Automatic design of sequential interlocking assemblies

Pierre Gilibert

École des Ponts
ParisTech

THÈSE DE DOCTORAT
de l'École des Ponts ParisTech

Generative Design of Interlocking Sequential Assemblies

École doctorale N°531, Sciences Ingénierie et Environnement (SIE)

Structures et Matériaux

Thèse préparée au sein du Laboratoire Navier (UMR 8205)

Thèse soutenue le 31/01/2023, par
Pierre GILIBERT

Composition du jury:

Stefana, PARASCHO Professeure Assistante, EPFL	*Présidente*
Bernard, MAURIN Professeur des universités, Université de Montpellier	*Rapporteur*
Thomas, Bock Professeur, Technical University of Munich	*Rapporteur*
Melina, SKOURAS Directrice de recherche, INRIA Grenoble	*Examinatrice*
Olivier, BAVEREL Professeur, ENPC	*Directeur de thèse*
Romain, MESNIL Ingénieur de recherche, ENPC	*Co-directeur de thèse*

Generative Design of Interlocking Sequential Assemblies

Ecole des Ponts ParisTech

École des Ponts

ParisTech

Pierre Gilibert

January 2023

CONTENTS

ABSTRACT

In the context of sustainable construction in a circular economy, demountable wooden buildings represent an interesting research perspective. The question of reuse, i.e. the best possible reuse of the finished products resulting from the dismantling of buildings, requires that the building not be demolished, which makes it difficult to use nails or any other assembly devices of the same type. On the other hand, the elimination of modern metal assemblies common in wooden structures greatly reduces the carbon footprint of the building. Concerning their environmental impact, wooden assemblies offer much better prospects than their metal counterparts. However, few modern buildings are built without metal; for example, the Louis Vuitton Foundation is extremely consumer of metal products even though it clearly states the ambition to make it out of wood. Designing demountable structures and buildings is an ancient problem: some Austrian chalets are 500 years old, and the oldest inn in the world, the *Nishiyama Onsen Keiunkan* in Japan, was built in 705! their longevity is due to their demountability: when a part of the structure ages and begins to show signs of weakness, it is partially dismantled and replaced by a new one. Thus, element after element, these structures have withstood the test of time and are still functional today. What could be more sustainable than a 1300-year-old old building still in operation? The question also arises on the other hand: what to do with a 30- or 50-year-old building which, as the city and society change, has reached its expiration date and is no longer used? Here again, the question of dismantling and reusing its structural elements as much as possible for new projects (which potentially greatly reduces their environmental footprint) is part of a sustainable construction approach. Finally, with recent and future advances in robotic construction, the cost and time of manufacturing such assemblies will likely become increasingly competitive with their modern counterparts, both in economical and ecological terms. This PhD project ambitions to participate in the reflection on interlocking assemblies by exploring the feasible space of such assemblies and building generative tools to design them.

CHAPTER 1

INTRODUCTION

Social, economical and ecological contexts

The 19th and 20th centuries saw two concomitant phenomena in the construction sector. Labour legislation modifications and major improvements in health and safety on the construction site led to an increase in the cost of labour. Simultaneously, the rapid decrease of the prices of construction materials was made possible both by the explosive economic growth, seen especially in western countries, throughout the 20th century (which was enabled by cheap energy coming from the burning of fossil fuels) and by the omission the environmental and life destruction brought by the over-consumption of natural resources in the calculation of the price. In this context, it was, and still is, in the interest of construction companies to standardize their products: they simplify the design of structures and eased the assembling process as much as possible to shorten working hours and increase productivity, even if it implies consuming more resources than necessary.

Yet this evolution may soon reverse as the parenthesis of abundant energy seems to be closing. Helle Kristoffersen, the president of Strategy and Innovation at TotalEnergies (the major French company in oil and gas) stated that by 2025 the world will lack 10 million barrels of petrol per day [73], more than 10% of the current oil production. By comparison, the second oil crisis in 1979 saw a reduction of 7% of the oil production of the time. Yet contrary to the recovery that happened in the eighties, forecasts on the future of oil production show that the peak is likely to be reached in the 2020 decade, see FIGURE 1.1, mainly because of a depletion of the stock, the low rate of discovery of profitable deposit, and because shale oil in the US as yet to be proven cost-effective ([8]).

Coal production is also entering a period of high uncertainty. While acknowledging that the data to which they have access is limited, poorly detailed, and hard to check [8], academic researchers agree that the peak of coal production in China will likely

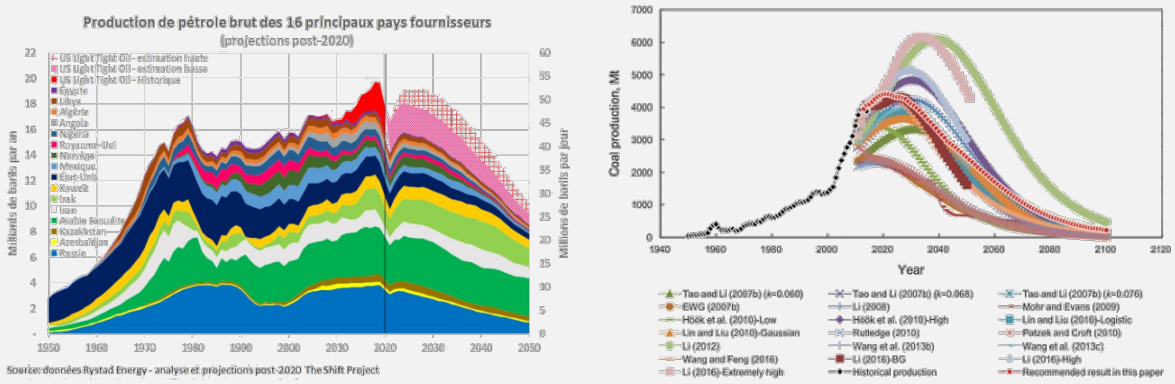


Figure 1.1| Left: forecast of the oil production of countries supplying France, from [7]. Right: forecast of the coal production in China, from [104].

be reached this decade, or at the latest in the 2030s, as shown in FIGURE 1.1. This could have major various impacts, as was seen recently when coal prices rose up in 2021 (before the war in Ukraine): gas prices surged in Europe, power outages happened in India [81], and the price of silicon, vital to make solar panels, has tripled [82].

Natural gas (mostly methane) production in the European Union (EU) has been steadily declining for the past two decades, while its consumption increased [65] resulting in varying degrees of dependence of the EU members on gas exporting countries. This provoked social unrest in the aftermath of the Ukraine war due to a soaring life price (which was already quite high with the disruptions of the supply chains caused by the coronavirus pandemic). Even if the war had not happened, the inexorable decline of domestic production due to stock depletion on the one hand and on the other "*an increase in demand due to the exit from coal and the intermittence of renewable energies will surely further aggravate the collapse of Europe's most energy-intensive industrial activities*", [8]. At the time of this writing, inflation in Europe led to a drastic reduction of production in some industries [19, 80].

Setting aside these economical and geopolitical considerations, climate imperatives put even more pressure on fossil fuels. As the International Energy Agency (IEA) urges to stop investments in new fossil fuel supply projects [76], a recent article [110] estimates that for the world to have a one-out-of-two chance of limiting global warming to 1.5°C , 58% of oil, 59% of fossil methane gas and 89% of coal should stay in the ground by 2050 (taking as reference the reserves found in 2018). While the description of the effects of climate change is not the focus of this manuscript the reader is referred to, *e.g.*, [60, 63] for a chilling description of the world we are

on track to living in.

What must be done

The main challenges facing all industries in the coming decades are two-fold: energetic sobriety on the one hand as fossil fuels are coming to an end, possibly faster than expected, and with them so is the era of abundant energy and cheap materials. On the other hand, there must be a massive decrease in the environmental footprints of the products manufactured to avoid the worst of suffering due to climate change, but also to preserve critical resources (sand, minerals, arable lands, water, etc.) for the generations yet to come — or even ourselves in a few years.
While some governments loudly advertise green growth made possible by an energy shift towards renewable sources it remains to be seen whether the Earth has enough mineral resources to carry out this task [102]. So far it seems, to me at least, that we are just shifting from dependence on fossil fuels to dependence on fossil minerals, with high uncertainties on the stocks available (those that are known are in the hands of even fewer countries than oil deposits, diminishing further the sovereignty one could hope to have), and whose extraction exerts a heavy toll on ecosystems. According to Dr Fatih Birol, the IEA executive director, "*today, the data shows a looming mismatch between the world's strengthened climate ambitions and the availability of critical minerals that are essential to realising those ambitions*".

Thus, like all industries, the construction sector must reduce the embodied energy of the built environment, where the "*embodied energy can be viewed as the quantity of energy required to process, and supply to the construction site, the material under consideration*" [49], but also reduce the various impacts on the geochemical and bio-spheres it currently has (toxicity on humans, marine and terrestrial ecosystems, mineral resources used, water scarcity, climate change, land use change, etc.). While one may rightly be dubious about high-tech solutions in general as they are more energy intensive than low-tech ones, and high-tech construction will certainly not solve all these problems, it may help in some cases, and the numerical exploration of various alternatives of a given design may also put in evidence structures necessitating relatively few resources and with lower impacts on the world[1]. DiXite project [25], which finances this work, was created to study the role played by digital technologies in the construction industry. One of the main issues the project tackles is how the future built environment can be made sustainable.

[1] That being said, the most ecological edifice is the non-built one. A balance must be found between planetary limits and human needs, desires and hubris.

What did the Ancients do?

To answer this question we do not need to start from a blank page: instead, we may leverage the 12 000 years of sustainable civilisations preceding us. Coarsely speaking, our ancestors dealt with sustainable construction in two manners:

- **Build-to-last**:
  Whether we look at the Egyptian or Maya civilisations, with their impressive pyramids, or the ancient Greeks and Romans with some of their edifices still in a remarkable state of conservation - The Pont du Gard in France, the Pantheon in Rome or the Parthenon in Athens - several civilisations massively oversized their buildings, for reasons ranging from religious zeal to beautiful aesthetics to demonstration of political power, with the side effect of making them last millennia. Even more day-to-day buildings can last centuries: it is not that uncommon in Europe to live in a house built 500 years ago. These long-lasting buildings dilute their potential negative environmental impacts over their lifetime.
- **Build-to-maintain**:
  Other civilisations took a complementary approach: they based their built environment on demountable wooden structures so that damaged or weak parts can easily be replaced without taking down the entire building. By regular maintenance, some Alpine chalets are still in use centuries after being built, and in Japan, the inns *Nishiyama Onsen Keiunkan* and *Hōshi Ryokan* were respectively founded in 705 and 718 and have seen continuous service ever since[2]. A low-tech building in use for 1300 years surely qualifies as sustainable. Japanese mastered the art of demountable buildings, with in particular the Ise shrine, see FIGURE 1.2: founded around 690, this temple is to be rebuilt every 20 years. The next rebuilding is scheduled for 2033. One can also think of Scandinavian stave churches, some built in the 12th century and maintained to this day, or Chinese structures erected a thousand years ago.

This work explores the alliance between modern numerical tools and the historically proven approach of demountable structures. What explains the durability of demountable structures? Aside from the ease of maintenance, many interrelated factors play a role in this: according to Glahn who focused on Chinese architecture in [37], the quality of the wood, the flexibility of the joints which dampens the oscillations of the structure, and the fact that the columns merely rests on stone feet instead of being anchored, which prevents excessive strain during an earthquake, as well as the weight of the roof which counterbalance wind loads during typhoons, all explain the surviving of these structures for centuries. Aside from the strict durability of these edifices, in our ever-changing modern cities, demountable designs

[2] Operated by the same families for 52 and 46 generations!

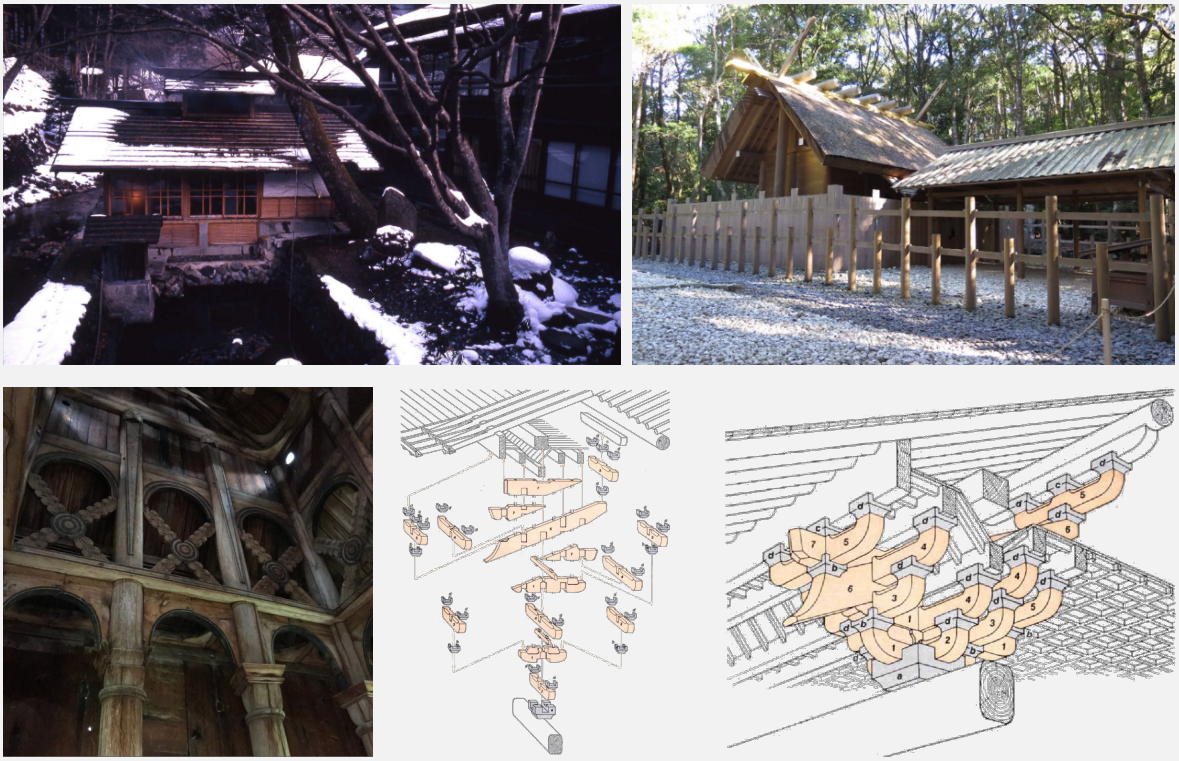


Figure 1.2| Top left: a view of the *Hōshi Ryokan*, Japan, source: Wikipedia. Right: a picture of the Ise shrine, Japan, source: the Internet. Bottom left: a view of the interior of the Torpo stave church in Hallingdal, Norway, source: the Internet. Bottom right: an exploded and assembled view of a bracket-arm set, typical of medieval Chinese architecture, from [37].

are of interest as they enable to reuse of structural members. Indeed, today, at the end of the lifetime of a building, assemblies are often destroyed which prevents the reuse of structural components and has a significant impact on the planet: according to the French agency for the environment and energy management, ADEME [23] the construction sector generated almost 70% of the waste in France in 2017. Conversely, new construction materials have a severe environmental impact, up to 7% of the emission of greenhouse gas in France in 2016 [67].

Yet, when we look at the wooden edifices built today, we see a profusion of irreversible joints making the structure permanent, whether because of metal fasteners such as nails, screws, bolts and nuts, or because of the use of glue. In the United States of America, 92% of the new residential buildings were made of wood in 2021

[100], mainly through a technique called platform framing which has extensive use of nails. It is thus no wonder that the technical know-how presiding the designs of these demountable joints virtually disappeared in the last two centuries in wood buildings despite their proven relevance. It turns out that historians precisely pinpoint the shift from the design of demountable structures to the irreversible ones that are common today, in the village of Chicago in the fall of 1832, with the invention of the so-called balloon frame.
The westwards expansion of the American people in the first half of the 19th century resulted in the rapid growth of the villages inhabited by pioneers. In Chicago, the population was multiplied by almost 12 in five years, from 30 inhabitants in 1829 to 350 in 1833 [91]. This sudden surge, or rather the associated demand for buildings, provoked a shortage of the large timbers that were traditionally used in log construction, which in any case could not have been worked with because there were no skilled carpenters in Chicago at the time. Simultaneously the industrialisation and mechanisation of nail factories drove their price down, from a historical 25 cents a pound to 5 cents a pound in 1833 [13]. Taking advantage of these cheap machine-cut nails, as well as of the abundance of small trees in the vicinity of the village, a man named George Snow [35] in the need of a warehouse [91] invented the balloon frame consisting of lumber of small dimensions permanently nailed to each other, see FIGURE 1.3, which differs greatly from the larger sections needed to carve traditional reversible assemblies. The term balloon frame was originally used to ridicule a structure that was deemed too thin and light (compared to traditional designs) to withstand loads. According to the New York engineer Francis W. Woodwards, the balloon frame could "*be put up for 40% less money than the mortice and tenon frame*" [114]. The further development of the nail and timber industries and the fabrication of standardised pieces, as well as the absence of skills required to build a balloon frame resulted in the adoption of this technique to build houses in the treeless regions of the West: "*the western prairies are dotted over with houses which have been shipped there all made, and the various pieces numbered, so that they could be put up complete, by any one*" [40]. The invention of the balloon frame "*converted building in wood from a complicated craft, practiced by skilled labor, into an industry*" [35], and led to the disappearance of master carpenters which were replaced by unskilled labourers. In the course of the 20th century, balloon framing progressively made way to platform framing, easier and safer to erect and more fireproof.

Since balloon frames are thinner and use less material than their traditional counterparts and given the availability of iron in the planet's crust to make structural steel with, the reader might rightly wonder why we should bother in finding

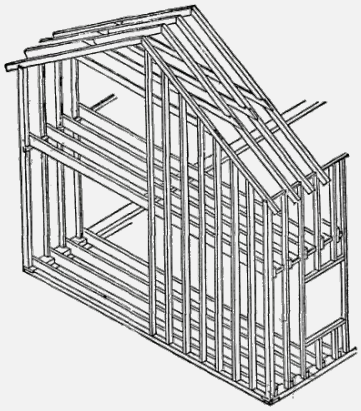


Figure 1.3| Isometric view of a balloon frame.
From [114].

novel and complex assemblies when we could keep using balloon-frame-like ones. As said above a decrease in the availability of energy is to be expected. According to [95] the energy needed to process a cube meter of steel is 42.9 MWh. By comparison, T. Gobin estimates the embodied energy of machined wood to be around 0.5 MWh/m^3 [39]. If relatively little steel is needed to make nails, its high embodied energy may make a steel-based assembly less attractive than a wood-wood joint, also called *integral joints*; thus, this embodied energy ratio of about 80 shows that careful calculations should be carried out when designing assemblies as to choose one with the least embodied energy. The environmental footprints of assemblies were recently investigated in depth in the doctoral thesis of K. Mam [58]. He states that "*the production of steel in assemblies can account for up to 61% of the total climate change impact of a timber structure, almost all (97%) of the carcinogenic toxicity impact and 32% of the fine particle formation impact*". He concludes that "*the idea appears that the optimisation of the structure would not necessarily involve reducing the volume of wood, but rather thinking about the manufacturing processes and the types of assembly of the elements*". It seems therefore to be a good idea to try and reduce the quantity of steel used in an assembly, which implies shifting back to tradition-inspired wood-wood assemblies, precisely the focus of this manuscript.
The above remarks should be qualified: indeed a recent study [62] finds that, by 2100, if only half of the new urban dwellers live in timber buildings, forest plantations area should expand by 160%. If we consider instead 90% of new urban dwellers in timber buildings, said area should triple! The authors specify that this estimate is conservative as it does not take into account adverse effects of climate change (e.g. mega-fires and tree dieback) and the renovation or replacement of already-existing edifices. The ADEME estimates that by 2050 about 85% of the consumption of construction materials for housing and tertiary buildings will be con-

crete constituents (sand, gravels, cement) [79]. If timber buildings certainly has a huge potential for carbon emission mitigation [62], it does not seem to become the main construction material of tomorrow and will not suffice to answer the industry demand. That is why in this dissertation we adopt a material-agnostic stance: we will not consider a specific material for our assemblies.

Intent of the dissertation

This dissertation investigates the automatic design of integral joints between parts of an assembly. We aim at numerically finding novel assemblies, with surprising geometrical features, that are relevant to the construction industry. The work presented in this dissertation is still in its infancy: we focus almost exclusively on the geometrical aspect of an assembly. While some work has been done to assess the mechanical relevance of the generated assemblies, the fabricability aspect has not been investigated at all. Thus, there is still a long way to go before having a streamlined workflow from the numerical finding of an assembly (studied here) to the digital fabrication of mechanically relevant designs.

Organisation

This manuscript is organised as follows: CHAPTER 2 defines what we mean by assemblies and presents some past assemblies revealing the skills of master carpenters. It is followed by a review of the work done in puzzle generation, assembly planning and robotic fabrication, fields that are related to ours. The short CHAPTER 3 is an in-depth presentation of the knowledge the scientific community has on assessing the interlocking of an assembly. While no new concepts are introduced there, it is necessary to read it to understand this manuscript, as our whole approach is to reverse-engineer key principles presented there. CHAPTER 4 is the bulk of this dissertation: we derive the mathematical equations that must be obeyed by a 2D assembly whose kinematics are prescribed, adapt an optimisation algorithm to generate such an assembly and study in depth the mechanical properties of its constitutive parts, as well as the influence played by fabrication imperfections on the kinematics of the assembly. Finally, in this chapter, we introduce what we deem to be our most significant contribution, namely the robust optimisation of assemblies with regard to imperfections. The last CHAPTER 5 simply extends the concepts presented in the previous chapter to 3D assemblies. The main interest of this chapter, in our opinion, lies in the thorough and detailed introduction of unit dual quaternions, little-known mathematical objects that perfectly describe a 3D assembly kinematics.

CHAPTER 2

SEQUENTIAL ASSEMBLIES

2.1 INTRODUCTION

This chapter aims to introduce and define what assemblies are, as well as to give examples of research works and built structures.

2.1.1 OVERVIEW

This chapter naturally begins by defining the concept of assembly. We briefly present several kinds of assemblies necessitating an intermediary body, typically a nail, to hold together before looking at the specific kind of assembly that will be the focus of this manuscript, namely reversible interlocking assemblies held together through the geometrical features of its constitutive parts.

2.1.2 CONTRIBUTION OF THIS CHAPTER

- A (non-exhaustive) catalogue of existing timber assemblies, spanning continents and ages, is presented to reveal the richness and technical know-how of past carpenters and to ponder on human ingenuity when it comes to designing such non-intuitive mechanical puzzles.
- An up-to-date review of the scientific literature on the computational design of interlocking assemblies is established.

2.2 DEFINITION

A mechanical assembly is the connection of different parts of an assembly or product. Assemblies are ubiquitous: whether it is the supporting structure of a house, the container and cap of a toothpaste tube, or a robotic arm transferring motion or applying force, everything from everyday objects to highly sophisticated scientific machines is, ultimately, parts and pieces holding together.
Despite this huge variety, assemblies can be broadly classified into two categories: an assembly is either direct or indirect. Each kind of assembly is also either reversible or non-reversible, meaning that once in the assembled state, it is possible, or not, to disassemble the assembly.

2.2.1 INDIRECT ASSEMBLIES

Indirect assemblies refer to assemblies where two parts are connected together through a third, intermediary body. This body may be:

- Glue is used to bind parts together by adhesion. Several examples could be given: a windshield is glued onto a car, parts of some home furniture are glued as well, and the mortar between two bricks is a glue... More often than not such assemblies are non-reversible.
- A staple is a fastener, often metallic, used for joining material together. Stapled elements are often quite thin, even though, in ancient times it was not uncommon in masonry works to tighten two stones using a metallic staple, see for instance the dovetail staples from Pasargadae, Persia, on the inset on the right (source: Wikipedia). This kind of assembly may be reversible, at the cost of the destruction of the staple.

- A key, sometimes called a wedge, is a small component that connects two parts in rotation. The assembly is reversible.
- A nail is a fastener, often metallic, used in construction and woodworking to pin two parts together, by friction in the axial direction and shear strength laterally. This kind of assembly may be reversible, at the cost of the destruction of the nail.
- A pin is a metal cylinder intended to be loaded in shear at relatively low forces. The connection is due to the adhesion between the pin and the connected parts. When the connection of a shaft to a hub is subject to high loads, a key is used instead. The assembly is reversible.
- A rivet, usually a cylindrical metallic rod with a "head" at one end, is a permanent fastener. It is typically used to join metallic pieces, be it on a plane, a ship or a bridge structure. The assembly is reversible.
- A fishplate is intended to immobilise several moving parts of a mechanical assembly (rails, members, chords, etc.) or to stiffen and support a soft or flexible body (cheese, member, etc.). The assembly is reversible.
- A screw is a mechanical part, made of a threaded shaft and a head; intended to fix one or more parts by pressure, several mechanical fasteners work in the same way: nut, bolt, stud, thread, tap etc. The assembly is reversible.

2.2.2 DIRECT ASSEMBLIES

An assembly is said *direct* when it does not require any intermediate bodies, the shape of the parts in contact is sufficient for its realization.

- Welding, one of the most common techniques, is a non-reversible joining process that ensures the continuity of the material between the parts joined. It is done by heating the parts. The continuity happens at the atomic level in the case of metal or molecular level in the case of plastics. Note that it is also possible to weld wood: two pieces of wood are rubbed against each other; the high temperature resulting from the friction as well as the pressure applied on the wood softens the lignin and hemicellulose at the interface, [34]. Once softened, these polymers get entangled, and the pieces of wood are welded together.
- What seems to be the most intuitive direct assembly is the friction fit: parts with complementary shapes are tightly held together by friction once they are pushed together. Following [107], as the shapes are complementary, we define as *integral joints* the portion of each part in contact with adjacent parts. Assemblies with integral joints that may optionally hold through friction will be the focus of this manuscript. In a friction fit assembly, parts may be forced together by hand or using a hammer or a press. Lego bricks are a typical example of such a fit. When the structural integrity of the parts is required and it is therefore desirable not to hammer them against each other, they may be joined using a shrink fit. Depending on the friction force, this kind of assembly may not always be reversible.

- A shrink fit aims at joining metallic parts by changing their relative size after assembling using thermal expansion. One part is heated and thus expands, making it possible to fit the other part, at room temperature, without using too much force. Once the first part cools down it shrinks (hence the name of the fit) and tightly holds the second part. It is a non-reversible assembly. As a side, yet quite interesting, note, the metal structure of the Grand Palais in Paris, France, (1897-1899) was assembled by shrink fit, using rivets (hence it qualifies as an indirect assembly): three workers would work together around one rivet; the first heated the rivet to a blank, while the other two hammered each end (which had become soft due to the heat) to round them and thus wedge the rivet between the parts to be joined [54]. Once cooled, the rivet shrinks and presses the two parts together.

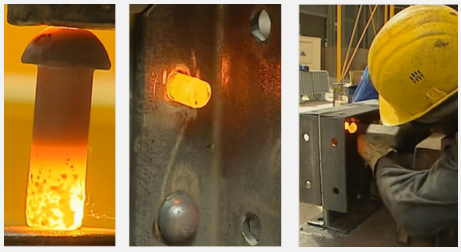


Figure 2.1| A rivet is heated, and inserted in the parts before its end is hammered. Renovations of the Grand Palais, Paris, France. Source: [1].

- Clinching is a method to join thin sheets of metal in a non-reversible manner: a tool presses several sheets together until they are plastically deformed and tightly interlocked.
- Snap-fit aims at attaching flexible parts. One of the two parts, the *male* part, is elastically deformed during the introduction into the more rigid *female* part. Once assembled, the male part has room to return to its original shape and is blocked by the female part. Such process may be reversible, for instance the snap-fit buttons of some coats and jackets or the cap of some pens, or non-reversible such as what can be seen on some mobile phones where unscrupulous manufacturers make sure that once assembled, the product cannot be disassembled and thus, when a component is defective it cannot be replaced and the whole phone must be thrown away and bought anew.

2.3 REVIEW

As hinted in the previous section, this manuscript studies exclusively the automatic design of reversible integral joint assemblies. Historically, several civilisations used such shape-fitting assemblies in their timber constructions, be it in the Alps, Scandinavia or Japan. Centuries of human ingenuity and trial and error gave joints that take into account the shrinkage of the wood as well as naturally balance shear, compression, bending and torsion. Also, as they are reversible, the design of such assemblies is an active research topic in the construction sector as they enable the reuse of structural members and as such help reduce the amount of new material that must be extracted and stir the industry towards a more circular economy. As remarkable examples of durability, consider the Japanese inns *Nishiyama Onsen Keiunkan* built in 705 or *Hōshi Ryokan* founded in 718: they have been in used for more than 1300 years and are fully dismountable. As such, whenever a part has decayed it can simply be replaced by a new one. As for the reusing, the Ise shrine, in Japan, has continuously been rebuilt every 20 years for more than 1300 years, while reusing as much as possible the parts from one instance to the other ([118]).

2.3.1 BUILT TIMBER ASSEMBLIES

It would be impossible to give an exhaustive list of past and existing timber assemblies. Indeed, as any carpenter would know, [118], wood is a capricious material that has a life of its own. Whether it is the overall shape of the piece of wood, the orientation of the grain, its density or the presence of bulks, no two pieces are alike. As such, the skilled carpenter adapts the geometry of the joint he/she aims to build to the nature of the wood at hand and the geometry of the structure surrounding the joint. That being said, a classification of wood joints is not totally out of reach, and some authors did try to list the most common ones [43,

77, 101, 118]. While Klaus Zwerger in [118] compares wood joints in Europe and Asia in a rather literary style and provides numerous pictures of built assemblies in these continents, Torashichi and Matsui in [101] have a more engineering approach as they provide a short description and pictures of toy assemblies, detailed diagrams if one wants to try to fabricate some, as well as, quite interestingly, mechanical test results to compare the geometrical resistance and failure modes of similar joints. They also provide, for some joints, the algorithm and geometrical formulae one should follow to carve them. In a similar spirit, Sato and Nakahara, [43] describe at length (and it resonates with some paragraphs of [118]) the tools needed to build traditional Japanese joinery, as well as the posture one should adopt on its workbench for each tool, before giving detailed and illustrated instructions on how to build such assemblies. They also provide a thorough chapter on where in a building should a given joint be used. Hereafter the enumeration of traditional timber joints is mainly inspired from [118] and [101], as well as the online dictionary [74].

Let us first begin with splice joints, aimed at joining structural members end-to-end. Numerous joints can be categorised as such, and are presented FIGURE 2.2. On the right of each joint, the cone of translational freedom of the blue part is depicted in blue: it means that the blue part may be disassembled from the black part by translating along any vector whose tip is on the blue cap. The simplest joint of all is the butt-joint, FIGURE 2.2a. It consists of two pieces of wood laid together without any real interlocking. The edge-halved scarf FIGURE 2.2b is also quite basic. It admits two variations obtained by rotating negatively, FIGURE 2.2c, and positively, FIGURE 2.2d, the cut line parallel to the grain of the wood. Note that contrary to FIGURE 2.2c, FIGURE 2.2d may withstand a limited amount of tension. A first kind of tenon joint is presented on FIGURE 2.2e: the mortise is open and the assembly may be (dis) assembled along planar translation only. This joint may withstand a small amount of tension if the tenon is inserted by force in the mortise, thus creating friction at the two interfaces parallel to the grain of the wood. Otherwise, the assembly may easily be pulled apart. FIGURE 2.2f presents a very common type of tenon joint, the stepped dovetailed splice (*koshikake aritsugi* in Japanese), to be used when the pieces of wood have to withstand both tension and compression. [101] specifies that the timber sections typically range between 105 and 120 mm. Special attention must be taken by the carpenter regarding the angle of the cheek of the dovetail: too small and the assembly could be still disassembled by tension. Too large and the shearing capacity of the joint would decrease. Torashichi and Matsui conducted a tensile test in [101]: the female part developed a crack along the grain of the wood, thus failing. Note that such dovetail joint and the edge-halved scarf (FIGURE 2.2b) were frequently combined to make the joint suitable for unsupported compressive loads ([118]). Used on larger timber pieces than the dovetail joint (150 to 200m), the stepped gooseneck splice (*koshikake kamatsugi*) was quite common in Japan, much less in Europe. A tensile test conducted on similar pieces of wood by [101] shows that the gooseneck joint has a stiffness about 7 times greater[1] than the dovetail joint and fails for a load 5 times greater. Two failure modes were observed at the head of the gooseneck: crushing of the wood on one side on the male part, and shearing on the other side on the female part. To prevent out-of-plane buckling and the male part slipping out of the female part, abutments could be added on both sides of the gooseneck, FIGURE 2.2h. For sections greater than 200mm that had to withstand tension forces, the rabbeted oblique scarf splice (*okkake daisen tsugi*), FIGURE 2.2i, is the most indicated. The two parts are identical. Additionally, to prevent lateral motions, two pins, depicted in red, could be inserted in mortises carved in the adhesion planes. While the pins are inserted, the blue part cannot be disassembled. The same tensile test shows a stiffness about 10 times greater and a maximal load 8.3 times greater than what was observed on the dovetail joint. A failure occurred through shearing through one of the adhesion planes. A close cousin is the mortised rabbeted oblique splice (*kanawa tsugi*) on FIGURE 2.2j. Parts must be assembled first vertically, then they must slide longitudinally onto each other to create room to insert a single draw pin, in red. Lateral motions are blocked by the abutments. According to [101], for the same tensile test, the ulti-

[1] Stiffness ratios were eyeballed from graphs in the book.

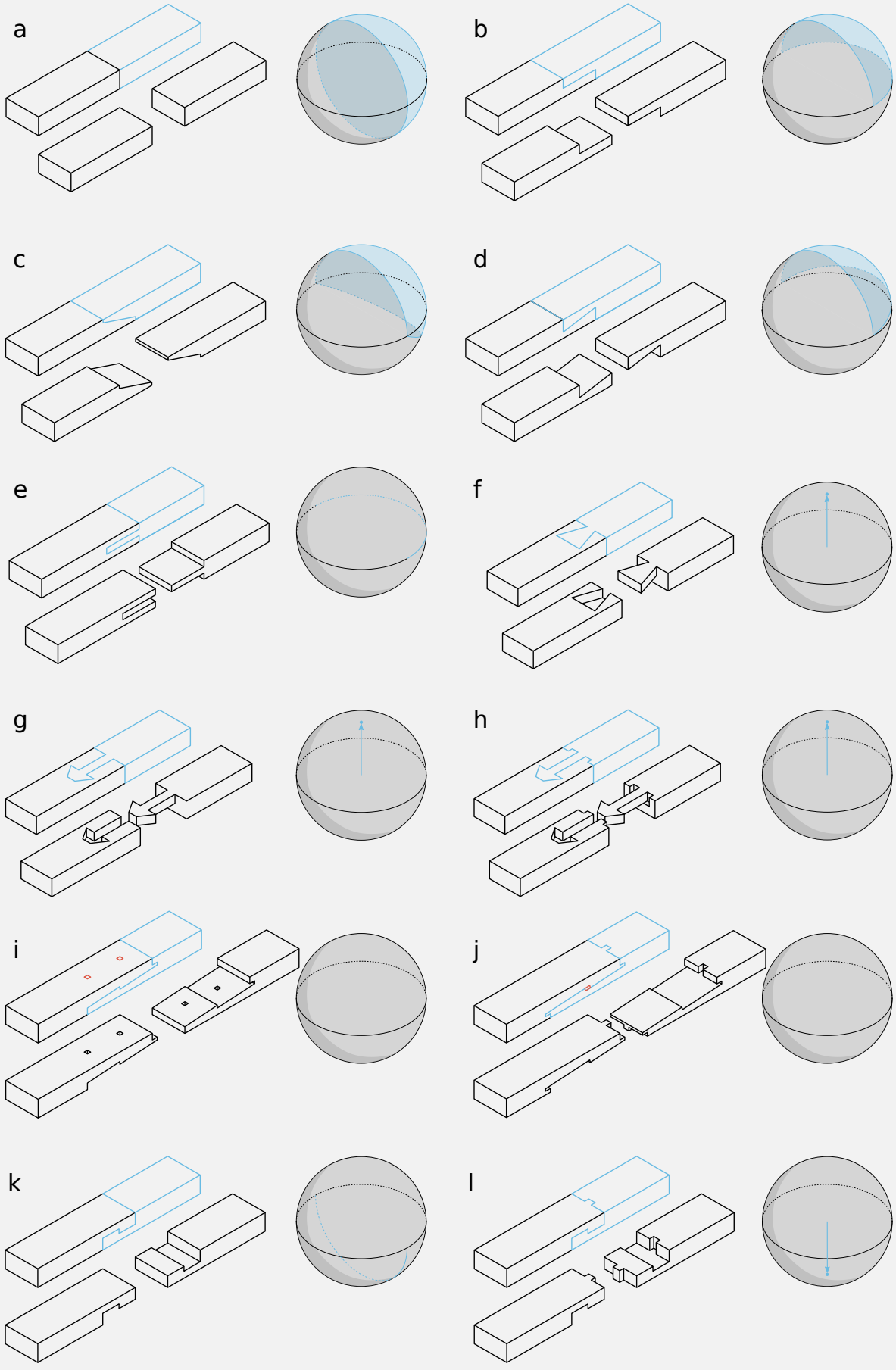


Figure 2.2| Several common splice joints and the possible translational disassembly kinematics represented on the sphere.

mate load is the same as for the previous joint. However, the nonlinear behaviour of this joint is much less stiff than the previous one. Two other joints, suitable to carry tension forces are depicted FIGURE 2.2k,l: the halved and tabled joint and its abutted version respectively.

Tenon joints are also a very large group. The oldest example known so far of a tenon and mortise joint dates back to 6000-4000 BC and was found in ancient water well in Germany [98]. Examples of such joints were already given FIGURE 2.2e,f,g,h, but the list goes on. They are used to assemble wood pieces end-to-end as well as orthogonally, FIGURE 2.3a. Among the significant joints, one may cite the most common one, be it in Japan or Europe FIGURE 2.3b. In the case of two orthogonal pieces of wood, the mortise may be open so that the tenon goes completely through it. In such a case, a pin may be inserted in the tenon to prevent disassembly. Otherwise, such joints are not suitable to carry tensile forces and, consequently, are sometimes combined with other joints (or fasteners) to prevent accidental disassembly. FIGURE 2.3c presents the cross-shaped tenon and mortise splice (*juji mechiire*), effective against torsion, [101], but having the inconvenient of showing a jagged line across all faces. A more aesthetic option is given FIGURE 2.3d (*kaneori mechiire*), where two faces out of four reveal a straight line. On FIGURE 2.3e (*kakushi mechiire*), only one face shows a jagged line. Note that this is the only joint in this figure where the disassembling motions are not restricted to the longitudinal axis. Finally, a technically difficult but aesthetically pleasing 2.3f (*hako mechiire*) shows only a straight line on all four faces of the assembly.

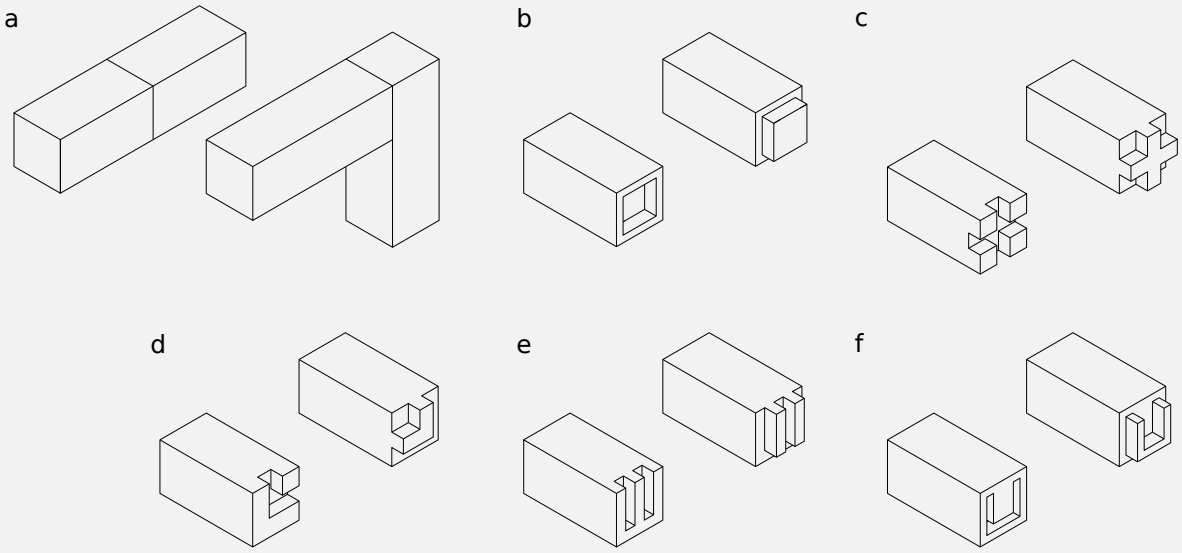


Figure 2.3| Several common tenon and mortise joints. Depending on the geometry of the joint, the seam between the two parts may be visible on a varying number of faces. After [101, 118].

If a mortise and tenon joint is employed to join pieces of wood edge-to-edge, *i.e.* is the direction of the grain, it would be referred to as tongue-and-groove, to assemble floor planks or mural panels, FIGURE 2.4b (*honzanehagi*). Hardwood should be used. According to [118], such assembly is not very stable and should be nailed to a supporting frame. To qualify this statement, such boards were used as mural panels in some stave churches in Scandinavia and stood the test of time. The simplest of such an edge-to-edge joint is the butted boards (*imohagi*), see FIGURE 2.4a, which does not constitute a proper assembly. A very stable edge-to-edge assembly is presented FIGURE 2.4c: a dovetailed plank is inserted in a corresponding groove orthogonal to the direction of the butted planks. It joins them into one unified surface, particularly fitted for flooring. The V-groove joint of 2.4d (*hibukurahagi*) is not strong enough for flooring but works well for panelling. Similarly the rabbeted joint 2.4e (*chigaihagi*) was good for paneling and exterior siding. It could also replace the abutted boards of the dovetail joint 2.4c ([118]). Finally, the spline joint of 2.4f

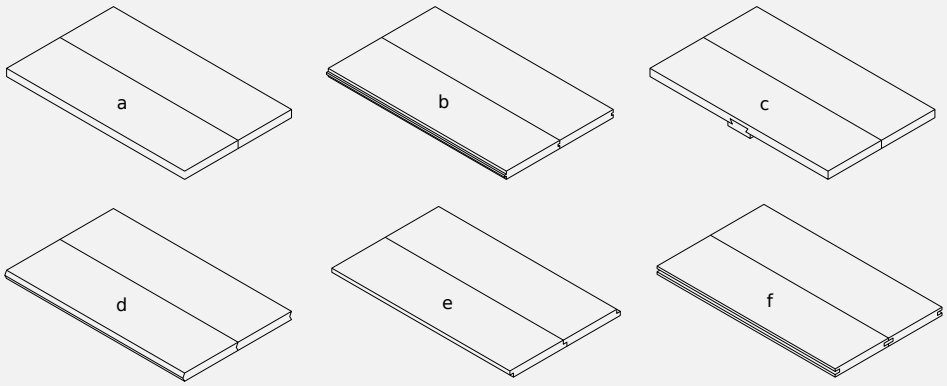


Figure 2.4| Some floor and panel joints, after [74, 101, 118].

(*yatoizanehagi*) should be made of hardwood and inserted into a groove cut in the sides of planks made of softwood.

A particularly interesting category of splices that were quite common in Japan but virtually unknown in Europe [118] were the family of the edge-and-faces-halved scarf, also called rabbeted oblique scarf splice in [101]. They were decorative splices with no particular structural application. Contrary to the joints presented FIGURE 2.2 which could be used both horizontally and vertically, the oblique scarf splices could only be used horizontally. The length of the inclined plane of the halved rabbeted oblique scarf splice, shown on FIGURE 2.5a (*isuka tsugi)*, should lie between once and twice the side of the cross-section. The shorter the more structurally able ([101]). To this design could be added abutments and/or a key to prevent disassembly, as depicted 2.5b. Note that if the key is added, as depicted in red in the figure, it does not reach through the lower surface. It was used where the outer appearance of the joint was of no concern, typically the dormitories of temples' monks, [74]. The elegant quadruple-faced halved rabbet oblique scarf splice, 2.5c, requires sophisticated craftsmanship to complete it. A variant, also technically difficult to manufacture, is the triple-faced halved rabbeted oblique scarf splice (here with a key) on FIGURE 2.5d.

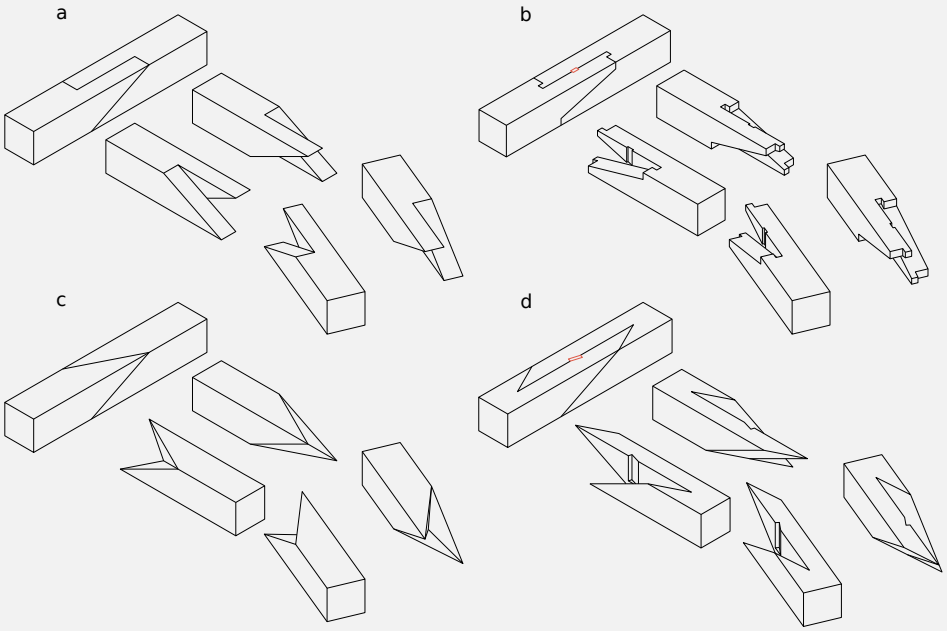


Figure 2.5| Decorative oblique splices, after [101, 118].

Several other joints destined to finishing material are listed by [101] and [43] and are presented on FIGURE 2.6. The housed rabbeted oblique scarf splice 2.6a (*kakushi kanawa*) produces a clean straight line on

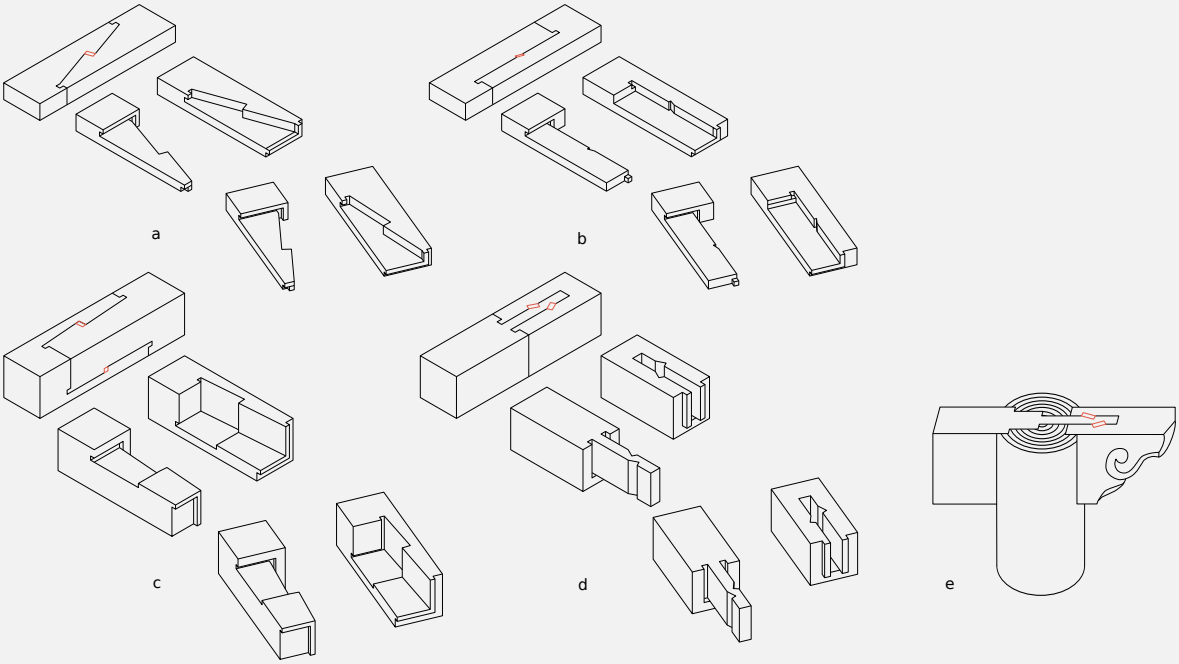


Figure 2.6| Decorative oblique splices. The geometry of the joint leads to various seams on the visible faces of the parts. After [101]. The *saotsugi* illustration (e) is adapted from [43].

what should be the two visible faces (these two faces are not shown in the figure). Its close cousin is the blind tenon and mortise 2.6b (*hako daimochi*). The blind pin 2.6c (*hako sen*) has the particularity of having two keys. It is said to be cumbersome to carve [101]. The pole tenon 2.6d (*saotsugi)* is used on exposed ceiling elements. Compared to a dovetail or a gooseneck joint (FIGURE 2.2f,g,h) the assembling motion of the two pieces is longitudinal and only the keys are to be slid down vertically. Such a joint is necessary to assemble elements crossing a post, (which blocks any vertical motion and thus forbids the use of the dovetail or gooseneck joints), as illustrated 2.6e.

Crossed right-angle joints were used to connect two horizontal orthogonal members of a roof or floor structure, see FIGURE 2.7. These joints are simple enough so that they can easily be adapted to non-orthogonal beams. The simplest of such joints, the stop dadoed crossed lap joint of 2.7a (*watari kaki*) consists simply of a notched beam inserted on top of another. A slightly more complex version consists in hollowing partly the other beam so it also has a notch, 2.7b, (*watari ago*). An elegant way to connect two beams so that the whole assembly is of constant height is to use a cross-lap joint 2.7c (*ai kaki*) where each beam is hollowed at half its depth. FIGURE 2.7d depicts an original joint (*tasuki kake watari ago*) where two opposing quadrants of a cross are hollowed out.

To assemble a column with ties, groundsills, girders etc. so-called connecting joints are used, FIGURE 2.8. The *suitsuki sashi shikuchi* of 2.8a consists of a tenon and a dovetail groove in which the horizontal part slides during (dis)assembly. The housed dovetail joint 2.8b (*okuri ari*) was frequently used on hanging posts ([101]). The dovetail is inserted in the largest opening before being slid to the narrower complementary hole. A plug (not represented in the figure) can be pounded in the larger hole to prevent easy disassembly. FIGURE 2.8c depicts the sumiyoshi double tenon, named after one of the coauthor of [101]. Despite the easiness of assembly, as the tenon simply slides diagonally until reaching its final position, this joint seems to be cumbersome to fabricate. FIGURE 2.8d shows a wedging joint (*wari kusabi*), an original joint aimed

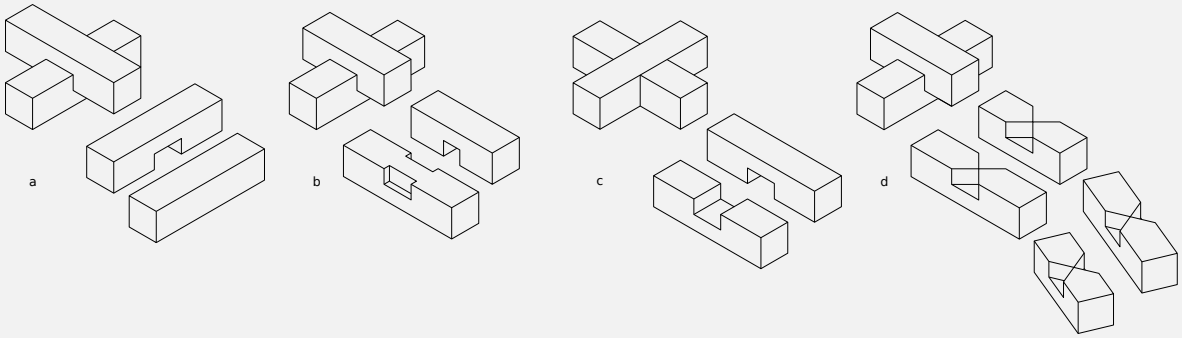


Figure 2.7| Crossed right angle joints. By tuning the depth of the hollowing the assembly reaches various heights. After [43].

at creating a non-reversible[2] assembly. Two small notches (blue) are cut on an otherwise standard tenon. Once inserted in the mortise and in the final position, two wedges (red) are pounded inside the notches, which has the effect of splitting the tenon open and locking it, by friction, on the side of the mortise. A blind version *jigoku hozo*, *i.e.* a version where the tenon does not reach through the full depth of the other beam, is shown 2.8e. There the tenon beam is pounded on its back side until the tenon is split. The half dovetail joint 2.8f (*katasage ari*) connects a tie to a column. The dovetail is inserted by sliding its upper face (the non-inclined plane) on the top of the mortise hole. Once fully in, it is slid downwards until the small abutment at the start of the dovetail, lower face, fits in the corresponding hole in the column. This opens a rectangular hole on top of the joint. Finally, a plug is driven in this hole to prevent accidental disassembly. The same process happens for the *nimai kama tsugi* 2.8g: the mortise hole is made taller than the extent of the two horizontal beams to leave enough room to slide one over the other. Once the end sides of the beams are touching each other, the upper beam is slid downwards and two wedges are symmetrically pounded in the hole to lock the assembly. FIGURE 2.8h shows how several joints are used together to create a sophisticated assembly connecting three beams to a column. The shorter male piece, a simple tenon with a hole for a pin, is inserted in the column and locked in place thanks to a transversal draw pint. The two other parts, long male and female pieces, are simply a pole tenon (*saotsugi*) whose working is presented FIGURE 2.6d,e with the addition of a transversal pin to further prevent any disassembling motion. When seen from below, this assembly seems seamless.

Let us finish this overview of built timber assemblies by the column splices FIGURE 2.9 (for illustrative convenience, the assembled columns are shown lying on their sides). Suitable for large sections of wood, these joints were used at the base of gates, shrines or belfry posts, [74]. Hardwood should be used, [101]. The first of such joints are the four faces dovetail splices (*shiouhari*) where two dovetail joints (see FIGURE 2.2f) are carved on two adjacent faces of the post and then extruded diagonally to the other two faces. As such, the joint has the particularity of showing a dovetail on each of its four faces. Its close cousin, the four faces gooseneck joint 2.9b, is better able to withstand tensile forces, but much harder to manufacture. Both of the joints should be assembled diagonally, in the direction of the extrusion. FIGURE 2.9c shows a splice that has only been found at the Otemon gate of Osaka castle. Contrary to the previous two joints, it should be assembled in a diagonal-downwards direction. While the blind splice, shown on 2.9d (*hako tsugi*), is said to be inferior in strength to the rabbeted oblique scarf splice (see FIGURE 2.2i, [101]), this joint has the advantage of being seamless, as the jointing lines are on the edge of the section, thus making it attractive when aesthetics is required. The last joint of this series is the clam-shaped splice (*kai no guchi*) which is to be assembled vertically (which is inconvenient for underpinning) and is, in a sense, similar to the *tasuki kake watari ago* shown on FIGURE 2.7d. Two opposite quadrants of a cross are hollowed, and the extremities of

[2] That cannot be disassembled.

each quadrant exhibit a tenon-and-mortise pattern. This joint was used for prominent elements, [101].

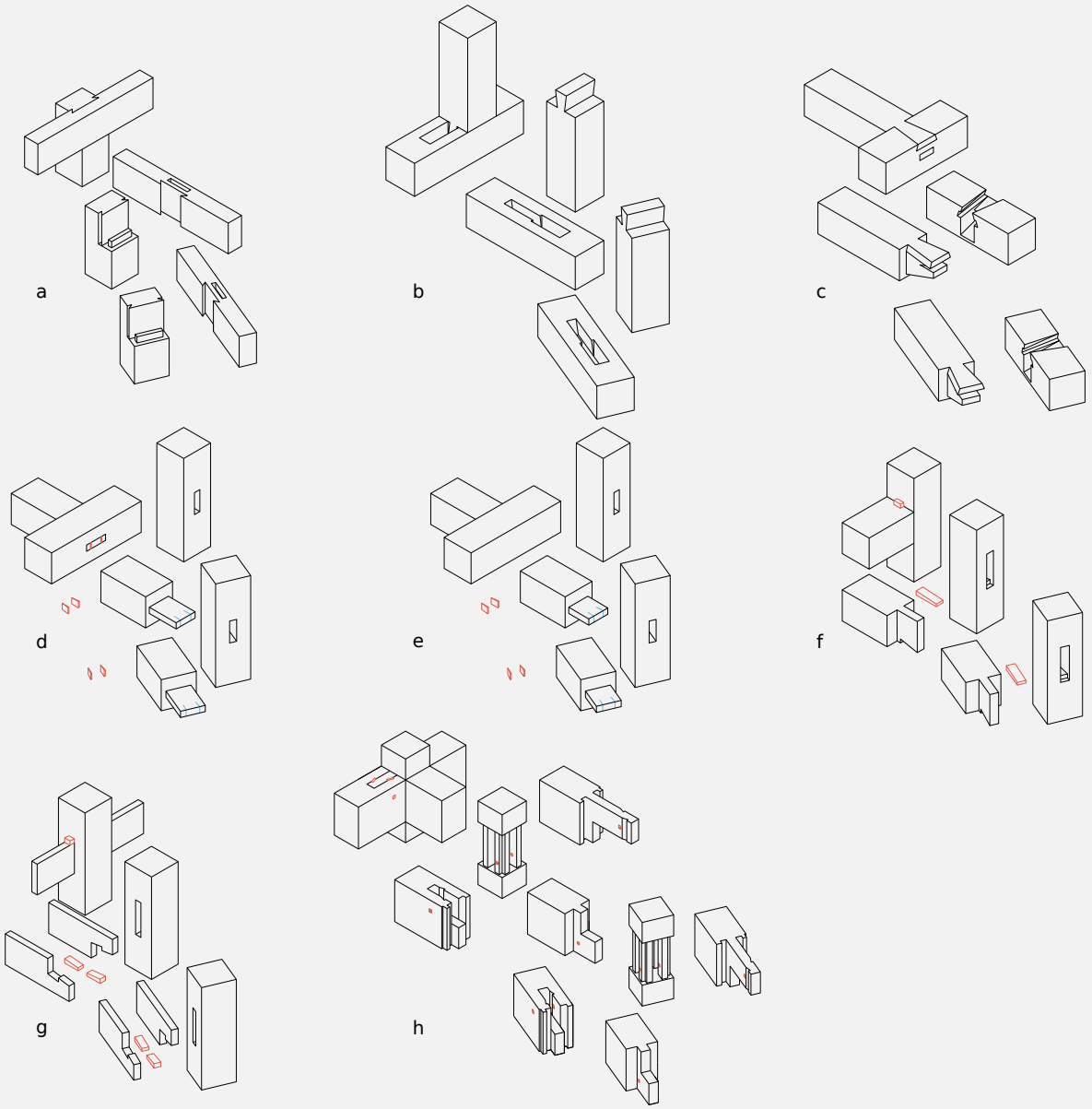


Figure 2.8| Connecting joints. When notches are shown (in blue, d) and e)) the wedges (in red) make the assembly permanent. After [101].

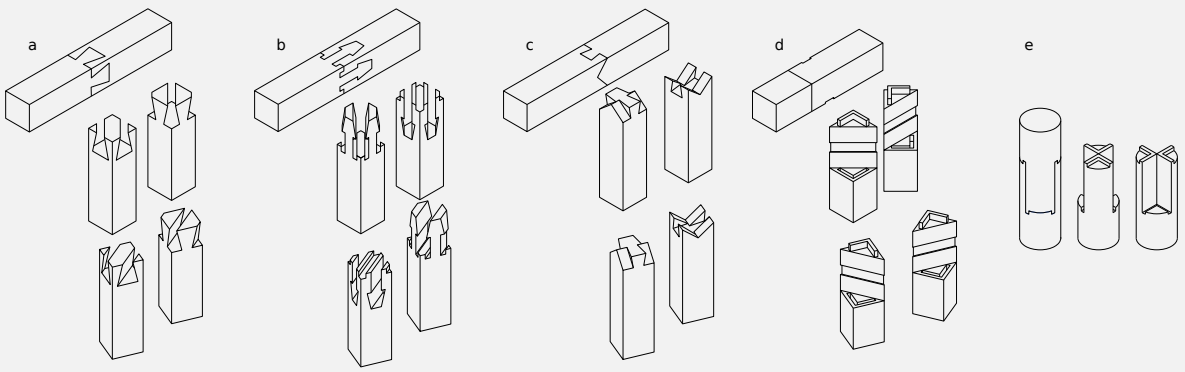


Figure 2.9| Some column splices. Of large section these assemblies were used on prominent columns. After [101].

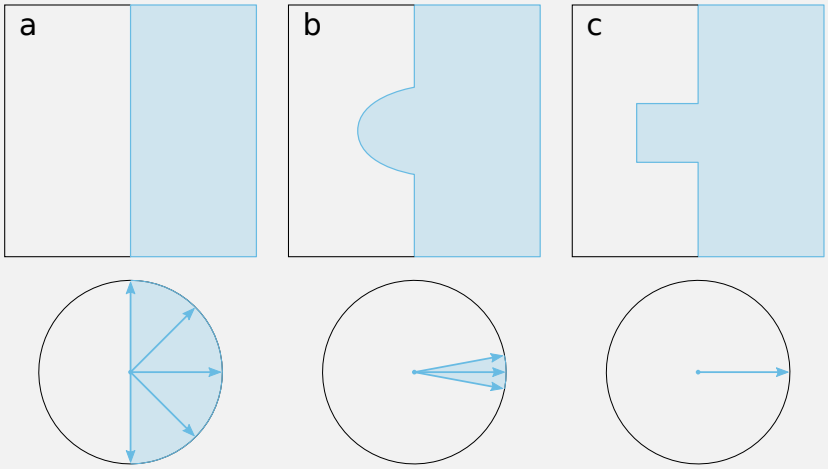


Figure 2.10| Three kinds of integral joints. a) - planar contact joints; b) - curved contact joints; c) - traditional (here tenon and mortise) joints. The blue cones on the bottom row depict the cone of translational freedom of the corresponding blue part. Blue arrows represent several valid directions of translation to disassemble the parts. Adapted from [107].

This brief overview shows us the extent, in form and function, a skilled carpenter might reach when building complex joints. As we have just seen, this set is very rich but is ultimately limited by the manufacturing tools: saws and chisels can only cut straight edges. Thus, traditional joinery leaves unexplored a large range of design options. With modern digital fabrication technologies, such as CNC milling, robotic manipulation, laser cutting and additive manufacturing, the space of manufacturable assemblies get much broader. The will to explore this space has pushed numerous researchers to investigate various computational approaches to design interlocking puzzles as well as robotic assemblies. SECTION 2.3.2 presents a review of the scientific literature on this subject.

2.3.2 COMPUTATIONAL DESIGN OF INTERLOCKING ASSEMBLY

2.3.2.1 Designing interlocking assemblies

Interlocking assemblies are defined by Song *et al.* in [89] as an assembly of rigid parts such that only one of them, the *key*, is movable while any other part or subset of parts are immobilised relative to one another. The literature on the subject is rich with, for instance, [117] who designed furniture joinery and study the stability of the structure, [53] who introduces a remarkable software to design wood joints with a special focus on fabricability, or [105] who builds a framework aimed at generating novel assemblies and presents examples of voxelised puzzles. These approaches can be categorised into two families: they are *catalogue-based*, meaning that possible joints are predefined in some catalogue (or similarly that the user is supposed to already have some knowledge of the geometry of the joint) or *voxel-based* which limits the space of accessible shapes for a given voxel resolution.

Catalogue-based designs:

As recalled by [107], there mainly are three kinds of integral joints[3], shown on FIGURE 2.10, and numerous the authors who derived assemblies whose joints are of one of these types. The simplest of all, planar contact joint 2.10a, is typically used in masonry modelling, be it to procedurally model historically-inspired buildings [111] stable under self-weight (see FIGURE 2.11a), puzzles [29] 2.11b or, a rather rich field of research, to design topological interlocking assemblies. In [28] Estrin and coauthors define *"a topological*

[3] Joints created by the complementary shapes of parts in contact.

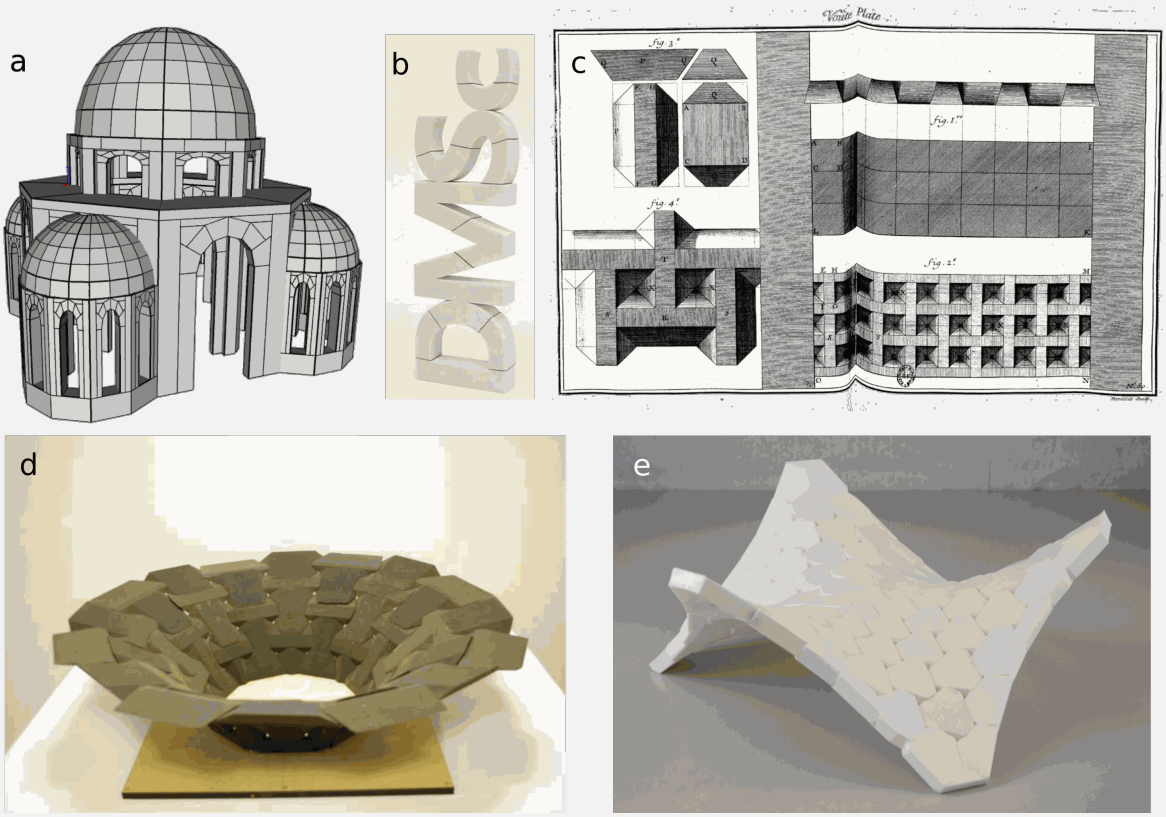


Figure 2.11| Assemblies joined through planar contact. a)- masonry modelling, from [111]. b) - stable puzzle, from [29]. c) - a drawing of an Abeille's vault, from [32]. d) and e) - topologically interlocked structures, respectively from [56] and [108].

interlocking is a design principle by which elements (blocks) of special shape are arranged in such a way that the whole structure can be held together by a global peripheral constraint, while locally the elements are kept in place by kinematic constraints imposed through the shape and mutual arrangement of the elements". To the best of our knowledge, the first recorded example of such assembly is Abeille's flat vault, [32] 2.11c, patented in 1699, which consists in the geometric arrangement of a single repeated truncated tetrahedron. Two faces of the block are supported by two adjacent blocks while two others faces carry two neighbouring blocks. The structure assumes a fixed surrounding frame. This geometrical finding was further extended in numerous works, with for instance [56] (2.11d) or [108] (2.11e) who, by slightly modifying the geometry of each convex part, find a stable (under gravity) topologically interlocked design approaching a free-form goal surface. In [109], Weizmann and coauthors study the mechanical behaviour of floors made of various types of topological interlocked blocks. They found that it led to an increase in material consumption compared to more traditional flooring techniques and finish their article by applying topology optimisation on a block already possessing the interlocking qualities in order to reduce its mass. Wang *et al.* [108] also developed an optimisation scheme, but this time to make a topological interlocking structure stable under a given amount of lateral loading. The mechanical behaviour of such Abeille's type bound has also been studied extensively in for instance [17] and, in a nutshell, the results are that in addition to the ease of (dis)assembling such structures, they exhibit large energy absorption capacity, high resistance to crack propagation and tolerance to local failures [28].

The second class of integral joint is the curved contact, shown on FIGURE 2.10b. Quite interestingly, according to J. Gallon in [33], father S. Truchet was dissatisfied with the inverted-pyramidal hole left in the extrados of Abeille's surface (see FIGURE 2.11c). As a consequence, he developed right after Abeille a non-

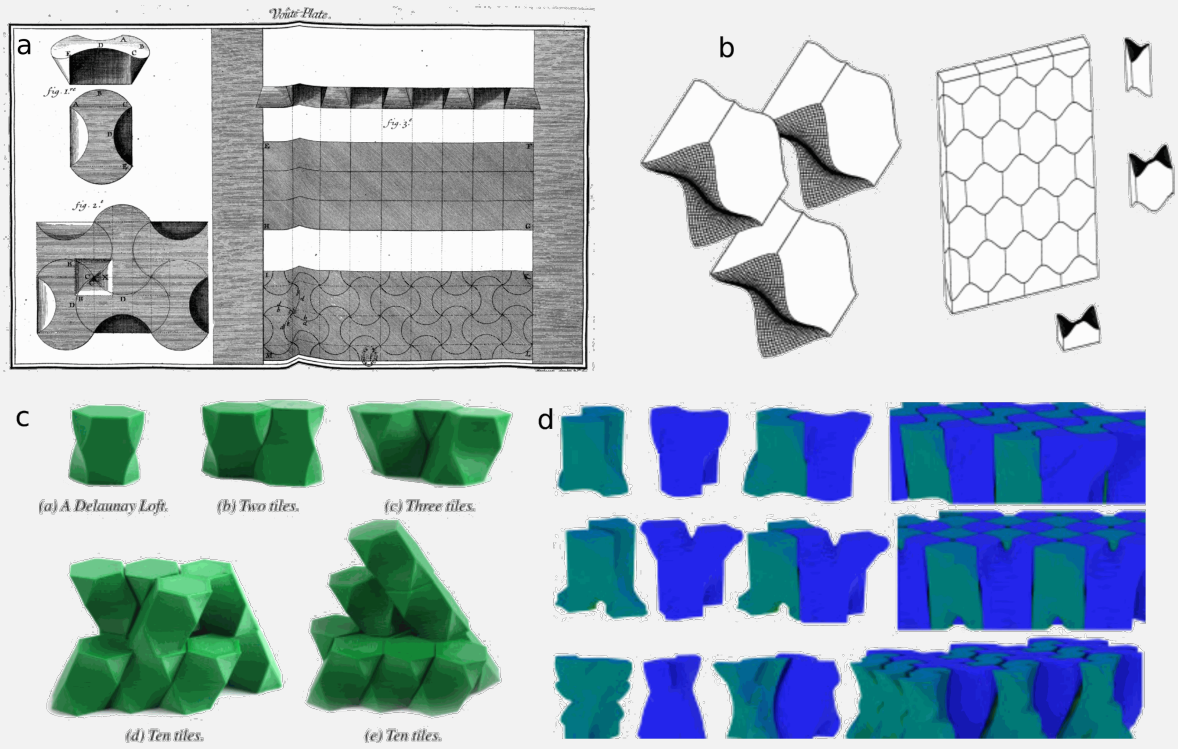


Figure 2.12| Assemblies joined through curved contact. a) - Non-convex Abeille's vault, from [33]. b) - osteomorphic blocks from [27]. c) and d) 3D tile components, respectively from [94] and [5].

convex element so that both the intrados and extrados of the surface would be filled, see FIGURE 2.12a. The principle stays the same: a repeated element carries two neighbours and is supported by two others. Extension to this work were made in [27] with the so-called osteomorphic block (2.12b), in [28] or recently in [4]. The mechanics of curved topological interlocking assemblies are studied in [26]. In a different spirit, curved assemblies are also used to tile 3D-space with Delaunay lofts [94], 2.12c, or bi-axial generalised Abeille's tiles [5], 2.12d. Wang *et al.* built a tool in [106] aimed at creating curved cone joints between user-given parts. Their work is hybrid: while a joint is always modelled by a cubic spline (or a discretised counterpart) and thus their method qualifies as catalogue-based, the specific shape of each joint is optimised, taking into account the geometrical features of the other parts and some stability measure, thus making each joint different from the others.

The third and last kind of integral joints listed in the survey [107] are conventional, or traditional, joints. Significant work has been made to automatically generate assemblies made of parts chosen in a catalogue of such conventional joints: several authors [20, 44, 87], investigated the creation of 3d assemblies made of planar pieces that are laser cut. The joint is always a notch in at least one of the two parts in contact. Testuz and coauthors [99] build hollow 3D shapes using a finite set of Lego bricks, which are mortise and tenon. Lo *et al.* [55] create 3D puzzles by meshing a given surface using quads, extruding them to give the puzzle some depth and merging adjacent quads to create polynomios (Tetris-like shapes). Careful consideration of the motion space for each polynomio gives an assembly direction and thus the orientation of the tenons and mortises assembling two parts. Xin *et al.*, [115], explore the partitioning of 3D shapes into a set of 6-parts burr puzzles, consisting of notched sticks, much like what is shown on FIGURE 2.7a and c; Fu and coauthors [30] developed a method aimed at creating global interlocking furniture assembly from a model of orthogonally intersecting 3D shapes and generated the joints from a lookup table containing, among others, different kind of dovetails and mortise and tenon joints. In a different spirit, Luo *et al.*, [57], partition a 3D shape into printable parts and assembles them through mortise and tenon kind of joints and Yang *et al.*

[116] present a material-aware algorithm to modify the overall shape of a structure but connects parts with mortise and tenon joints.

The interested reader is referred to the survey [107], which thoroughly reviews the literature on several aspects, unexplored in this manuscript, regarding assemblies with rigid parts: structural stability of an assembly, packing efficiency, fabrication aware assembly (3d printing, CNC machining), or reconfigurable assemblies.

Voxel-based designs:
Most of the other methods available in the literature to generate interlocking puzzles are voxel-based which restrict the assembling motions to the three canonical orthogonal directions of $\mathbb{R}^3$: [89] focuses on recursive interlocking puzzles where at each step of the assembly sliding sequences of 3 parts are tightly interlocked. Thus only one assembling sequence is possible. This work was later refined in [90] who carefully subdivides an input mesh into a set of voxelised parts such that every $K \geq 3$ parts are tightly interlocked, FIGURE 2.13a.

Yao *et al.* [117] implemented a tool that asks the user for the exterior appearance of the joints between structural components and automatically computes the internal solid geometry needed to connect and assemble the parts. Remarkably they were able to rediscover many traditional Japanese joints with intricate geometry. The main drawback of their approach is that the assembling motions are restricted to a set of 26 directions of translation in 3D (8 in 2D).

A compelling study has recently been made by the authors of [53]: their work about the design of wood joints is both voxel-based and catalogue-based (but supports adaptation to non-orthogonal and non-square joints by linearly deforming the grid of voxels) and primarily focuses on interaction with the human user, fabrication and mechanical relevance. An example of a built joint obtained through their method is presented on FIGURE 2.13b.

Aharoni *et al.* wrote an elegant paper [3] that does not fit in any of the above categories. They generate multi-part interlocking assemblies through a density-based topology optimisation scheme taking into account both the assembling (which must be as easy as possible) and the interlock level (as high as possible). A prominent feature of their work is the fact that the structural behaviour of the parts is considered and guides the optimisation. The main drawback is that the user must specify in advance which parts are in contact, and what motion should each part block with respect to its neighbours. As such it is a tool for optimisation rather than exploration, more suitable in later design phases. We shall focus on an earlier stage, where the user inputs less information and thus enjoy a greater design freedom, but where no structural optimisation is performed.

2.3.2.2 Disassembly planning

Assembly planning (or its pendant disassembly planning) refers to the problem of finding a sequence to fully (dis)assemble the parts constituting an assemblage, [48]. Several methods were developed and are thoroughly reviewed by [52], [107]. An interesting approach, on which we put an emphasis, was first proposed in [113]: given an assembly made of various parts, the authors introduced the concept of *Non Directional Blocking Graph* (NDBG) to encode blocking relations between the parts in directed graphs. By analysing these graphs the authors are able to find, for each step of the (dis)assembly process, which set of parts to move and what motion to follow to perform the task. While the method works theoretically both for translation and rotation, in practice they implemented an algorithm that "*considers all pure translations plus some suggested generalised motions*" without adding more details. Their method, for translation only, was later improved in [85] which makes further use of local contact information between parts.

Wang and coauthors in [105] were the first to leverage the kind of graph analysis introduced in [113] to

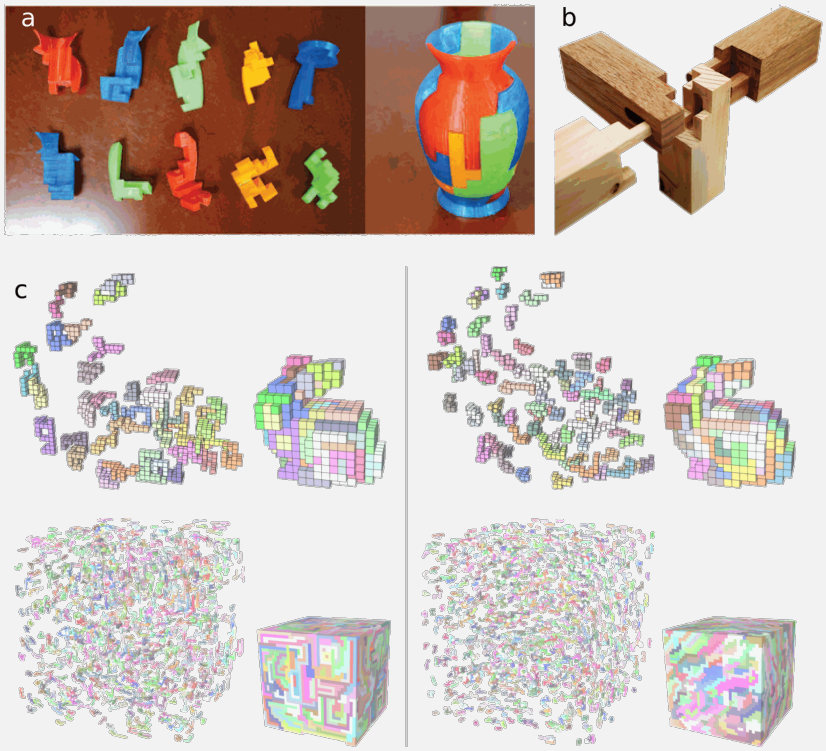


Figure 2.13| Voxel-based assemblies. a)- 3D printed pieces forming an interlocked vase-like puzzle, from [90]. b) - Complex joint obtained by [53]. c) - 3D puzzles automatically generated by [105].

automatically generate voxel-based interlocking assemblies. They designed an efficient puzzle generator that can be assembled along orthogonal translations. Even though they restrict themselves to voxelised structures, the authors fully explore the accessible design space and successfully manage to generate globally interlocking pieces without much computational effort, see FIGURE 2.13c. Their work serves as the basis of ours.

2.3.3 ROBOTIC ASSEMBLY

Among many possible solutions, a route worth exploring to assemble in practice the assemblies involves robots helping skilled human workers to assemble reversible structures. Several national and international initiatives are already undertaken in this direction, DFab at ETH Zurich, the Cluster of Excellence in Stuttgart, and the DiXite project in France, of which this thesis is a part.
In an environmental stance akin to ours Kunic and colleagues [51] developed a set of 13 timber elements that can be assembled pairwise in numerous ways; the discrete design space has a size combinatorial in the number of base elements, similar to a Lego kit. In a seamless workflow, a stress-optimised structure is assembled by a robotic arm working collaboratively with a human through reversible bolts and nuts assemblies. The human is made necessary by the build-up of errors and tolerances along the construction of the structure. Individual elements can therefore be extracted at the end of the lifetime of the structure, and their versatility makes them suitable for being assembled in a different geometry. Apolinarska *et al.* further automatised the assembly process in [6]. They used reinforcement learning to have a robot learn to assemble beams through integral lap joints subjected to tolerance, by leveraging contact information between the parts, see FIGURE 2.15a. In their book [12] (chapter 6) Bock and Linner argue that tolerances and accuracy are of major importance in robotic fabrication give guidelines to improve the construction design: the

overall number of joints should be low, the joining mechanism should be simple (riveting instead of screwing for instance), and the joining system should be standardized. They also introduce the idea of compliant design to help for the self-adjustment and self-fixation of the parts of the assembly, see FIGURE 2.14, where the edges of the separating surfaces between the parts are bevelled so that when assembling, any error on the locations of the parts can be corrected by sliding. In this context of toleranced assembly, the geometrical features of the parts guide the assembling agent. In the course of this work (SECTION 4.3.4) special care is taken to robustly optimise the assemblies generated through our method to ease the (dis)assembling process.

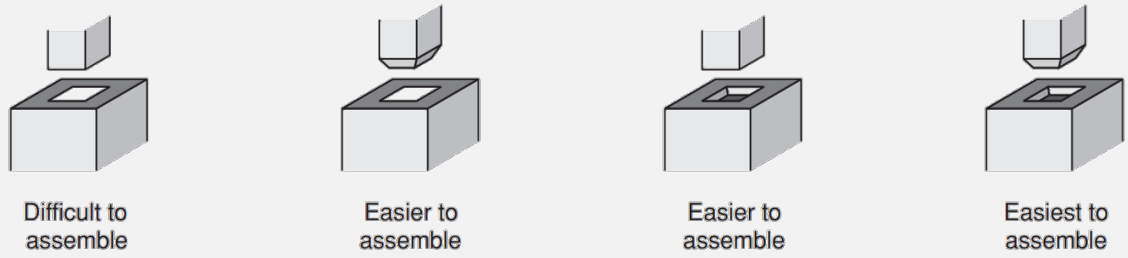


Figure 2.14| Compliant design: mechanisms for self-adjustment and self-fixation can be built into components. Source (caption and image): [12]

On the subject of robot-based timber assembly, Helm *et al.* [42] carefully designed a workflow to leverage the flexibility and capacity for precise manipulation of a robotic arm to create an intricate truss structure where beams are glued onto each other. On integral joints, Robeller and Weinand [83] designed a pavilion with folded geometry by assembling timber plates edge-to-edge through CNC-milled dovetail joints. Mesnil *et al.* in [61] present a computation and fabrication workflow of a shell-nexorade hybrid. The high manufacturing complexities of the joints made it necessary to use synchronously two robots forming a 6-axis milling machine.

The use of cooperative robots has also been extended to build non-planar structures. In [70] Parascho and coauthors used two robotic arms to sequentially built a structure made of triangulated discrete steel tubes. The robots switch their roles iteratively, alternating between a placing and supporting role: the first robot manipulates a steel tube to its position in the structure, maintains it while a human welds it to other tubes, and then acts as temporary support to hold the unstable structure. The second robot then becomes the manipulator of another tube before becoming in turn the support, which frees the first robot to go back to being a manipulator. This alternate process goes on until the completion of the structure. A prominent challenge with this approach lies in the fact that the robots must be carefully coordinated to avoid collisions. Yet, aside from making scaffolding obsolete, this method has the great advantage of increasing the space of topologically and geometrically feasible structures, compared to using only one robot. This process was later refined in [71, 72] where a masonry vault is built in tandem by two brick-laying robots: first, a central arch is built by both robots, and then each completes half of the structure. On top of the cooperation needed to address such a challenge, this research also used the full payload capacity of the robots, for them to support the self-weight of the bricks, see FIGURE 2.15c and d. Yet there are two problems with this method: when supporting a brick of the central arch out-of-plane motions arise due to an offset support point, which forbids the scalability of this method to larger structures. To circumvent this issue, the authors had the idea to use a mortar (epoxy) between the bricks, so that a few consecutive bricks could support their self-weight in tension, letting the supporting robot only grip the bricks in the line of thrust of the arch, which prevents bending moments. Yet again, this method is not scalable for larger structures due to the time needed for the mortar to take, so the researchers envisioned adding a third robot to the process. An optimisation program is executed at each step to select the brick that should be gripped by the third robot to optimise various

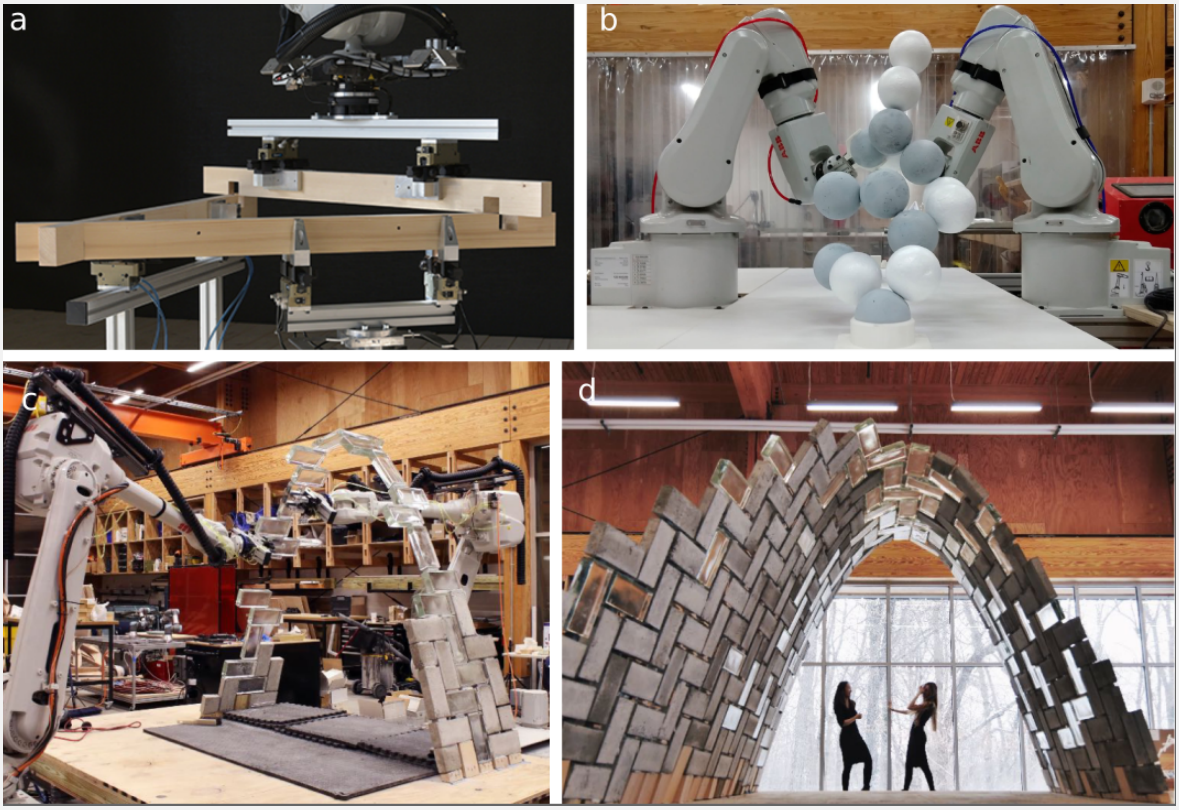


Figure 2.15| a) - Through reinforcement learning, a robot learned to join beams, from [6]. b) - Creative robots dynamically build a discrete-sphere structure, from [18]. c) and d) - Cooperative robots build a masonry vault, from [72].

construction-related metrics. Again, the three robots alternatively switch roles. Finally, on the subject of human-robot cooperation, an exciting paper [18] was recently written by Bruun and coauthors. From the observation that in the aforementioned papers (and others) robots are always used as performers following a predefined execution plan but never as designers, the authors proposed an original design process where a duo of robots and a human operator communicate on the design *while it is being constructed*, allowing it to change dynamically during the erection process; neither the robots nor the human know in advance the structure. In concrete terms, a discrete sphere structure is incrementally built by two robots in tandem along the aforementioned process support-then-place, see FIGURE 2.15b. At each step, the location of a new sphere is randomly decided. If the placing robot evaluates it to be reachable, the location is then proposed to the human who validates it based on arbitrary criteria (aesthetics for instance). The central place the authors give to randomness ensures that the geometrical and topological spaces are thoroughly searched, which is also something we put an emphasis on in our work. The evaluation made by the robot reduces this broad space to the smaller feasible one, while the evaluation made by the human further bias this space to the desirable one.

