca si un vértice está activo o no. De este modo, una vez se tienen marcados los que se quieren visualizar, solo se

tiene que actualizar el vector que utiliza el VBO. En cuanto a los colores de cada vértice, tenemos también dos vectores como en el caso de los vértices: uno con todos los colores (tanto de vértices activos como no activos) y otro que se envía a la GPU.

Es interesante destacar las funciones para leer y guardar los archivos, sobre todo esta última. Cuando hemos modificado un objeto, ya sea porque hemos alterado su posición o hemos eliminado puntos, necesitamos mantener las relaciones de vecindad del conjunto que teníamos originalmente (para poder calcular las normales, por ejemplo). También, para el caso de que se hayan borrado puntos, el archivo que contiene el modelo modificado debería ocupar menos espacio. Por eso, para guardar, lo que hacemos es encontrar la mínima matriz que contiene a todos los vértices activos y solo guardar dichas filas y columnas. Es posible que esta submatriz contenga puntos que no estén activos. Como no es posible guardar dicha información en el formato PTX, que es con el tipo de datos que estamos trabajando, se ha optado por darle unas coordenadas a esos vértices muy alejadas. Así, al leer el modelo, vemos si la distancia de cada vértice al origen está dentro de una cota. Si no la cumple, marcamos el vértice como no activo y no se dibujaría, pero se seguiría manteniendo la coherencia espacial. Esta cota también la podemos utilizar para filtrar directamente puntos al cargar un archivo y ahorrarnos el eliminarlos manualmente (ver figura 13.2). Al cargar también tenemos en cuenta otro parámetro que nos aporta el formato PTX: el coseno del ángulo de incidencia en el plano. De este modo, solo nos quedamos con aquellos puntos que no estén muy oblicuos, ya que eso implica mayor error en la toma de la muestra. Tanto estos valores de filtrado como los de los algoritmos se han pasado al programa mediante lectura de ficheros. Apuntar que el simplificado de las mallas que hemos mencionado en capítulos anteriores, se hace igual que la operación de guardar, solo aumentamos en cada iteración el incremento del índice para recorrer la matriz.

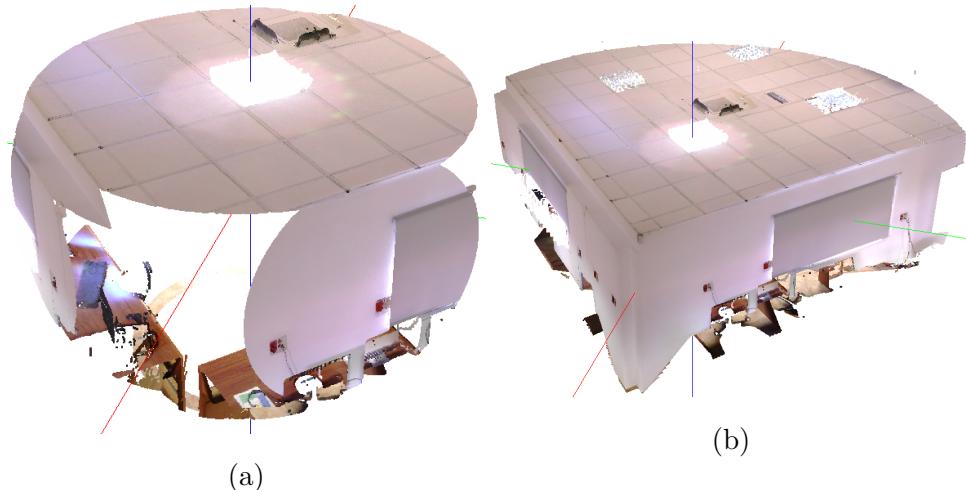


Figura 13.2: Ejemplo de filtrado según la distancia al escáner al cargar un modelo: (a) filtrado a 2 metros y (b) filtrado a 4 metros.