2.4 PROBLEM STATEMENT

Generating interlocking assemblies is a difficult geometric challenge ([89]) and the methods reviewed in the literature attempt to simplify the problem by making strong assumptions on the shape of the assembly and the (dis)assembling motions which negatively impact the freedom needed to design novel assemblies.

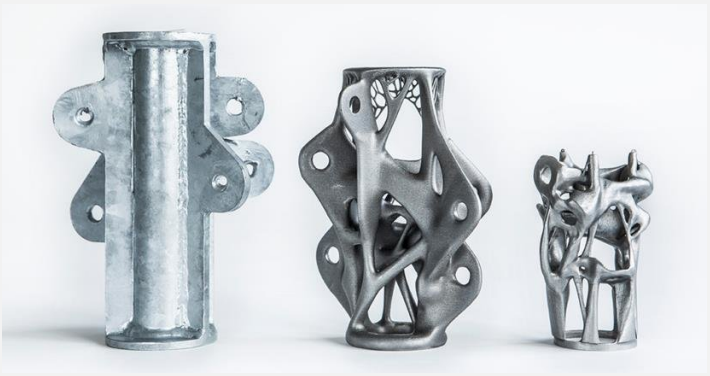
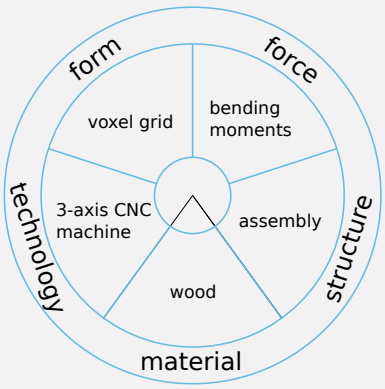


Figure 2.16| Digital technology makes it possible to envisage completely different methods of application, beyond traditional carpentry. Source: [68].



We argue that these assumptions ultimately stem from the fact that designing interlocking assemblies is essentially a *wicked problem*: each problem is unique, can be approached by many different methods, infinitely many designs are solutions to it, and because of competing goals (ease of fabrication, of assembly, mechanical relevance, etc.) no solution is best, one can only say that some designs are better than others. More formally, as for any structural object, the quality of an interlocking assembly strongly depends on the interaction of the five axes of design proposed by [9], namely form, force, structure, material, and technology. This conceptual approach is exemplified on the work of Larsson *et al.*, [53], on the inset.

On this example, while delivering stunning results, the authors had to assume an assembly (structure) made of wood (material), carrying most probably bending moments (force), milled with a 3-axis CNC machine (technology), with a grid of voxel as a design space (form), as well as additional assumptions such as a cube as a design domain and a single axis of assembly. Any change in those premises, for instance switching to a 5-axis CNC machine, greatly impacts the space of solutions and requires another algorithm to search it. More generally a good approach to designing assemblies, shown in FIGURE 2.17, would be through a multi-criteria optimisation where several designs are proposed to the designer who makes the final choice as to orient her work. These criteria (choice of material, technology, etc) are problem-dependent and could therefore be implemented *a posteriori* to curate the space of solutions, once the user knows how to navigate and explore the space of possible assemblies. As an example, for timber assemblies, the milling technology (3 to 5 axes, size of milling tools) and mechanical performance (governed by the strong anisotropy of wood) are obvious practical constraints that will dictate the performance of the assembly and, thus, the subset of suitable assembly shapes. Other technologies and material, like 3d-printed steel nodes [68] FIGURE 2.16, would come with different sets of feasibility constraints which would be met by different geometries of assembly.

2.4.1 PROBLEM STATEMENT

Our aim is to build design tools that fit in with the design for (dis)assembly movement. We want to explore the feasible design space rather than to optimise a single solution (FIGURE 2.17), and generate novel assemblies with the minimum of user inputs, and avoid a local optimum. To that end randomness plays a central

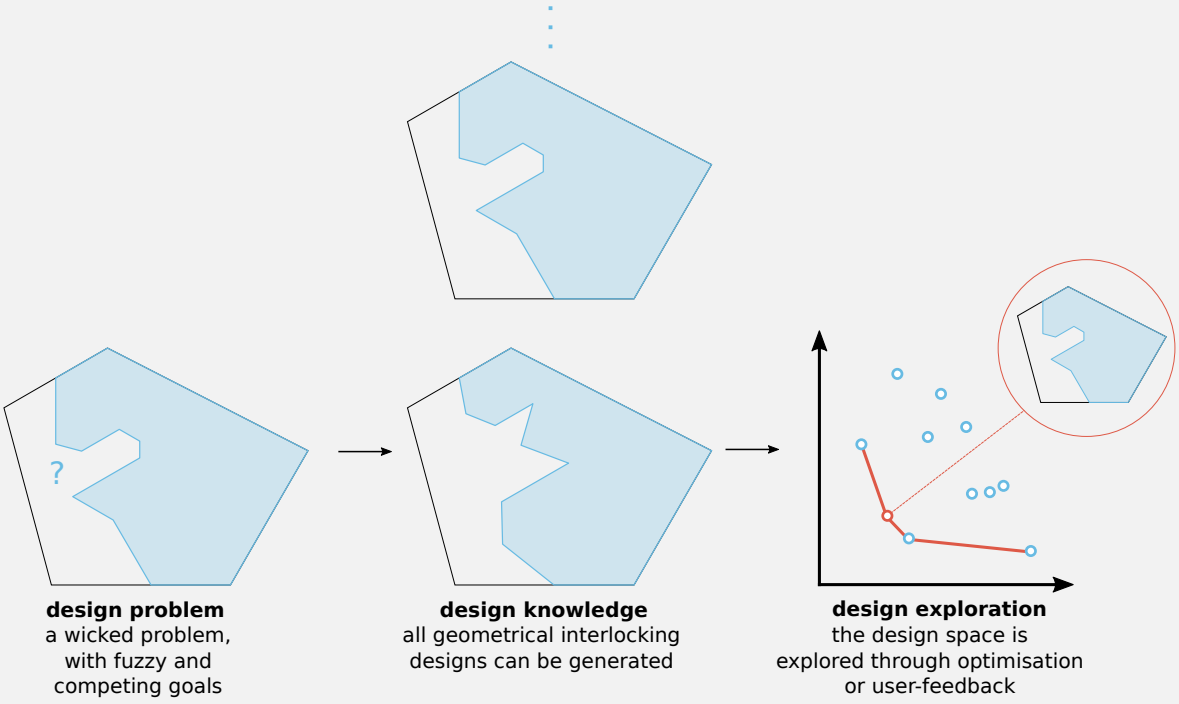


Figure 2.17| This dissertation mainly focuses on the *design knowledge* step of the process to design interlocking assemblies: we shall generate all possible geometrical interlocking design.

role in our approach.

Given a design domain and an ordered list of N motions for disassembling (translations and/or rotations in 2D, generalised motions in 3D), the aim of this work is to partition the domain into $N+1$ parts forming a sequential assembly $A = \{P_0, P_1, ..., P_N\}$. Two parts play a special role: part P_0 is the reference part: it is held in place throughout the (dis)assembly sequence to prevent any ambiguity where the whole assembly may obey a rigid body motion; part P_1 is the *key* - as long as P_1 has not been removed from the assembly, no other part can move - and as such the assembly is interlocked. While the reference part P_0 is fixed, each $P_i, i \geq 1$, can only move along its prescribed motion in an infinitesimal sense, meaning that a motion of arbitrarily small magnitude does not lead a part to collide with any other. The puzzle created should not be recursively interlocking: P_1 shall be a key such as when not moved the entire assembly is blocked, but as soon as P_1 is removed, P_2, P_3, ..., P_N can be removed as well. The only criterion that will be imposed is that the disassembling sequence "*remove* P_1, *then* P_2, *..., then* P_N" always exists (as illustrated on FIGURE 2.18), possibly among others.

A word of caution: the fact that the assemblies created and studied in this manuscript have parts that can only obey an infinitesimal motion may lead to degenerate cases where a part does obey an infinitesimal motion but will collide with another part for a motion of finite magnitude, and as such making the assembly deadlocked[4] for any practical purpose, see FIGURE 2.19.

[4] Impossible to (dis)assemble

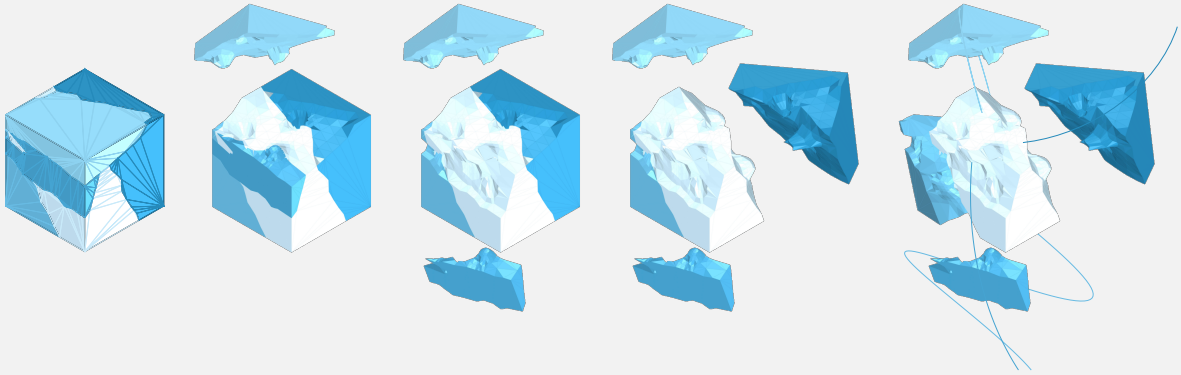


Figure 2.18| A 3D assembly is disassembled sequentially. The whitest part in the middle is the reference part, P_0, and the part in the lightest shade of blue, at the top of each frame, is the key P_1.

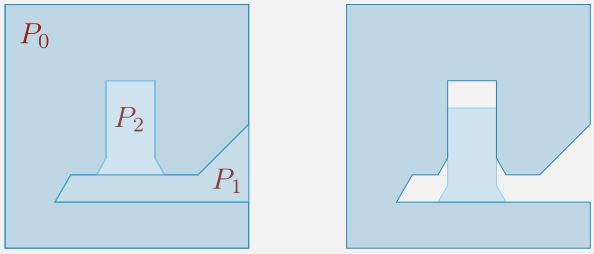


Figure 2.19| Even though P_2 obeys an infinitesimal downwards translation, a finite motion will lead to a collision with P_0.

A clarification is needed concerning the definition of "interlocking". Some authors, [108], say that a structure made of multiple parts is interlocked when, assuming that one part is fixed in place (to prevent global rigid body motion of the whole structure), it is in equilibrium under any arbitrary set of external forces (i.e. the interaction forces between the parts match the external forces applied on the assembly). In other words, by applying any (possibly different) motion(s) to any subassembly, nothing moves. This is typically the case for an Abeille's flat vault, FIGURE 2.11c.

On the contrary, an assembly is not interlocked when by applying a motion along one or many directions of translation to a subassembly some parts move, with an example given on FIGURE 2.20.

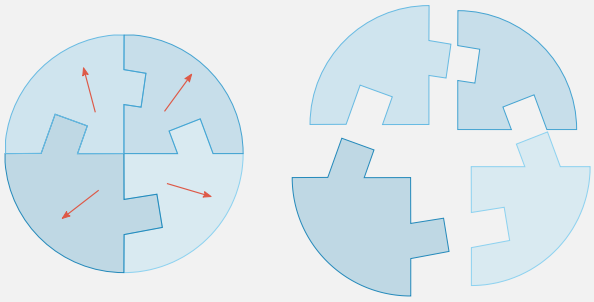


Figure 2.20| An example of a non interlocked assembly: by applying simultaneously a motion along the directions depicted by red arrows, the puzzle can be disassembled. Inspired by Julien Glath's work, [38].

That being said, in this manuscript an assembly is said to be interlocked when, given a fixed key, every part or set of parts is immobilised for all possible motions applied on each part, once a a time. Once the key is removed, the other parts may be taken off one at a time, such as illustrated on FIGURE 2.18. More details

are given SECTION 3.1.3.

2.4.2 CONTRIBUTIONS

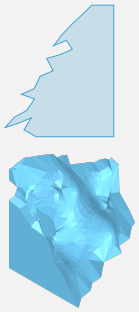
The aim of this dissertation is to contribute to the research effort on the field of generating assemblies mainly along the following points:

- With the notable, recent, exception of [3] which has the drawback of specifying which parts are in contact, the literature so far only considers either catalogue-based joints, which do not explore at all the space of possible joins, or voxel-based, which explores only a very limited subset of this space. The first aim of this document is to fully explore the space of interlocking assemblies.
- To the best of our knowledge, no research has ever been conducted on the generation of assemblies whose parts are to be (dis)assembled along rotational motions in 2D, or generalised 3D motions. The second aim of this document is therefore to unveil this blind spot.
- While actual robotic manipulation was not a part of this PhD, extensive statistical studies of the role played by imperfections (on the geometry of the assembly and/or on the location of the actuator manipulating it) were conducted as to give guidelines and heuristics to better optimise assemblies and to control the amount of freedom given to the operator (human or robot) tasked with (dis)assembling them. To our knowledge, it is the first time that this type of robust geometrical optimisation has been conducted on assemblies.

2.4.3 DEFINITIONS

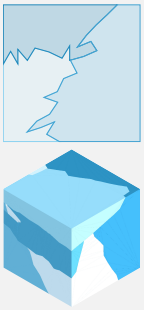
Using the terminology presented in [107], we restrict our study to interlocking sequential assemblies made of rigid polygonal parts in two dimensions (2D), and polyhedral parts in three dimensions (3D), obeying infinitesimal motions. Several terms need to be explained and clarified using [41]:

- **Workspace**: In this study, the workspace refers to the physical space in which the assembly is. It is modelled either by $\mathbb{R}^2$ or $\mathbb{R}^3$.
- **Design domain:** The design domain is a rigid body, modelled by a finite compact manifold, which is partitioned into parts forming an assembly. Throughout this manuscript, unless specified otherwise, the design domain will be a square or a cube.
- **Part**: A part is a rigid body modelled by a finite compact manifold in the workspace.
- **Polygonal part:** In $\mathbb{R}^2$, the boundary separating a part from the rest of the assembly is a polyline, i.e. a closed curve made of end-to-end concatenated line segments such as illustrated on the inset on the right, top.
- **Polyhedral part:** In $\mathbb{R}^3$, the boundary separating a part from the rest of the workspace is a closed triangular mesh, i.e. a mesh, with no boundary edge, made of triangular faces such as illustrated on the inset on the right, bottom.



- **Separating curve/surface of a part:** The separating curve (in 2D) or surface (in 3D) is the portion of the boundary of the part that separates it from the rest of the assembly. By construction, it is always in the interior of the design domain.
- **Motion:** In this work, a motion refers to any translation, rotation or combination of both used to move a point in the workspace.
- **A part obeys an infinitesimal motion:** Given two parts P_i and P_j in contact (i.e. a subset of each of their boundary overlaps), P_i obeys an infinitesimal motion if, at the end of a trajectory of arbitrarily small length along that motion, no point in P_i belongs to the interior of P_j. Otherwise, P_i does not obey that motion and the two parts collide. By extension, we will also say that an assembly obeys a set of motions if and only if each of its constitutive parts obeys an infinitesimal motion from that set.

- **Assembly:** An assembly refers to the set of parts in contact with each other whose union is the design domain, and the intersection of two distinct parts only yields a subset of the boundary of these parts, see the inset on the right.
- **Interlocking:** An assembly is interlocked if no single part can obey any motion while holding in place a specific part called the *key*.
- **Sequential:** An assembly is sequential if one needs a (non-necessarily unique) ordering of the parts such that each successive part can be disassembled by obeying a motion.



REFERENCES

1 C'est pas sorcier - France 3. *C'est pas sorcier - Le Grand Palais épate la galerie*. Youtube. 2013. URL: https://youtu.be/9huxiit6bME?t=1189.

3 Lior Aharoni, Ido Bachelet, and Josephine Carstensen. "Topology optimization of rigid interlocking assemblies". In: *Computers & Structures* 250 (July 2021), p. 106521.

4 Ergun Akleman et al. "Generalized Abeille Tiles: Topologically Interlocked Space-Filling Shapes Generated Based on Fabric Symmetries". In: *Computers and Graphics* 89 (May 2020).

5 Ergun Akleman et al. "Generalized Abeille tiles: Topologically interlocked space-filling shapes generated based on fabric symmetries". In: *Computers & Graphics* 89 (2020), pp. 156–166. ISSN: 0097-8493. URL: https://www.sciencedirect.com/science/article/pii/S0097849320300674.

6 Aleksandra Apolinarska et al. "Robotic assembly of timber joints using reinforcement learning". In: *Automation in Construction* 125 (May 2021), p. 103569.

9 Marine Bagneris et al. "Structural Morphology Issues in Conceptual Design of Double Curved Systems". In: *International Journal of Space Structures* 23 (June 2008), pp. 79–87.

12 Thomas Bock and Thomas Linner. *Robot-Oriented Design: Design and Management Tools for the Deployment of Automation and Robotics in Construction*. Cambridge University Press, 2015.

17 M. Brocato and L. Mondardini. "A new type of stone dome based on Abeille's bond". In: *International Journal of Solids and Structures* 49.13 (2012), pp. 1786–1801.

18 Edvard P. G. Bruun et al. "Human–robot collaboration: a fabrication framework for the sequential design and construction of unplanned spatial structures". In: *Digital Creativity* 31 (2020), pp. 320–336.

20 Paolo Cignoni et al. "Field-Aligned Mesh Joinery". In: *ACM Trans. Graph.* 33.1 (Feb. 2014). ISSN: 0730-0301. URL: https://doi.org/10.1145/2537852.

26 Lee Djumas et al. "Deformation mechanics of non-planar topologically interlocked assemblies with structural hierarchy and varying geometry". In: *Scientific Reports* 7 (Sept. 2017).

27 Arcady Dyskin et al. "Fracture Resistant Structures Based on Topological Interlocking with Non-planar Contacts". In: *Advanced Engineering Materials* 5 (Mar. 2003), pp. 116–119.

28 Yuri Estrin, Arcady Dyskin, and E. Pasternak. "Topological Interlocking as a Material Design Concept". In: *Materials Science and Engineering: C* 31 (Aug. 2011), pp. 1189–1194.

29 Ursula Frick, Tom Van Mele, and Philippe Block. "Decomposing Three-Dimensional Shapes into Self-supporting, Discrete-Element Assemblies". In: (2015). Ed. by Mette Ramsgaard Thomsen et al., pp. 187–201. URL: http://link.springer.com/10.1007/978-3-319-24208-8_16 (visited on 10/17/2019).

30 Chi-Wing Fu et al. "Computational Interlocking Furniture Assembly". In: *ACM Transactions on Graphics* 34.4 (July 2015). ISSN: 0730-0301.

32 J. Gallon. *Machines et inventions approuvées par l'Académie royale des Sciences*. Vol. 1. *Mémoire concernant la voûte plate inventée par M. Abeille*. Académie royale des Sciences, 1735, pp. 159–162.

33 J. Gallon. *Machines et inventions approuvées par l'Académie royale des Sciences*. Vol. 1. *Voûte plate inventée par le père Sébastien de l'Académie Royale des Sciences*. Académie royale des Sciences, 1735, pp. 163–165.

34 B. Gfeller et al. "Wood bonding by vibrational welding". In: *Journal of Adhesion Science and Technology* 17.11 (2003), pp. 1573–1589. eprint: https://doi.org/10.1163/156856103769207419. URL: https://doi.org/10.1163/156856103769207419.

38 Julien Glath et al. “Thinking and Designing Reversible Structures with Non-sequential Assemblies”. In: Sept. 2022, pp. 249–259. ISBN: 978-3-031-13248-3.

41 D. Halperin, J.-C. Latombe, and R. H. Wilson. “A General Framework for Assembly Planning: The Motion Space Approach”. In: *Algorithmica* 26.3 (Mar. 1, 2000), pp. 577–601. ISSN: 0178-4617, 1432-0541. URL: http://link.springer.com/10.1007/s004539910025 (visited on 11/08/2019).

42 Volker Helm et al. “Additive robotic fabrication of complex timber structures”. In: 2015.

43 Yasua Nakahara Hideo Sato. *The Complete Japanese Joinery*. Hartley and Marks Publishers, 2000.

44 Kristian Hildebrand, Bernd Bickel, and Marc Alexa. “crdbrd: Shape Fabrication by Sliding Planar Slices”. In: *Computer Graphics Forum* 31 (May 2012), pp. 583–592.

48 Pablo Jimenez. “Survey on assembly sequencing: A combinatorial and geometrical perspective”. In: *Journal of Intelligent Manufacturing* 24 (Apr. 2011), pp. 1–16.

51 Anja Kunic et al. “Design and assembly automation of the Robotic Reversible Timber Beam”. In: *Automation in Construction* 123 (2021), p. 103531.

52 A.J.D. Lambert. “Disassembly sequencing: A survey”. In: *International Journal of Production Research* 41 (Nov. 2003), pp. 3721–3759.

53 Maria Larsson et al. “Tsugite: Interactive Design and Fabrication of Wood Joints”. In: UIST ’20 (2020). pp. 317–327.

54 *Le chantier du grand palais - dossier pédagogique du grand palais n°2*. Accessed: 2022-08-03. 2013. URL: https://www.grandpalais.fr/sites/default/files/user_images/30/dossier_pedago_chantier_grand_palais.pdf.

55 Kui-Yip Lo, Chi-Wing Fu, and Hongwei Li. “3D Polyomino Puzzle”. In: *ACM Trans. Graph.* 28.5 (Dec. 2009), pp. 1–8. ISSN: 0730-0301. URL: https://doi.org/10.1145/1618452.1618503.

56 Vianney Loing et al. “Free-form structures from topologically interlocking masonries”. In: *Automation in Construction* 113 (2020), p. 103117. ISSN: 0926-5805. URL: https://www.sciencedirect.com/science/article/pii/S0926580519310957.

57 Linjie Luo et al. “Chopper: Partitioning Models into 3D-Printable Parts”. In: *ACM Transactions on Graphics* 31.6 (Nov. 2012). ISSN: 0730-0301.

61 Romain Mesnil et al. “Form finding of nexorades using the translations method”. In: *Automation in Construction* (2018).

68 *Optimizing structural building elements in metal by using additive manufacturing*. Accessed: 2021-12-07. 2015. URL: https://www.ingentaconnect.com/content/iass/piass/2015/00002015/00000002/art00016.

70 Stefana Parascho et al. “Cooperative Fabrication of Spatial Metal Structures”. In: 2017.

71 Stefana Parascho et al. “Robotic vault: a cooperative robotic assembly method for brick vault construction”. In: 2020.

72 Stefana Parascho et al. “LightVault: A Design and Robotic Fabrication Method for Complex Masonry Structures”. In: 2021.

74 Dr. Mary Neighbour Parent. *JAANUS - Japanese Architecture and Art Net Users System*. Internet. 2003. URL: https://www.aisf.or.jp/~jaanus/.

77 Ercüment Erman Ph.D. “A Survey on Structural Timber Joint Classifications and a Proposal Taxonomy”. In: *Architectural Science Review* 42.3 (1999), pp. 169–180. eprint: https://doi.org/10.1080/00038628.1999.9696874. URL: https://doi.org/10.1080/00038628.1999.9696874.

83 Christopher Robeller and Yves Weinand. “Interlocking Folded Plate – Integral Mechanical Attachment for Structural Wood Panels”. In: *International Journal of Space Structures* 30 (2015), pp. 111–122.

85 Bruce Romney et al. “An Efficient System for Geometric Assembly Sequence Generation and Evaluation”. In: Sept. 1995, pp. 699–712.

87 Yuliy Schwartzburg and Mark Pauly. “Fabrication-aware Design with Intersecting Planar Pieces”. In: *Computer Graphics Forum* 32 (May 2013).

89 Peng Song, Chi-Wing Fu, and Daniel Cohen-Or. “Recursive Interlocking Puzzles”. In: *ACM Transactions on Graphics* 31.6 (Nov. 2012). ISSN: 0730-0301.

90 Peng Song et al. “Printing 3D objects with interlocking parts”. In: *Computer Aided Geometric Design* 35-36 (Mar. 2015), pp. 137–148.

94 Sai Ganesh Subramanian et al. “Delaunay Lofts: A biologically inspired approach for modeling space filling modular structures”. In: *Computers & Graphics* 82 (2019), pp. 73–83. ISSN: 0097-8493. URL: https://www.sciencedirect.com/science/article/pii/S0097849319300822.

98 Willy Tegel et al. “Early Neolithic Water Wells Reveal the World’s Oldest Wood Architecture”. In: *PLoS ONE* 7 (Dec. 2012), e51374.

99 Romain Testuz, Yuliy Schwartzburg, and Mark Pauly. “Automatic Generation of Constructable Brick Sculptures”. In: *Eurographics 2013 - Short Papers*. Ed. by M.- A. Otaduy and O. Sorkine. The Eurographics Association, 2013.

101 Gengo Matsui Torashichi Sumiyoshi. *Wood Joints in Classical Japanese Architecture*. Kajima Institute Publishing, 1991.

105 Ziqi Wang, Peng Song, and Mark Pauly. “DESIA: A General Framework for Designing Interlocking Assemblies”. In: *ACM Transactions on Graphics* 37.6 (Dec. 2018). ISSN: 0730-0301.

106 Ziqi Wang, Peng Song, and Mark Pauly. “MOCCA: Modeling and Optimizing Cone-Joints for Complex Assemblies”. In: *ACM Trans. Graph.* 40.4 (July 2021). ISSN: 0730-0301. URL: https://doi.org/10.1145/3450626.3459680.

107 Ziqi Wang, Peng Song, and Mark Pauly. “State of the Art on Computational Design of Assemblies with Rigid Parts”. In: *Computer Graphics Forum* 40 (May 2021), pp. 633–657.

108 Ziqi Wang et al. “Design and Structural Optimization of Topological Interlocking Assemblies”. In: *ACM Transactions on Graphics* 38.6 (2019), p. 13.

109 Michael Weizmann, Oded Amir, and Jacob Grobman. “Topological interlocking in buildings: A case for the design and construction of floors”. In: *Automation in Construction* 72 (June 2016), pp. 18–25.

111 Emily Whiting, John Ochsendorf, and Frédo Durand. “Procedural Modeling of Structurally-Sound Masonry Buildings”. In: *ACM Trans. Graph.* 28.5 (Dec. 2009), pp. 1–9. ISSN: 0730-0301. URL: https://doi.org/10.1145/1618452.1618458.

113 Randall H. Wilson and Jean-Claude Latombe. “Geometric reasoning about mechanical assembly”. In: *Artificial Intelligence* 71.2 (Dec. 1994), pp. 371–396.

115 Shiqing Xin et al. “Making Burr Puzzles from 3D Models”. In: *ACM Transactions on Graphics* 30.4 (July 2011). ISSN: 0730-0301.

116 Yong-Liang Yang, Jun Wang, and Niloy J. Mitra. “Reforming Shapes for Material-Aware Fabrication”. In: *Proceedings of the Eurographics Symposium on Geometry Processing*. SGP ’15. pp. 53–64. Graz, Austria: Eurographics Association, 2015.

117 Jiaxian Yao et al. “Interactive Design and Stability Analysis of Decorative Joinery for Furniture”. In: *ACM Transactions on Graphics* 36.4 (Mar. 2017). ISSN: 0730-0301.

118 Klaus Zwerger. *Wood and Wood Joints - Building Traditions of Europe, Japan and China*. Basel: Birkhäuser Verlag GmbH, 2011.

CHAPTER 3

NON DIRECTIONAL BLOCKING GRAPH

3.1 BLOCKING RELATIONSHIP IN ASSEMBLIES

For this exposition to be self-contained we recall some elementary facts about blocking relationships in assemblies. Additional details and proofs can be found in [113].
The aim of this chapter is, given a 2D assembly $A = \{P_0, P_1, ..., P_N\}$, to explain how to check whether it is interlocked and which set of motions each part may obey. To that end, SECTION 3.1.1 explores the relationship between a given 2D assembly $A = \{P_0, P_1\}$ and the set of motions P_1 can obey, SECTION 3.1.2 explains how, given an assembly and a motion (translation or rotation), one can assess the blocking relationships between the parts for that motion, and SECTION 3.1.3 investigates how, given an assembly, its interlocking for all directions of motion is determined only by the interlocking of a small discrete set of motions.

3.1.1 CONE OF INFINITESIMAL FREEDOM OF MOTION IN 2D

3.1.1.1 Translation

For the sake of simplicity and illustrative purposes, we begin this study by only considering 2D assemblies obeying translations.
A formal way to represent a direction of translation in 2D is by the mean of a vector $\boldsymbol{x}_t \in \mathbb{R}^2$ (subscript $_t$ for translation). As we consider infinitesimal motions only, what matters to us is the direction and not the magnitude of $\boldsymbol{x}_t$. As such we can, without loss of generality, scale $\boldsymbol{x}_t$ to be a unit vector: $||\boldsymbol{x}_t|| = 1$. Vector $\boldsymbol{x}_t$ can be reinterpreted as being a point in $\mathbb{R}^2$. Because this point is at a distance 1 from the origin, it lies on the unit circle. This being true for any unit $\boldsymbol{x}_t \in \mathbb{R}^2$ we denote the locus of all directions of translation in $\mathbb{R}^2$ by the unit circle $\mathcal{S}^1$. In other words, any point on the 2D unit circle $\mathcal{S}^1$ is a direction of translation. This section aims to understand the relationship between a given 2-parts assembly $A = \{P_0, P_1\}$ and the set of translational motions P_1 can obey.

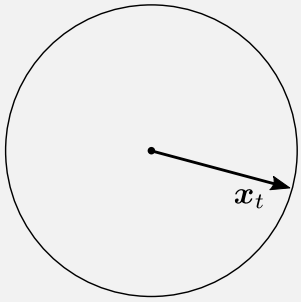


Figure 3.1| The 2D unit circle $\mathcal{S}^1$ is seen as the locus of the directions of translation in $\mathbb{R}^2$.

The simplest polygonal assembly $A = \{P_0, P_1\}$ we can think of is such that the separating curve between P_0 and P_1 is simply a line segment. On FIGURE 3.2, the design domain is the square on the left, the separating curve between the two parts is the line segment highlighted in blue and the green arrows represent several valid directions of translation such that P_1 can obey them. The red ones depict invalid directions of translation as moving P_1 along them will lead the two parts to intersect. The set of all valid translations constitutes a so-called *half-space of motion*. On FIGURE 3.2, let $\boldsymbol{n}$ be the unit normal vector of the separating curve oriented from P_0 to P_1 (the black arrow). Then we can state that P_1 may obey a direction $\boldsymbol{x}_t \in \mathbb{R}^2$ if and only if $\boldsymbol{n} \cdot \boldsymbol{x}_t \geq 0$, that is to say if $\boldsymbol{x}_t$ belongs to the blue semi-circle oriented by $\boldsymbol{n}$, on the right of the figure.

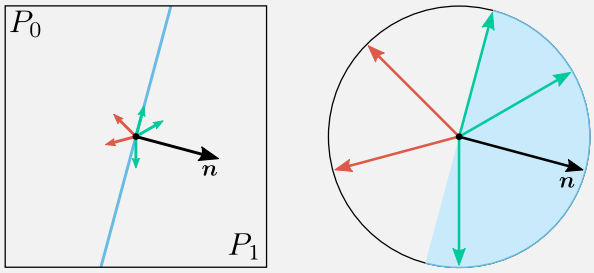


Figure 3.2| On the left an assembly $A = \{P_0, P_1\}$. On the right, the semi unit disc in blue represents the half-space of motion of P_1.

A somewhat more complex result can be obtained by analysing a separating curve made of two line segments. On FIGURE 3.3 each line segment of the separating curve defines a half-space of motion. P_1 can obey any $\boldsymbol{x}_t$ that is in both half-spaces of motion i.e. any $\boldsymbol{x}_t$ such that $\boldsymbol{n}_A \cdot \boldsymbol{x}_t \geq 0$ and $\boldsymbol{n}_B \cdot \boldsymbol{x}_t \geq 0$. We call *cone of translational freedom* the cone resulting from the intersection of the half-spaces of motion defined by the normal of each line segment of the separating curve.

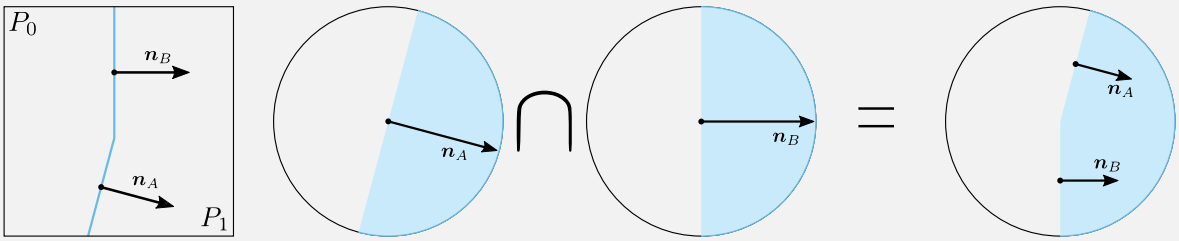


Figure 3.3| On the left an assembly $A = \{P_0, P_1\}$. On the right, the cone of translational freedom results from the intersection of the half-spaces of motion associated with each line segment of the separating curve.

More generally, for a polyline made of k line segments, any vector $\boldsymbol{x}_t$ in the cone of translational freedom is a solution to the linear system

$$\begin{cases} \boldsymbol{n}_1 \cdot \boldsymbol{x}_t \geq 0 \\ \vdots \\ \boldsymbol{n}_k \cdot \boldsymbol{x}_t \geq 0 \end{cases} \tag{3.1}$$

Note that when two half-spaces of motion are antipodal (meaning there exists two normals $\boldsymbol{n}_p$ and $\boldsymbol{n}_q$ negative of each other, $\boldsymbol{n}_p = -\boldsymbol{n}_q$, and $\boldsymbol{n}_q \cdot \boldsymbol{x}_t = 0$), the resulting cone of freedom reduces to a single direction, as depicted on FIGURE 3.4. Observe that given two parts in contact P_i and P_j, the cone of freedom of P_j

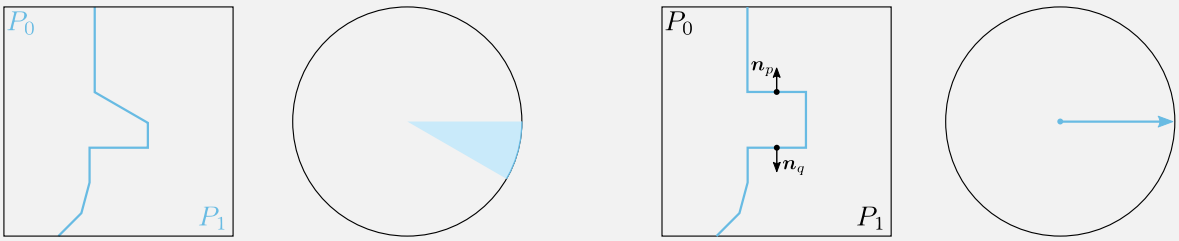


Figure 3.4| By intersecting half-spaces of motion, the cone of translational freedom of P_1 can be defined. Note that this cone may be reduced to a single direction as highlighted in the example on the right.

relative to P_i is exactly the antipodal cone of P_i relative to P_j: indeed if P_i is able to translate along a given $\boldsymbol{x}_t \in \mathbb{R}^2$ while P_j is held in place, then P_j can translate along $-\boldsymbol{x}_t$ while P_i is fixtured, [113]. In that sense, we can focus on only one half of the total set of directions of translation (so only one semicircle of the locus of all directions of translation in 2D) as the other half conveys the same information. That is why only the cone of freedom of one part relative to the other will be represented as the antipodal cone does not add to the sum of information available.

System (3.1) can be rewritten in a matrix form:

$$\boldsymbol{A}_t \boldsymbol{x}_t \geq 0 \tag{3.2}$$

Where the subscript $_t$ stands for translation, and:

$$\boldsymbol{A}_t = \begin{pmatrix} \vdots & \vdots \\ y_i - y_{i+1} & x_{i+1} - x_i \\ \vdots & \vdots \end{pmatrix} \in \mathbb{R}^{k \times 2}$$

Using this formalism, the cone of translational freedom is the set

$$\boxed{\mathcal{C}_t = \{\boldsymbol{x} \in \mathcal{S}^1,\ \boldsymbol{A}_t \boldsymbol{x} \geq 0\}} \tag{3.3}$$

3.1.1.2 Rotation

The mathematical formula that finds, given a 2-parts assembly, the cone of translational freedom of P_1 is quite straightforward to derive. When considering rotational motions, things are almost as forthright, especially thanks to the infinitesimal motion hypothesis.
In a general setting, the point $\hat{\boldsymbol{p}} = (\hat{x}, \hat{y})^T$ obtained by rotating a point $\boldsymbol{p} = (x, y)^T$ by an angle ψ around a centre point $\boldsymbol{x}_r = (x_r, y_r)$ (subscript $_r$ for rotation) is given by:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{pmatrix} \begin{pmatrix} x - x_r \\ y - y_r \end{pmatrix} + \begin{pmatrix} x_r \\ y_r \end{pmatrix}$$

Because we restrict this study to infinitesimal motions, we assume $|\psi| \ll 1$ and a first-order Taylor expansion yields

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \psi \begin{pmatrix} y_r - y \\ x - x_r \end{pmatrix} \tag{3.4}$$

Equation (3.4) states that an infinitesimal rotation of $\boldsymbol{p}$ around $\boldsymbol{x}_r$ is the same as an infinitesimal translation of $\boldsymbol{p}$ along the vector $(y_r - y, x - x_r)^T$.

Let the vector-valued function

$$\begin{array}{rccc} \boldsymbol{m} & : & \mathbb{R}^2 \times \mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ & & \left(\boldsymbol{p} = \begin{pmatrix} x \\ y \end{pmatrix}, \boldsymbol{x}_r = \begin{pmatrix} x_r \\ y_r \end{pmatrix}\right) & \mapsto & \begin{pmatrix} y_r - y \\ x - x_r \end{pmatrix} \end{array} \tag{3.5}$$

We call $\boldsymbol{m}(\boldsymbol{p}, \boldsymbol{x}_r)$ the *instantaneous direction of motion of point $\boldsymbol{p}$ relative to $\boldsymbol{x}_r$*, abbreviated $\boldsymbol{m}_p$ when the location of $\boldsymbol{x}_r$ has been specified and is not ambiguous. Note that this vector is orthogonal to the line going through $\boldsymbol{x}_r$ and $\boldsymbol{p}$. These quantities are illustrated on FIGURE 3.5.

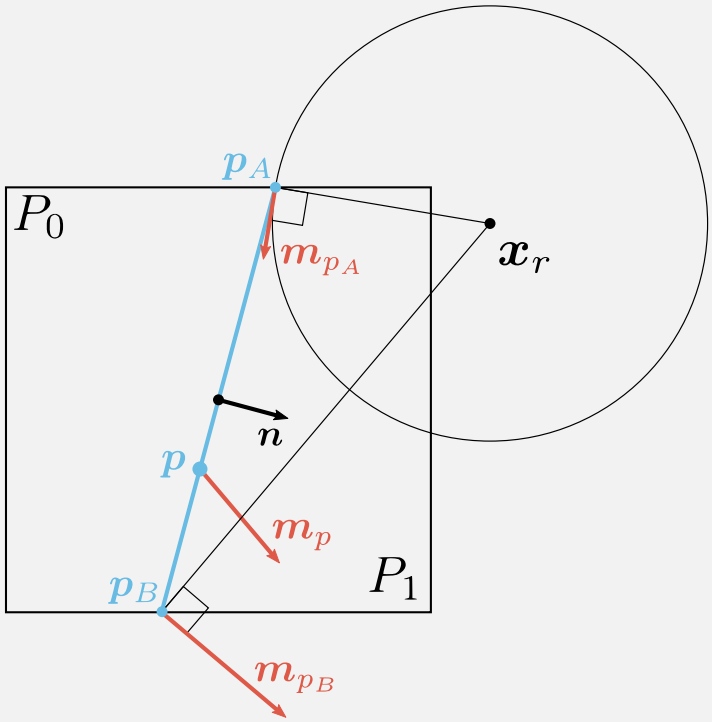


Figure 3.5| The instantaneous direction of motion of point $\boldsymbol{p}_A$ (resp. $\boldsymbol{p}_B, \boldsymbol{p}$) is the vector, $\boldsymbol{m}_{p_A}$ (resp. $\boldsymbol{m}_{p_B}, \boldsymbol{m}_p$).

A necessary and sufficient condition for the part P_1 to obey a (counterclockwise) rotation around a given centre $\boldsymbol{x}_r$ is to have the instantaneous directions of motion of all points $\boldsymbol{p}$ on the boundary of P_1 **not** pointing towards the interior of P_0. Indeed if there is one point $\boldsymbol{p}$ on the boundary of P_1 such that $\boldsymbol{m}_p$ points towards the interior of P_0 then an infinitesimal rotation around $\boldsymbol{x}_r$ will send that point to collide with P_0 which exactly means that P_1 does not obey $\boldsymbol{x}_r$. Moreover, since we focus on infinitesimal motions, we only need to study the points on the boundary of both P_0 and P_1. In other words the fact that P_1 obeys $\boldsymbol{x}_r$ depends only on the geometry of the separating curve between P_0 and P_1.
On the assembly depicted on FIGURE 3.5, the separating curve between the two parts is the line segment $[\boldsymbol{p}_A, \boldsymbol{p}_B]$ of normal $\boldsymbol{n}$. Having the instantaneous direction of motion, $\boldsymbol{m}_p$, of each point $\boldsymbol{p}$ of the line segment

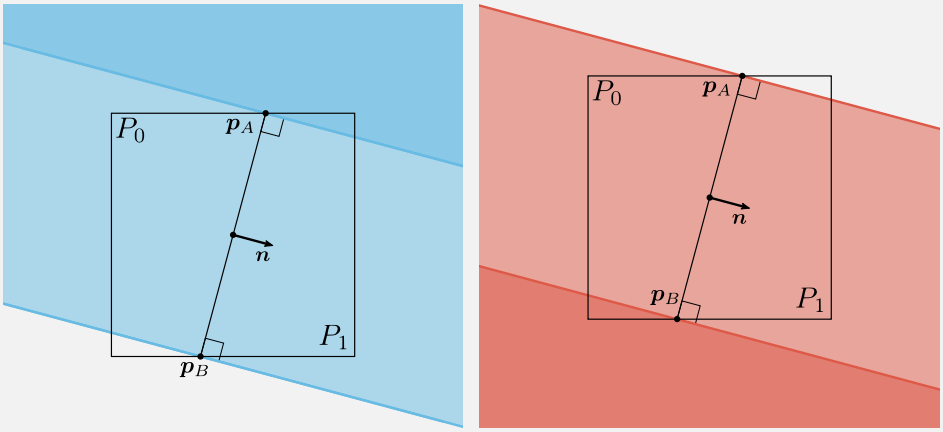


Figure 3.6| P_1 may obey counterclockwise (on the left in blue), (resp. clockwise on the right in red) rotations around centres in the intersection of both half-planes, depicted as the darkest shade of blue (resp. red) in the top right (resp. bottom left).

to be pointing inside P_1 means that $\forall \boldsymbol{p} \in [\boldsymbol{p}_A, \boldsymbol{p}_B]\ \boldsymbol{m}_p \cdot \boldsymbol{n} \geq 0$. Thus, one has the equivalences:

$$P_1 \text{ obeys a rotation around } \boldsymbol{x}_r \iff \forall \boldsymbol{p} \in [\boldsymbol{p}_A, \boldsymbol{p}_B]\ \boldsymbol{m}_p \cdot \boldsymbol{n} \geq 0 \tag{3.6}$$

$$\iff \forall \boldsymbol{p} \in [\boldsymbol{p}_A, \boldsymbol{p}_B]\ \exists t \in [0,1] \begin{cases} \boldsymbol{p} = (1-t)\boldsymbol{p}_A + t\boldsymbol{p}_B \\ \begin{pmatrix} y_r - ((1-t)y_A + ty_B) \\ (1-t)x_A + tx_B - x_r \end{pmatrix} \cdot \boldsymbol{n} \geq 0 \end{cases} \tag{3.7}$$

$$\iff \forall t \in [0,1]\ ((1-t)\boldsymbol{m}_{p_A} + t\boldsymbol{m}_{p_B}) \cdot \boldsymbol{n} \geq 0 \tag{3.8}$$

$$\iff \begin{cases} \boldsymbol{m}_{p_A} \cdot \boldsymbol{n} \geq 0 \\ \boldsymbol{m}_{p_B} \cdot \boldsymbol{n} \geq 0 \end{cases} \tag{3.9}$$

With $\Delta x = x_A - x_B$ and $\Delta y = y_A - y_B$, EQUATION (3.9) can be rewritten as a matrix inequality:

$$\text{EQUATION } (3.9) \iff \begin{pmatrix} \Delta x & \Delta y \\ \Delta x & \Delta y \end{pmatrix} \begin{pmatrix} x_r \\ y_r \end{pmatrix} \geq \begin{pmatrix} x_A \Delta x + y_A \Delta y \\ x_B \Delta x + y_B \Delta y \end{pmatrix} \tag{3.10}$$

EQUATION (3.10) reveals the geometrical meaning of the equations: it states that for P_1 to obey a rotation around $\boldsymbol{x}_r$ then $\boldsymbol{x}_r$ must lie in the intersection of two half-planes whose respective boundaries are lines orthogonal to the segment $[\boldsymbol{p}_A, \boldsymbol{p}_B]$ and going through $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$, such as illustrated on the left of FIGURE 3.6. Note that one can readily find the half-plane containing points around which P_1 obeys a clockwise rotation by inverting the inequality in EQUATION (3.10), as illustrated on the right part of FIGURE 3.6.

To sum up, given the simplest 2D polygonal assembly where the separating curve is the segment $[\boldsymbol{p}_A, \boldsymbol{p}_B]$, the plane $\mathbb{R}^2$ can be partitioned in 3 semi-infinite regions:

- $\left\{ \boldsymbol{x} \in \mathbb{R}^2, \begin{cases} \boldsymbol{m}(\boldsymbol{p}_A, \boldsymbol{x}) \cdot \boldsymbol{n} \geq 0 \\ \boldsymbol{m}(\boldsymbol{p}_B, \boldsymbol{x}) \cdot \boldsymbol{n} \geq 0 \end{cases} \right\}$ and P_1 obeys a counterclockwise rotation around any point in that set. On FIGURE 3.7, this set is the half-plane "above" the blue line.
- $\left\{ \boldsymbol{x} \in \mathbb{R}^2, \begin{cases} \boldsymbol{m}(\boldsymbol{p}_A, \boldsymbol{x}) \cdot \boldsymbol{n} < 0 \\ \boldsymbol{m}(\boldsymbol{p}_B, \boldsymbol{x}) \cdot \boldsymbol{n} \geq 0 \end{cases} \right\}$ and P_1 cannot obey a rotation around any point in that set. On

FIGURE 3.7, this set lies "between" the blue and red lines.

- $\left\{ \boldsymbol{x} \in \mathbb{R}^2, \left\{ \begin{array}{l} \boldsymbol{m}(\boldsymbol{p}_A, \boldsymbol{x}) \cdot \boldsymbol{n} < 0 \\ \boldsymbol{m}(\boldsymbol{p}_B, \boldsymbol{x}) \cdot \boldsymbol{n} < 0 \end{array} \right. \right\}$ and P_1 obeys a clockwise rotation around any point in that set. On FIGURE 3.7, this set is the half-plane "below" the red line.

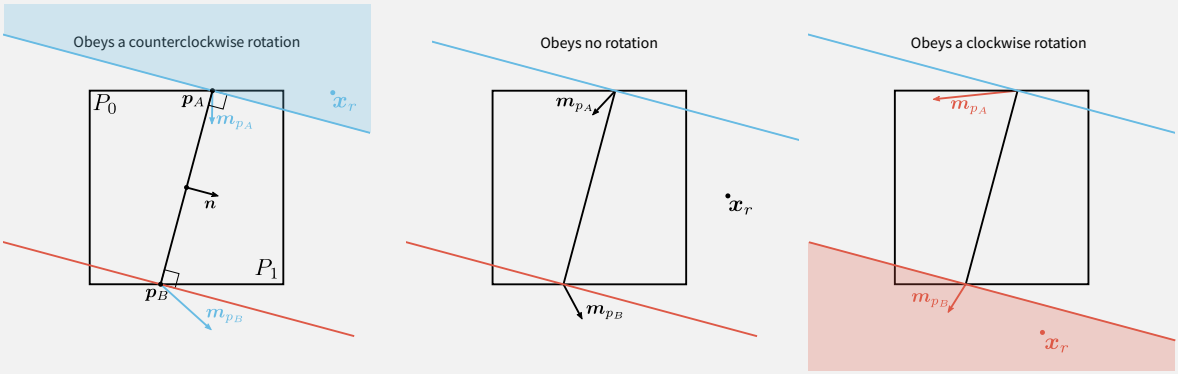


Figure 3.7| Depending on the location of the centre of rotation $\boldsymbol{x}_r$, part P_1 may obey a counterclockwise rotation (left), no rotation (middle), or a clockwise rotation (right).

Stereographic projection

For illustrative purposes, it may be easier not to manipulate these semi-infinite planes but rather their projection on $\mathcal{S}^2$, the unit sphere embedded in $\mathbb{R}^3$. Indeed, as the whole plane is mapped to the sphere, such half planes are mapped to spherical caps, making the illustration much more compact and often more readable. The interested reader is referred to APPENDIX A for a rigorous definition of the stereographic projection used in this manuscript. On FIGURE 3.8, the (inverse) stereographic projection maps the half-planes corresponding to the set of counterclockwise and clockwise centres of rotation to two spherical caps. These caps are termed *caps of counterclockwise (or clockwise) rotational freedom*. The infinite stripe separating the half-planes (in white on the left of FIGURE 3.8) is mapped to the grey area on the sphere. These three regions are tangent at the north pole (the projection point, $(0,0,1)^T$ in the usual cartesian frame): indeed as the north pole on $\mathcal{S}^2$ is mapped to infinity on $\mathbb{R}^2$, and a rotation with a centre at infinity is the same as a translation, the fact that the north pole belongs simultaneously to the three regions (to both caps, as well as to the remaining, grey, region), simply means that, for this assembly, one can find directions of translation that could be obeyed (for instance a translation to the right, on FIGURE 3.8), or not (translation upwards), by P_1.

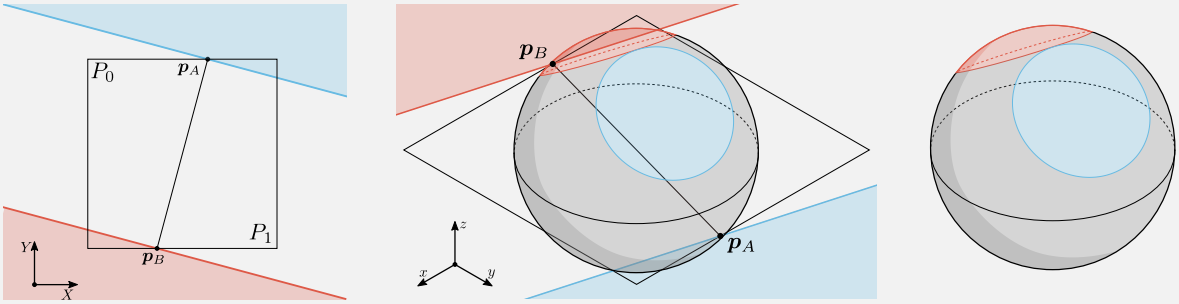


Figure 3.8| The half-planes of possible centres of rotation are projected onto the unit sphere $\mathcal{S}^2$ as spherical caps, the so-called *caps of rotational freedom*, that are tangent at the north pole $(0,0,1)^T$.

In the remainder of this manuscript, when dealing with 2D rotations, $\boldsymbol{x}_r$ will refer indistinctively to a centre of rotation $\boldsymbol{x}_r \in \mathbb{R}^2$ or its projection $\boldsymbol{x}_r \in \mathcal{S}^2$. Any ambiguity can be removed by simply moving back and forth between these two representations.

Let's further our understanding of the set of valid centres of rotation. If the separating curve is made of two line segments between points $\boldsymbol{p}_1, \boldsymbol{p}_2$ and $\boldsymbol{p}_3$, of normal vectors $\boldsymbol{n}_{12}$ and $\boldsymbol{n}_{23}$ oriented from P_0 to P_1, the set of possible centres of rotation is given as the solution to the system of unknown $\boldsymbol{x}_r \in \mathbb{R}^2$:

$$\begin{cases} \boldsymbol{m}(\boldsymbol{p}_1, \boldsymbol{x}_r) \cdot \boldsymbol{n}_{12} \star 0 \\ \boldsymbol{m}(\boldsymbol{p}_2, \boldsymbol{x}_r) \cdot \boldsymbol{n}_{12} \star 0 \\ \boldsymbol{m}(\boldsymbol{p}_2, \boldsymbol{x}_r) \cdot \boldsymbol{n}_{23} \star 0 \\ \boldsymbol{m}(\boldsymbol{p}_3, \boldsymbol{x}_r) \cdot \boldsymbol{n}_{23} \star 0 \end{cases} \tag{3.11}$$

where operator $\star$ stands for $\geq$ to find the set of centres of counterclockwise rotation or $\leq$ to find the set of clockwise centres. Geometrically, this set is found by intersecting the relevant half-planes in $\mathbb{R}^2$, or, equivalently, the relevant caps on $\mathcal{S}^2$, as depicted on FIGURE 3.9. Once projected on the sphere $\mathcal{S}^2$, such a set can be seen as a cone, hence the name *cone of rotational freedom*.

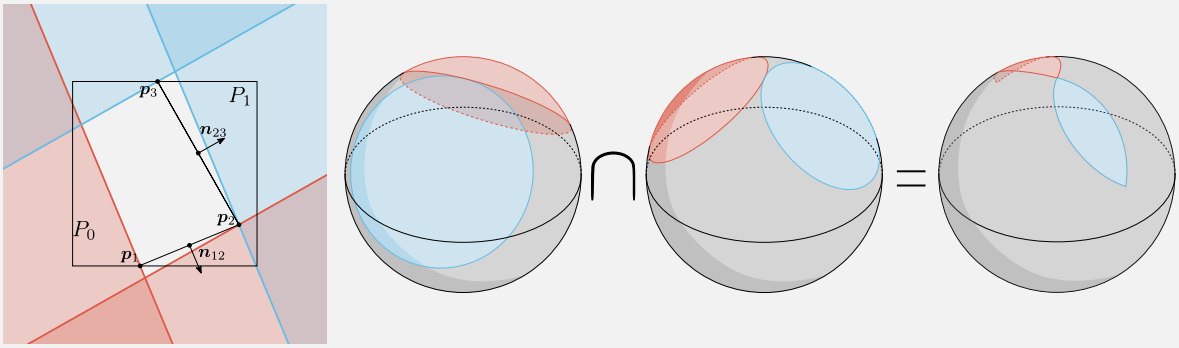


Figure 3.9| Given an assembly of two parts, whose separating curve is made of two line segments, the cones of rotational freedom are obtained by intersecting the caps built with each line segment.

More generally, for a polyline made of k line segments, linking points $p_1, ..., p_{k+1}$, with normal vectors $\boldsymbol{n}_1, ..., \boldsymbol{n}_k$, the cone of rotational freedom is the set of points solution to the system of unknown $\boldsymbol{x} \in \mathbb{R}^2$:

$$\begin{cases} \boldsymbol{m}(\boldsymbol{p}_1, \boldsymbol{x}_r) \cdot \boldsymbol{n}_1 \star 0 \\ \boldsymbol{m}(\boldsymbol{p}_2, \boldsymbol{x}_r) \cdot \boldsymbol{n}_1 \star 0 \\ \boldsymbol{m}(\boldsymbol{p}_2, \boldsymbol{x}_r) \cdot \boldsymbol{n}_2 \star 0 \\ \vdots \\ \boldsymbol{m}(\boldsymbol{p}_k, \boldsymbol{x}_r) \cdot \boldsymbol{n}_k \star 0 \\ \boldsymbol{m}(\boldsymbol{p}_{k+1}, \boldsymbol{x}_r) \cdot \boldsymbol{n}_k \star 0 \end{cases} \iff \forall i \in [\![1, k]\!], \begin{cases} \boldsymbol{m}(\boldsymbol{p}_i, \boldsymbol{x}_r) \cdot \boldsymbol{n}_i \star 0 \\ \boldsymbol{m}(\boldsymbol{p}_{i+1}, \boldsymbol{x}_r) \cdot \boldsymbol{n}_i \star 0 \end{cases} \tag{3.12}$$

For instance, on FIGURE 3.10, given the assembly on the left, there only exists the cone of clockwise rotation: no counterclockwise rotation (and no translation since the north pole does not belong to the set) can be obeyed by P_1. The cone of rotation may also be reduced to a single point, as depicted on FIGURE 3.10, right.

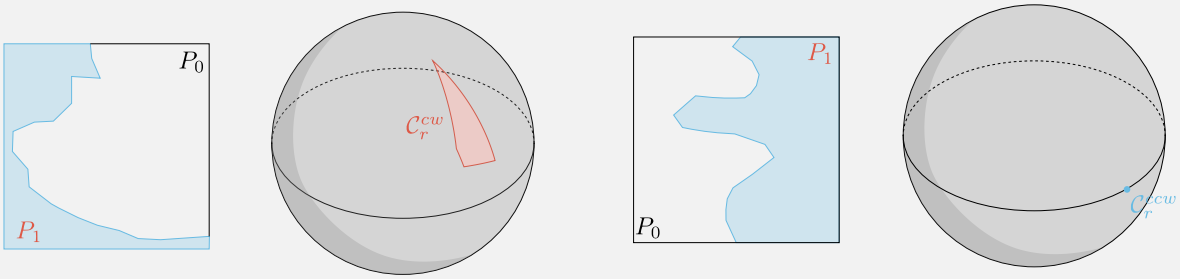


Figure 3.10| P_1 may obey only clockwise rotations around centres belonging to a closed surface on the left; it may only obey a counterclockwise rotation around a unique centre on the right.

SYSTEM (3.12) can be written in matrix form:

$$\boldsymbol{A}_r \boldsymbol{x}_r \star \boldsymbol{b} \tag{3.13}$$

Where subscript $_r$ stands for rotation, and:

$$\boldsymbol{A}_r = \begin{pmatrix} x_2 - x_1 & y_2 - y_1 \\ x_2 - x_1 & y_2 - y_1 \\ x_3 - x_2 & y_3 - y_2 \\ x_3 - x_2 & y_3 - y_2 \\ \vdots & \vdots \\ x_{k+1} - x_k & y_{k+1} - y_k \\ x_{k+1} - x_k & y_{k+1} - y_k \end{pmatrix} \in \mathbb{R}^{2k \times 2} \qquad \boldsymbol{b} = \begin{pmatrix} (x_2 - x_1)x_1 + (y_2 - y_1)y_1 \\ (x_2 - x_1)x_2 + (y_2 - y_1)y_2 \\ (x_3 - x_2)x_2 + (y_3 - y_2)y_2 \\ (x_3 - x_2)x_3 + (y_3 - y_2)y_3 \\ \vdots \\ (x_{k+1} - x_k)x_k + (y_{k+1} - y_k)y_k \\ (x_{k+1} - x_k)x_{k+1} + (y_{k+1} - y_k)y_{k+1} \end{pmatrix} \in \mathbb{R}^{2k}$$

With this formalism, the cones of rotational freedom (be it counterclockwise or clockwise) are the sets:

$$\mathcal{C}_r^{ccw} = \{\boldsymbol{x} \in \mathbb{R}^2,\ \boldsymbol{A}_r \boldsymbol{x} \geq \boldsymbol{b}\} \tag{3.14a}$$

$$\mathcal{C}_r^{cw} = \{\boldsymbol{x} \in \mathbb{R}^2,\ \boldsymbol{A}_r \boldsymbol{x} \leq \boldsymbol{b}\} \tag{3.14b}$$

Where superscript ^{ccw} (resp. ^{cw}) stands for counterclockwise (resp. clockwise).

3.1.1.3 Translation and rotation

As hinted on FIGURE 3.9, when a cone of freedom extends to the north pole of $\mathcal{S}^2$ (the projection point), then the part obeys a translation (as a rotation around a centre at infinity is the same as a translation). Given a polyline, one only needs to build the matrices $\boldsymbol{A_t}$, $\boldsymbol{A}_r$ and vector $\boldsymbol{b}$ and compute the sets $\mathcal{C}_t$, $\mathcal{C}_r^{ccw}$ and $\mathcal{C}_r^{cw}$ given by SYSTEMS (3.3) and (3.14), see FIGURE 3.11.

Note that, quite surprisingly, it is possible to find assemblies obeying rotations but whose translation cone is reduced to a single direction, as shown in FIGURE 3.12. In this case, the centres of rotations are aligned.

3.1.2 DIRECTIONAL BLOCKING GRAPH - DBG

Should the reader be unfamiliar with graph theory, and particularly with the notion of strong connectedness, a quick introduction to this area of mathematics is given in APPENDIX B.

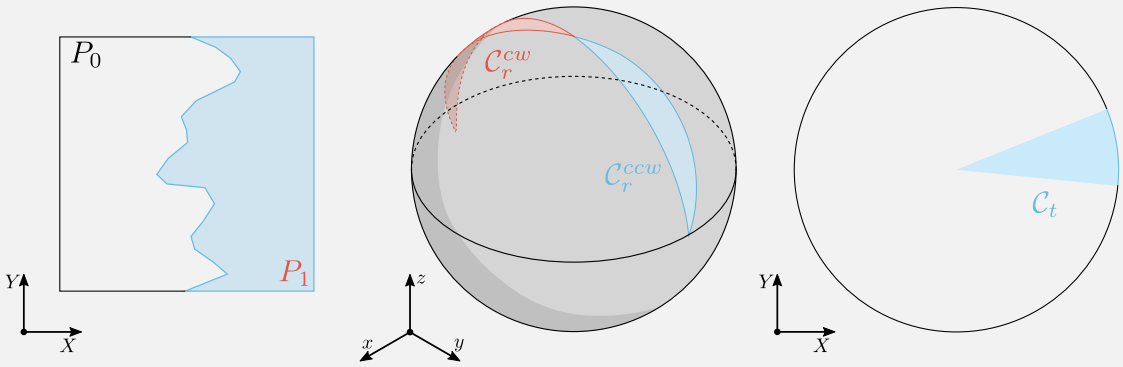


Figure 3.11| From left to right: an assembly made of two parts, the cones of rotational freedom (counterclockwise in blue, clockwise in red) and the cone of translational freedom.

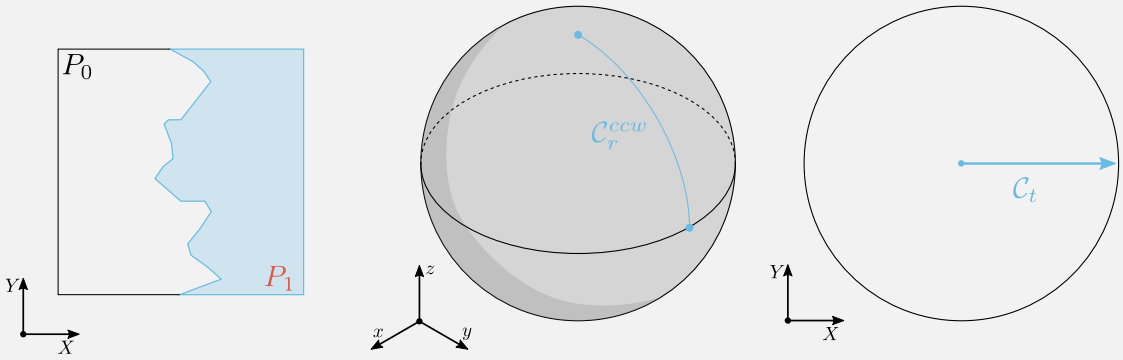


Figure 3.12| The (counterclockwise) rotational cone is reduced to an arc, implying that the translational cone is reduced to a unique direction.

3.1.2.1 Definition

A **Directional Blocking Graph** (DBG), defined by Wilson and Latombe in [113], of an assembly A for a motion $\boldsymbol{x}$ is a directed graph $G(\boldsymbol{x}, A)$ whose vertices P_i correspond to each of the $N+1$ parts in the assembly A and an arc from vertex P_i to vertex P_j means that part P_i *is blocked by* part P_j in A for an infinitesimal motion of part P_i along motion $\boldsymbol{x}$ while holding part P_j in place.

For instance, referring to FIGURE 3.13, the DBGs $G(\boldsymbol{x}, A)$ for different motions $\boldsymbol{x}$ are given. Let us first take a look at the behaviour in translation of the assembly (the rightmost part of the figure). The cone of translational freedom $\mathcal{C}_t$ is depicted in blue, and for any $\boldsymbol{x} \in \mathcal{C}_t$ (on FIGURE 3.13 one such $\boldsymbol{x}$, in blue, is taken in the interior of the cone and another one on its extremity), then by definition of $\mathcal{C}_t$, P_1 obeys this $\boldsymbol{x}$, that is to say that P_1 is not blocked by P_0. On the other hand, P_0 is blocked by P_1 along such translation. Thus, the DBG $G(\boldsymbol{x}, A)$ contains only one arc, $e_{0\to1}$ (from P_0 to P_1), stating that P_0 is blocked by P_1 for a translation along $\boldsymbol{x}$. Conversely, as the arc $e_{1\to0}$ (from P_1 to P_0) does not exist, the graph encodes the fact that P_1 obeys $\boldsymbol{x}$. If $\boldsymbol{x}$ is taken outside of the cone of translational freedom of P_1 (but not inside the antipodal cone), as figured by the two $\boldsymbol{x}$ in black, then neither P_1 nor P_0 obey $\boldsymbol{x}$. This blocking relationship is encoded in the relevant $G(\boldsymbol{x}, A)$ by two arcs, from P_1 to P_0 and *vice-versa*. Finally, because the relation P_i *is locally free to translate along* $\boldsymbol{x}$ *relative to* P_j implies that P_j *is locally free to translate along the opposite direction* $-\boldsymbol{x}$ *relative to* P_i the DBG $G(-\boldsymbol{x}, A)$ can be deduced from $G(\boldsymbol{x}, A)$ by simply reversing the orientation of the arcs in the latter graph, [113].

The same principles apply to build a DBG in rotation: on FIGURE 3.13, middle, for any $\boldsymbol{x} \in (\mathcal{C}_r^{ccw} \cup \mathcal{C}_r^{cw})$ (shown in blue and red), the DBG $G(\boldsymbol{x}, A)$ contains only one arc, $e_{0\to1}$. For any $\boldsymbol{x} \in \mathcal{S}^2 \setminus (\mathcal{C}_r^{ccw} \cup \mathcal{C}_r^{cw})$

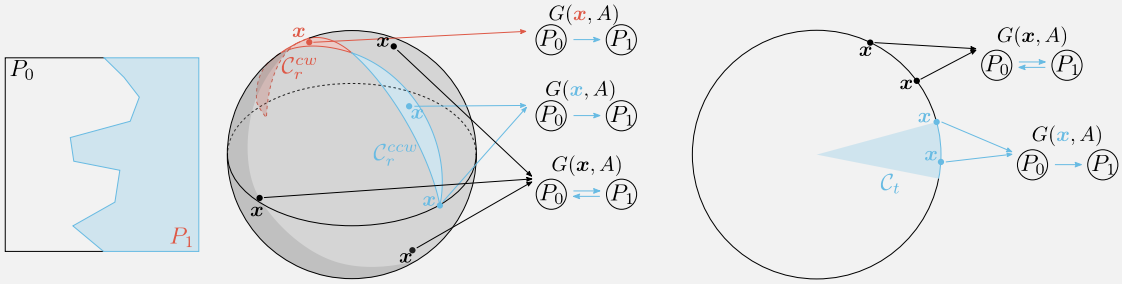


Figure 3.13| An assembly made of two parts P_0 and P_1 and the DBGs associated with different centres of rotation (middle) or directions of translation (right).

(shown in black) neither P_0 nor P_1 can obey a rotation around $\boldsymbol{x}$ and thus both arcs $e_{1\to0}$ and $e_{0\to1}$ exist in $G(\boldsymbol{x}, A)$.

A useful property of a DBG $G(\boldsymbol{x}, A)$ is that we can easily deduce whether the assembly A is interlocked for $\boldsymbol{x}$ by checking the strong-connectedness[1] of the graph: if the graph is strongly connected then every part is blocked by another. If not, then it can be decomposed into strongly connected components (possibly reduced to a single vertex) where at least one component has no outgoing arc and, as such, the corresponding subassembly obeys $\boldsymbol{x}$. For instance, on FIGURE 3.14 (where motions are restricted to translation, for illustrative simplicity), the cones of translational freedom $\mathcal{C}_{t,i}$ of part P_i, $i \in \{1, 2\}$, are shown on the left; both are reduced to single directions, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Top row: the DBG associated with the horizontal direction of translation, $G(\boldsymbol{x}_1, A)$ has two strongly connected components that are colour-coded: $\{P_1\}$ in red and $\{P_0, P_2\}$ in blue. Indeed, starting from P_1 one cannot reach any other node but P_1 while following the arcs' orientation and, similarly, if one starts from any node in $\{P_0, P_2\}$, one can only go back and forth between these two nodes. As no edge starts from P_1, it is not blocked by any other node and hence it obeys $\boldsymbol{x}_1$. On the top row still, DBG $G(\boldsymbol{x}_2, A)$ is strongly connected: its strongly connected component is $\{P_0, P_1, P_2\}$ as one can convince oneself by walking from any node to any other. Thus every part is blocked by another and the assembly is deadlocked for $\boldsymbol{x}_2$ (an upwards motion of translation). In short, these two DBGs say that A may be disassembled by moving P_1 along $\boldsymbol{x}_1$, but it is impossible to disassemble A by moving any part P_i along $\boldsymbol{x}_2$.

On the bottom row of FIGURE 3.14 part P_1 has been removed and the assembly is now made of two parts $\{P_0, P_2\}$. $G(\boldsymbol{x}_1, A)$ is strongly connected, meaning that one cannot move any part along $\boldsymbol{x}_1$, and the one associated to the upwards translation, $G(\boldsymbol{x}_2, A)$ has two strongly connected components, namely $\{P_0\}$ (blue) and $\{P_2\}$ (red). Moreover, since no edge starts from P_2, the latter is not blocked by P_0 for that upwards direction $\boldsymbol{x}_2$: it can obey such $\boldsymbol{x}_2$ and as such A can be disassembled.

Note that identifying the strongly connected components of a graph can be solved by polynomial algorithms, [97], and thus, from a computational point of view, it is a cheap operation.

3.1.2.2 Computation

The computation of a DBG for a given motion $\boldsymbol{x}$ is straightforward. Let $A = \{P_0, ..., P_N\}$ an assembly made of N+1 polygonal parts. To build the DBG $G(\boldsymbol{x}, A)$, the parts are looped upon, and the DBG is built incrementally, starting from a graph with no edge, by considering every pair of parts in contact.

Let P_i and P_j, $j < i$, be two parts in contact and let $\mathbb{S}$ denote the set of line segments shared by both parts. Assume the normal vectors $\boldsymbol{n}_k$ of the segments $s_k \in \mathbb{S}$ to be pointing towards P_i.

[1] Informally speaking, a directed graph is **strongly connected** if one can walk along a path respecting the orientation of the edges between any couple of vertices (P_i, P_j). See APPENDIX B for a rigorous definition.

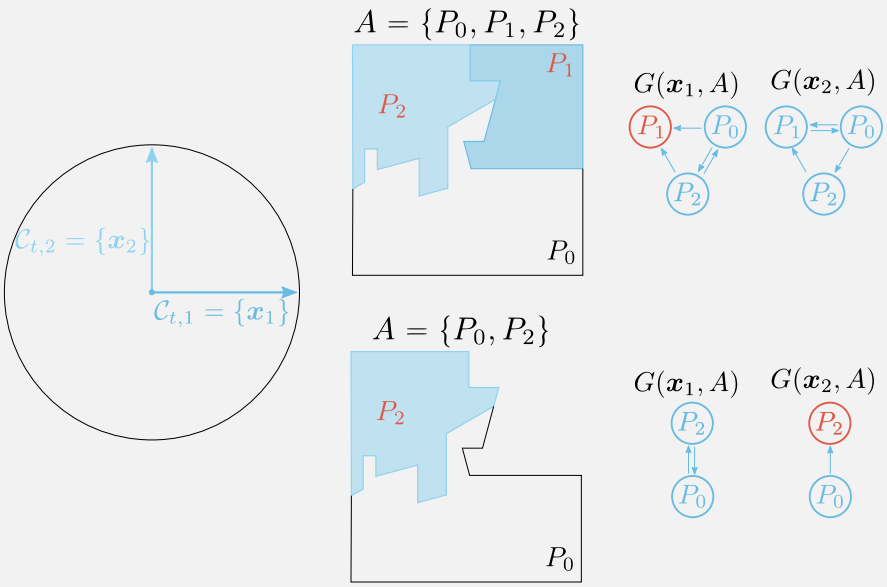


Figure 3.14| The strongly connected components of each DBG are colour-coded (blue and red). Top row: an assembly made of 3 parts and the DBGs associated with the horizontal to the right and vertical upwards directions of translation. Bottom row: assembly obtained after removing P_1 and the corresponding DBGs.

Translation

If the motion considered in the DBG is a translation along a given vector $\boldsymbol{x}_t \in \mathcal{S}^1$, then P_i is blocked by P_j if at least one segment in $\mathbb{S}$ has its normal vector pointing away from $\boldsymbol{x}_t$:

$$\exists s_k \in \mathbb{S},\ \boldsymbol{n}_k \cdot \boldsymbol{x} < 0 \iff P_i \text{ is blocked by } P_j$$

And edge $e_{i\to j}$ is added to the graph $G(\boldsymbol{x}_t, A)$. Conversely

$$\exists s_k \in \mathbb{S},\ \boldsymbol{n}_k \cdot \boldsymbol{x}_t > 0 \iff P_j \text{ is blocked by } P_i$$

And edge $e_{j\to i}$ is added to the graph $G(\boldsymbol{x}_t, A)$.

Rotation

The motion is a counterclockwise rotation around a given centre $\boldsymbol{x}_r \in \mathbb{R}^2$. For each segment $s_k = [\boldsymbol{p}_k, \boldsymbol{p}_{k+1}] \in \mathbb{S}$, the instantaneous directions of motion of the endpoints, $\boldsymbol{m}_{p_k}$ and $\boldsymbol{m}_{p_{k+1}}$, are calculated. Part P_i is blocked by P_j if the instantaneous direction of motion of a point on the boundary is pointing inside P_j:

$$\exists s_k \in \mathbb{S},\ \boldsymbol{m}_{p_k} \cdot \boldsymbol{n}_k < 0 \text{ or } \boldsymbol{m}_{p_{k+1}} \cdot \boldsymbol{n}_k < 0 \iff P_i \text{ is blocked by } P_j$$

And edge $e_{i\to j}$ is added to the graph $G(\boldsymbol{x}, A)$. Conversely

$$\exists s_k \in \mathbb{S},\ \boldsymbol{m}_{p_k} \cdot \boldsymbol{n}_k > 0 \text{ or } \boldsymbol{m}_{p_{k+1}} \cdot \boldsymbol{n}_k > 0 \iff P_j \text{ is blocked by } P_i$$

And edge $e_{j\to i}$ is added to the graph $G(\boldsymbol{x}, A)$. Had we considered a clockwise rotation, the role played by $<$ and $>$ in the two equivalences above would have been switched.

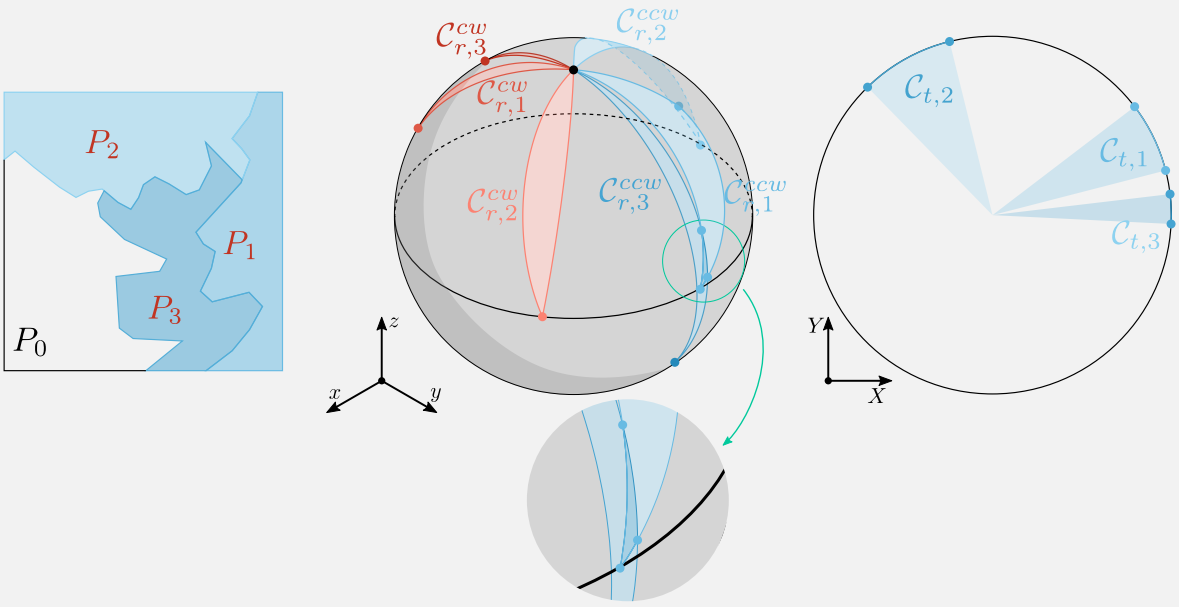


Figure 3.15| $\mathcal{S}^1$ and $\mathcal{S}^2$ are partitioned into cells of dimension 0, 1, and (in the case of $\mathcal{S}^2$ only) 2.

3.1.3 NON DIRECTIONAL BLOCKING GRAPH - NDBG

Following the definition of a DBG, Wilson and Latombe introduced in [113] the concept of **Non Directional Blocking Graph** (NDBG). Simply put, the NDBG of an assembly A is simply the concatenation of all DBGs $G(\boldsymbol{x}, A)$ for any motion $\boldsymbol{x}$ (be it a rotation or a translation). In concrete terms, given an assembly $A = \{P_0, ..., P_N\}$, for any pair of parts (P_i, P_j) in contact, System (3.1) and System (3.9) are solved to get the cones $\mathcal{C}_t, \mathcal{C}_r^{ccw}$ and $\mathcal{C}_r^{cw}$. The set of cones $\mathcal{C}_t$ partitions the unit circle $\mathcal{S}_1$ into an arrangement of cells: cells of dimension 0 (the endpoints of the cone) and of dimension 1 (the open arc of circle between two consecutive endpoints). Similarly, the set of cones of rotational freedom partition $\mathcal{S}^2$ into cells of dimension 0 (endpoints of the cone), 1 (open arc of circle between two consecutive endpoints) or 2 (open surface between arcs). Such cells are illustrated on FIGURE 3.15: each part P_i, $i \in \{1, 2, 3\}$, defines cones $\mathcal{C}_{t,i}$, $\mathcal{C}_{r,i}^{ccw}$ and $\mathcal{C}_{r,i}^{cw}$ that partition $\mathcal{S}^1$ or $\mathcal{S}^2$ into cells of dimension 0, represented by dots of colours, dimension 1, depicted by the arcs between two dots, and in the case of the rotational cones on $\mathcal{S}^2$, cells of dimension 2 (patches of surface between arcs). Note that the intersections of cones $\mathcal{C}_{r,1}^{ccw}$ and $\mathcal{C}_{r,3}^{ccw}$ defines cells of dimension 0, 1 and 2, as exemplified in the zoomed-in portion of the figure.

The DBGs over a given cell are **regular** meaning that the directed graphs remain constant when $\boldsymbol{x}$ varies over it, [113]. Hence one can associate each cell (be it of dimension 0, 1 or 2) with a unique DBG. The NDBG of an assembly A can therefore be defined as the locus of motions ($\mathcal{S}^1$ in translation, $\mathcal{S}^2$ seen as the inverse stereographic projection of $\mathbb{R}^2$ in the case of rotation) partitioned into cells along with the DBGs associated to each cell. Wilson and Latombe in [113] also show that for a given cell, a subassembly S of A is locally free to translate/rotate if and only if there is no arc connecting S to $A \setminus S$ in $G(\boldsymbol{x}, A)$ for a $\boldsymbol{x}$ taken in that cell. Worded differently, if $G(\boldsymbol{x}, A)$ is strongly connected, then no subassembly can move along $\boldsymbol{x}$; if there is at least one strongly connected component (that is not the full graph) in $G(\boldsymbol{x}, A)$ without any outgoing arc then the corresponding subassembly is free to translate along $\boldsymbol{x}$. What makes NDBG a powerful tool to compute the relative motions of parts is that one needs only to look for strong connectedness in its constitutive DBGs to find the interlocking state of an assembly.

Also, in [112] Wilson and Matsui state the following property:

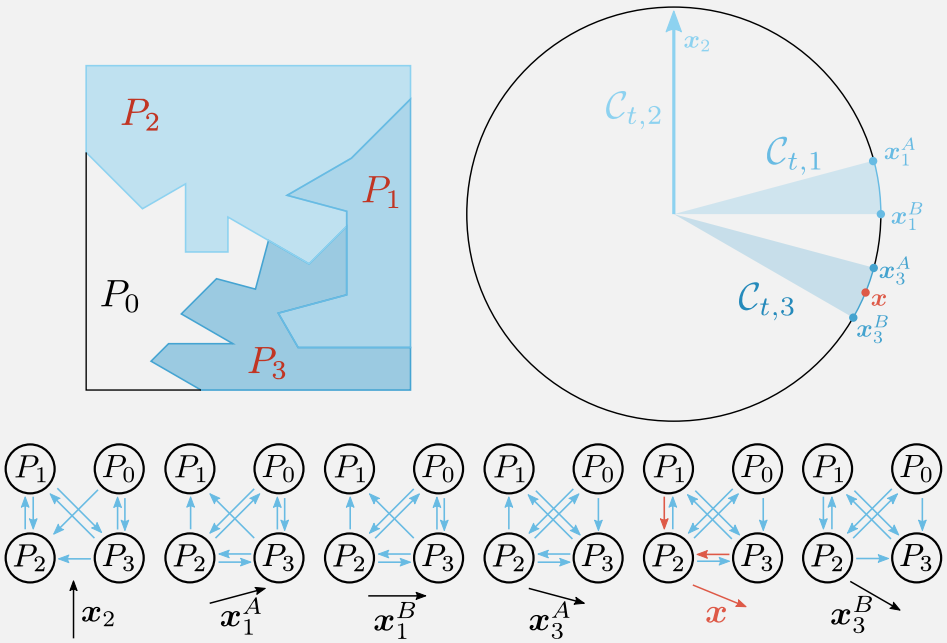


Figure 3.16| The DBG $G(\boldsymbol{x}, A)$ is obtained by performing a union operation of the DBGs $G(\boldsymbol{x}_3^A, A)$ and $G(\boldsymbol{x}_3^B, A)$.

Property:
For any two cells f_1 and f_2 such as f_1 is on the boundary of f_2, if there exists an arc from P_i to P_j in $G(f_1)$, this arc also exists in $G(f_2)$.

Where $G(f)$ denotes the DBG of cell f. What this property means is that if there is an arc in the DBG of a cell of a given dimension, then this arc also exists in the DBG of neighbouring cells of higher dimensions. This property is illustrated on FIGURE 3.16: $\boldsymbol{x}$, in red, represents a direction of translation taken in the open cone $\mathring{\mathcal{C}}_{t,3}$ and one can notice that $\boldsymbol{G}(x, A)$ is the union of $G(\boldsymbol{x}_3^A, A)$ and $G(\boldsymbol{x}_3^B, A)$, the DBGs of the endpoints of the cone; arc $e_{3\to 2}$ is taken from $G(\boldsymbol{x}_3^A, A)$ and arc $e_{1\to 2}$ from $G(\boldsymbol{x}_3^B, A)$ (highlighted in red on FIGURE 3.16), while all other arcs are present in both DBGs.

This property proves to be extremely useful: as we have already understood, the DBGs associated with each cell state whether a part may obey any motion in that cell. Yet, as the DBG of a cell of higher dimension can be deduced from the DBGs of neighbouring cells of lower dimensions, one notices that to fully characterise the interlocking state of an assembly, *i.e.* to build the NDBG, it is sufficient to calculate only the DBGs associated to cells of dimension 0, which are in finite (relatively low) number! Following the definition of [105], we call these DBGs the **base DBGs** of the assembly. If all such base DBGs are strongly connected, but the one associated with the motion of the key P_1 which must have 2 strongly connected components (one being reduced to vertex P_1, the other being all other vertices), then, using the property, the DBGs for every other motion are also strongly connected, and thus the assembly is interlocked.

We would like to insist on the definition of "interlock" in this dissertation. By definition, a DBG treats only the motion of one part relative to another one. It does not convey any information on the blocking relationship between a part and other ones when applying a motion to many parts at once. In that sense, a DBG does not carry enough information to say whether an assembly is interlocked. That being said, even though we should say that an assembly A is *weakly interlocked* for a given $\boldsymbol{x}$ when the associated DBG $G(\boldsymbol{x}, A)$ is strongly connected, we will, in the remainder of this manuscript, use in its place the word **interlock**, for simplicity. For instance, regarding translation only, the assembly of FIGURE 3.17 has a DBG (regular over all $\mathcal{S}^1$) whose graph is strongly connected, meaning that by applying a translation $\boldsymbol{x}_t \in \mathcal{S}^1$ to every single part, once at a time, the puzzle cannot be disassembled. Yet, as seen in this figure, it is not interlocked as

simultaneous motions along specific directions allow the disassembling of the assembly.

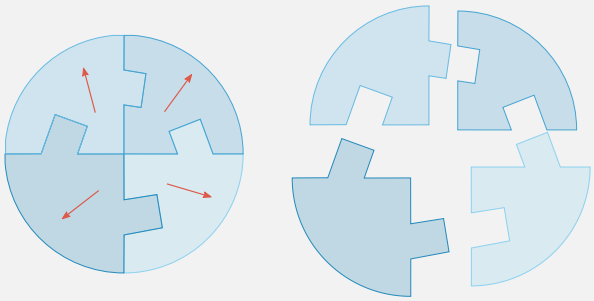


Figure 3.17| An example of a non interlocked assembly: by applying simultaneously a motion along the directions depicted by red arrows, the puzzle can be disassembled. Inspired by Julien Glath's work, [38].

3.2 CONCLUSION

This short chapter aimed at recalling a few basic facts about 2D interlocking assemblies. SECTION 3.1.1 showed that given a separating polyline $(\boldsymbol{p}_i)_{i\in[\![1,k]\!]}$, the **cone of freedom of motion**, i.e the set of motion it can obeys is given in translation by the set $\mathcal{C}_t$ and in rotation by the cones $\mathcal{C}_r^{ccw}$ and $\mathcal{C}_r^{cw}$. These sets are built using inequalities of the form $\boldsymbol{n}_i \cdot \boldsymbol{y} \geq 0$ where $\boldsymbol{y}$ stands for either the instantaneous direction of motion of the end points of the segment i with respect to a centre of rotation, or a direction of translation (which is remarked to be essentially the same thing as an instantaneous direction of motion, thus unifying the concepts in translation and rotation).

Introducing these inequalities helped us understand how the **Directional Blocking Graph** of an assembly is built for a given motion $\boldsymbol{x}$, as seen in SECTION 3.1.2. Once the graph is available, its edges (or lack of) teaches us about the blocking relationships in the assembly along $\boldsymbol{x}$. In particular, we understood that if a DBG $G(\boldsymbol{x}, A)$ is strongly connected, then the assembly is interlocked for motion $\boldsymbol{x}$.

The **Non Directional Blocking Graph** (NDBG), presented in SECTION 3.1.3, is built upon the notion of DBG and fully assess the interlocking of an assembly for all directions of motion. Only a discrete and finite set of base DBGs needs to be calculated for the interlocking to be fully known. Moreover the base DBGs give all the disassembly sequences (and thus, by reversing the order, all the assembly sequences). Indeed at each disassembling step, the strong-connectedness of the DBGs are calculated. When one DBG is not strongly connected, the part(s) associated to one of the strongly connected component can be removed along the motion associated with that DBG. The corresponding vertex (or vertices) and adjacent edges are removed from all other graphs and the next disassembling step may start, and such algorithm goes on until we are left with part P_0.

While the theory behind the NDBG completely works for 3D assemblies, we have not explained how 3D motions are encoded, and thus we cannot explain which set of motion, which cone, a 3D part may obey. Unit dual quaternions will be introduced in SECTION 5.1, and following that the computation of the cone of freedom of motion for a 3D part will be explained in SECTION 5.2. As for now we leave aside the 3D world, and CHAPTER 4 focuses on reverse-engineering the knowledge gained in this chapter so it can be used to generate 2D interlocking assemblies.

REFERENCES

38 Julien Glath et al. “Thinking and Designing Reversible Structures with Non-sequential Assemblies”. In: Sept. 2022, pp. 249–259. ISBN: 978-3-031-13248-3.

97 Robert Tarjan. “Depth-First Search and Linear Graph Algorithms”. In: *SIAM Journal on Computing* 1.2 (1972), pp. 146–160. eprint: https://doi.org/10.1137/0201010. URL: https://doi.org/10.1137/0201010.

105 Ziqi Wang, Peng Song, and Mark Pauly. “DESIA: A General Framework for Designing Interlocking Assemblies”. In: *ACM Transactions on Graphics* 37.6 (Dec. 2018). ISSN: 0730-0301.

112 R.H. Wilson and T. Matsui. “Partitioning An Assembly For Infinitesimal Motions In Translation And Rotation”. In: 2 (1992), pp. 1311–1318. URL: http://ieeexplore.ieee.org/document/594555/ (visited on 11/08/2019).

113 Randall H. Wilson and Jean-Claude Latombe. “Geometric reasoning about mechanical assembly”. In: *Artificial Intelligence* 71.2 (Dec. 1994), pp. 371–396.

CHAPTER 4

2-D ASSEMBLIES

4.1 CREATING AN ASSEMBLY

In SECTION 3.1.1, we have understood how the cones of freedom of motion can be calculated given an assembly. The aim of this chapter is to reverse-engineer this relationship: how, given an ordered list of desired cones of translational freedom and centres of rotation, a 2D assembly obeying these motions can be automatically generated. We will first investigate how shall the first part P_1, the key, be created, or equivalently how to create a 2-parts assembly. Two approaches are explored: the first one, described SECTION 4.1.1, calls to basic and intuitive concepts but becomes cumbersome to implement when it comes to a part obeying a rotation. Still, it seems to be a good and easy way to first discover the issue we are trying to solve. The second one, described SECTION 4.1.2, is much more mathematically-inclined, but has the advantage of keeping things simple in rotation and is readily scalable in 3D. Once the key is created, SECTION 4.1.3 focuses on how the rest of the parts shall be created.

Let us first start this study by creating a 2-parts assembly.
At the most fundamental level, generating such an assembly boils down to drawing a separating curve linking two points on the boundary of the design domain. A first part is created by the geometry on one side of the curve, and a second part by the geometry on the other side.

4.1.1 CREATING A 2-PARTS ASSEMBLY WITH *TURTLE GRAPHICS* AND A *MARKOV PROCESS*

While this section goes into greater details, most of the results presented here are taken from our article [36].

4.1.1.1 Definitions

Turtle graphics is a popular way to introduce children to the basics of coding: a virtual `Turtle` is displayed on the screen of the computer and moves in the 2D plane according to instructions given by the user while leaving a trace on its path. These instructions are of the form "*Walk by l unit*"; "*Rotate by θ radians*". The children are then tasked to find and code a sequence of instructions that lead the `Turtle` to draw some objective design: a square, a star of David, or any more complex shape. For instance the sequence of instructions:

- `walk` 1
- `rotate` 90°
- `walk` 1
- `rotate` 90°
- `walk` 1

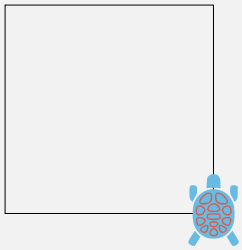


Figure 4.1| A polygon drawn by a `Turtle` .

- `rotate` 90°
- `walk` 1
- `rotate` 90°

yields the 1×1 square presented on FIGURE 4.1 (where the `Turtle` is shown in both its initial and final position and orientation).

In a spirit similar to [69], in this first approach we will use such `Turtle` as an agent that draws the polylines partitioning the design domain into parts constituting our assembly.

In probability theory, a **Markov process**, or Markov chain, ([31]) is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event (it is *memoryless*). A state is said absorbing if once entered the probability to leave it is 0. More specifically a discrete-time Markov process is a Markov chain with a countable number of states and the chain iteratively transitions between states at discrete time steps according to some probabilistic rules. In the present study, we introduce a discrete-time Markov chain with a finite number of states and one absorbing state, herein referred to simply as Markov chain, to play the role of children in Turtle graphics as the one making up the sequence of orders to move the `Turtle` with.

4.1.1.2 Overview

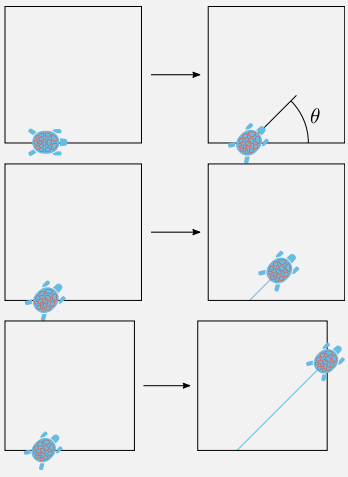
Formally speaking, such a Markov process $\mathcal{M}$ is defined as a tuple $(\mathcal{V},\mathcal{P})$ where

- $\mathcal{V}$ is the set of possible states that the chain will transition between.
  In our case $\mathcal{V}=\{$`start, rotate, walk, snap, end`$\}$.
- State `end` is absorbing: once $\mathcal{M}$ reaches this state it cannot leave it.
- $\mathcal{P}$ is the set of probabilistic rules specifying which transitions are available as well as their weights. It defines mappings $p:\mathcal{V}\to\mathcal{V}$. We provide here a succinct description of the rules emitted by the chain $\mathcal{M}$ and how they are interpreted by the `Turtle` . More details are given below.
  Seven rules are defined in $\mathcal{P}$:

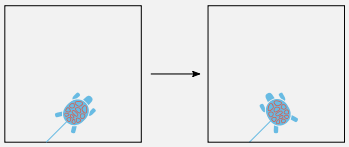
  0. `start` $\mapsto$ `rotate`
  The `Turtle` is randomly initialised on the boundary of the design domain and this order simply tells it to choose a random orientation parameterised by angle θ.

  1. `rotate` $\mapsto$ `walk`
  The `Turtle` has already chosen an orientation and must now walk forwards by a random amount.

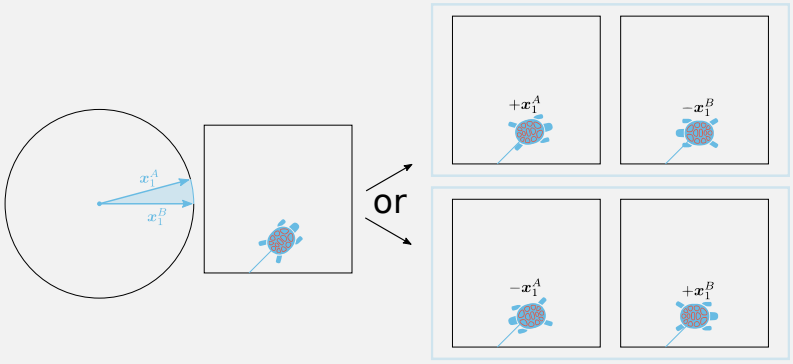
  2. `rotate` $\mapsto$ `end`
  The `Turtle` has already chosen an orientation and walks forward until meeting an edge of the design domain.



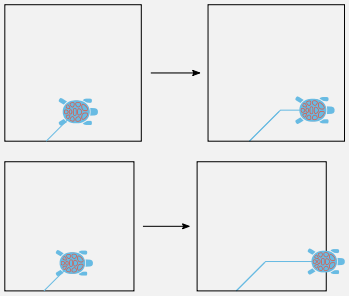
3. `walk` ↦ `rotate`
   The `Turtle` has walked to a new position and must now choose a new random orientation compatible with the prescribed cone of translation or centre of rotation.



4. `walk` ↦ `snap`
   Context: the leftmost blue cone represents the cone of translational freedom prescribed by the user: the final design must obey any direction in the cone bounded by $\boldsymbol{x}_1^A$ and $\boldsymbol{x}_1^B$. The `Turtle` has walked to a new position and must now choose to orient along either $\pm\boldsymbol{x}_1^A$ or $\mp\boldsymbol{x}_1^B$.



5. `snap` ↦ `walk`
   Similar to `rotate` ↦ `walk`: the `Turtle` has snapped to an orientation and walks forward by a random amount.
6. `snap` ↦ `end`
   Similar to `rotate` ↦ `end`: the `Turtle` has snapped to an orientation and walks forward until meeting an edge of the design domain.



When two rules apply to the same left-hand side (for instance `rotate` ↦ `walk` and `rotate` ↦ `end`) then the Markov chain $\mathcal{M}$ randomly chooses one of the two with some predefined probability as depicted on FIGURE 4.2.

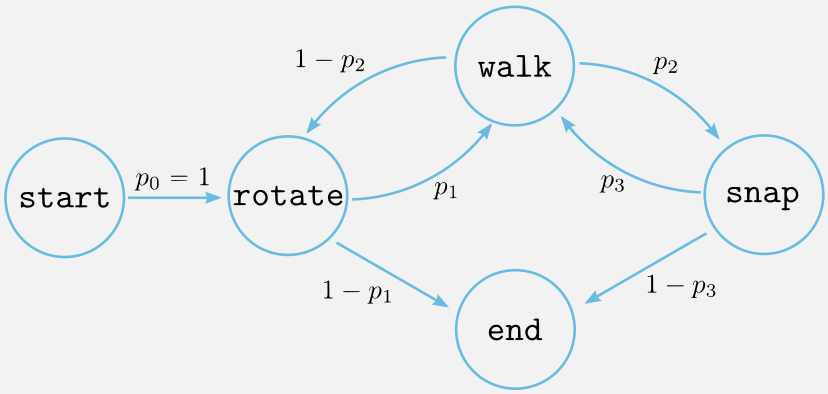


Figure 4.2| $\mathcal{M}$ transitions between states with the predefined probabilities p_i.

These strings are iteratively composed into a random sentence, for instance on FIGURE 4.3 the full sequence emitted by $\mathcal{M}$ is `start` ↦ `rotate` ↦ `walk` ↦ `snap` ↦ `walk` ↦ `rotate` ↦ `walk` ↦ `rotate` ↦ `walk` ↦ `rotate` ↦ `walk` ↦ `snap` ↦ `walk` ↦ `rotate` ↦ `walk` ↦ `rotate` ↦ `end`. At each iteration of the algorithm, the `Turtle` receives one of these five strings and acts accordingly.

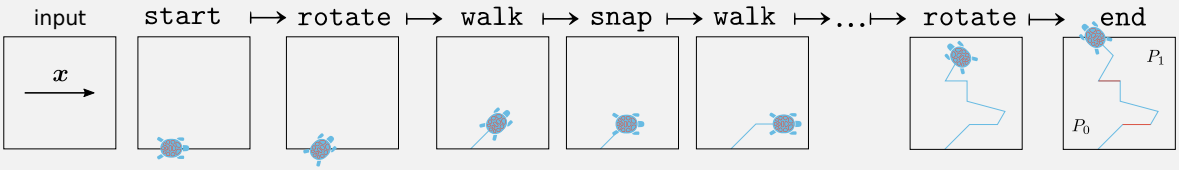


Figure 4.3| A step-by-step decomposition of the `Turtle` 's trajectory. Highlighted in red on the far right are the line segments corresponding to a `snap` order.

Since a sequence of the form [`rotate` or `snap`] $\mapsto$ `walk` defines a segment and a polyline is simply constituted of concatenated line segments, this Markov process $\mathcal{M}$ is sufficient to draw a polygonal assembly. We do not need to add rules of the kind `rotate` $\mapsto$ `rotate` as the final orientation could very well be obtained after a single `rotate` order. Similarly a `walk` $\mapsto$ `walk` can be obtained with the sequence `walk` $\mapsto$ `rotate` $\mapsto$ `walk` where the `Turtle` happens to keep the same orientation before and after the `rotate` order. We will prove below that the Markov chain $\mathcal{M}$ aided by the `Turtle` can reach the full search space of polygonal assembly.

We propose to define a Markov process $\mathcal{M}$ that will instruct a `Turtle` so that the latter draws a polyline dividing the design domain into two parts, one of which obeying a user-prescribed motion $\boldsymbol{x}$.

4.1.1.3 Generating a part obeying a translation

Let us first describe how to create a 2-parts assembly obeying a translation.
The user decides on two vectors of translation $\boldsymbol{x}_1^A$ and $\boldsymbol{x}_1^B$ bounding the cone of translational freedom of the would-be part P_1. Enforcing $\boldsymbol{x}_1^A = \boldsymbol{x}_1^B$ leads to the special case where the cone is reduced to a single vector that is to say where P_1 must translate along one direction only. Then a Markov chain $\mathcal{M}$ emits the `start` order which initialises a `Turtle` randomly on the boundary of the design domain. In subsequent iterations, $\mathcal{M}$ tells the `Turtle` whether to orient itself or to move. To comply with a `walk` order, the length l by which the `Turtle` moves is randomly picked in a user-defined interval $[l_{\min}, l_{\max}]$ (to have consistent step size, or edge length). To obey a `rotate` order, the rotation angle θ is randomly chosen in an interval such that the normal vector $\boldsymbol{n}$ of the line segment is such that $\boldsymbol{n} \cdot \boldsymbol{x}_1^A \geq 0$ and $\boldsymbol{n} \cdot \boldsymbol{x}_1^B \geq 0$. The reader's attention is drawn to the fact that this constraint on θ is enough to prevent the `Turtle` from crossing its path.

If the sequence of orders and random values leads the `Turtle` to wander outside of the design domain, a backtracking procedure is executed to replace it inside. At the end of an iteration, $\mathcal{M}$ randomly applies a production rule on the latest order it gave to get the one for the next iteration. Finally, when $\mathcal{M}$ emits the `end` order, the `Turtle` walks until meeting an edge of the design domain.
Letting the `Turtle` move like this is likely to yield appalling results for two obvious reasons:

- First, if the `Turtle` cannot cross its own path, nothing prevents it from drawing very fine details (where the angle between two successive line segments is close to π), which are impossible to manufacture and would anyway be very brittle. To prevent that, the `Turtle` is instructed to keep away from the previously drawn segment, by the mean of a user-chosen maximal angle between two successive segments.
- Second, and more importantly, the random separating polyline obtained at the end will be such that the actual cone of translational freedom of P_1 strictly includes the user-prescribed cone bounded by $\boldsymbol{x}_1^A$ and $\boldsymbol{x}_1^B$. As an extreme example, imagine that the user specified a vector $\boldsymbol{x}_1^A = \boldsymbol{x}_1^B \equiv \boldsymbol{x}_1 = (1, 0)^T$ aligned with the x axis (meaning that the user wants P_1 to translate along the horizontal axis only) and the `Turtle` drew a single line segment, as illustrated on the top row of FIGURE 4.4. Even though P_1 does

obey $\boldsymbol{x}_1$ it also obeys a full half-space of motion which is undesirable behaviour.

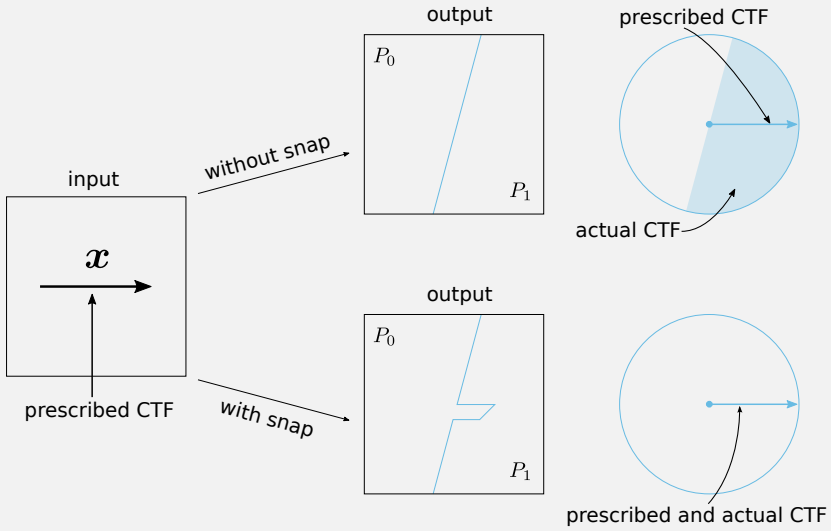


Figure 4.4| CTF stands for *Cone of translational freedom*. Two snaps are necessary to match the actual CTF of the polyline to the user-prescribed one.

To reduce the actual cone of translational freedom of P_1 to the user-prescribed one, the Markov chain $\mathcal{M}$ is enriched by a special state that we call `snap`: it forces the parameter θ to be chosen such that the `Turtle` is oriented either along $\pm\boldsymbol{x}_1^A$ or $\mp\boldsymbol{x}_1^B$. The signs ± 1 and ∓ 1 are randomly chosen at the initialisation of the `Turtle` . This ensures that when the `Turtle` `snaps` it draws a line segment whose unit normal vector $\boldsymbol{n}$ is such that either $\boldsymbol{n} \cdot \boldsymbol{x}_1^A = 0$ and $\boldsymbol{n} \cdot \boldsymbol{x}_1^B \geq 0$ or *vice-versa* by switching the superscripts ^{A} and ^{B}. As such, the `Turtle` draws a valid polyline (i.e., such that the cone of translational freedom of P_1 exactly matches the user-prescribed cone) if and only if it snapped at least twice, once along $\pm\boldsymbol{x}_1^A$ and once along $\mp\boldsymbol{x}_1^B$, see FIGURE 4.4 bottom row, which gives a computationally light manner to check whether a polyline is valid.

4.1.1.4 Surjectivity of the Markov process $\mathcal{M}$

We justify here that the Markov process $\mathcal{M}$ associated with the `Turtle` are sufficient to reach any polygonal assembly, i.e the mapping from the set made of $\mathcal{M}$ and the space of the `Turtle` 's parameters (l and θ) to the space of polygonal assembly is surjective.

Any polyline separating two parts must fit in the design domain. This observation gives an obvious upper bound on the value of $l_{\max}$, which could be the length of the diagonal of the bounding square of the design domain. In addition, setting $l_{\min} = 0$ ensures that the `Turtle` can draw infinitely small line segments and as such the full space of polygonal parts can be reached. But the mapping is not injective: for instance it is possible, although unlikely, that the magnitudes to walk or rotate by chosen by the `Turtle` for the sequence `rotate` $\mapsto$ `walk` $\mapsto$ `rotate` $\mapsto$ `walk` on the one hand and `rotate` $\mapsto$ `walk` on the other lead the same line segment, see FIGURE 4.5. As such two identical polylines can be obtained through two different sequences of orders and the mapping is not injective. Moreover, from a practical point of view, letting the `Turtle` draw infinitely small segments might not be desirable and the user may want to reduce the search space to the subset of polylines having a minimal segment length $l_{\min} > 0$. The upper bound $l_{\max}$ can also be reduced to some smaller value as any polyline with a segment length greater than $l_{\max}$ can still be reached by walking several times in the same direction. Thus the mapping to this subset is still surjective. Note that this mapping can be made injective by reducing a sequence emitted by $\mathcal{M}$ to the smallest possible word by tracking the times where the `Turtle` rotated by 0 rad (or snapped consecutively) and replacing instructions "`rotate`$(\theta_i) \mapsto$ `walk`$(l_i) \mapsto$ `rotate`$(0) \mapsto$ `walk`(l_{i+1})" with "`rotate`$(\theta_i) \mapsto$ `walk`$(l_i + l_{i+1})$".

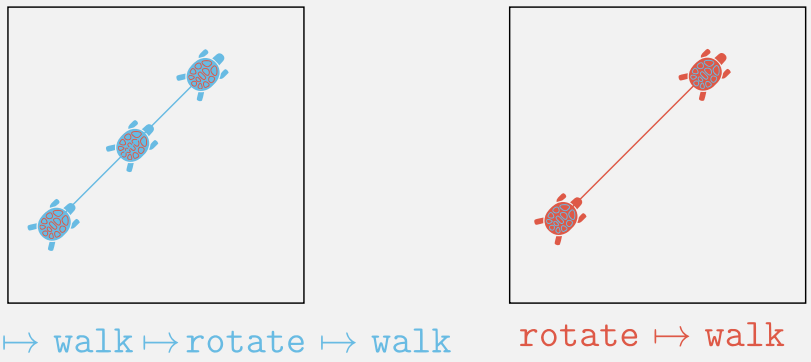
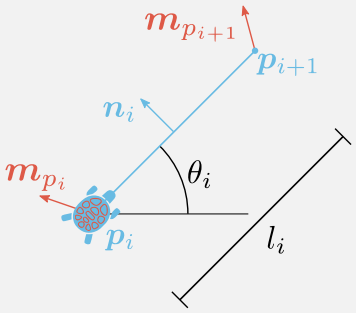


Figure 4.5| The same line segment can be reached by walking twice in the same direction by a magnitude $\frac{l}{2}$ or once by a magnitude l, for some $l \in [l_{\min}, l_{\max}]$, and thus the map is not injective.

4.1.1.5 Generating a part obeying a rotation

The process to create a 2-parts assembly obeying a rotation is basically identical to the one creating an assembly obeying a translation: the user specifies a centre of rotation $\boldsymbol{x}_r \in \mathbb{R}^2$ ($_r$ for rotation) and the Markov process $\mathcal{M}$ successively tells the `Turtle` to `walk`, `snap` or `rotate` until the `end` is reached. The only key difference is in the choice of the angle θ to orient the `Turtle` . Indeed, in translation, the cone given by the bounding directions $\boldsymbol{x}_t^A$ and $\boldsymbol{x}_t^B$ is chosen once by the user and stays fixed. It is thus easy to calculate the bounds in which θ_i must lie for the normal $\boldsymbol{n}_i$ of the i^{th} line segment to lie in the cone and thus for this segment to obey the translation. When drawing a polyline obeying a rotation, the angle θ_i depends on the location of the `Turtle` with respect to the location of the centre of rotation $\boldsymbol{x}_r$ as well as the step size l_i. Let us explore this relationship.

Notations: let $\boldsymbol{p}_i = (x_i, y_i)^T \in \mathbb{R}^2$ be the current position of the `Turtle` , and let θ_i and l_i be the orientation and length of the segment drawn by the `Turtle` between $\boldsymbol{p}_i$ and its next position $\boldsymbol{p}_{i+1} = (x_{i+1}, y_{i+1})^T = \boldsymbol{p}_i + l_i(\cos\theta_i, \sin\theta_i)^T$, as shown on the inset. The goal of this section is to find in which set shall θ_i and l_i be chosen so that the line segment $[\boldsymbol{p}_i, \boldsymbol{p}_{i+1}]$ may obey a rotation around a given centre point $\boldsymbol{x}_r = (x_r, y_r)^T \in \mathbb{R}^2$. Assuming $\boldsymbol{p}_i \neq \boldsymbol{x}_r$ (a very reasonable assumption stating that the `Turtle` is not exactly on the centre of rotation) we can define the instantaneous directions of motion of point $\boldsymbol{p}_i$, $\boldsymbol{p}_{i+1}$ with respect to $\boldsymbol{x}_r$, $\boldsymbol{m}(\boldsymbol{p}_i, \boldsymbol{x}_r)$ and $\boldsymbol{m}(\boldsymbol{p}_{i+1}, \boldsymbol{x}_r)$, abbreviated $\boldsymbol{m}_{p_i}$ and $\boldsymbol{m}_{p_{i+1}}$.



Referring to SYSTEM (3.12), one has (with $\boldsymbol{n}_i = (-\sin\theta_i, \cos\theta_i)^T$ the unit normal vector of segment $[\boldsymbol{p}_i, \boldsymbol{p}_{i+1}]$)

$$[\boldsymbol{p}_i, \boldsymbol{p}_{i+1}] \text{ obeys a rotation around } \boldsymbol{x}_r \iff \begin{cases} \boldsymbol{n}_i \cdot \boldsymbol{m}_{p_i} \star 0 \\ \boldsymbol{n}_i \cdot \boldsymbol{m}_{p_{i+1}} \star 0 \end{cases} \tag{4.1}$$

With $\star$ standing for $\geq$ if the rotation is counterclockwise, $\leq$ if clockwise. Let us introduce a sign $s = +1$ if the rotation is counterclockwise, -1 otherwise. Then SYSTEM (4.1) is rewritten:

$$[\boldsymbol{p}_i, \boldsymbol{p}_{i+1}] \text{ obeys a rotation around } \boldsymbol{x}_r \iff \begin{cases} s\boldsymbol{n}_i \cdot \boldsymbol{m}_{p_i} \geq 0 \\ s\boldsymbol{n}_i \cdot \boldsymbol{m}_{p_{i+1}} \geq 0 \end{cases} \tag{4.2}$$

$$\iff \begin{cases} s \begin{pmatrix} -\sin\theta_i \\ \cos\theta_i \end{pmatrix} \cdot \boldsymbol{m}_{p_i} \geq 0 \\ s \begin{pmatrix} -\sin\theta_i \\ \cos\theta_i \end{pmatrix} \cdot \left(\boldsymbol{m}_{p_i} + l_i \begin{pmatrix} -\sin\theta_i \\ \cos\theta_i \end{pmatrix} \right) \geq 0 \end{cases} \tag{4.3}$$

With $\boldsymbol{m}_{p_{i+1}} = \boldsymbol{m}_{p_i} + l_i(-\sin\theta_i, \cos\theta_i)^T$. For notational convenience let, in this section, $\Delta x = x_i - x_r$ and $\Delta y = y_i - y_r$. APPENDIX C.1 carries out the calculations leading to the angles defined below. When the right hand side is defined let:

$$\begin{cases} \theta_{12}^{\ominus} &= 2\arctan\frac{\Delta y - ||\boldsymbol{x}_i - \boldsymbol{x}_r||}{\Delta x} \\ \theta_{12}^{\oplus} &= 2\arctan\frac{\Delta y + ||\boldsymbol{x}_i - \boldsymbol{x}_r||}{\Delta x} \\ \theta_{34}^{\ominus} &= 2\arctan\frac{-\Delta y - \sqrt{||\boldsymbol{x}_i - \boldsymbol{x}_r||^2 - l_i^2}}{l_i - \Delta x} \\ \theta_{34}^{\oplus} &= 2\arctan\frac{-\Delta y + \sqrt{||\boldsymbol{x}_i - \boldsymbol{x}_r||^2 - l_i^2}}{l_i - \Delta x} \end{cases} \qquad \begin{cases} \theta_0 &= 2\arctan\frac{-l_i}{\Delta y} \\ \theta_1 &= \min(\theta_{12}^{\ominus}, \theta_{12}^{\oplus}) \\ \theta_2 &= \max(\theta_{12}^{\ominus}, \theta_{12}^{\oplus}) \\ \theta_3 &= \min(\theta_{34}^{\ominus}, \theta_{34}^{\oplus}) \\ \theta_4 &= \max(\theta_{34}^{\ominus}, \theta_{34}^{\oplus}) \end{cases}$$

A few calculations, developed in APPENDIX C.3, provide the enumeration of cases one need to look at to solve for θ_i, presented below.

1. If $\Delta x = l_i$

(a) If $\Delta y = 0$:

- If $s = -1$: $\theta_i \in \emptyset$, no solution.
- If $s = +1$: $\theta_i \in [\theta_1, \theta_2]$

(b) If $\Delta y \neq 0$:

- If $s = +1$: $\theta_i \in [\theta_1, \theta_2]$
- Else:
  - If $\Delta y > 0$: $\theta_i \in [-\pi, \theta_0]$
  - If $\Delta y < 0$: $\theta_i \in [\theta_0, \pi]$

2. Else if $||\boldsymbol{p}_i - \boldsymbol{x}|| < l_i$

(a) If $s = -1$: $\theta_i \in \emptyset$, no solution.

(b) If $s = +1$:

- If $\Delta x = 0$:
  - If $\Delta y < 0$: $\theta_i \in [-\pi, 0]$
  - If $\Delta y > 0$: $\theta_i \in [0, \pi]$
- If $\Delta x > 0$: $\theta_i \in [\theta_1, \theta_2]$
- If $\Delta x < 0$: $\theta_i \in [-\pi, \theta_1] \cup [\theta_2, \pi]$

3. Else if $||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i$

(a) If $\Delta x = 0$:

- If $s = -1$: $\theta_i \in [\theta_3, \theta_4]$
- If $s = +1$:
  - If $\Delta y < 0$: $\theta_i \in [-\pi, 0]$

- If $\Delta y > 0$: $\theta_i \in [0, \pi]$

(b) If $\Delta x \neq 0$:

- If $s(l_i - \Delta x) < 0$:
  - If $s\Delta x > 0$: $\theta_i \in [\max(\theta_3, \theta_1), \min(\theta_4, \theta_2)]$
  - If $s\Delta x < 0$: $\theta_i \in [\theta_3, \theta_4]$
- If $s(l_i - \Delta x) > 0$:
  - If $s\Delta x > 0$: $\theta_i \in [\theta_1, \theta_2]$
  - If $s\Delta x < 0$: $\theta_i \in [-\pi, \min(\theta_3, \theta_1)] \cup [\max(\theta_4, \theta_2), \pi]$

Once this enumeration of *if- else if - else* conditions is implemented, the orientation of the `Turtle` is chosen in the corresponding interval, and we can be sure that the polyline drawn as such obeys a rotation around $\boldsymbol{x}_r$.

4.1.1.6 Notes and issues related to the behaviour of the `Turtle`

APPENDIX C.4 proves an intriguing property of the `Turtle` :

- If $s = -1$, (i.e. the user asks for a part P_1 obeying a *clockwise* rotation around centre $\boldsymbol{x}_r$) APPENDIX C.4 proves that in this case, centre $\boldsymbol{x}_r$ becomes attractive, meaning that the `Turtle` must always get closer to $\boldsymbol{x}_r$: $||\boldsymbol{p}_{i+1} - \boldsymbol{x}_r|| < ||\boldsymbol{p}_i - \boldsymbol{x}_r||$.
- On the contrary, if $s = +1$ (one wants a *counterclockwise* rotation) then $\boldsymbol{x}_r$ becomes repulsive: $||\boldsymbol{p}_{i+1} - \boldsymbol{x}_r|| > ||\boldsymbol{p}_i - \boldsymbol{x}_r||$.

To further our understanding of the `Turtle` , several other properties may be proven:

- We state that $\theta_2 - \theta_1 = \pi$:

  Proof. Since $x \mapsto \arctan(x)$ takes value in $]-\frac{\pi}{2}, \frac{\pi}{2}[$ and on this interval cosine is positive, one has for $x \in \mathbb{R}$: $\cos\arctan(x) = \frac{1}{\sqrt{1+x^2}}$. Moreover $\sin\arctan(x) = \frac{x}{\sqrt{1+x^2}}$. Let $a_{1,2}$ be such that $\theta_{1,2} = 2\arctan(a_{1,2})$.

$$\begin{aligned}\cos\frac{\theta_2 - \theta_1}{2} &= \cos\frac{\theta_2}{2}\cos\frac{\theta_1}{2} + \sin\frac{\theta_2}{2}\sin\frac{\theta_1}{2} \\ &= \frac{1}{\sqrt{1+a_2^2}}\frac{1}{\sqrt{1+a_1^2}} + \frac{a_2}{\sqrt{1+a_2^2}}\frac{a_1}{\sqrt{1+a_1^2}} \\ &= \frac{1 + a_1 a_2}{\sqrt{1+a_1^2}\sqrt{1+a_2^2}}\end{aligned}$$

  And:

$$\begin{aligned}1 + a_1 a_2 &= 1 + \frac{\Delta_y - ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta_x}\frac{\Delta_y + ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta_x} \\ &= \frac{\Delta_x^2 + \Delta_y^2 - ||\boldsymbol{p}_i - \boldsymbol{x}||^2}{\Delta_x^2} \\ &= 0\end{aligned}$$

  We have proven that $\cos\frac{\theta_2 - \theta_1}{2} = 0$ and hence (since $\theta_2 \geq \theta_1$) that $\theta_2 - \theta_1 = \pi$. It means that when the the orientation of the `Turtle` must be either $\theta \in [\theta_1, \theta_2]$ or $\theta \in [-\pi, \theta_1] \cup [\theta_2, \pi]$ then angle θ points in the half-plane whose bounding line crosses $\boldsymbol{p}_i$ and is orthogonal to the vector $\boldsymbol{p}_i - \boldsymbol{x}_r$. □

- A careful examination of the enumeration of cases above, as well as APPENDIX C.3, shows that when $s = +1$ angles θ_3 and θ_4 do not appear: the `Turtle` must choose its orientation θ either in the interval $[\theta_1, \theta_2]$ or $[-\pi, \theta_1] \cup [\theta_2, \pi]$. Bearing in mind the previous remark, we see that when $s = +1$, the `Turtle` must orient itself in a half-plane pointing away from the centre $\boldsymbol{x}_r$.

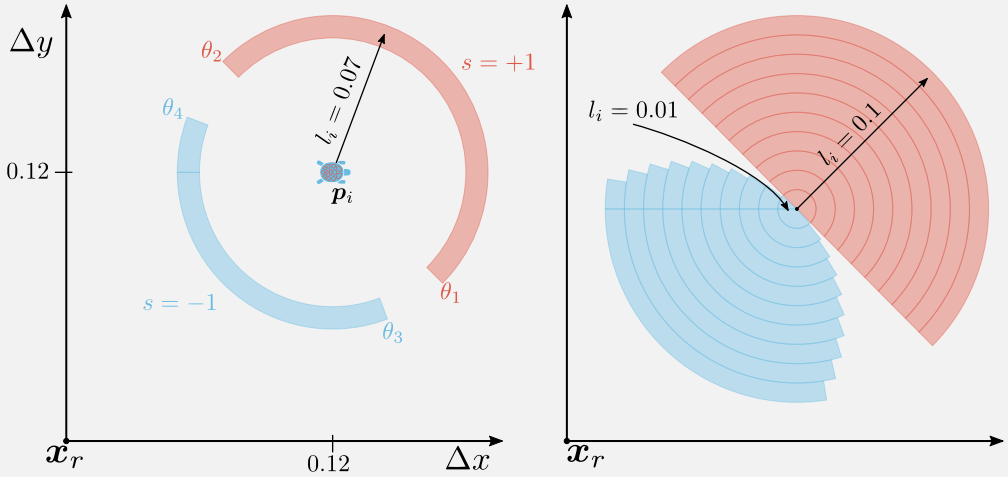


Figure 4.6| Left: sets of possible next position of the `Turtle` , $\boldsymbol{p}_{i+1}$, for a fixed l_i. Right: sets of position when l_i is left variable between two bounds.

These properties are illustrated on FIGURE 4.6: the current position of the `Turtle` is $\boldsymbol{p}_i$ at $(0.12, 0.12)^T$ from the centre $\boldsymbol{x}_r$. On the left of this figure, for a given $l_i = 0.07$, angles θ_i, $i \in [\![1, 4]\!]$ are calculated using the enumeration given above. The `Turtle` will have to move to its next position $\boldsymbol{p}_{i+1}$ either in the range of positions in blue if $s = -1$ or in red if $s = +1$. On the right of FIGURE 4.6 l_i is left variable in the range $l_{min} = 0.01$ and $l_{max} = 0.1$; as such it allows to see the region on which the `Turtle` shall go. FIGURE 4.7 shows the same data but for multiple `Turtle` 's position $\boldsymbol{p}_i$, arranged in a square grid centred on $\boldsymbol{x}_r$ on the left, along a spiral around $\boldsymbol{x}_r$ on the right. Positions $\boldsymbol{p}_i$ are shown using small black dots while the centre $\boldsymbol{x}_r$ is depicted as the biggest black dot. On the left, we can confirm that, if $s = +1$ the `Turtle` shall move in a half-plane opposite to where the centre $\boldsymbol{x}_r$ lies, and there is no relationship between the angle θ_i and the step length l_i (the red semi-disks are pointing away from $\boldsymbol{x}_r$). The blue cones show that, when $s = -1$, the `Turtle` shall only move towards $\boldsymbol{x}_r$, and the set of possible positions is highly non-linear in l_i, θ_i, Δx and Δy. On the right of FIGURE 4.7, only the regions of possible positions for $s = -1$ is shown. This zoomed-in view illustrates again the non-linearity of these sets, and also shows that if $\boldsymbol{p}_i$ is too close to $\boldsymbol{x}_r$ and l_i is too large, no solution can be found, the `Turtle` is stuck.

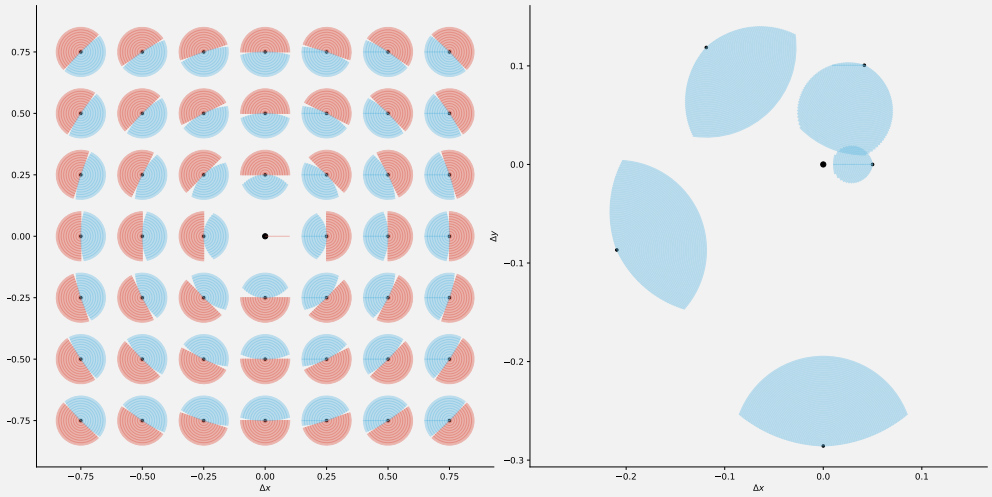


Figure 4.7| Left: zoomed out view of the sets of possible positions for the next position $\boldsymbol{p}_{i+1}$ for various positions of $\boldsymbol{p}_i$.. Right: zoomed in view of the sets for $s = -1$ and for various positions of $\boldsymbol{p}_i$.

FIGURE 4.8 shows the limit trajectory of the `Turtle` in blue for $s = -1$ (resp. in red for $s = +1$): for a given l_i, the same angle θ_i, $i \in \{3, 4\}$ (resp. $i \in \{1, 2\}$) was chosen as to maximise (resp. minimise) the ratio $\frac{||\boldsymbol{p}_{i+1} - \boldsymbol{x}_r||}{||\boldsymbol{p}_i - \boldsymbol{x}_r||}$ at each successive iteration. The starting position of the `Turtle` is depicted by the cartoon image of a turtle. In literary words, it is as if the `Turtle` was trying as much as possible not to go near $\boldsymbol{x}_r$ when $s = -1$ (resp. not to move away from $\boldsymbol{x}_r$ when $s = +1$). Yet, as seen in this figure, when $s = -1$ the `Turtle` is inescapably drawn to the centre $\boldsymbol{x}_r$, acting like a black hole, while when $s = -1$ the `Turtle` can do nothing but drift away from $\boldsymbol{x}$.

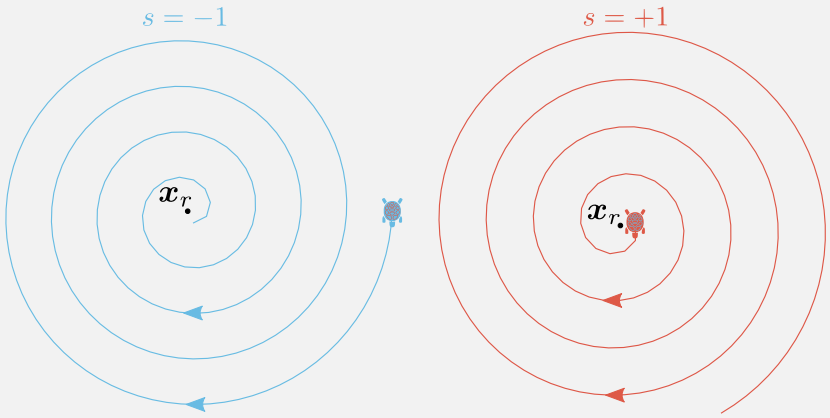


Figure 4.8| The limit trajectories of the `Turtle` : if $s = +1$ the `Turtle` is drawn to the centre $\boldsymbol{x}_r$; if $s = -1$ the `Turtle` is repelled by it.

This attractiveness of $\boldsymbol{x}_r$ happens to be quite problematic: if $\boldsymbol{x}_r$ is inside the design domain and $s = +1$ then the `Turtle` may get closer and closer to $\boldsymbol{x}_r$ up to the point where $||\boldsymbol{p}_i - \boldsymbol{x}_r|| < l_i$ and no solution exists: it will be unable to draw a valid partitioning curve. Such situation is illustrated on FIGURE 4.9.

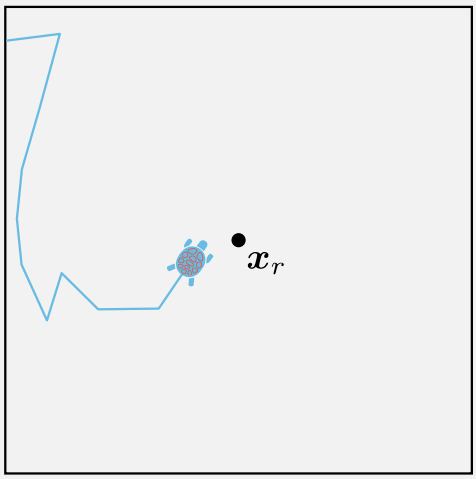


Figure 4.9| The `Turtle` is stuck!

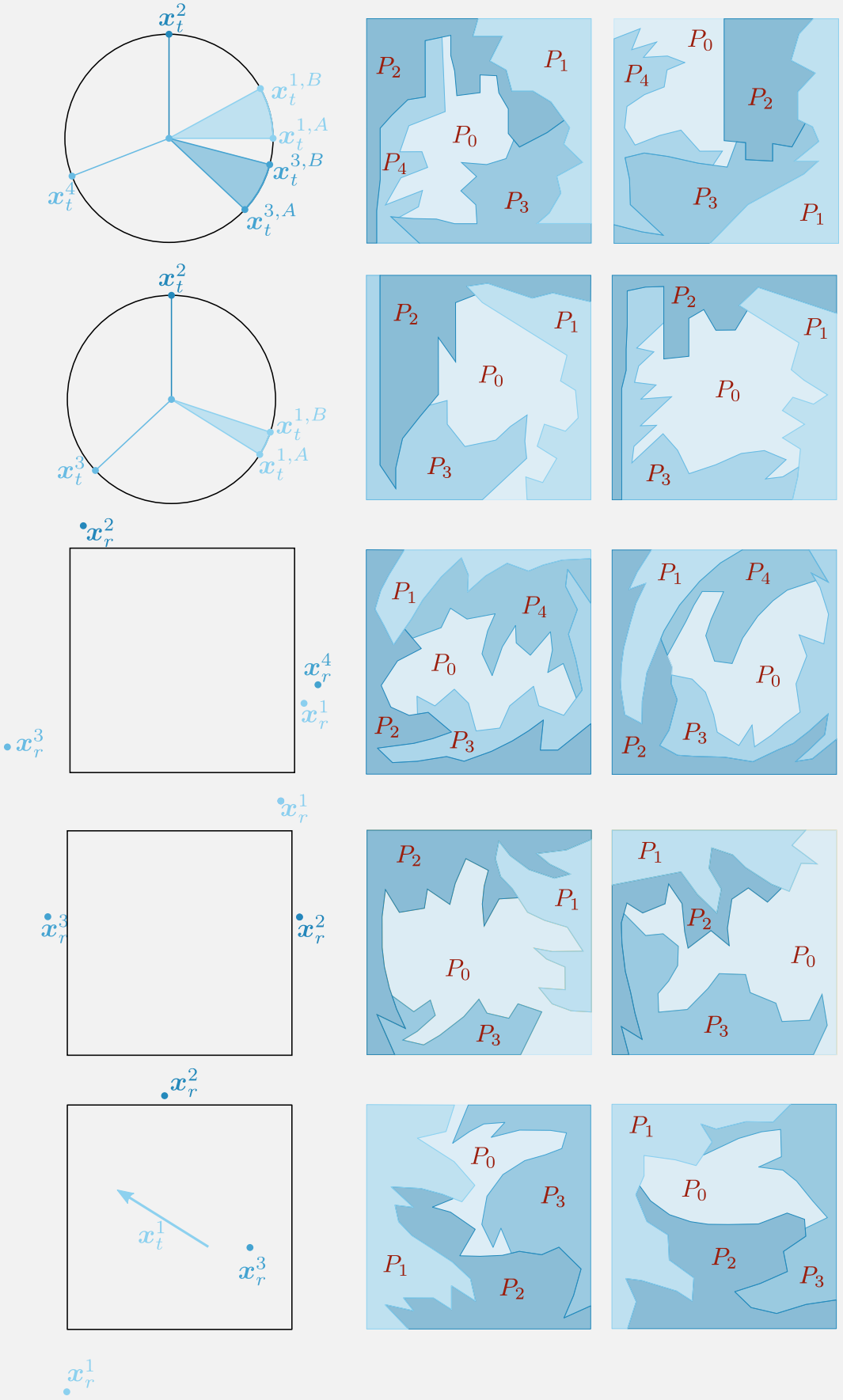


Figure 4.10| Designs obtained with the `Turtle` and the Markov process. From top to bottom: the first two rows present assemblies obeying pure translations with the prescribed cone of freedom on the left. The next two rows show assemblies obeying pure rotations, with the location of the centres shown on the left. The bottom row shows an assembly whose key P_1 obeys both a translation and a rotation.

4.1.2 A BETTER APPROACH TO CREATE A 2-PARTS ASSEMBLY: *GUIDED PROJECTION ALGORITHM*

The approach combining a Markov process and Turtle graphics turns out to be problematic for three reasons:

- As explained at the end of the previous section and shown on FIGURE 4.9, it may happen that the `Turtle` gets stuck when $s = +1$ and the centre of rotation is ill-placed. This property is highly undesirable, as one can only start again the algorithm from scratch when it happens, thus wasting both time and computational resources.
- Secondly, while this approach ensures that the whole space of separating polyline is explored, it leaves very little control to the user: one can, at most, only specify the starting point of the `Turtle` and the range $[l_{min}, l_{max}]$ of the step length to guide the design. This lack of control can lead to the creation of degenerate designs that are of no interest for any practical use.
- Finally, this approach does not seem to be scalable to 3D: we could not find any way in which a `Turtle` (or several) flying in space would generate a 3D assembly.

That being said, it is still possible to generate interlocking assemblies with the `Turtle` . Our article [36], as well as our video [78] show built assemblies with novel shapes that are interlocking. Some examples are shown in FIGURE 4.10.

The desire to do better led to an approach that solves these three points while still leaving the possibility to explore the full space of polygonal assembly. It is derived from the *guided projection algorithm* (GPA), presented by Tang and coauthors in [96]. Their paper is briefly summed up hereunder.

This article is part of the field of computational geometry for architectural design. It develops a framework for fast, interactive, form-finding of physically stable polyhedral meshes through a constrained iterative optimisation scheme. The authors pay special attention to several constraints, such as boundary interpolation, planarity of faces, statics, panel size and shape or enclosed volume, but their approach can easily be adapted to handle other constraints. The problem to solve is simplified by introducing auxiliary variables and equations to ensure that the constraints are at most quadratic. The gradient is therefore easy to calculate and takes part in an iterative, Newton-type, optimisation program. Successive solutions are biased (projected) towards both low-energy of a fairness metric and low distance to the previous solution.

This summary may feel quite abstract, so let's dig into the specifics of the framework by solving a toy problem.

4.1.2.1 Toy example

Assume we want to solve the following cubic equation:

$$x^3 - 2x^2 - x + 2 = 0 \tag{4.4}$$

Whose roots trivially are 1, -1 and 2. The method described in [96] shall find one of these roots. The first step is to introduce as many variables as needed to make the problem at most quadratic. Here we only need to introduce one variable $y = x^2$:

$$\text{EQUATION } (4.4) \iff \begin{cases} xy - 2y - x + 2 = 0 \\ x^2 - y = 0 \end{cases} \tag{4.5}$$

The unknowns x and y are stored in a vector $\boldsymbol{X} = \begin{pmatrix} x \\ y \end{pmatrix}$. System (4.5) is written in the standard form $\phi_i(\boldsymbol{X}) = \frac{1}{2}\boldsymbol{X}^T\boldsymbol{H}_i\boldsymbol{X} + \boldsymbol{b}_i^T\boldsymbol{X} + c_i = 0, i \in \{1,2\}$ (as we only have two equations) with $\boldsymbol{H}_i$ a symmetric matrix, $\boldsymbol{b}_i$ a column vector and c_i as scalar. Here:

$$\boldsymbol{H}_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \boldsymbol{b}_1 = \begin{pmatrix} -1 \\ -2 \end{pmatrix} \qquad c_1 = 2$$
$$\boldsymbol{H}_2 = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix} \qquad \boldsymbol{b}_2 = \begin{pmatrix} 0 \\ -1 \end{pmatrix} \qquad c_2 = 0$$

The goal is to find a solution $\boldsymbol{X}^\star$ such that for $i \in \{1,2\}$, $\phi_i(\boldsymbol{X}^\star) = 0$. At iteration n we have an instance of an almost-solution $\boldsymbol{X}_n$. Using a first-order Taylor expansion one gets:

$$\begin{aligned} &\phi_i(\boldsymbol{X}^\star) = 0 \\ &\phi_i(\boldsymbol{X}^\star) \simeq \phi_i(\boldsymbol{X}_n) + \nabla\phi_i(\boldsymbol{X}_n)^T(\boldsymbol{X}^\star - \boldsymbol{X}_n) \end{aligned} \tag{4.6}$$

Where $\nabla\phi_i(\boldsymbol{X}_n)$ is the gradient of ϕ_i evaluated at point $\boldsymbol{X}_n$:

$$\nabla\phi_i(\boldsymbol{X}_n) = \boldsymbol{H}_i\boldsymbol{X}_n + \boldsymbol{b}_i$$

By setting equation (4.6) to 0 one gets:

$$\forall i,\ \phi_i(\boldsymbol{X}_n) + \nabla\phi_i(\boldsymbol{X}_n)^T(\boldsymbol{X} - \boldsymbol{X}_n) = 0 \iff \boldsymbol{H}\boldsymbol{X} = \boldsymbol{r} \tag{4.7}$$

With

$$\boldsymbol{H} = \begin{pmatrix} \vdots \\ \nabla\phi_i(\boldsymbol{X}_n)^T \\ \vdots \end{pmatrix} \qquad \boldsymbol{r} = \begin{pmatrix} \vdots \\ -\phi_i(\boldsymbol{X}_n) + \nabla\phi_i(\boldsymbol{X}_n)^T\boldsymbol{X}_n \\ \vdots \end{pmatrix}$$

In our specific example:

$$\boldsymbol{H} = \begin{pmatrix} y_n - 1 & x_n - 2 \\ 2x_n & -1 \end{pmatrix} \qquad \boldsymbol{r} = \begin{pmatrix} x_n y_n - 2 \\ x_n^2 \end{pmatrix}$$

While in our example $\boldsymbol{H}$ happens to be square and invertible for most values of (x_n, y_n), it is typically not the case in a general setting. The linear system $\boldsymbol{H}\boldsymbol{X} = \boldsymbol{r}$ is often underdetermined, and the space it maps to is ill-conditioned, making this system unsuitable for further computation. The trick used in [96] is to solve this system in the least-square sense by using the distance from the previous solution $\boldsymbol{X}_n$ as a regularizer[1]; the solution obtained is denoted by $\boldsymbol{X}_{n+1}$:

$$||\boldsymbol{H}\boldsymbol{X}_{n+1} - \boldsymbol{r}||^2 + \epsilon^2||\boldsymbol{X}_{n+1} - \boldsymbol{X}_n||^2 \to \min \tag{4.8}$$

[1] A fairness energy is also used in the paper, but we do not present it here as such energy will not be used to generate assemblies.

Where ϵ is a small value (typically $\epsilon = 0.001$); the larger ϵ, the closer $\boldsymbol{X}_{n+1}$ to $\boldsymbol{X}_n$. EQUATION (4.8) is quadratic and has thus only one minimum, found by differentiation:

$$(\boldsymbol{H}^T\boldsymbol{H} + \epsilon^2\boldsymbol{I})\boldsymbol{X}_{n+1} = \boldsymbol{H}^T\boldsymbol{r} + \epsilon^2\boldsymbol{X}_n \tag{4.9}$$

$\boldsymbol{X}_{n+1}$ is computed from SYSTEM (4.9) by Cholesky factorization. The additive term $\epsilon^2\boldsymbol{I}$ is of importance: while in theory $\boldsymbol{H}^T\boldsymbol{H}$ has non negative eigenvalues, in practice it is often near singular and numerical imprecision may shift one eigenvalue below or equal to 0. Adding this diagonal of small values ensures that the matrix $(\boldsymbol{H}^T\boldsymbol{H} + \epsilon^2\boldsymbol{I})$ is symmetric definite positive. This process is known as the Tikhonov regularization. FIGURE 4.11 top row shows the basins of attraction of the method for different starting value x_0. One sees that depending on the initial value x_0, the algorithm finds either one of the three roots -1, 1 and 2, highlighted using coloured dots. The curve highlighted in red shows the convergence of the algorithm for a starting value $x_0 = -3$; next, at iteration 1 $x_1 \simeq -1.95$, then $x_2 \simeq -1.3$ etc. until convergence to the root -1. FIGURE 4.11 bottom row shows the absolute error of successive solutions x_i, starting at $x_0 = -3$, with respect to the root -1: the convergence is quadratic, as expected for such method. Finally, one notices that the algorithm could not converge for $x_0 = -0.2$ (no curve starts from that value on the top graph): this is to be expected with Newton's method, it can be unstable.

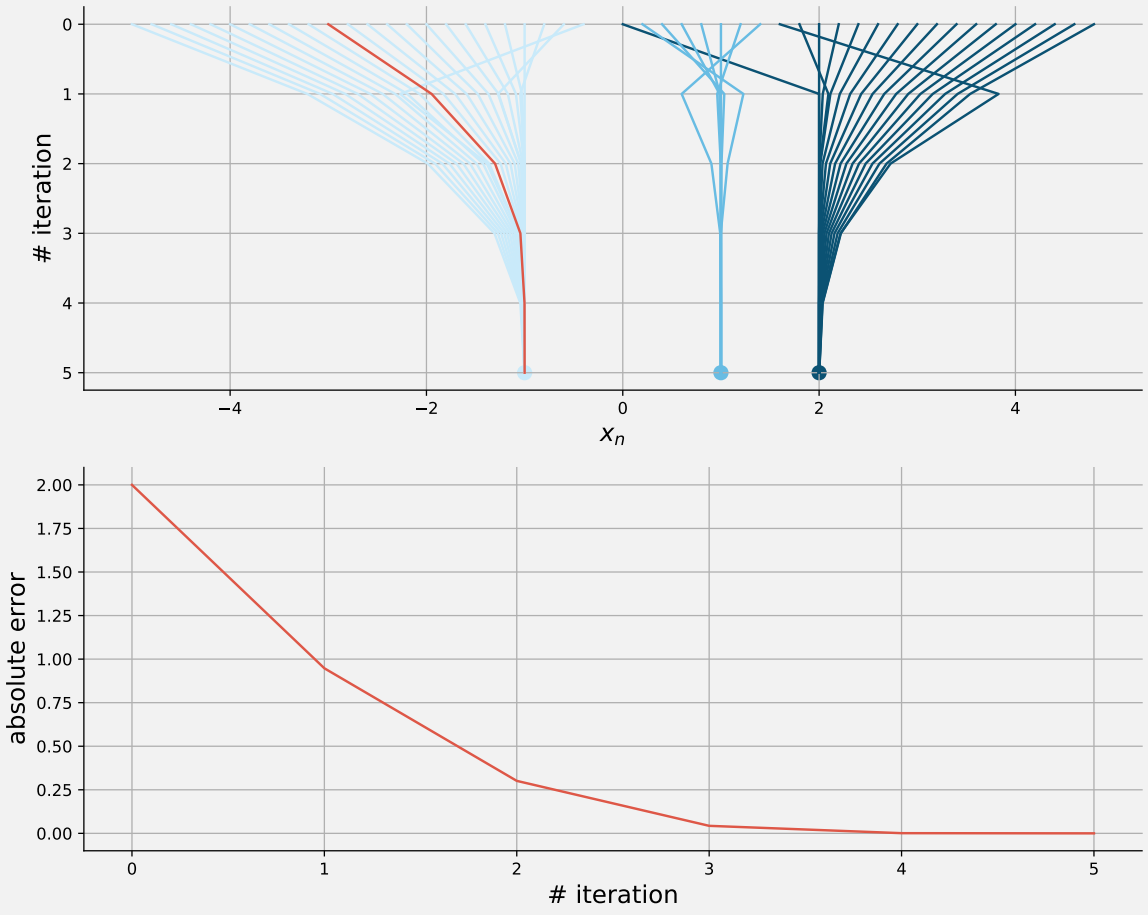


Figure 4.11| Basins of attraction and convergence of the error. The three roots are highlighted with the coloured dots.

4.1.2.2 On the generation of a 2-parts assembly obeying a translation using the *Guided Projection Algorithm*

This section aims to optimise, through the guided projection algorithm (GPA), the geometry of a polyline so that it obeys a translation.
We want to partition the design domain into two parts such that one of them obeys a direction of translation $\boldsymbol{x} \in \mathcal{S}^1$. In SECTION 3.1.1.1 we understood that given a polyline whose constitutive points are $\boldsymbol{p}_1, ..., \boldsymbol{p}_k$, of coordinates $\boldsymbol{p}_i = (x_i, y_i)^T$, it obeys $\boldsymbol{x}$ if and only if

$$\boldsymbol{A}_t \boldsymbol{x} \geq 0 \qquad\qquad \boldsymbol{A}_t = \begin{pmatrix} \vdots & \vdots \\ y_i - y_{i+1} & x_{i+1} - x_i \\ \vdots & \vdots \end{pmatrix} \in \mathbb{R}^{k \times 2}$$

Constraining the polyline to obey a translation

In SECTION 3.1.1.1 the polyline was given, and the set of linear inequalities $\boldsymbol{A}_t \boldsymbol{x}_t \geq 0$, of unknown $\boldsymbol{x}_t$, gives the cone of translational freedom of the polyline. In this section, we reverse our point of view: the cone of translational freedom is fixed, decided upon by the user (and for simplicity, we first assume the cone to be reduced to a single direction $\boldsymbol{x}_t \in \mathcal{S}^1$), and we want to find a polyline obeying it: in $\boldsymbol{A}_t \boldsymbol{x}_t \geq 0$, matrix $\boldsymbol{A}_t$ is now the variable, and $\boldsymbol{x}_t$ is now the parameter. To use the GPA, we have to transform the set of linear inequalities $\boldsymbol{A}_t \boldsymbol{x}_t \geq 0$ into a set of at most quadratic equations $\phi_i(\boldsymbol{X}) = \frac{1}{2} \boldsymbol{X}^T \boldsymbol{H}_i \boldsymbol{X} + \boldsymbol{b}_i^T \boldsymbol{X} + c_i = 0$.
Let $\boldsymbol{x}_t = (x, y)^T$.

$$\begin{aligned} \boldsymbol{A}_t \boldsymbol{x}_t \geq 0 &\iff \forall i \in [\![1, k-1]\!],\ (y_i - y_{i+1})x + (x_{i+1} - x_i)y \geq 0 \\ &\iff \forall i \in [\![1, k-1]\!]\, \exists \alpha_i \in \mathbb{R},\ (y_i - y_{i+1})x + (x_{i+1} - x_i)y = \alpha_i^2 \\ &\iff \forall i \in [\![1, k-1]\!]\, \exists \alpha_i \in \mathbb{R},\ -\alpha_i^2 + \begin{pmatrix} -y & x & y & -x \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ x_{i+1} \\ y_{i+1} \end{pmatrix} = 0 \end{aligned}$$

Notation

Let $\boldsymbol{M} = (m_{i,j}) \in \mathbb{R}^{n \times n}$ a symmetric matrix and $\boldsymbol{v} = (v_i) \in \mathbb{R}^n$ for some $n \in \mathbb{N}$. The notations

$$\boldsymbol{M} = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \begin{matrix} \leftarrow i \\ \leftarrow j \end{matrix} \qquad\qquad \boldsymbol{v} = \begin{pmatrix} d \\ e \end{pmatrix} \begin{matrix} \leftarrow k \\ \leftarrow l \end{matrix}$$

means that $\boldsymbol{M}$ is filled with 0, except at then entries crossing indices i and j: $m_{i,i} = a$, $m_{i,j} = m_{j,i} = b$ and $m_{j,j} = c$. Similarly, $\boldsymbol{v}$ has 0 coefficients everywhere, except $v_k = d$ and $v_l = e$. In the following definitions $\boldsymbol{H}_i$ is a symmetric matrix of size the dimension of vector $\boldsymbol{X}$.

Let $o = 2k$ be the index such that $\boldsymbol{X}[o+i]$ maps to α_i. Let

$$\boldsymbol{X} = \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ \vdots \\ y_k \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{k-1} \end{pmatrix} \qquad \boldsymbol{H}_i = \begin{pmatrix} -2 \end{pmatrix} \leftarrow o+i \qquad \boldsymbol{b}_i = \begin{pmatrix} -y \\ x \\ y \\ -x \end{pmatrix} \begin{matrix} \leftarrow 2i \\ \leftarrow 2i+1 \\ \leftarrow 2i+2 \\ \leftarrow 2i+3 \end{matrix} \qquad c_i = 0$$

Thus, one has:

$$\boldsymbol{A}_t \boldsymbol{x} \geq 0 \iff \forall i \in [\![1, k-1]\!],\ \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_i \boldsymbol{X} + \boldsymbol{b}_i^T \boldsymbol{X} + c_i = 0$$

Then, as explained in SECTION 4.1.2.1, the gradients of each ϕ_i are calculated, $\nabla\phi_i(\boldsymbol{X}) = \boldsymbol{H}_i\boldsymbol{X} + \boldsymbol{b}_i$ and stacked in a matrix $\boldsymbol{H}$. Similarly, scalars $-\phi_i(\boldsymbol{X}) + \nabla\phi_i(\boldsymbol{X})^T\boldsymbol{X}$ are stored in a vector $\boldsymbol{r}$, and the next almost-solution is solution to the system $(\boldsymbol{H}^T\boldsymbol{H} + \epsilon^2\boldsymbol{I})\boldsymbol{X}_{n+1} = \boldsymbol{H}^T\boldsymbol{r} + \epsilon^2\boldsymbol{X}_n$.
The first instance $\boldsymbol{X}_0$ is initialised as follows: the user inputs a polyline and the coordinates x_i, y_i of each point $\boldsymbol{p}_i$ are stored at the appropriate location ($2i$ and $2i+1$) in $\boldsymbol{X}_0$. Regarding α_i, if $(y_i - y_{i+1})x + (x_{i+1} - x_i)y \geq 0$, meaning that the segment $[\boldsymbol{p}_i, \boldsymbol{p}_{i+1}]$ obeys the direction of translation $\boldsymbol{x}_t$, then $\alpha_i = \sqrt{(y_i - y_{i+1})x + (x_{i+1} - x_i)y}$; otherwise $\alpha_i = 0$. α_i is then stored at index $2k+i$ in $\boldsymbol{X}_0$.

Note that had we wanted the polyline to obey a cone bounded by two vectors $\boldsymbol{x}_t^A$ and $\boldsymbol{x}_t^B$ instead of a single direction $\boldsymbol{x}_t$, one should only double the number of constraints $\phi_i \rightarrow \phi_i^A, \phi_i^B$ and the number of variables $\alpha_i \rightarrow \alpha_i^A, \alpha_i^B$.

FIGURE 4.12 shows a polyline before and after a single step of the optimisation (the optimisation was stopped after $\boldsymbol{X}_1$ being calculated). The direction of translation is horizontal: $\boldsymbol{x}_t = (1,0)^T$. On the left, the provided polyline is not compatible with this direction, one clearly sees line segments whose normal vectors are pointing away from $\boldsymbol{x}_t$. On the contrary, after a GPA step, the polyline on the right shows that all its segments have their normal vectors $\boldsymbol{n}_i$ compatible with $\boldsymbol{x}_t$: $\boldsymbol{n}_i \cdot \boldsymbol{x}_t \geq 0$.

With the given constraints ϕ_i, we can only make a polyline floating in the plane obey a translation. Yet to generate a 2-parts assembly, several additional constraints must be implemented:

- The endpoints of the polyline shall precisely lie on the edges of the design domain.
- To prevent too fine details:
  - The length of each line segment should be greater than some threshold l_{min}.
  - The curvature at each vertex of the polyline should not be too high, so as to avoid thin tines. Thus the angle between two successive segments should be greater than some threshold.
  - the so-called interior points of the polyline (all the points but the endpoints) should not be too close to the edges of the design domain, so as to avoid bottleneck-like features.
- Obviously, all the polyline points should be inside the design domain.
- A counterpart of the `snap` order should be implemented, to make sure that the actual cone of freedom of the polyline precisely matches the user-given ones.

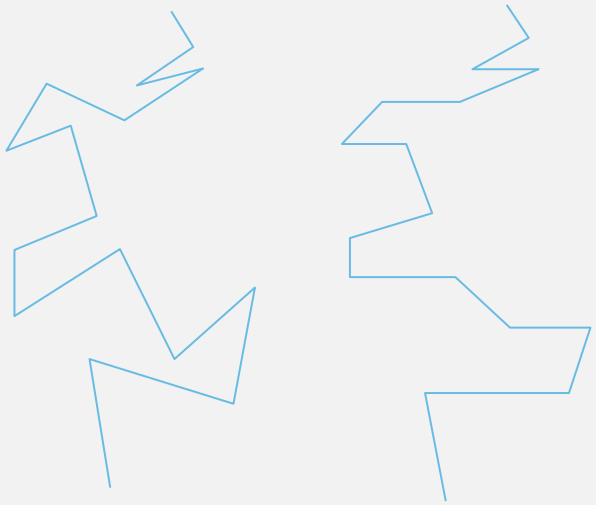


Figure 4.12| A polyline before (left) and after (right) a single step of GPA.

The following paragraphs address these points.

Constraint on the minimal length of the segments

Two constraints must be implemented: one that calculates the length of the segment, the other that compares it to the user-given threshold l_{min}.

1. Constraint to calculate the length l_i of a segment:
For a segment $i \in [\![1, k-1]\!]$ its length l_i is given by $l_i^2 = (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2$. Let the integer o be the index such that $\boldsymbol{X}[o+i]$ stores the value of l_i. With:

$$\boldsymbol{H}_i = \begin{pmatrix} 2 & 0 & -2 & 0 & 0 \\ 0 & 2 & 0 & -2 & 0 \\ -2 & 0 & 2 & 0 & 0 \\ 0 & -2 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & -2 \end{pmatrix} \begin{matrix} \leftarrow 2i \\ \leftarrow 2i+1 \\ \leftarrow 2i+2 \\ \leftarrow 2i+3 \\ \leftarrow o+i \end{matrix} \qquad \boldsymbol{b}_i = \boldsymbol{0} \qquad c_i = 0$$

One has

$$(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 - l_i^2 = 0 \iff \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_i \boldsymbol{X} + \boldsymbol{b}_i^T \boldsymbol{X} + c_i = 0$$

2. Constraint imposing $l_i \geq l_{min}$:
Note that $l_i \geq l_{min} \iff l_i^2 - l_{min}^2 - \mu_i^2 = 0$ for some μ_i. Let o_l (resp. o_m) be the index such that $\boldsymbol{X}[o_l + i]$ (resp. $\boldsymbol{X}[o_m + i]$) maps to l_i (resp. μ_i). With:

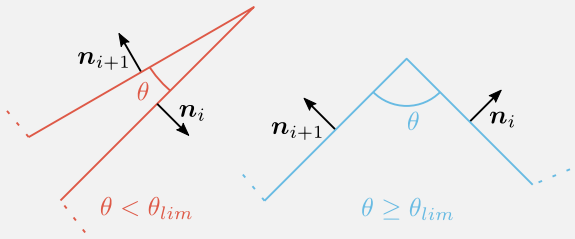
$$\boldsymbol{H}_i = \begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix} \begin{matrix} \leftarrow o_l + i \\ \leftarrow o_m + i \end{matrix} \qquad \boldsymbol{b}_i = \boldsymbol{0} \qquad c_i = -l_{min}^2$$

One has

$$l_i^2 - l_{min}^2 - \mu_i^2 = 0 \iff \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_i \boldsymbol{X} + \boldsymbol{b}_i^T \boldsymbol{X} + c_i = 0$$

Constraint on the minimal angle between two successive segments

To prevent thin tines (such as the one in red on the left of the inset) a constraint is implemented to ensure that the angle between two successive segments is greater than some threshold θ_{lim}. Such angle is given by $\theta = \arccos(-\boldsymbol{n}_i \cdot \boldsymbol{n}_{i+1})$ where $\boldsymbol{n}_j$ is the unit normal vector of segment $[\boldsymbol{p}_j, \boldsymbol{p}_{j+1}]$. Thus the constraint to implement is $\boldsymbol{n}_i \cdot \boldsymbol{n}_{i+1} + \cos\theta_{lim} \geq 0 \iff \boldsymbol{n}_i \cdot \boldsymbol{n}_{i+1} + \cos\theta_{lim} - \epsilon_i^2 = 0$ for some $\epsilon_i \in \mathbb{R}$. As the unit normal vectors are needed to express this constraint, several steps (related to the computation of the normal) are required before implementing it.



1. Constraint to compute a (non-unit) normal vector:
The two coordinates n_{ix}, n_{iy} of the normal vector $\boldsymbol{n}_i$ are introduced as variables. To constrain this vector to be orthogonal to the segment, one implements $\boldsymbol{n}_i \cdot (\boldsymbol{p}_{i+1} - \boldsymbol{p}_i) = 0 \Leftrightarrow -n_{ix}x_i - n_{iy}y_i + n_{ix}x_{i+1} + n_{iy}y_{i+1} = 0$. Let o be the index such that $\boldsymbol{X}[o+2i]$ and $\boldsymbol{X}[o+2i+1]$ map to n_{ix} and n_{iy}.

$$\boldsymbol{H}_i = \begin{pmatrix} 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{matrix} \leftarrow 2i \\ \leftarrow 2i+1 \\ \leftarrow 2i+2 \\ \leftarrow 2i+3 \\ \leftarrow o+2i \\ \leftarrow o+2i+1 \end{matrix} \qquad \boldsymbol{b}_i = \boldsymbol{0} \qquad c_i = 0$$

And:

$$\boldsymbol{n}_i \cdot (\boldsymbol{p}_{i+1} - \boldsymbol{p}_i) = 0 \iff \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_i \boldsymbol{X} + \boldsymbol{b}_i^T \boldsymbol{X} + c_i = 0$$

2. Constraint to consistently calculate the normal vector:
As the normal vector can be ambiguous (if $\boldsymbol{n}_i$ is a normal vector, then so is $-\boldsymbol{n}_i$), a constraint is implemented to consistently define the normal vector to be positively collinear with a $\frac{\pi}{2}$ counterclockwise rotation of the tangent vector $\boldsymbol{p}_{i+1} - \boldsymbol{p}_i$. Mathematically speaking, one wants $(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i) \times \boldsymbol{n}_i \geq 0$ i.e. $(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i) \times \boldsymbol{n}_i - \delta_i^2 = 0$ for some $\delta_i \in \mathbb{R}$. Let o_n be the index such that $\boldsymbol{X}[o_n + 2i]$ and $\boldsymbol{X}[o_n + 2i + 1]$ map to n_{ix} and n_{iy}, and o_d such that $\boldsymbol{X}[o_d + i]$ maps to δ_i.

$$\boldsymbol{H}_i = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 \end{pmatrix} \begin{matrix} \leftarrow 2i \\ \leftarrow 2i+1 \\ \leftarrow 2i+2 \\ \leftarrow 2i+3 \\ \leftarrow o_n+2i \\ \leftarrow o_n+2i+1 \\ \leftarrow o_d+i \end{matrix} \qquad \boldsymbol{b}_i = \boldsymbol{0} \qquad c_i = 0$$

And:

$$(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i) \times \boldsymbol{n}_i - \delta_i^2 = 0 \iff \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_i \boldsymbol{X} + \boldsymbol{b}_i^T \boldsymbol{X} + c_i = 0$$

3. Constraint to unitise the normal vector:
To calculate the angle between two segments, each normal vector must be of unit length, implemented as $||\boldsymbol{n}_i||^2 - 1 = 0$. Let o be the index such that $\boldsymbol{X}[o + 2i]$ and $\boldsymbol{X}[o + 2i + 1]$ map to n_{ix} and n_{iy}.

$$\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \begin{matrix} \leftarrow o + 2i \\ \leftarrow o + 2i + 1 \end{matrix} \qquad \boldsymbol{b}_i = \boldsymbol{0} \qquad c_i = -1$$

And:

$$||\boldsymbol{n}_i||^2 - 1 = 0 \iff \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_i \boldsymbol{X} + \boldsymbol{b}_i^T \boldsymbol{X} + c_i = 0$$

4. Constraint on the angle between two successive segments:
The constraint $\boldsymbol{n}_i \cdot \boldsymbol{n}_{i+1} + \cos\theta_{lim} - \epsilon_i^2 = 0$ can now be implemented. Let o_n be the index such that $\boldsymbol{X}[o_n + 2i]$, $\boldsymbol{X}[o_n + 2i + 1]$, $\boldsymbol{X}[o_n + 2i + 2]$ and $\boldsymbol{X}[o_n + 2i + 3]$ respectively map to n_{ix}, n_{iy}, n_{i+1x} and n_{i+1y}. Let also o_e be the index such that $\boldsymbol{X}[oe + i]$ maps to ϵ_i.

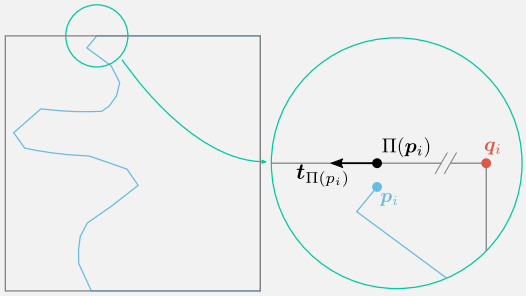
$$\boldsymbol{H}_i = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 \end{pmatrix} \begin{matrix} \leftarrow o_n + 2i \\ \leftarrow o_n + 2i + 1 \\ \leftarrow o_n + 2i + 2 \\ \leftarrow o_n + 2i + 3 \\ \leftarrow o_e + i \end{matrix} \qquad \boldsymbol{b}_i = \boldsymbol{0} \qquad c_i = \cos(\theta_{lim})$$

And:

$$\boldsymbol{n}_i \cdot \boldsymbol{n}_{i+1} + \cos\theta_{lim} - \epsilon_i^2 = 0 \iff \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_i \boldsymbol{X} + \boldsymbol{b}_i^T \boldsymbol{X} + c_i = 0$$

All the aforementioned constrained exhibit matrices $\boldsymbol{H}_i$, vectors $\boldsymbol{b}_i$ and scalars c_i which are constant: their values do not depend on the actual locations of points $\boldsymbol{p}_i$, $i \in [\![1, k]\!]$. They only depend on the ordering of the $\boldsymbol{p}_i$, i.e. on the topology of the polyline. As such, assuming the polyline to not change its topology, these constraints can be computed once at the beginning of the optimisation, and they are thus computationally cheap. They are referred to as **topological constraints**.
On the contrary, the following constraints depend on the locations of the polyline points, i.e. on the geometry of the polyline. Because the whole point of the optimisation is to move around the $\boldsymbol{p}_i$, their values change at each iteration and thus they have to be recalculated at each iteration, making them computationally more expensive. They are referred to as **geometrical constraints**.

Constraint requiring the endpoints of the polyline to slide on the edges of the design domain



Following [96], this constraint is implemented as $(\boldsymbol{p}_i - \boldsymbol{q}_i) \times \boldsymbol{t}_{\Pi(\boldsymbol{p}_i)} = 0$ where

- $\boldsymbol{p}_i$ refers to the endpoints of the polyline: $i \in \{1, k\}$.
- $\Pi(\boldsymbol{p}_i)$ refers to the projection of point $\boldsymbol{p}_i$ onto the edges of the design domain; it is the closest point to $\boldsymbol{p}_i$ on the design domain.
- $\boldsymbol{q}_i$ is one of the two points defining the segment of the design domain on which $\Pi(\boldsymbol{p}_i)$ is.

- $\boldsymbol{t}_{\Pi(p_i)} = (t_{ix}, t_{iy})^T$ refers to the tangent of the edge of the design domain on which $\boldsymbol{p_i}$ is projected through Π.

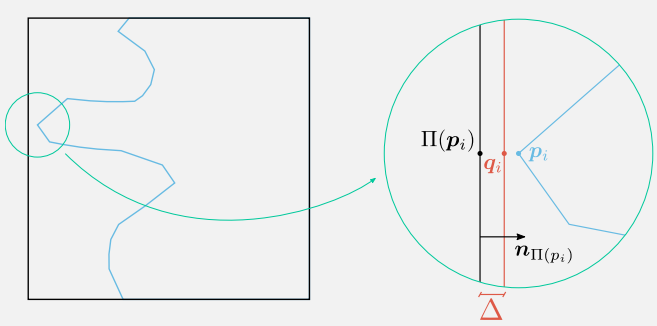
It constrains the vector $\boldsymbol{p}_i - \boldsymbol{q}_i$ to be collinear to the edge of the design domain, effectively pushing $\boldsymbol{p}_i$ to slide on the edge. To prevent numerical instabilities, among the two possible locations of point $\boldsymbol{q}_i$, it is selected as the one the farthest away from $\boldsymbol{p}_i$. For $i \in \{1, k\}$ let:

$$\boldsymbol{H}_i = \boldsymbol{0} \qquad \boldsymbol{b}_i = \begin{pmatrix} t_{iy} \\ -t_{ix} \end{pmatrix} \begin{matrix} \leftarrow 2i \\ \leftarrow 2i+1 \end{matrix} \qquad c_i = -\boldsymbol{q}_i \times \boldsymbol{t}_{\Pi(p)}$$

And:

$$(\boldsymbol{p}_i - \boldsymbol{q}_i) \times \boldsymbol{t}_{\Pi(p_i)} = 0 \iff \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_i \boldsymbol{X} + \boldsymbol{b}_i^T \boldsymbol{X} + c_i = 0$$

Constraint requiring the interior points to be inside the design domain, but preventing them from being too close to the edges of the design domain



To prevent bottleneck-like features, a constraint is implemented to ensure that each of the interior points of the polyline stays at a distance larger than some predefined Δ from the edges of the design domain. Also, to prevent interior points from wandering outside of the design domain, a constraint must be implemented to ensure they stay inside it. Fortunately, a single equation can handle both constraints. As shown on the inset, each interior point $\boldsymbol{p}_i$ is projected onto the edges of the design domain $\Pi(\boldsymbol{p}_i)$. The unit normal vector (pointing inside the design domain) of the edge containing $\Pi(\boldsymbol{p}_i)$ is called $\boldsymbol{n}_{\Pi(p_i)}$ of coordinates $(n_{\Pi(p_i)x}, n_{\Pi(p_i)y})^T$. Point $\Pi(\boldsymbol{p}_i)$ is translated to get $\boldsymbol{q}_i = \Pi(\boldsymbol{p}_i) + \Delta \boldsymbol{n}_{\Pi(p_i)}$. The constraint is then implemented as: $\boldsymbol{n}_{\Pi(p_i)} \cdot (\boldsymbol{p}_i - \boldsymbol{q}_i) \geq 0 \iff \boldsymbol{n}_{\Pi(p_i)} \cdot (\boldsymbol{p}_i - \boldsymbol{q}_i) - \zeta_i^2 = 0$ for some $\zeta_i \in \mathbb{R}$.
Let o be the index such that $\boldsymbol{X}[o+i]$ maps to ζ_i. For $i \in [\![2, k-2]\!]$, let:

$$\boldsymbol{H}_i = \begin{pmatrix} -2 \end{pmatrix} \leftarrow o+i \qquad \boldsymbol{b}_i = \begin{pmatrix} n_{\Pi(p_i)x} \\ n_{\Pi(p_i)y} \end{pmatrix} \begin{matrix} \leftarrow 2i \\ \leftarrow 2i+1 \end{matrix} \qquad c_i = -\boldsymbol{n}_{\Pi(p_i)} \cdot \boldsymbol{q}_i$$

And:

$$\boldsymbol{n}_{\Pi(p_i)} \cdot (\boldsymbol{p}_i - \boldsymbol{q}_i) - \zeta_i^2 = 0 \iff \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_i \boldsymbol{X} + \boldsymbol{b}_i^T \boldsymbol{X} + c_i = 0$$

Constraint requiring segments to snap

To ensure that the polyline obeys exactly the direction of translation $\boldsymbol{x}_t$, we saw in SECTION 4.1.1.3 that at least two segments must *snap*: the segments should have opposite orientation and their normal vectors should be orthogonal to $\boldsymbol{x}_t$. To model that using a quadratic equation $\phi_i(\boldsymbol{X}) = 0$, assume that the indices $\mathcal{I} \subset [\![1, k-1]\!]$ of the segments that must snap are given. Note that $\mathcal{I}$ contains at least two elements: $|\mathcal{I}| \geq 2$. SECTION 4.1.3.2 deals with the relevant strategies to compute such set $\mathcal{I}$. The constraint to implement is simply $\forall i \in \mathcal{I},\ \boldsymbol{n}_i \cdot \boldsymbol{x} = 0$. Let o be the index such that $\boldsymbol{X}[o+2i]$ and $\boldsymbol{X}[o+2i+1]$ map to the coordinates n_{ix} and n_{iy} of $\boldsymbol{n}_i$. For $i \in \mathcal{I}$, let:

$$\boldsymbol{H}_i = \boldsymbol{0} \qquad \boldsymbol{b}_i = \begin{pmatrix} x \\ y \end{pmatrix} \begin{matrix} \leftarrow o+2i \\ \leftarrow o+2i+1 \end{matrix} \qquad c_i = 0$$

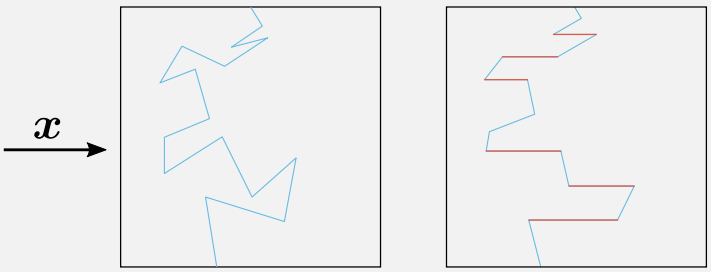


Figure 4.13| Input (left) and output (right) of ALGORITHM 1. Snap segments are highlighted in red.

And:

$$\boldsymbol{n}_i \cdot \boldsymbol{x} = 0 \Longleftrightarrow \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_i \boldsymbol{X} + \boldsymbol{b}_i^T \boldsymbol{X} + c_i = 0$$

Even though the coefficients in $\boldsymbol{b}_i$ are constant, this is a geometrical constraint as the set $\mathcal{I}$ depends on the geometry of the polyline.
Had we wanted the polyline to obey a cone bounded by $\boldsymbol{x}^A$ and $\boldsymbol{x}^B$, we would just have had to distinguish between $\mathcal{I}^A$ and $\mathcal{I}^B$, and nothing else would have changed.