Esta clase proporciona los funciones con los métodos de alineado que hemos

estudiado. Destacamos que los procesos se han ido dividiendo en etapas. Esto se debe a que lo que nos interesaba era conocer el proceso en sí y no tanto tener un único mecanismo para hacerlo completo. Por eso, en ICP tenemos tanto la función de prealineado junto con otra para ejecutar solamente un paso del algoritmo (aunque también hay una función para ejecutar varias iteraciones con condición de parada la original y un número máximo de iteraciones si fuese necesario). Por su parte, el proceso de RANSAC se ha dividido a su vez en el cálculo de planos, poder cargarlos a partir de un archivo, cálculo de intersecciones entre los mismos, etc. Esta división también facilita las tareas de depuración, ya que tenemos cada tarea por separado así como poder reproducir los resultados una vez tenemos calculados correctamente algunos datos.

13.2. Picking

Esta sección junto con la siguiente podemos considerar que son más interesantes dentro del ámbito de la Informática Gráfica. El *picking* consiste en seleccionar un único elemento de la escena. En nuestra aplicación, queremos seleccionar los vértices para el proceso de prealineado. Este proceso debe ser tanto rápido como interactivo. Por ello, una de las soluciones más usadas para solucionar este problema es la llamada *selección por color*. Este método consiste en tener un identificador único para cada elemento y convertirlo de manera unívoca en un color. De este modo, podemos pasar de identificador a color y viceversa sin ningún tipo de problemas. Así, lo único que hay que hacer es dibujar el objeto con estos colores modificados, obtener el color de la posición que se ha seleccionado y recuperar el identificador. Como un color en RGB (*Red Blue Green*) se representa mediante tres bytes, tenemos un total de 16 777 216 de combinaciones. A esta cantidad habría que quitarle la combinación del blanco que se utiliza para indicar que no se ha seleccionado nada.

La relación entre identificador y color es de la siguiente manera:

- Identificadores de 0 a 255 van a la componente azul.
- Identificadores de 256 a 65 535 van a la componente verde.
- Identificadores de 65 535 a 16 777 215 van a la componente roja.

Esta conversión se hace fácilmente mediante máscaras y rotaciones de los bits. Ya tenemos la manera de asociar colores e identificadores, ahora, ¿cuál es el identificador para cada vértice que vamos a usar?. Usaremos la posición que ocupa cada vértice en el vector del VBO y que nos lo aporta directamente la directiva *gl_PrimitiveID* del *fragment shader*. Como se ha mencionado en 13.1 necesitamos por tanto otro programa *shader* diferente: en el *vertex* solo calculamos la posición en el mundo de cada vértice y en el *fragment* hacemos la conversión de identificador a color.

Volviendo al proceso, hemos comentado que tenemos que dibujar el objeto con los nuevos colores asignados. Esto no lo podemos hacer en la pantalla, ya que ahí queremos visualizar el objeto normal. Si se hiciese de ese modo se observaría un parpadeo. Se utiliza entonces un *framebuffer*, esto es, una zona de memoria donde se dibuja la imagen y se hacen otras operaciones como el cálculo de las profundidades (algoritmo *z-buffer*). El *framebuffer* principal es el único que permite mostrar por la

pantalla por lo que debemos crear uno propio, asignándole tanta zona de memoria para los colores como otra para el *z-buffer*. Finalmente, solo falta obtener el color del pixel seleccionado mediante la función *glReadPixels*. En la figura 13.3, encontramos un ejemplo de visualización normal y otras con los colores para el *picking*.

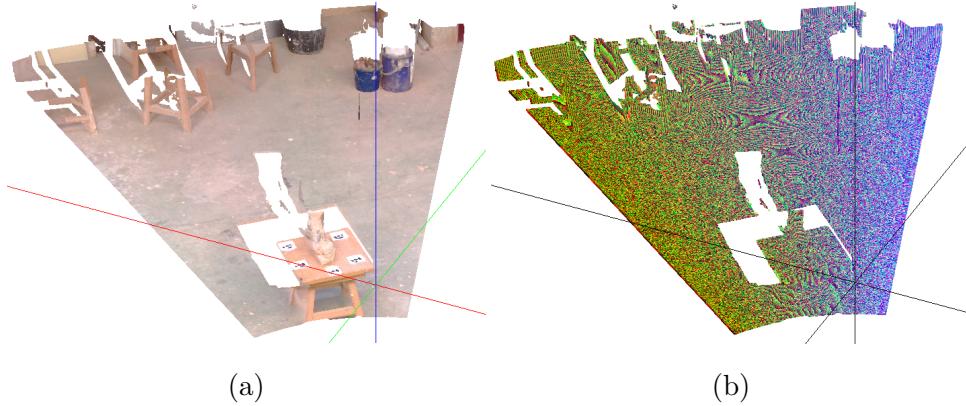


Figura 13.3: Visualización con los dos programas *shader*: (a) visualización normal y (b) visualización con la asignación de colores según identificador para el picking