Once all these topology and geometry constraints are taken into account, one can easily generate a 2-parts assembly obeying a translation. The algorithm is given below:

Algorithm 1: OptimisePolyline()

Input: inputPolyline: An ordered list of polyline points

Start:
$\boldsymbol{X}$ = ComputeVectorX(inputPolyline)
$\boldsymbol{H}_g, \boldsymbol{b}_g, c_g$ = ComputeGeometryConstraints($\boldsymbol{X}$)
$\boldsymbol{H}_t, \boldsymbol{b}_t, c_t$ = ComputeTopologyConstraints()
$\boldsymbol{H}_s, \boldsymbol{b}_s, c_s$ = Stack($\boldsymbol{H}_g, \boldsymbol{H}_t$), Stack($\boldsymbol{b}_g, \boldsymbol{b}_t$), Stack($c_g, c_t$)
for *epoch in range(MAXEPOCH)* **do**
    $\boldsymbol{X}$ = SolveForX($\boldsymbol{X}, \boldsymbol{H}_s, \boldsymbol{b}_s, c_s$) // Build matrix $\boldsymbol{H}$ and vector r, and solve $(\boldsymbol{H}^T\boldsymbol{H} + \epsilon^2\boldsymbol{I})\boldsymbol{X}_{n+1} = \boldsymbol{H}^t\boldsymbol{r} + \epsilon^2\boldsymbol{X}_n$
    $\boldsymbol{H}_g, \boldsymbol{b}_g, c_g$ = ComputeGeometryConstraints($\boldsymbol{X}$)
    $\boldsymbol{H}_s, \boldsymbol{b}_s, c_s$ = Stack($\boldsymbol{H}_g, \boldsymbol{H}_t$), Stack($\boldsymbol{b}_g, \boldsymbol{b}_t$), Stack($c_g, c_t$)
    residual = ComputeResidual($\boldsymbol{X}, \boldsymbol{H}_s, \boldsymbol{b}_s, c_s$) // Compute the norm of the vector obtained by stacking the $\phi_i(\boldsymbol{X})$
    if *residual < threshold* **then**
        break
optimisedPolyline = ExtractPolylineFromX($\boldsymbol{X}$)
return *optimisedPolyine*

FIGURE 4.13 shows the user-given polyline on the left, obviously not obeying the translation $\boldsymbol{x} = (1, 0)^T$, and the result of the optimisation on the right. One notices that the tine on the top has been opened to meet with the minimal angle constraint, the endpoints precisely are on the design domain, and all segments are well oriented with respect to the direction $\boldsymbol{x}$, some of them having snapped (in red).

4.1.2.3 On the generation of a 2-parts assembly obeying a rotation using the *Guided Projection Algorithm*

Let $\boldsymbol{x}_r = (x, y)^T \in \mathbb{R}^2$ be a centre of rotation. This section aims to formulate the constraints ϕ_i so that a polyline is optimised to obey a rotation around $\boldsymbol{x}_r$. As stated by SYSTEM (3.12), a polyline $(\boldsymbol{p}_i)$, $i \in [\![1, k]\!]$

obeys a rotation around $\boldsymbol{x}_r$ if and only if:

$$\forall i \in [\![1, k-1]\!], \begin{cases} s\boldsymbol{m}(\boldsymbol{p}_i, \boldsymbol{x}_r) \cdot \boldsymbol{n}_i \geq 0 \\ s\boldsymbol{m}(\boldsymbol{p}_{i+1}, \boldsymbol{x}_r) \cdot \boldsymbol{n}_i \geq 0 \end{cases} \tag{4.10}$$

Where $s = \pm 1$ stipulates whether the rotation is counterclockwise or clockwise, as understood in SECTION 4.1.1.5. The task at hand is straightforward: a first constraint shall calculate the instantaneous directions of motion $\boldsymbol{m}_{p_i}$ while a second constraint ensure that SYSTEM (4.10) is met.

Constraint to calculate the instantaneous direction of motion $\boldsymbol{m}_{p_i}$

Recall that $\boldsymbol{m}_{p_i} = \begin{pmatrix} y - y_i \\ x_i - x \end{pmatrix}$. By introducing the coordinates (m_{ix}, m_{iy}) of vector $\boldsymbol{m}_{p_i}$, one needs to implement two equations: $m_{ix} - y + y_i = 0$ and $m_{iy} - x_i + x = 0$. Let o be the index such that $\boldsymbol{X}[o + 2i]$ and $\boldsymbol{X}[o + 2i + 1]$ map to m_{ix} and m_{iy}. For $i \in [\![1, k-1]\!]$

$$\boldsymbol{H}_{2i} = \boldsymbol{0} \qquad \boldsymbol{b}_{2i} = \begin{pmatrix} 1 \leftarrow o + 2i \\ 1 \leftarrow 2i + 1 \end{pmatrix} \qquad c_{2i} = -y$$

$$\boldsymbol{H}_{2i+1} = \boldsymbol{0} \qquad \boldsymbol{b}_{2i+1} = \begin{pmatrix} 1 \leftarrow o + 2i + 1 \\ -1 \leftarrow 2i \end{pmatrix} \qquad c_{2i+1} = x$$

And thus:

$$m_{ix} - y + y_i = 0 \iff \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_{2i} \boldsymbol{X} + \boldsymbol{b}_{2i}^T \boldsymbol{X} + c_{2i} = 0$$
$$m_{iy} - x_i + x = 0 \iff \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_{2i+1} \boldsymbol{X} + \boldsymbol{b}_{2i+1}^T \boldsymbol{X} + c_{2i+1} = 0$$

Note that this is a topological constraint.

Constraint requiring the polyline to obey a rotation

The constraints to implement are $s\boldsymbol{m}_{p_j} \cdot \boldsymbol{n}_i \geq 0 \iff s m_{jx} n_{ix} + s m_{jy} n_{iy} - \eta_{2i}^2 = 0$ for $j \in \{i, i+1\}$. Let o_e be the index such that $\boldsymbol{X}[o_e + 2i]$ and $\boldsymbol{X}[o_e + 2i + 1]$ map to η_{2i} and η_{2i+1}. Let also o_m be the index such that $\boldsymbol{X}[o_m + 2i]$ and $\boldsymbol{X}[o_m + 2i + 1]$ map to m_{ix} and m_{iy}. Similarly, let o_n be the index such that $\boldsymbol{X}[o_n + 2i]$

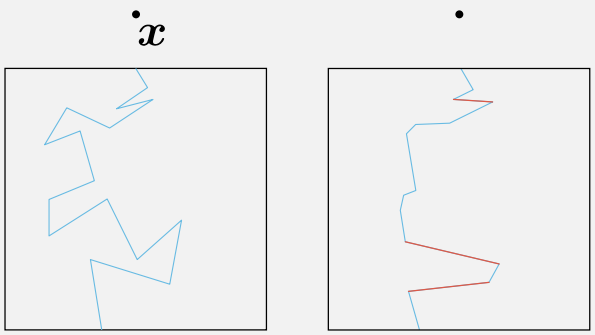


Figure 4.14| Input (left) and output (right) of ALGORITHM 1 adapted to work in rotation. Snap segments are highlighted in red.

and $\boldsymbol{X}[o_n + 2i + 1]$ map to n_{ix} and n_{iy}. For $i \in [\![1, k-1]\!]$:

$$\boldsymbol{H}_{2i} = \begin{pmatrix} 0 & 0 & s & 0 & 0 \\ 0 & 0 & 0 & s & 0 \\ s & 0 & 0 & 0 & 0 \\ 0 & s & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 \end{pmatrix} \begin{matrix} \leftarrow o_n + 2i \\ \leftarrow o_n + 2i + 1 \\ \leftarrow o_m + 2i \\ \leftarrow o_m + 2i + 1 \\ \leftarrow o_e + 2i \end{matrix} \qquad \boldsymbol{b}_{2i} = \boldsymbol{0} \qquad c_{2i} = 0$$

$$\boldsymbol{H}_{2i+1} = \begin{pmatrix} 0 & 0 & s & 0 & 0 \\ 0 & 0 & 0 & s & 0 \\ s & 0 & 0 & 0 & 0 \\ 0 & s & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 \end{pmatrix} \begin{matrix} \leftarrow o_n + 2i + 2 \\ \leftarrow o_n + 2i + 3 \\ \leftarrow o_m + 2i \\ \leftarrow o_m + 2i + 1 \\ \leftarrow o_e + 2i \end{matrix} \qquad \boldsymbol{b}_{2i+1} = \boldsymbol{0} \qquad c_{2i+1} = 0$$

And:

$$s\boldsymbol{m}_{p_i} \cdot \boldsymbol{n}_i - \eta_{2i}^2 = 0 \iff \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_{2i} \boldsymbol{X} + \boldsymbol{b}_{2i}^T \boldsymbol{X} + c_{2i} = 0$$
$$s\boldsymbol{m}_{p_i} \cdot \boldsymbol{n}_i - \eta_{2i+1}^2 = 0 \iff \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_{2i+1} \boldsymbol{X} + \boldsymbol{b}_{2i+1}^T \boldsymbol{X} + c_{2i+1} = 0$$

It is also a topological constraint.

Constraint requiring segments to snap

Very similar to the case in translation: given a set of indices $\mathcal{I} \subset [\![1, k-1]\!]$, the constraint is implemented as $\boldsymbol{m}_{p_i} \cdot \boldsymbol{n}_i = 0$, for $i \in \mathcal{I}$. Hence, let o_m be the index such that $\boldsymbol{X}[o_m + 2i]$ and $\boldsymbol{X}[o_m + 2i + 1]$ map to m_{ix} and m_{iy}; let o_n be the index such that $\boldsymbol{X}[o_n + 2i]$ and $\boldsymbol{X}[o_n + 2i + 1]$ map to n_{ix} and n_{iy}:

$$\boldsymbol{H}_i = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{matrix} \leftarrow o_n + 2i \\ \leftarrow o_n + 2i + 1 \\ \leftarrow o_m + 2i \\ \leftarrow o_m + 2i + 1 \end{matrix} \qquad \boldsymbol{b}_i = \boldsymbol{0} \qquad c_i = 0$$

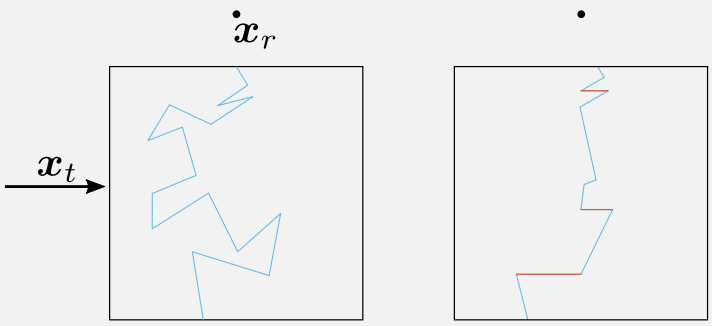


Figure 4.15| Input (left) and output (right) of ALGORITHM 1 adapted to work in rotation and in translation. Snap segments are highlighted in red.

And

$$\boldsymbol{m}_{p_i} \cdot \boldsymbol{n}_i = 0 \iff \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_i \boldsymbol{X} + \boldsymbol{b}_i^T \boldsymbol{X} + c_i = 0$$

FIGURE 4.14 shows the input/output pair of the GPA where the aforementioned constraints were added to make it work in rotation.

4.1.2.4 On the generation of a polyline obeying both a translation and a rotation

Should the user want the polyline to obey both a translation along $\boldsymbol{x}_t \in \mathcal{S}^1$ and a rotation around $\boldsymbol{x}_r \in \mathbb{R}^2$, then he/she should only implement both constraints, and the GPA optimises the polyline to meet this goal. FIGURE 4.15 shows the input/output pair of the GPA adapted to work both in translation and rotation. In red are the snap segments. Note that the algorithm has moved one of the endpoint of each snap segment *exactly* underneath the centre of rotation $\boldsymbol{x}_r = (0, 1)^T$ (the design domain being centered at $\mathbf{0}$). As such, for these points, the instantaneous direction of motion $\boldsymbol{m}_{p_i}$ is collinear with the horizontal axis, which happens to be the prescribed direction of translation $\boldsymbol{x}_t = (1, 0)^T$. Thus these three segments, being aligned with $\boldsymbol{x}_t$, simultaneously snap in translation as $\boldsymbol{n}_i \cdot \boldsymbol{x}_t = 0$, and in rotation as $\boldsymbol{m}_{p_i} \propto \boldsymbol{x}_t \implies \boldsymbol{n}_i \cdot \boldsymbol{m}_{p_i} = 0$.

A word of caution: special care should be taken by the user when choosing the direction of translation $\boldsymbol{x}_t$ and the centre of rotation $\boldsymbol{x}_r$: one cannot ask for a polyline to simultaneously obey a rotation that would lift it up while obeying a translation pointing downwards.

4.1.2.5 Results

FIGURE 4.16 shows a typical convergence of a GPA optimisation. Let $\boldsymbol{\phi}(\boldsymbol{X}_n) = (\phi_1(\boldsymbol{X}_n), \ldots, \phi_i(\boldsymbol{X}_n), \ldots)^T$ be the vector obtained by stacking the residuals $\phi_i(\boldsymbol{X}_n)$ at each iteration n. FIGURE 4.16 displays the graph of the evolution of the logarithm of the euclidean norm of the residual vector $\boldsymbol{\phi}$, i.e. $\log ||\boldsymbol{\phi}||$. It shows that whenever the residual falls below a given threshold ($\simeq 10^{-11}$ here) either a new constraint is turned on or the final solution is reached. The highlighted red points correspond to iterations where a new constraint was switched on: at first, the constraint requiring the endpoints of the polyline to slide on the edges of the design domain, and after two iterations the snap constraint.

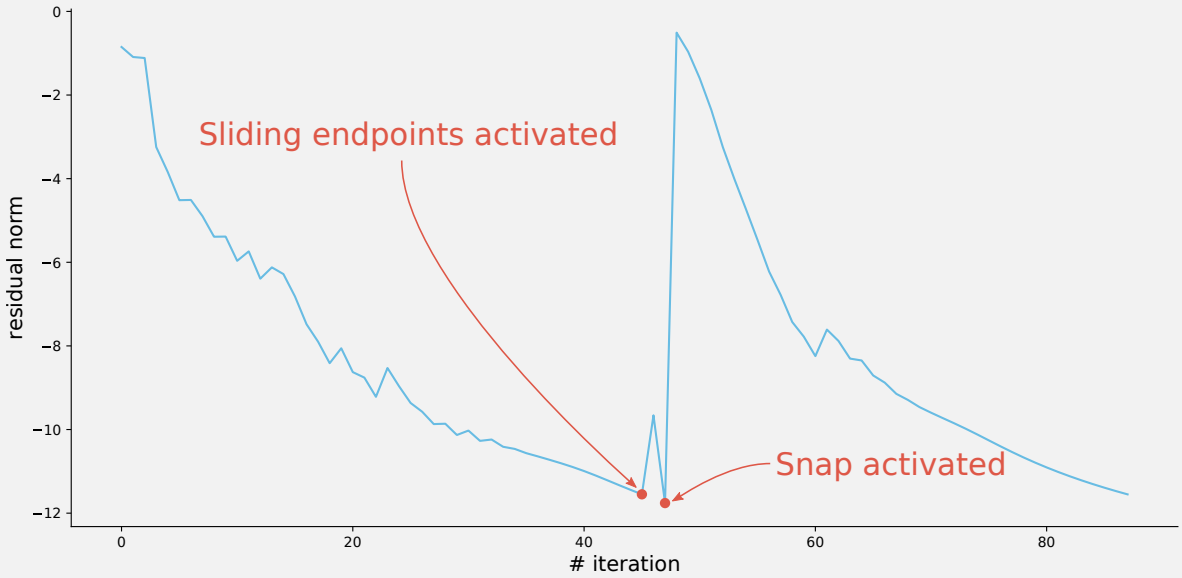


Figure 4.16| Evolution of the norm of the residual (log y-scale).

We can also display the residuals of each constraint (in blue on FIGURE 4.17), as well as the associated weight (in red). While Tang and colleagues [96] talk about smoothly varying weights, here we adopt a more brutal approach where the weights are binary, in $\{0, 1\}$. These graphs clearly show when the sliding endpoints and snap constraints are activated. It is also noticeable that the length constraint (weight constantly equal to 1) was met throughout the execution with a 0 residual. It is also the case for the "assemblability-Translation" (stating that $\boldsymbol{n}_i \cdot \boldsymbol{x}_t \geq 0$) but here it is simply because we optimised the polyline to obey a pure rotation, and thus this weight has been constantly set to 0. On FIGURE 4.17 some constraints ("repelSegments" for instance) are shown but are not described in this dissertation as they are of minor importance.

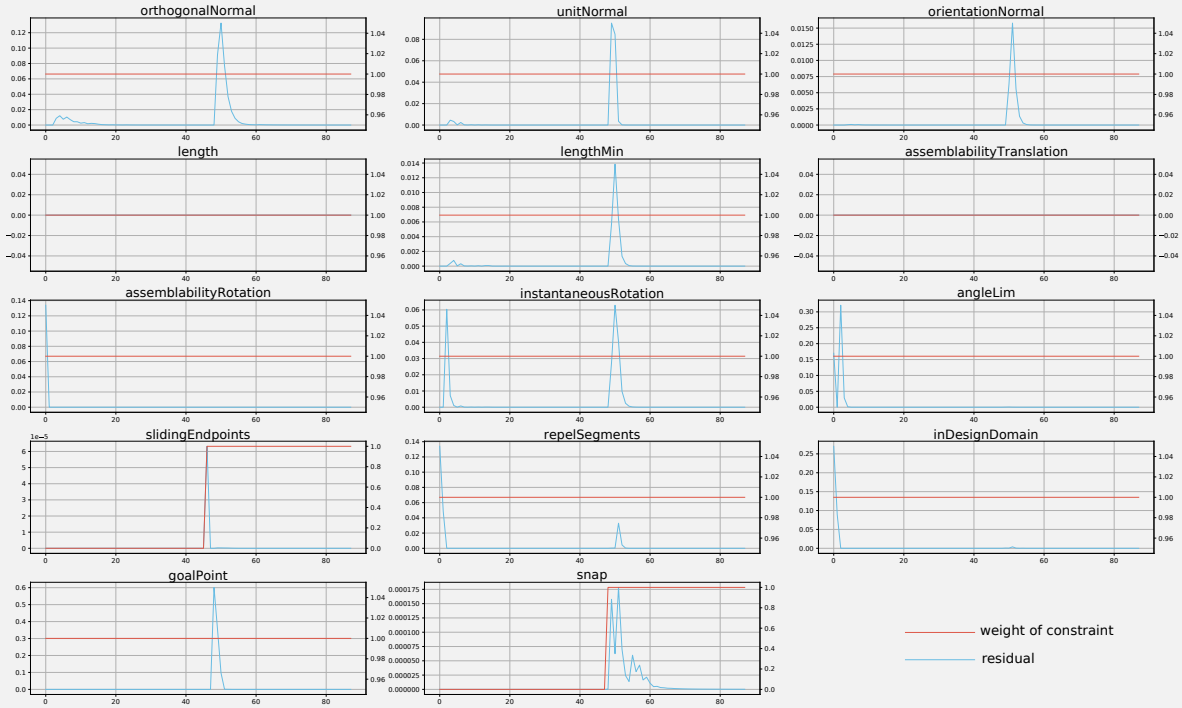


Figure 4.17| Evolution of the norm of the residuals for each constraint (left y-scale) and the associated weights (right y-scale).

4.1.2.6 Using the optimised separating polyline to partition the design domain into two parts

The last step of creating a 2-parts assembly is to actually create the parts. The goal of this section is to explain how given a polyline and a design domain, the latter can be partitioned into two parts P_0 and P_1, with P_1 obeying the prescribed motion.

Before the GPA optimisation is run, the input polyline is oriented so that its normal vectors point towards what will be the part to move P_1. In layman's words, given the ordering of the point $(\boldsymbol{p}_i)_{1\leq i\leq k}$, we must check whether this ordering leads to the normal vectors being somewhat in the direction of translation (and thus pointing towards P_1). If not, then the reversed ordering $(\boldsymbol{p}_{k-i+1})$ shall be considered.

Translation

Should the optimised polyline obey a translation $\boldsymbol{x}_t \in \mathcal{S}^1$, then to check whether the input polyline is well oriented is straightforward: the algorithm loops over each initial segment $[\boldsymbol{p}_i, \boldsymbol{p}_{i+1}]$ and compute its normal vector $\boldsymbol{n}_i$ by rotating its unit tangent by $\frac{\pi}{2}$ counterclockwise. Depending on the sign of $\boldsymbol{n}_i \cdot \boldsymbol{x}_t$, a positive or negative counter is incremented. At the end of the loop a majority rule is adopted: if the positive counter is greater than the negative, then the ordering $(\boldsymbol{p}_i)$ is kept. Otherwise the polyline is flipped and the points are ordered $(\boldsymbol{p}_{k-i+1})$. This ensures that most of the segments have their normal such that $\boldsymbol{n}_i \cdot \boldsymbol{x}_t \geq 0$.

Rotation

Should the polyline obey a rotation around $\boldsymbol{x}_r \in \mathbb{R}^2$, then a simple trick is implemented: the rotation shall be approximated by a translation, to return to the previous case. Let $\boldsymbol{o}$ denote the centroid of the design domain. If $s = +1$ (counterclockwise rotation), then the centre of rotation is projected at infinity along vector $\boldsymbol{x}_r - \boldsymbol{o}$. Indeed, the further away is $\boldsymbol{x}_r$ from $\boldsymbol{o}$ the more collinear are the instantaneous directions of motion $\boldsymbol{m}_{p_i}$. At the limit they all are orthogonal to $\boldsymbol{x}_r - \boldsymbol{o}$. This is illustrated on FIGURE 4.18: the square is centered at the origin and, on the left, $\boldsymbol{x}_r = (0, 1)^T$ and the instantaneous direction of motion $\boldsymbol{m}_{p_i}$ are depicted using red arrows. In the middle $\boldsymbol{x}_r$ (not shown) is moved vertically by a finite amount (to $(0, 3)$); it can already be noticed that the $\boldsymbol{m}_{p_i}$ are more aligned with each other. At the limit where $\boldsymbol{x}_r$ is moved upwards towards infinity, all $\boldsymbol{m}_{p_i}$ are collinear with the direction $\boldsymbol{x}_t = (1, 0)^T$. Effectively, in such a case, the polyline is tasked to obey the translation $\boldsymbol{x}_t$. Had s been negative, $\boldsymbol{x}_r$ would have been moved to infinity using the opposite vector $-(\boldsymbol{x}_r - \boldsymbol{o})$.
This trick gives us a direction of translation $\boldsymbol{x}_t$ such that the instantaneous direction of motion (collinear to $\boldsymbol{x}_t$) are somewhat close to $\boldsymbol{m}(\boldsymbol{p}_i, \boldsymbol{x}_r)$. The ordering of the points $(\boldsymbol{p}_i)_{1\leq i\leq k}$ is kept or flipped according to the aforementioned algorithm in translation.

The polyline now being well oriented, it is optimised using the GPA. The design domain is ordered in a counterclockwise fashion so that the normal vectors of its edges point inside it. At the end of the optimisation, the edge of the design domain on which the last point $\boldsymbol{p}_k$ of the polyline is recorded. Starting from the endpoint of that edge, the vertices of the design domain are added in a list in order until arriving at the edge where the start point $\boldsymbol{p}_1$ is. There the polyline points $\boldsymbol{p}_1, ..., \boldsymbol{p}_k$ are added in that order. This list contains all the points of the part P_1.
To create part P_0 the same algorithm is used but the ordering of the polyline vertices is flipped. FIGURE 4.19 illustrate the process of creating part P_1. The polyline is oriented from $\boldsymbol{p}_1$ to $\boldsymbol{p}_k$, as suggested by the blue arrow, and the design domain from $\boldsymbol{v}_1$ to $\boldsymbol{v}_4$, see the black arrow. The last point $\boldsymbol{p}_k$ lies on edge $[\boldsymbol{v}_1, \boldsymbol{v}_2]$ of the design domain. The endvertex $\boldsymbol{v}_2$ is added to the list. Secondly the end vertex $\boldsymbol{v}_3$ of the following edge $[\boldsymbol{v}_2, \boldsymbol{v}_3]$ is added. As the first point $\boldsymbol{p}_1$ lies on edge $[\boldsymbol{v}_3, \boldsymbol{v}_4]$, the whole polyline $\boldsymbol{p}_1, ..., \boldsymbol{p}_k$ is added to the list. Thus part P_1 is made of points $\boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{p}_1, ..., \boldsymbol{p}_k$. To create part P_0 the same process happens by considering

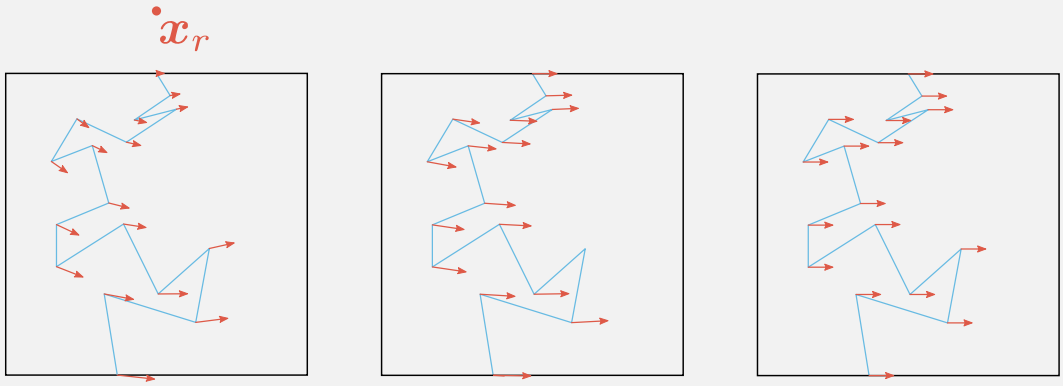


Figure 4.18| The further away $\boldsymbol{x}_r$ from the design domain, the more aligned are the instantaneous directions of motion $\boldsymbol{m}_{p_i}$.

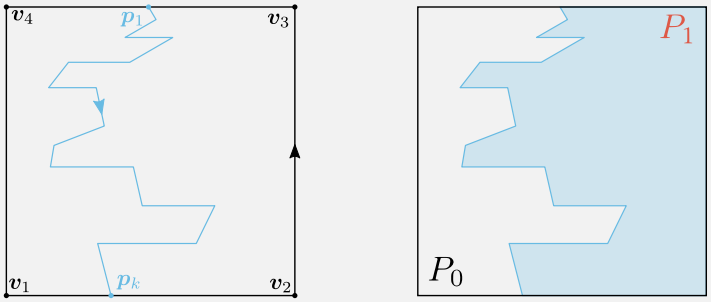


Figure 4.19| By carefully ordering the polyline points and the design domain vertices, parts P_0 and P_1 can easily be built.

the flipped polyline $\boldsymbol{p}_k, ..., \boldsymbol{p}_1$.

The first task is now complete! We have now understood how to create a 2-parts assembly $A = \{P_0, P_1\}$, such that P_1, thanks to the snap segments, obeys precisely a given direction of translation $\boldsymbol{x}_t \in \mathcal{S}^1$ (or, with little work, a cone bounded by two vectors $\boldsymbol{x}_t^A$ and $\boldsymbol{x}_t^B$), and/or a rotation around a centre $\boldsymbol{x}_r \in \mathbb{R}^2$. We can get on to the next task consisting in generating a N+1-parts assembly, with $N > 1$.

4.1.3 ON THE CREATION OF THE FOLLOWING PARTS

The process to create each of the following parts $P_i, i > 1$ is the same as the one used to create a 2-parts assembly described in SECTION 4.1.2. What is part P_0 at the end of iteration i becomes the design domain at the beginning of iteration $i + 1$. Thus, 2-parts assemblies keep being created in a design domain that gets smaller and smaller, this process is illustrated on FIGURE 4.20 for a 3+1 parts assembly. As such, creating a N+1-parts assembly is straightforward. The difficulty lies in creating an assembly that is interlocked, *i.e.*

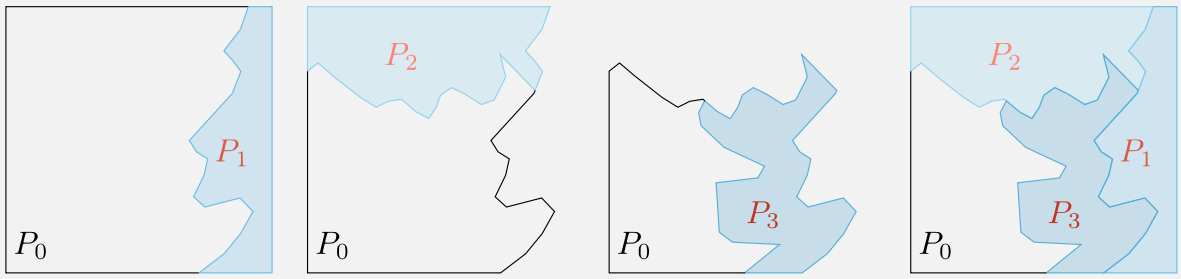


Figure 4.20| Each part P_i is created in what was P_0 at the previous iteration.

whose sole part that can be removed without colliding with any other is the key P_1.

4.1.3.1 Assessing the interlocking of an assembly

As explained on SECTION 3.1.3, the Non Directional Blocking Graph (NDBG) of an assembly encodes all the information necessary to assess the blocking relationships between its constitutive parts. Thus, to ensure that the assembly is interlocked, at the end of the creation of each part P_i, $i > 1$, the NDBG is calculated: the cones of freedom in translation $\mathcal{C}_t$ and counterclockwise and clockwise rotations $\mathcal{C}_r^{ccw}$ and $\mathcal{C}_r^{cw}$ of each created part are calculated. The vectors bounding the cones of translational freedom $\mathcal{C}_t$ define the direction of translation of the base DBGs in translation. The cells of dimension 0 bounding (or resulting from the intersection of) the cones of rotational freedom $\mathcal{C}_r^{ccw}$ and $\mathcal{C}_r^{cw}$ constitute the centres of translation of the base DBGs in rotation. For each such identified motion, the corresponding base DBG is calculated, the computation being summed up in SECTION 3.1.2.2. If the NDBG says that all base DBGs are strongly connected (but the one(s) associated to P_1 that must have two strongly connected components, one being reduced to P_1 the other being the other parts) then the assembly consisting of parts $\{P_0, ..., P_i\}$ is interlocked and the creation of the next part can start. Else, the assembly is not interlocked and the next iterations will not make it interlocked. Indeed, in this case, some other part(s) than the key can move and (since the successive parts will be created inside the current P_0) the further partitioning of P_0 will not change that fact. Thus, the latest polyline is deemed invalid and must be optimised again from scratch, using a different input polyline.

Therefore, if a polyline is invalid and the optimisation must be run again, the input polyline must change (otherwise the optimisation will deliver the same result). A first possibility is to ask the user to manually input another polyline. A second option is to automatically generate a polyline using, for instance, the couple `Turtle` and Markov process: the `Turtle` draws an initial polyline that is then optimised using GPA. To avoid falling into the pitfalls listed at the beginning of SECTION 4.1.2, the `Turtle` shall only generate a part obeying a translation; if a rotation is imposed by the user, then the `Turtle` draws a polyline obeying the pseudo-translation obtained by moving the centre of rotation to infinity, along the same process as the one described SECTION 4.1.2.6. This first instance of a solution is, in general, good enough for the GPA to take over. Moreover, the user can specify goal points for the endpoints of the polyline: $\boldsymbol{p}_1$ and $\boldsymbol{p}_k$ should be within a given radius from two specified points on the edge of the design domain. Such a constraint must, of course, be implemented within the GPA framework. If this choice is made, it gives back to the user more control over the polyline.
The outline of the algorithm is as follows:

1. Take the part P_0 obtained at the end of the previous iteration, and consider it to be the design domain.
2. Either the user inputs a polyline or the `Turtle` is tasked with drawing one. In either case, an input polyline is available. Goal points can optionally be specified.
3. Run the GPA to optimise the polyline and compute the newest part P_i as well as the remaining part P_0.
4. Compute the NDBG associated with the current assembly. If it says the assembly is indeed interlocked, then this iteration is finished. Else, go back to STEP 2.

This outline is illustrated on FIGURE 4.21 for the creation of an assembly of 4+1 parts; parts P_i, $i \in 1, 2, 3$ obey a rotation around $\boldsymbol{x}_r^i$ and part P_4 obeys a translation along $\boldsymbol{x}_4^t$. The top two rows show the input and the successive iterations, as well as the fact that the NDBG is calculated from iteration 2, and as long as it is not valid, the problematic part keeps being recreated. In the end, the design domain is partitioned into 5 parts, and the NDBG, on the bottom row, show that all base DBGs are strongly connected, except for the DBG associated with the key which states that P_1 is free to rotate around $\boldsymbol{x}_r^1$. Cones $\mathcal{C}_t$, $\mathcal{C}_r^{ccw}$ and $\mathcal{C}_r^{cw}$ have been calculated and were all reduced to the single $\boldsymbol{x}_r^i$ ($i \in \{1, 2, 3\}$) or $\boldsymbol{x}_t^4$, thus proving that the four DBGs

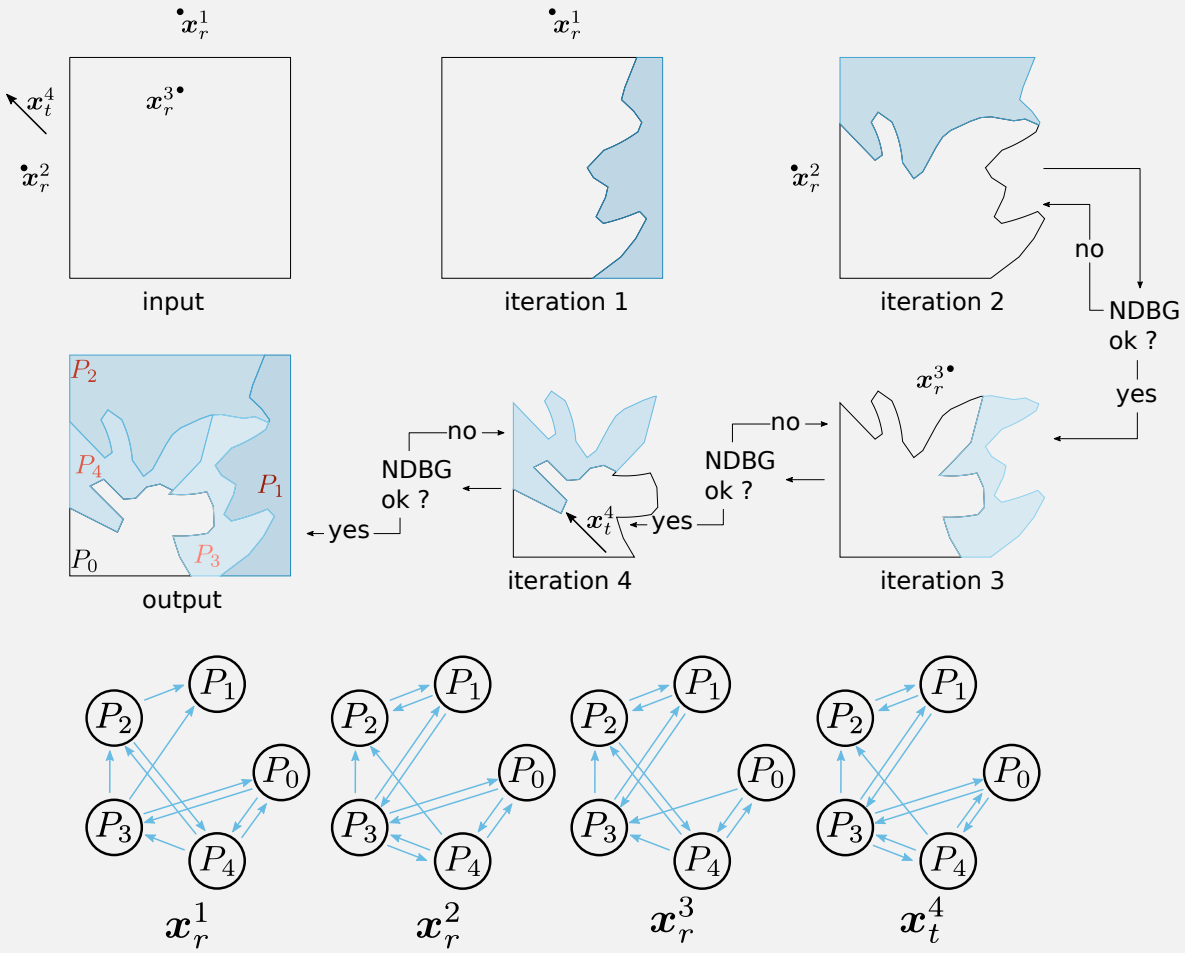


Figure 4.21| Workflow of the generation of an assembly made of N+1 parts.

on the figure are indeed the base DBGs of the assembly.

4.1.3.2 Further details and trick to increase the odd of an assembly to be interlocked

An important detail concerning the implementation of the GPA is that not all constraints ϕ_i are used simultaneously: as some are conflicting with each other, the GPA would sometimes get stuck, oscillating between solutions that alternatively increase and decrease a given constraint. For instance, the goal points constraint and the constraint requiring the endpoints to slide on the edges of the design domain were found to be often at odd with each other: the former, trying to minimise the distance between the endpoints and the goal points, would try to make the endpoints move directly towards the goal points; the latter would force the endpoints to follow the direction of the edges, leading to poor results (often slow convergence, sometimes no convergence at all) when the edges are not oriented towards the goal points. To prevent conflicting situations like this, an on-off iterative approach is used, where constraints are alternatively switched on and off, to aid the convergence (at the end, of course, all constraints are switched on). In our example, the sliding-endpoints constraint is first switched off while the goal points constraint is on: this leads the endpoints of the polyline to move quickly close to the goal points without taking into account the design domain edges. Once they are close enough, the sliding-endpoints constraint is switched on, resulting in the endpoints being projected onto the edges of the design domain, close to the goal. This on-off approach helps the solver find solutions faster.

In practice, all constraints are at first on, except the sliding-endpoints and snap ones. When a first stable solution is found, the sliding-endpoints constraint is activated. Once a new solution is found, the snap constraint is switched on, until the final convergence of the algorithm.

There are numerous ways to choose the snap segments, but we found two, in particular, to be well-suited to the generation of an assembly:

1. The user specifies that $n \geq 2$ segments must snap. When the snap constraint is activated then a list $\mathcal{L}$ of snap segments is initialised and:

- if the polyline must obey a translation: its segments are ranked by how close they already are from their snap position, *i.e.* how close their normal is from being orthogonal to the prescribed direction of translation $\boldsymbol{x}_t$ (or one of the two directions $\boldsymbol{x}_t^A$ and $\boldsymbol{x}_t^B$ bounding the prescribed cone). Let $\boldsymbol{t}_i$ denotes the unit tangent of segment $[\boldsymbol{p}_i, \boldsymbol{p}_{i+1}]$ (oriented from $\boldsymbol{p}_i$ to $\boldsymbol{p}_{i+1}$); the larger $|\boldsymbol{t}_i \cdot \boldsymbol{x}_t|$ the closer is the segment from its snap position. If $|\boldsymbol{t}_i \cdot \boldsymbol{x}_t| = 1$ the segment has already snapped. These dot products are sorted and the two segments with the lowest (closest to -1) and highest (closest to 1) values are added to the list $\mathcal{L}$. It ensures that at least two segments snap opposite to each other: one snaps to be oriented along $+\boldsymbol{x}_t$ (or $\pm\boldsymbol{x}_t^A$) and the other snaps to be along $-\boldsymbol{x}_t$ (or $\mp\boldsymbol{x}_t^B$), a necessary and sufficient condition for the assembly to obey $\boldsymbol{x}$ (or the cone $\boldsymbol{x}_t^A$ and $\boldsymbol{x}_t^B$) as seen in SECTION 4.1.1.3 and illustrated on FIGURE 3.4. The $n-2$ following snap segments are added to $\mathcal{L}$ according to how close to 1 is $|\boldsymbol{t}_i \cdot \boldsymbol{n}_i|$, without considering a positive or negative orientation.
- if the polyline must obey a rotation around a centre $\boldsymbol{x}_r \in \mathbb{R}^2$: the algorithm is similar to the translation case. The segments $[\boldsymbol{p}_i, \boldsymbol{p}_{i+1}]$ of the polyline are ranked according to how orthogonal their normal vector is to the instantaneous directions of motion $\boldsymbol{m}_{p_i}, \boldsymbol{m}_{p_{i+1}}$. Let $\boldsymbol{t}_i$ denotes the unit tangent of segment $[\boldsymbol{p}_i, \boldsymbol{p}_{i+1}]$ (oriented from $\boldsymbol{p}_i$ to $\boldsymbol{p}_{i+1}$) and let j refers to both i and $i+1$; the larger $|\boldsymbol{t}_i \cdot \boldsymbol{m}_{p_j}|$ the closer is the segment from its snap position. If $|\boldsymbol{t}_i \cdot \boldsymbol{m}_{p_j}| = 1$ the segment has already snapped. Then again, the segments corresponding to the smallest (close to -1) and highest (close to 1) dot products are added to $\mathcal{L}$. The list is completed with the $n-2$ segments with the largest $|\boldsymbol{t}_i \cdot \boldsymbol{m}_{p_j}|$.

2. The second method is a trick to increase the odds of interlocking. While the first method works well for high values of n, for low values of n it may lead to designing parts that are hard to interlock with successive ones. For instance, on FIGURE 4.22, for $n = 2$, it is absolutely impossible to create an 2+1 parts interlocking assembly on the left: the geometry of P_1 makes it impossible for P_2 to be blocked. Even if the polyline separating P_2 from P_0 started in the small recess at the bottom of the boundary of P_1, one could still lift P_1 and P_2 up simultaneously, making the assembly not interlocked. On the right, the polyline bounding P_1 snapped in the opposite directions. Unless the polyline of P_2 starts at the top or bottom vertical segments (small probability, assuming it may start uniformly anywhere), the assembly will always be interlocked. Therefore, FIGURE 4.22 illustrate the importance of the directions of the snaps for low value of n. This second method simply ensures that exactly two snaps are made, in the directions that increase the odds of the interlocking. To select the two segments that should go into the list $\mathcal{L}$, the segments are weighted according to how close to the extremities of the polyline they are (*i.e.* how close their index is close to 2 or $k-2$, k being the number of polyline points) and how close to a snap in the correct direction they already are (similar to the first method). Segments 1 and $k-1$ are left out as snapping them is often undesirable as it may lead a segment to be contained in an edge of the design domain, as would be the case in the figure. By modulating the importance given to the weights, one reaches the full spectrum between the first method for $n = 2$ and the certain snapping of segments 2 and $k-2$ in the directions that maximise the interlocking (as P_1 on the right of FIGURE 4.22).

This second method becomes useless if the user manually inputs a polyline and makes sure that two segments are already close to a snap in the correct directions at the correct positions on the polyline.

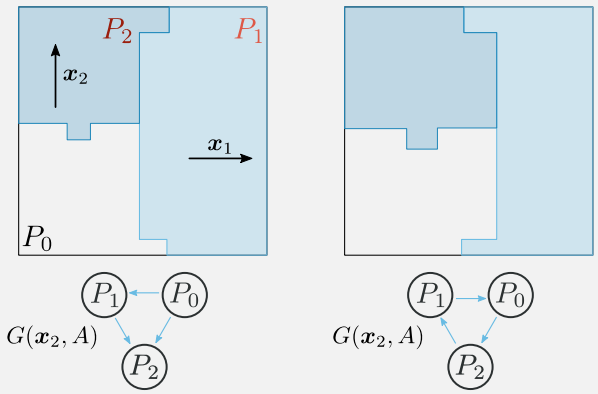


Figure 4.22| On the left the assembly is not interlocked. A small change in the geometry of P_1 leads to an interlocking assembly on the right.

Still, it proved to be useful when the input polyline was generated by the `Turtle` .

4.1.3.3 Conclusion

The task of generating an interlocking assembly made of $N+1$ parts obeying translation and/or rotation is surprisingly fast once one knows how to generate a 2-parts assembly. As understood, the same algorithm (generating a 2-parts assembly) is run on the successive remaining parts labelled as P_0. At each iteration $i > 1$, the remaining part becomes the design domain and is partitioned into a P_i and a new P_0. Following this step, the NDBG is calculated. If it says that the assembly is not interlocked, the latest optimised polyline is discarded and either the user or the `Turtle` provides a new input for the GPA. If the assembly is interlocked, then P_0 becomes the design domain at the next iteration $i+1$.

4.2 MECHANICAL PROPERTIES OF A GENERATED ASSEMBLY

4.2.1 LITERATURE REVIEW

While this manuscript is based on the observation that new manufacturing technologies can fabricate assemblies with geometrical features inaccessible to more traditional methods, and therefore it becomes interesting to explore the space of interlocking assemblies, we have to show that the generated assemblies, with their strange-looking features, are relevant from a mechanical perspective. The literature on the mechanical analysis of self-fitting joints is plentiful but mostly focuses on indirect assemblies using dowels (or similar intermediary bodies). Seldom are the papers focusing on self-fitting carpentry joints. For instance, Parisi *et al.*, [75], studied a specific joint, used to build roofs around the Mediterranean and the Alps, the birdsmouth joint. They provide an experimental and numerical analysis of such joints and compare options to retrofit them. In exactly the same vein, Branco *et al.* [15] further studies this joint to evaluate different strengthening techniques. In [14], Branco and coauthors sum up the technique consisting of modelling joints as equivalent springs to perform a semi-rigid analysis and give various recommendations regarding the design and reinforcement of various traditional joints (tenon, lap, scarf). Quite recently, Braun *et al.*, [16], build a numerical model performing linear (thus fast) analysis of single- and double-step joints under compression. The model is calibrated using experimental data. Moradei *et al.*, [64], also used numerical models to assess the behaviour of traditional Chinese and Japanese assemblies subjected to bending moments. What the literature lacks is a unified approach to model joint geometries, but as put by Branco "*because of the wide variety of carpentry joint geometries in existence, studying them with an exhaustive approach is [not] realistic*". Indeed, all the aforementioned papers used the geometrical features of each joint they focused on to propose simplifying assumptions or to calibrate their model. Such an approach is entirely out of reach

in this PhD as the goal is to automatically generate new joints, and thus no two joints are alike. It would be impossible to dedicate enough time to precisely model each joint, even less when taking into account the material (if wood, the study is even more complex due to the anisotropy of the material) as was done in the cited papers.

4.2.2 OVERVIEW

As a model both precise and versatile is still an active research topic well beyond the scope of this manuscript, a more qualitative approach is needed. The goal is to build a numerical model that, for two given assemblies A_1 and A_2, if A_1 is said to perform better than A_2 for some metric, it would not be far-fetched to say that a more sophisticated model, or even the built assemblies in real life, would arrive to the same conclusion, namely that A_1 performs better than A_2 on that metric. In other words, we are not interested in the absolute values of the model (be it the stress, strain, displacements etc.) but rather in the partial ordering of these values when comparing two assemblies. The model should only be simple enough that extrapolating the results to reality would not change their ordering.
This section aims to find a generated assembly that performs better, for some relevant metric, than a hand-drawn, tradition-inspired, assembly. If so, then it is not impossible that in real life, the former is also better than the latter, which justifies the point of this manuscript, namely the exploration of the design space of interlocking assemblies to discover novel, relevant, assemblies.
Two phenomena must absolutely be taken into account when modelling an assembly. For two parts in contact:

- The interface cannot transmit tensile force but can carry compression forces: if tension arises between the two parts, they would simply move away from each other. However, if in compression, one part simply pushes on the other, the contact is not broken.
- Thanks to friction, an assembly may be in equilibrium even if the forces are not orthogonal to the interface.

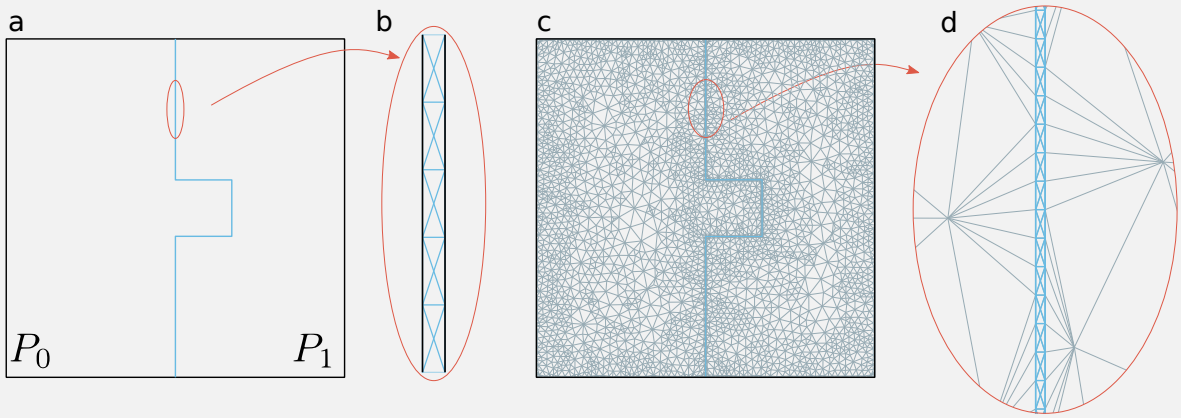


Figure 4.23| Left: the gap between two parts is bridged by links, in blue. Right: 2D shell elements of the parts are connected to the 1D beam element of the links.

These two non-linearities can be summed up in one observation: if the resulting force at a point of the interface is not in the cone of friction, equilibrium is out of reach and disassembly occurs. A 2-parts assembly $A = \{P_0, P_1\}$ is modelled as follows: the separating curve is offset by a small amount along the normal vectors of its segments, as shown on FIGURE 4.23a. Part P_0 is created using the original curve, P_1 using the offset one; thus there is a gap between these two parts. Crosses are created in the gap, their geometry

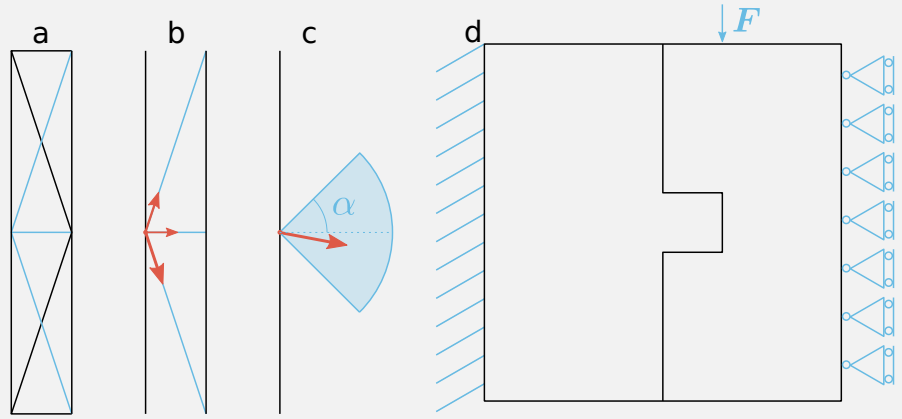


Figure 4.24| a) - links are grouped 3 by 3. b) - they can only transmit axial forces. c) - the resultant force must lie inside the friction cone. d) - load distribution and boundary conditions.

being defined on FIGURE 4.23b in blue. These crosses are made of two elements, that we call **links**: links orthogonal to the interface (the short ones) and diagonal links (the long ones). The interiors of the parts are meshed using triangular shell elements, as shown on FIGURE 4.23c, and the links are modelled using standard beam elements. Special care is taken to connect the endpoints of the links to the shell elements of the parts, 4.23d.

Links are grouped three by three: each orthogonal link is grouped with the two diagonal links whose common endpoint is on P_0. Such a group is highlighted in blue in FIGURE 4.24a. The beam elements associated with the links are pinned at both ends. Thus, they can only transmit axial forces, be it tensile or compressive, see 4.24b. The resulting force is calculated at the point common to the three links. For this point to be in equilibrium, this force must lie in a friction cone parametrised by an angle α, see 4.24c.

For the remaining of this section part P_0 will always be on the left of part P_1, we can thus use words like “left”, “top” etc. As for the boundary condition and load distribution, the leftmost edge of part P_0 is clamped, the rightmost edge of P_1 cannot move in the horizontal direction and a downwards nodal force $\boldsymbol{F}$ is applied on the top edge of P_1, exactly above the rightmost point of the separating curve, see FIGURE 4.24d for an illustration of these concepts.

As of yet, the model is far from modelling contact between two parts. Indeed, if an analysis is carried out, an equilibrium is likely to be found where most of the resulting forces are not contained in the friction cone. To perfect the model, an iterative algorithm is implemented. Each link is modelled as a tube with a given diameter ϕ. At iteration n, a finite element analysis using linear theory is conducted and gives the axial forces in each link, from which the resultant forces can easily be calculated. There are two non-excluding possibilities:

1. A link carries a tensile force. Then its diameter is set to an extremely small value $\phi = \phi_{min}$ (in this study $\phi_{min} = 1e-08$ unit), see FIGURE 4.25a. As such, the link is prevented from transmitting a significant amount of force, effectively cutting it from the study. This step models the fact that a link in tension is synonymous with parts breaking contact: they cannot transmit force.
2. The resultant force is not inside the friction cone. It means that, at this point in the interface, the tangential forces are too great with respect to normal forces. These tangential forces being transmitted by the two diagonal links only, their diameters are reduced: they are multiplied by a factor $\beta \in [0, 1[$ (in this study $\beta = 0.5$), as illustrated on FIGURE 4.25b.

Otherwise (links are in compression and the resultant is in the friction cone), the diameters of the links stay the same (4.25d). At the beginning of the first iteration, all links are initialised to a default value $\phi_{max} = 1$

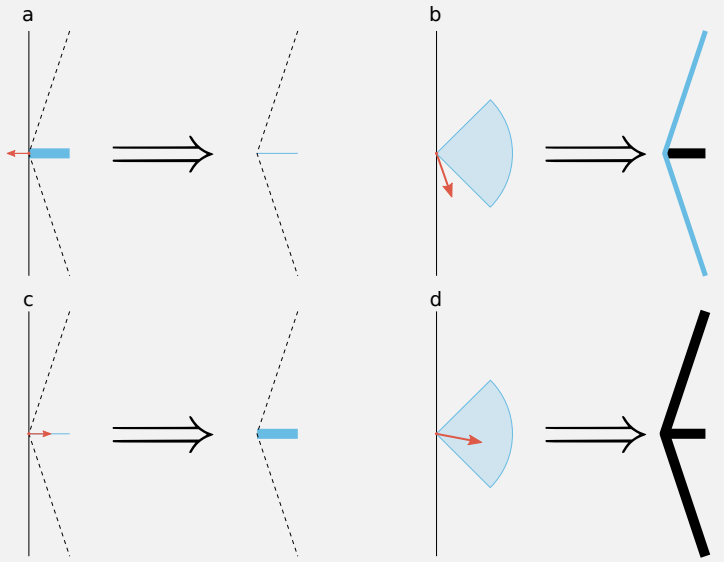


Figure 4.25| Depending on the situation, the diameter of a link stays constant, is decreased or increased.

unit; as a side note, the thickness of a tube is always set to be a hundredth of its diameter. A link that was once in tension (and thus having its diameter $\phi \leq \phi_{min}$) is not completely deleted from the study: if an analysis concludes that it is now transmitting (a small amount of) compression, its diameter is brought back to $\phi = \phi_{max}$, as shown on 4.25c. This small algorithm outputs two kinds of results:

- The diameter of an orthogonal link is either $\phi = \phi_{max}$ or $\phi \leq \phi_{min}$.
- The diameter of a diagonal link lives in the range $\phi \in]0, \phi_{max}]$ (it can be smaller than ϕ_{min} as if a link is in tension its diameter is set to $\phi = \phi_{min}$ and if the resultant force is not in the cone, it is further multiplied by $\beta < 1$: $\phi = \beta\phi_{min}$).

Thus, if the resultant force is outside the friction cone, the diameters of the diagonal links are reduced while the diameter of the orthogonal one stays constant, which diminishes the contribution of the tangential force compared to the normal one, hence bringing the resultant closer to the friction cone. If a link is in tension, its diameter becomes negligible: the tensile component disappears from the resultant, bringing it closer to the friction cone. These new diameters are used to create the beam elements at the beginning of the next iteration. This iterative algorithm reaches convergence when all resultant forces are either negligible or in the friction cone.

The geometry is handled in Grasshopper®, an add-on of the software Rhinoceros®. Finite element analysis was performed with the plugin Karamba3d and the loop (necessary for the iterative algorithm) was coded using components from the Anemone library.

4.2.3 CHOICE OF METRIC AND JUSTIFICATION

Once a stable solution is found by the algorithm, we use a measure of the maximal internal stress (calculated as the sum of the two principal stresses) in the parts as the metric to compare assemblies between them. Yet, because of the unpredictability of the shape of the generated designs, it proved to be impossible to craft an algorithm that could generate a quality mesh for all designs: at a point of high curvature of the separating polyline, for instance, *i.e* a point of stress concentration, it happens that a bad triangle leads to extremely high maximal stress while a slightly different mesh would yield much lower maximal stress in P_0.

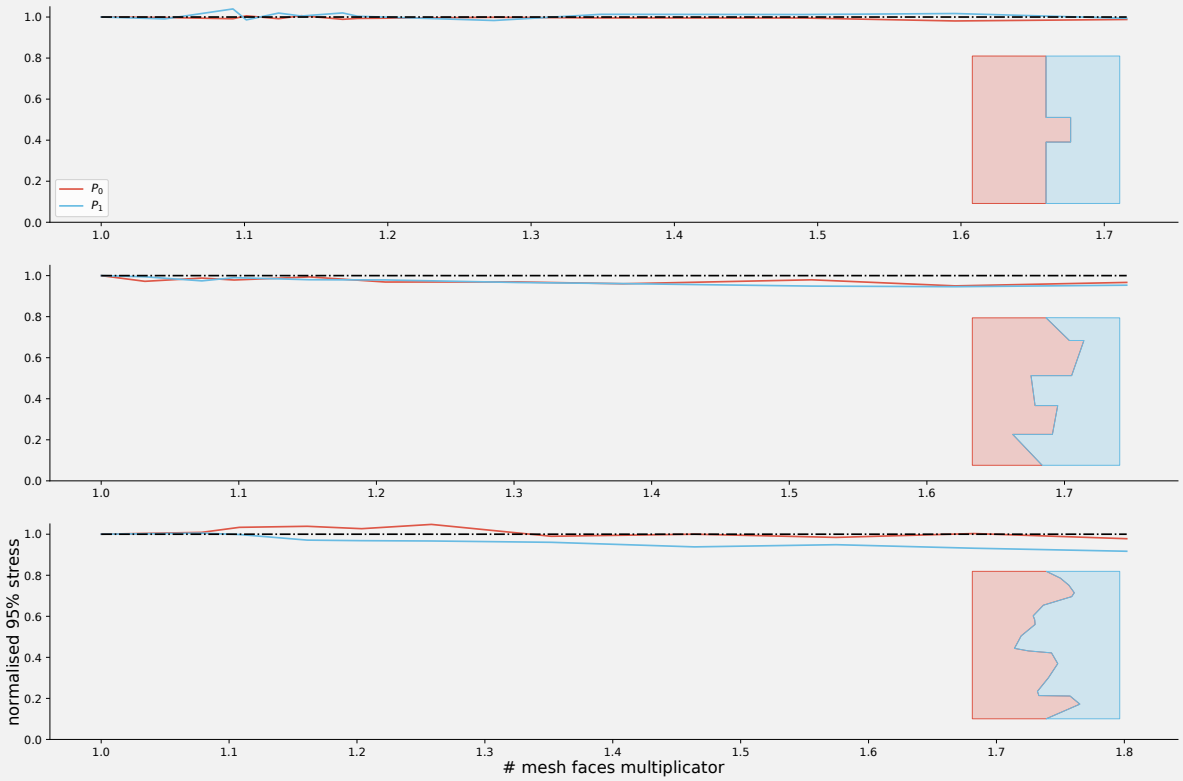


Figure 4.26| Refining the mesh by adding up to $\simeq 75\%$ more faces does not significantly affect the 95% stress value.

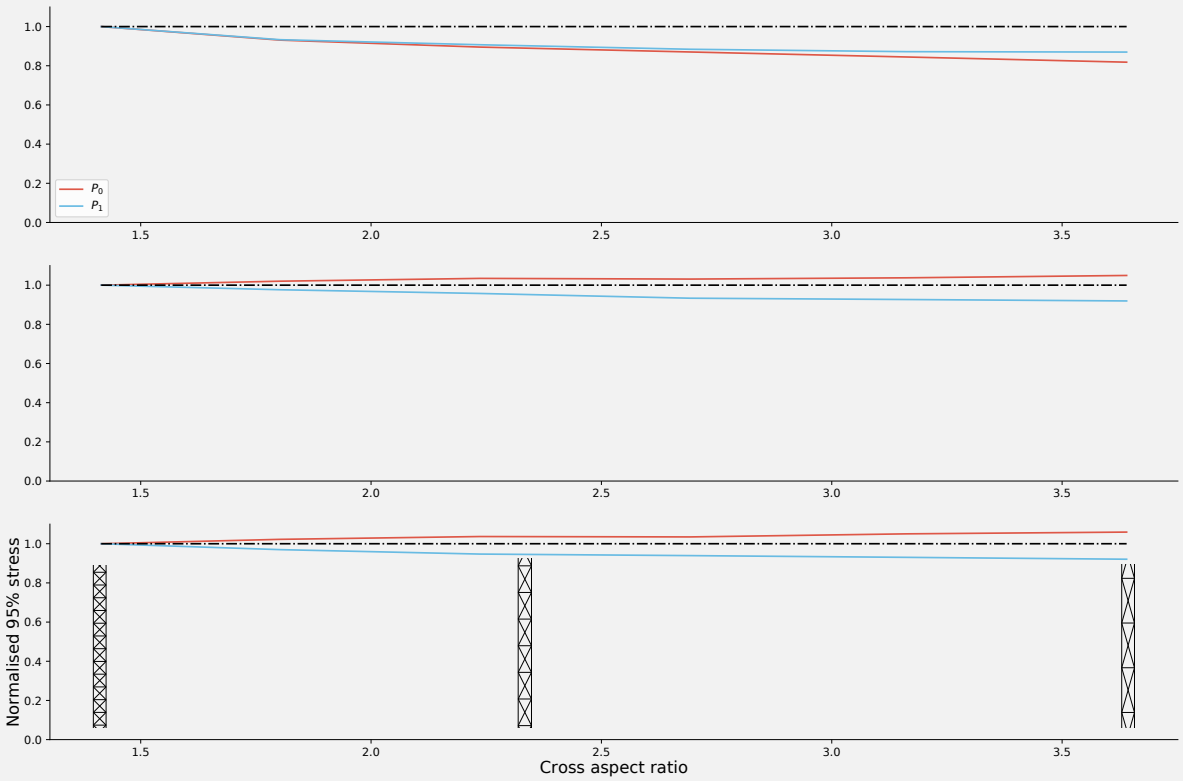


Figure 4.27| The aspect ratio of the cross has a significant impact on the 95% stress.

In addition, the point where the nodal force is applied on P_1 is where the maximum stress is reached, yet it is not an interesting point to study. The calculation of the maximal stress *per se* being too mesh-dependent for P_0 and uninteresting for P_1, the choice was made to consider the 95% stress: stresses at the calculation points are ranked according to their increasing value and the one indexed at 95% of the length of the list is kept. This value often occurs either near the separating polyline or close to the boundaries of the mesh, and is still representative of the high stresses in the mesh, while being completely mesh-independent, as shown on FIGURE 4.26. In this figure, three designs are studied: the top one is hand drawn and takes inspiration from a traditional mortise and tenon assembly. The middle one has been automatically generated and obeys a translation to the right. The bottom one has also been automatically generated and obeys a rotation. The mesh constituting the parts has been gradually refined, up to adding about 75% additional faces, from approximately 9500 faces to 17500 faces (the exact number varies with the assembly). For each mesh, the aforementioned algorithm was executed and the 95% stresses were recorded in each part. These graphs show that the 95% stresses do not change significantly with mesh variations, thus proving that it is a stable metric that we can use to quantitatively compare assemblies with.

Aside from the mesh quality, the aspect ratio of the crosses bridging the gap between P_0 and P_1, calculated as the ratio between the length of a diagonal link and the length of an orthogonal link, is a parameter whose influence is of interest. FIGURE 4.27 shows a variation of the 95% stress with regards to the cross aspect ratio; it is argued that the larger the aspect ratio, the fewer the crosses that can be fitted along the separating curve, and thus the higher the force transmitted per link, which modifies the distribution of stresses, especially near points of high curvature. That being said, the net variation does not exceed 10% of the reference value, implying that the choice of the aspect ratio is not too influential on the stress result. In the following, the aspect ratio is fixed to $\sqrt{10} \simeq 3.2$ (see FIGURE 4.24 for a visual representation of the crosses), as it seems to be a value around which the 95% stress does not change much.

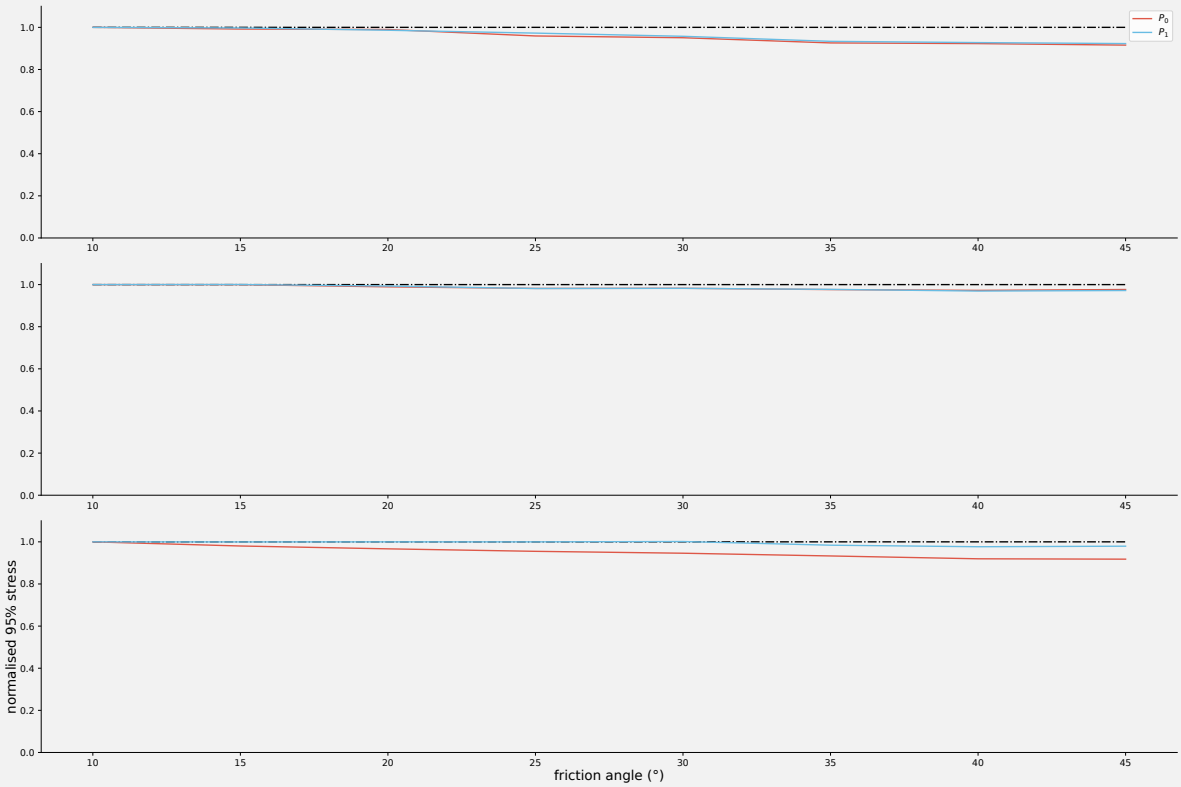


Figure 4.28| The friction angle has a predictable impact on the 95% stress.

Last but not least, the friction angle α plays a significant role in the algorithm outlined above. Its influence on the 95% stress is thus studied, by having it varying in the range $[10°, 45°]$, see FIGURE 4.28. Unsurprisingly, the higher α the lower the 95% stress. Indeed, as α increases, the interface between the parts can carry more and more tangential loads, thus distributing more evenly the forces and reducing the stresses. As such the value of α does not change the ordering of the 95% stress of two assemblies. Moreover, it has been visually verified that α does not influence much the qualitative distribution of forces at the interface of the part. Hence its specific value is of no importance for our needs and is fixed to $\alpha = 30°$.

This small study shows that our choice of metric behaves nicely with the variation of key parameters, hence comforting us in this choice. Yet, to further test the quality of the model and of the algorithm a qualitative comparison with real-life assemblies is conducted.

4.2.4 IMAGE CORRELATION AND QUALITATIVE COMPARISON

In addition to the 95% stress, it is possible to extract from the model the resulting force at the endpoint on P_0 of each orthogonal link. As the resulting forces can only be compressive, they inform of how much a part is pushing onto the other. In real life, this would lead to points on either side of the interface getting closer and closer. Such behaviour can be captured and quantified using digital image correlation, and compared to the results of the numerical model. As a consequence, several assemblies were laser cut and loaded as to closely mimic the boundary conditions and load distribution presented on FIGURE 4.24d; see FIGURE 4.29. This study (the making of the laser-cut specimens, the experimental apparatus, as well as the image correlation) was done by Antoine Bayard during his internship at Navier laboratory in the summer of 2022. The image correlation was done using the software GOM Correlate.

FIGURE 4.29 presents the experimental apparatus used to conduct the study: each assembly has black speckles sprayed over a layer of white paint. The left side of the assembly (part P_0) is clamped on a supporting frame by the means of two planks tightly enclosing the part. The right part (P_1) is pinned such that only vertical motion is possible; the rightmost metal rod prevents any rotation around the plane's normal axis and any translation that is both horizontal and in the plane of the assembly. The two metal rods in the middle prevent out-of-plane motion. These rods as well as the clamping planks are held together using a rope, tightly enough to ensure contact between the parts and the supporting frame, but sufficiently loose to prevent compression forces in the assembly. Thus the boundary conditions of the physically built assembly match the ones of our numerical model, presented on FIGURE 4.24d. As for the load distribution, the nodal force is enacted by the means of weights (16 kg of weights + 0.3kg of the basket) that are suspended with a string to a point on the top edge of P_1 that is slightly offset to the right of the rightmost point of the assembly (it is not precisely directly above it, because the string would prevent from seeing well the interface, hindering the quality of the images). The scene is lit using a projector, and a camera (not shown in the picture as it was used to take it) is placed on a fixed table next to the projector. The basket is loaded 2kg by 2kg, and each time weight is added, a picture is taken. The set of pictures is then loaded in the software GOM Correlate to perform the image correlation.

On FIGURE 4.30 and FIGURE 4.31, the snapshots from the image correlation show only the vertical deformation ϵ_y, calculated using the standard formula $\epsilon_y = \frac{\Delta l}{l}$. ϵ_x is not shown as it is not the main deformation and values are lower making thus the noise higher, and images are often less readable. Each snapshot can be considered as 3-colours coded: the interior body of the parts is in green, meaning that vertical deformation is close to 0, showing that the interiors of the parts move as rigid bodies. Colours red and blue are respectively used to encode the distance between points expanding and narrowing. The deeper the blue or brighter the red, the more significant the corresponding phenomenon. The interface is regularly tagged and the values of the deformation ϵ_y of the associated points are written.

On the left of each image, the resultant (magnitude and direction) of each assembly analysed by Karamba

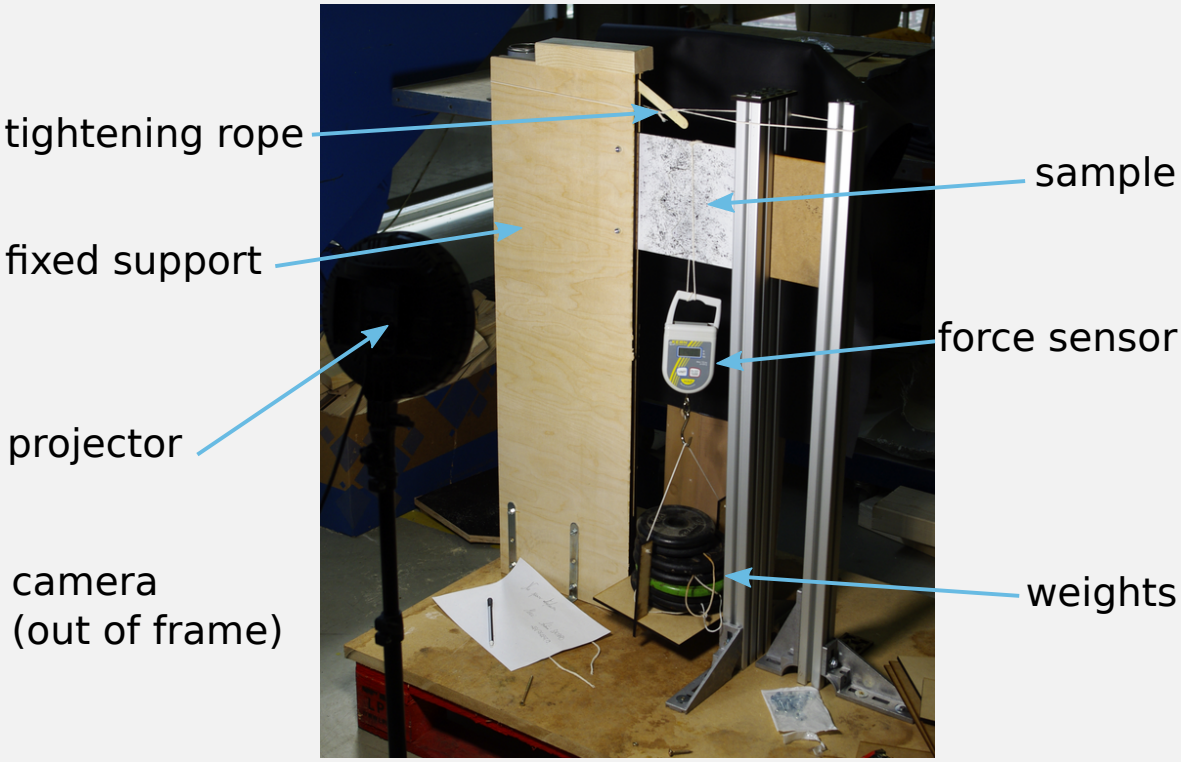


Figure 4.29| The experimental apparatus.

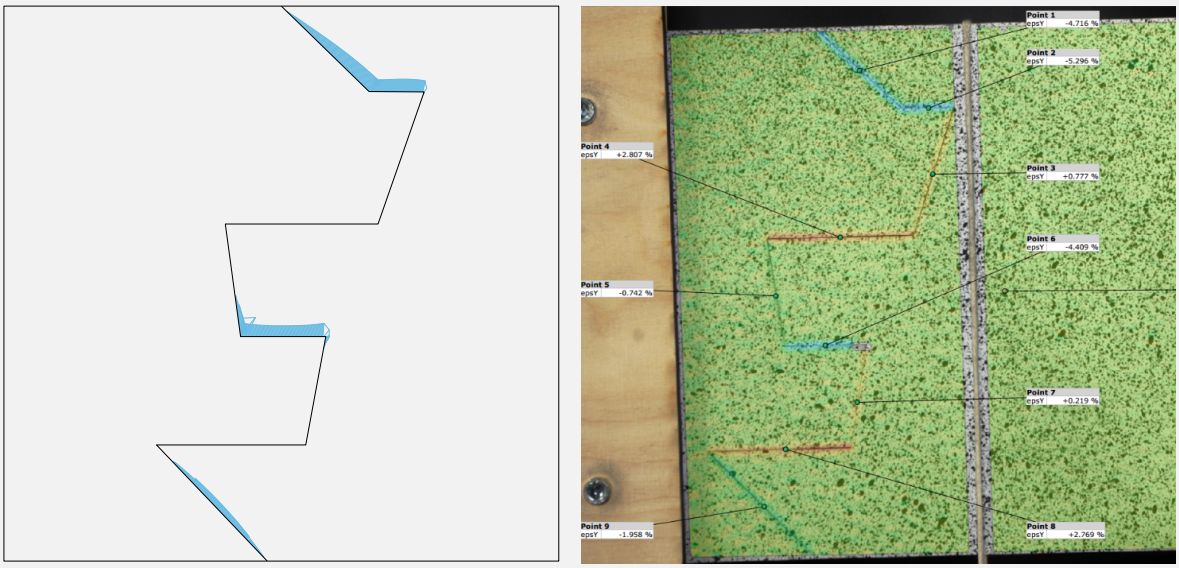


Figure 4.30| Left: our analysis of the forces in the links. Right: vertical deformation of the corresponding fabricated assembly: red indicates positive strains and thus tension, while blue indicates negative strains, i.e. compression. The two models are in qualitative agreement.

is shown in blue. A first obvious remark is that the image correlation captures the widening of the gap of the interface (in red), which is completely missed by our model: because these regions of the interface work in tension, our model can only output a binary answer, namely whether it is in tension or not. That being said, the results in compression can be compared, and one sees on FIGURE 4.30 a perfect match between the location of the points in compression predicted by our model and the points on both sides getting closer on the image correlation (in blue). Better, the magnitude of the compression, calculated by our model and displayed on the image correlation, seems to be in line with each other: the longer the vector on the left, the deeper the blue colour on the right. Again, our approach is purely qualitative, so we do not compare values, we just want to see if our model agrees with the image correlation. As far as the assembly on FIGURE 4.30 is concerned, our model fully fulfill our requirements.

Things degrade a bit when looking at the results on FIGURE 4.31. Indeed, while in most cases the two approaches show the same points in compression, our model sometimes finds an equilibrium that does not fully correspond to the observation of the image correlation. For instance, on 4.31a and d, our model does not find that the segments circled in red are in compression whereas the image correlation does; 4.31c shows another perfect match between our model and experimental data. FIGURE 4.31b shows the surprising and interesting case of a mismatch, a colour swap: focusing on the bottom two segments of the polyline, our model predicts that the top one is in compression and the bottom one in tension. The image displays the opposite, which does not seem to make sense. Indeed, when looking at the somewhat horizontal segments in the middle of the polyline, one sees that both approaches agree in saying that they are in compression. The fact that the correlation says that the bottom segment is also in compression implies that the bottom half of the assembly is squished, which cannot be explained given the boundary conditions and the location of the nodal force. Moreover, the geometry of the polyline on 4.31d is close to the one on 4.31b (both were obtained using the same input polyline, one was optimised in translation, the other in traction), but on the former, the image correlation shows that the bottom segment is in tension, which contradicts the results of the latter. Finally, physical manipulation of the assembly shows that, on 4.31b, our model is closer to reality than the image correlation.

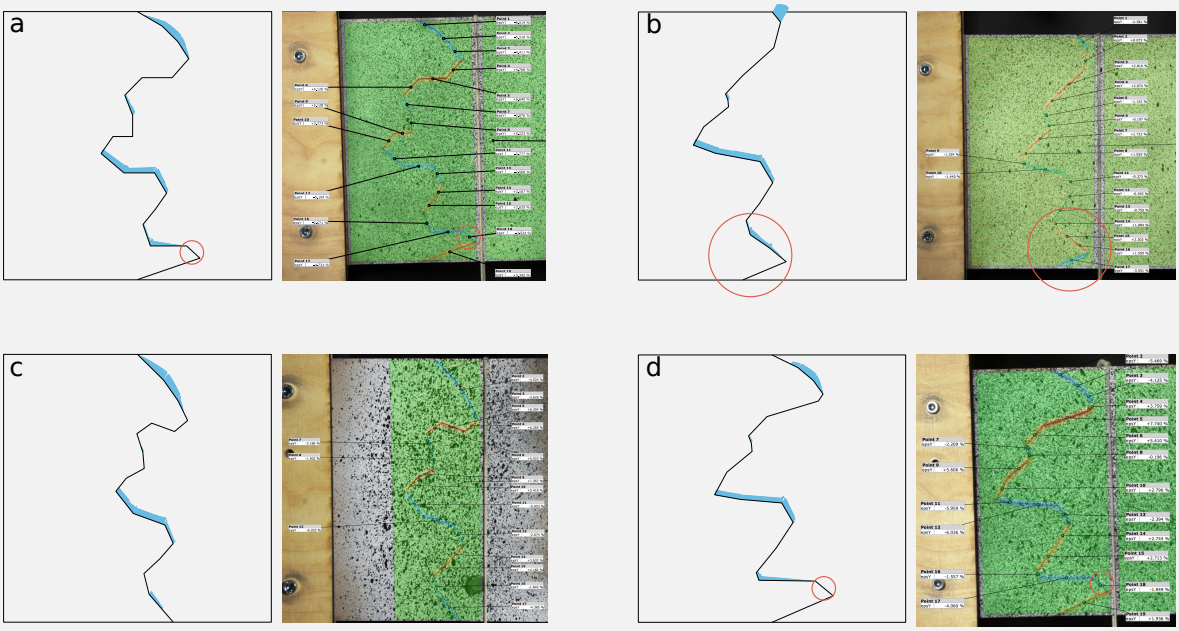


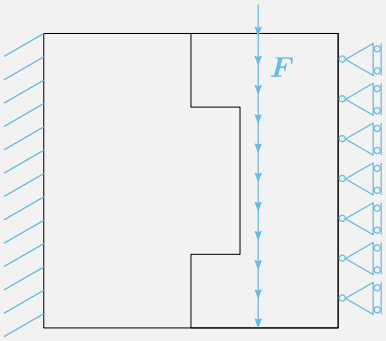
Figure 4.31| Left: Karamba analysis of the forces in the links. Right: vertical deformation of the corresponding fabricated assembly. Issues and mismatches are circled in red.

There are multiple possible explanations to account for this mismatch. A first explanation is that in the

software GOM Correlate, to our knowledge, the mesh used to carry the calculations is hidden from the user. Therefore, one cannot specify that there is an interface between two parts where the mesh shall be refined. Hence, it is certain that elements of the mesh were laid over the interface, and results are to be taken with suspicion. In other words, it would just be bad luck that several such elements were all around the bottom segments. A second, and we think quite likely, explanation is as follows: the software is highly sensitive to the black speckles on the specimen, a sensitivity that led to conundrums: on two successive images, with necessarily the very same speckles distribution, the software would deem the first image has to have excellent distribution and the next one as having a poor one making the calculations difficult. It is thus entirely possible that for the last image, the software considered the area around the bottom segments as having a bad distribution, and the results are thus unpredictable. The final, maybe most likely, explanation is simply an error in the boundary conditions during this specific experiment. This seems however surprising as experiments were all done in a row, and this experiment is the only one showing an issue.

We took care of explaining where this mismatch potentially comes from to highlight the following point: the aim of comparing our model with the image correlation approach is to qualitatively assess their differences, and to better calibrate our model using experimental data. Yet, as we saw, these images may provide the wrong data, preventing us from further improving our model.
Nevertheless, the agreement between the two approaches is good enough for us to deem our model well suited for our needs: being both numerically stable as justified on SECTION 4.2.3, as well as mostly in line with experimental data as seen in the current section, it is fit for comparing automatically generated assemblies between them consistently and reliably, which is the focus of the next section.

4.2.5 COMPARISON OF AUTOMATICALLY GENERATED ASSEMBLIES



This section aims to find automatically generated designs of 2-parts assemblies that have a lower 95% stress in both P_0 and P_1 than a human-designed, tradition-inspired, assembly. To prevent stress concentration at the point of application of the nodal force, the load is now distributed along a vertical line, at a constant distance to the right from the rightmost point of the polyline. A tenon and mortise-like assembly A_r ($_r$ for reference) has been created by hand, and the height of the tenon was optimised as to minimise the 95% stress calculated in both parts, see the inset. Our algorithm was run 50 independent times to generate as many designs constrained to have exactly two snap segments, like the reference assembly.

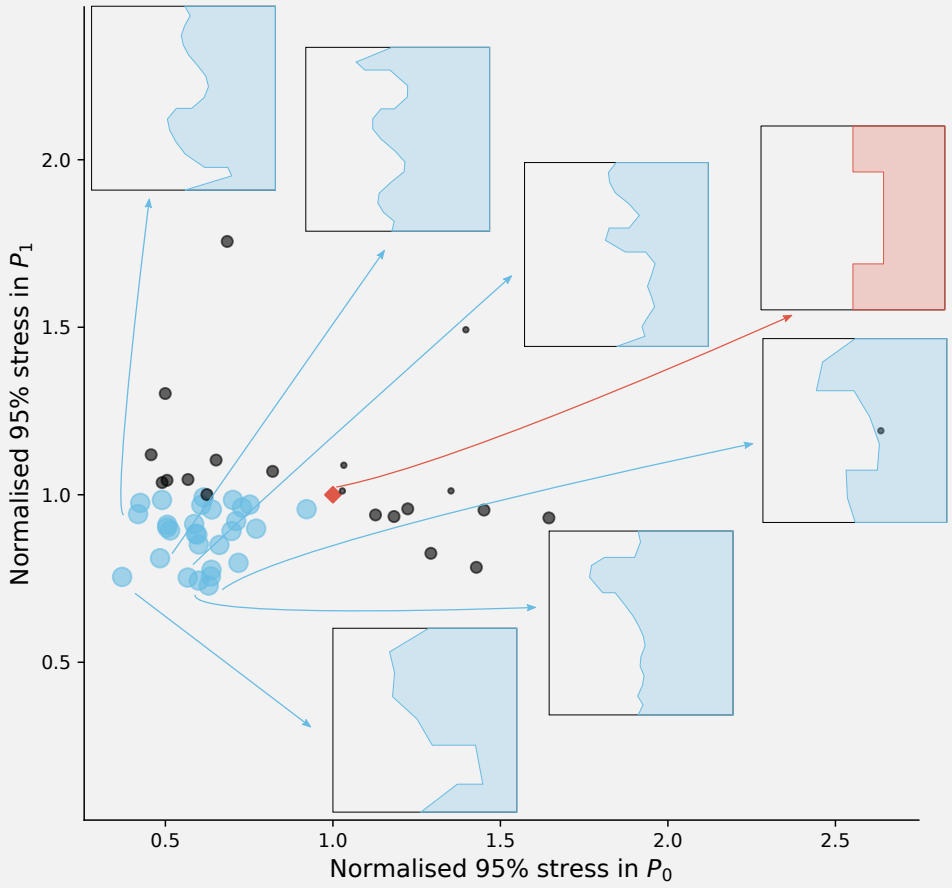


Figure 4.32| Most generated designs perform better than the reference.

The 95% stresses in both parts of each generated design have been calculated and compared with the results for A_r. Of those 50 designs (that were generated in a couple of seconds) 29 dominate the reference A_r, meaning that they are calculated as having lower 95% stresses in both parts than A_r. On FIGURE 4.32 the reference assembly A_r is highlighted in red. The 95% stress values are normalised by the values calculated for A_r and are displayed in the plane (stress in P_0, stress in P_1). Hence the stress values of A_r are displayed at (1,1), with a red diamond. The normalised stress values of the generated assemblies are depicted with dots. The 29 big blue dots correspond to the designs performing better than the reference and the 6 best performing assemblies are shown. Only five designs were strictly dominated by the reference and their stress values are depicted with small black dots. The remaining 16 designs are equivalent to the reference

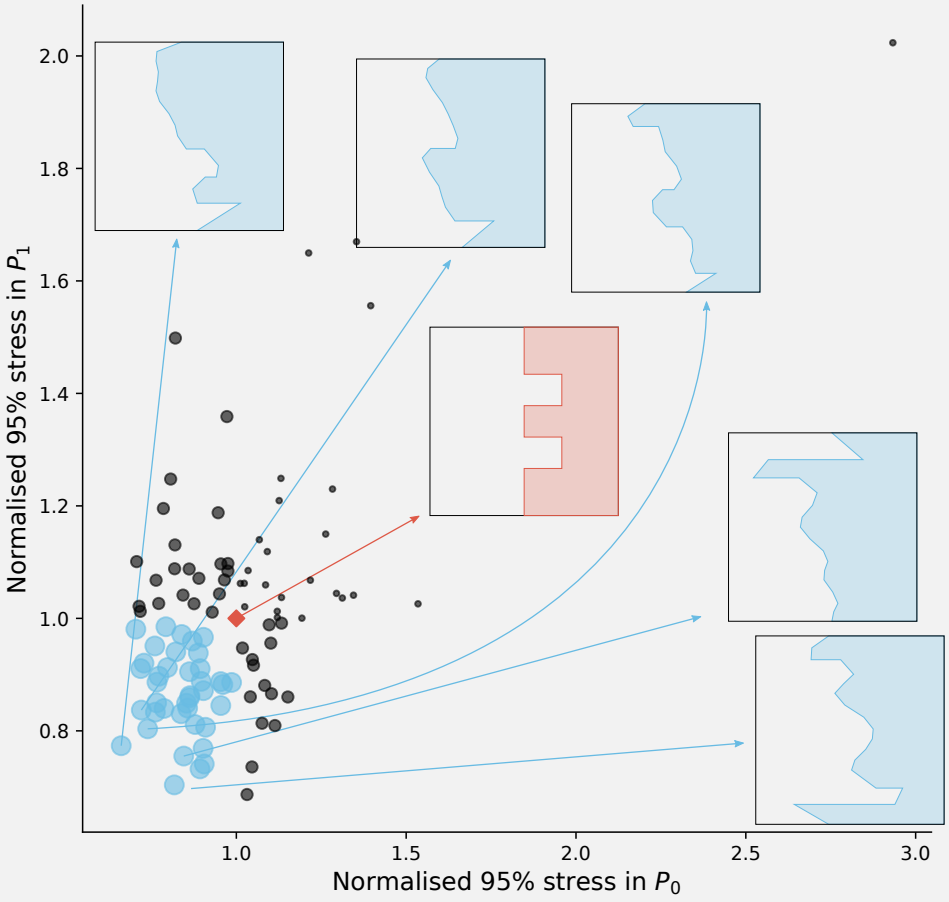


Figure 4.33| 39 out of 99 generated designs perform strictly better than the reference.

with the stress measure improved in one part but degraded in the other. This figure hints at a seemingly pleasing feature of the design space of 2-parts assemblies: a random walk in this space seems likely to yield assemblies that are relevant with regards to our numerical analysis. However, of course, not all such random designs are good: FIGURE 4.32 shows that for some designs, stress was multiplied by as much as 2.6.
The design problem with the point load (see FIGURE 4.24d) was also investigated, though the results are not shown as they are very close to the ones above: out of 50 designs, 23 performed better than the reference, which again hints that a random walk gives useful results.

For a hand-drawn assembly more carefully designed, with four snap segments to distribute more evenly the forces at the interface, results are similar. Out of 99 generated designs, 39 dominated the reference, as shown on FIGURE 4.33.

This hints that a random search in the design space of 2-parts assemblies is powerful enough to find novel designs performing better than human ones. But the evaluation takes time: while generating the designs took a couple of seconds, the iterative mechanical analysis is slow, taking approximately half a minute per design. This is mostly due to the use of tools that are not intended to be used in the way we did (Grasshopper does not support loop, the plugin Anemone bypasses this, but at the cost of slowness), and to create the meshes, components performing more calculation than necessary were used. A custom implementation would certainly speed up the process, making such a search must more tractable.

4.2.6 CONCLUSION

Time was spent building a finite-element computational tool that can consistently and reliably analyse any kind of 2-parts assembly. Numerical stability and agreement with another method were studied, and this section culminates in SECTION 4.2.5, which, albeit quite short, shows a key feature of the design space of polygonal assemblies and helps in justifying why this manuscript bother in exploring it. Indeed, it turns out to be quite easy to find assemblies performing better than reference ones, by simply randomly walking in the design space. A promising route for further research would be, given designs performing better than or close to the reference, to execute a local optimisation search, where polyline vertices are moved in ways that improve the stresses. A naive algorithm can be outlined: each vertex is randomly moved in a small disk centred at its location. Whenever such a motion improves the stress in the parts, the direction in which the vertex moved is used to bias further its further displacement (akin to a "gradient direction"). The stress measure is not expected to dramatically improve with such a naive algorithm, but it surely leads to better designs.

4.3 STUDIES RELATED TO THE CONES OF ROTATIONAL FREEDOM

4.3.1 A BETTER UNDERSTANDING OF THE CONES OF ROTATIONAL FREEDOM

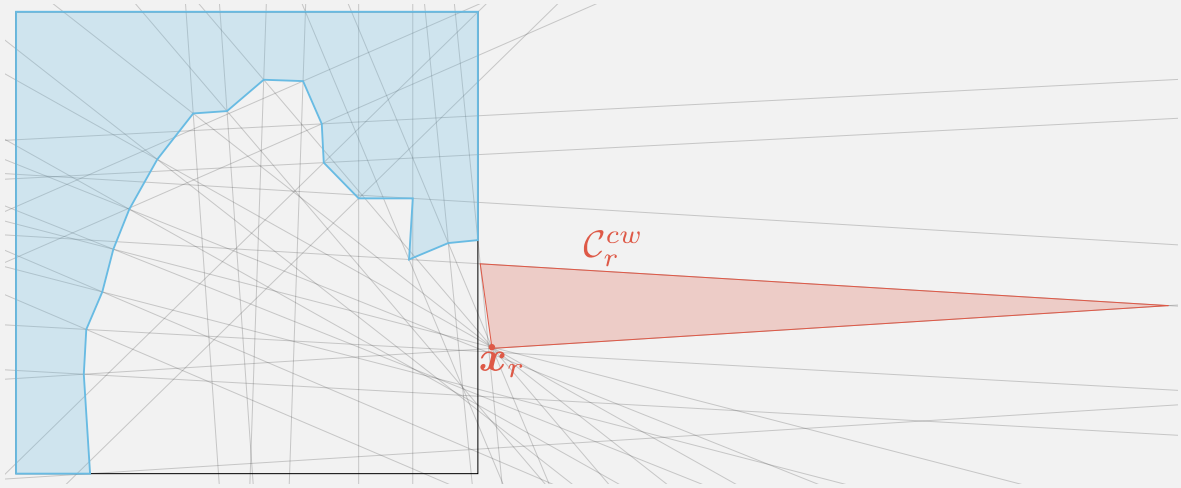


Figure 4.34| An assembly and the $\mathcal{C}_r^{cw}$ (in red) of part P_1 (in blue)

FIGURE 4.34 shows a 2-parts assembly $A = \{P_0, P_1\}$, with P_1 (in blue) obeying a clockwise rotation around $\boldsymbol{x}_r$. The cone $\mathcal{C}_r^{cw}$, shown in red, has been calculated and is compact enough to be displayed in the plane instead of projected onto the sphere. The grey lines depict the boundary of the half-planes encoded in the system $\boldsymbol{A}_r\boldsymbol{x} \leq \boldsymbol{b}$. Two situations are zoomed upon on FIGURE 4.35: on the left the centre of rotation $\boldsymbol{x}$ is taken as a vertex of the cone $\mathcal{C}_r^{cw}$; on the right $\boldsymbol{x}$ is taken on an edge. Let us first study the left situation.

$\boldsymbol{x}$ is a vertex of $\mathcal{C}_r^{cw}$

There exists a set of indices I_x (with $|I_x| \geq 2$) such as the lines defined by the system

$$\boldsymbol{A}_{I_x}\boldsymbol{x} = \boldsymbol{b}_{I_x}$$

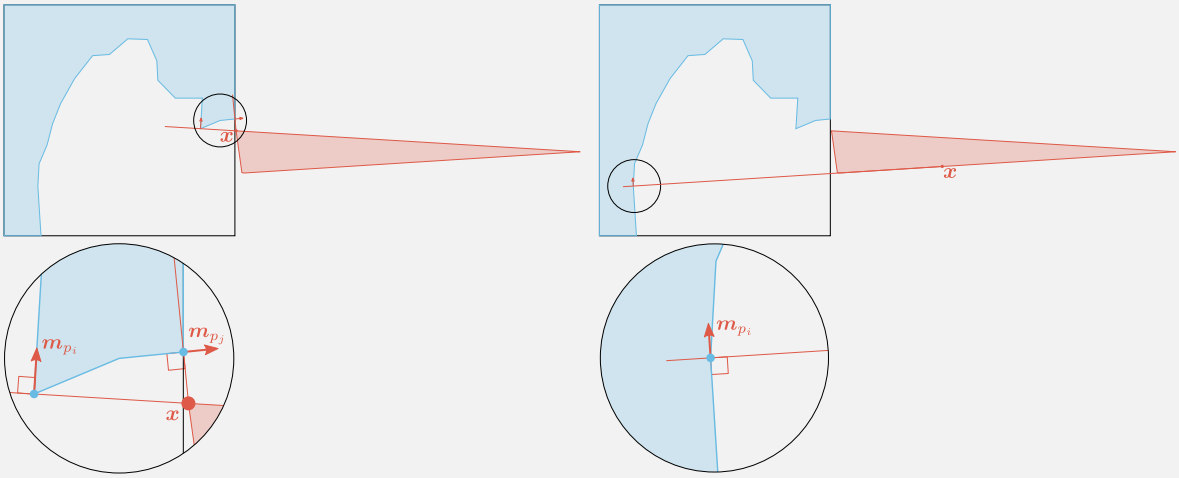


Figure 4.35| Zooms on instantaneous directions of motion $\boldsymbol{m}(\boldsymbol{p}, \boldsymbol{x})$, depending on whether $\boldsymbol{x}$ is a vertex of $\mathcal{C}_r^{cw}$ or on an edge.

intersect in $\boldsymbol{x}$, where $\boldsymbol{A}_{I_x}$ and $\boldsymbol{b}_{I_x}$ means that these quantities are made of the rows indexed by I_x in $\boldsymbol{A}_r$ and $\boldsymbol{b}$. Each of these lines is derived from one of the two equations:

$$\begin{cases} \text{Either } \boldsymbol{m}_{p_i} \cdot \boldsymbol{n}_i = 0 \\ \text{Or } \boldsymbol{m}_{p_{i+1}} \cdot \boldsymbol{n}_i = 0 \end{cases}$$

for $i \in I_x$ which exactly means that the segment $[\boldsymbol{p}_i, \boldsymbol{p}_{i+1}]$ is perpendicular to either the line $[\boldsymbol{p}_i, \boldsymbol{x}]$ or $[\boldsymbol{p}_{i+1}, \boldsymbol{x}]$, as shown on FIGURE 4.35 left. Informally speaking, these segments snapped from the point of view of centre $\boldsymbol{x}$. This means that an infinitesimal rotation around $\boldsymbol{x}$ would make the points indexed by I_x *slide* on the boundary. As such they are critical points guiding, helping, the (dis)assembly.

$\boldsymbol{x}$ is a vertex of an open edge of $\mathcal{C}_r^{cw}$

This situation is illustrated on the right of FIGURE 4.35. The same analysis as for the previous case applies, if not for the fact that now $|I_x| = 1$, meaning that there is only one point sliding on the interface between the parts.

$\boldsymbol{x}$ belongs to the interior of $\mathcal{C}_r^{cw}$

In such a case there are no critical points, and all points on the boundary move away from it.

We emphasize the study of these three cases to highlight a phenomenon that may ease the (dis)assembling process: much like we use sensory inputs when we put a key into a lock (does it slide ? is it stuck ?) a (dis)assembling agent - be it a robot or human - could greatly benefit from contact information between the part to move (P_1) and the part remaining (P_0) to perform the task. To that end rotating about a point $\boldsymbol{x}$ on the boundary of the cone $\mathcal{C}_r^i, i \in \{cw, ccw\}$ seems to be a better heuristic than rotating about a point in the interior of the cone. Indeed at least one point stays on the boundary of the fixtured part for infinitesimal motions and can thus inform the agent of the relative location of the parts. The most advantageous from that point of view would be to rotate around the prescribed $\boldsymbol{x}_r$ as it is defined by at least two snaps, i.e. there are at least two contact points (and potentially many more, on FIGURE 4.34 four segments have snapped) between the parts.

4.3.2 NUMBER OF SNAP SEGMENTS AND CONES OF ROTATIONAL FREEDOM

Contrary to the translation case where two opposite snap segments completely close the cone of translation freedom $\mathcal{C}_t$ to the user-prescribed cone, as explained in SECTION 3.1.1.1, in rotation the cones $\mathcal{C}_r^{ccw}$ and $\mathcal{C}_r^{cw}$ are often not reduced to the user prescribed centre $\boldsymbol{x}_r$: instead, they sometimes extend to infinity (meaning that the part obeys a translation as well), which may very well be an undesirable feature of the polyline, if only because for a multi-parts assembly, these cones may intersect, thus increasing the number of cells of dimension 0 of the NDBG and hence increasing the number of base DBGs that must be calculated to assess the interlocking of the assembly. It turns out that the user has indirect control over the cones $\mathcal{C}_r^{ccw}$ and $\mathcal{C}_r^{cw}$ by the means of the number of snap segments: the more snap segments, the narrower the cone. Thus, should the user wants the cones of rotational freedom to be small, or even one reduced to $\{\boldsymbol{x}_r\}$ the other empty if he/she only wants the part to obey a specific rotation in one direction, two strategies coexist:

1. The user asks for a large number of snap segments.
2. Once the polyline is optimised, the user asks for the introduction of new points on the snap segments and runs the optimisation again by specifying that these new, smaller, segments must snap.

The second strategy was implemented to generate the 4+1 parts assembly on FIGURE 4.21, to ensure that only four DBGs had to be calculated. Understanding why more snaps implies a narrower cone is straightforward: recall that the cones $\mathcal{C}_r^{ccw}$ and $\mathcal{C}_r^{cw}$ are calculated by solving SYSTEM (3.14), in the form $\boldsymbol{A}_r\boldsymbol{x} \star b$, $\star$ standing for $\geq$ and $\leq$. The second strategy creates more segments on the polyline, thus more constraints in $\boldsymbol{A}_r$: the more constraints, the less likely a point is to meet all of them, and thus the narrower the cone. The first strategy does not change the number of rows in the system but ensures that a greater number of them are equations instead of inequalities: a greater number of half-planes have their boundary intersecting precisely on the prescribed centre $\boldsymbol{x}_r$, thus narrowing the cone. To accurately measure the "closing" of cones that may be infinite (see FIGURE 3.11 where the cone extends to infinity), the spherical area and perimeter of its stereographic projection onto the unit sphere are measured. The perimeter is used as an additional measure to take into account the cases where the cone is reduced to an infinite half-line, as shown on FIGURE 3.12. In this case, indeed, the stereographic projection of the half-line is circular arc, of zero area but a finite perimeter. On FIGURE 4.36 twelve assemblies are displayed. The same input polyline was optimised to obey a counterclockwise rotation four independent times for an increasing number $n \in \{2, 3, 4, 5\}$ of snap segments (strategy 1, vertical axis). For each design with n snap segments, the snap segments were subdivided from zero to two times (strategy 2, horizontal axis). Hence, for instance, the design at row indexed by 3 and the columns indexed by 0 means that the input polyline was prescribed to snap 3 times, but no subdivision was required. The one at (row 3, column 1) took each of the 3 snap segments and subdivided them once by introducing a vertex in the middle. The polyline was optimised again to snap these 6 (smaller) segments. The design in (row 3, column 2) took each of the 6 previous snap segments and subdivided them again, totalling 12 snap segments that were optimised to snap. The design at (row i, column j) has $i2^j$ snap segments. As such, the design with the smaller amount of snap segments is at (row 2, column 0) with two snap segments, the one with the greatest at (row 5, column 2) with twenty snap segments.
The cones of rotational freedom $\mathcal{C}_r^{ccw}$ and $\mathcal{C}_r^{cw}$ are calculated and are displayed on the sphere, in blue and red respectively. The area and perimeter of each such spherical surface are displayed as coloured circles whose radii are proportional to their values: in light blue the area of $\mathcal{C}_r^{ccw}$; in dark blue the perimeter of $\mathcal{C}_r^{ccw}$. In light and dark red respectively the area and perimeter of $\mathcal{C}_r^{cw}$. One sees that the more snap segments the narrower the cones. The assembly at (3 vertical, 1 horizontal) shows the noticeable case where $\mathcal{C}_r^{cw} = \{\emptyset\}$ and $\mathcal{C}_r^{ccw}$ is almost reduced to a line of zero area. When no coloured circles are displayed, it means that the counterclockwise cone is reduced to the prescribed $\boldsymbol{x}_r$ and the clockwise cone is empty.
As a side note, FIGURE 4.36 also shows the range of possible designs that are accessible given the same input polyline.

Increasing the number of snap segments, be it by strategy 1 or strategy 2, is a conceptually cheap method to reduce the actual cone of rotational freedom to the user-prescribed centre of rotation, thus making the enumeration of the base DBGs short and brief, and the assessment of the interlocking of an assembly more direct. Note that strategy 2 barely changes the shape of the polyline: on each row i of FIGURE 4.36, the design at (row i, column 0) closely looks like the design at (row i, column 2).

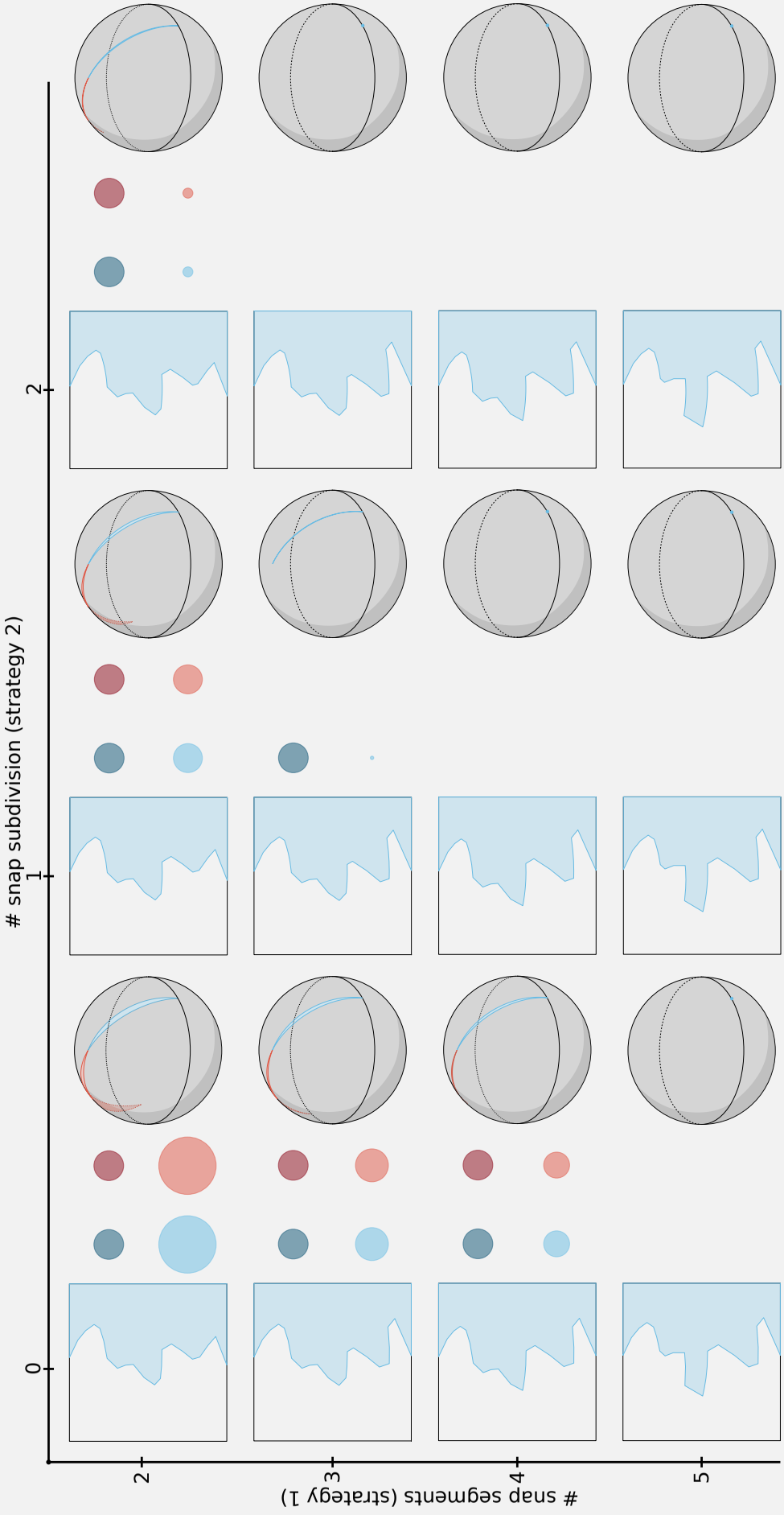


Figure 4.36| The greater the number of snaps and/or snap segment subdivision, the narrower the cones of rotational freedom.

4.3.3 ON FABRICATION IMPERFECTIONS AND MOTION TOLERANCE

What follows is a study aimed at mimicking the behaviour of a real-life assembly which calls into question the very possibility of obeying a rotation around the prescribed centre $\boldsymbol{x}_r$. In this section, $\mathcal{C}_r$ refers to both $\mathcal{C}_r^{ccw}$ and $\mathcal{C}_r^{cw}$, the direction of the rotation is of no matter; $\star$ refers to $\geq$ and $\leq$; $\partial\mathcal{C}_r$ refers to the boundary of the cone, i.e. the set of vertices and edges defining the cone and $\mathring{\mathcal{C}}_r$ defines the interior of the cone. We will conclude that in practice the rotation about $\boldsymbol{x}_r$ is the one of the least likely to happen among all the rotations around points in $\mathcal{C}_r$. To understand why we must consider two uncertainties: one on the location of the centre of rotation $\boldsymbol{x}_r$ and one on the location of the points $(\boldsymbol{p}_i)_{i\in[\![1,k]\!]}$ constituting the separating polyline.

4.3.3.1 Uncertainty on the location of the centre of rotation

For $\epsilon \ll 1$ let $\boldsymbol{x} \in \mathcal{C}_r$ and let

$$\boldsymbol{x}_\epsilon \sim \boldsymbol{x} + \mathcal{N}(0, \epsilon^2)$$

be the centre of rotation of a part taking into account some imprecision on its location. This could model the fact that a (dis)assembling robot is ill-calibrated or that it vibrates around the mean position $\boldsymbol{x}$. Then, obviously if $\boldsymbol{x} \in \partial\mathcal{C}_r$ there is a chance that $\boldsymbol{x}_\epsilon \notin \mathcal{C}_r$: if $\boldsymbol{x}$ is on an open edge between two vertices, then the probability that $\boldsymbol{x}_\epsilon \notin \mathcal{C}_r$ is $\frac{1}{2}$; if $\boldsymbol{x}$ is a vertex of $\mathcal{C}_r$ (which is the case for $\boldsymbol{x}_r$) then the probability that $\boldsymbol{x}_\epsilon \notin \mathcal{C}_r$ is even higher. For that reason obeying a rotation about $\boldsymbol{x}_r$ is less likely than obeying a rotation about any $\boldsymbol{x} \in \mathring{\mathcal{C}}_r$. Yet the main reason that explains why the rotation about $\boldsymbol{x}_r$ is less likely than the one about any other $\boldsymbol{x} \in \mathcal{C}_r$ is more subtle and comes from taking into consideration uncertainties on the position of the vertices $(\boldsymbol{p}_i)_{i\in[\![1,k]\!]}$ of the separating polyline.

4.3.3.2 Uncertainty on the location of the points of the separating polyline

Let

$$\forall i \in [\![1,k]\!] \;\; \boldsymbol{p}_i^\epsilon \sim \boldsymbol{p}_i + \mathcal{N}(0, \epsilon^2)$$

Such $(\boldsymbol{p}_i^\epsilon)_{i\in[\![1,k]\!]}$ models the tolerance of fabrication of a part: each point $\boldsymbol{p}_i^\epsilon$ is randomly shifted around its expected position $\boldsymbol{p}_i$ by a random variable following a Gaussian law with an $\epsilon \ll 1$ variance. This modelling of imperfections might be rightly deemed as quite crude, but more sophisticated models would not change the heart of the conclusion we will arrive to.

By construction of our algorithm, at least two segments of the polyline snapped for the assembly to obey $\boldsymbol{x}_r$. Therefore there exists a subset of indices I_{x_r} such as one has the equality $\boldsymbol{A}_{I_{x_r}}\boldsymbol{x}_r = \boldsymbol{b}_{I_{x_r}}$. As illustrated on FIGURE 4.34, this system defines $|I_{x_r}|$ lines that intersect in $\boldsymbol{x}_r$. The question that arises is "*What happens to this system when we use the perturbed polyline points $(\boldsymbol{p}_i^\epsilon)_{i\in[\![1,k]\!]}$ to build the matrix $\boldsymbol{A}_r^\epsilon$ and the vector $\boldsymbol{b}^\epsilon$?*" where $\boldsymbol{A}_r^\epsilon$ and $\boldsymbol{b}^\epsilon$ are built using the $(\boldsymbol{p}_i^\epsilon)_{i\in[\![1,k]\!]}$ in a similar fashion as $\boldsymbol{A}_r$ and $\boldsymbol{b}$ are built with $(\boldsymbol{p}_i)_{i\in[\![1,k]\!]}$, see SECTION 3.1.1.2 for a definition. Let us focus on the subset of lines indexed by I_{x_r}. Because the $\boldsymbol{p}_i^\epsilon$ are randomly shifted, in all generality one has

$$\boldsymbol{A}^\epsilon_{I_{x_r}}\boldsymbol{x}_r - \boldsymbol{b}^\epsilon_{I_{x_r}} \neq \boldsymbol{0}$$

Geometrically, the lines defined by $\boldsymbol{A}^\epsilon_{I_{x_r}}$ and $\boldsymbol{b}^\epsilon_{I_{x_r}}$ are randomly shifted and tilted compared to their counterparts defined by $\boldsymbol{A}_{I_{x_r}}$ and $\boldsymbol{b}_{I_{x_r}}$. This means that they do not all intersect in $\boldsymbol{x}_r$; in all generality, they do not even all intersect at the same point. So unless the point $\boldsymbol{x}_r \in \mathcal{C}_r^\epsilon$ (where $\mathcal{C}_r^\epsilon = \{\boldsymbol{x} \in \mathbb{R}^2 \mid \boldsymbol{A}_r^\epsilon\boldsymbol{x} - \boldsymbol{b}^\epsilon \star 0\}$) the rotation around $\boldsymbol{x}_r$ will be impossible.

The probability of the event $\boldsymbol{x}_r \in \mathcal{C}_r^\epsilon$ is easily quantified. For $j \in I_{x_r}$ let $\boldsymbol{a}_j$ and b_j be respectively the

row of $\boldsymbol{A}_r$ and the entry of $\boldsymbol{b}$ indexed by j. Similarly we define $\boldsymbol{a}_j^\epsilon$ and b_j^ϵ. As said above, in all generality one has $\boldsymbol{a}_j^\epsilon \boldsymbol{x}_r - b_j^\epsilon \neq 0$, and because the mapping $(\boldsymbol{a}_j, b_j) \mapsto (\boldsymbol{a}_j^\epsilon, b_j^\epsilon)$ is random, there is a 50-50 chance that $\boldsymbol{a}_j^\epsilon \boldsymbol{x}_c - b_j^\epsilon > 0$. Geometrically and in layman terms, since $\boldsymbol{x}_r$ is on the line defined by $\boldsymbol{a}_j$ and b_j and since the line defined by $\boldsymbol{a}_j^\epsilon$ and b_j^ϵ is randomly shifted and tilted, then there is as much chance for $\boldsymbol{x}_r$ to be "above" this shifted line (i.e. $\boldsymbol{a}_j^\epsilon \boldsymbol{x}_r - b_j^\epsilon \geq 0$) than "underneath" ($\boldsymbol{a}_j^\epsilon \boldsymbol{x}_r - b_j^\epsilon \leq 0$). At this step of the study, one would conclude that the probability the $\boldsymbol{x}_r \in \mathcal{C}_r^\epsilon$ is $\mathbb{P}(\boldsymbol{x}_c \in \mathcal{C}_r^\epsilon) = \frac{1}{2^{|I_{x_c}|}}$.
We can go further in this study. It is possible, especially for a thin $\mathcal{C}_r$, that a line j bounding the cone $\mathcal{C}_r$ but not indexed by I_{x_r}, i.e. not going through $\boldsymbol{x}_r$ (e.g. the upper line bounding $\mathcal{C}_r^{cw}$ on FIGURE 4.34) shifts on the other side of $\boldsymbol{x}_r$ after the random transformation $(\boldsymbol{A}_r, \boldsymbol{b}) \mapsto (\boldsymbol{A}_r^\epsilon, \boldsymbol{b}^\epsilon)$. In other words, if one has $\boldsymbol{a}_j \boldsymbol{x}_r > b_j$ one could get, for a transformation of sufficiently high magnitude, $\boldsymbol{a}_j^\epsilon \boldsymbol{x}_r < b_j^\epsilon$. As this could happen for any line not indexed in I_{x_r}, and even if we are lucky and all lines indexed by I_{x_r} are randomly moved on the "right side" of $\boldsymbol{x}_r$ ($\boldsymbol{A}_{I_{x_r}}^\epsilon \boldsymbol{x}_r \star \boldsymbol{b}_{I_{x_r}}^\epsilon$) with a probability of exactly $\frac{1}{2^{|I_{x_r}|}}$), we arrive at the conclusion that:

$$\boxed{\mathbb{P}(\boldsymbol{x}_r \in \mathcal{C}_r^\epsilon) \leq \frac{1}{2^{|I_{x_r}|}}}$$

The key takeaway is that the more snap segments (i.e., the greater $|I_{x_r}|$) the less likely an imperfect part is to obey the prescribed centre $\boldsymbol{x}_r$. In the best-case scenario, there is a $\frac{1}{4}$ probability that it happens. Had we modelled the imperfections in a more sophisticated fashion, the exact value of this probability would certainly have changed, but not the fact that $\boldsymbol{x}_r$ is the least likely point of $\mathcal{C}_r$ to be obeyed. This property is illustrated on FIGURE 4.37: the same initial polyline has been optimised ten times, for prescribed snap numbers varying from 0 to 9. For each of such ten optimised polyline $(\boldsymbol{p}_i)_{i\in[\![1,k]\!]}$, 200 imperfects designs $(\boldsymbol{p}_i^\epsilon)_{i\in[\![1,k]\!]}$ were generated by slightly moving the polyline's vertices (with $\epsilon = 0.001$). We kept track of the number of times an imperfect polyline obeys the prescribed centre $\boldsymbol{x}_r$ to estimate the probability of obedience $\mathbb{P}(\boldsymbol{x}_r \in \mathcal{C}_r^\epsilon)$ (sampled mean). FIGURE 4.37 clearly shows that this (estimated) probability is capped by $\frac{1}{2^{|I_{x_r}|}}$, as expected.

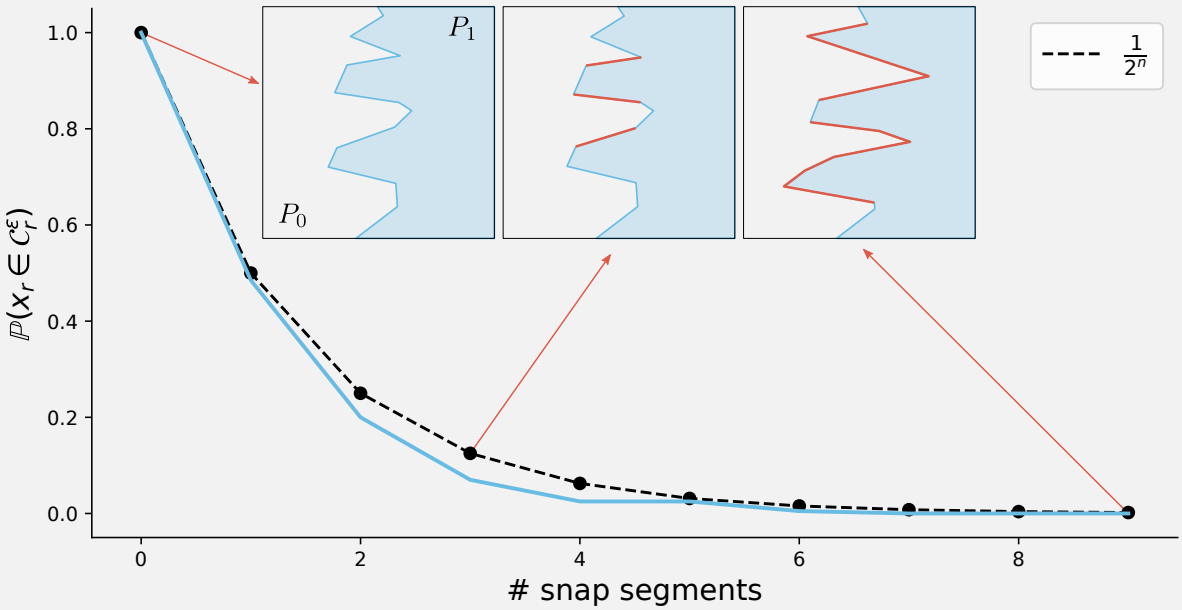


Figure 4.37| The probability that an imperfect design obeys the prescribed centre $\boldsymbol{x}_r$ is upper-bounded by $\frac{1}{2^n}$, with $n = |I_{x_r}|$. The snap segments are in red.

This study naturally raises the question "*Given a class of imperfect parts $(\boldsymbol{p}_i^\epsilon)_{i\in[\![1,k]\!]}$, what is the probability of a point $\boldsymbol{x} \in \mathbb{R}^2$ to actually be obeyed to ?*"

4.3.3.3 Likelihood of a point to be obeyed by a class of imperfect assemblies.

Given a polyline $(\boldsymbol{p}_i)_{i\in[\![1,k]\!]}$ and a law to model the imperfections $(\boldsymbol{p}_i^\epsilon)_{i\in[\![1,k]\!]}$, a numerical study can swiftly be executed to find the set of points likely to be obeyed by the part. For instance, on FIGURE 4.38, the separating polyline was perturbed using $\epsilon = 0.002$ in $\boldsymbol{p}_i^\epsilon \sim \boldsymbol{p}_i + \mathcal{N}(0, \epsilon^2)$ (for reference, the length of the diagonal of the design domain is 1). For each of such imperfect polylines, cone $\mathcal{C}_r^{cw}$ was calculated. The results are averaged over 1000 imperfect designs. The point in red is $\boldsymbol{x}_r$: one sees that it does not belong to the set of points likely to be obeyed. The curve in black shows the isoline 99: it encloses points having a probability greater than 0.99 of being obeyed by an imperfect polyline. The highlighted black point was found to be amongst the most likely to be obeyed (out of the 1000 designs, all obey it). This numerical study confirms that, given the modelling of imperfections, the prescribed centre $\boldsymbol{x}_r$ is extremely unlikely to be obeyed.

To get a better idea of the set of points likely to be obeyed, the same study has been repeated for varying

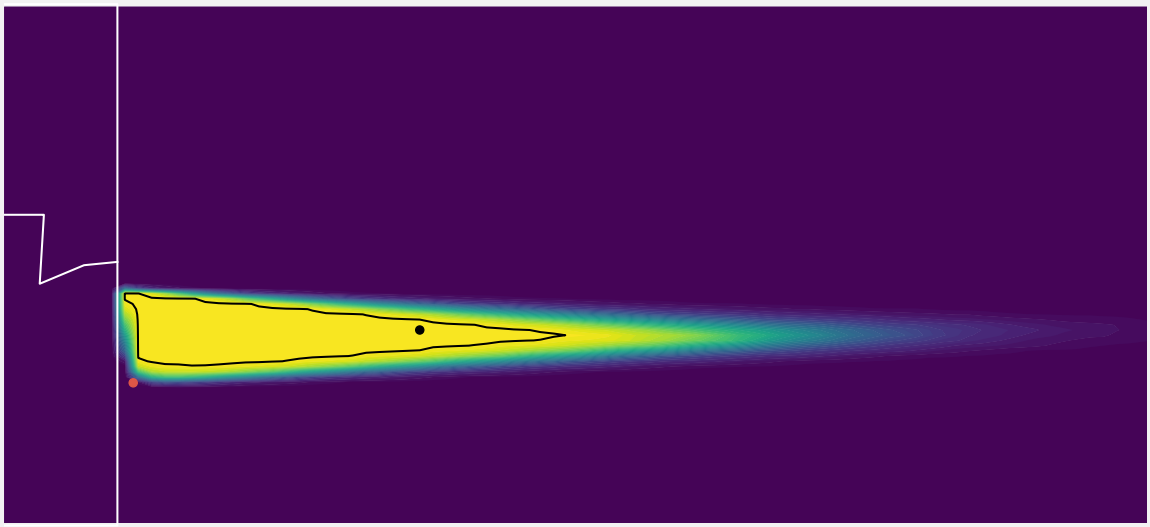


Figure 4.38| Heatmap of likelihood of obedience. In deep purple probability 0, in bright yellow, probability 1.

values of ϵ, see FIGURE 4.39: the box on the top right of each subfigure shows the estimated probability $\mathbb{P}(\boldsymbol{x}_r \in \mathcal{C}_r^\epsilon)$. The obedient set gracefully degrades, and one sees a subset (in yellow on all figures) that constantly has a high probability of being obeyed, no matter the magnitude of the imperfection. FIGURE 4.39 gives a choice to the user: if the location of the actual centre of rotation is not a hard constraint, then such heatmaps give the set of points with a high probability of being obeyed, thus letting the user choose *a posteriori* what the centre should be. On the other hand, if the location of the actual centre of rotation must be precisely the prescribed $\boldsymbol{x}_r$, then one should modify the separating polyline for it to robustly obey $\boldsymbol{x}_r$, even given the presence of some imperfection. Increasing the robustness of the separating polyline to some imperfection is what we call the **opening of the cone of rotational freedom**. It is the focus of the next section.

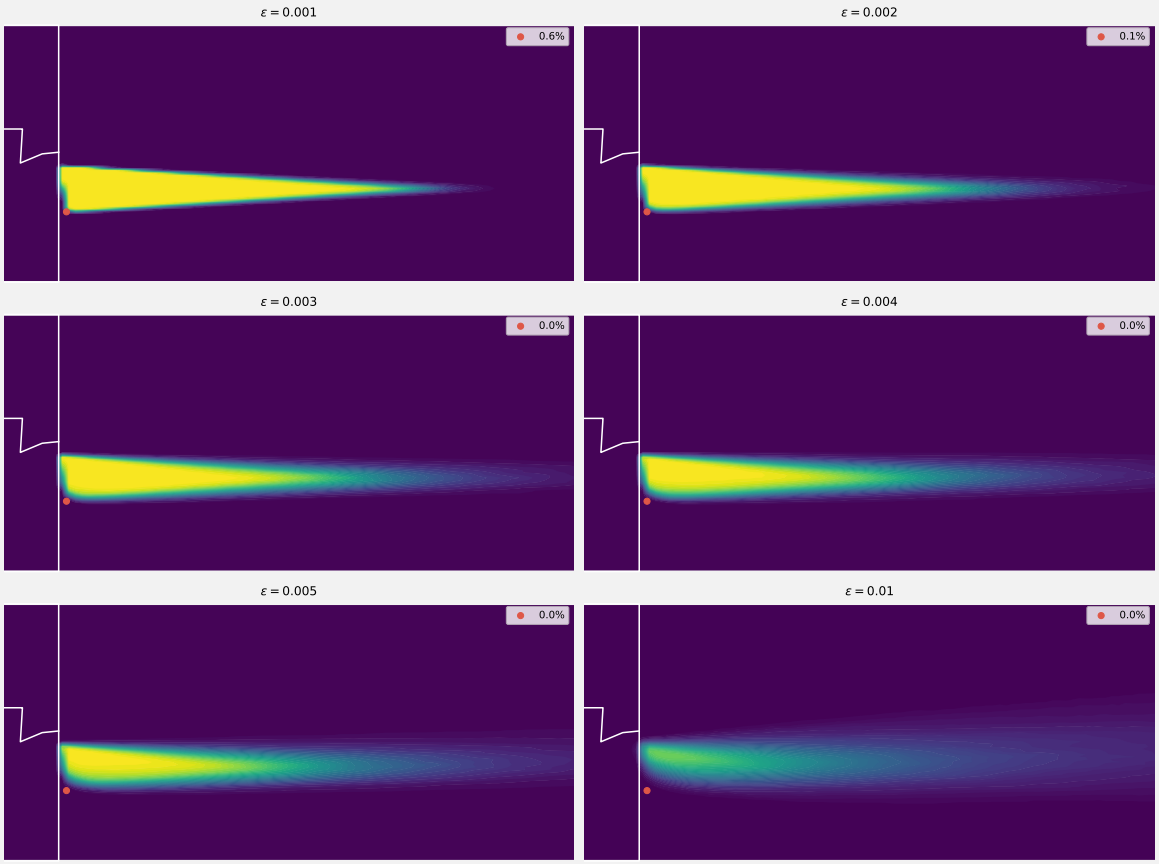
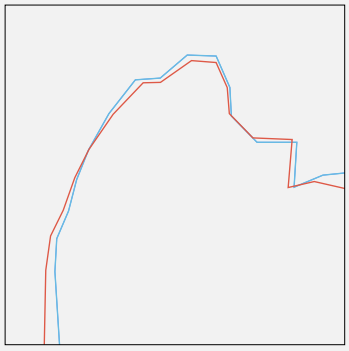


Figure 4.39| Heatmaps of the likelihood of obedience for several ϵ.

4.3.4 ROBUSTNESS TO IMPERFECTION



We see two manners in which one can perform the so-called opening of the cone of rotational freedom, for it to contain the prescribed centre $\boldsymbol{x}_r$, even in the presence of some imperfection. The first method consists of taking the separating polyline at the end of the GPA optimisation, and optimise it again by implementing a constraint of the form

$$s\boldsymbol{n}_i \cdot \boldsymbol{m}_{p_i} \geq \alpha$$

for some $\alpha \in \mathbb{R}^+$. The greater α, the more open the cone and the more $\boldsymbol{x}_r$ is in the interior of $\mathcal{C}_r$. In this section $\alpha = 0.1$. Of course, in such a case, the snap constraint must be deactivated, as it is incompatible with this constraint. The inset shows the input polyline in blue (so the one obtained at the end of the first round of GPA optimisation is the same as the one shown on FIGURE 4.34) and the output in red, where the snap constraint has been deactivated.

The same numerical study is conducted, and is presented FIGURE 4.40. The probability $\mathbb{P}(\boldsymbol{x}_r \in \mathcal{C}_r^\epsilon)$ is estimated on 1000 imperfect designs, for various magnitude ϵ, and displayed in each subfigures. For small enough ϵ, this probability is 1 or close to 1, meaning that the polyline is now robust to small imperfections. Obviously, the greater ϵ the smaller $\mathbb{P}(\boldsymbol{x}_r \in \mathcal{C}_r^\epsilon)$, but even for a relatively large magnitude $\epsilon = 0.01$, there is still more than a fourth of a chance for the assembly to obey $\boldsymbol{x}_r$. The downside of this method is that it is very likely to increase the area of the cone $\mathcal{C}_r$.

The second method consists of implementing a fake prescribed centre $\boldsymbol{x}_f$, so that the centre wanted,

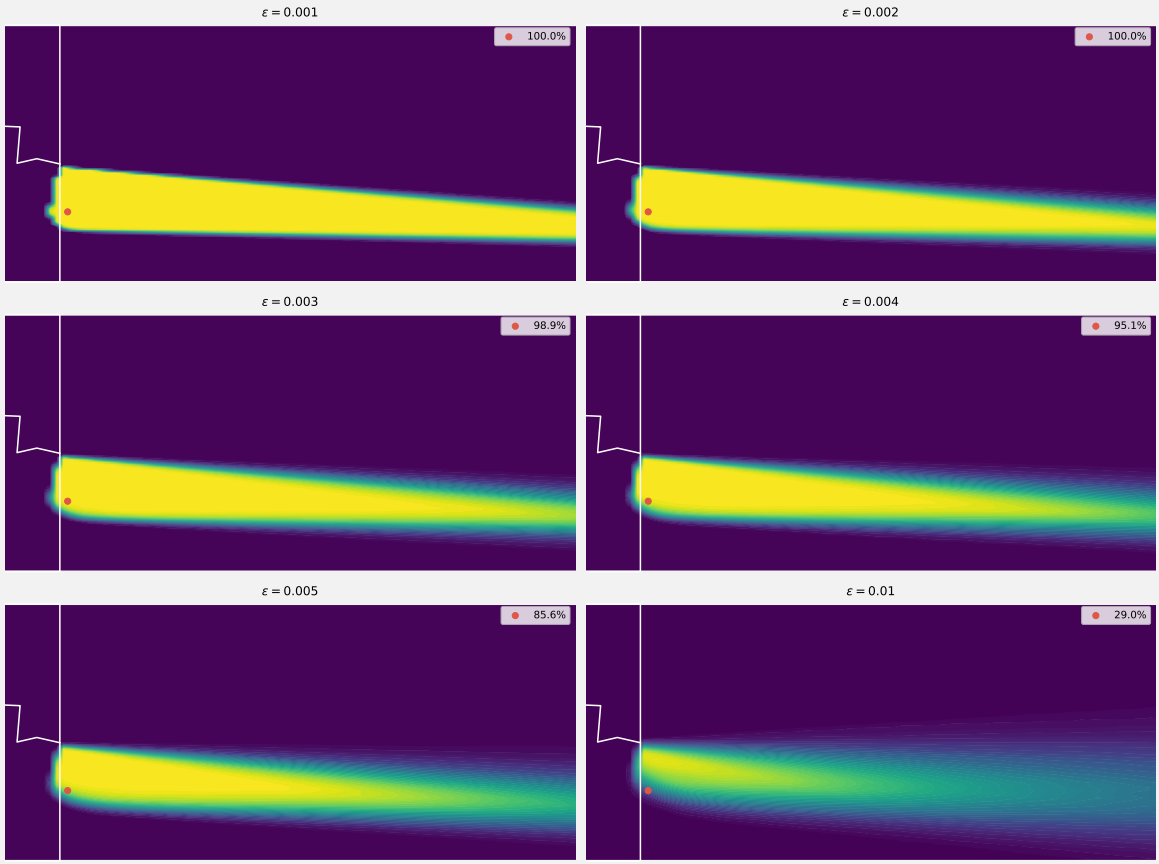


Figure 4.40| Heatmaps of likelihood of obedience for several ϵ after the opening of the cone through the $s\boldsymbol{n}_i\boldsymbol{m}_{p_i} \geq \alpha$ method.

$\boldsymbol{x}_r$, belongs to the interior of the cone of freedom calculated for $\boldsymbol{x}_f$. This supposes having an idea of where to put $\boldsymbol{x}_f$, thus one should know how the heatmap degrades with ϵ. On FIGURE 4.39 one sees that the set of points most likely to be obeyed is up to $\boldsymbol{x}_r$, slightly to the right. So one should set $\boldsymbol{x}_f$ to be below $\boldsymbol{x}_r$, slightly to the left. On FIGURE 4.41, the fake centre $\boldsymbol{x}_f$ is shown in light blue, while the centre $\boldsymbol{x}_r$ is depicted in red. Similar to the previous methods, one sees that the probability $\mathbb{P}(\boldsymbol{x}_r \in \mathcal{C}_r^\epsilon)$ is much higher. The main drawback of this method is that several trials must be done before arriving at a satisfying result.

4.3.5 CONCLUSION - ROBUST OPTIMISATION

SECTION 4.3.1 recalled the salient aspects of the cone of rotational freedom $\mathcal{C}_r$, for us to better understand what it means for a part to obey a point in this cone. It highlighted one crucial aspect of our work: vertices of the cone are critical centres of rotation such that, when performing an infinitesimal rotation around one of them, a vertex of the polyline slides on the interface between the parts. The more constraints meet at a vertex of a cone, the more vertices of the polyline slide. With this in mind, much like we use sensory inputs when inserting a key in a lock, the operator tasked with (dis)assembling two parts, human or robot, greatly benefits from contact information at their interface, if only to further guide the motion. As such, rotating around a vertex of the cone is a relevant heuristic, as at least two vertices of the polyline stay in contact with the other part, the most critical vertex being the user-given centre of rotation $\boldsymbol{x}_r$, as the number of constraints meeting there is at least equal to the number of snap prescribed by the user.

In SECTION 4.3.2 we saw that the greater the number of snaps, the smaller the cone of rotational freedom $\mathcal{C}_r$. It provides a cheap method to shrink the cone to the user-given rotation centre $\boldsymbol{x}_r$, and thus gives the

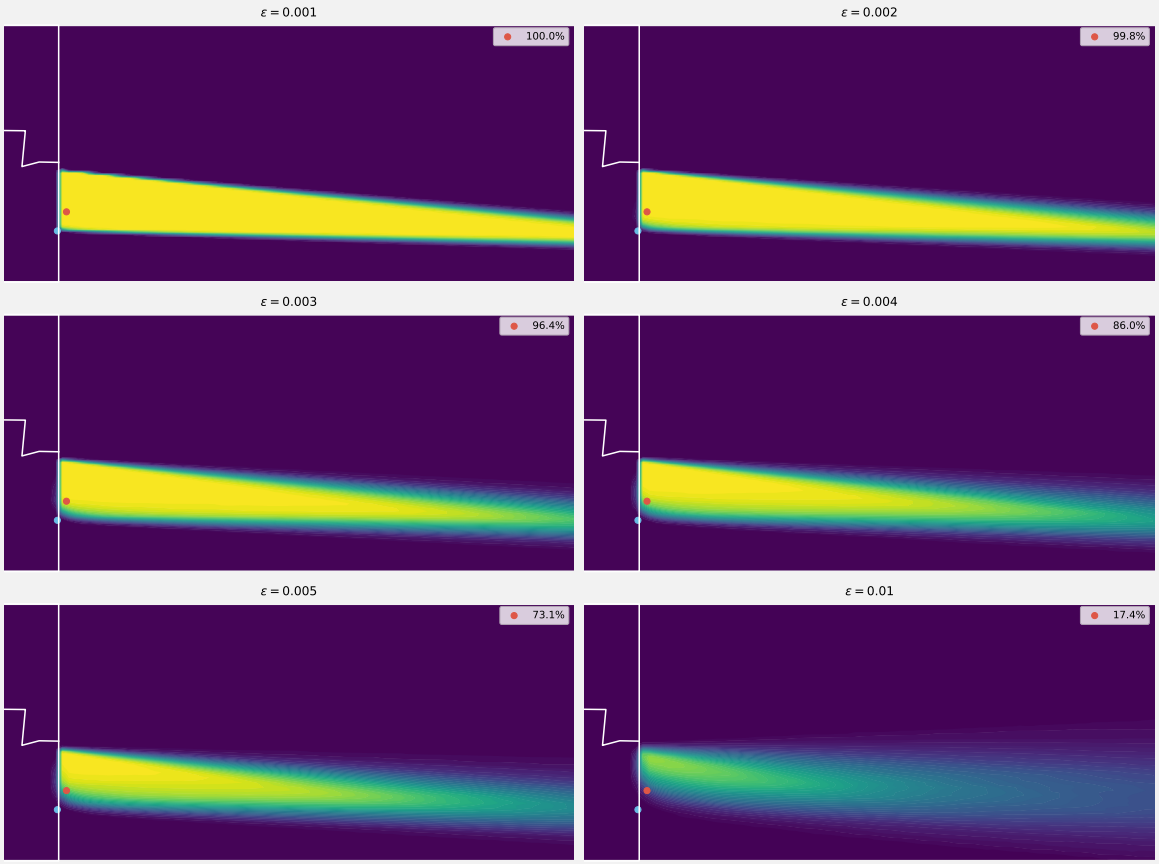


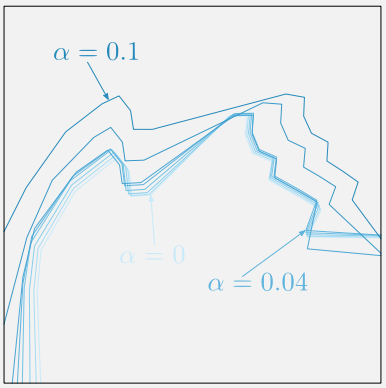
Figure 4.41| Heatmaps of the likelihood of obedience for several ϵ after the opening of the cone through the fake centre method.

user the same amount of control as in translation, where shrinking the cone $\mathcal{C}_t$ to the user given $\boldsymbol{x}_t$ is immediate given exactly two well-oriented snap segments. Since, in addition, we have seen in SECTION 3.1.3 that to assess the interlocking of a multi-parts assembly we must compute the DBGs of the cells of dimension 0 bounding and/or resulting from the intersection of different cones, we see that, because shrinking the cone to $\mathcal{C}_r = \{\boldsymbol{x}_r\}$ for all parts required to rotate ensures that no intersection is possible with any other cone, the number of cells of dimension 0 is minimal, namely equal to the number of parts having to rotate around their prescribed centres (and the DBGs in translation must be checked). Finally and quite importantly, from a practical point of view shrinking $\mathcal{C}_r$ decreases the odds of accidental disassembly as fewer points are valid centres of rotation.

At this step shrinking $\mathcal{C}_r$ to $\{\boldsymbol{x}_r\}$ by increasing the number of snaps seems an excellent idea: by increasing the number of constraints meeting at $\boldsymbol{x}_r$, one multiplies the points giving contact information at the interface between the parts on one hand, and on the other, it reduces the complexity of assessing the interlocking of the assembly. Yet, in SECTION 4.3.3 we understood that if we factor in uncertainties, be it on the location of $\boldsymbol{x}_r^\epsilon$ or on the geometry of the separating polyline $(\boldsymbol{p}_i^\epsilon)_{i\in[\![1,k]\!]}$, the more snap segments, the less likely is the part to obey $\boldsymbol{x}_r$. Indeed, because the snap constraints coincide in $\boldsymbol{x}_r$, any random change dramatically decreases the odd of $\boldsymbol{x}_r$ meeting all of them, making the assembly highly unlikely to obey its prescribed rotation.

To increase the odds of obedience, two strategies were presented in SECTION 4.3.4: either we forget about the snap constraint and, for some $\alpha > 0$ we impose that for all segment i $s\boldsymbol{n}_i \cdot \boldsymbol{m}_j \geq \alpha$ (for $j \in \{i, i+1\}$),

but then we lose control on the shape of the cone $\mathcal{C}_r$, or we introduce a fake centre of rotation, located such that the actual centre $\boldsymbol{x}_r$ lies deep in the region of a high probability of obedience, but this require many trials and errors. Whichever the strategy, they are likely to adversely affect the area of the cone $\mathcal{C}_r$, which may increase the number of base DBGs to calculate, but also means that the separating polyline obeys more rotations: it is less robust to parasitic rotations.



In this concluding section, we offer guidelines to keep the number of DBGs tractable while ensuring that $\boldsymbol{x}_r$ stays likely to be obeyed, even after the introduction of imperfections. We simply mix what we summed up above. First, we increase the number of snap segments until we get $\mathcal{C}_r = \{\boldsymbol{x}_r\}$. Then we take the polyline resulting from this GPA optimisation (in the lightest shade of blue in the inset), and we feed it as input to a new round of GPA optimisation where the snap constraint is deactivated, and the segments are constrained to $s\boldsymbol{n}_i \cdot \boldsymbol{m}_j \geq \alpha$, for some $\alpha > 0$. Since we started from a situation where constraints were met by $\boldsymbol{x}_r$, (i.e. for some indices we had $\boldsymbol{n}_i \cdot \boldsymbol{m}_j = 0$), this new optimisation pushes the boundaries of the corresponding half-planes encoded in $\boldsymbol{A}_r\boldsymbol{x} \star \boldsymbol{b}$ away from $\boldsymbol{x}_r$. Because all other constraints are already met by the input polyline, this round of optimisation stops as soon as $s\boldsymbol{n}_i \cdot \boldsymbol{m}_j \geq \alpha$ for all i, thus the polyline is only slightly modified, especially for small α: on the inset the darker a polyline, the bigger the corresponding α. The values of α are given FIGURE 4.42. Parameter α is the minimal distance at which the boundaries are pushed from $\boldsymbol{x}_r$, it provides thus an indirect tool to manipulate the shape of the cone of freedom $\mathcal{C}_r$.

On FIGURE 4.42, the imperfect cones of freedom $\mathcal{C}_r^\epsilon$ are superimposed for various values of α. In this figure, the standard deviation of the normal law is $\epsilon = 0.002$. Four segments snapped on the initial polyline, thus we have $\mathbb{P}(\boldsymbol{x}_r \in \mathcal{C}_r^\epsilon) \leq \frac{1}{16}$, which is consistent with the observed sampled mean of about 0.04 on the top left subfigure, for $\alpha = 0$. Then, for increasing values of α, the cones $\mathcal{C}_r^\epsilon$ gets broader and broader and $\mathbb{P}(\boldsymbol{x}_r \in \mathcal{C}_r^\epsilon)$ soon reaches high values. In this figure, it seems that a good balance between the size of the cone and the probability that $\boldsymbol{x}_r$ is obeyed is obtained for $\alpha = 0.03$ (and for the estimation of the probability, $\epsilon = 0.002$).

In a nutshell, while ideally we want $\mathcal{C}_r = \{\boldsymbol{x}_r\}$, imperfections impose to have a broader cone. Yet to avoid having too many base DBGs to calculate, a small cone is desirable. Parameter α permits to indirectly manipulate the shape of $\mathcal{C}_r$, thus adjusting its size. We can outline a very simple algorithm, should the user wants all the cone $\mathcal{C}_r$ of the different parts of the assembly to have similar areas, between two thresholds A_{min} and A_{max}: at each iteration, when a polyline is optimised (using the snap constraint, and $\alpha = 0$), the second round of optimisation for $\alpha > 0$ is executed several times, to get to a cone area $A_{min} \leq A \leq A_{max}$ by dichotomy: if the area is too great decrease α; else if too small increase it. By placing the centres of rotation of the different parts sufficiently away from each other, such an algorithm ensures that no two cones intersect, keeping the number of base DBGs low. Worst case scenario, if there is indeed an intersection, simply diminish the value of α for the corresponding polylines until their cones do not touch anymore.

Imperfections on a polyline obeying a translation are much more simple to address: it suffices to slightly increase the angular opening of the cone $\mathcal{C}_t$ to give room to the polyline to giggle while keeping $\boldsymbol{x}_t$ (or $\boldsymbol{x}_t^A$ and $\boldsymbol{x}_t^B$) inside the perturbed cone $\mathcal{C}_t^\epsilon$. It can be done at the first round of GPA optimisation, by simply directly prescribing this slightly bigger cone. Moreover, as intersecting such cones of dimension 1 does not result in more cells of dimension 0 (they keep being the endpoints of the cones) we do not need to give particular care to the DBGs associated with directions of translation. The downside is the same as in rotation: the cone

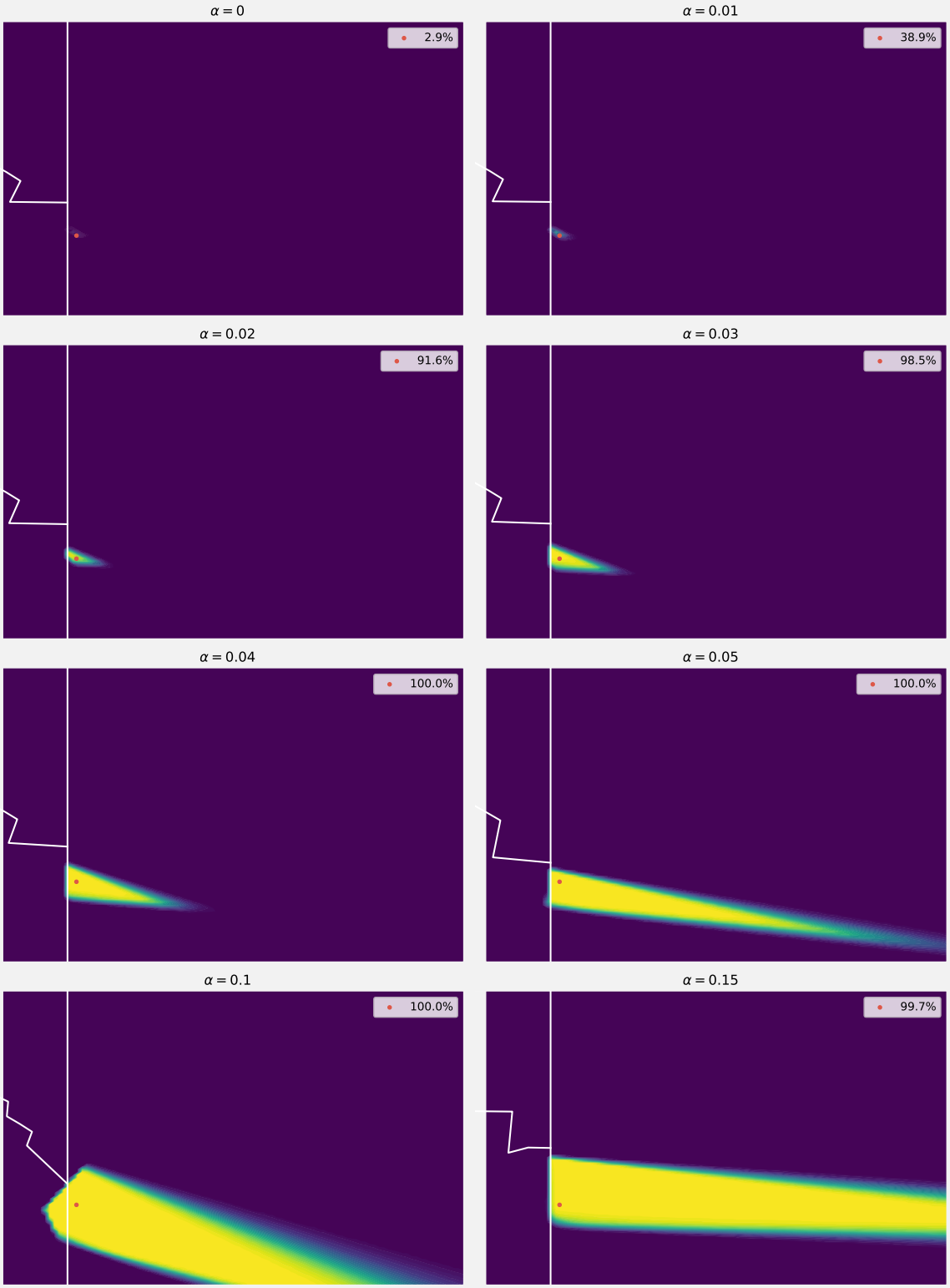


Figure 4.42| Parameter α gives an indirect control over the shape of the cone of rotational freedom $\mathcal{C}_r$.

being greater, the assembly obeys more motions, and is therefore less robust to parasitic rotations: it may become quite easy to disassemble. The user should thus use α to increase the size of the cones of freedom, but not too much as otherwise the assembly would easily break apart.

4.4 BACKLASHES IN ASSEMBLIES

4.4.1 OVERVIEW

A backlash between two imperfect parts refers to the gap between them in the assembled position. Backlashes are a necessity for any real assembly as it is impossible to make parts with perfect geometry.
In the present study, the hypothesis of infinitesimal motion makes it impossible to finely study the consequences of a backlash between two parts. On FIGURE 4.43 three assemblies are shown. On the left column the backlash between the parts is exemplified by the white space between the red part P_0 and the blue part P_1. While the gap is much larger than any realistic backlash, we will understand that because of the infinitesimal motion hypothesis, the actual width of the gap matters not. The middle column shows the parts in an assembled state: part P_1 is translated to the left until touching P_0. As one can see, even in this state, gaps still exist on the upper and lower part of the tenon. On the right, the so-called perfect cones $\mathcal{C}_t$, $\mathcal{C}_r^{ccw}$ and $\mathcal{C}_r^{cw}$ are the cones of freedom associated with a perfect assembly, with no backlash. The so-called backlash cones are calculated taking into account the backlashes on the upper and lower segments of the tenon: because any infinitesimal motion of P_1 cannot lead it to collide with such segments on P_0 (as they are at a *finite* distance from it), any translation and/or rotation can be obeyed by these segments. As a consequence, the cones of freedom can only be calculated with the segments of the interface touching each other. In each of the three cases presented on FIGURE 4.43, only the three vertical segments of the interface can be considered. This leads to backlash cones of freedom of motion much larger than the perfect ones.

Following this theoretical study, backlashes seem to be a major impediment to our work: since any real assembly has backlash leading to large cones of freedom of infinitesimal motion, should our work not be obsolete? Fortunately these theoretical results are of negligible consequences in real life: in a laser cutter the order of magnitude of the kerf (the width of the groove burned by the laser) is 0.1mm = 10^{-4}m. The side of the square design domain of the built assemblies is of about 10cm=10^{-1}m. For such a backlash ratio of about 10^{-3} we find that the physical manipulation of the assembly is coherent with the perfect numerical model (without backlash): for most assemblies the backlash is not perceptible at all, and for a few we can feel that small motions other than the prescribed ones are possible, but as soon as there is an additional contact between the parts, the geometry of the interface naturally guides the motion towards the prescribed one.

In translation, when the cone $\mathcal{C}_t$ is bounded by two distinct vectors $\boldsymbol{x}_t^A$, $\boldsymbol{x}_t^B$ (by opposition to being reduced to a single direction $\boldsymbol{x}_t$) we observe that in a tenon-and-mortise like assembly, it is possible to have the backlash cone equals to the perfect one by having the tenon slightly larger than the mortise. Assuming the angle difference to be greater than the width of the backlash, the two parts will be in contact at exactly two points, ensuring that the backlash and perfect cones match, as show on FIGURE 4.44. The cost of this operation is the stress concentration happening at the contact points.

4.4.2 NUMERICAL STUDY OF THE KINEMATICS OF AN ASSEMBLY WITH BACKLASH

Yet, the fact that built assemblies behave well in practice does not mean that the theoretical kinematics of assemblies with backlash should not be investigated. We present below a succinct study of the cone of freedom of a 2-parts assembly with backlash obeying a *finite* rotation. It is the only time in this manuscript that we stray away from the infinitesimal motion hypothesis.

In this section, we call the polyline of P_i, $i \in \{0, 1\}$, the portion of the boundary of P_i that would have been the separating polyline between the parts had they been perfect, with no backlash. Let $(\boldsymbol{p}_i)_{i\in[\![1,k]\!]}$ be

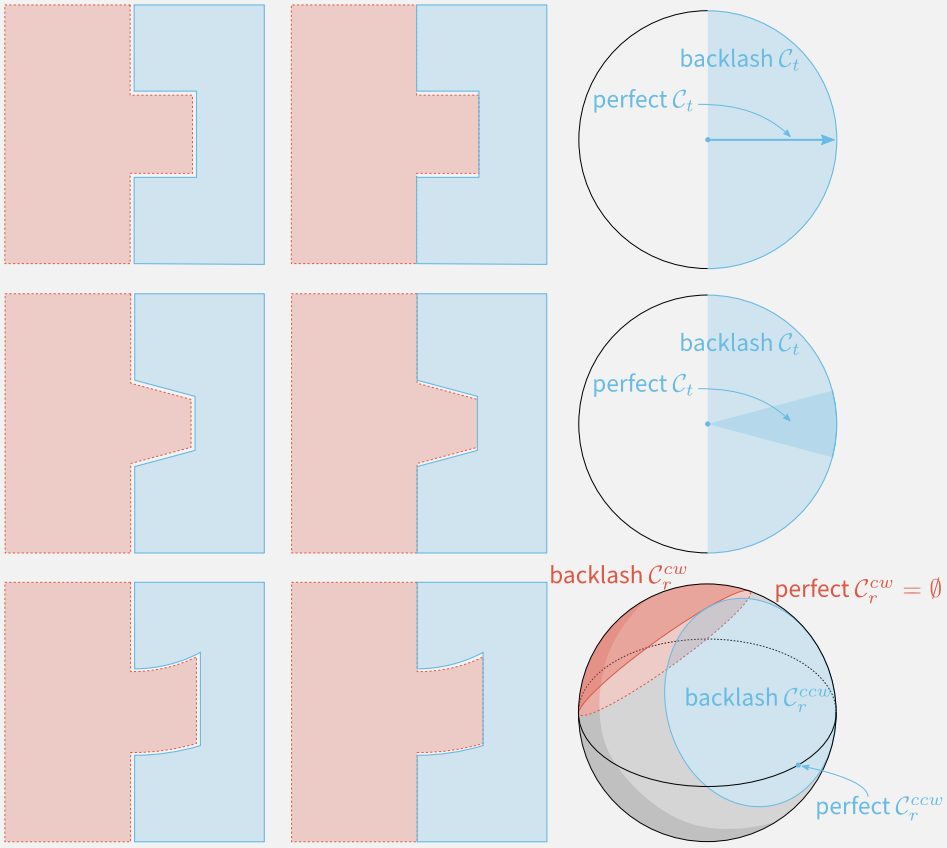


Figure 4.43| Backlashes greatly expand the cones of freedom.

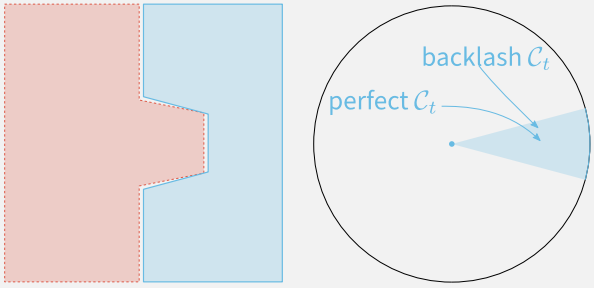


Figure 4.44| By playing with the respective angles of the tenon and mortise we ensure that the backlash cone matches the perfect one.

the polyline of P_1. The original position of a point on the polyline of P_1 refers to its position before any rotation is done: it is $\boldsymbol{p}_i$. Once the rotation is done, the new position of this point is denoted by $\tilde{\boldsymbol{p}}_i$.

4.4.2.1 Constant width backlash

The goal of this section is to find the set of centres of rotation such that a finite rotation around any point in that set does not lead P_1 to collide with P_0. We make two simplifying hypotheses:

- We assume that each point of the polyline of P_1 moves at a constant distance $r > 0$ from its original position while rotating around a given centre $\boldsymbol{x}$. This is a strong assumption: in reality, the length of the trajectory of a point is proportional to its distance to the centre of rotation, and the closer the centre to a point, the more spread the lengths of trajectories of the other points of the polyline. Thus this assumption only becomes reasonable when looking at centres of rotation far away from the polyline.
- While in reality, the trajectory done by a point is a circular arc, we model it by a line segment, tangent to the trajectory at the original position of the point. Mathematically speaking, given a centre of rotation

$\boldsymbol{x}$ each point $\boldsymbol{p}_i$ moves along the instantaneous direction of motion $\boldsymbol{m}(\boldsymbol{p}_i, \boldsymbol{x})$. Being a first-order Taylor expansion of the trajectory, this is a reasonable assumption as in practice the width of the gap between two parts is small compared to the radius of the trajectory and the error made by considering a line segment instead of a circular arc is minute.

To insist on the fact that the motion we model is not a proper rotation (it is not a rigid motion), we will refer to it as "rotation". These two hypotheses mean that given a centre $\boldsymbol{x}$, each point $\boldsymbol{p}_i$ of the polyline of P_1 is moved at

$$\tilde{\boldsymbol{p}}_i = \boldsymbol{p}_i + r\frac{\boldsymbol{m}(\boldsymbol{p}_i, \boldsymbol{x})}{||\boldsymbol{m}(\boldsymbol{p}_i, \boldsymbol{x})||}$$

and denote by $\tilde{P}_1$ the part after the "rotation" is done. Our goal is thus to find the region:

$$\mathcal{R}_\delta = \{\boldsymbol{x} \in \mathbb{R}^2,\ \tilde{P}_1 \text{ does not collide with } P_0\}$$

Parameter δ will be introduced hereunder. Assume in this section that the backlash is of constant width, denoted by Δ. For our "rotation" to possibly lead to a collision with P_0, we must have $r > \Delta$.

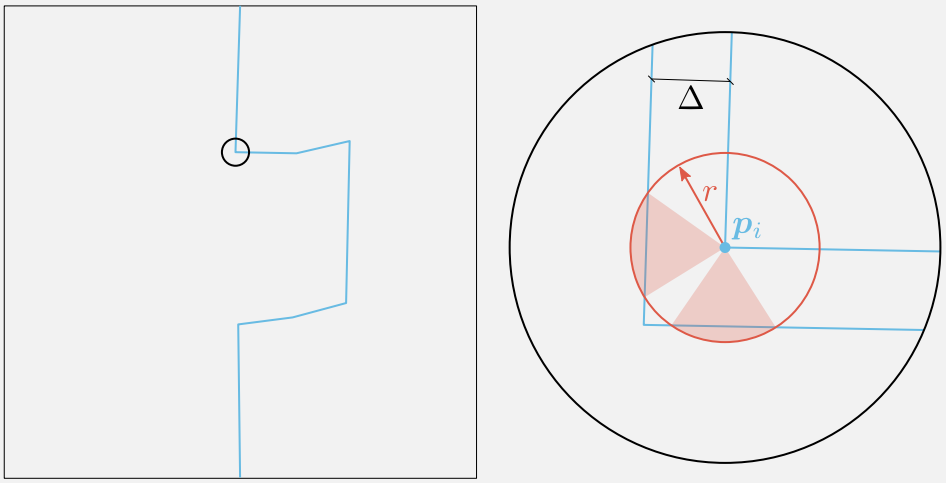
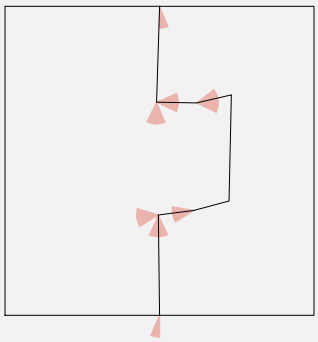


Figure 4.45| Illustration of the notations.

FIGURE 4.45 illustrates the notations we have introduced: the red circle shows the set of possible location of point $\tilde{\boldsymbol{p}}_i$ after a "rotation" of $\boldsymbol{p}_i$. Two cones are highlighted in red: they correspond to the set of positions on which $\tilde{\boldsymbol{p}}_i$ is in the interior of P_0, meaning that $\tilde{P}_1$ collides with P_0. Thus we want to avoid the $\boldsymbol{x} \in \mathbb{R}^2$ such that $\boldsymbol{m}(\boldsymbol{p}_i, \boldsymbol{x})$ is in the interior of one of the two cones. Assuming the rotation to be counterclockwise, this means that $\boldsymbol{x}$ cannot be in the cones obtained by rotating the ones shown on the figure by $\frac{\pi}{2}$ counterclockwise. We call these rotated cones the *forbidden cones*, noted $\mathcal{C}_{i\delta}$ for each vertex $\boldsymbol{p}_i$. Note that the forbidden cones only depend of the ratio $\delta = \frac{r}{\Delta}$. They are shown for all $(\boldsymbol{p}_i)_{i\in[\![1,k]\!]}$ on the inset. As seen on this figure, several cases may occur, ranging from $\mathcal{C}_{i\delta} = \emptyset$ to $\mathcal{C}_{i\delta}$ is constituted of several cones.



Thus, we want to find the set of points $\boldsymbol{x}$ such that none of the $(\boldsymbol{m}(\boldsymbol{p}_i, \boldsymbol{x}))_{i\in[\![1,k]\!]}$ lie in a cone $\mathcal{C}_{i\delta}$. This is therefore given by:

$$\mathcal{R}_\delta = \left\{\mathbb{R}^2 \setminus \bigcup_{i=1}^{k} \mathcal{C}_{i\delta}\right\}$$

Sets $\mathcal{R}_\delta$, for various δ, are shown in blue on FIGURE 4.46. They are not convex, not even necessarily con-

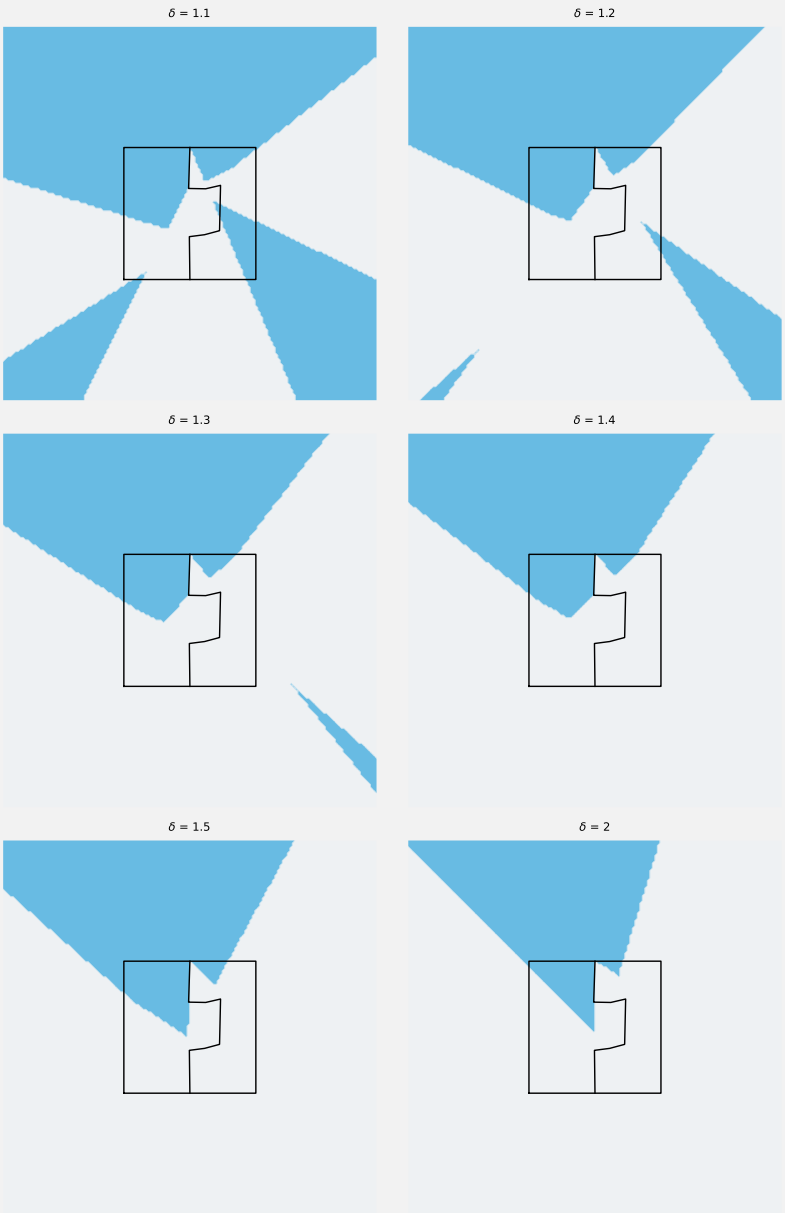


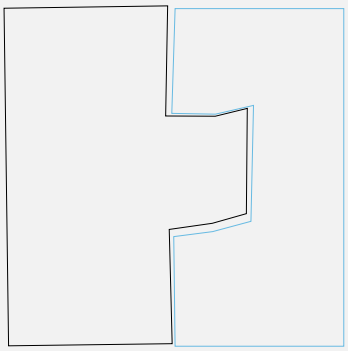
Figure 4.46| Several regions $\mathcal{R}_\delta$, for various values of δ.

nex. But recall that our first assumption (all $\boldsymbol{p}_i$ move by a constant value $r = \delta\Delta$) is highly unrealistic near the polyline of P_1; thus, the shape of each region $\mathcal{R}_\delta$ in the vicinity of the separating polyline should be taken with a grain of salt. Far away from the polyline, $\mathcal{R}_\delta$ becomes much more realistic and means that during the disassembling process, the operator (robot or human) can rotate P_1 around any centre of rotation $\boldsymbol{x} \in \mathcal{R}_\delta$ by an angle $\theta \simeq \frac{r}{||\boldsymbol{p}_i - \boldsymbol{x}||}$, assuming $\boldsymbol{x}$ to be sufficiently far from the $(\boldsymbol{p}_i)_{i\in[\![1,k]\!]}$ so that the distances $||\boldsymbol{p}_i - \boldsymbol{x}||$ are all close to each other.

The pendant map obtained for clockwise rotations (the rotation to obtain the forbidden cones is clockwise and not counterclockwise) would show the region of the plane compatible with a finite rotation aimed at assembling the parts. Such maps may be useful in the context of a cluttered assembling space where the operator would benefit from choosing the centre of rotation to operate from.

4.4.2.2 Varying width backlash

In the previous section, we assumed the gap between the part to be of constant width Δ. In practice, in the absence of contact information, it seems nearly impossible to neatly place the parts as such. In this section, we take a more realistic stance by randomly moving P_0 by a small amount so that the width of the gap is not constant, as illustrated on the inset. The goal is now to find the set $\mathcal{R}_\delta$ of points *likely* to be obeyed given the imprecision on the location and orientation of one part with respect to the other.



The process is the same as before: we find the set of forbidden cones and deduce $\mathcal{R}_\delta$ by excluding them from $\mathbb{R}^2$. FIGURE 4.48 shows the probability heatmap of a point to be obeyed for a finite rotation given

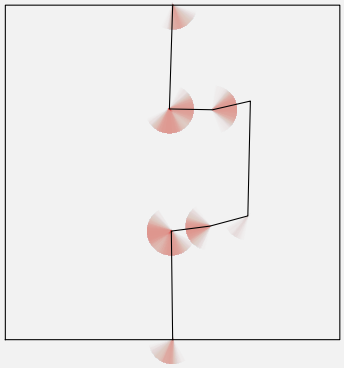


Figure 4.47| Forbidden cones, averaged for various random positioning of a part with respect to the other, for $\delta = 1.1$.

imprecise respective placements of the part. In bright yellow, the probability is 1, and in deep purple 0. Rotating around points highlighted in yellow means that P_1 is very likely not to collide with P_0 for a finite "rotation". Again, the closer a possible centre of rotation to the polyline, the less accurate the heatmap.

4.5 INTERPOLATION BETWEEN 2-PARTS ASSEMBLIES

We leave the study of imperfections, backlashes and robust optimisation to enter the realm of smooth interpolation between assemblies. Given two generated assemblies, the aim is to find the set of assemblies bearing varying amount of resemblance to either of the two.

4.5.1 THEORY OF ELASTIC DEFORMATION OF OPEN PLANAR CURVES

This section aims at briefly summing up theory of elastic deformation of open planar curves, presented in [93], and developed *in extenso* in the book [92], especially in chapters 4 and 5 to which the curious reader is referred for more in-depth explanations. Indeed, technical explanations will be glossed over as we focus more on conveying the main ideas of the approach, taking close inspiration from [11] who neatly synthesise the main results. The open curve assumption is important: the mathematical process is much more complex for closed polylines and is not presented here.

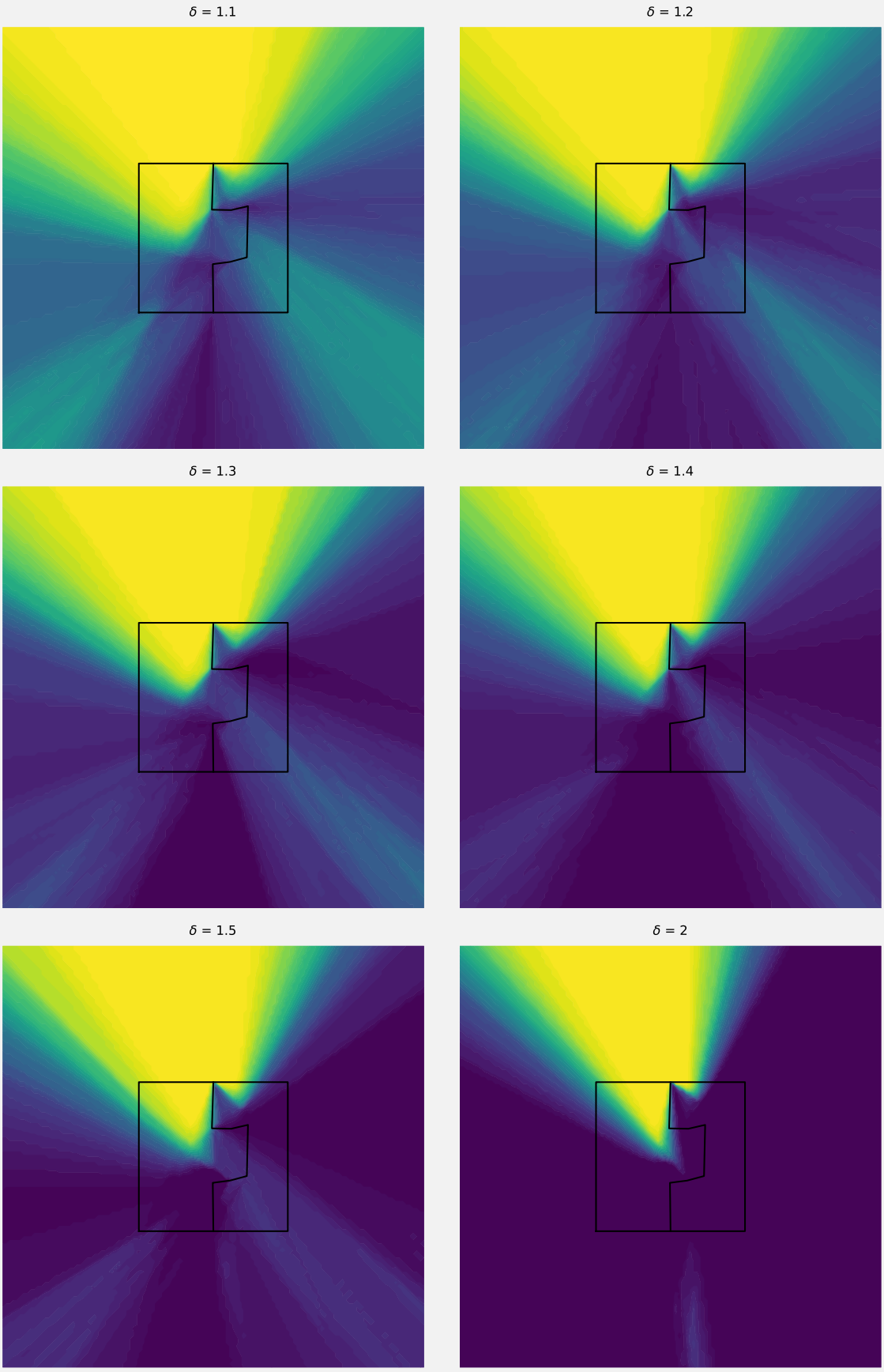


Figure 4.48| Several regions $\mathcal{R}_\delta$, for various values of δ.

Consider an open curve parametrised in the form:

$$\begin{array}{rccc} \beta & : & [0,1] & \longrightarrow & \mathbb{R}^2 \\ & & t & \mapsto & \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \end{array}$$

As the parameter t varies in its domain, the point $\beta(t)$ traces a path from $\beta(0)$ to $\beta(1)$. This is called a **parametrisation** of β, it dictates the rate at which the curve is traced, [92]. More parametrisations are possible: any orientation-preserving (meaning $\dot{\gamma} > 0$ on its domain) diffeomorphism $\gamma : [0,1] \to [0,1]$ defines a **reparametrisation** of β through the composition $\beta \circ \gamma$, see FIGURE 4.49.

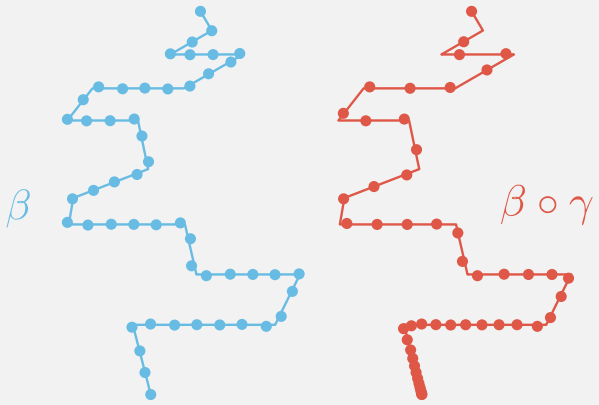


Figure 4.49| Two parametrisations of the same curve.

We are interested in the shape of the curve, thus the results should be invariant to shape-preserving transformations (rigid motions, scaling and reparametrisation). These nuisances are listed below:

- its location in space; let $\mathbb{R}^2$ denote the set of planar translations.
- its orientation in space: let $SO(2)$ denote the set of planar rotations.
- its scale; let $\mathbb{R}^{+*}$ denote the set of scale parameters.
- its parametrisation; let $\Gamma = \{\gamma : [0,1] \to [0,1], \ \gamma \text{ is an orientation-preserving diffeomorphism}\}$.