13.3. Selección mediante rectángulo

Otra de las funciones a nombrar y que se han incluido en el banco de pruebas es la posibilidad de seleccionar puntos mediante un rectángulo. Esto es clave para eliminar las partes que no nos interesan de las tomas. Uno de los aspectos que involucra a este tema ya se ha explicado y es mantener la información de si un punto está activo o no para no elevar la carga de trabajo. El segundo aspecto a destacar lo tratamos aquí es la propia detección de puntos que están dentro del rectángulo y que se ha abordado de manera diferente al *picking*.

En primer lugar, necesitamos dibujar el propio rectángulo que indica la selección. Para ello se ha usado un objeto la clase *QRubberBand* que proporciona *Qt*. Además, ya tiene un método que nos proporciona si un determinado elemento está dentro del mismo o no. ¿Cuál es el aspecto interesante? Se trata del tipo de coordenadas que recibe como entrada dicha función y que veremos a continuación.

Para poder dibujar correctamente la figura en pantalla, hay que tener en cuenta no solo el lugar del objeto en el mundo virtual sino también la posición de la cámara, modificaciones en el propio objeto, etc. Esto ya se ha tratado al hablar de las traslaciones del objeto mediante modificaciones de la matriz de modelado, por ejemplo. Así, el flujo para obtener el lugar de la pantalla donde se proyecta cada vértice es el siguiente:

1. Los vértices vienen dados en coordenadas de objeto que es la posición según el marco de referencia del propio objeto.

2. Primero aplicamos la matriz de modelado para obtener las coordenadas del mundo que son comunes a toda la escena.
3. A continuación, mediante la matriz de vista, se obtienen las coordenadas de ojo y que se puede ver como la transformación que nos sitúa en el marco de referencia de la cámara.
4. Una vez hecho esto, necesitamos recortar la imagen indicando la región que queremos que sea visible así como el tipo de proyección que queremos, en nuestro caso principalmente con perspectiva. Esto se hace mediante la matriz de proyección.

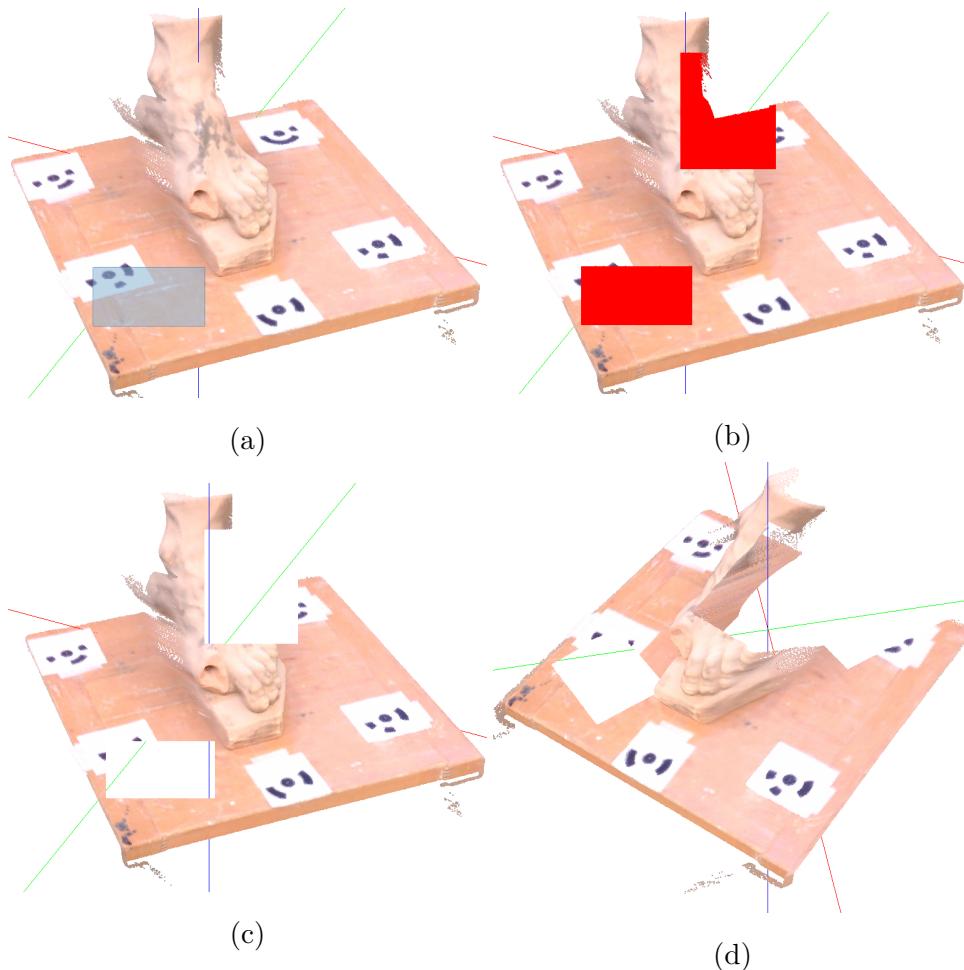


Figura 13.4: Borrado de puntos: (a) rectángulo de selección, (b) zona a borrar con posibilidad de mantener los seleccionados anteriores, (c) y (d) modelo tras borrar los puntos seleccionados.

5. A continuación se obtienen las coordenadas normalizadas de dispositivo, para que las coordenadas del punto se encuentren entre $[-1, 1]$. Esto se hace

mediante la división de un cuarto parámetro que obtenemos directamente del paso anterior.

6. Por último, se tienen las coordenadas de dispositivo mediante la transformación lineal de llevar el intervalo $[-1, 1]$ al tamaño y posición del *viewport* dentro de la ventana actual.

Una vez explicado el proceso, notamos que las matrices de modelado y de vista son relativamente fáciles de calcular: la primera porque es resultado de transformaciones que hemos ido aplicando y para la segunda hay que tener en cuenta que la cámara viene dada en nuestro caso en coordenadas esféricas. En cuanto a la matriz de proyección, la podemos obtener directamente pasando los parámetros del área que queremos visualizar. Así, la función *contains* de nuestro rectángulo recibe las coordenadas de dispositivo y que son necesarias calcularlas manualmente. En la figura 13.4 vemos un ejemplo de selección y borrado de un conjunto de puntos.

En definitiva, aunque el proyecto se haya centrado en otros aspectos, gracias al desarrollo de este banco de pruebas se han podido conocer nuevos entornos de desarrollo. También se han profundizado en otros aspectos relacionados con la informática gráfica que en mayor o menor medida ya se habían estudiado anteriormente pero se ha considerado que eran lo suficientemente relevantes para volver a tratarlos. Además, se han debido de solucionar problemas, como guardar modelos modificados, que aunque no son relevantes en las conclusiones obtenidos, son de suma importancia a la hora de poder trabajar.

13.4. Código, instalación y uso

Los archivos correspondientes a la aplicación se encuentran dentro de la carpeta *Banco de pruebas*. No hace falta realizar ningún tipo de instalación para poder ejecutar el banco de pruebas. En la documentación entregada se encuentra el archivo ejecutable denominado *Application* que puede ser ejecutado directamente desde una terminal en *Linux*. En la carpeta *src* se encuentra el código del programa y en *ES* están los distintos archivos auxiliares que hemos mencionado para pasar parámetros al programa. También, se adjunta un ejemplo de un modelo en formato PTX dentro de la carpeta *Ejemplo*. Para cargarlo, seleccionamos en el menú la opción *Archivos > Abrir*.