Let $[\beta]$ be the orbit of β, i.e. the set of curves with the same shape as β but with different transformations in space:

$$[\beta] = \left\{\sigma O(\beta \circ \gamma) + x, \ \sigma \in \mathbb{R}^{+*}, O \in SO(2), \gamma \in \Gamma, x \in \mathbb{R}^2\right\}$$

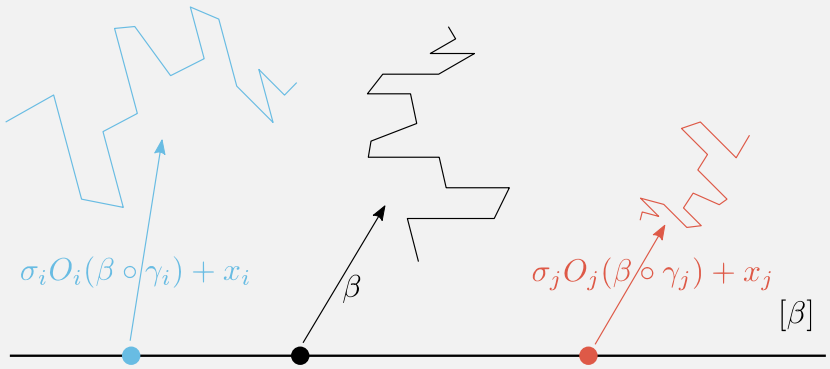


Figure 4.50| Cartoon drawing of the orbit of a curve.

As our goal is to smoothly interpolate between two given curves β_1 and β_2, it is necessary to know the

distance between these curves. We can consider for instance the usual $\mathbb{L}^2$ distance defined as:

$$d(\beta_1, \beta_2) = \left(\int_0^1 ||\beta_1(t) - \beta_2(t)||^2 dt \right)^{\frac{1}{2}}$$

Where $||\cdot||$ is the usual euclidean distance in $\mathbb{R}^2$, and we remark that in general $d(\beta_1, \beta_2) \neq d(\beta_1 \circ \gamma, \beta_2 \circ \gamma)$, making this metric not invariant to reparametrisation: we lose some information on the shape of the curves when using the $\mathbb{L}^2$ distance to compare them. Srivastava and coauthors, [92, 93], introduce the so-called **square-root velocity function** (SRVF) of a curve β as

$$q(t) = F(\dot{\beta}(t)) = \begin{cases} \frac{\dot{\beta(t)}}{\sqrt{||\dot{\beta(t)}||}} \text{ if } \dot{\beta}(t) \neq 0 \\ 0 \text{ otherwise} \end{cases}$$

F being continuous, if β is continuous, then q is square-integrable, it belongs to the space $\mathbb{L}^2([0,1], \mathbb{R}^2)$. Geometrically, it is a scaled velocity vector, tangent to the curve at each point. Curve β can be reconstructed (up to a translation) from q using $\beta(t) = \int_0^t q(\tau)||q(\tau)||d\tau$, showing that the SRVF is translation invariant. For a given $\gamma \in \Gamma$, the SRVF of a parametrised curve $\beta \circ \gamma$ is given by $(q \circ \gamma)\sqrt{\dot{\gamma}}$, and for two SRVFs $q_1, q_2 \in \mathbb{L}^2([0,1], \mathbb{R}^2)$ one can verify that

$$d(q_1, q_2) = d\left((q_1 \circ \gamma)\sqrt{\dot{\gamma}}, (q_2 \circ \gamma)\sqrt{\dot{\gamma}} \right)$$

And the $\mathbb{L}^2$ metric is invariant to reparametrisation. To remove the scaling variability, all curves β are rescaled to be of unit length; as one notices, one has

$$d(q, q)^2 = \int_0^1 ||q(t)||^2 dt = \int_0^1 ||\dot{\beta}(t)||dt = 1$$

which means that the SRVF of a unit-length curve is an element of $\mathcal{S}$, the unit sphere in the Hilbert manifold $\mathbb{L}^2([0,1], \mathbb{R}^2)$, which will prove to be quite handy to easily interpolate between curves. Being translation and scale invariant, one defines the orbit of q as:

$$[q] = \left\{ O(q \circ \gamma)\sqrt{\dot{\gamma}},\ O \in SO(2), \gamma \in \Gamma \right\}$$

Given two SRVFs $q_1, q_2 \in \mathbb{L}^2([0,1], \mathbb{R}^2)$, let

$$\rho(q_1, q_2) = \arccos \langle q_1, q_2 \rangle = \arccos \int_0^1 q_1(t) \cdot q_2(t) dt$$

be the distance between q_1 and q_2. It corresponds to the geodesic distance (angle) between the two points q_1 on q_2 on the sphere $\mathcal{S}$; it is invariant to reparametrisation and rotation. The shortest distance between two shapes is given by finding the closest points on each orbit:

$$\rho_s([q_1], [q_2]) = \min_{(O,\gamma) \in SO(2) \times \Gamma} \arccos \langle q_1, O(q_2 \circ \gamma)\sqrt{\dot{\gamma}} \rangle$$

Let $q_2^* = O^*(q_2 \circ \gamma^*)$ be the optimal point given by the minimisation above. Denoting $\nu = \rho(q_1, q_2^*)$ one defines the shortest arc between these two points using the standard formula for interpolating on a sphere:

$$\alpha^*(s) = \frac{1}{\sin \nu}(q_1 \sin(\nu(1-s)) + q_2^* \sin(\nu s))$$

for $0 \leq s \leq 1$. Details of the optimisation are given in [92]. We have been using the python package `fdasrsf`.

4.5.2 RESULTS

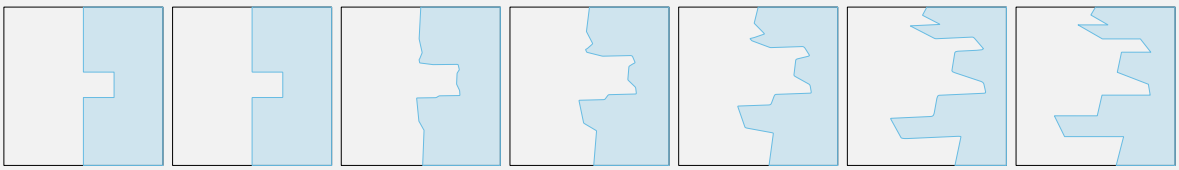


Figure 4.51| Interpolation between two designs

FIGURE 4.51 shows the decoding of regularly placed points on the geodesic α^*, traced between the two end designs: one sees that the separating polyline is smoothly interpolated. But this figure hides that we cheated: because the interpolation is invariant to translation, we do not know where to put the polyline in the design domain. On FIGURE 4.51, the polylines were arbitrarily placed so that their first point is on the design domain above 0 (the design domain being centred in 0). But we cannot interpolate between polylines at different locations in space, without placing them arbitrarily in the design domain. The scale invariance is problematic too: for the interpolation to happen, all points must be on the unit sphere of $\mathbb{L}^2([0,1],\mathbb{R}^2)$, which means, as we saw earlier, that the curves must be of unit length. Obviously, on FIGURE 4.51, the two end polylines have very different lengths: they were scaled so that their endpoints lie on the edges of the design domain. Another issue occurs because of the invariance to rotation: as shown on FIGURE 4.52, top row, the interpolation between the two end polylines yielded the ones in red; they are obviously out of the design domain and cannot be considered as separating curves. On the bottom row, while all interpolated polylines do belong to the design domain, the invariance to rotation leads the penultimate (on the right, highlighted in red) to be at a different orientation from the last one, whereas the shapes of the polylines are almost identical. This leads to different assemblies, not only from the kinematic point of view as the penultimate one does not obey the same motion as the last one, but also from a mechanical point of view, as in this case the geometry but also the location and orientation in space of the separating polyline play an important role.

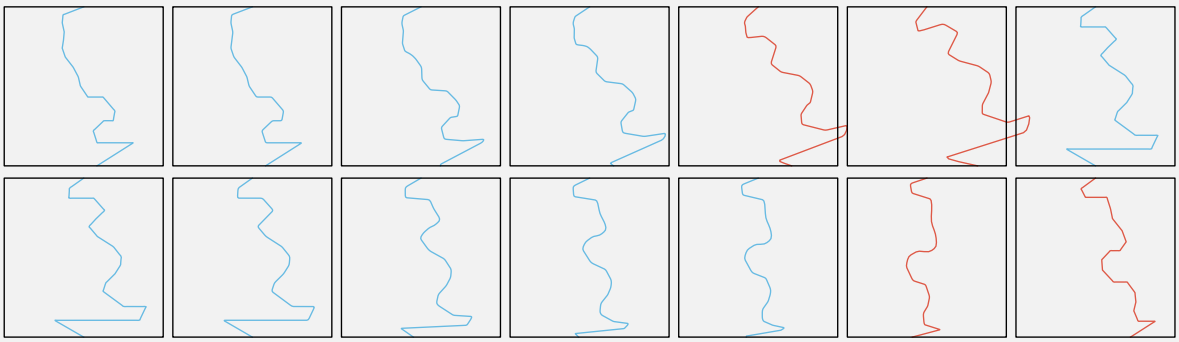
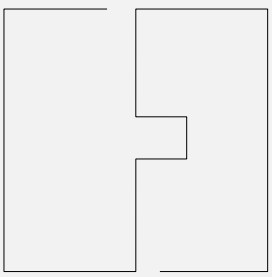


Figure 4.52| While the separating polyline are similar in shape, their different orientation leads to dramatically different assemblies.

To correct for the three invariances (in translation, scale and rotation) we consider an open curve, such as the one on the inset: the design domain is merged with the separating curve, and small segments near the extremities of the polyline are removed, to ensure that analysed curve stays open. Because the vertices of the design domain are at a fixed location and orientation in space, such an open curve encodes enough information to locate, scale and orient unequivocally the interpolated designs.



Moreover, the interpolation is not constrained to meet the kinematic constraint of a separating curve: the motion prescribed by the user may not always be the one obeyed by an interpolated polyline. To correct that, the GPA optimisation must be executed, where each interpolated polyline is taken as the input of the algorithm. FIGURE 4.53 shows three interpolations between the end designs of each row. In blue is the output of the interpolation, and in red is the result of the GPA optimisation. One sees a close match between each pair of polylines, the main changes happening at the snap segments. It can also be noticed that the invariances in scale, rotation and translation are now taken care of, even before the GPA optimisation, which shows that one can successfully interpolate between 2-parts assemblies.

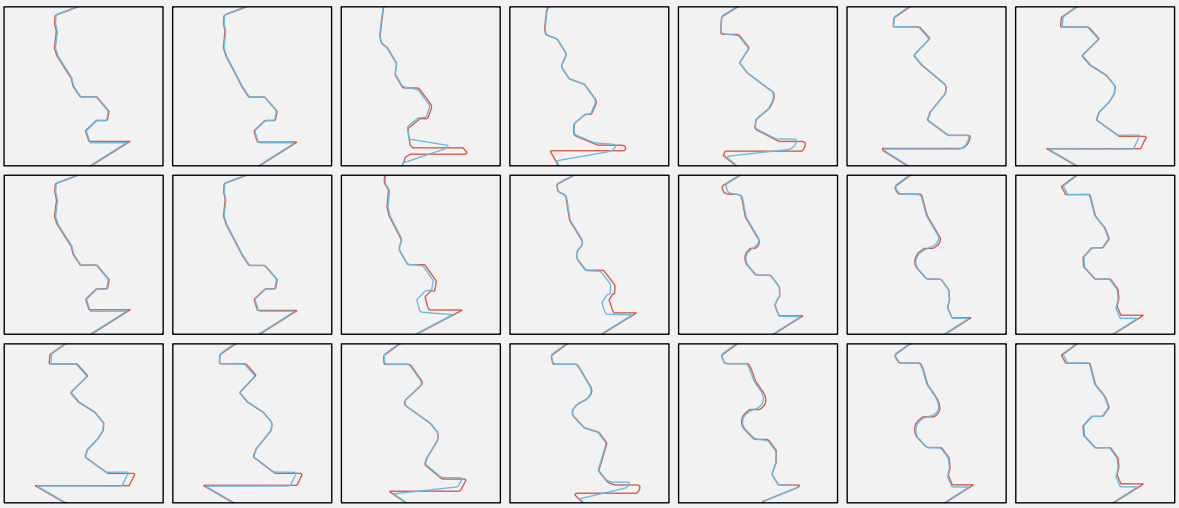


Figure 4.53| The GPA slightly changes the shapes of the interpolated polylines.

The three different end designs are chosen among those dominating the reference in the 95% stresses plane, see FIGURE 4.33. A numerical analysis (see SECTION 4.2) aimed at calculating the 95% stresses in each part is carried out, whose results are shown in FIGURE 4.54. The three highlighted points in red correspond to the 95% stresses of the three designs at the end of the geodesics. Points in blue show the result for the interpolated designs. The polyline in blue is a circuit to help see the trajectory made by the stresses when one smoothly interpolates between the three designs. The 95% stresses of each interpolated design live closely to every other, which shows that we are able to populate the region of the stress plane dominating the reference. In other words, we can find novel designs with good mechanical behaviour in the vicinity of some given designs without much computational effort.

Another promising possibility of this interpolation method lies in the exploration of variations of existing traditional assemblies. For instance on FIGURE 4.55 the triangle consisting of a dovetail joint (top right), a gooseneck joint (bottom) and an abutted gooseneck joint (top left) is regularly interpolated.

While not implemented during the course of this PhD, we think it possible to interpolate between more-than-2-parts assemblies by taking inspiration from Wang *et al.* in [103] who, in particular, interpolate between models of plant roots seen as a collection of open curves with a constraint on the location of one of their endpoints, exactly like a multi-part polygonal assembly is nothing else but a collection of separating polylines each with at least one end constrained to be on another.

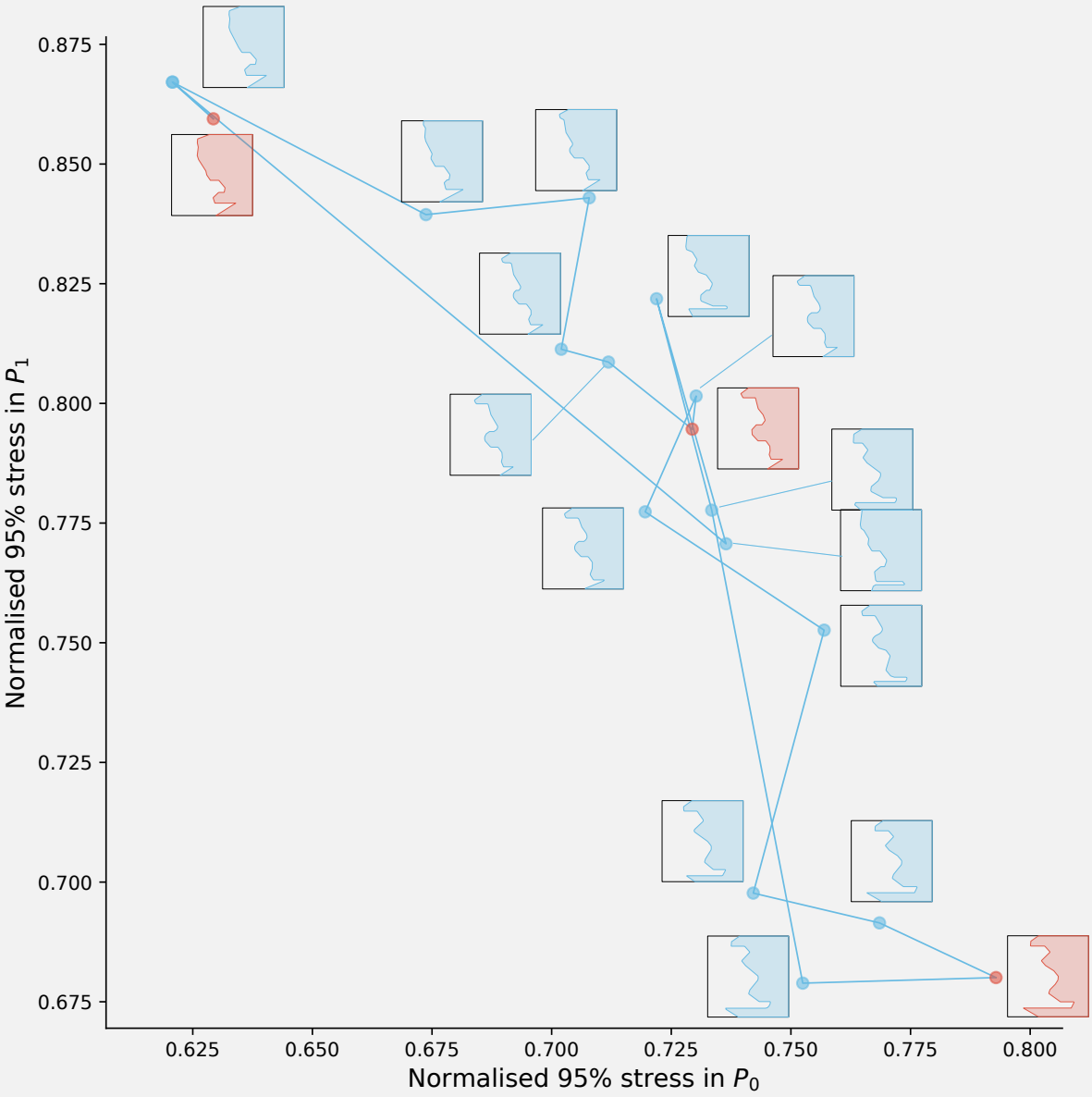


Figure 4.54| The interpolated designs behave similarly to each other. The normalisation is done by the reference design shown in red in FIGURE 4.33.

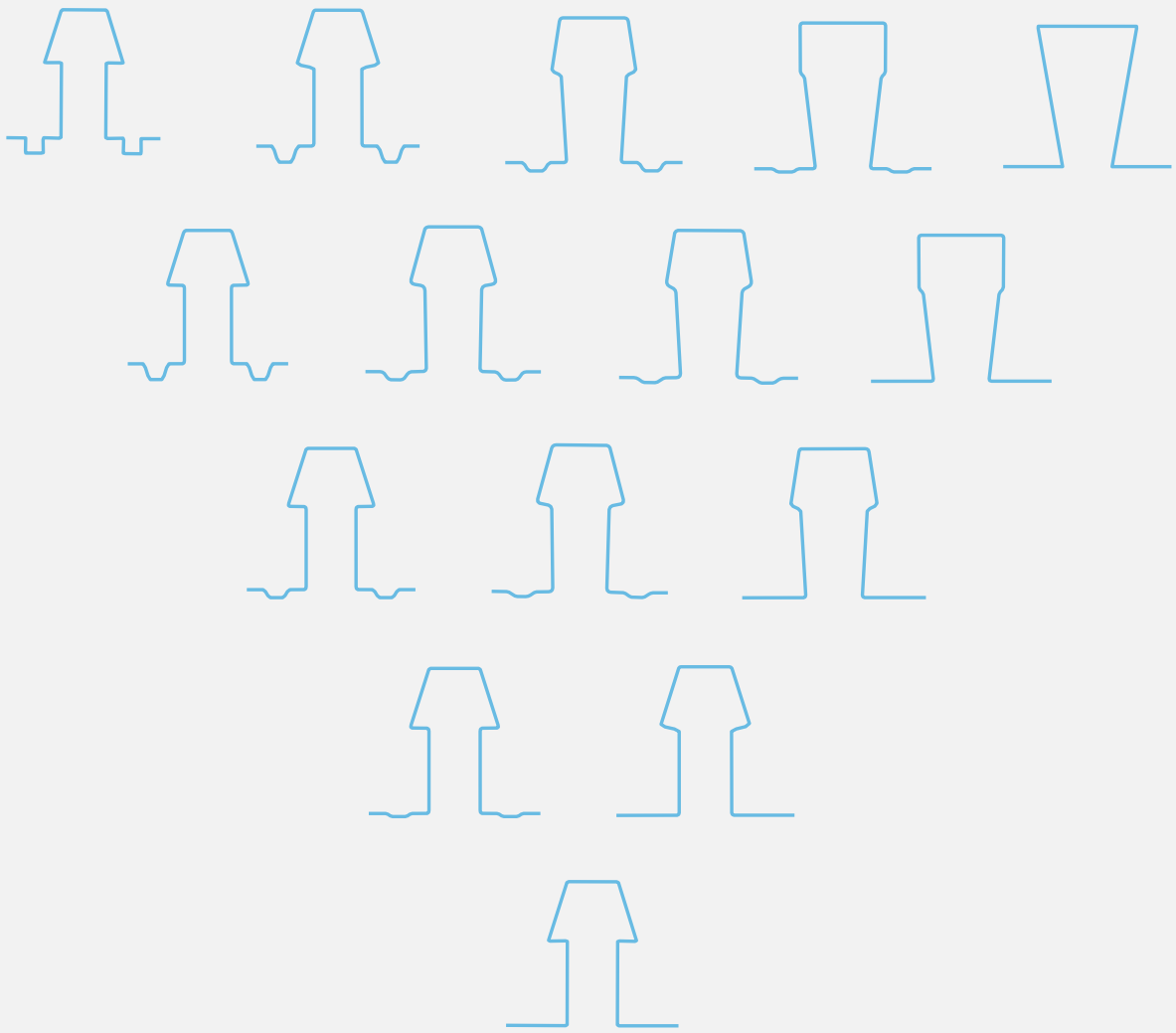


Figure 4.55| Exploration of the variations between three traditional assemblies.

4.6 CONCLUSION

This section concludes what probably is the most important chapter of this manuscript. Indeed, this chapter dealt with the central concepts, tools and properties to automatically generate sequential interlocking assemblies. As we will see in CHAPTER 5, much of what we have used and created in this chapter to generate 2D assemblies is scalable to 3D without much effort. In SECTION 4.1.3 we first talked about the `Turtle` and the Markov process to generate a polyline obeying user-given motions and explained that it ended up in a dead-end: cases happened where the `Turtle` was inescapably drawn to the centre of rotation $\boldsymbol{x}_r$ until no more motion were possible, and the total lack of control on the overall shape of the polyline often led to degenerate designs. Yet, we decided to include this part of our research in this manuscript because we feel that it explains at an intuitive level the constraints that must be met by a polyline to obey a motion. A better approach was then introduced in SECTION 4.1.2, in the form of the **Guided Projection Algorithm** (GPA). This method let the user choose the amount of control he/she wants to impose on the final design, as it is relatively easy to add new constraints. The `Turtle` approach was not completely discarded as a simplified version may be used to generate the first instance of an almost-solution polyline. This approach proved to be powerful and can quickly generate a 2-parts assembly. SECTION 4.1.3 explained that generating a multi-parts assembly is easy, one needs only to successively generate a 2-parts assembly in the remaining part P_0. What is more difficult is to generate an *interlocking* assembly.

While in this PhD we simply observed a simple heuristic to increase the odds of interlocking by judiciously placing two snap segments on the polyline, a weakness of our work lies in the lack of a more sophisticated method to ensure the interlocking. We can cite Wang *et al.* in [105] who, prior to the generation of a part $P_i, i > 1$, builds the DBGs by introducing the vertex P_i and finds the most favourable set of graph edges to ensure the interlocking across all graphs while minimising the shape complexity of the would-be part P_i. The creation of part P_i is then guided by the blocking relationships encoded by the edge, and according to the author, this method successfully increases the odds of creating an interlocking assembly. As such, future work should try to add this kind of algorithm to our approach. Another route for future research would be to investigate the role of friction in the creation of the DBGs. Indeed, at this step the information encoded by each graph is binary: either P_i is blocked by P_j, or not, and all depends on the signs of the dot products of the normals $\boldsymbol{n}_i$ and the instantaneous direction of motions, be it $\boldsymbol{m}_{p_i}$ in rotation or $\boldsymbol{x}_t$ in translation. In real life, for assemblies we built, we find that sometimes a part P_i may push P_j out of the way, despite being blocked by it in theory: the segments of the polyline that are deemed to be blocking are oriented in such a way that in practice the parts may slide onto each other; there is not enough friction for the interlocking to happen in practice. To correct for that we suggest considering the standard Mohr-Coulomb law of friction when building the DBGs (see SECTION 3.1.2.2): not only the sign but also the actual values of the dot products should be used: if all dot products of the same sign sum over a given threshold, then we can consider that the blocking happens in practice.

SECTION 4.2 justifies why we bother at creating assemblies with novel, random, shapes. For the numerical mechanical model used, as well as for the choice of metric (the 95% stresses in both parts of a 2-parts assembly), we find indeed that a random sampling of the design space of polygonal assemblies very often yields designs strictly dominating human-drawn references, thus hinting at the potential of our approach for real-life assemblies. As studying the mechanical aspect of an assembly was not the primary goal of this PhD, it is obvious that many aspects of our work are to be improved from that point of view. In our opinion, future work shall be oriented in two ways. First, a more sophisticated (and faster!) numerical model should be implemented to better assess the mechanical properties of the assemblies. Second, an optimisation scheme should be found to modify the geometry of an assembly for it to behave better under loads. If a

relationship can be established between some geometrical features of the polyline and good mechanical properties, then the GPA optimisation can be used with the additional constraints to create such features. Another possibility would be to enrich the Markov process by adding a state such that the `Turtle` automatically draws at once these features.

In our opinion, SECTION 4.3 deals with the most intriguing geometrical aspects of an assembly obeying rotations. After giving a better understanding of the cone of rotational freedom, this section shows that by playing with the number of snap segments, the user may reach the same amount of control on the cone of rotational freedom as he/she enjoys with the cone of translational freedom: for both motions, the cones may be shrunken precisely to the prescribed motions: $\mathcal{C}_r = \{\boldsymbol{x}_r\}$ and $\mathcal{C}_t = \{\boldsymbol{x}_t\}$ (or $\mathcal{C}_t$ is precisely bounded by the vectors $\boldsymbol{x}_t^A$ and $\boldsymbol{x}_t^B$ if the user asks for a non-degenerate cone). Shrinking the cones to singletons is desirable for two reasons: it gives sensory feedback when (dis)assembling the pieces and dramatically reduces the number of base DBGs needed to be calculated to assess the interlocking of an assembly. Yet this comes at a significant cost: (dis)assembling may become impossible when introducing imperfections. This section concludes by giving guidelines for the polyline to have both a small cone $\mathcal{C}_r$ and a high probability of obeying its prescribed centre $\boldsymbol{x}_r$ in the presence of imperfections, by optimising first with the snap constraint on, then by ensuring that all $s\boldsymbol{n}_i \cdot \boldsymbol{m}_{p_j} \geq \alpha$, where $\alpha > 0$ is the smallest distance between $\boldsymbol{x}_r$ and the boundaries of the half-planes of the constraints.

SECTION 4.4 carries the modelling of imperfections in another direction: it investigates the consequences of a backlash between two parts in an assembly. We quickly reached the conclusion that one of our central hypothesis, the fact that all motions considered are infinitesimal, prevents us from inquiring further in a meaningful way: we saw that backlashes greatly expand the cones of freedom of motion, but also that in practice it does not seem to matter much. Yet, we tried to investigate the issue in a theoretical way by looking at finite motions, at the cost of making crude assumptions on the "rotation" considered. Bearing this in mind, we found regions of the plane that were likely to be obeyed for a finite "rotation", even when the parts are not precisely positioned with respect to each other. Considering finite motion is an active research subject, extending well beyond the scope of this PhD, and as such, we leave to future work the making of a more sophisticated model.

This chapter finishes with SECTION 4.5 where the theoretical background of the elastic deformation of a planar open curve is presented to explain how one may smoothly interpolate between two assemblies. While interpolation helps in finding new designs with good mechanical properties, it has to be said that the main reason behind this section is simply because it is fun: it is extremely pleasing to trace geodesics in the $\mathbb{L}^2([0,1],\mathbb{R}^2)$ space, to observe the polylines getting morphed into one another and to understand the mathematical theory behind it (even if much more work needs to be done to fully master these concepts).

CHAPTER 5 will be relatively brief: we will simply understand how to adapt the tools developed in this chapter for them to generate 3d assemblies.

REFERENCES

11 Karthik Bharath and Sebastian Kurtek. "Analysis of shape data: From landmarks to elastic curves". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 12 (Jan. 2020).

14 Jorge M. Branco and Thierry Descamps. "Analysis and strengthening of carpentry joints". In: *Construction and Building Materials* 97 (2015). Special Issue: Reinforcement of Timber Structures, pp. 34–47. ISSN: 0950-0618. URL: https://www.sciencedirect.com/science/article/pii/S0950061815006029.

15 Jorge Manuel Branco, Maurizio Piazza, and Paulo J.S. Cruz. "Experimental evaluation of different strengthening techniques of traditional timber connections". In: *Engineering Structures* 33.8 (2011), pp. 2259–2270. ISSN: 0141-0296. URL: https://www.sciencedirect.com/science/article/pii/S0141029611001490.

16 Matthias Braun et al. "Calibration and Validation of a Linear-Elastic Numerical Model for Timber Step Joints Based on the Results of Experimental Investigations". In: *Materials* 15.5 (2022), p. 1639.

31 Paul A Gagniuc. *Markov Chains: From Theory to Implementation and Experimentation*. ISBN: 978-1-119-38759-6 978-1-119-38755-8. Hoboken, NJ, USA: John Wiley & Sons, Inc., June 22, 2017. (Visited on 09/17/2021).

36 Pierre Gilibert, Romain Mesnil, and Olivier Baverel. "Rule-based generative design of translational and rotational interlocking assemblies". In: *Automation in Construction* 135 (2022), p. 104142. ISSN: 0926-5805. URL: https://www.sciencedirect.com/science/article/pii/S0926580522000152.

64 Julieta Moradei et al. "Structural Characterization of Traditional Moment-Resisting Timber Joinery". In: (July 2018).

69 Robin Oval et al. "Feature-based Topology Finding of Patterns for Shell Structures". In: *Automation in Construction* (Feb. 2019).

75 Maria Parisi and Maurizio Piazza. "Mechanics of Plain and Retrofitted Traditional Timber Connections". In: *Journal of Structural Engineering-asce - J STRUCT ENG-ASCE* 126 (Dec. 2000).

78 Gilibert Pierre, Mesnil Romain, and Baverel Olivier. *Rule-based generative design of translational and rotational interlocking assemblies*. Version v0. Aug. 2021. URL: https://doi.org/10.5281/zenodo.5158465.

92 Anuj Srivastava and Eric Klassen. *Functional and Shape Data Analysis*. Jan. 2016. ISBN: 978-1-4939-4018-9.

93 Anuj Srivastava et al. "Shape Analysis of Elastic Curves in Euclidean Spaces". In: *IEEE transactions on pattern analysis and machine intelligence* (Sept. 2010).

96 Chengcheng Tang et al. "Form-Finding with Polyhedral Meshes Made Simple". In: *ACM Trans. Graph.* 33.4 (July 2014). ISSN: 0730-0301. URL: https://doi.org/10.1145/2601097.2601213.

103 Guan Wang et al. "Statistical analysis and modeling of the geometry and topology of plant roots". In: *Journal of Theoretical Biology* 486 (2020), p. 110108. ISSN: 0022-5193. URL: https://www.sciencedirect.com/science/article/pii/S0022519319304771.

105 Ziqi Wang, Peng Song, and Mark Pauly. "DESIA: A General Framework for Designing Interlocking Assemblies". In: *ACM Transactions on Graphics* 37.6 (Dec. 2018). ISSN: 0730-0301.

CHAPTER 5

3-D ASSEMBLIES

Let us now focus our attention on the creation of 3D assemblies. This chapter is organised as follows: SECTION 5.1 introduces the mathematical object of dual quaternions. We will see indeed that any rigid body motion in 3D space can be unequivocally represented by a unit dual quaternion. Dual quaternions, therefore, encode rigid body motions in space, much like a 2D vector $\boldsymbol{x}_t \in \mathcal{S}^1$ or a rotation centre $\boldsymbol{x}_r \in \mathbb{R}^2$ respectively encode a translation and a rotation in the plane. SECTION 5.1 is divided in two parts: the reader is kindly invited to read the first subsections, up to SECTION 5.1.3 included, but may gloss over the remaining subsections as they go more in depth in the technicalities behind unit dual quaternions. The focus of SECTION 5.2 is to define the cone of freedom of motion in 3D, in a similar fashion as what we did in translation and rotation in 2D in CHAPTER 3. SECTION 5.4 describes how the guided projection algorithm is used to optimise a separating surface so that it obeys a given unit dual quaternion. SECTION 5.5 explains how to create the successive parts of a N+1-parts assembly, in the same spirit as what was done in 2D in the previous chapter. Finally SECTION 5.6 deals with the relationships between the cone of freedom and the snap face-vertex pairs (defined later); it also shows how to create 3D assemblies robust to imperfections.

5.1 DUAL QUATERNIONS

5.1.1 GOAL

In the context of 3D assemblies, the user may have a prior idea of the approximate shape of a part and the motion to disassemble it, but have no formal way to mathematically express this motion. In this case, knowing the assembled and disassembled poses of a part, we want to find a rotation and a translation mapping one to the other. It turns out that unit dual quaternions are extremely convenient at formally representing such a motion. To illustrate this, on FIGURE 5.1, the design domain is the cube and the user partitions it to get approximate shapes P_0 in blue and P_1 in red in the assembled position. We call these approximate parts **pseudo-parts**. Pseudo-part P_1 is then placed onto the desired disassembled pose consisting of a different location and orientation in space, in faint red. Our goal is to express mathematically the rigid motion mapping pseudo-P_1 in the assembled state (deep red) to pseudo-P_1 in the disassembled state (faint red). Here, this mapping is figured by the helical trajectory in red around the screw axis in black. We will understand in this section that such a screw motion is encoded by a unit dual quaternion. Put simply, this section aims at explaining how to find the rigid motion mapping two similar bodies with different locations and orientations in space using unit dual quaternions.

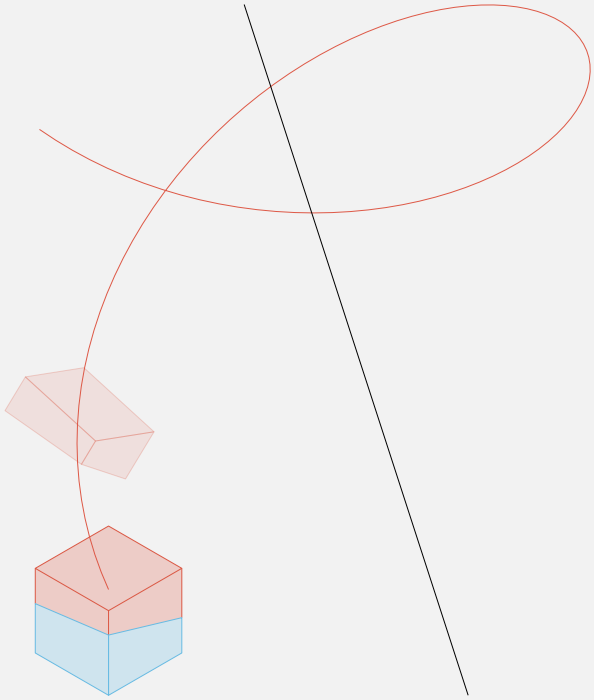


Figure 5.1| This sections describes the process to find the unit dual quaternion encoding the screw motion mapping pseudo-P_1 in its assembled pose (deep red) to pseudo-P_1 in its disassembled pose (faint red).

5.1.2 MOTIVATION

Our motivation for using dual quaternions to rigidly move 3D objects instead of the more common rotation matrices and vector algebra stems from the very reason quaternions and not matrices are used to rotate 3D objects in the computer graphics and spacecraft industries. Rotation matrices use the so-called Euler's angles to manipulate objects, typically noted ψ, β, ϕ and called the yaw, pitch and roll angles (not necessarily in that order, it seems to be author-dependent). These angles are defined in a pre-chosen arbitrary cartesian frame, and the ordering of the axes is of crucial importance as in general rotations do not commute. Thus each person using Euler's angles must unequivocally define the order in which the angles are to be composed and also the convention of the ordering of the cartesian axes. More than one student, and I can relate, had a headache figuring out which order should be used to achieve the desired result. All these conventions lead to a plethora of possibilities, making the comparison of the works of two distinct persons devilishly difficult. Even the name of the angles is a matter of convention: they can be α, β, γ. The dedicated Wikipedia page lists as many as 10 different conventions. From a code complexity point of view, the sine and cosine of each of these three angles must be calculated and composed using matrix multiplication, operations that are not cheap. Moreover, the interpolation between two orientations is convoluted due to the arbitrary nature of the rotation axes. Finally, and it is quite problematic for practical use, rotation matrices are subjected to gimbal lock when the determinant becomes zero: in robotic applications, this can lead to surprisingly fast and unpredictable motions, potentially dangerous for nearby humans.

By comparison quaternions, despite obeying seemingly more complex multiplicative rules, neatly encode a rotation: out of the four coordinates defining a quaternion, three correspond to the coordinates of the axis of rotation and the last encodes the angle of rotation. They are simple to compose and, by design, they are free from Gimbal locks. For these reasons, they are massively used in the computer graphics community. Even the popular cross-platform game engine Unity uses them, even though it is targeted to hobbyist game creators without necessarily a deep mathematical background.

Since quaternions are more convenient to use, we made the choice of using them instead of rotation ma-

trices. But as we are interested in generalised motions and not just rotations, we have to take into account translations. While vector algebra is suited for this task, it simply seems more practical to use in its place dual quaternions which encompass in a single well-defined theory both the translation and the rotation defining a 3D motion.

5.1.3 LONG STORY SHORT

We present here a succinct summary of the results developed hereunder. This subsection is intended for the hurried or not-so-mathematically-inclined reader who do not wish to dwell on the technicalities behind unit dual quaternions.

Let $\mathbb{DH}$ denotes the set of dual quaternions. Any dual quaternion $\hat{q} \in \mathbb{DH}$ can be written as:

$$\hat{q} = \left[\cos\frac{\hat{\theta}}{2}, \hat{\boldsymbol{u}}\sin\frac{\hat{\theta}}{2}\right]$$

with $\hat{\theta}$ a dual scalar and $\hat{\boldsymbol{u}}$ a dual vector:

$$\hat{\theta} = \theta_0 + \epsilon\theta_\epsilon \quad \theta_0, \theta_\epsilon \in \mathbb{R}$$
$$\hat{\boldsymbol{u}} = \boldsymbol{u}_0 + \epsilon\boldsymbol{u}_\epsilon \quad \boldsymbol{u}_0, \boldsymbol{u}_\epsilon \in \mathbb{R}^3$$

ϵ is the so-called *dual unit*. It is a number such that:

$$\epsilon \neq 0 \qquad \epsilon^2 = 0$$

We are interested in unit dual quaternions, whose set is denoted $\mathcal{U}(\mathbb{DH})$. For $\hat{q}$ to be a unit dual quaternion, it must be such that:

$$||\boldsymbol{u}_0|| = 1 \qquad \boldsymbol{u}_0 \cdot \boldsymbol{u}_\epsilon = 0$$

In such a case, $\hat{q}$ encodes a screw motion:

- The screw axis goes through the point $\boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon \in \mathbb{R}^3$.
- The screw axis is oriented by $\boldsymbol{u}_0$.
- The angle of the rotation is θ_0.
- The magnitude of the translation is θ_ϵ.

The following subsections go into greater details to define and prove the aforementioned statements and properties. Yet they are not necessary to understand the remaining of this dissertation: one can can readily deem unit dual quaternions as a black box encoding rigid body motions in $\mathbb{R}^3$. As such the reader may gloss over the following subsections and start the reading at SECTION 5.2.

5.1.4 QUATERNIONS

Before talking about dual quaternions, it may be convenient to introduce first the simpler quaternions, used in the aerospace or video games industries to rotate 3D objects. This SECTION 5.1.4 provides a comprehensive introduction to quaternions, with the hope that anybody with a light mathematical background would be able to understand it. The interested reader is referred to [88] for a more concise (and, frankly, a quite pleasant and entertaining) introduction to quaternion and spherical interpolation.

Quaternions are an absolutely fascinating and often unappreciated area of mathematics. They are four-dimensional numbers that were first described by the Irish mathematician Hamilton on October the 16th

1843[1]. A quaternion involves a scalar part (sometimes termed real part) and a 3D imaginary part (also called the vector part). They can be represented in many forms, among which but not exhaustively:

- $q = a + \boldsymbol{i}b + \boldsymbol{j}c + \boldsymbol{k}d$ with $a, b, c, d \in \mathbb{R}$
- $q = \begin{bmatrix} a, \begin{pmatrix} b \\ c \\ d \end{pmatrix} \end{bmatrix}$
- $q = [\rho \cos\theta, \rho \boldsymbol{w} \sin\theta]$ where $\rho = ||q||$ is the norm of q (defined later) and $[\cos\theta, \boldsymbol{w}\sin\theta]$ is the unit quaternion pointing towards q with $\cos\theta = \frac{a}{\sqrt{a^2+b^2+c^2+d^2}}$ and $\sin\theta = \frac{\sqrt{b^2+c^2+d^2}}{\sqrt{a^2+b^2+c^2+d^2}}$.
- $q = [s, \boldsymbol{v}]$ where $s = a$ is the scalar part and $\boldsymbol{v} = (b, c, d)^T$ is the vector part.

The set of quaternions is denoted as $\mathbb{H}$ in homage to Hamilton.

5.1.4.1 Multiplication

In the first representation numbers $1, \boldsymbol{i}, \boldsymbol{j}$ and $\boldsymbol{k}$ are the quaternions units and follow a certain number of multiplicative rules that are summed up in TABLE 5.1. Note that the multiplication between two quaternions is not commutative : $\boldsymbol{ij} \neq \boldsymbol{ji}$.

$\times$	1	$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{k}$
1	1	$\boldsymbol{i}$	$\boldsymbol{j}$	$\boldsymbol{k}$
$\boldsymbol{i}$	$\boldsymbol{i}$	-1	$\boldsymbol{k}$	$-\boldsymbol{j}$
$\boldsymbol{j}$	$\boldsymbol{j}$	$-\boldsymbol{k}$	-1	$\boldsymbol{i}$
$\boldsymbol{k}$	$\boldsymbol{k}$	$\boldsymbol{j}$	$-\boldsymbol{i}$	-1

Table 5.1| Quaternion units multiplication rules (the multiplication order is row times column)

The additive rule is straight forward

$$\begin{aligned} q_1 + q_2 &= (a_1 + \boldsymbol{i}b_1 + \boldsymbol{j}c_1 + \boldsymbol{k}d_1) + (a_2 + \boldsymbol{i}b_2 + \boldsymbol{j}c_2 + \boldsymbol{k}d_2) \\ &= a_1 + a_2 + \boldsymbol{i}(b_1 + b_2) + \boldsymbol{j}(c_1 + c_2) + \boldsymbol{k}(d_1 + d_2) \end{aligned}$$

Given the two quaternions on the previous equation, their multiplication using TABLE 5.1 results in:

$$\begin{aligned} q_1 q_2 = (a_1 a_2 - (b_1 b_2 + c_1 c_2 + d_1 d_2)) + \boldsymbol{i}(a_1 b_2 + a_2 b_1 + c_1 d_2 - d_1 c_2) + \\ \boldsymbol{j}(a_1 c_2 + a_2 c_1 + d_1 b_2 - d_2 b_1) + \boldsymbol{k}(a_1 d_2 + a_2 d_1 + b_1 c_2 - b_2 c_1) \end{aligned}$$

Computing quaternion multiplication using these rules can be quite cumbersome which might explain why quaternions are sometimes considered as a "*positive evil of no inconsiderable magnitude*" as Heaviside once put it. However the last representation $q = [s, \boldsymbol{v}]$ can neatly define multiplication using our modern vector algebra, namely the dot product $(. \cdot .)$ and cross product $(. \times .)$:

$$q_1 q_2 = [s_1, \boldsymbol{v_1}][s_2, \boldsymbol{v_2}] \tag{5.1}$$

$$= [(s_1 s_2 - \boldsymbol{v_1} \cdot \boldsymbol{v_2}), (s_1 \boldsymbol{v_2} + s_2 \boldsymbol{v_1} + \boldsymbol{v_1} \times \boldsymbol{v_2})] \tag{5.2}$$

[1] A headstone marks the bridge on which he was the moment when he got the idea.

The addition is also very simple:

$$\begin{aligned} q_1 + q_2 &= [s_1, \boldsymbol{v_1}] + [s_2, \boldsymbol{v_2}] \\ &= [s_1 + s_2, \boldsymbol{v_1} + \boldsymbol{v_2}] \end{aligned}$$

5.1.4.2 The conjugate, norm and reciprocal of a quaternion

The definition of the conjugate of a quaternion is simply extended from the definition of the conjugate of a complex number. As a reminder let $z = a + \boldsymbol{i}b \in \mathbb{C}$ be a complex number. Then the conjugate of z is $z^* = a - \boldsymbol{i}b$.
The conjugate of the different representations given earlier are:

- $q = a + \boldsymbol{i}b + \boldsymbol{j}c + \boldsymbol{k}d \implies q^* = a - \boldsymbol{i}b - \boldsymbol{j}c - \boldsymbol{k}d$
- $q = \left[a, \begin{pmatrix} b \\ c \\ d \end{pmatrix}\right] \implies q^* = \left[a, \begin{pmatrix} -b \\ -c \\ -d \end{pmatrix}\right]$
- $q = [\rho\cos\theta, \rho\boldsymbol{w}\sin\theta] \implies q^* = [\rho\cos\theta, -\boldsymbol{w}\rho\sin\theta] = [\rho\cos(-\theta), \rho\boldsymbol{w}\sin(-\theta)]$.
- $q = [s, \boldsymbol{v}] \implies q^* = [s, -\boldsymbol{v}]$.

The conjugate of the product of two quaternions is equal to the product of the conjugates computed in the reverse order: let $q_1 = [s_1, \boldsymbol{v_1}]$ and $q_2 = [s_2, \boldsymbol{v_2}]$

$$\begin{aligned} \big(q_1 q_2\big)^* &= [s_1 s_2 - \boldsymbol{v_1} \cdot \boldsymbol{v_2}, s_1\boldsymbol{v_2} + s_2\boldsymbol{v_1} + \boldsymbol{v_1} \times \boldsymbol{v_2}]^* \\ &= [s_1 s_2 - \boldsymbol{v_1} \cdot \boldsymbol{v_2}, s_1\boldsymbol{v_2} + s_2\boldsymbol{v_1} + \boldsymbol{v_1} \times \boldsymbol{v_2}] \\ &= [s_1 s_2 - (-\boldsymbol{v_1}) \cdot (-\boldsymbol{v_2}), s_1(-\boldsymbol{v_2}) + s_2(-\boldsymbol{v_1}) + (-\boldsymbol{v_2} \times (-\boldsymbol{v_1})] \\ &= [s_2, -\boldsymbol{v_2}][s_1, -\boldsymbol{v_1}] \\ &= q_2^* q_1^* \end{aligned}$$

Recall that the norm of a complex number $z = a + \boldsymbol{i}b \in \mathbb{C}$ is the euclidean distance between the point represented by z and the origin of $\mathbb{C}$: $||z|| = \sqrt{a^2 + b^2} = \sqrt{zz^*}$. The norm of a quaternion $q = a + \boldsymbol{i}b + \boldsymbol{j}c + \boldsymbol{k}d \in \mathbb{H}$ is simply

$$||q|| = \sqrt{a^2 + b^2 + c^2 + d^2} = \sqrt{s^2 + \boldsymbol{v} \cdot \boldsymbol{v}} = \sqrt{qq^*} = \sqrt{q^* q}$$

which also corresponds to the euclidean distance between $\mathbf{0}$ and the point $(a, b, c, d)^T \in \mathbb{R}^4$.
A unit quaternion q is simply such that $||q|| = 1$. Let $\mathcal{U}(\mathbb{H}) = \{q \in \mathbb{H}, ||q|| = 1\}$ denote the set of unit quaternions.

The reciprocal of a *non zero* quaternion q is by definition the quaternion q^{-1} such as $qq^{-1} = q^{-1}q = [1, \mathbf{0}]$ where $[1, \mathbf{0}]$ is the quaternion representing the real number 1. The conjugate of a complex number $z = a + \boldsymbol{i}b$ is $z^{-1} = \frac{z^*}{zz^*} = \frac{a - \boldsymbol{i}b}{||z||^2}$. This definition is extended for quaternions and:

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

Note that for a unit quaternion $q \in \mathcal{U}(\mathbb{H})$, the conjugate is the reciprocal:

$$||q|| = 1 \implies q^* = q^{-1} \tag{5.3}$$

5.1.5 QUATERNIONS AND MULTIPLICATION

Just as complex (2D) numbers neatly define rotations in the plane, involving some scaling and rotating, quaternions are quite surprisingly very pragmatic tools to describe rotations in 3D. They are more efficient and less prone to numerical errors than other methods, including Euler angles and rotation matrices, to handle rotations in 3D.

Notations: Let $\boldsymbol{p} \in \mathbb{R}^3$ be a point and $q = \left[\cos\frac{\theta}{2}, \boldsymbol{u}\sin\frac{\theta}{2}\right] \in \mathcal{U}(\mathbb{H})$ a unit quaternion. The quaternionic representation of point $\boldsymbol{p}$ is denoted as the quaternion $p = [0, \boldsymbol{p}] \in \mathbb{H}$. The result $\tilde{\boldsymbol{p}}$ of counterclockwise rotation of $\boldsymbol{p}$ around axis $\boldsymbol{u}$ by angle θ is given by:

$$\tilde{p} = [0, \tilde{\boldsymbol{p}}] = qpq^{-1}$$

To make this dissertation more digest, the explanations and details of this formula are glossed over in this chapter. That being said, the interested reader is referred to APPENDIX D.1 which goes into great detail and tries to provide intuition on the why and how of this formula. Only the most salient results are stated hereunder.

Intuition

The computation of the product qpq^{-1} and the proof (see APPENDIX D.1) that it does the rotation wanted are quite cumbersome and do not let us really understand what is going on. Let's try to understand why defining q using half the angle we want to rotate $\left(\frac{\theta}{2}\right)$ and multiplying on the right and on the left by q and $q^{-1} = q^*$ (recall that $||q|| = 1$, see EQUATION (5.3)) yields a satisfying result.

APPENDIX D.1 shows that for any point $\boldsymbol{s} \in \mathbb{R}^3$ represented by the quaternion $s = [0, \boldsymbol{s}]$, and for any unit quaternion $r = [\cos\alpha, \boldsymbol{v}\sin\alpha]$ the quaternions rs and sr form an angle α with quaternion s. Moreover this transformation is distance-preserving: $||sr|| = ||rs|| = ||s||$: it is thus a rotation. Multiplying s by r on the left means that the rotation happens in an anticlockwise manner about $\boldsymbol{v}$ to get rs and multiplying on the right means that the rotation is clockwise.
This rotation happens in $\mathbb{R}^4$ and any projection into the workspace $\mathbb{R}^3$ leads to distortions: the dot product is not preserved by this projection, hence neither are the angles nor the distances between points, which means that the transformation is not rigid and thus the product rs (or sr) cannot be considered as encoding a 3D rotation.
Let's zoom in the construction of $p_r = qpq^*$ where $q = [\cos\frac{\theta}{2}, \boldsymbol{u}\sin\frac{\theta}{2}]$ for some angle θ and unit vector $\boldsymbol{u} \in \mathbb{R}^3$. The quaternion qp is forming an angle $\frac{\theta}{2}$ with p as the rotation happened about $\boldsymbol{u}$ in an anticlockwise manner. Taking the quaternion qp and multiplying it on the right by $q^{-1} = q^*$ (EQUATION (5.3)) might at first seem odd as one is rotating in an clockwise manner through $\frac{\theta}{2}$, so it could seem that we are rotating qp back to p. Yet, a more careful analysis shows that q^* can be interpreted in two ways:

1. $q^* = [\cos\frac{\theta}{2}, -\boldsymbol{u}\sin\frac{\theta}{2}]$
2. $q^* = [\cos\frac{-\theta}{2}, \boldsymbol{u}\sin\frac{-\theta}{2}]$

Bearing in mind that qp is multiplied on the right by q^*, the first interpretation means that the rotation of $\frac{\theta}{2}$ happens in an clockwise manner (multiplication on the right) but about the negative axis of rotation $-\boldsymbol{u}$. The second interpretation suggests that the clockwise rotation is about $\boldsymbol{u}$ but through the opposite angle $\frac{-\theta}{2}$. Either way, a clockwise rotation through the positive angle about the negative axis or a clockwise rotation about the positive axis through the negative angle yields the same result: the rotation of qp to qpq^* happens in the same direction as the rotation from p to qp does, i.e. an anticlockwise rotation through the positive angle about the positive axis. All in all the operation taking p to qp is a rotation through half the angle, and

the one bringing qp to qpq^* completes this rotation through the other half angle.
As shown by EQUATIONS (D.2) and (D.3), not only does the multiplication on the right by q^* and on the left by q sums up to one rotation through the full angle θ, it keeps the scalar part of the resulting quaternion to zero: it defines an isomorphism on the set of quaternion with a zero scalar-part, precisely the set that can be interpreted as 3D points.

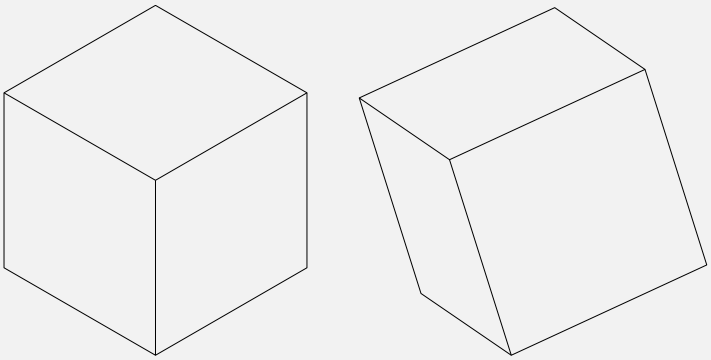


Figure 5.2| Left: a cube. Right: the image of this cube through a rotation qpq^*.

The interested reader is strongly encouraged to watch the outstanding playlist on quaternions by the youtuber 3Blue1Brown, [2] to learn more at an intuitive level about this fascinating area of mathematics. Most of my understanding comes from watching his videos.

5.1.6 DUAL NUMBERS

Dual numbers were developed by Clifford in 1873 [21]. A dual real number, which will be denoted using a hat $\hat{}$ is defined to be

$$\hat{x} = x_0 + \epsilon x_\epsilon$$

Where x_0 and x_ϵ are real numbers respectively referred to as the **real** and **dual** parts of $\hat{x}$ and ϵ is the **dual unit** such that $\epsilon^2 = 0$ (and more generally any power of ϵ greater than 2 is equal to 0). The addition and multiplication of two dual numbers $\hat{x}$ and $\hat{y}$ are defined as follows:

$$\hat{x} + \hat{y} = (x_0 + y_0) + \epsilon(x_\epsilon + y_\epsilon) \tag{5.4}$$

$$\hat{x}\hat{y} = x_0 y_0 + \epsilon(x_0 y_\epsilon + y_0 x_\epsilon) \tag{5.5}$$

A **pure dual number** is defined as a dual number having a zero real part $\hat{x} = \epsilon x_\epsilon$. The inverse of a dual number is defined for a non-pure dual number as:

$$\hat{x}^{-1} = \left(\frac{1}{x_0} + \epsilon\frac{x_\epsilon}{x_0^2}\right)$$

And it can easily be verified that $\hat{x}^{-1}\hat{x} = 1$.

5.1.6.1 Dual vectors

The following definitions could be generalized to any dimension, but as we are working in $\mathbb{R}^3$ we are only going to focus on vectors and dual vectors in three dimensions.

A dual vector $\hat{\boldsymbol{x}}$ is a vector whose entries are all dual numbers. Superscript $.^i$ denotes the i^{th} component.

$$\hat{\boldsymbol{x}} = \begin{pmatrix} \hat{x}^1 \\ \hat{x}^2 \\ \hat{x}^3 \end{pmatrix} = \begin{pmatrix} x_0^1 + \epsilon x_\epsilon^1 \\ x_0^2 + \epsilon x_\epsilon^2 \\ x_0^3 + \epsilon x_\epsilon^3 \end{pmatrix}$$

The product of a dual number $\hat{x}$ with a dual vector $\hat{\boldsymbol{x}}$ is obtained under the standard multiplicative rule:

$$\hat{x}\hat{\boldsymbol{x}} = \begin{pmatrix} \hat{x}\hat{x}^0 \\ \hat{x}\hat{x}^1 \\ \hat{x}\hat{x}^2 \end{pmatrix}$$

The dot product and cross product between dual vectors $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ are respectively defined to be:

$$\hat{\boldsymbol{x}} \cdot \hat{\boldsymbol{y}} = \hat{x}^1\hat{y}^1 + \hat{x}^2\hat{y}^2 + \hat{x}^3\hat{y}^3 \tag{5.6}$$

$$\hat{\boldsymbol{x}} \times \hat{\boldsymbol{y}} = \begin{pmatrix} \hat{x}^2\hat{y}^3 - \hat{x}^3\hat{y}^2 \\ \hat{x}^3\hat{y}^1 - \hat{x}^1\hat{y}^3 \\ \hat{x}^1\hat{y}^2 - \hat{x}^2\hat{y}^1 \end{pmatrix} \tag{5.7}$$

5.1.7 DUAL QUATERNIONS

A dual quaternion $\hat{q}$ is defined by extending the definition of a dual number:

$$\hat{q} = q_0 + \epsilon q_\epsilon$$

where q_0 and q_ϵ are the quaternions:

$$q_0 = [s_{q_0}, \boldsymbol{v_{q_0}}]$$
$$q_\epsilon = [s_{q_\epsilon}, \boldsymbol{v_{q_\epsilon}}]$$

By additive and multiplicative rules of quaternion, a dual quaternion can also be viewed as built of a dual scalar part and dual vector part:

$$\begin{aligned} \hat{q} &= q_0 + \epsilon q_\epsilon \\ &= [s_{q_0}, \boldsymbol{v_{q_0}}] + \epsilon[s_{q_\epsilon}, \boldsymbol{v_{q_\epsilon}}] \\ &= [s_{q_0} + \epsilon s_{q_\epsilon}, \boldsymbol{v_{q_0}} + \epsilon \boldsymbol{v_{q_\epsilon}}] \\ &= [\hat{s}_q, \hat{\boldsymbol{v}}_{\boldsymbol{q}}] \end{aligned}$$

The sum and product of two dual quaternion $\hat{q}$ and $\hat{r}$ are:

$$\begin{aligned} \hat{q} + \hat{r} &= \left([s_{q_0}, \boldsymbol{v_{q_0}}] + \epsilon[s_{q_\epsilon}, \boldsymbol{v_{q_\epsilon}}]\right) + \left([s_{r_0}, \boldsymbol{v_{r_0}}] + \epsilon[s_{r_\epsilon}, \boldsymbol{v_{r_\epsilon}}]\right) \\ &= [s_{q_0} + s_{r_0}, \boldsymbol{v_{q_0}} + \boldsymbol{v_{r_0}}] + \epsilon[s_{q_\epsilon} + s_{r_\epsilon}, \boldsymbol{v_{q_\epsilon}} + \boldsymbol{v_{r_\epsilon}}] \\ &= [(s_{q_0} + s_{r_0}) + \epsilon(s_{q_\epsilon} + s_{r_\epsilon}), (\boldsymbol{v_{q_0}} + \boldsymbol{v_{r_0}}) + \epsilon(\boldsymbol{v_{q_\epsilon}} + \boldsymbol{v_{r_\epsilon}})] \\ &= [(s_{q_0} + \epsilon s_{q_\epsilon}) + (s_{r_0} + \epsilon s_{r_\epsilon}), (\boldsymbol{v_{q_0}} + \epsilon \boldsymbol{v_{q_\epsilon}}) + (\boldsymbol{v_{r_0}} + \epsilon \boldsymbol{v_{r_\epsilon}})] \\ &= [\hat{s}_q + \hat{r}_q, \hat{\boldsymbol{v}}_{\boldsymbol{q}} + \hat{\boldsymbol{v}}_{\boldsymbol{r}}] \end{aligned}$$

$$
\begin{aligned}
\hat{q}\hat{r} &= (q_0 + \epsilon q_\epsilon)(r_0 + \epsilon r_\epsilon) \\
&= q_0 r_0 + \epsilon(q_0 r_\epsilon + q_\epsilon r_0) \\
&= [s_{q_0} s_{r_0} - \boldsymbol{v_{q_0}} \cdot \boldsymbol{v_{r_0}}, s_{q_0}\boldsymbol{v_{r_0}} + s_{r_0}\boldsymbol{v_{q_0}} + \boldsymbol{v_{q_0}} \times \boldsymbol{v_{r_0}}] + \\
&\quad \epsilon[s_{q_0} s_{r_\epsilon} + s_{q_\epsilon} s_{r_0} - \boldsymbol{v_{q_0}} \cdot \boldsymbol{v_{r_\epsilon}} - \boldsymbol{v_{q_\epsilon}} \cdot \boldsymbol{v_{r_0}}, s_{q_0}\boldsymbol{v_{r_\epsilon}} + s_{r_\epsilon}\boldsymbol{v_{q_0}} + s_{q_\epsilon}\boldsymbol{v_{r_0}} + s_{r_0}\boldsymbol{v_{q_\epsilon}} + \boldsymbol{v_{q_0}} \times \boldsymbol{v_{r_\epsilon}} + \boldsymbol{v_{q_\epsilon}} \times \boldsymbol{v_{r_0}}] \\
&= [(s_{q_0} s_{r_0} - \boldsymbol{v_{q_0}} \cdot \boldsymbol{v_{r_0}}) + \epsilon(s_{q_0} s_{r_\epsilon} + s_{q_\epsilon} s_{r_0} - \boldsymbol{v_{q_0}} \cdot \boldsymbol{v_{r_\epsilon}} - \boldsymbol{v_{q_\epsilon}} \cdot \boldsymbol{v_{r_0}}), \\
&\quad (s_{q_0}\boldsymbol{v_{r_0}} + s_{r_0}\boldsymbol{v_{q_0}} + \boldsymbol{v_{q_0}} \times \boldsymbol{v_{r_0}}) + \epsilon(s_{q_0}\boldsymbol{v_{r_\epsilon}} + s_{r_\epsilon}\boldsymbol{v_{q_0}} + s_{q_\epsilon}\boldsymbol{v_{r_0}} + s_{r_0}\boldsymbol{v_{q_\epsilon}} + \boldsymbol{v_{q_0}} \times \boldsymbol{v_{r_\epsilon}} + \boldsymbol{v_{q_\epsilon}} \times \boldsymbol{v_{r_0}})] \\
&= [\big(s_{q_0} s_{r_0} + \epsilon(s_{q_0} s_{r_\epsilon} + s_{q_\epsilon} s_{r_0})\big) - \big(\boldsymbol{v_{q_0}} \cdot \boldsymbol{v_{r_0}} + \epsilon(\boldsymbol{v_{q_0}} \cdot \boldsymbol{v_{r_\epsilon}} + \boldsymbol{v_{q_\epsilon}} \cdot \boldsymbol{v_{r_0}})\big), \\
&\quad \big((s_{q_0}\boldsymbol{v_{r_0}} + \epsilon(s_{q_0}\boldsymbol{v_{r_\epsilon}} + s_{r_\epsilon}\boldsymbol{v_{q_0}})\big) + \big(s_{r_0}\boldsymbol{v_{q_0}} + \epsilon(s_{q_\epsilon}\boldsymbol{v_{r_0}} + s_{r_0}\boldsymbol{v_{q_\epsilon}})\big) + \big(\boldsymbol{v_{q_0}} \times \boldsymbol{v_{r_0}} + \epsilon(\boldsymbol{v_{q_0}} \times \boldsymbol{v_{r_\epsilon}} + \boldsymbol{v_{q_\epsilon}} \times \boldsymbol{v_{r_0}})\big)] \\
&= [\hat{s}_q \hat{s}_r - \hat{\boldsymbol{v}}_{\boldsymbol{q}} \cdot \hat{\boldsymbol{v}}_{\boldsymbol{r}}, \hat{s}_q \hat{\boldsymbol{v}}_{\boldsymbol{r}} + \hat{s}_r \hat{\boldsymbol{v}}_{\boldsymbol{q}} + \hat{\boldsymbol{v}}_{\boldsymbol{q}} \times \hat{\boldsymbol{v}}_{\boldsymbol{r}}]
\end{aligned}
$$

The results bear close resemblance to the sum and products of standard quaternions. A **pure dual quaternion** is a dual quaternion $\hat{q}$ whose real quaternion $q_0 = 0$. If $\hat{q}$ is not a pure dual quaternion, then the inverse of $\hat{q}$ is

$$\hat{q}^{-1} = q_0^{-1} - \epsilon q_\epsilon q_0^{-2}$$

Two **conjugates** of dual quaternion $\hat{q}$ are defined:

$$
\begin{aligned}
\hat{q}^* &= q_0^* + \epsilon q_\epsilon^* \\
\hat{q}^\circ &= q_0^* - \epsilon q_\epsilon^*
\end{aligned}
\tag{5.8}
$$

The product $\hat{q}\hat{q}^*$ is a dual scalar as proved in the following equations:

$$
\begin{aligned}
\hat{q}\hat{q}^* &= (q_0 + \epsilon q_\epsilon)(q_0^* + \epsilon q_\epsilon^*) \\
&= q_0 q_0^* + \epsilon(q_0 q_\epsilon^* + q_\epsilon q_0^*) \\
&= ||q_0||^2 + \epsilon\big([s_{q_0}, \boldsymbol{v}_{q_0}][s_{q_\epsilon}, -\boldsymbol{v}_{q_\epsilon}] + [s_{q_\epsilon}, \boldsymbol{v}_{q_\epsilon}][s_{q_0}, -\boldsymbol{v}_{q_0}]\big) \\
&= ||q_0||^2 + \epsilon\big([s_{q_0} s_{q_\epsilon} + \boldsymbol{v_{q_\epsilon}} \cdot \boldsymbol{v_{q_0}}, s_{q_\epsilon}\boldsymbol{v_{q_0}} - s_{q_0}\boldsymbol{v_{q_\epsilon}} - \boldsymbol{v}_{q_0} \times \boldsymbol{v_{q_\epsilon}}] + [s_{q_\epsilon} s_{q_0} + \boldsymbol{v_{q_0}} \cdot \boldsymbol{v_{q_\epsilon}}, s_{q_0}\boldsymbol{v_{q_\epsilon}} - s_{q_\epsilon}\boldsymbol{v_{q_0}} - \boldsymbol{v_{q_\epsilon}} \times \boldsymbol{v}_{q_0}]\big) \\
&= ||q_0||^2 + \epsilon[2 s_{q_0} s_{q_\epsilon} + 2\boldsymbol{v_{q_\epsilon}} \cdot \boldsymbol{v_{q_0}}, \mathbf{0}] \\
&= ||q_0||^2 + 2\epsilon(s_{q_0} s_{q_\epsilon} + \boldsymbol{v_{q_\epsilon}} \cdot \boldsymbol{v_{q_0}})
\end{aligned}
$$

By definition, a **unit dual quaternion** is a dual quaternion $\hat{q}$ such that $\hat{q}\hat{q}^* = 1$. Hence, by identifying the real and dual part of $\hat{q}\hat{q}^*$, a unit dual quaternion $\hat{q}$ must satisfy:

$$||q_0|| = 1 \tag{5.9}$$

$$s_{q_0} s_{q_\epsilon} + \boldsymbol{v_{q_\epsilon}} \cdot \boldsymbol{v_{q_0}} = 0 \tag{5.10}$$

When one interprets quaternions q_0 and q_ϵ as vectors in four dimensions : $q_0 = (s_{q_0}, \boldsymbol{v}_{\boldsymbol{q_0}}^T)^T$ and $q_\epsilon = (s_{q_\epsilon}, \boldsymbol{v}_{\boldsymbol{q_\epsilon}}^T)^T$, then one finds that EQUATION (5.10) reads that q_ϵ and q_0 are orthogonal in $\mathbb{R}^4$. The fact that q_ϵ and q_0 are orthogonal and that q_0 is a unit quaternion as stated by EQUATION (5.9) implies that the set of unit dual quaternions is the union of the unit hypersphere in $\mathbb{R}^4$ and the set of hyperplanes tangent to this hypersphere (translated to the origin of $\mathbb{R}^4$ to contain the zero quaternion).
One can also verify that EQUATIONS (5.9) and (5.10) reduce the degrees of freedom when choosing the components of $\hat{q}$ from 8 to 6, exactly the number of degrees of freedom of a particle in 3D undergoing a rigid body motion. As stated by McCarthy in [59], the set of unit dual quaternions "*is a six-dimensional algebraic submanifold of* $\mathbb{R}^8$*, termed the* ***image space of spatial displacements***".

In this document, we will refer to the set of dual quaternions as $\mathbb{DH}$ and the set of unit dual quaternions as $\mathcal{U}(\mathbb{DH})$.
In what follows we develop the most salient results concerning the relationships between a rigid body motion and a unit dual quaternion.

5.1.8 SCREW AXIS AND RIGID BODY MOTION

5.1.8.1 Unit dual quaternion and generalised 3D motion

Let $\hat{q} = q_0 + \epsilon q_\epsilon$ be a unit dual quaternion. Let $\theta \in [0, \pi]$ and $\boldsymbol{u} \in \mathbb{R}^3$ be an angle and a unit vector such that

$$q_0 = [s_{q_0}, \boldsymbol{v_{q_0}}] = [\cos\frac{\theta}{2}, \boldsymbol{u}\sin\frac{\theta}{2}]$$

Meaning that q_0 is a quaternion encoding a rotation about axis $\boldsymbol{u}$ by angle θ. Let also t be a quaternion such that $t = 2q_\epsilon q_0^*$. Expanding t yields:

$$\begin{aligned} t &= 2q_\epsilon q_0^* \\ &= 2[s_{q_\epsilon}, \boldsymbol{v_{q_\epsilon}}][s_{q_0}, -\boldsymbol{v_{q_0}}] \\ &= [s_{q_\epsilon}s_{q_0} + \boldsymbol{v_{q_\epsilon}}\cdot\boldsymbol{v_{q_0}}, s_{q_0}\boldsymbol{v_{q_\epsilon}} - s_{q_\epsilon}\boldsymbol{v_{q_0}} + \boldsymbol{v_{q_0}}\times\boldsymbol{v_{q_\epsilon}}] \\ &= [0, 2\big(s_{q_0}\boldsymbol{v_{q_\epsilon}} - s_{q_\epsilon}\boldsymbol{v_{q_0}} + \boldsymbol{v_{q_0}}\times\boldsymbol{v_{q_\epsilon}}\big)] \end{aligned}$$

Where the last step made use of the fact that since $||\hat{q}|| = 1$, q_0 and q_ϵ are orthogonal. Because t has a zero scalar part, its imaginary part $\boldsymbol{t} = 2\big(s_{q_0}\boldsymbol{v_{q_\epsilon}} - s_{q_\epsilon}\boldsymbol{v_{q_0}} + \boldsymbol{v_{q_0}}\times\boldsymbol{v_{q_\epsilon}}\big)$ can be interpreted as a vector in $\mathbb{R}^3$. Since $t = 2q_\epsilon q_0^* \implies q_\epsilon = \frac{1}{2}tq_0$, $\hat{q}$ can be rewritten to be:

$$\hat{q} = q_0 + \frac{1}{2}\epsilon t q_0$$

Whereas the quaternion representation of a point $\boldsymbol{p} \in \mathbb{R}^3$ is $p = [0, \boldsymbol{p}]$, the conversion to a dual quaternion is defined to be:

$$\hat{p} = [1, \boldsymbol{0}] + \epsilon[0, \boldsymbol{p}] = 1 + \epsilon p$$

Meaning that a dual quaternion $\hat{r} = r_0 + \epsilon r_\epsilon$ can be interpreted as a point in $\mathbb{R}^3$ if and only if the non dual part r_0 is equal to 1 (as a real number) and the dual part r_ϵ is a quaternion with a zero scalar part. In that case, the vector part of the dual part can be seen as a vector in $\mathbb{R}^3$.

Let us compute the product of the dual representation of a point $\boldsymbol{p} \in \mathbb{R}^3$ with the unit dual quaternion $\hat{q}$ defined above using the conjugation given EQUATION (5.8):

$$\hat{q}\hat{p}\hat{q}^\circ = \left(q_0 + \frac{1}{2}\epsilon t q_0\right)\left([1, \boldsymbol{0}] + \epsilon[0, \boldsymbol{p}]\right)\left(q_0^* - \frac{1}{2}\epsilon(tq_0)^*\right) \tag{5.11a}$$

$$= \left(q_0 + \epsilon\left(q_0 p + \frac{1}{2}tq_0\right)\right)\left(q_0^* - \frac{1}{2}\epsilon q_0^* t^*\right) \tag{5.11b}$$

$$= q_0 q_0^* + \epsilon\left(-\frac{1}{2}q_0 q_0^* t^* + q_0 p q_0^* + \frac{1}{2}tq_0 q_0^*\right) \tag{5.11c}$$

$$= 1 + \epsilon\left(-\frac{1}{2}[0, -\boldsymbol{t}] + q_0 p q_0^* + \frac{1}{2}[0, \boldsymbol{t}]\right) \tag{5.11d}$$

$$= 1 + \epsilon(q_0 p q_0^* + t) \tag{5.11e}$$

$q_0 p q_0^*$ is exactly the result of the rotation of $\boldsymbol{p}$ through quaternion q_0 as we saw SECTION 5.1.5. Therefore the scalar part of $q_0 p q_0^*$ is 0 and its imaginary part can be interpreted as a rotated point. Quaternion t also

has a zero scalar part and can be interpreted as a 3D vector. Thus, the dual quaternion $\hat{q}\hat{p}\hat{q}^\circ$ meets all conditions to be viewed as a point in $\mathbb{R}^3$ and one can interpret the operation $\hat{q}\hat{p}\hat{q}^\circ$ as rotating $\boldsymbol{p}$ by θ around $\boldsymbol{u}$ followed by a translation of magnitude $||\boldsymbol{t}||$ along direction $\frac{\boldsymbol{t}}{||\boldsymbol{t}||}$. In other words, $\hat{q}$ enables $\boldsymbol{p}$ to be subjected to a rigid body motion !

Had we wanted to translate first $\boldsymbol{p}$ and then rotate it about $\boldsymbol{u}$, we would have used the quaternion $\hat{r} = r_0 + \frac{1}{2}\epsilon r_0 t$ with $r_0 = q_0 = [\cos\frac{\theta}{2}, \boldsymbol{u}\sin\frac{\theta}{2}]$. Indeed, expanding $\hat{r}\hat{p}\hat{r}^\circ$ yields:

$$\begin{aligned}
\hat{r}\hat{p}\hat{r}^\circ &= \left(r_0 + \frac{1}{2}\epsilon r_0 t\right)(1+\epsilon p)\left(r_0^* - \frac{1}{2}\epsilon(r_0 t)^*\right) \\
&= \left(r_0 + \epsilon\left(r_0 p + \frac{1}{2}r_0 t\right)\right)\left(r_0^* + \epsilon\left(-\frac{1}{2}t^* r_0^*\right)\right) \\
&= r_0 r_0^* + \epsilon\left(-\frac{1}{2}r_0 t r_0^* + r_0 p r_0^* + \frac{1}{2}r_0 t r_0^*\right) \\
&= 1 + \epsilon r_0\left(-\frac{1}{2}[0,-\boldsymbol{t}] + p + \frac{1}{2}[0,\boldsymbol{t}]\right) r_0^* \\
&= 1 + \epsilon r_0 (p+t) r_0^*
\end{aligned} \tag{5.12}$$

For the same reason as above, $\hat{r}\hat{p}\hat{r}^\circ$ can be interpreted as a point $\boldsymbol{p}$ that is first translated along $\boldsymbol{t}$ and then rotated through quaternion r_0.

The reader's attention is drawn to the fact that a rotation without translation can be computed with a unit dual quaternion $\hat{q}$ having $q_\epsilon = 0$ and a translation without rotation would be given by $\hat{q}$ with $q_0 = 1$. Note that in that case $\hat{q} = 1 + \frac{1}{2}\epsilon t$ works with *half* the amount of translation, quite similarly to a pure rotation working with half the angle $\frac{\theta}{2}$.

5.1.8.2 Mozzi–Chasles' theorem

The Mozzi–Chasles' theorem (discovered by the Italian G. Mozzi in 1763 but attributed to M. Chasles in 1830) states that ([45] page 666)

> **Mozzi–Chasles' theorem**
> *The most general rigid motion is resultant of a rotation and a translation parallel to the axis of the rotation*

A neat and rigorous demonstration of this theorem is provided in [45].

Following [50] and [22] we prove in APPENDIX D.2 that any unit dual quaternion can be written down as:

$$\hat{q} = \left[\cos\frac{\hat{\theta}}{2}, \hat{\boldsymbol{u}}\sin\frac{\hat{\theta}}{2}\right]$$

where $\hat{\theta}$ is a dual angle and $\hat{\boldsymbol{u}}$ is a dual vector in $\mathbb{R}^3$. Such concise expression will turn out to be particularly convenient to encode a rigid body motion: we will understand hereunder that the dual angle $\hat{\theta} = \theta_0 + \epsilon\theta_\epsilon$ contains the angle of rotation (θ_0) and the magnitude of the translation (θ_ϵ), while the dual vector $\hat{\boldsymbol{u}} = \boldsymbol{u}_0 + \epsilon\boldsymbol{u}_\epsilon$ is constituted of the so-called **Plücker coordinates** of the screw axis.

5.1.8.3 Plücker coordinates

As stated in [22], Plücker coordinates [66] are a convenient way to store a line in $\mathbb{R}^3$ as a sextuplet $[\boldsymbol{l}, \boldsymbol{m}]$ with $\boldsymbol{l}, \boldsymbol{m} \in \mathbb{R}^3$. Let us consider a line going through two distinct points $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$. A direction of the line is the obtained as the vector: $\boldsymbol{l} = \boldsymbol{p}_B - \boldsymbol{p}_A$. The moment of the line is defined as $\boldsymbol{m} = \boldsymbol{p}_A \times \boldsymbol{p}_B$. Moment $\boldsymbol{m}$ is a vector normal to the plane defined by $\boldsymbol{p}_A, \boldsymbol{p}_B$ and the origin $\boldsymbol{o}$ of $\mathbb{R}^3$ and whose length is equal to twice the area of triangle $\boldsymbol{o}\boldsymbol{p}_A\boldsymbol{p}_B$. Note that $\boldsymbol{m} = \boldsymbol{p} \times \boldsymbol{l}$ for any point $\boldsymbol{p}$ on the line. The tuple $[\boldsymbol{l}, \boldsymbol{m}]$ defines a line

up to a scalar multiplication $[\lambda \boldsymbol{l}, \lambda \boldsymbol{m}]$ with $\lambda \neq 0$. Ensuring $||\boldsymbol{l}|| = 1$ and $\boldsymbol{l} \cdot \boldsymbol{m} = 0$ reduces the number of degree of freedom from 6 to 4, i.e. exactly the number of degree of freedom of a 3D line, which shows that the Plücker coordinates $[\boldsymbol{l}, \boldsymbol{m}]$ uniquely defines a line.

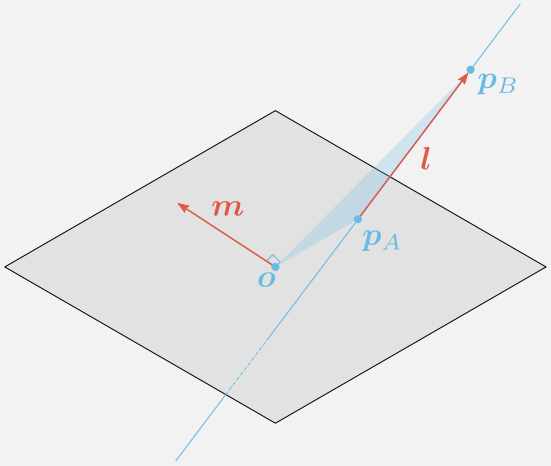


Figure 5.3| A line and its Plücker coordinates.

We solve the problem of finding the line associated with given Plücker coordinates. Let $[\boldsymbol{l}, \boldsymbol{m}]$ be the Plücker coordinates of a line with $||\boldsymbol{l}|| = 1$ and $\boldsymbol{l} \cdot \boldsymbol{m} = 0$. Finding the line associated with the tuple $[\boldsymbol{l}, \boldsymbol{m}]$ means finding a point on that line since we already know the line orientation $\boldsymbol{l}$. Let $\boldsymbol{p}$ be such a point. $\boldsymbol{p}$ can be decomposed into two parts: $\boldsymbol{p}_{\parallel}$ and $\boldsymbol{p}_{\perp}$ respectively the collinear and orthogonal components of $\boldsymbol{p}$ with respect to $\boldsymbol{l}$. By definition $\boldsymbol{m} = \boldsymbol{p} \times \boldsymbol{l}$ and so $\boldsymbol{m} = \boldsymbol{p}_{\perp} \times \boldsymbol{l}$. Note that $\boldsymbol{p}_{\perp}$ is the projection of the origin O onto the line $[\boldsymbol{l}, \boldsymbol{m}]$, i.e. $\boldsymbol{p}_{\perp}$ is the point on the line the closest to O. Expanding the cross product $\boldsymbol{l} \times \boldsymbol{m}$ yields:

$$\begin{aligned} \boldsymbol{l} \times \boldsymbol{m} &= \boldsymbol{l} \times (\boldsymbol{p}_{\perp} \times \boldsymbol{l}) \\ &= (\boldsymbol{l} \cdot \boldsymbol{l})\boldsymbol{p}_{\perp} - (\boldsymbol{l} \cdot \boldsymbol{p}_{\perp})\boldsymbol{l} \\ &= \boldsymbol{p}_{\perp} \end{aligned}$$

We have thus shown that the line given by the Plücker coordinates $[\boldsymbol{l}, \boldsymbol{m}]$ such that $||\boldsymbol{l}|| = 1$ and $\boldsymbol{l} \cdot \boldsymbol{m} = 0$ goes through point $\boldsymbol{l} \times \boldsymbol{m}$ with orientation $\boldsymbol{l}$. Note also that when $||\boldsymbol{l}|| = 1$ then $||\boldsymbol{p}_{\perp}|| = ||\boldsymbol{m}||$.

5.1.8.4 Unit dual quaternions and Plücker coordinates

APPENDIX D.2 proves that for a unit dual quaternion $\hat{q}$ written in the form $\hat{q} = [\cos \frac{\hat{\theta}}{2}, \hat{\boldsymbol{u}} \sin \frac{\hat{\theta}}{2}]$ with $\hat{\theta} = \theta_0 + \epsilon\theta_\epsilon$ and $\hat{\boldsymbol{u}} = \boldsymbol{u}_0 + \epsilon \boldsymbol{u}_\epsilon$, then the tuple $[\boldsymbol{u}_0, \boldsymbol{u}_\epsilon]$ defines the Plücker coordinates of a line:

$$\boxed{||\hat{q}|| = 1 \Leftrightarrow [\boldsymbol{u}_0, \boldsymbol{u}_\epsilon] \text{ are the Plücker coordinates of a line}}$$

5.1.8.5 But where is the screw motion ?

Remember that for a point $\boldsymbol{p} \in \mathbb{R}^3$ represented by the dual quaternion $\hat{p} = 1 + \epsilon p$ where p is the quaternion $[0, \boldsymbol{p}]$ and a unit dual quaternion $\hat{q} = q_0 + \frac{\epsilon}{2} t q_0$ (with t and q_0 quaternions) we saw on EQUATION (5.11e) that:

$$\hat{\tilde{p}} \equiv \hat{q}\hat{p}\hat{q}^{\circ} = 1 + \epsilon(q_0 p q_0^* + t)$$

The transformed dual quaternion $\hat{\tilde{p}}$ can be interpreted as a point in $\mathbb{R}^3$ (it is in the form $1 + \epsilon\tilde{p}$ where $\tilde{p}$ is a quaternion with a 0 scalar part: $\tilde{p} = [0, \tilde{\boldsymbol{p}}]$, $\tilde{\boldsymbol{p}} \in \mathbb{R}^3$. $\tilde{\boldsymbol{p}}$ can be decomposed in to parts: $q_0 p q_0^*$ on the one hand and t and the other. Bearing in mind SECTION 5.1.4, the first part is interpreted as the quaternion $p = [0, \boldsymbol{p}]$ rotated by the quaternion q_0 around an axis going through the origin of $\mathbb{R}^3$; the second part t is merely a

translation of this rotated point along an axis that may or may not have anything to do with the rotation axis hidden in q_0. The question is: where is the screw motion, i.e. where is the motion of rotation and translation along the same axis that is not necessarily going through the origin? To answer that we will need Plücker and the fact that any unit dual quaternion $\hat{q}$ can be written down as $\hat{q} = \left[\cos\frac{\hat{\theta}}{2}, \hat{\boldsymbol{u}}\sin\frac{\hat{\theta}}{2}\right]$.

Let θ_0 and $\theta_\epsilon \in \mathbb{R}$ be such as $\hat{\theta} = \theta_0 + \epsilon\theta_\epsilon$. Let also $\boldsymbol{u}_0, \boldsymbol{u}_\epsilon \in \mathbb{R}^3$ such that $\hat{\boldsymbol{u}} = \boldsymbol{u}_0 + \epsilon\boldsymbol{u}_\epsilon$ and $||\boldsymbol{u}_0|| = 1$ and $\boldsymbol{u}_0 \cdot \boldsymbol{u}_\epsilon = 0$ so that the tuple $[\boldsymbol{u}_0, \boldsymbol{u}_\epsilon]$ can be safely interpreted as a line in $\mathbb{R}^3$. We are going to show that the unit dual quaternion $\hat{q} = \left[\cos\frac{\hat{\theta}}{2}, \hat{\boldsymbol{u}}\sin\frac{\hat{\theta}}{2}\right]$ encodes a rotation about θ_0 and a translation of θ_ϵ along the line $[\boldsymbol{u}_0, \boldsymbol{u}_\epsilon]$, i.e precisely a screw motion of axis $[\boldsymbol{u}_0, \boldsymbol{u}_\epsilon]$.

As seen above, with $c = \cos\frac{\theta_0}{2}$, $s = \sin\frac{\theta_0}{2}$ one can develop $\hat{q}$ as

$$\begin{aligned}\hat{q} &= q_0 + \epsilon q_\epsilon \\ q_0 &= [c, \boldsymbol{u}_0 s] \\ q_\epsilon &= [-\frac{\theta_\epsilon}{2}s, \boldsymbol{u}_0\frac{\theta_\epsilon}{2}c + \boldsymbol{u}_\epsilon s]\end{aligned}$$

Developing the product $\hat{q}\hat{p}\hat{q}^\circ$ for some dual quaternion $\hat{p}$ representing a point in $\mathbb{R}^3$ yields:

$$\hat{q}\hat{p}\hat{q}^\circ = 1 + \epsilon(q_0 p q_0^* + q_\epsilon q_0^* - q_0 q_\epsilon^*)$$

The rotation term $q_0 p q_0^*$ encodes a rotation of angle θ along an axis oriented by $\boldsymbol{u}_0$ going through 0. We have the correct angle but not yet the correct axis of rotation: as far as this term is concerned the axis of rotation is $[\boldsymbol{u}_0, \boldsymbol{0}]$ and not $[\boldsymbol{u}_0, \boldsymbol{u}_\epsilon]$ as we would like. Expanding the translation term $q_\epsilon q_0^* - q_0 q_\epsilon^*$ yields:

$$q_\epsilon q_0^* - q_0 q_\epsilon^* = [0, \theta_\epsilon \boldsymbol{u}_0 + 2\boldsymbol{u}_\epsilon sc + 2\boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon s^2]$$

The good news is that the quantity $\theta_\epsilon \boldsymbol{u}_0$ is a translation of magnitude θ_ϵ along the unit vector $\boldsymbol{u}_0$: we already have one bit of what we want. But the bad news is that rotation term $q_0 p q_0^*$ is still a rotation around an axis going through 0 and the other part of the translation term $2\boldsymbol{u}_\epsilon sc + 2\boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon s^2$ does not have an easy geometrical interpretation. We are going to transform it using two identities : $\boldsymbol{u}_0 \cdot \boldsymbol{u}_0 = 1$ and $\boldsymbol{u_0} \cdot \boldsymbol{u}_\epsilon = 0$

$$2sc\boldsymbol{u}_\epsilon + 2s^2\boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon = -2sc((\boldsymbol{u_0} \cdot \boldsymbol{u}_\epsilon)\boldsymbol{u}_0 - (\boldsymbol{u}_0 \cdot \boldsymbol{u}_0)\boldsymbol{u_\epsilon}) + (s^2 + s^2 - c^2 + c^2)\boldsymbol{u}_0 \times \boldsymbol{u_\epsilon}$$

Using the triple product: $a \times (b \times c) = (a \cdot c)b - (a \cdot b)c$ we can rewrite the latter equation as:

$$2sc\boldsymbol{u}_\epsilon + 2s^2\boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon = \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon - \big((c^2 - s^2)\boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon + 2sc\boldsymbol{u}_0 \times (\boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon)\big)$$

Recall that the quantity $\boldsymbol{p}_\perp = \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon$ is the point in $\mathbb{R}^3$ that is the orthogonal projection of the origin onto the line $[\boldsymbol{u}_0, \boldsymbol{u}_\epsilon]$. Also, naturally, $\boldsymbol{u}_0 \cdot \boldsymbol{p}_\perp = 0$. EQUATION (D.3) tells us that developing the product $q_0[0, \boldsymbol{p}_\perp]q_0^*$ yields:

$$\begin{aligned}q_0[0, \boldsymbol{p}_\perp]q_0^* &= [0, 2s^2(\boldsymbol{u}_0 \cdot \boldsymbol{p}_\perp)\boldsymbol{u}_0 + (c^2 - s^2)\boldsymbol{p}_\perp + 2sc(\boldsymbol{u}_0 \times \boldsymbol{p}_\perp) \\ &= [0, (c^2 - s^2)(\boldsymbol{u}_0 \times \boldsymbol{u_\epsilon}) + 2sc(\boldsymbol{u}_0 \times (\boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon))]\end{aligned}$$

Hence:

$$\begin{aligned} q_\epsilon q_0^* - q_0 q_\epsilon^* &= [0, \theta_\epsilon \boldsymbol{u}_0 + \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon - \left((c^2 - s^2)\boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon + 2sc\boldsymbol{u}_0 \times (\boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon)\right)] \\ &= [0, \theta_\epsilon \boldsymbol{u}_0 + \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon] - q_0[0, \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon]q_0^* \end{aligned}$$

And finally:

$$\hat{q}\hat{p}\hat{q}^\circ = 1 + \epsilon \left(q_0 p q_0^* + [0, \theta_\epsilon \boldsymbol{u}_0 + \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon] - q_0[0, \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon]q_0^*\right) \tag{5.13}$$

$$= 1 + \epsilon \left(q_0 \left(p - [0, \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon]\right) q_0^* + [0, \theta_\epsilon \boldsymbol{u}_0 + \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon]\right) \tag{5.14}$$

This is it: this latest expression is precisely a screw motion around $[\boldsymbol{u}_0, \boldsymbol{u}_\epsilon]$!. To understand why, let us look at each of the terms:
As stated above $\boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon$ is the orthogonal projection of the origin onto the line $[\boldsymbol{u}_0, \boldsymbol{u}_\epsilon]$; it is the point on this line the closest to 0. Hence the quaternion $p - [0, \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon]$ is the quaternionic representation of the point $\boldsymbol{p}$ that we are interested in transforming translated by the amount $\boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon$ towards the origin. The product $q_0 \left(p - [0, \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon]\right) q_0^*$ is therefore to be understood as "*take $\boldsymbol{p}$, translate it by the amount $\boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon$ and rotate this shifted point by θ_0 around the axis $[\boldsymbol{u}_0, \boldsymbol{0}]$*. The next quaternion is the translation term: $[0, \theta_\epsilon \boldsymbol{u}_0 + \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon]$. The first part of this term, $[0, \theta_\epsilon \boldsymbol{u}_0]$ merely says "*Once you have shifted and rotated $\boldsymbol{p}$, move it by θ_ϵ along the vector $\boldsymbol{u}_0$*. The operation $q_0 \left(p - [0, \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon]\right) q_0^* + [0, \theta_\epsilon \boldsymbol{u}_0]$ is thus a screw motion of axis $[\boldsymbol{u}_0, 0]$ where $\boldsymbol{p} - \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon$ is rotated about θ_0 and translated by θ_ϵ. The last translation term is $[0, \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon]$ and states "*now that the screw motion of axis $[\boldsymbol{u}_0, \boldsymbol{0}]$ is done, shift everything back to its original position by translating the resulting point by $\boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon$ away from the origin*". With $\boldsymbol{p}_\perp = \boldsymbol{u}_0 \times \boldsymbol{u}_\epsilon$ the whole process is therefore "*perform a screw motion of point $\boldsymbol{p} - \boldsymbol{p}_\perp$ at the axis oriented by $\boldsymbol{u}_0$ and going through 0 and then add $\boldsymbol{p}_\perp$ to the result*" which rigorously states the same thing as "*Perform a screw motion of point $\boldsymbol{p}$ at the axis oriented by $\boldsymbol{u}_0$ and going through the point $\boldsymbol{p}_\perp$*", precisely the screw motion that we were looking for! On FIGURE 5.4 the blue bow is mapped to the red one using a dual quaternion whose axis is the line in black. The trajectory in blue shows the screw motion around the axis. Arrows on the boxes hopefully help to see their respective orientations.

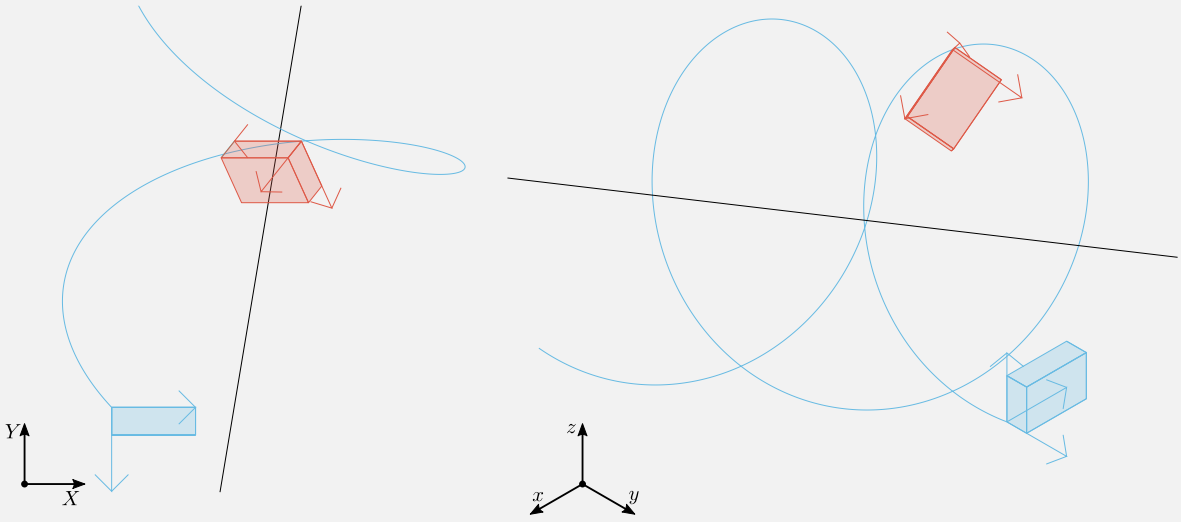


Figure 5.4| A unit dual quaternion maps one box to the other.

5.1.8.6 Finding the unit dual quaternion associated to a rigid body motion

Let us take a step back. This rather long exposition on unit dual quaternions has shown that they are particularly fit at encoding rigid body motions. When we looked at 2D assemblies in CHAPTER 4, we saw that

to generate them the user must provide a list of motions, translation and/or rotation. In 3D this principle stays the same, except that screw motions are much harder to intuitively see. As such the user may find it cumbersome to specify unit dual quaternions. Yet, if the user gives as well an approximate shape of the part, a so-called pseudo-part, (much like in 2D the user could input a polyline to be optimised), he/she can also move it in space at its resting position, and then ask to compute the unit dual quaternion mapping the former to the latter. This process is much more intuitive as it simply consists of placing and orienting a solid body in space. This section aims at calculating this unit dual quaternion.

Say we have two particles[2] $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$, like on FIGURE 5.5, and we want to find the unit dual quaternion $\hat{q} = q_0 + \epsilon q_\epsilon$ transforming $\boldsymbol{p}_A$ into $\boldsymbol{p}_B$: $\hat{q}_B = \hat{q}\hat{q}_A\hat{q}^\circ$ with $\hat{q}_A$ and $\hat{q}_B$ the dual quaternions representing points $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$. We are first going to find the quaternion q_0 which rotates the orthonormal basis associated with $\boldsymbol{p}_A$ into the one associated with $\boldsymbol{p}_B$ before computing q_ϵ.

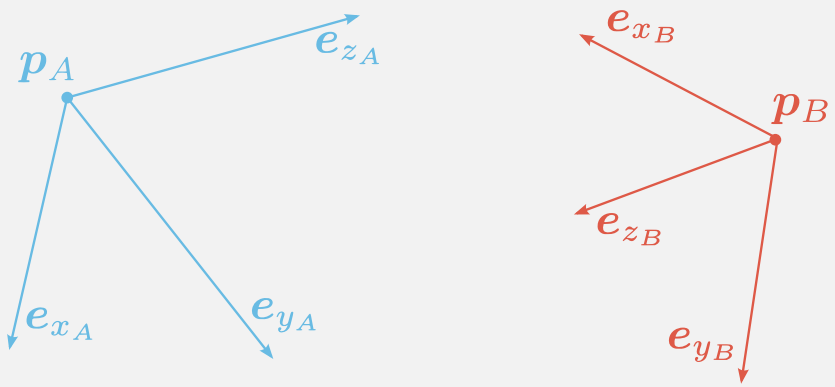


Figure 5.5| Two particles $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$.

Calculating q_0

Given $\begin{pmatrix} \boldsymbol{e}_{x_A} \\ \boldsymbol{e}_{y_A} \\ \boldsymbol{e}_{z_A} \end{pmatrix}$ and $\begin{pmatrix} \boldsymbol{e}_{x_B} \\ \boldsymbol{e}_{y_B} \\ \boldsymbol{e}_{z_B} \end{pmatrix}$ orthonormal basis, we wish to find the unit quaternion $q_0 = [c, \boldsymbol{u}_0 s]$ (with $c = \cos\frac{\theta_0}{2}$; $s = \sin\frac{\theta_0}{2}$ and $||\boldsymbol{u}_0|| = 1$) such that:

$$\begin{cases} q_0[0, \boldsymbol{e}_{x_A}]q_0^* = [0, \boldsymbol{e}_{x_B}] \\ q_0[0, \boldsymbol{e}_{y_A}]q_0^* = [0, \boldsymbol{e}_{y_B}] \\ q_0[0, \boldsymbol{e}_{z_A}]q_0^* = [0, \boldsymbol{e}_{z_B}] \end{cases} \qquad \star$$

$$(\star) \Leftrightarrow \begin{cases} (c^2 - s^2)\boldsymbol{e}_{x_A} + 2s^2(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A})\boldsymbol{u}_0 + 2sc(\boldsymbol{u}_0 \times \boldsymbol{e}_{x_A}) = \boldsymbol{e}_{x_B} \\ (c^2 - s^2)\boldsymbol{e}_{y_A} + 2s^2(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{y_A})\boldsymbol{u}_0 + 2sc(\boldsymbol{u}_0 \times \boldsymbol{e}_{y_A}) = \boldsymbol{e}_{y_B} \\ (c^2 - s^2)\boldsymbol{e}_{z_A} + 2s^2(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{z_A})\boldsymbol{u}_0 + 2sc(\boldsymbol{u}_0 \times \boldsymbol{e}_{z_A}) = \boldsymbol{e}_{z_B} \end{cases}$$

[2] A point and an orthonormal basis in $\mathbb{R}^3$

Since $\boldsymbol{e}_{i_A} \cdot \boldsymbol{e}_{j_A} = 0$ for $i, j \in \{x, y, z\}, i \neq j$ one has:

$$(\star) \Leftrightarrow \begin{cases} \begin{cases} 2s^2(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A})(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{y_A}) + 2sc(\boldsymbol{u}_0 \times \boldsymbol{e}_{x_A}) \cdot \boldsymbol{e}_{y_A} = \boldsymbol{e}_{x_B} \cdot \boldsymbol{e}_{y_A} \\ 2s^2(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A})(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{z_A}) + 2sc(\boldsymbol{u}_0 \times \boldsymbol{e}_{x_A}) \cdot \boldsymbol{e}_{z_A} = \boldsymbol{e}_{x_B} \cdot \boldsymbol{e}_{z_A} \end{cases} \\ \begin{cases} 2s^2(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{y_A})(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A}) + 2sc(\boldsymbol{u}_0 \times \boldsymbol{e}_{y_A}) \cdot \boldsymbol{e}_{x_A} = \boldsymbol{e}_{y_B} \cdot \boldsymbol{e}_{x_A} \\ 2s^2(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{y_A})(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{z_A}) + 2sc(\boldsymbol{u}_0 \times \boldsymbol{e}_{y_A}) \cdot \boldsymbol{e}_{z_A} = \boldsymbol{e}_{y_B} \cdot \boldsymbol{e}_{z_A} \end{cases} \\ \begin{cases} 2s^2(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{z_A})(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A}) + 2sc(\boldsymbol{u}_0 \times \boldsymbol{e}_{z_A}) \cdot \boldsymbol{e}_{x_A} = \boldsymbol{e}_{z_B} \cdot \boldsymbol{e}_{x_A} \\ 2s^2(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{z_A})(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{y_A}) + 2sc(\boldsymbol{u}_0 \times \boldsymbol{e}_{z_A}) \cdot \boldsymbol{e}_{y_A} = \boldsymbol{e}_{z_B} \cdot \boldsymbol{e}_{y_A} \end{cases} \end{cases}$$

Subtracting (we only develop the equations for one set of equations, the others being obtained by permuting $\{x, y, z\}$):

$$(\star) \Leftrightarrow \begin{cases} 2sc\left(-(\boldsymbol{u}_0 \times \boldsymbol{e}_{y_A}) \cdot \boldsymbol{e}_{x_A} + (\boldsymbol{u}_0 \times \boldsymbol{e}_{x_A}) \cdot \boldsymbol{e}_{y_A}\right) = -\boldsymbol{e}_{y_B} \cdot \boldsymbol{e}_{x_A} + \boldsymbol{e}_{x_B} \cdot \boldsymbol{e}_{y_A} \\ \ldots \end{cases}$$

Knowing that $b \cdot (c \times a) = a \cdot (b \times c)$:

$$\begin{aligned} (\star) &\Leftrightarrow \begin{cases} 2sc\left(-\boldsymbol{u}_0 \cdot (\boldsymbol{e}_{y_A} \times \boldsymbol{e}_{x_A}) + \boldsymbol{u}_0 \cdot (\boldsymbol{e}_{x_A} \times \boldsymbol{e}_{y_A})\right) = \boldsymbol{e}_{x_B} \cdot \boldsymbol{e}_{y_A} - \boldsymbol{e}_{y_B} \cdot \boldsymbol{e}_{x_A} \\ \ldots \end{cases} \\ &\Leftrightarrow \begin{cases} 4sc(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{z_A}) = \boldsymbol{e}_{x_B} \cdot \boldsymbol{e}_{y_A} - \boldsymbol{e}_{y_B} \cdot \boldsymbol{e}_{x_A} \\ \ldots \end{cases} \end{aligned}$$

Let

$$\begin{aligned} a_x &= \boldsymbol{e}_{x_B} \cdot \boldsymbol{e}_{y_A} - \boldsymbol{e}_{y_B} \cdot \boldsymbol{e}_{x_A} \\ a_y &= \boldsymbol{e}_{y_B} \cdot \boldsymbol{e}_{z_A} - \boldsymbol{e}_{z_B} \cdot \boldsymbol{e}_{y_A} \\ a_z &= \boldsymbol{e}_{z_B} \cdot \boldsymbol{e}_{x_A} - \boldsymbol{e}_{x_B} \cdot \boldsymbol{e}_{z_A} \end{aligned}$$

Then (since $4sc = 2\sin\theta_0$)

$$(\star) \Leftrightarrow \begin{cases} 2(\boldsymbol{e}_{z_A} \cdot \boldsymbol{u}_0)\sin\theta_0 = a_x \\ 2(\boldsymbol{e}_{x_A} \cdot \boldsymbol{u}_0)\sin\theta_0 = a_y \\ 2(\boldsymbol{e}_{y_A} \cdot \boldsymbol{u}_0)\sin\theta_0 = a_z \end{cases}$$

And so, assuming $\theta_0 \notin \{0, \pi\}$:

$$(\star) \Leftrightarrow \begin{pmatrix} \boldsymbol{e}_{z_A}^T \\ \boldsymbol{e}_{x_A}^T \\ \boldsymbol{e}_{y_A}^T \end{pmatrix} \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} = \frac{1}{2\sin\theta_0} \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$$

Where $\boldsymbol{u}_0 = (u_x, u_y, u_z)^T$.

Let $\boldsymbol{M} = \begin{pmatrix} \boldsymbol{e}_{z_A}^T \\ \boldsymbol{e}_{x_A}^T \\ \boldsymbol{e}_{y_A}^T \end{pmatrix}$ and $\boldsymbol{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$. $\boldsymbol{M}$ is orthonormal and thus invertible with $\boldsymbol{M}^{-1} = \boldsymbol{M}^T$. Assuming

$\boldsymbol{a} \neq \boldsymbol{0}$ one has:

$$(\star) \Leftrightarrow \boldsymbol{u}_0 = \frac{1}{2\sin\theta_0}\boldsymbol{M}^T\boldsymbol{a}$$

Recall that $||\boldsymbol{u_0}|| = 1$ and so:

$$\begin{aligned}
||\boldsymbol{u_0}|| = 1 &\Leftrightarrow \boldsymbol{u}_0^T\boldsymbol{u}_0 = 1 \\
&\Leftrightarrow \frac{1}{4\sin^2\theta_0}\boldsymbol{a}^T\boldsymbol{M}\boldsymbol{M}^T\boldsymbol{a} = 1 \\
&\Leftrightarrow \boldsymbol{a}^T\boldsymbol{a} = 4\sin^2\theta_0 \\
&\Leftrightarrow \sin^2\theta_0 = \frac{1}{4}||\boldsymbol{a}||^2 \\
&\Leftrightarrow \theta_0 = \arcsin\left(\pm\frac{1}{2}||\boldsymbol{a}||\right)
\end{aligned}$$

Two angles, negatives of each other, are solutions to the problem. As these angles are used to compute the vector $\boldsymbol{u}_0 = \frac{1}{2\sin\theta_0}\boldsymbol{M}^T\boldsymbol{a}$ we obtain two vectors $\boldsymbol{u}_0$, each the opposite of the other. The final rotation is therefore the same whether we choose one θ_0 (and thus one $\boldsymbol{u}_0$) or the other as rotating by an angle around an axis is the same as rotating by the opposite angle around the reversed axis (all rotations are anticlockwise). Hence we could safely choose $\theta_0 = \arcsin\frac{1}{2}||\boldsymbol{a}||$ and $\boldsymbol{u}_0 = \frac{\boldsymbol{M}^T\boldsymbol{a}}{||\boldsymbol{a}||}$. Yet, because the function $x \mapsto \arcsin x$ takes values in $[-\frac{\pi}{2}, \frac{\pi}{2}]$ the θ_0 obtained might not reflect the actual angle between $\boldsymbol{p}_A$'s basis and $\boldsymbol{p}_B$'s basis. To know more about θ_0 we need to look at, say, the equation (with $c = \cos\frac{\theta_0}{2}$ and $s = \sin\frac{\theta_0}{2}$):

$$\begin{aligned}
\boldsymbol{e}_{x_B} &= (c^2 - s^2)\boldsymbol{e}_{x_A} + 2s^2(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A})\boldsymbol{u}_0 + 2sc(\boldsymbol{u}_0 \times \boldsymbol{e}_{x_A}) \\
&= \cos\theta_0\boldsymbol{e}_{x_A} + (1 - \cos\theta_0)(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A})\boldsymbol{u}_0 + 2sc(\boldsymbol{u}_0 \times \boldsymbol{e}_{x_A})
\end{aligned}$$

Projecting on $\boldsymbol{e}_{x_A}$ yields:

$$\cos\theta_0 + (1 - \cos\theta_0)(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A})^2 = \boldsymbol{e}_{x_B} \cdot \boldsymbol{e}_{x_A}$$

And:

$$\theta_0 = \arccos\frac{\boldsymbol{e}_{x_B} \cdot \boldsymbol{e}_{x_A} - (\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A})^2}{1 - (\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A})^2} \tag{5.15}$$

All in all, since $x \mapsto \arccos x$ takes value in [0, π], if $\arccos\frac{\boldsymbol{e}_{x_B} \cdot \boldsymbol{e}_{x_A} - (\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A})^2}{1 - (\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A})^2} < \frac{\pi}{2}$ then $\theta_0 = \arcsin\frac{||\boldsymbol{a}||}{2}$, else $\theta_0 = \pi - \arcsin\frac{||\boldsymbol{a}||}{2}$.
This can be rewritten as $\theta_0 = \operatorname{atan2}\left(\frac{1}{2}||\boldsymbol{a}||, \frac{\boldsymbol{e}_{x_B} \cdot \boldsymbol{e}_{x_A} - (\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A})^2}{1 - (\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A})^2}\right)$, where $\operatorname{atan2}$ is a function provided in most programming languages. In a nutshell the quaternion q_0 looked for is given by (assuming $\boldsymbol{a} \neq \boldsymbol{0}$):

$$\begin{aligned}
q_0 &= \left[\cos\frac{\theta_0}{2}, \boldsymbol{u}_0\sin\frac{\theta_0}{2}\right] \\
&\begin{cases}
\theta_0 = \operatorname{atan2}\left(\frac{1}{2}||\boldsymbol{a}||, \frac{\boldsymbol{e}_{x_B} \cdot \boldsymbol{e}_{x_A} - (\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A})^2}{1 - (\boldsymbol{u}_0 \cdot \boldsymbol{e}_{x_A})^2}\right) \\
\boldsymbol{u}_0 = \frac{\boldsymbol{M}^T\boldsymbol{a}}{||\boldsymbol{a}||}
\end{cases}
\end{aligned}$$

If $\boldsymbol{a} = \boldsymbol{0}$ it is easy to see that either $\theta_0 = 0$ or $\theta_0 = \pi$: indeed recall that:

$$(\star) \Leftrightarrow \begin{cases}
2(\boldsymbol{e}_{z_A} \cdot \boldsymbol{u}_0)\sin\theta_0 = a_x \\
2(\boldsymbol{e}_{x_A} \cdot \boldsymbol{u}_0)\sin\theta_0 = a_y \\
2(\boldsymbol{e}_{y_A} \cdot \boldsymbol{u}_0)\sin\theta_0 = a_z
\end{cases}$$

Hence, since $\boldsymbol{u}_0$ cannot be orthogonal to $\boldsymbol{e}_{x_A}, \boldsymbol{e}_{x_A}$ and $\boldsymbol{e}_{z_A}$ at the same time, necessarily $\boldsymbol{a} = \boldsymbol{0} \Leftrightarrow \sin\theta_0 = 0$ and so $\theta_0 = 0$ or $\theta_0 = \pi$. To know whichever is true we only need to check the value of the $\boldsymbol{e}_{i_A} \cdot \boldsymbol{e}_{i_B}$ for $i \in \{x, y, z\}$: if for all i, $\boldsymbol{e}_{i_A} \cdot \boldsymbol{e}_{i_B} = 1$ then $\theta_0 = 0$ and $q_0 = [1, 0]$. Else, we have $\theta_0 = \pi$, $c = 0$ and $s = 1$ and we are left to find the vector $\boldsymbol{u}_0$

$$
\begin{aligned}
(\star) &\Leftrightarrow \begin{cases} 2(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{xA})\boldsymbol{u}_0 - \boldsymbol{e}_{xA} = \boldsymbol{e}_{xB} \\ \dots \end{cases} \\
&\Leftrightarrow \begin{cases} 2(\boldsymbol{u}_0 \cdot \boldsymbol{e}_{xA})\underbrace{\boldsymbol{u}_0 \cdot \boldsymbol{u}_0}_{=1} - \boldsymbol{e}_{xA} \cdot \boldsymbol{u}_0 = \boldsymbol{e}_{xB} \cdot \boldsymbol{u}_0 \\ \dots \end{cases} \\
&\Leftrightarrow \begin{cases} (\boldsymbol{e}_{xA} - \boldsymbol{e}_{xB}) \cdot \boldsymbol{u}_0 = 0 \\ (\boldsymbol{e}_{yA} - \boldsymbol{e}_{yB}) \cdot \boldsymbol{u}_0 = 0 \\ (\boldsymbol{e}_{zA} - \boldsymbol{e}_{zB}) \cdot \boldsymbol{u}_0 = 0 \end{cases}
\end{aligned}
$$

Let $\boldsymbol{B}$ be the matrix:

$$
\boldsymbol{B} = \begin{pmatrix} (\boldsymbol{e}_{xA} - \boldsymbol{e}_{xB})^T \\ (\boldsymbol{e}_{yA} - \boldsymbol{e}_{yB})^T \\ (\boldsymbol{e}_{zA} - \boldsymbol{e}_{zB})^T \end{pmatrix}
$$

Then

$$
\begin{aligned}
(\star) &\Leftrightarrow \boldsymbol{B}\boldsymbol{u}_0 = 0 \\
&\Leftrightarrow \begin{cases} \boldsymbol{u}_0 \in \ker(\boldsymbol{B}) \\ ||\boldsymbol{u}_0|| = 1 \end{cases}
\end{aligned} \tag{5.16}
$$

The basis of the nullspace of $\boldsymbol{B}$ can be calculated by performing a $\boldsymbol{QR}$ factorization of $\boldsymbol{B}^T$ and taking the column of $\boldsymbol{Q}$ that is indexed by a null coefficient on the diagonal of $\boldsymbol{R}$. Now that we have calculated q_0 we can focus on q_ϵ.

Calculating q_ϵ

Recall that we can always express the conjugation of $\hat{q}_A$ (the dual quaternion representing point $\boldsymbol{p}_A$) as a rotation by θ_0 around the axis $\boldsymbol{u}_0$ followed by a translation $\boldsymbol{t}$ represented by the quaternion $t = [0, \boldsymbol{t}]$ such that $t = 2q_\epsilon q_0^*$: $\hat{q}\hat{q_A}\hat{q}^* = 1 + \epsilon(q_0 q_A q_0^* + t)$. The translation quaternion t is trivially given as $t = q_B - q_0 q_A q_0^*$, and $q_\epsilon = \frac{1}{2} t q_0$. As such, $s = \sin\frac{\theta_0}{2}$ and $c = \cos\frac{\theta_0}{2}$:

$$
\begin{aligned}
t &= q_B - q_0 q_A q_0^* \\
&= [0, \boldsymbol{p}_B] - [0, 2s^2(\boldsymbol{u}_0 \cdot \boldsymbol{p}_A)\boldsymbol{u}_0 + (c^2 - s^2)\boldsymbol{p}_A + 2sc(\boldsymbol{u}_0 \times \boldsymbol{p}_A)] \\
\boldsymbol{t} &= \boldsymbol{p}_B - (1 - \cos\theta_0)(\boldsymbol{u}_0 \cdot \boldsymbol{p}_A)\boldsymbol{u}_0 - \cos\theta_0 \boldsymbol{p}_A - \sin\theta_0(\boldsymbol{u}_0 \times \boldsymbol{p}_A)
\end{aligned} \tag{5.17}
$$

Finding the screw axis

Now that we have $\hat{q} = q_0 + \epsilon q_\epsilon$ we may want to find the screw axis, i.e. to rewrite $\hat{q}$ as $\hat{q} = \left[\cos\frac{\hat{\theta}}{2}, \hat{\boldsymbol{u}}\sin\frac{\hat{\theta}}{2}\right]$ where $\hat{\theta} = \theta_0 + \epsilon\theta_\epsilon$ and $\hat{\boldsymbol{u}} = \boldsymbol{u}_0 + \epsilon\boldsymbol{u}_\epsilon$ such as the screw axis is given by the Plücker coordinates $[\boldsymbol{u}_0, \boldsymbol{u}_\epsilon]$ and the magnitude of the translation is θ_ϵ.

We have proven earlier that with such notations one can write q_ϵ as:

$$q_\epsilon = [-\frac{\theta_\epsilon}{2}s, \boldsymbol{u}_0\frac{\theta_\epsilon}{2}c + \boldsymbol{u}_\epsilon s]$$

Where $s = \sin\frac{\theta_0}{2}$ and $c = \cos\frac{\theta_0}{2}$. Also, $q_\epsilon = \frac{1}{2}tq_0$.
Assuming $\theta_0 \notin \{0, \pi\}$:

$$\begin{aligned} q_\epsilon &= \frac{1}{2}tq_0 \\ &= \frac{1}{2}[0, \boldsymbol{t}][c, \boldsymbol{u}_0 s] \\ &= \frac{1}{2}[-s\boldsymbol{u_0}\cdot\boldsymbol{t}, c\boldsymbol{t} + s\boldsymbol{t}\times\boldsymbol{u}_0] \end{aligned}$$

By identification:

$$\theta_\epsilon = \boldsymbol{u}_0\cdot\boldsymbol{t}$$

And:

$$\begin{aligned} \boldsymbol{u}_0\theta_\epsilon + 2s\boldsymbol{u}_\epsilon &= c\boldsymbol{t} + s\boldsymbol{t}\times\boldsymbol{u}_0 \\ 2s\boldsymbol{u}_\epsilon &= c\boldsymbol{t} + s\boldsymbol{t}\times\boldsymbol{u}_0 - c(\boldsymbol{u}_0\cdot\boldsymbol{t})\boldsymbol{u}_0 \\ &= c\left((\boldsymbol{u}_0\cdot\boldsymbol{u_0})\boldsymbol{t} - (\boldsymbol{u}_0\cdot\boldsymbol{t})\boldsymbol{u}_0\right) + s\boldsymbol{t}\times\boldsymbol{u}_0 \\ &= c\boldsymbol{u}_0\times(\boldsymbol{t}\times\boldsymbol{u}_0) + s\boldsymbol{t}\times\boldsymbol{u}_0 \end{aligned}$$

And thus:

$$\boldsymbol{u}_\epsilon = \frac{1}{2}\left(\cot\frac{\theta_0}{2}\boldsymbol{u}_0\times(\boldsymbol{t}\times\boldsymbol{u}_0) + \boldsymbol{t}\times\boldsymbol{u}_0\right)$$

If $\theta_0 = 0$ then the vector $\boldsymbol{u}_0$ can be defined as $\boldsymbol{u}_0 = \frac{\boldsymbol{p}_B - \boldsymbol{p}_A}{||\boldsymbol{p}_B - \boldsymbol{p}_A||}$ and $\theta_\epsilon = ||\boldsymbol{p}_B - \boldsymbol{p}_A||$. Since this case is about a pure translation the screw axis goes through point $\boldsymbol{p}_A$ and $\boldsymbol{u}_\epsilon = \boldsymbol{p}_A\times\boldsymbol{u}_0$.
Finally if $\theta_0 = \pi$, θ_ϵ is still given as $\theta_\epsilon = \boldsymbol{u}_0\cdot\boldsymbol{t}$ and we know that the screw axis goes through point $\frac{\boldsymbol{p}_A+\boldsymbol{p}_B}{2}$. Hence $\boldsymbol{u}_\epsilon = \frac{\boldsymbol{p}_A+\boldsymbol{p}_B}{2}\times\boldsymbol{u}_0$.

This section has shown, how given the same rigid body in two different positions and orientations, one finds the unit dual quaternion mapping one to the other, and as such the user can intuitively place and rotate pseudo-parts in their disassembled state and let the aforementioned algorithm find the unit dual quaternions mapping each pseudo-part in its assembled position to its counterpart in the disassembled state. We almost have everything ready to generate 3D assemblies, we only need to find the instantaneous direction of motion of a point obeying a unit dual quaternion $\hat{q}$.

5.2 CONE OF INFINITESIMAL FREEDOM OF MOTION IN 3D

5.2.1 QUATERNIONS, DUAL QUATERNIONS AND TANGENTIAL VELOCITY

5.2.1.1 Tangential velocity and quaternions

We may be interested in the tangential velocity along the trajectory of a point $\boldsymbol{p}\in\mathbb{R}^3$ subjected to a rotation through a quaternion (not a dual one) $q = \left[\cos\frac{\theta}{2}, \boldsymbol{u}\sin\frac{\theta}{2}\right]$. Indeed, the tangential velocity is what we called the **instantaneous direction of motion** in CHAPTER 4. Such trajectory, in $\mathbb{R}^3$, is an circular arc bounded by

$\boldsymbol{p}$ and the vector part of $p_r = qpq^*$ ($p = [0, \boldsymbol{p}]; p_r = [0, \boldsymbol{p}_r]$), centered in 0 and in the plane of normal $\boldsymbol{u}$. Any point on this arc can be parametrised by a given $\lambda \in [0, 1]$ such that $p(\lambda) = q(\lambda)pq^*(\lambda)$ with $q(\lambda) = \left[\cos\frac{\lambda\theta}{2}, \boldsymbol{u}\sin\frac{\lambda\theta}{2}\right]$. Since $q(0) = [1, 0]$ and $q(1) = q$, one has that $p(0) = p$ and $p(1) = p_r$. Thus, the mapping

$$\begin{array}{rccc} p & : & [0,1] & \longrightarrow & \mathbb{H} \\ & & \lambda & \mapsto & p(\lambda) \end{array}$$

is a bijection between the unit segment and the arc of a circle mentioned above (embedded in $\mathbb{H}$).

The tangential velocity at a point $\boldsymbol{p}(\lambda)$ is therefore the vector part of $\frac{\partial qpq^*}{\partial\lambda}(\lambda)$. Introducing λ in EQUATION (D.4) reads:

$$\begin{aligned} p(\lambda) &= q(\lambda)p(0)q^*(\lambda) \\ &= [0, (1-\cos(\lambda\theta))(\boldsymbol{u}\cdot\boldsymbol{p})\boldsymbol{u} + \cos(\lambda\theta)\boldsymbol{p} + \sin(\lambda\theta)(\boldsymbol{u}\times\boldsymbol{p})] \end{aligned}$$

A simple derivation gives that the tangential velocity $\boldsymbol{v}(\lambda) \in \mathbb{R}^3$ is:

$$\boldsymbol{v}(\lambda) = \theta\left(\sin(\lambda\theta)((\boldsymbol{u}\cdot\boldsymbol{p})\boldsymbol{u} - \boldsymbol{p}) + \cos(\lambda\theta)(\boldsymbol{u}\times\boldsymbol{p})\right) \tag{5.18}$$

In particular, the initial velocity, also called the instantaneous direction of motion of $\boldsymbol{p}$ with respect to q, is given by:

$$\boxed{\boldsymbol{v}(0) = \theta(\boldsymbol{u}\times\boldsymbol{p})}$$

5.2.1.2 Tangential velocity and dual quaternions

The same study can be done with dual quaternions. In a similar fashion as in SECTION 5.2.1.1, let us introduce a parameter λ such that $\hat{q} = \left[\cos\frac{\lambda\hat{\theta}}{2}, \hat{\boldsymbol{u}}\sin\frac{\lambda\hat{\theta}}{2}\right]$. EQUATION (5.14) then reads

$$\hat{q}\hat{p}\hat{q}^\circ = 1 + \epsilon(q_0(\lambda)(p - [0, \boldsymbol{u}_0\times\boldsymbol{u}_\epsilon])q_0^*(\lambda) + [0, \lambda\theta_\epsilon\boldsymbol{u}_0 + \boldsymbol{u}_0\times\boldsymbol{u}_\epsilon]$$

And the tangential velocity becomes:

$$\boldsymbol{v}(\lambda) = \theta_0\left(\sin(\lambda\theta_0)\left((\boldsymbol{u}_0\cdot\boldsymbol{p})\boldsymbol{u}_0 - \boldsymbol{p} + \boldsymbol{u}_0\times\boldsymbol{u}_\epsilon\right) + \cos(\lambda\theta_0)\left(\boldsymbol{u}_0\times\boldsymbol{p} + \boldsymbol{u}_\epsilon\right) +\right) + \theta_\epsilon\boldsymbol{u}_0$$

In particular, the initial velocity, or instantaneous direction of motion of $\boldsymbol{p}$ with respect to $\hat{q}$ is given by:

$$\boxed{\boldsymbol{v}(0) = \boldsymbol{m}(\boldsymbol{p}, \hat{q}) = \theta_0(\boldsymbol{u}_0\times\boldsymbol{p} + \boldsymbol{u}_\epsilon) + \theta_\epsilon\boldsymbol{u}_0}$$

When the unit dual quaternion is not ambiguous, we will abbreviate $\boldsymbol{m}(\boldsymbol{p}, \hat{q})$ as $\boldsymbol{m}_p$. FIGURE 5.6 shows the instantaneous directions of motions (blue vectors) of regularly spaced points on a plane for various $\hat{q}$ (with constant $\boldsymbol{u}_0$ and $\boldsymbol{u}_\epsilon$ and varying $\theta_0 = \cos\theta$ and $\theta_\epsilon = \sin\theta$): the red line shows the screw axis, and the red curve is the trajectory of the highlighted red point for an extended motion encoded in $\hat{q}$. On the left a pure translation is obtained for $\theta = \pm\frac{\pi}{2}$, in the middle a pure rotation is obtained for $\theta = k\pi$, $k \in \mathbb{Z}$ and on the right a screw motion is given for $\theta = 0.5$.

5.2.2 THEORETICAL RESULTS

This section aims at finding the set of unit dual quaternions that can be obeyed by a given separating surface. By definition a surface Σ (be it smooth or discrete) obeys a motion $\hat{q}$ if and only if each of its constitutive

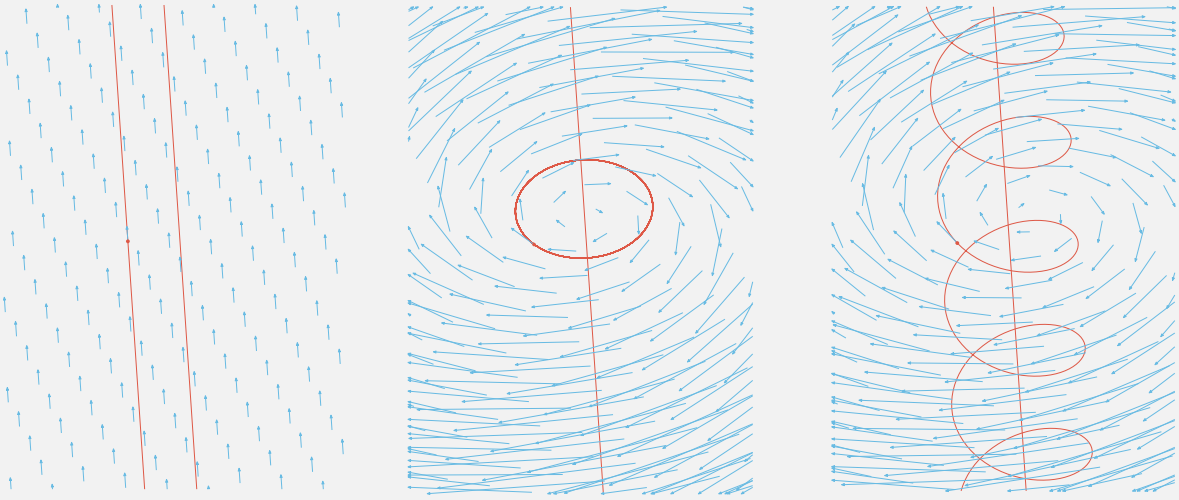


Figure 5.6| Instantaneous directions of motion for three different unit dual quaternions $\hat{q}$.

points obey $\hat{q}$:

$$\Sigma \text{ obeys } \hat{q} \iff \forall \boldsymbol{v} \in \Sigma,\ \boldsymbol{n}_v \cdot \boldsymbol{m}_v \geq 0$$

where $\boldsymbol{v}$ is a point of the surface, $\boldsymbol{n}_v$ the normal vector at that point and $\boldsymbol{m}_v$ the instantaneous direction of motion of $\boldsymbol{v}$ with respect to $\hat{q}$ that is defined just above, see FIGURE 5.7 for an illustration.

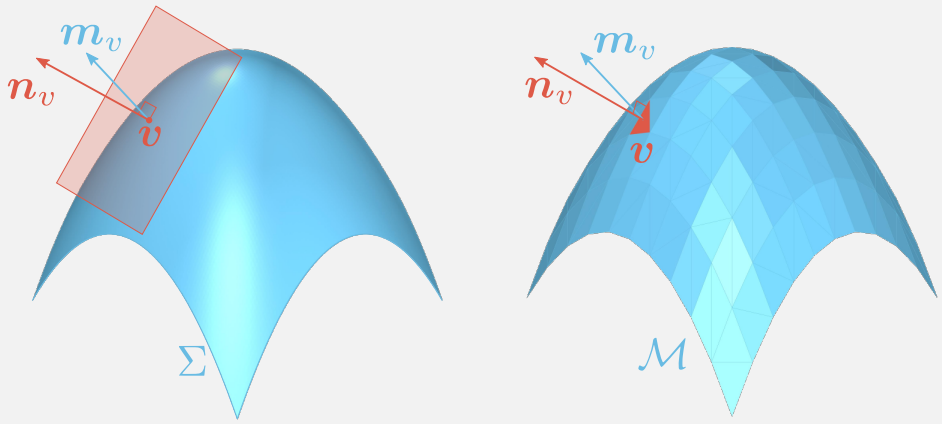


Figure 5.7| Smooth and discrete surfaces obeying a unit dual quaternion $\hat{q}$.

Notations:
Let $\mathcal{M} = (V, E, F)$ a discrete surface represented by a triangular mesh: V is its set of vertices, E of edges and F of faces. For a face $f \in F$ let $V(f) = \{\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{v}_k\}$ refers to its vertices in a counterclockwise order; let $\boldsymbol{n}_f$ be the normal of face f.

In this section we prove the following equivalence:

$$\boxed{\mathcal{M} \text{ obeys } \hat{q} \iff \forall f \in F\, \forall \boldsymbol{v} \in V(f)\ \boldsymbol{m}_v \cdot \boldsymbol{n}_f \geq 0}$$

which implies that to check whether a triangular mesh obeys a motion $\hat{q}$, it suffices to check for the mesh vertices.

Proof.

- Proof of the implication $\implies$:
  If $\mathcal{M}$ obeys $\hat{q}$, then by definition all points $\boldsymbol{v}$ on the surface (vertices or not) are such that $\boldsymbol{n}_v \cdot \boldsymbol{m}_v \geq 0$.
  It is true in particular for the vertices of the mesh which is proof of the implication $\implies$.
- Proof of the implication $\impliedby$:
  Assume that $\forall f \in F \, \forall \boldsymbol{v} \in V(f) \; \boldsymbol{m}_v \cdot \boldsymbol{n}_f \geq 0$. Let us show that $\mathcal{M}$ obeys $\hat{q}$.
  Let $f \in F$ be a face, and $\boldsymbol{v}_i, \boldsymbol{v}_j$ and $\boldsymbol{v}_k$ its defining vertices. Let $\boldsymbol{v} \in f$ be any point in the triangle. As f is a triangle,

$$\exists (\lambda_i, \lambda_j, \lambda_k) \in [0,1]^3 \text{ with } \lambda_i + \lambda_j + \lambda_k = 1 \text{ such that } \boldsymbol{v} = \lambda_i \boldsymbol{v}_i + \lambda_j \boldsymbol{v}_j + \lambda_k \boldsymbol{v}_k$$

  And

$$\begin{aligned}
\boldsymbol{m}_v &= \theta_0(\boldsymbol{u}_0 \times \boldsymbol{v}) + \theta_0 \boldsymbol{u}_\epsilon + \theta_\epsilon \boldsymbol{u}_0 \\
&= \theta_0 \left(\boldsymbol{u}_0(\lambda_i \boldsymbol{v}_i + \lambda_j \boldsymbol{v}_j + \lambda_k \boldsymbol{v}_k)\right) + \underbrace{(\lambda_i + \lambda_j + \lambda_k)}_{=1}(\theta_0 \theta_0 \boldsymbol{u}_\epsilon + \theta_\epsilon \boldsymbol{u}_0) \\
&= \lambda_i(\theta_0(\boldsymbol{u}_0 \times \boldsymbol{v}_i) + \theta_0 \boldsymbol{u}_\epsilon + \theta_\epsilon \boldsymbol{u}_0) + \lambda_j(\theta_0(\boldsymbol{u}_0 \times \boldsymbol{v}_j) + \theta_0 \boldsymbol{u}_\epsilon + \theta_\epsilon \boldsymbol{u}_0) + \lambda_k(\theta_0(\boldsymbol{u}_0 \times \boldsymbol{v}_k) + \theta_0 \boldsymbol{u}_\epsilon + \theta_\epsilon \boldsymbol{u}_0) \\
&= \lambda_i \boldsymbol{m}_{v_i} + \lambda_j \boldsymbol{m}_{v_j} + \lambda_k \boldsymbol{m}_{v_k}
\end{aligned}$$

  Therefore:

$$\boldsymbol{m}_v \cdot \boldsymbol{n}_f = \underbrace{\lambda_i}_{\geq 0} \underbrace{\boldsymbol{m}_{v_i} \cdot \boldsymbol{n}_f}_{\geq 0} + \underbrace{\lambda_j}_{\geq 0} \underbrace{\boldsymbol{m}_{v_j} \cdot \boldsymbol{n}_f}_{\geq 0} + \underbrace{\lambda_k}_{\geq 0} \underbrace{\boldsymbol{m}_{v_k} \cdot \boldsymbol{n}_f}_{\geq 0} \geq 0$$

  which shows that $\boldsymbol{v}$ obeys $\hat{q}$.
  This being true $\forall \boldsymbol{v} \in f$ and $\forall f \in F$ we have successfully shown the implication $\impliedby$, which ends the proof.

□

5.2.3 COMPUTATION OF THE CONE OF FREEDOM

Let $\boldsymbol{m}$ be the function mapping a point and a unit dual quaternion to the instantaneous direction of motion of that point when moved using that dual quaternion:

$$\begin{array}{rccc}
\boldsymbol{m} & : & \mathbb{R}^3 \times \mathcal{U}(\mathbb{DH}) & \longrightarrow & \mathbb{R}^3 \\
 & & (\boldsymbol{p}, \hat{q}) & \mapsto & \theta_0(\boldsymbol{u}_0 \times \boldsymbol{p}) + \theta_0 \boldsymbol{u}_\epsilon + \theta_\epsilon \boldsymbol{u}_0
\end{array}$$

Let $\mathcal{M} = (V, E, F)$ be a triangular mesh embedded in $\mathbb{R}^3$. For a face $f \in F$ let the set $V(f)$ denote the three vertices in V defining the face f and $\boldsymbol{n}_f$ be a unit normal vector of that face (consistently defined across all faces in F).
Let $\hat{\mathcal{C}}_\mathcal{M}$ be the set of unit dual quaternions such that $\mathcal{M}$ may obey the motion encoded by each $\hat{q} \in \hat{\mathcal{C}}_\mathcal{M}$:

$$\hat{\mathcal{C}}_\mathcal{M} = \{\hat{q} \in \mathcal{U}(\mathbb{DH}), \, \forall f \in F \; \forall \boldsymbol{v} \in V(f) \; \boldsymbol{m}(\boldsymbol{v}, \hat{q}) \cdot \boldsymbol{n}_f \geq 0\}$$

The aim of this section is to find $\hat{\mathcal{C}}_\mathcal{M}$.
Let $f \in F$ of normal vector $\boldsymbol{n}_f$, $\boldsymbol{v} \in V(f)$ and $\hat{q} \in \mathcal{U}(\mathbb{DH})$. Then $\boldsymbol{m}(\boldsymbol{v}, \hat{q}) = \theta_0(\boldsymbol{u}_0 \times \boldsymbol{v}) + \theta_0 \boldsymbol{u}_\epsilon + \theta_\epsilon \boldsymbol{u}_0$. This vector can be rewritten as

$$\boldsymbol{m}(\boldsymbol{v}, \hat{q}) = \boldsymbol{\Omega} \times \boldsymbol{v} + \boldsymbol{t}$$

with

$$\begin{cases} \boldsymbol{\Omega} = \theta_0 \boldsymbol{u}_0 \\ \boldsymbol{t} = \theta_0 \boldsymbol{u}_\epsilon + \theta_\epsilon \boldsymbol{u}_0 \end{cases}$$

and we recognise in this expression the usual expression of the tangential velocity of a rigid body. Thus

$$\begin{aligned} \hat{q} \in \hat{\mathcal{C}}_{\mathcal{M}} &\implies \boldsymbol{m}(\boldsymbol{v}, \hat{q}) \cdot \boldsymbol{n}_f \geq 0 \\ &\Leftrightarrow (\boldsymbol{\Omega} \times \boldsymbol{v} + \boldsymbol{t}) \cdot \boldsymbol{n}_f \geq 0 \\ &\Leftrightarrow (\boldsymbol{v} \times \boldsymbol{n}_f) \cdot \boldsymbol{\Omega} + \boldsymbol{n}_f \cdot \boldsymbol{t} \geq 0 \\ &\Leftrightarrow \boldsymbol{r}^T \boldsymbol{x} \geq 0 \end{aligned}$$

With

$$\boldsymbol{r} = \begin{pmatrix} \boldsymbol{n}_f \\ \boldsymbol{v} \times \boldsymbol{n}_f \end{pmatrix} \in \mathbb{R}^6 \qquad \boldsymbol{x} = \begin{pmatrix} \boldsymbol{t} \\ \boldsymbol{\Omega} \end{pmatrix} \in \mathbb{R}^6$$

And thus $\hat{q} \in \hat{\mathcal{C}}_{\mathcal{M}}$ if and only if this latest inequality is true for each vertex of each face of $\mathcal{M}$:

$$\hat{q} \in \hat{\mathcal{C}}_{\mathcal{M}} \Leftrightarrow A_{\mathcal{M}} \boldsymbol{x} \geq 0 \tag{5.19}$$

where

$$A_{\mathcal{M}} = \begin{pmatrix} \boldsymbol{n}_{f_1} & \boldsymbol{v}_{f_1}^1 \times \boldsymbol{n}_{f_1} \\ \boldsymbol{n}_{f_1} & \boldsymbol{v}_{f_1}^2 \times \boldsymbol{n}_{f_1} \\ \boldsymbol{n}_{f_1} & \boldsymbol{v}_{f_1}^3 \times \boldsymbol{n}_{f_1} \\ \boldsymbol{n}_{f_2} & \boldsymbol{v}_{f_2}^1 \times \boldsymbol{n}_{f_2} \\ \vdots & \vdots \\ \boldsymbol{n}_{f_{|F|}} & \boldsymbol{v}_{f_{|F|}}^3 \times \boldsymbol{n}_{f_{|F|}} \end{pmatrix} \in \mathbb{R}^{3|F| \times 6}$$

and for $i \in [\![1, |F|]\!]$, f_i denotes the i^{th} face of mesh $\mathcal{M}$, and for $j \in \{1, 2, 3\}$, $\boldsymbol{v}_{f_i}^j$ denotes the j^{th} vertex of $V(f_i)$.

EQUATION (5.19) defines a 6-dimensional cone whose tip is on $\mathbf{0}$: if $\boldsymbol{x}$ satisfies EQUATION (5.19) then $\lambda \boldsymbol{x}$, $\lambda \geq 0$, does too. To find a conical section of this cone let us assume that we have a solution $\tilde{\boldsymbol{x}}$ satisfying EQUATION (5.19). A conical section is given by intersecting the plane of normal $\tilde{\boldsymbol{x}}$, centered in $\tilde{\boldsymbol{x}}$, with the cone.

In practice, to find such a section, we find the vertices of the 6D polytope defined by:

$$\begin{pmatrix} A_{\mathcal{M}} \\ -\tilde{\boldsymbol{x}}^T \end{pmatrix} \boldsymbol{x} \geq \begin{pmatrix} 0 \\ -||\tilde{\boldsymbol{x}}||^2 \end{pmatrix} \tag{5.20}$$

where the last inequality models $(\boldsymbol{x} - \tilde{\boldsymbol{x}}) \cdot \tilde{\boldsymbol{x}} \leq 0$, i.e. $\boldsymbol{x}$ should be in the half plane of normal $\tilde{\boldsymbol{x}}$ containing $\mathbf{0}$. SYSTEM (5.20) is solved numerically to enumerate the vertices of the polyhedron using the python package `pypoman`. A cartoon illustration in $\mathbb{R}^3$ of this process is given on FIGURE 5.8.

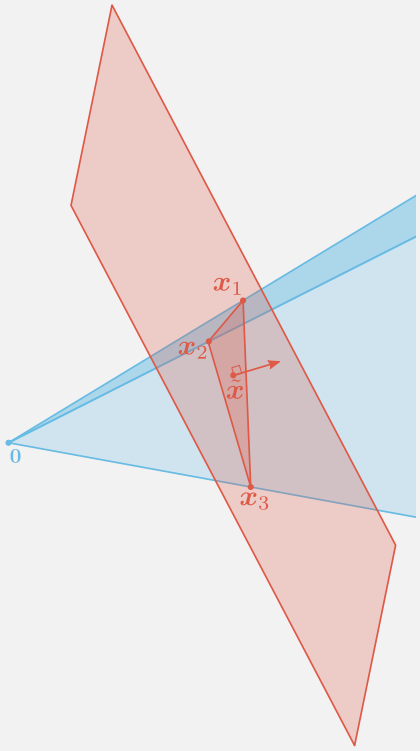


Figure 5.8| Given a solution $\tilde{\boldsymbol{x}}$, the conical section $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$ is found by intersecting the cone in blue with the plane going through $\tilde{\boldsymbol{x}}$ and of normal $\frac{\tilde{\boldsymbol{x}}}{||\tilde{\boldsymbol{x}}||}$.

By construction, the list of the 6D vertices of the polytope contains the vertex $\mathbf{0}$. Let $\{\mathbf{0}, \boldsymbol{x}_1, ..., \boldsymbol{x}_k\}$ denotes these vertices. The $\boldsymbol{x}_i, i \in [\![1, k]\!]$, are on the plane of normal $\tilde{\boldsymbol{x}}$ containing this point, i.e $(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}) \cdot \tilde{\boldsymbol{x}} = 0$. As such, a conical section of the cone is given by the convex set of $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_k\}$. Now let $\boldsymbol{x} = \begin{pmatrix} \boldsymbol{t} \\ \boldsymbol{\Omega} \end{pmatrix}$ be a point in such section: $\exists (\alpha_i)_{i \in [\![1,k]\!]} \geq 0$, $\sum_{i=1}^k \alpha_i = 1$, $\boldsymbol{x} = \sum_{i=1}^k \alpha_i \boldsymbol{x}_i$. To retrieve the unit dual quaternion $\hat{q} \in \mathcal{U}(\mathbb{DH})$ encoding the same motion as the one encoded in $\boldsymbol{x}$, let us introduce the function

$$
\begin{array}{rccc}
f & : & \mathbb{R}^6 & \longrightarrow & \mathcal{U}(\mathbb{DH}) \\
 & & \boldsymbol{x} = \begin{pmatrix} \boldsymbol{t} \\ \boldsymbol{\Omega} \end{pmatrix} & \mapsto & \hat{q} = \left[\cos\frac{\hat{\theta}}{2}, \hat{\boldsymbol{u}} \sin\frac{\hat{\theta}}{2}\right]
\end{array}
\quad \text{with} \quad
\begin{cases} \hat{\theta} &= \theta_0 + \epsilon\theta_\epsilon \\ \hat{\boldsymbol{u}} &= \boldsymbol{u}_0 + \epsilon\boldsymbol{u}_\epsilon \end{cases}
\quad \text{and} \quad
\begin{cases} \theta_0 &= ||\boldsymbol{\Omega}|| \\ \boldsymbol{u}_0 &= \frac{\boldsymbol{\Omega}}{||\boldsymbol{\Omega}||} \\ \theta_\epsilon &= \boldsymbol{t} \cdot \boldsymbol{u}_0 \\ \boldsymbol{u}_\epsilon &= \frac{1}{\theta_0}(\boldsymbol{t} - \theta_\epsilon \boldsymbol{u}_0) \end{cases}
$$

We can now justify that the convex set of $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_k\}$ encodes all the motions that may be obeyed by $\mathcal{M}$. Indeed, suppose $\boldsymbol{x}$ to be a convex combination of the $(\boldsymbol{x}_i)_{1 \leq i \leq k}$. Then, as said above, for any $\lambda \geq 0$, $\mathcal{M}$ obeys the motion encoded by $\lambda\boldsymbol{x}$. However one notices that for any point $\boldsymbol{p} \in \mathbb{R}^3$, $\boldsymbol{m}(\boldsymbol{p}, f(\lambda\boldsymbol{x})) = \lambda\boldsymbol{m}(\boldsymbol{p}, f(\boldsymbol{x}))$, that is to say that the initial velocity of point $\boldsymbol{p}$ subjected to the motion $\lambda\boldsymbol{x}$ is scaled by a factor λ compared to the motion encoded in $\boldsymbol{x}$, but its orientation does not change. Since we are only considering infinitesimal motions, the norm of the tangential velocity is of no importance, we only care for its direction. From that point of view, the motions encoded by $\boldsymbol{x}$ and $\lambda\boldsymbol{x}$ are the same. For that reason we may scale in sync the values of θ_0 and θ_ϵ and suppose them to be on the unit circle: $\theta_0 \leftarrow \frac{\theta_0}{\sqrt{\theta_0^2 + \theta_\epsilon^2}}$ and $\theta_\epsilon \leftarrow \frac{\theta_\epsilon}{\sqrt{\theta_0^2 + \theta_\epsilon^2}}$. As such, we may introduce a parameter θ: $\exists \theta \in \mathcal{S}^1$, $\begin{cases} \theta_0 = \cos\theta \\ \theta_\epsilon = \sin\theta \end{cases}$ which reduces the number of parameter.

We can finally conclude:

$$\hat{\mathcal{C}}_{\mathcal{M}} = \left\{ \hat{q} \in \mathcal{U}(\mathbb{DH}),\ \left\{ \begin{array}{l} \hat{q} = f(\boldsymbol{x}) \\ \boldsymbol{x} = \sum_{i=1}^{k} \alpha_i \boldsymbol{x}_i \\ \sum_{i=1}^{k} \alpha_i = 1 \\ \alpha_i \geq 0, \forall i \in [\![1, k]\!] \end{array} \right. \right\} \tag{5.21}$$

EQUATION (5.21) tells us that given the generating rays $\boldsymbol{x}_i$ of the cone solution to SYSTEM (5.20), a solution $\boldsymbol{x}$ is obtained by convex combination of the $\boldsymbol{x}_i$. This solution is mapped, through function f, to a unit dual quaternion $\hat{q} = f(\boldsymbol{x})$ which can be obeyed by the mesh $\mathcal{M}$. The cone of freedom $\hat{\mathcal{C}}_{\mathcal{M}}$ of mesh $\mathcal{M}$ is the set of all such unit dual quaternions, images by f of all possible convex combinations of the generating rays of the cone encoded in SYSTEM (5.20).

5.3 NDBG

As understood in SECTION 3.1.3, the NDBG of an assembly is based on the strong-connectedness of several base DBGs. The principle to compute a DBG in 3D is the same as in 2D: for a given unit dual quaternion $\hat{q}$ and an assembly A, initialise an empty graph $G(\hat{q}, A)$ and loop over each pair of parts in contact. Given two polyhedral parts in contact P_i and P_j, represented by closed triangular meshes, denote by $\mathbb{F}$ the set of faces shared by both parts, and for $f \in \mathbb{F}$, let $\boldsymbol{n}_f$ be the normal of face f pointing from P_j to P_i. For each $f = \{\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{v}_k\} \in \mathbb{F}$ and for each $\boldsymbol{v} \in f$ there are two possibilities:

- Either $\boldsymbol{m}_v \cdot \boldsymbol{n}_f \geq 0$, in which case P_j is blocked by P_i because of this vertex but P_i is locally free to move: add edge $e_{j \to i}$ in the graph.
- Or $\boldsymbol{m}_v \cdot \boldsymbol{n}_f \leq 0$, in which case P_i is blocked by P_j because of this vertex but P_j is locally free to move: add edge $e_{i \to j}$ in the graph.

This process is illustrated on FIGURE 5.9: a 2-parts assembly $A = \{P_0, P_1\}$ is studied, with P_0 the lower part and P_1 the upper one. Two faces f_A and f_B are highlighted in red and blue respectively. Assume that we build a DBG $G(\hat{q}, A)$ such that the instantaneous direction of motion of a vertex $\boldsymbol{v}$ adjacent to both faces is $\boldsymbol{m}(\boldsymbol{v}, \hat{q})$ such as shown in green on the figure. The directions of the vectors $\boldsymbol{n}_{f_A}, \boldsymbol{n}_{f_B}, \boldsymbol{m}(\boldsymbol{v}, \hat{q})$ are exemplified on the sphere. One sees that $\boldsymbol{m}(\boldsymbol{v}, \hat{q}) \cdot \boldsymbol{n}_{f_A} \leq 0$ and $\boldsymbol{m}(\boldsymbol{v}, \hat{q}) \cdot \boldsymbol{n}_{f_B} \geq 0$ as $\boldsymbol{m}(\boldsymbol{v}, \hat{q})$ belongs to the blue hemisphere oriented by $\boldsymbol{n}_{f_B}$ but not the red one oriented by $\boldsymbol{n}_{f_A}$. As such face f_A prevents the upper part P_1 from obeying $\hat{q}$, hence the edge $e_{1 \to 0}$ is added to the DBG $G(\hat{q}, A)$. Conversely, face f_B prevents the lower part P_0 from obeying $\hat{q}$ and the edge $e_{0 \to 1}$ is added to $G(\hat{q}, A)$.

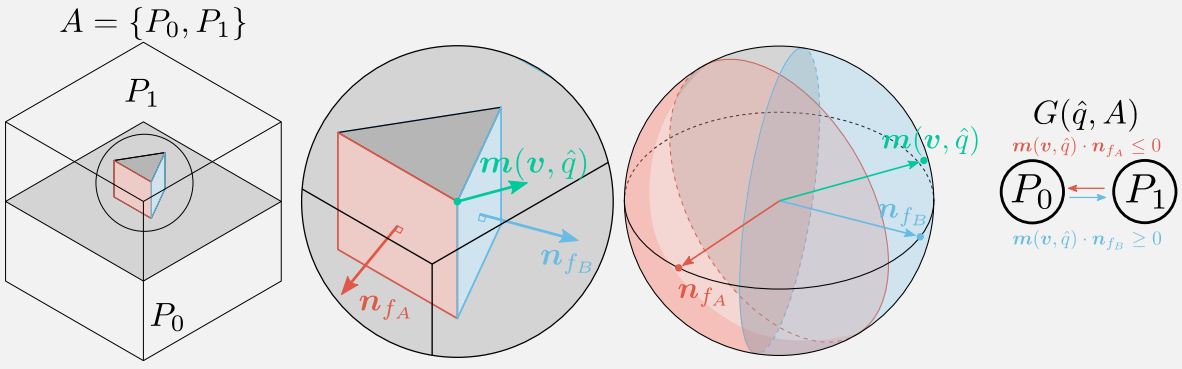


Figure 5.9| Example of a calculation of a DBG.

5.4 CREATING A 2-PARTS ASSEMBLY

Given a design domain, which throughout this section will be a cube of unit length centred at the origin of $\mathbb{R}^3$, this section aims at understanding how it can be partitioned into two polyhedral parts, P_0 and P_1, with P_1 obeying a user-given unit dual quaternion $\hat{q}$.

5.4.1 INPUT

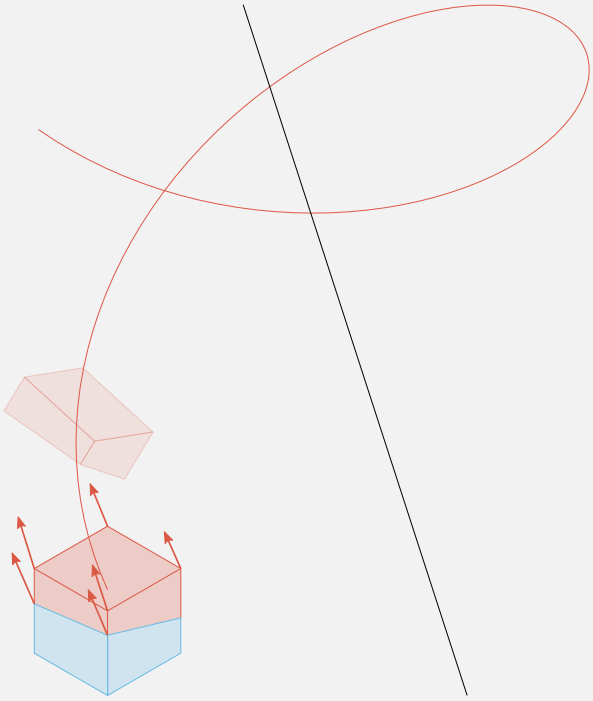


Figure 5.10| The input to generate a 2-parts assembly.

Even though, or perhaps because, screw motions encode all rigid body motions, they are not especially easy to visualise. To help the user specify a unit dual quaternion $\hat{q}$, he/she may wish to partition the design domain in two pseudo-parts of approximate shape: on FIGURE 5.10 the design domain is cut in two by a plane. Pseudo-P_0 is in blue and pseudo-P_1 in red. Then, the user may either play with the parameters $\boldsymbol{u}_0, \boldsymbol{u}_\epsilon$ and θ (with $\theta_0 = \cos\theta$ and $\theta_\epsilon = \sin\theta$) and see where it leads pseudo-P_1 after a motion of finite magnitude along $\hat{q}$, or it can move pseudo-P_1 in space in an arbitrary position and location and automatically compute $\hat{q}$ afterwards, using the equations outlined in SECTION 5.1.8.6. On FIGURE 5.10 the screw axis is the black line, the trajectory followed by P_1 is depicted with the red curve, the instantaneous directions of motion of the visible vertices of P_1 are shown using red arrows, and the resting position of P_1 used to compute $\hat{q}$ is the body in faint red.

Once satisfied with $\hat{q}$ and with the approximate shapes of the parts, a planar triangular mesh is generated in the design domain on the separating plane. At this step there are two options:

- The user manually deforms the mesh by vertex painting: some vertices are moved (up or down) in a direction orthogonal to the plane. The user can also deform the mesh by any other mean.
- The mesh may be deformed (in an orthogonal direction to the plane) using “organic” noise applied on the vertices. Purely random noise (each vertex is moved randomly independently from the others) yields dull and uninteresting meshes, whereas correlating the out-of-plane motion among neighbouring vertices gives a qualitatively much more pleasant mesh. The current implementation of this algorithm uses Perlin noise.

Once the input mesh is provided, the vertices are moved according to the following algorithm. The signed distance d between each vertex and the separating plane is divided by a factor $n \in \mathbb{N}^*$; Each vertex $\boldsymbol{v}$ is moved back onto the plane and its instantaneous direction of motion $\boldsymbol{m}_v$ is calculated. The vertex is then moved in the direction of $\boldsymbol{m}_v$ by a distance $\frac{d}{n}$. This iteration is repeated n times until each vertex has moved by a distance d. On FIGURE 5.11, the original position of the mesh vertices are shown in blue, along the black line. The mesh and the motions are projected on the plane orthogonal to $\boldsymbol{u}_0$, the direction of the screw axis, the latter being represented by the red point. By random motion or vertex painting, the mesh vertices were moved horizontally up to the positions in large grey dots; the signed distances d are figured by the dotted straight lines. The original vertices are then iteratively moved by $\frac{d}{n}$ in the direction $\boldsymbol{m}_v$ (here $n = 4$), and at each new position, represented with small grey dots, the process is repeated. While it does not necessarily help to achieve a faster convergence with the GPA for a randomly deformed mesh (it does not necessarily hurt either), it does better condition the mesh for a more regular vertex painted one. FIGURE 5.12 shows, for a given $\hat{q}$, on the top row input meshes before the aforementioned conditioning, and below the same meshes after. The leftmost mesh is initialised with Perlin noise, the middle one with vertex painting, and the right one is a custom user-given mesh.

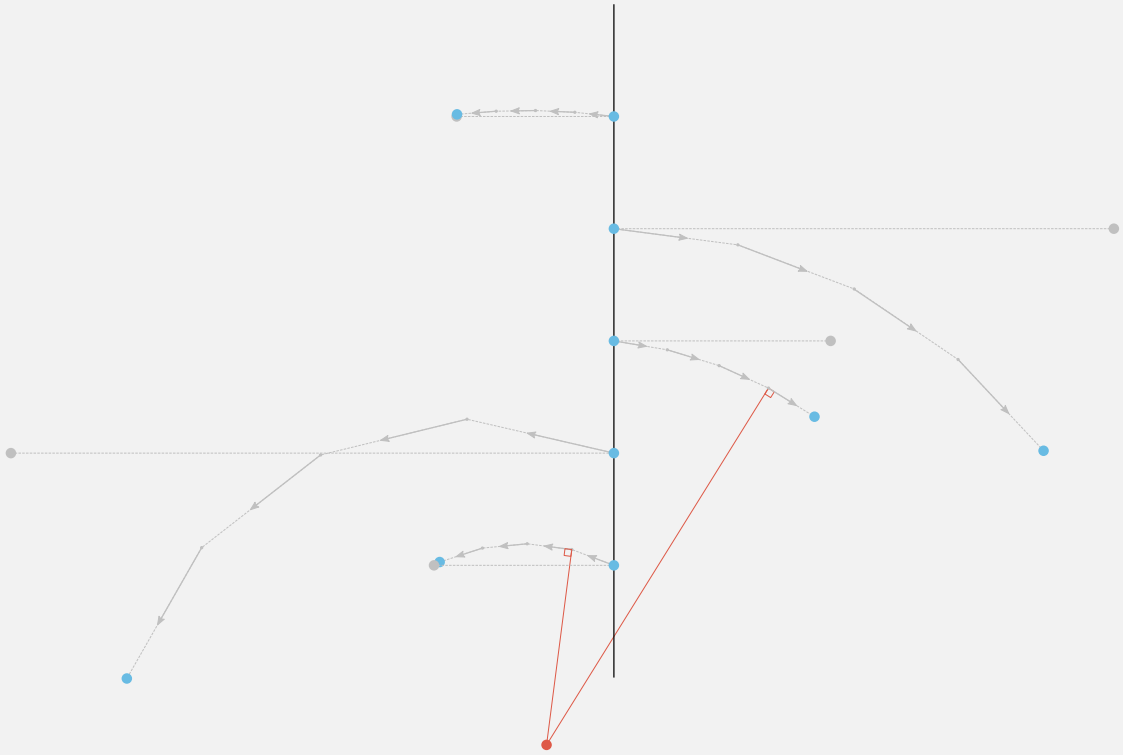


Figure 5.11| Process to better condition the mesh.

5.4.2 GUIDED PROJECTION ALGORITHM

Given a unit dual quaternion $\hat{q}$ used to condition a mesh $\mathcal{M} = (V, E, F)$, the GPA optimisation (see SECTION 4.1.2) is executed: the goals are:

- Every face $f \in F$ must obey $\hat{q}$.
- To prevent too small details, the dihedral angle θ between two adjacent faces must be greater than a threshold θ_{lim}.
- The boundary edges of $\mathcal{M}$ must slide on the design domain faces.
- As in 2D, a snap constraint must be implemented.

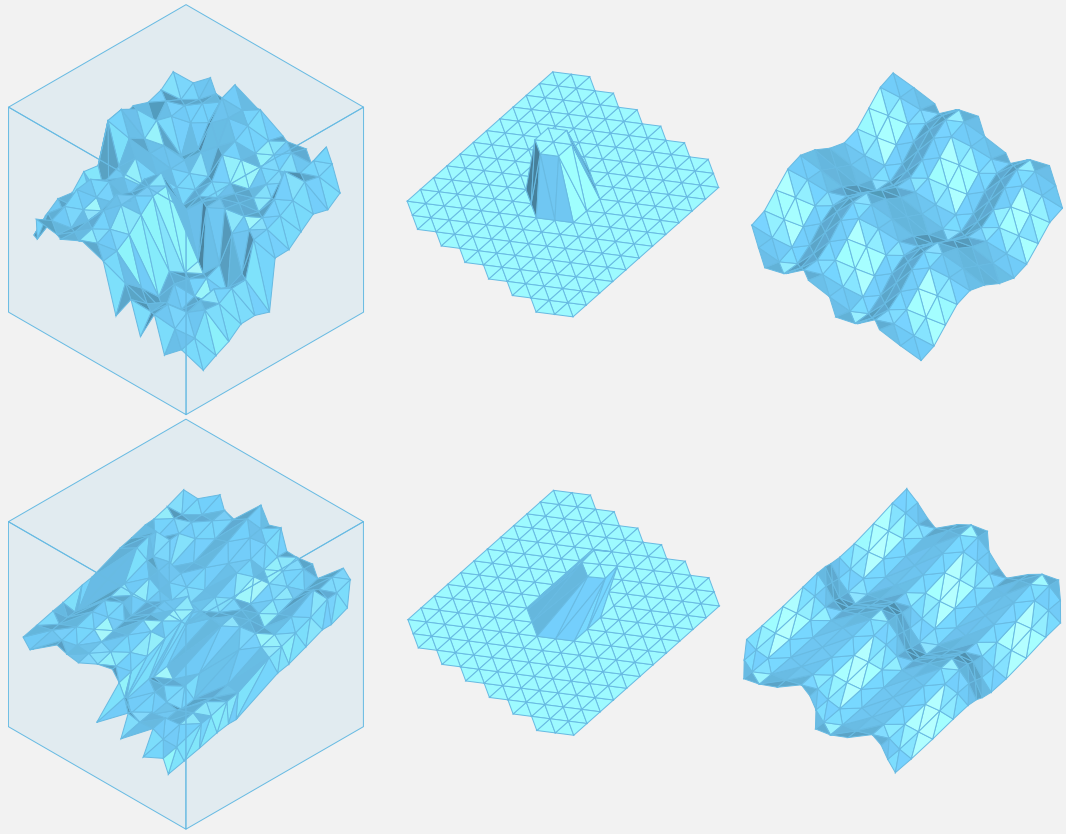


Figure 5.12| Top row: before conditioning the meshes. Bottom row: after.

Vector $\boldsymbol{X}$ is initialised with the necessary number of entries, the first $3|V|$ of them being the coordinates of each vertex $\boldsymbol{v} \in V$.

5.4.2.1 Every face must obey $\hat{q}$

We want to impose the constraint that $\forall \boldsymbol{v} \in V(f)\ \boldsymbol{m}_v \cdot \boldsymbol{n}_f \geq 0$. We need first to calculate the normal vector of the face $\boldsymbol{n}_f$.

Calculation of the normal vector

By definition, $\boldsymbol{n}_f$ is orthogonal to face f, is of unit length and must be consistently defined across all faces, as to always be on the same side; this is implemented as $\boldsymbol{n}_f$ is positively proportional to the non-unit normal vector $\boldsymbol{n}_f^{\neq 1} = (\boldsymbol{v}_j - \boldsymbol{v}_i) \times (\boldsymbol{v}_k - \boldsymbol{v}_i)$.

Calculation of the non-unit normal vector: Let o and f be the indices such that $\boldsymbol{X}[o+3f], \boldsymbol{X}[o+3f+1]$ and $\boldsymbol{X}[o+3f+2]$ respectively map to $n_{fx}^{\neq 1}, n_{fy}^{\neq 1}$ and $n_{fz}^{\neq 1}$, the coordinates of the vector $\boldsymbol{n}_f^{\neq 1}$. By permuting the indices x, y, z and denoting (x_p, y_p, z_p) the coordinates of vertex $\boldsymbol{v}_p \in \{\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{v}_k\}$ one gets that:

$$\begin{cases} n_{fx}^{\neq 1} = \frac{1}{2}\left(y_i(z_j - z_k) + z_i(y_k - y_i) + y_j(z_k - z_i) + z_j(y_i - y_k) + y_k(z_i - z_j) + z_k(y_j - y_i)\right) \\ \dots \end{cases}$$

Let i be the index such that $\boldsymbol{X}[3i], \boldsymbol{X}[3i+1], \boldsymbol{X}[3i+2]$ map to the coordinates x_i, y_i and z_i, indices j, k are similarly defined. For $q \in 0, 1, 2$ let:

$$\boldsymbol{H}_{3f+q} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} \leftarrow 3i + ((1+q) \mod 3) \\ \leftarrow 3i + ((2+q) \mod 3) \\ \leftarrow 3j + ((1+q) \mod 3) \\ \leftarrow 3j + ((2+q) \mod 3) \\ \leftarrow 3k + ((1+q) \mod 3) \\ \leftarrow 3k + ((2+q) \mod 3) \end{matrix} \quad \begin{cases} \boldsymbol{b}_{3f+q} = \begin{pmatrix} -1 \end{pmatrix} \leftarrow o + 3i + q \\ c_{3f+q} = 0 \end{cases}$$

One has

$$(\boldsymbol{v}_j - \boldsymbol{v}_i) \times (\boldsymbol{v}_k - \boldsymbol{v}_i) - \boldsymbol{n}_f^{\neq 1} = \boldsymbol{0} \iff \forall q \in \{0,1,2\}\ \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_{3f+q} \boldsymbol{X} + \boldsymbol{b}_{3f+q}^T \boldsymbol{X} + c_{3f+q} = 0$$

Calculation of a consistent normal vector: The normal $\boldsymbol{n}_f$ must be positively collinear to $\boldsymbol{n}_f^{\neq 1}$: $\exists \alpha_f > 0,\ \boldsymbol{n}_f = \alpha_f \boldsymbol{n}_f^{\neq 1}$. $\boldsymbol{X}$ is initialised with $\alpha_f = \frac{1}{||\boldsymbol{n}_f^{\neq 1}||}$. Let

- on, f be the indices such that $\boldsymbol{X}[on + 3f], \boldsymbol{X}[on + 3f + 1], \boldsymbol{X}[on + 3f + 2]$ map to the coordinates n_{fx}, n_{fy} and n_{fz} of $\boldsymbol{n}_f$.
- $o_{\neq 1}$ be defined similarly for $\boldsymbol{n}_f^{\neq 1}$.
- oa be the index such that $\boldsymbol{X}[oa + f]$ maps to α_f.

$$\boldsymbol{H}_{3f+q} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{matrix} \leftarrow o_{\neq 1} + 3f + q \\ \leftarrow oa + f \end{matrix} \qquad \boldsymbol{b}_{3f+q} = \begin{pmatrix} -1 \end{pmatrix} \leftarrow on + 3i + q \qquad c_{3f+q} = 0$$

Where $q \in \{0,1,2\}$. One has

$$\alpha_f \boldsymbol{n}_f^{\neq 1} - \boldsymbol{n}_f = \boldsymbol{0} \iff \forall q \in \{0,1,2\}\ \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_{3f+q} + \boldsymbol{X} + \boldsymbol{b}_{3f+q}^T \boldsymbol{X} + c_{3f+q} = 0$$

Constraint requiring $\boldsymbol{n}_f$ to be of unit length: We implement $||\boldsymbol{n}_f||^2 - 1 = 0$. Let o, f be the indices such that $\boldsymbol{X}[o + 3f], \boldsymbol{X}[o + 3f + 1], \boldsymbol{X}[o + 3f + 2]$ map to the coordinates n_{fx}, n_{fy} and n_{fz} of $\boldsymbol{n}_f$.

$$\boldsymbol{H}_f = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} \begin{matrix} \leftarrow o + 3f \\ \leftarrow o + 3f + 1 \\ \leftarrow o + 3f + 2 \end{matrix} \qquad \boldsymbol{b}_f = \boldsymbol{0} \qquad c_f = -1$$

One has

$$||\boldsymbol{n}_f||^2 - 1 = 0 \iff \forall q \in \{0,1,2\}\ \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_{3f+q} \boldsymbol{X} + \boldsymbol{b}_{3f+q}^T \boldsymbol{X} + c_{3f+q} = 0$$

Constraint requiring f to obey $\hat{q}$

For each vertex $\boldsymbol{v} \in \{\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{v}_k\}$ of face f, indexed by v such that $\boldsymbol{X}[3v]$, $\boldsymbol{X}[3v+1]$ and $\boldsymbol{X}[3v+2]$ map to the coordinates of $\boldsymbol{v}$, we require that $\boldsymbol{m}_v \cdot \boldsymbol{n}_f \geq 0$, which means that for some η_v, $\boldsymbol{m}_v \cdot \boldsymbol{n}_f - \eta_v^2 = 0$. Recall that $\boldsymbol{m}_v = \theta_0(\boldsymbol{u}_0 \times \boldsymbol{p}) + \theta_\epsilon \boldsymbol{u}_0 + \theta_0 \boldsymbol{u}_\epsilon$. Let u_{0x},..., $u_{\epsilon z}$ refer to the coordinates of $\boldsymbol{u}_0$ and $\boldsymbol{u}_\epsilon$. Finally let oe be the index such that $\boldsymbol{X}[oe + v]$ maps to η_v and on such that $\boldsymbol{X}[on + 3f], \boldsymbol{X}[on + 3f + 1], \boldsymbol{X}[on + 3f + 2]$ map to the coordinates n_{fx}, n_{fy} and n_{fz} of $\boldsymbol{n}_f$.

$$\boldsymbol{H}_{v,f} = \begin{pmatrix} 0 & 0 & 0 & 0 & \theta_0 u_{0z} & -\theta_0 u_{0y} & 0 \\ 0 & 0 & 0 & -\theta_0 u_{0z} & 0 & \theta_0 u_{0x} & 0 \\ 0 & 0 & 0 & \theta_0 u_{0y} & -\theta_0 u_{0x} & 0 & 0 \\ 0 & -\theta_0 u_{0z} & \theta_0 u_{0y} & 0 & 0 & 0 & 0 \\ \theta_0 u_{0z} & 0 & -\theta_0 u_{0x} & 0 & 0 & 0 & 0 \\ -\theta_0 u_{0y} & \theta_0 u_{0x} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 \end{pmatrix} \begin{matrix} \leftarrow 3v \\ \leftarrow 3v+1 \\ \leftarrow 3v+2 \\ \leftarrow on+3f \\ \leftarrow on+3f+1 \\ \leftarrow on+3f+2 \\ \leftarrow oe+v \end{matrix}$$

$$\boldsymbol{b}_{v,f} = \begin{pmatrix} \theta_0 u_{\epsilon x} + \theta_\epsilon u_{0x} \\ \theta_0 u_{\epsilon y} + \theta_\epsilon u_{0y} \\ \theta_0 u_{\epsilon z} + \theta_\epsilon u_{0z} \end{pmatrix} \begin{matrix} \leftarrow on+3f \\ \leftarrow on+3f+1 \\ \leftarrow on+3f+2 \end{matrix} \qquad c_{v,f} = 0$$

One has

$$\boldsymbol{m}_v \cdot \boldsymbol{n}_f \geq 0 \Longleftrightarrow \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_{v,f} \boldsymbol{X} + \boldsymbol{b}_{v,f}^T \boldsymbol{X} + c_{v,f} = 0$$

5.4.2.2 To prevent too small details, the dihedral angle θ between two adjacent faces must be greater than a threshold θ_{lim}

We make use here of the fact that the normal vector $\boldsymbol{n}_f$ of a face $f \in F$ is of unit length: $||\boldsymbol{n}_f|| = 1$. We require that for two adjacent faces indexed by p, q of normal $\boldsymbol{n}_p, \boldsymbol{n}_q$:

$$\boldsymbol{n}_p \cdot \boldsymbol{n}_q + \cos\theta_{lim} - \zeta_{p,q}^2 = 0$$

for some $\zeta_{p,q} \in \mathbb{R}$. Let o be the index such that $\boldsymbol{X}[o+3p], \ldots, \boldsymbol{X}[o+3q+2]$ map to $\boldsymbol{n}_{px}, \ldots, \boldsymbol{n}_{qz}$, and opq such that $\boldsymbol{X}[opq]$ maps to $\zeta_{p,q}$. Thus:

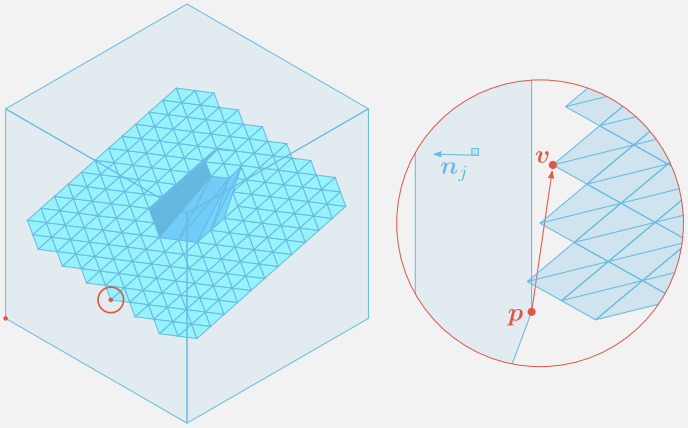
$$\boldsymbol{H}_{p,q} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 \end{pmatrix} \begin{matrix} \leftarrow o+3p \\ \leftarrow o+3p+1 \\ \leftarrow o+3p+2 \\ \leftarrow o+3q \\ \leftarrow o+3q+1 \\ \leftarrow o+3q+2 \\ \leftarrow opq \end{matrix} \qquad \boldsymbol{b}_{p,q} = \boldsymbol{0} \qquad c_{p,q} = \cos(\theta_{lim})$$

One has

$$\boldsymbol{n}_p \cdot \boldsymbol{n}_q + \cos\theta_{lim} \geq 0 \Longleftrightarrow \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_{p,q} \boldsymbol{X} + \boldsymbol{b}_{p,q}^T \boldsymbol{X} + c_{p,q} = 0$$

All the above constraints are topological: to express them, we only need adjacency information between faces and vertices, not their actual location in space.