Para mover la cámara se usan las flechas del teclado y para acercar y alejar la misma se usan las teclas menos y más respectivamente. Destacar que las combinaciones teclado para modificar la matriz de modelado son: *CTRL+f* para mover al frente, *CTRL+b* para mover atrás, *CTRL+u* para mover arriba, *CTRL+d* mover abajo, *CTRL+l* mover a la izquierda y *CTRL+r* mover a la derecha. Para activar la selección de puntos aislados usamos la tecla *s* y para la selección por rectángulo *r* (si mantenemos pulsado *CTRL* podemos seleccionar varias zonas a la vez). Finalmente, para borrar los puntos seleccionados se usa la tecla *d*.

Conclusiones y vías futuras

Tanto en el ámbito matemático como informático, los objetivos que nos marcamos en la propuesta inicial se han alcanzado satisfactoriamente. En el primero de ellos, incluso se ha realizado una incursión no prevista en la teoría minimax de manos de los teoremas de la alternativa. Ello ha permitido obtener una visión completa de las técnicas y aplicaciones de los teoremas de la alternativa.

En el caso del digitalizado 3D, se ha realizado una sólida introducción con los procedimientos básicos existentes. El estudio hecho en este trabajo se podría completar posteriormente con varias vías debido a lo amplitud del tema. Sería posible plantear otra serie de mejoras a los algoritmos como por ejemplo, el uso de *voxels* para tener una mejor división espacial de la nube de puntos y acelerar el proceso del cálculo del punto más cercano. Otro posible camino a seguir sería estudiar algoritmos para la resolución de las otras dos etapas del proceso: fusión y triangulación. Finalmente, también se podría mejorar el banco de pruebas con la finalidad de que sea un *software* funcional para el público en general.

Bibliografía

- [1] J.M. Borwein, A.S. Lewis, *Convex analysis and nonlinear optimization: theory and examples*, Second Edition, CMS Books in Mathematics/Ourages Mathématiques de la SMC 3, Springer, New York, 2006.
- [2] R.J. Elliott, R.E. Kopp, *Mathematics of financial markets*, Second Edition, Springer Finance, New York, 2005.
- [3] G. Giorgi, A. Guerraggio, J. Thierfelder, *Mathematics of optimization: smooth and nonsmooth case*, Elsevier Science B.V., Amsterdam, 2004.
- [4] H. König, *Über das von Neumannsche minimax-theorem*, Archiv der Mathematik 19(1968), 482-487.
- [5] H. König, *Sublinear functionals and conical measures*, Archiv der Mathematik 77(2001), 56-64.
- [6] S. Mazur, W. Orlicz, *Sur les espaces métriques linéaires II*, Studia Mathematica 13(1953), 137-179.
- [7] E. Schechter, *Handbook of analysis and its foundations*, Academic Press, Inc., San Diego, CA, 1999.
- [8] S. Simons, *From Hahn-Banach to monotonicity*, 2nd edition, Lectures notes in Mathematics 1693, Springer, New York, 2008.
- [9] Jernej Barbic, *Quaternions and Rotations*, CSCI 520 Computer Animation and Simulation, University of Southern California.
- [10] Paul J.Besl y Neil D.McKay, *A method for registration of 3d-shapes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1992.
- [11] Martin A Fischler y Robert C Bolles, *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*, Communications of the ACM, 24(6):381–395, 1981.
- [12] Richard Hartley y Andrew Zisserman, *Multiple view geometry in computer vision*, Cambridge university press, 2003.
- [13] Yan-Bin Jia, *Quaternions and rotations*, Com S 477/577, 2013. Notes.
- [14] Domingo Martín Perandrés, *Informática Gráfica con OpenGL 4*, 2018.
- [15] Gary KL Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank C Langbein, Yonghuai Liu, David Marshall, Ralph R Martin, Xian-Fang Sun, Paul L Rosin, *Registration of 3D point clouds and meshes: A survey from rigid to nonrigid*, IEEE transactions on visualization and computer graphics, 2012.

- [16] PW Theiler, K Schindler, et al, *Automatic registration of terrestrial laser scanner point clouds using natural planar surfaces*, ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, 3:173–178, 2012.
- [17] *Qt Documentation*, <https://doc.qt.io/qt-5/reference-overview.html>
Último acceso: 26/06/2020.
- [18] *Eigen C++ library documentation*, <http://eigen.tuxfamily.org/dox/> Último acceso: 26/06/2020.