5.4.2.3 The boundary edges of $\mathcal{M}$ must slide on the design domain faces.



Let $\boldsymbol{v}$ be a boundary vertex and let $\boldsymbol{p}$ be a vertex of the face of the design domain the closest to $\boldsymbol{v}$. Let $\boldsymbol{n}_j$ be the unit normal vector of face j. We require that $\boldsymbol{n}_j \cdot (\boldsymbol{v} - \boldsymbol{p}) = 0$, as illustrated on the inset. To prevent the algorithm from optimising towards $\boldsymbol{v} = \boldsymbol{p}$, point $\boldsymbol{p}$ is dynamically chosen among the vertices of face j so that it is not the closest to vertex $\boldsymbol{v}$. This is a geometrical constraint as we need the location of $\boldsymbol{v}$ in space to select face j as well as point $\boldsymbol{p}$. Let v be the index such that $\boldsymbol{X}[3v], \boldsymbol{X}[3v+1]$ and $\boldsymbol{X}[3v+2]$ map to the coordinates of vertex $\boldsymbol{v}$, and denote by n_{jx}, n_{jy} and n_{jz} the coordinates of normal $\boldsymbol{n}_j$.

$$\boldsymbol{H}_v = \boldsymbol{0} \qquad \boldsymbol{b}_v = \begin{pmatrix} n_{jx} \\ n_{jy} \\ n_{jz} \end{pmatrix} \begin{matrix} \leftarrow 3v \\ \leftarrow 3v+1 \\ \leftarrow 3v+2 \end{matrix} \qquad c_v = -\boldsymbol{p} \cdot \boldsymbol{n}_j$$

One has

$$\boldsymbol{n}_j \cdot (\boldsymbol{v} - \boldsymbol{p}) = 0 \iff \frac{1}{2}\boldsymbol{X}^T \boldsymbol{H}_v \boldsymbol{X} + \boldsymbol{b}_v^T \boldsymbol{X} + c_v = 0$$

5.4.2.4 Snap constraint

A face f adjacent to a vertex $\boldsymbol{v}$ snaps (we will refer to f as a snap face but also to $\boldsymbol{v}$ as a snap vertex) if and only if, when obeying $\hat{q}$ for an infinitesimal motion, $\boldsymbol{v}$ slides on the plane defined by f:

$$\boldsymbol{m}_v \cdot \boldsymbol{n}_f = 0$$

We will discuss in SECTION 5.4.3 the manner in which f and $\boldsymbol{v}$ are chosen. To compute the constraint $(\boldsymbol{H}_{v,f}, \boldsymbol{b}_{v,f}, c_{v,f})$, please refer to the detail of "Constraint requiring f to obey $\hat{q}$" and replace $\boldsymbol{H}_{v,f}[oe+v, oe+v] = -2$ by $\boldsymbol{H}_{v,f}[oe+v, oe+v] = 0$.

5.4.2.5 Other constraints

Several other constraints may be implemented, but are not described in this manuscript as they are not the most important ones, and anyway can easily be derived once one has understood the logic of the expression of constraints in the GPA. We can think of constraining the boundary vertices to follow a goal curve, drawn on the design domain mesh; ensuring that the area of each face is greater than some threshold (to avoid small triangles); or, as done in the version at the time of this writing, implementing circle packing ([86]) to well-condition the mesh.

5.4.3 CHOOSING WHERE TO SNAP

5.4.3.1 Preliminaries

This section explains where the so-called snap face-vertex pairs must be chosen. We saw in 2D that to reduce the close of translational freedom $\mathcal{C}_t$ to the user prescribed cone, exactly 2 snaps segments were required

(SECTION 3.1.1). In rotation there is no minimal number of snap segments, we understood that it depends on the geometry of the separating polyline.
In 3D, if we are looking at pure translations only, then we state that the minimal number of snap faces is three: as illustrated on FIGURE 5.13 the locus of all directions of translation in 3D is the unit sphere $\mathcal{S}^2$. Each snap face of an assembly (highlighted in red) defines a hemisphere of valid directions of translation, encoded by $\boldsymbol{n}_f \cdot \boldsymbol{x}_t \geq 0$ (where $\boldsymbol{x}_t \in \mathcal{S}^2$ is a direction of translation in 3D), and the cone of translational freedom is found by intersecting the hemispheres between them. One needs at least three snap faces to reduce the cone of freedom to two antipodal points, shown in blue and red on the figure, and one non-snap face to cull one of the two directions, leaving the cone reduced to a single direction of translation (the red point if we are considering the upper part, the blue one if we are considering the lower part).

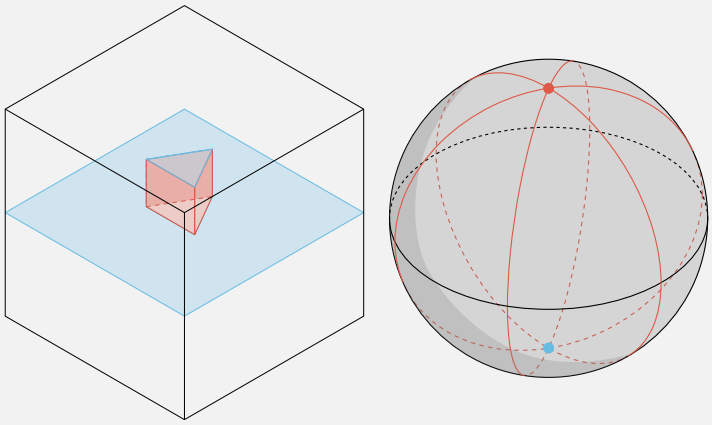


Figure 5.13| Three snap faces are required in 3D to reduce the cone of translational freedom to the user-prescribed cone.

5.4.3.2 Choosing the first snap face-vertex pair.

This small study gives us a lower bound of the number of snap faces: in the general case where $\hat{q}$ does not encode a pure translation, the minimum number of snap faces is thus three. Therefore, let $n \geq 3$ be the number of snap faces prescribed by the user. This section explains how the snap faces and snap vertices are chosen.

Let $G(\mathcal{M})$ be the Gauss map of mesh $\mathcal{M} = (V, E, F)$: it consists of the constellation of normal vectors $(\boldsymbol{n}_f)_{f \in F}$ seen as points on the unit sphere $\mathcal{S}^2 \subset \mathbb{R}^3$. The mesh being triangular, to each face $f = \{\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{v}_k\}$ are associated the three instantaneous directions of motion $(\boldsymbol{m}_{v_i}, \boldsymbol{m}_{v_j}, \boldsymbol{m}_{v_k})$. As shown on FIGURE 5.14 they can be visualised on the Gauss map as three vectors rooted in each $\boldsymbol{n}_f$.

Having f to snap means that there is a $\boldsymbol{v} \in \{\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{v}_k\}$ whose instantaneous direction of motion is such that $\boldsymbol{m}_v \cdot \boldsymbol{n}_f = 0$: it is contained in the tangent plane at $\boldsymbol{n}_f$ of the sphere. Intuitively it means that when disassembling the part, vertex $\boldsymbol{v}$ slides on the other part. This gives an immediate choice for the first snap face/vertex: of all faces in F, we select the one with a $\boldsymbol{m}_v$ the closest to being in the tangent plane:

$$\text{Choose } f \in F, \boldsymbol{v} \in V(f) \text{ such that } |\boldsymbol{m}_v \cdot \boldsymbol{n}_f| \to \min$$

To select for the next $n-1$ snap faces, we cannot simply go and select the $n-1$ face-vertex pair with the smallest $|\boldsymbol{m}_v \cdot \boldsymbol{n}_f|$. To understand why, let us go back to the pure 3D translation case.

5.4.3.3 Where to snap in the pure translation case

Assume we want to partition the design domain into two parts such that P_1 obeys a vertical upwards (pure) translation. We cannot simply choose three random snap faces to reduce the cone of freedom to the north

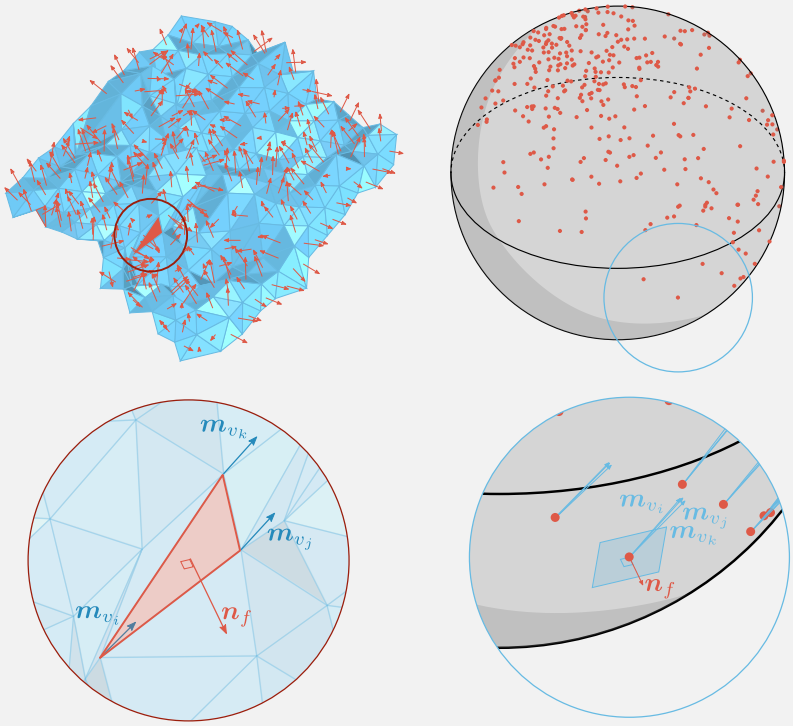


Figure 5.14| The normal vectors of $\mathcal{M}$ are visualised on $G(\mathcal{M})$; each normal is associated with a triplet of instantaneous directions of motion.

pole of the sphere as, in general, the actual cone of freedom is strictly larger. On FIGURE 5.15 the tenon-like feature is made of four non-horizontal faces. Three of them, in red, have snapped as they are vertical, while the last one, in blue, has not. On the right, the boundaries of the hemispheres defined by each face are of the same colour as their corresponding face. The cone of translational freedom is highlighted in red: it is not reduced to the prescribed north pole. For the actual cone to be reduced to the north pole, the third snap

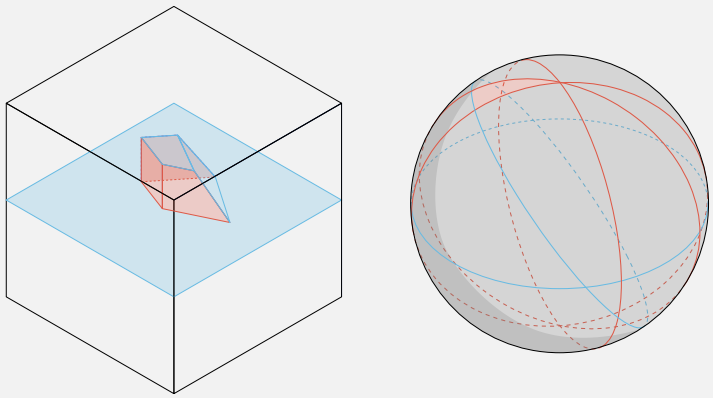


Figure 5.15| Three random snap faces do not reduce the cone of freedom to the prescribed direction of translation.

face f_C must depend of the other two snap faces f_A and f_B: the normal $\boldsymbol{n}_{f_c}$ must be chosen in the circular arc bounded by $-\boldsymbol{n}_{f_A}$ and $-\boldsymbol{n}_{f_B}$, see FIGURE 5.16. Indeed, in such a case, only the north pole $\boldsymbol{x}_t = \boldsymbol{e}_z$ and the south pole $\boldsymbol{x}_t = -\boldsymbol{e}_z$ are solutions to

$$\begin{cases} \boldsymbol{n}_{f_A} \cdot \boldsymbol{x}_t = 0 \\ \boldsymbol{n}_{f_B} \cdot \boldsymbol{x}_t = 0 \\ \boldsymbol{n}_{f_C} \cdot \boldsymbol{x}_t = 0 \end{cases}$$

where the direction of translation $\boldsymbol{x}_t$ is the instantaneous direction of motion $\boldsymbol{m}_v$ for any vertex $\boldsymbol{v}$ belonging to the three faces. The south pole $-\boldsymbol{e}_z$ is culled from the cone by any other non-snap face. We remark

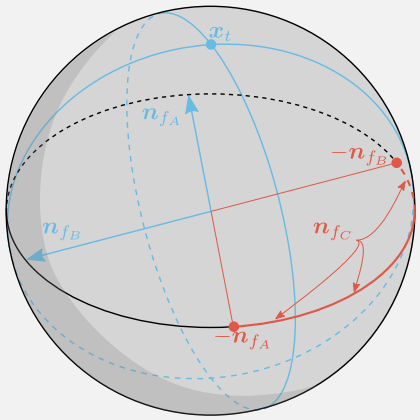


Figure 5.16| For the cone of freedom to be reduced to $\boldsymbol{x}_t$, $\boldsymbol{n}_{f_C}$ must be chosen on the highlighted circular arc.

that evenly distributing (each at an angle $\frac{2\pi}{3}$ from the others) the normal vectors $\boldsymbol{n}_{f_A}$, $\boldsymbol{n}_{f_b}$ and $\boldsymbol{n}_{f_c}$ on the equator solves the problem. It is the strategy we choose to implement to select the snap face-vertex pairs in the general case.

5.4.3.4 Choosing the next $n-1$ snap face-vertex pairs.

Let $\overline{\boldsymbol{m}} = \frac{1}{|V|}\sum_{v\in V}\boldsymbol{m}_v$ be the average direction of motion of all mesh vertices, and let $\overline{\mathcal{C}}$ be the unit circle of normal $\overline{\boldsymbol{m}}$. We could have calculated $\overline{\boldsymbol{m}}$ in a more sophisticated manner, for instance by weighting each vertex with the areas of its adjacent faces, but as we will understand, the exact value of $\overline{\boldsymbol{m}}$ matters not.

The algorithm to find the next $n-1$ snap face-vertex pairs is as follows. Assume the first snap face-vertex pair has been chosen (see SECTION 5.4.3.2), labelled f_1 and $\boldsymbol{v}_{f_1}$. Let $\boldsymbol{p}_1 \in \overline{\mathcal{C}}$ be the closest point from $\boldsymbol{n}_{f_1}$ on $\overline{\mathcal{C}}$. Starting from $\boldsymbol{p}_1$, points $\boldsymbol{p}_2, \dots, \boldsymbol{p}_n$ are evenly distributed on $\overline{\mathcal{C}}$, each at an angle $\frac{2\pi}{n}$ from the previous. Then, for each $\boldsymbol{p}_i$ ($2 \leq i \leq n$), we find its closest point $\boldsymbol{n}_{f_i} \in G(\mathcal{M})$. The faces $(f_i)_{i\in[\![2,n]\!]}$ so designed are the $n-1$ remaining snap faces of the mesh. To select for the associated snap vertices $(\boldsymbol{v}_{f_i})_{i\in[\![2,n]\!]}$ we simply take among the three vertices defining f_i the $\boldsymbol{v}_{f_i}$ such that $\boldsymbol{m}_{v_{f_i}} \cdot \boldsymbol{n}_{f_i}$ is the closest to 0. This algorithm is illustrated on FIGURE 5.17. The Gauss map $G(\mathcal{M})$ is depicted using semi-transparent black dots. The average direction $\overline{\boldsymbol{m}}$ and circle $\overline{\mathcal{C}}$ are shown in blue and the normal $\boldsymbol{n}_{f_1}$ of the first snap face and the projection $\boldsymbol{p}_1$ are highlighted (it is only by chance that $\boldsymbol{n}_{f_1}$ and $\boldsymbol{p}_1$ are almost coincident on the figure). Starting from $\boldsymbol{p}_1$ the five ($n=6$) other $\boldsymbol{p}_i$, in blue, are regularly placed on $\overline{\mathcal{C}}$. The closest $\boldsymbol{n}_{f_i} \in G(\mathcal{M})$ are shown in red, and the geodesic distances between them are figured by circular blue arcs. On the right, a zoom on a particular $\boldsymbol{n}_{f_i}$ shows that the associated snap vertex, $\boldsymbol{v}_{f_i}$, is the one having the instantaneous direction of motion in darker blue.

The exact value of $\overline{\boldsymbol{m}}$ matters very little: even though, as often in discrete geometry, there are several reasonable ways to define this average direction, they would all define directions close to each other. Circle $\overline{\mathcal{C}}$ would only change by a small amount from one definition to another, and the $(\boldsymbol{n}_{f_i})_{i\in[\![1,n]\!]}$ chosen would perhaps vary, but would still be well spread around the sphere, which is what matters to define the snap face-vertex pairs.

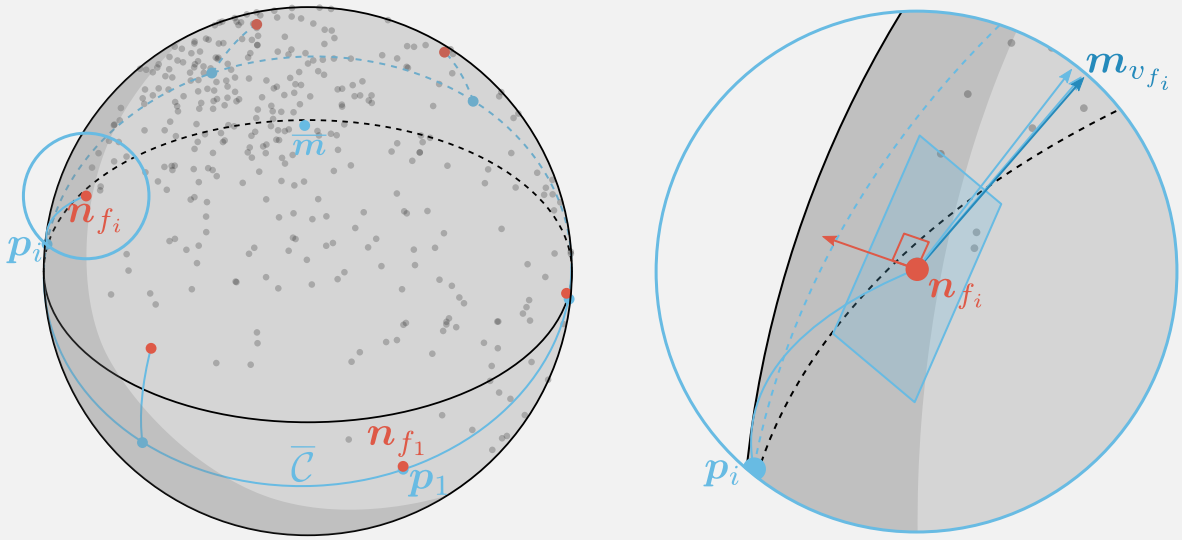


Figure 5.17| Illustration of the algorithm to choose the next $n-1$ snap face-vertex pairs.

5.4.4 SPLITTING THE DESIGN DOMAIN INTO TWO PARTS

5.4.4.1 Creating a water-tight mesh

Before splitting the design domain into two parts, we must be able to tell unequivocally which vertex of the design domain belongs to which part. This is done by stating on which side of the separating mesh a vertex of the design domain is. To that end, we ensure that the separating surface is tightly connected to the design domain: even if all boundary vertices of the separating mesh are constrained to slide on the faces of the design domain, there may still be holes in the vicinity of a design domain edge. Triangle faces (obeying $\hat{q}$) are created until the mesh tightly hangs to the design domain, see FIGURE 5.18 where the optimised separating surface after the GPA is shown on the left the newly created faces are highlighted in red on the right.

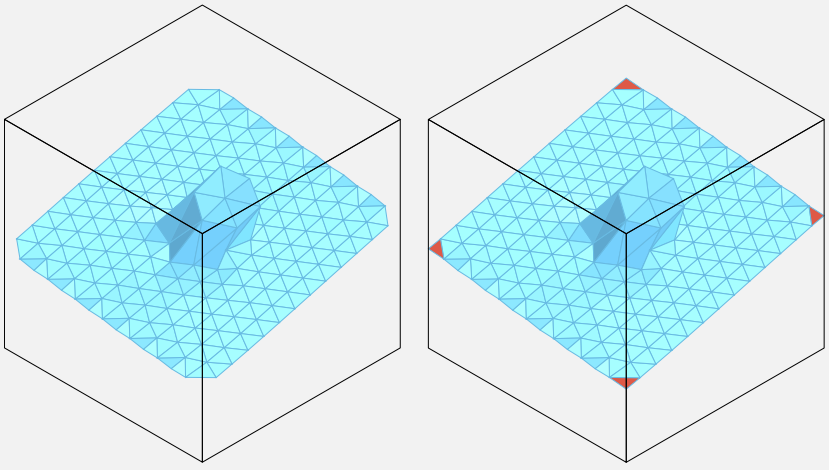


Figure 5.18| Faces are created so that the mesh tightly hangs to the design domain.

5.4.4.2 Splitting a mesh given a polyline

Now that the mesh tightly hangs to the edges of the design domain, we wish to split the design domain into two parts. The only thing we need from the separating surface is the boundary polyline drawn on the faces of the design domain. Thus the question we answer in this section is "*How to split a mesh along a given*

polyline into two sub-meshes?" This is a general question and we answer it in the general case, temporarily forgetting the assembly context. An illustration of the problem is given on FIGURE 5.19: we want to split the cube mesh into two sub-meshes along the rather complex red polyline.

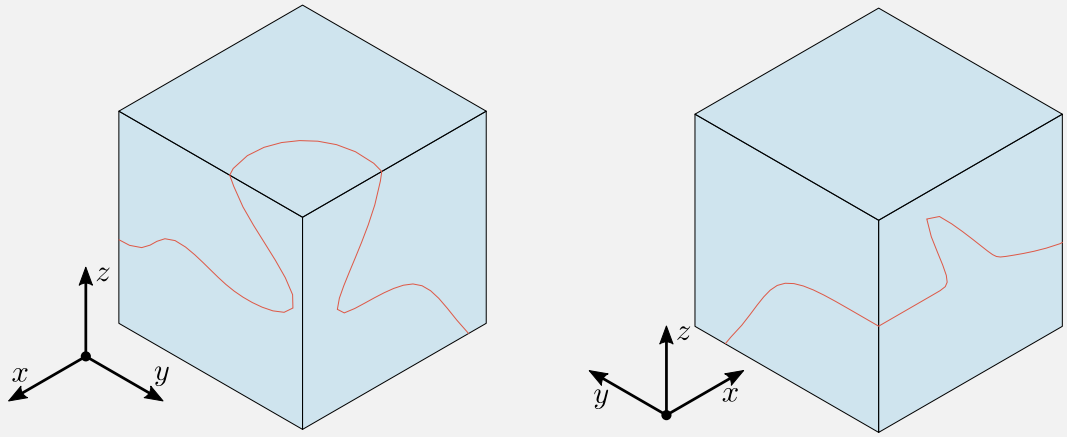
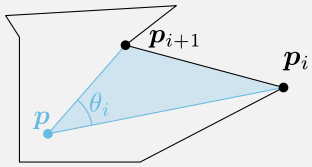


Figure 5.19| The cube shall be split along the red curve.



In two dimensions the **winding number** of a point $\boldsymbol{p} \in \mathbb{R}^2$, $w(\boldsymbol{p}) \in \mathbb{Z}$, is a signed integer associated to a curve telling the number of times the curve wraps around $\boldsymbol{p}$. For each counterclockwise (resp. clockwise) revolution $w(\boldsymbol{p})$ is incremented (resp. decremented). In the discrete case, let $(\boldsymbol{p}_i)_{i\in[\![1,k]\!]}$ denote the points of the polyline, and let θ_i be the signed angle between vectors $(\boldsymbol{p}_i - \boldsymbol{p})$ and $(\boldsymbol{p}_{i+1} - \boldsymbol{p})$, see the inset. Then $w(\boldsymbol{p})$ is given by

$$w(\boldsymbol{p}) = \frac{1}{2\pi}\sum_{i=1}^{k}\theta_i \qquad \theta_i = \operatorname{atan2}((\boldsymbol{p}_i - \boldsymbol{p}) \times (\boldsymbol{p}_{i+1} - \boldsymbol{p}), (\boldsymbol{p}_i - \boldsymbol{p}) \cdot (\boldsymbol{p}_{i+1} - \boldsymbol{p}))$$

Jacobson and coauthors observed in [46] that even if the curve is not closed, this calculation is robust enough to state whether $\boldsymbol{p}$ is on one side of the curve or the other. In the case of an open curve then $w(\boldsymbol{p}) \in \mathbb{R}$; the greater in absolute value, the more confident we can be that $\boldsymbol{p}$ is on a given side. We propose to use such a **generalized winding number** to categorise the vertices of the mesh according to on which side of the splitting curve they are deemed to be. A Delaunay triangulation then create the sub-mesh between the vertices on each side and the polyline. Since the generalized winding number is only defined in the XY plane, we proceed as follows:

- The triangular faces of the mesh are grouped by coplanarity and adjacency.
- Each group of faces is rotated, along with the splitting polyline vertices on them, to the XY plane. A quaternion q can typically be used, see FIGURE 5.20 step a.
- Non-boundary edges are dissolved: we are left with a planar n-gon (a polygon with n edges), 5.20 step b where the triangulation of the cube face was made with 4 triangles.
- The generalised winding numbers of the vertices of the n-gon are calculated. Depending on the sign, they are labelled as being on either side of the polyline, 5.20 step c.
- A planar Delaunay mesh is created twice, between the two groups of n-gon vertices and the points of the polyline, 5.20 step d.
- These two newly created planar meshes are rotated back to their original position using $\hat{q}^*$, 5.20 step e.

FIGURE 5.20 shows on the bottom row the two sub-meshes obtained after executing the above algorithm for

each of the cube faces.

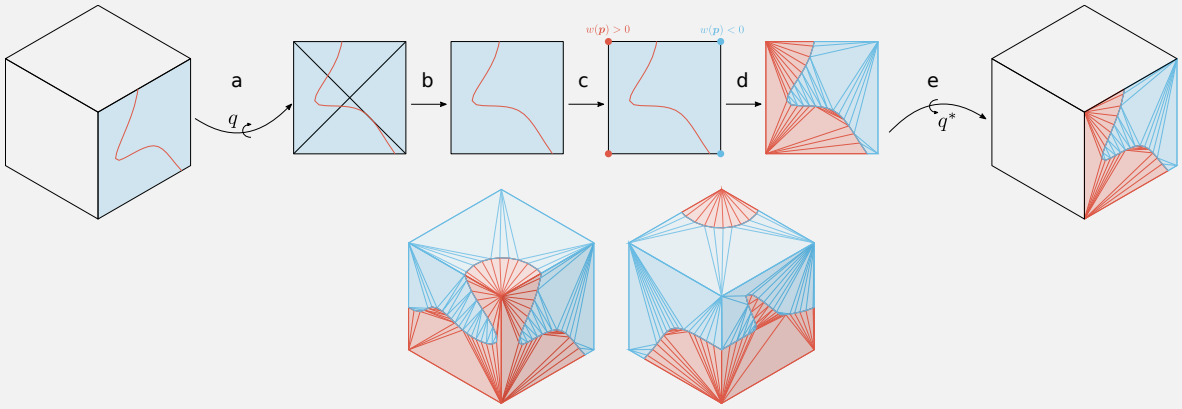


Figure 5.20| Step-by-step process to split a mesh along a polyline.

5.4.4.3 Creating the parts

Given the two open sub-meshes obtained at the end of the previous section, the separating mesh is duplicated, and the copies are joined to each mesh, making the parts P_0 and P_1 complete, with P_1 obeying $\hat{q}$; see FIGURE 5.21 for an illustration.

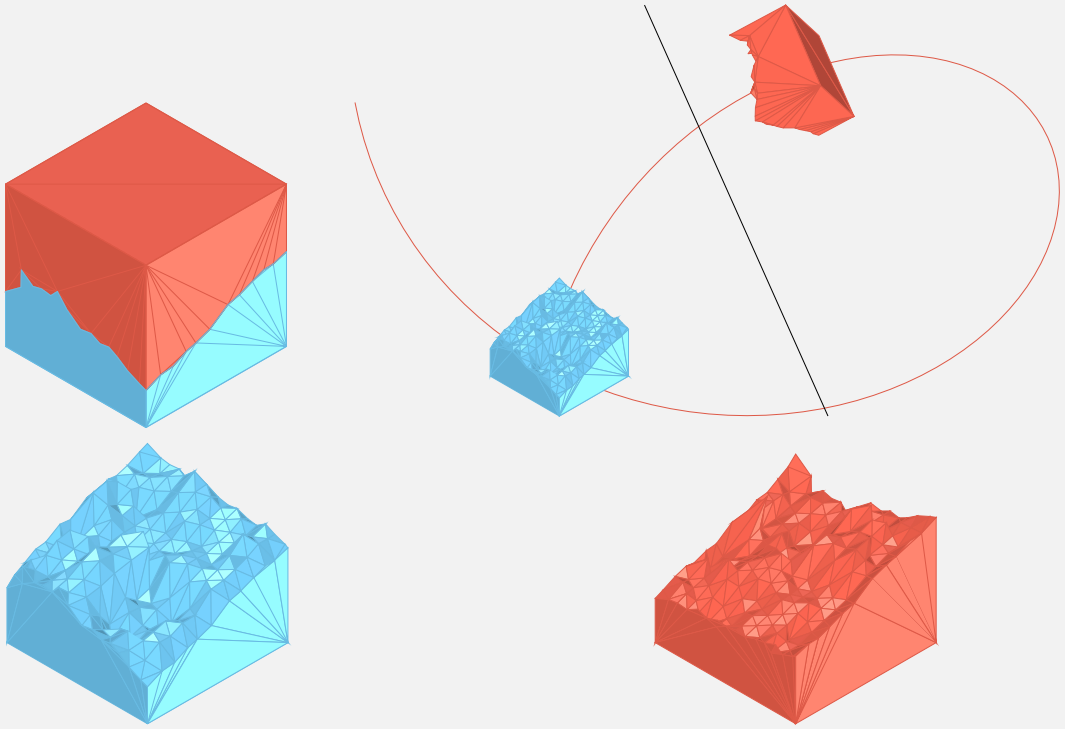


Figure 5.21| A 3D two-parts assembly and the disassembling trajectory.

5.5 ON THE CREATION OF THE FOLLOWING PARTS

The principle is the same as in 2D, see SECTION 4.1.3. Concretely the user specifies in advance the ordered list of unit dual quaternions that each part must obey, as well as the first separating mesh that must be optimised to create P_1. Then, at the end of each iteration, the remaining part P_0 (in blue on FIGURE 5.21) is considered as the next design domain to be partitioned into two, and the user is asked to provide a custom

or vertex painted mesh to create the next part (an automatically generated random mesh may also be used). The algorithm creates the next part and the process repeats until the completion of the assembly. At each iteration, the NDBG is calculated and must say that the assembly is interlocking. FIGURE 5.22 shows a 4+1 parts assembly in the assembled state, as well as each of the five parts. FIGURE 5.23 shows the standard disassembly sequence (P_1 then,..., then P_4), as well as the NDBG stating that the assembly is interlocked.

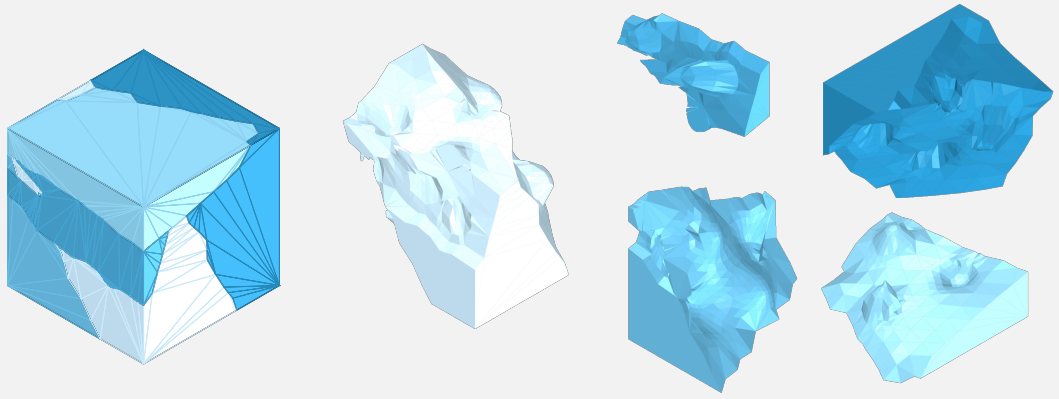


Figure 5.22| A 3D five-parts assembly.

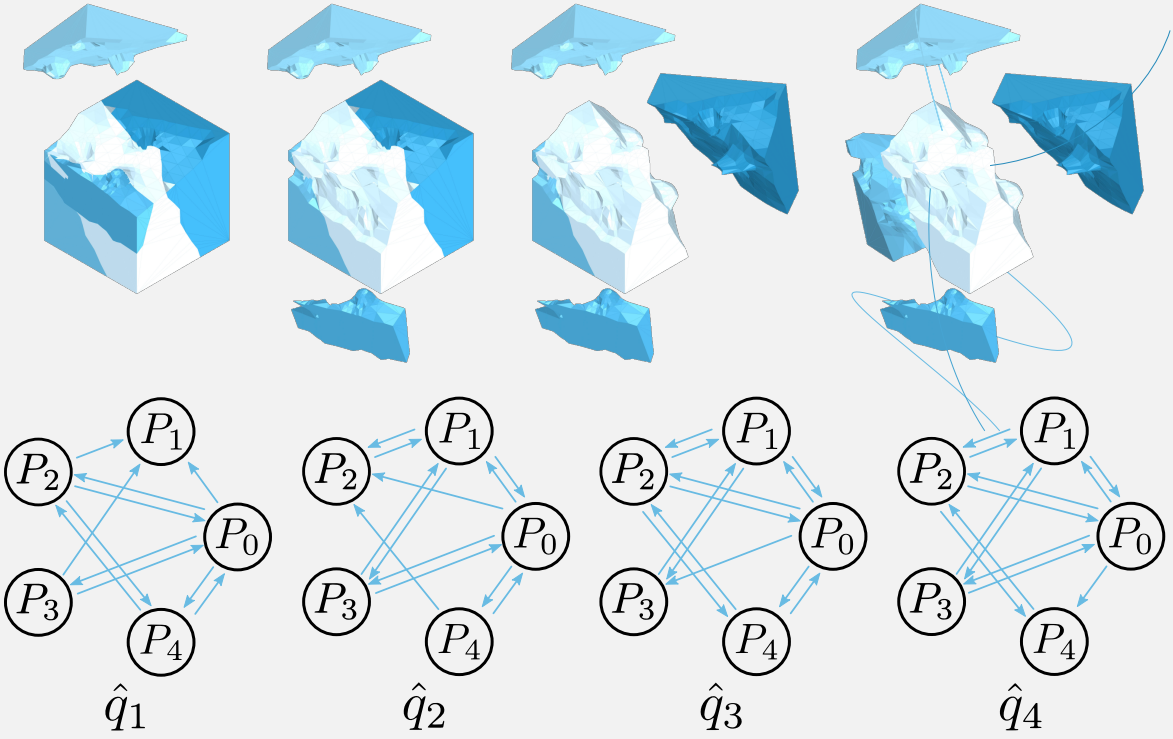


Figure 5.23| A disassembly sequence and the NDBG.

5.5.1 ISSUES WITH THE CURRENT COMPUTER IMPLEMENTATION

In practice, when generating part $P_i, i > 1$, the design domain is often highly irregular: it comes indeed from the part P_0 at the previous iteration whose boundary is itself a triangular mesh optimised to obey a motion, and as such not smooth. It is thus quite difficult for the algorithm to tightly hang the separating mesh to the design domain: it often happens that a face is outside the design domain, or that a boundary edge is not contained in a design domain face (even though its end vertices are sliding on the design domain faces). We

cannot skip this step and let the separating mesh loose, as it is necessary for it to tightly hang to the design domain to calculate the generalised winding number of the design domain vertices. If we try to do so, we are sure that the parts will be ill-constructed. Several code routines were implemented to clean and hang properly the separating mesh, but after seeing that despite weeks of effort there were always problems, we decided to let the user manually clean the mesh at the end of each iteration. Thus, at the time of this writing, generating a 3D assembly is a cumbersome and fastidious task. Most of the work is done by the computer, but the user often has to spend a dozen of minutes per separating mesh. For future work, we advise using a professional mesh handler (or to let the coding to someone with better skills than I!).

5.6 RELATIONSHIPS BETWEEN THE SNAP FACE-VERTEX PAIRS AND THE CONE OF FREEDOM OF MOTION

5.6.1 NUMBER OF SNAP FACES AND CONE OF FREEDOM

We saw in SECTION 4.3.2 that for 2D rotation the more snap segments, the thinner the cone of freedom. Indeed, the more snap segments, the more constraints meet at the prescribed rotation centre $\boldsymbol{x}_r$, and the less likely is any other point to belong to the cone of freedom. This line of reasoning holds in 3D: the more snap face-vertex pairs, the less likely any unit dual quaternion distinct from the prescribed $\hat{q}$ is to be obeyed by the separating mesh. To exemplify the shrinking of the cone of freedom with the number of snaps, we calculated the generating rays of the cones of freedom for various separating meshes with the increasing number of snap face-vertex pairs. Four categories of meshes were used: random ones (generated using Perlin noise), and vertex-painted ones, with one, two or four dents, as shown on FIGURE 5.24.

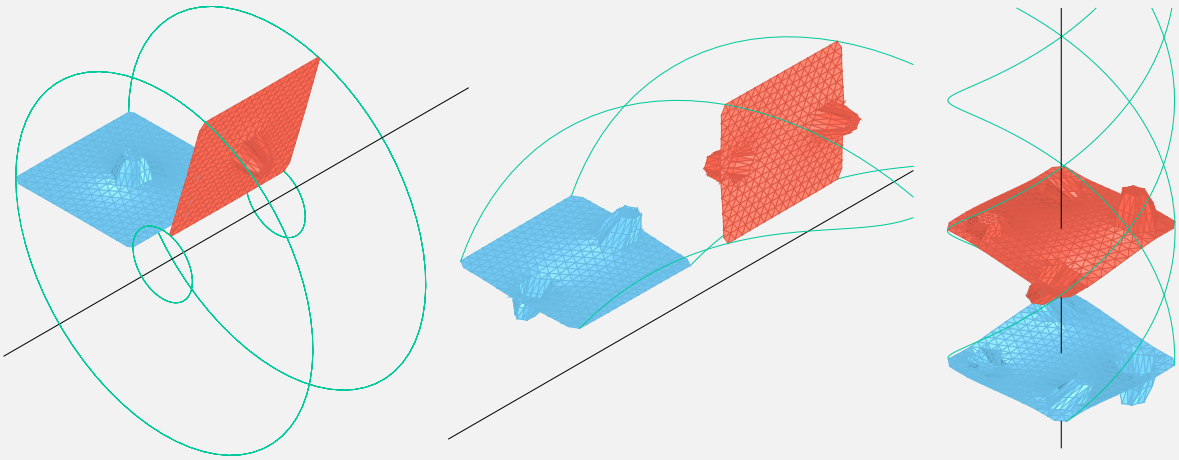


Figure 5.24| The three kinds of vertex painted meshes used, and the three kinds of prescribed motions. The trajectory of the corner vertices of the mesh are shown in green.

They were prescribed to obey either a pure translation (left of FIGURE 5.24) or a screw motion along a horizontal or a vertical axis (middle and right respectively). In total, we used 31 pairs of mesh-motion. Each couple mesh-motion was then optimised 10 times with an increasing number of prescribed snap face-vertex pairs, from 0 to 9. The generating rays of the cones of freedom of each of these 310 optimised designs are calculated and projected onto $\mathcal{S}^5$ the unit hypersphere in $\mathbb{R}^6$. While in 2D we could calculate exactly the area and perimeter of the cones of freedom because we were working in $\mathbb{R}^2$ (or equivalently on $\mathcal{S}^2$), in 3D it is much more difficult to calculate the (hyper)volume of the cone on $\mathcal{S}^5$. We choose thus to approximate this measure by calculating the volume of the convex hull formed by the unit generating rays. The evolution of the volume and area of the cones, as well as the number of unit generating rays, is shown on FIGURE 5.25

(bottom to top respectively). It clearly shows that the cones of freedom shrink with the number of snap face-vertex pairs. It is also remarkable that the volume and area of the cone seem loosely proportional to the number of generating rays. Because we should calculate a DBG for each of such generating rays, it seems preferable to try to have a small cone of freedom to ease the assessment of the interlocking of the assembly.

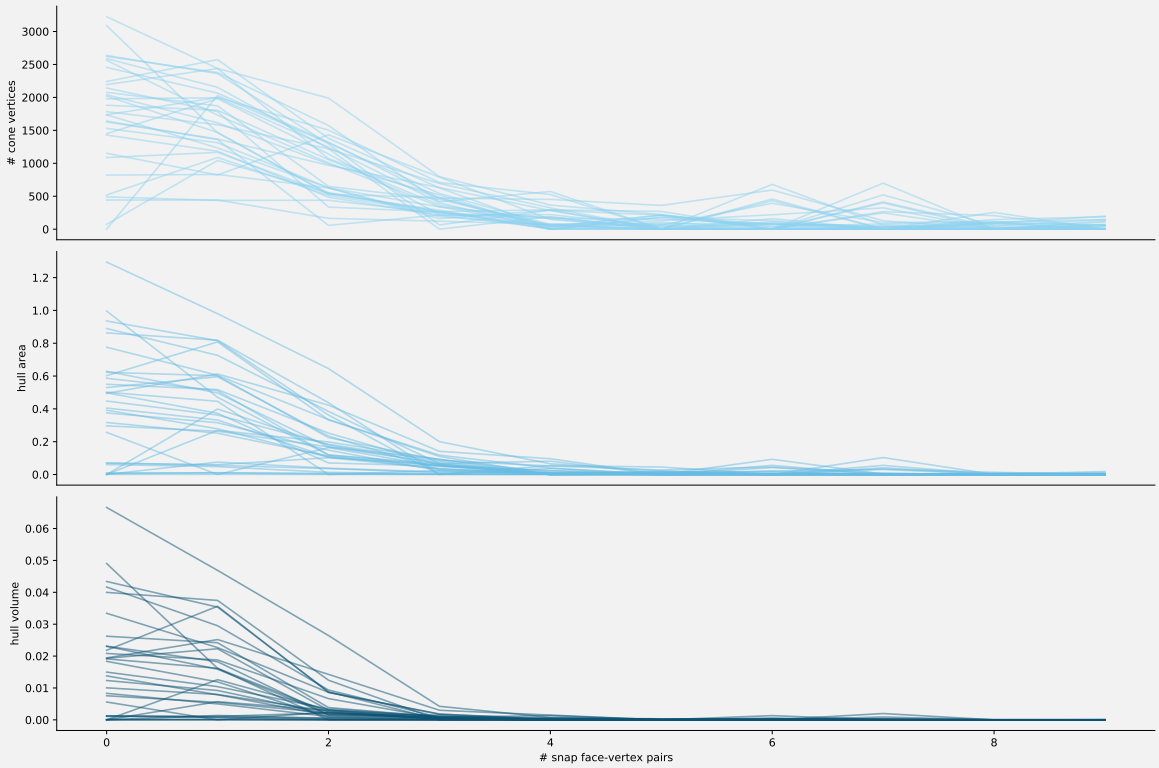


Figure 5.25| The cones of freedom shrink with the number of snaps.

As a side note, the interested reader may look at FIGURE 5.26, which shows the projection of the unit generating rays of a cone on the 15 cartesian planes embedded in $\mathbb{R}^6$.

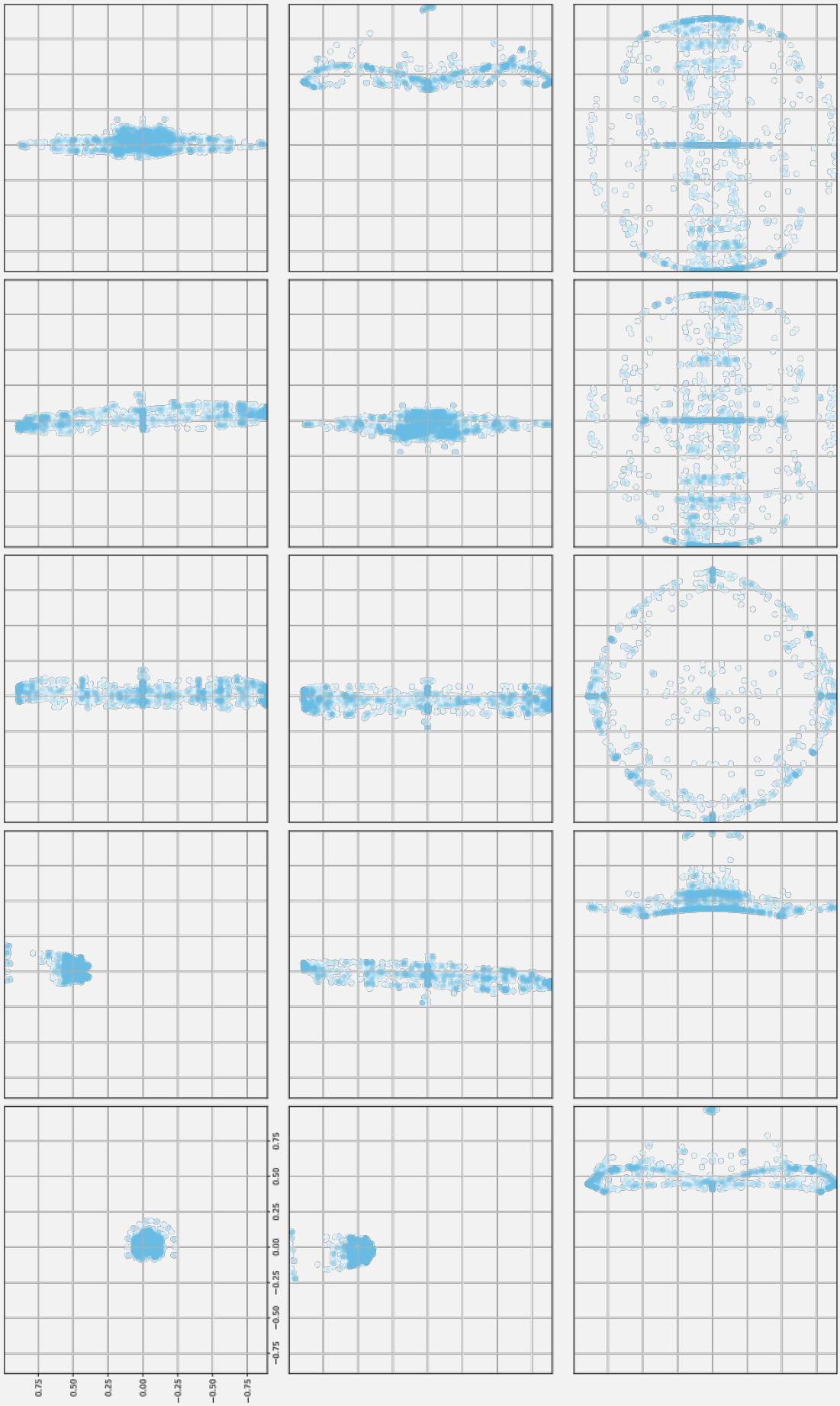


Figure 5.26| Projected unit generating rays. Scale is the same across all figures.

It is important to understand what such a cone means: to any 6D-vector obtained by a convex combination of the generating rays corresponds a unit dual quaternion (see SECTION 5.2.3) that can be obeyed by the mesh. For instance, FIGURE 5.27 shows the 5 unit dual quaternions (seen as 3D lines) corresponding to the five rays generating the cone of freedom for the given surface. Three are shown as blue lines, one is in red and the last is the bold green line. The redder, the greater the relative importance of the translation term θ_ϵ compared with the rotation term θ_0; the bluer, the more important the rotation versus the translation. The bold green line is the prescribed unit dual quaternion (with a pure translation prescribed around it), also corresponds to one of the generating rays of the cone. The dashed black line is calculated from a random convex combination of the five generating rays: it corresponds to a screw motion with a relatively large translation component, as shown by the displaced red surface and the red helical trajectory.

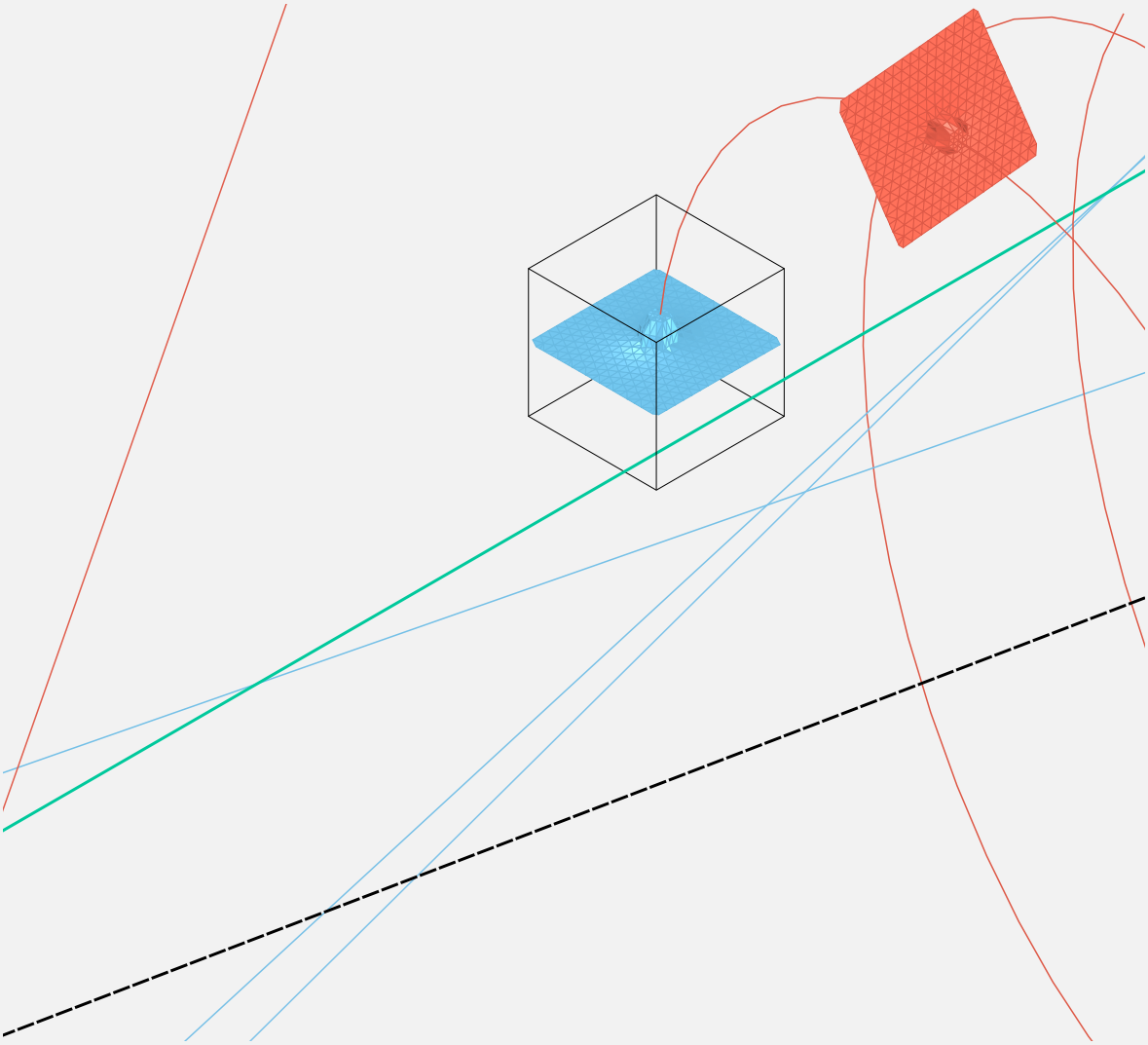


Figure 5.27| The dotted black line represents a unit dual quaternion obtained by convex combination of the generating rays shown as the straight lines in shades of blue and red.

5.6.2 ON FABRICATION IMPERFECTIONS AND MOTION TOLERANCE

As shown both mathematically and numerically in SECTION 4.3.3 in the 2D case, the number of snap elements negatively affects the probability that the prescribed motion $\hat{q}$ is obeyed by an imperfect mesh $\mathcal{M}^\epsilon$, even though the perfect mesh $\mathcal{M}$ does. By imperfect we mean a mesh $\mathcal{M}^\epsilon = (V^\epsilon, E, F)$ whose vertices are

not on their optimised location defined by $\mathcal{M} = (V, E, F)$. In this section, we first introduce a new tool to model imperfections, namely Gaussian processes. We then numerically study the probability that an imperfect mesh $\mathcal{M}^\epsilon$ obeys a prescribed quaternion $\hat{q}$, before performing a more robust optimisation of the mesh $\mathcal{M}$ so that an imperfect mesh $\mathcal{M}^\epsilon$ derived from it still has a high probability of obeying $\hat{q}$.

5.6.2.1 Gaussian processes

In SECTION 4.3.3, imperfections on the polyline were modelled using a centred Gaussian law with a diagonal covariance matrix: it means that each polyline point could move randomly in an isotropic manner and independently from the others. We justified that despite being quite crude, this model gave reasonable results on the probability that an imperfect polyline obeys a 2D motion. In 3D we could have chosen to model the imperfections on the mesh vertices similarly, but instead, we would like to introduce a new tool: Gaussian processes. It is more complex to use but can model a broader range of imperfections and this section could therefore be seen as a potential stepping stone to building a more refined model should the need arises in future work.

A Gaussian process is a stochastic process of variables following a Gaussian distribution, i.e. a dynamical system whose variables are randomly changing over time, each following a Gaussian distribution. The most simple example of a Gaussian process is to follow the altitude of a particle moving vertically by a random amount $\Delta y \sim \mathcal{N}(0, \Delta t)$, between two successive times, steps t and $t + \Delta t$; by construction, it is a Brownian motion. FIGURE 5.28 shows the evolution of the altitude y with respect to time t for five independent random variables. For each realisation (path), FIGURE 5.28 displays the altitude as a function of time: $y = f(t)$. We

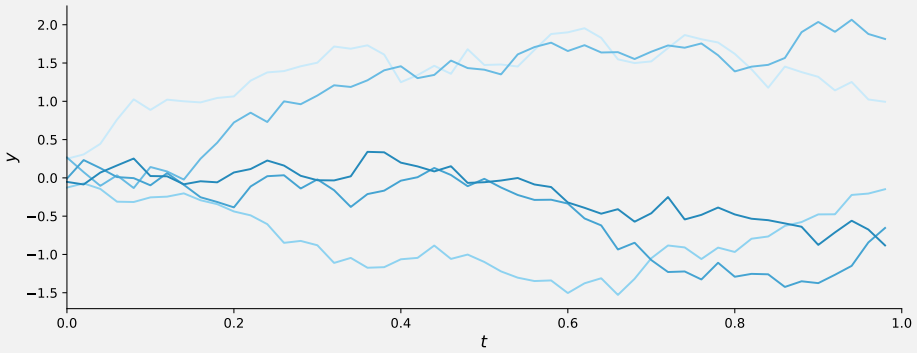


Figure 5.28| Five independent realisations of a 1D Brownian motion.

can therefore interpret this stochastic process as a random distribution over the space of real-valued unidimensional functions. With this in mind, Gaussian processes are distributions over functions f defined over a continuous domain $\mathcal{D} \subseteq \mathbb{R}$ such that:

$$f(x) \sim \mathcal{GP}(\mu(x), \kappa(x, x'))$$

with μ a mean function and κ the covariance function, $x \in \mathcal{D}$ and (x, x') referring to all (infinitely many) pairs of variables of the function domain. Shifting towards discrete settings, assume $\boldsymbol{X} = (x_1, \dots, x_n)^T \in \mathcal{D}^n$ be a vectors with a finite number of entries $x_i \in \mathcal{D}$. Let the mean vector $\boldsymbol{\mu} = \mu(\boldsymbol{X}) = (\mu(x_1), \dots, \mu(x_n))^T$ and the covariance matrix $\boldsymbol{\Sigma} = \kappa(\boldsymbol{X}, \boldsymbol{X})$. The multivariate Gaussian distribution of the subset $\boldsymbol{X}$ is

$$f(\boldsymbol{X}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

The covariance function κ (also called kernel) models the correlation between every pair of random vari-

ables x_i and x_j. The covariance function thus has a strong influence on the realisations of the distribution. To be valid κ must be chosen such that $\boldsymbol{\Sigma}$ is definite-positive. A well-known covariance is the exponentiated quadratic kernel:

$$\kappa(x_i, x_j, a, \sigma) = a^2 \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)$$

stating that the closer x_i and x_j, the more correlated $f(x_i)$ and $f(x_j)$ are. As the distance between them increases, the correlation decreases exponentially, as exemplified in FIGURE 5.29. Once the kernel is defined,

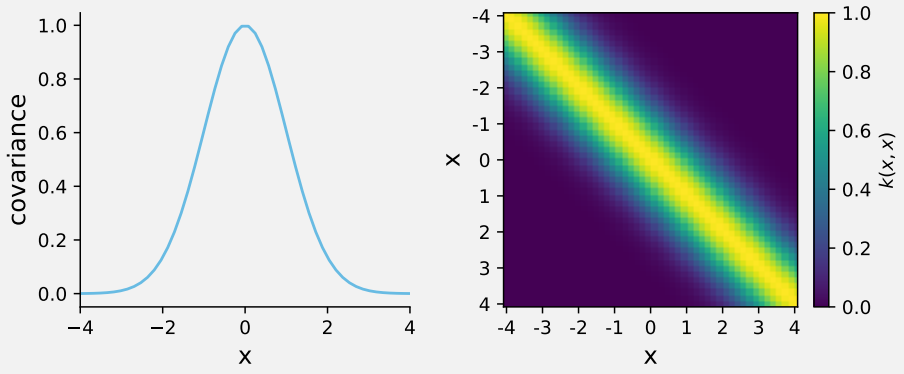


Figure 5.29| Right: exponentiated quadratic covariance matrix for $a = \sigma = 1$.
Left: a cross-cut view of the kernel function $\kappa(x, 0)$.

we choose a mean function μ and we can sample realisations of the Gaussian process, as shown in FIGURE 5.30. The interested reader is referred to APPENDIX E where more kernels are defined.

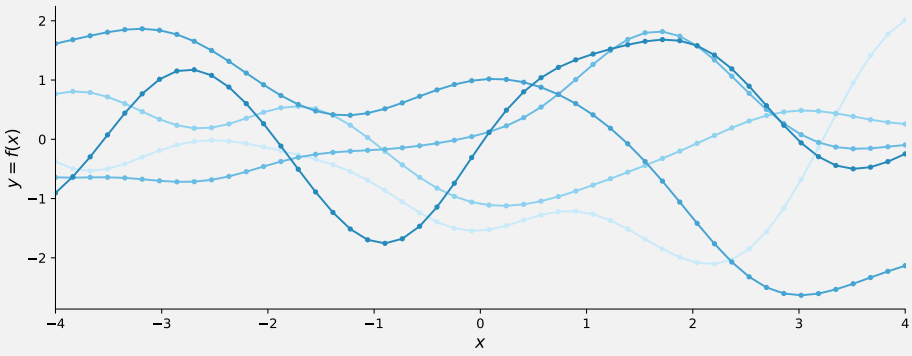


Figure 5.30| Independent samplings $f \sim \mathcal{N}(\mathbf{0}, \Sigma)$ defined using the exponentiated quadratic kernel.

Note that Gaussian processes can easily be adapted to provide the initial separating polyline of a 2D assemblies: the user has greater control over the shape of the polyline than the completely random `Turtle` by the means of judiciously choosing a kernel, but still enjoys the benefits of a fully automatic process (by opposition to manually inputting a polyline). The only downside is that while the `Turtle` is constrained to draw a polyline obeying the prescribed motion, a Gaussian process might output an initial polyline too far away from obeying the motion, resulting in slow or impossible convergence.

5.6.2.2 Imperfection and cone of freedom

We use a Gaussian process to model imperfections on the location of the vertices of the separating mesh: we use the exponentiated quadratic kernel to correlate the imperfections across close vertices. We can tune

parameter σ so that the area of highly correlated points matches the typical area of the imperfection of real assembly. For instance, in 3D printing by fused deposition modelling, imperfections arise from thermal expansion due to the addition of hot material at the printing head. An area of imperfection corresponds to the typical size of the region hotter than the room temperature. Also, the string of printing material may get entangled at the entrance of the heating zone of the printer. When that happens, one has to manually disentangle the string, which may make the printer head move relatively to the printing table, thus introducing imperfections on the remaining object.

We define the kernel for any pair of vertices $\boldsymbol{v}_i, \boldsymbol{v}_j \in V$ of the separating mesh $\mathcal{M} = (V, E, F)$:

$$\kappa(\boldsymbol{v}_i, \boldsymbol{v}_j, \sigma) = a^2 \exp\left(-\frac{d(\boldsymbol{v}_i, \boldsymbol{v}_j)^2}{2\sigma^2}\right)$$

Where $d(\boldsymbol{v}_i, \boldsymbol{v}_j)$ is (an approximation of) the geodesic distance between the two vertices on the mesh. Because calculating the exact geodesic distance is of little use given our motivation for this study (we simply want to numerically show that the prescribed unit dual quaternion $\hat{q}$ has little chance of being obeyed by an imperfect mesh) we use an approximation by computing the shortest path between vertices $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ on the graph $G = (V, E)$ derived from the mesh $\mathcal{M} = (V, E, F)$. Each edge in G is weighted by the euclidean length of the corresponding edge in $\mathcal{M}$. Computing the shortest path between each pair of vertices is done by repeatedly executing the Dijkstra algorithm [24]. This distance is used to compute the covariance matrix $\boldsymbol{\Sigma}$ and then three independent samplings over the mesh vertices V are made, one for each canonical axis of $\mathbb{R}^3$:

$$\tilde{\epsilon}_x(V), \tilde{\epsilon}_y(V), \tilde{\epsilon}_z(V) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$$

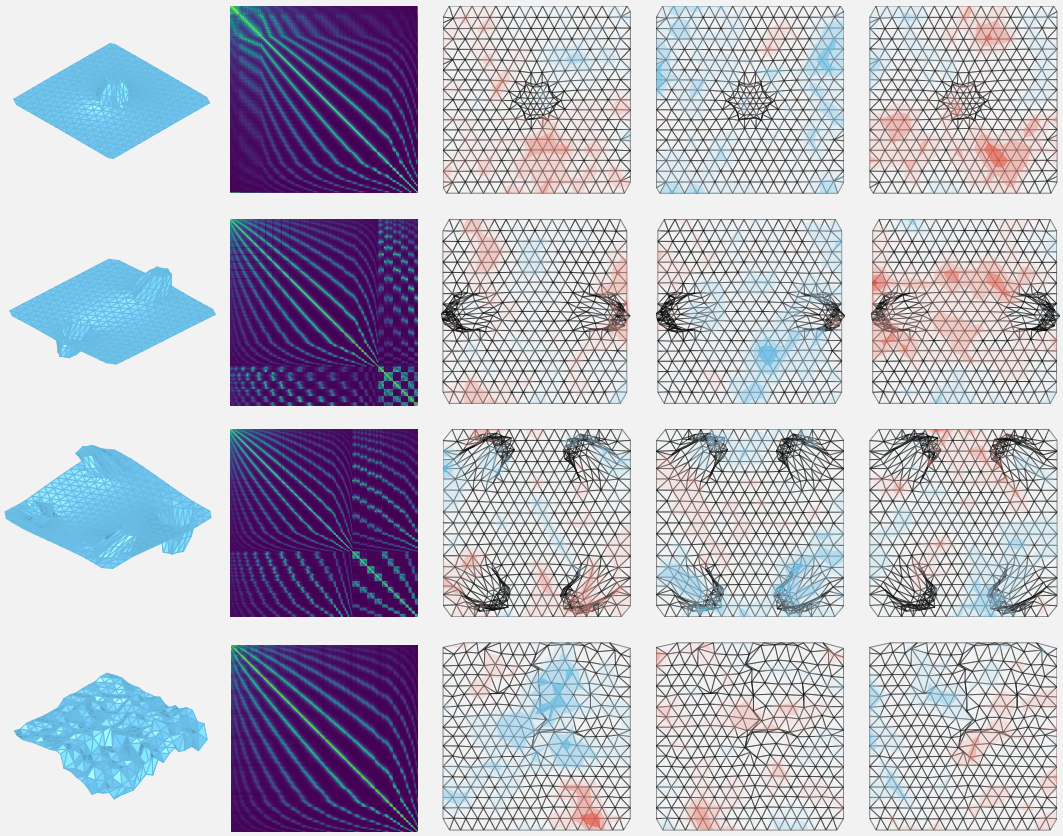


Figure 5.31| From left to right: a mesh, the corresponding covariance matrix $\boldsymbol{\Sigma}$, the heatmaps of the realisations $\tilde{\epsilon}_x(V)$, $\tilde{\epsilon}_y(V)$ and $\tilde{\epsilon}_z(V)$ respectively.

FIGURE 5.31 shows, for four meshes, the covariance matrix calculated using the approximate geodesic distance, as well as the heatmaps of the realisations $\tilde{\epsilon}_x(V)$, $\tilde{\epsilon}_y(V)$ and $\tilde{\epsilon}_z(V)$, vertically projected onto the mesh. The colour code qualitatively indicates the value of the imperfection: blue means strong imperfection to the negative x, y or z directions, red means a strong imperfection in the positive directions and white means an imperfection of small magnitude. The values are correlated to each other depending on the distance between the vertices: we see blobs of imperfection.

To model the fact that in practice the imperfections are more likely to send a vertex away from the mesh rather than on it (for instance, when fabricating a horizontal planar surface, it is more likely to have an error on the altitude of the fabricating device rather than on the horizontal directions; at the limit case a constant error would be irrelevant in the horizontal directions but significant in the vertical one), we build the imperfections as follows.
Once calculated for each vertex $\boldsymbol{v} \in V$ (FIGURE 5.32a), the values $\tilde{\epsilon}_x(\boldsymbol{v})$ and $\tilde{\epsilon}_y(\boldsymbol{v})$ are scaled down by a factor $0 < \alpha < 1$, which effectively nudge the imperfection vector $\tilde{\boldsymbol{\epsilon}}(\boldsymbol{v}) = (\alpha\tilde{\epsilon}_x(\boldsymbol{v}), \alpha\tilde{\epsilon}_y(\boldsymbol{v}), \tilde{\epsilon}_z(\boldsymbol{v}))^T$ to get closer to $\boldsymbol{e}_z$, the vertical direction of $\mathbb{R}^3$ (5.32b). For each vertex $\boldsymbol{v}$ of the mesh we define the vertex normal $\boldsymbol{n}_v$ as the average of the normal vectors of its adjacent faces weighted by their area (5.32c), and we rotate the imperfection vector $\tilde{\boldsymbol{\epsilon}}(\boldsymbol{v})$ using the quaternion q sending the normal vector $\boldsymbol{n}_v$ to the vertical direction $\boldsymbol{e}_z$ to get the final imperfection vector $\boldsymbol{\epsilon}(\boldsymbol{v})$ (5.32d). Thus each imperfection vector is biased to move the corresponding vertex in the direction locally orthogonal to the mesh. The imperfect location of a vertex $\boldsymbol{v}$ is then calculated as $\boldsymbol{v}^\epsilon = \boldsymbol{v} + \boldsymbol{\epsilon}(\boldsymbol{v})$, from which we get the set of imperfect vertices V^ϵ and the imperfect mesh $\mathcal{M}^\epsilon = (V^\epsilon, E, F)$. FIGURE 5.32e shows, for each vertex $\boldsymbol{v}$, 50 independent $\boldsymbol{v}^\epsilon$ in red: we see that they are all biased to move in the direction locally orthogonal to the mesh.

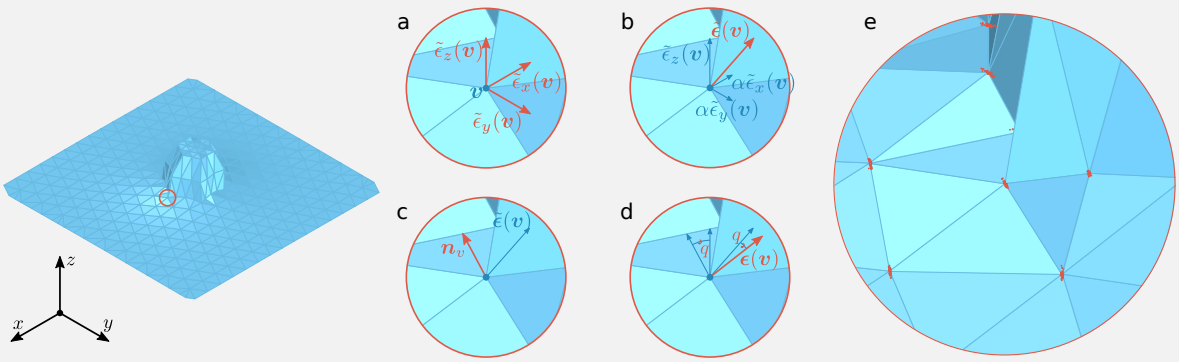


Figure 5.32| Process to build the imperfections: each vertex is moved in the direction locally orthogonal to the mesh.

Each optimised mesh $\mathcal{M}$ (of the 310 solutions) is deformed 100 times using the aforementioned process to get a family of imperfect meshes $\mathcal{M}^\epsilon$. The cone of freedom $\mathcal{C}_{\mathcal{M}^\epsilon}$ of each such imperfect mesh is calculated and we count the number of times the prescribed quaternion $\hat{q}$ belongs to the interior of the cone. From that number we can calculate the sampled mean, which estimates the probability $\mathbb{P}\left(\hat{q} \in \mathcal{C}_{\mathcal{M}^\epsilon}\right)$. FIGURE 5.33 shows the evolution of these estimated probabilities with the number of face-vertex snap pairs. To be clear, in this figure there are 31 curves made of 10 points. Each of these points represents the estimated probability (sample mean) calculated with 100 imperfect designs. Superimposed in black is the graph of the function $n \mapsto \frac{1}{2^n}$: we see that, as proven in SECTION 4.3.3, the decrease of the likelihood of obedience follows a power law.

We can also study in more detail the cone of freedom of the imperfect meshes, $\mathcal{C}_{\mathcal{M}^\epsilon}$. To that end, we calculated the number of generating rays, the (hyper) volume of the convex hull of the cone, as well as its area. FIGURE 5.34 is packed with information. The small figures on the top left displays the generating rays

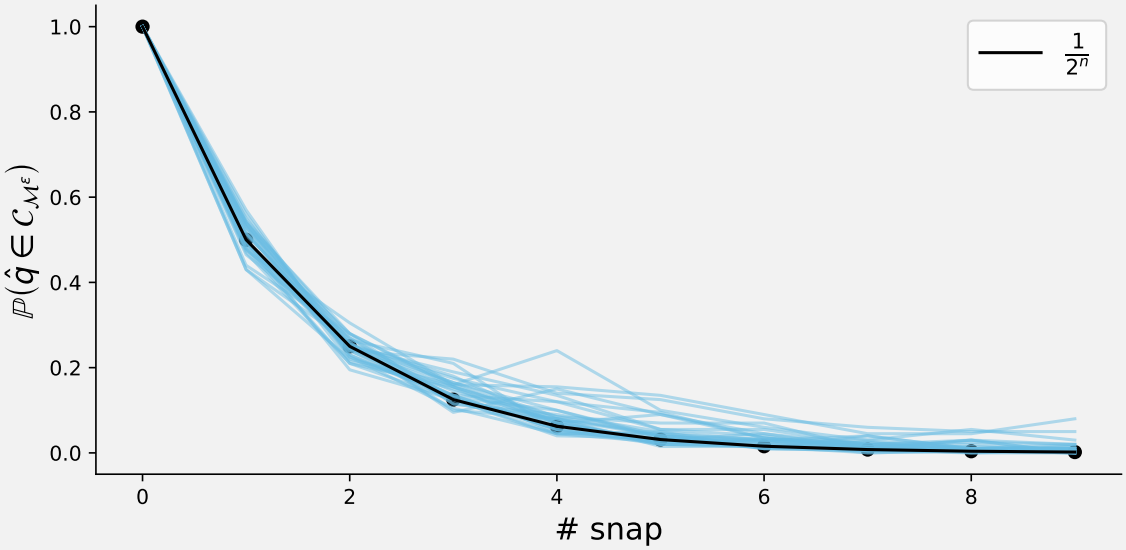


Figure 5.33| Evolution of the likelihood of obedience of an imperfect mesh $\mathcal{M}^\epsilon$ with the number of pairs of snap face-vertex.

of $\mathcal{C}_{\mathcal{M}^\epsilon}$ in the 15 cartesian planes, same as FIGURE 5.26. On the top middle is shown the 2D PCA performed on the rays, as well as the convex hull of the projection in red. Bottom left and middle: views of the mesh $\mathcal{M}^\epsilon$ with, in green, the prescribed motion (pure rotation in this case) and in colours the unit dual quaternions defining the cone; the brighter the red the more $\theta_0 > \theta_\epsilon$, the deeper the blue the more $\theta_\epsilon > \theta_0$. Right: plots showing the evolution of the area and volume of the convex hull of the cone as well as the number of vertices, in both standard and log scale. They were calculated for various families of $\mathcal{M}^\epsilon$, for various numbers of prescribed snap face-vertex pairs. The highlighted red points correspond to the mesh on the figure. It shows that for low values of snaps (3 or less), the cones of freedom of the imperfect meshes $\mathcal{C}_{\mathcal{M}^\epsilon}$ vary wildly: there is a relatively high spread of the values of the three measured metrics. For more than 3 snaps, the variation is of much lower magnitude, implying that even the imperfect meshes all have a rather small cone of freedom. The reader is referred to APPENDIX F for more examples of this kind of data, for other separating meshes.

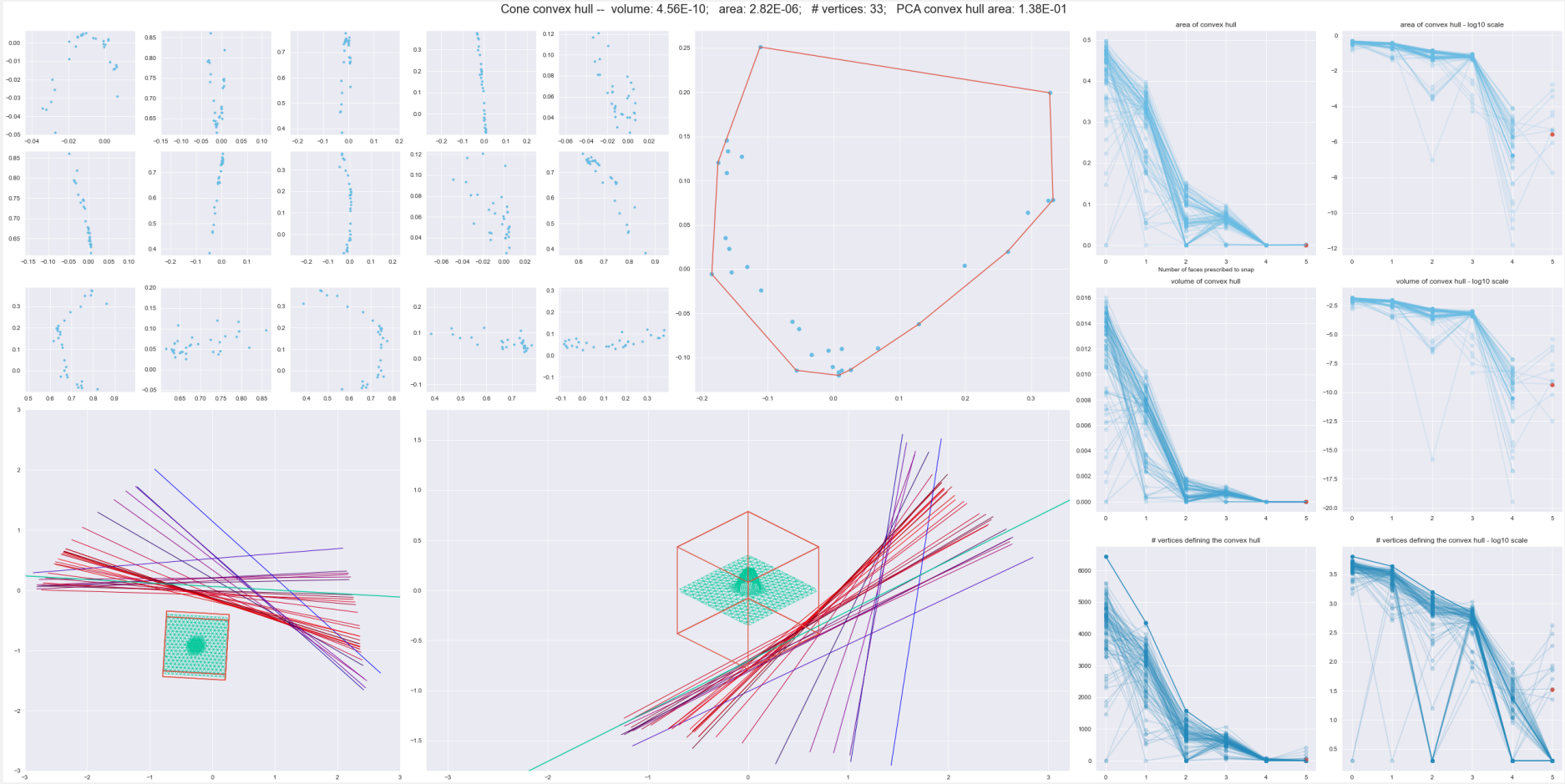


Figure 5.34| See text for a description.

5.6.2.3 Robustness to imperfection

To make the mesh $\mathcal{M} = (V, E, F)$ more robust to imperfection we propose to optimise it, similarly to what we did in 2D. We aim to find a mesh $\mathcal{M}$ such that any imperfect mesh $\mathcal{M}^\epsilon$ derived from it has a high probability of obeying to $\hat{q}$. As in 2D, we first increase the prescribed number of snap face-vertex pairs until the cone of freedom of the mesh is reduced to the prescribed motion: $\mathcal{C}_\mathcal{M} = \{\hat{q}\}$. Then, a second round of GPA optimisation is executed where the snap constraint is suppressed and, for some $\alpha > 0$, we impose that

$$\forall f \in F \ \forall \boldsymbol{v} \in V(f) \ \boldsymbol{m}(\boldsymbol{v}, \hat{q}) \cdot \boldsymbol{n}_f \geq \alpha \tag{5.22}$$

To confirm that this process does increase the robustness of $\mathcal{M}$ we calculated 100 imperfect mesh $\mathcal{M}^\epsilon$ and counted the number of times $\hat{q} \in \mathcal{C}_{\mathcal{M}^\epsilon}$, from which we can derive an estimated probability (sampled mean) $\mathbb{P}(\hat{q} \in \mathcal{C}_{\mathcal{M}^\epsilon})$, i.e. the probability that a mesh with imperfections $\mathcal{M}^\epsilon$ obeys the prescribed motion $\hat{q}$. The greater this probability, the more robust $\mathcal{M}$. The mesh used is shown on FIGURE 5.32; it obeys a pure rotation. FIGURE 5.35 shows the evolution of the probability $\mathbb{P}(\hat{q} \in \mathcal{C}_{\mathcal{M}^\epsilon})$ with respect to increasing α (top left). It also shows the main data on the cones $\mathcal{C}_{\mathcal{M}^\epsilon}$: the number of vertices (unit generating rays) defining it, as well as the (hyper)volume and (hyper)area of the convex hull of these vertices. The blue curve shows the mean value averaged on 100 imperfect meshes $\mathcal{M}^\epsilon$, and the filled region shows the inter-quartile range 25%-75%: it means that the middle half of the observed values lie in that range.
Unsurprisingly, the greater α (geometrically, the farther away are pushed the hyper-planes constraints from $\hat{q}$), the more likely an imperfect mesh $\mathcal{M}^\epsilon$ is to obey $\hat{q}$: the more robust the assembly to the prescribed motion and the more room the operator has when (dis)assembling the parts. The downside is that the greater α the more voluminous the cone, and thus the more motions can be obeyed by the mesh, making the assembly less robust to parasitic motions: it may become too easy to disassemble for any practical use. Moreover the top right plot of FIGURE 5.35 shows that the greater α the more rays define the cone. This is problematic as to assess the interlocking of the assembly a DBG must be calculated for each ray. If the cones of different parts intersect between them, the number of DBG quickly explodes. To prevent that from happening, we observe that if the cones associated with two distinct parts do not intersect, then the DBGs associated with the vertices of a given cone are all constant, by exploiting the regularity property of the DBG over a cell. Thus it is in our interest to reduce the extent of the cones to prevent any intersection and to reduce the complexity of the assessment of the interlocking.
We understand here that we are caught in a crossfire: on the one hand, increasing the size of the cone by the mean of α increases the probability of obedience of an imperfect mesh $\mathbb{P}(\hat{q} \in \mathcal{C}_{\mathcal{M}^\epsilon})$. On the other, too large cones are likely to intersect, making the calculation of the NDBG much more complicated.
This conundrum leads to an interesting optimisation problem: we can leverage the monotonicity of the evolution of the cone size (number of vertices (generating rays), volume and area of its convex hull) with respect to α to minimise the value of α still leading to a high probability of obedience:

$$\begin{aligned} \min \quad & \alpha \\ \text{s.t.} \quad & \mathbb{P}(\hat{q} \in \mathcal{C}_{\mathcal{M}^\epsilon}) \geq p_{\min} \end{aligned}$$

For some threshold $p_{\min}$ (typically $p_{\min} = 0.99$). Hence, one gets the optimal value α^* leading to the smallest possible cone $\mathcal{C}_{\mathcal{M}^\epsilon}$ (by monotonicity of the cone size with respect to α) having a sufficiently high probability of containing the prescribed motion encoded in $\hat{q}$. On FIGURE 5.35 this value is eyeballed to be $\alpha^* \simeq 0.048$. We insist on the fact that α^* is motion and geometry-dependent: for another mesh and/or another prescribed $\hat{q}$ it would have been different.

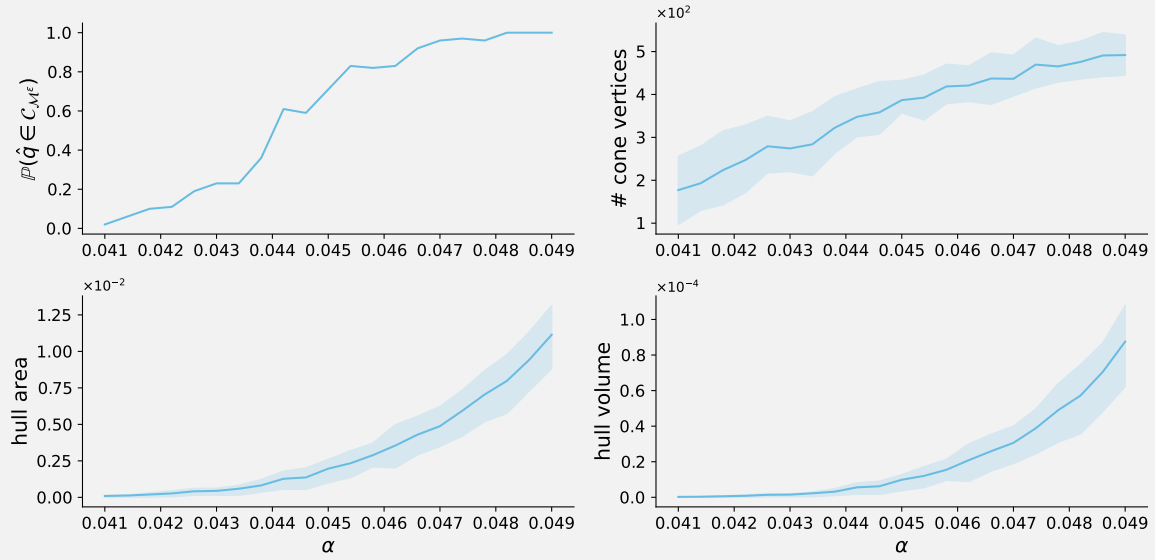


Figure 5.35| The greater α, the larger the cone, and the greater $\mathbb{P}(\hat{q} \in \mathcal{C}_{\mathcal{M}^\epsilon})$. For $\alpha \in [0.04, 0.05]$ the top left figure shows the probability that an imperfect mesh $\mathcal{M}^\epsilon$ obeys the prescribed $\hat{q}$: it goes from $\simeq 0$ to 1. Top right: the greater α, the more rays generates the cones of freedom $\mathcal{C}_{\mathcal{M}^\epsilon}$. Bottom left and right respectively show that the area and volume of the 6D convex hull of the generating rays increase with α: the greater α the wider the cones.

5.7 CONCLUSION

After a rather in-depth introduction to dual quaternions in SECTION 5.1, this chapter explains how we can use the tools and concepts introduced in CHAPTER 4 to automatically generate 3D polyhedral sequential assemblies. Several weaknesses of our work in 2D are listed in SECTION 4.6 and still apply in the 3D case, and will not be recalled here.

As understood in SECTION 5.1 unit dual quaternions are the perfect mathematical objects to encode rigid body motions. The trajectory of a point $\boldsymbol{p} \in \mathbb{R}^3$ transformed by a unit dual quaternion $\hat{q} \in \mathcal{U}(\mathbb{DH})$ is helical, and we can easily calculate its initial velocity, the so-called instantaneous direction of motion $\boldsymbol{m}(\boldsymbol{p}, \hat{q}) \in \mathbb{R}^3$. From there we proved the rather elegant result, that for a triangular mesh $\mathcal{M} = (V, E, F)$:

$$\mathcal{M} \text{ obeys } \hat{q} \iff \forall f \in F \, \forall \boldsymbol{v} \in V(f) \; \boldsymbol{m}_v \cdot \boldsymbol{n}_f \geq 0$$

Armed with $\boldsymbol{m}(\boldsymbol{p}, \hat{q})$ we easily adapted, in SECTION 5.4 the Guided Projection Algorithm to optimise a mesh so that it obeys a motion prescribed by the user. Special attention was given to the snap constraint: we understood that at least three snap face-vertex pairs must be prescribed if one hopes to reduce the cone of freedom of the mesh to the prescribed motion $\mathcal{C}_{\mathcal{M}} = \{\hat{q}\}$. A careful analysis of the Gauss map $G(\mathcal{M})$ was carried out to judiciously choose the snap face-vertex pairs, so as to decrease as much as possible the extent of the cone of freedom $\mathcal{C}_{\mathcal{M}}$: the normal vectors of the snap faces must be evenly distributed around the sphere.

Stepping away from these mathematical considerations, the rest of the section focuses on details of the actual computer implementation. While the numerical optimisation of the mesh is fast and yields good results, we find that, far from being a trivial task, handling and hanging irregular coarse meshes to one another to create parts still necessitates a human touch, much to our chagrin. At the end of this PhD, it is still a tedious

task to generate 3D assemblies.

SECTION 5.6 investigates the relationship between the number of snap face-vertex pairs and the cone of freedom. Much like in 2D (see SECTION 4.3), the greater the number of snaps the smaller the cone. Imperfections are modelled using Gaussian processes, where the kernel correlating the noise between the vertices takes into account the geodesic distances between them. By tuning the parameter σ we can change the size of the patches of highly correlated noise values, thus opening the way for the modelling of a broad range of imperfections. We saw that statistically, the more snap face-vertex pairs, the less likely an imperfect mesh is to obey the prescribed $\hat{q}$, which led us to robustly optimise the mesh to increase this probability by opening the cone of freedom, at the cost of making the assembly less robust to parasitic motions and potentially making the calculation of the NDBG much more complicated if two cones intersect.

REFERENCES

2 3Blue1Brown. *Visualizing quaternions (4d numbers) with stereographic projection*. Youtube. 2018. URL: https://www.youtube.com/watch?v=d4EgbgTm0Bg.

21 Clifford. "Preliminary Sketch of Biquaternions". In: *Proceedings of The London Mathematical Society* (1873), pp. 381–395.

22 Konstantinos Daniilidis. "Hand-Eye Calibration Using Dual Quaternions". In: *The International Journal of Robotics Research* 18.3 (Mar. 1999), pp. 286–298. ISSN: 0278-3649, 1741-3176. URL: http://journals.sagepub.com/doi/10.1177/02783649922066213 (visited on 01/09/2020).

24 Edsger W Dijkstra et al. "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1 (1959), pp. 269–271.

45 Dunham Jackson. "The Instantaneous Motion of a Rigid Body". In: *The American Mathematical Monthly* (Dec. 1942), 661 to 667.

46 Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. "Robust Inside-Outside Segmentation Using Generalized Winding Numbers". In: *ACM Trans. Graph.* 32.4 (July 2013). ISSN: 0730-0301. URL: https://doi.org/10.1145/2461912.2461916.

50 Ladislav Kavan et al. "Dual Quaternions for Rigid Transformation Blending". In: (2006), p. 10.

59 J.M. McCarthy. *An Introduction to Theoretical Kinematics*. The MIT Press, 1990.

66 *Neue Geometrie des Raumes gegründet auf die Betrachtung der geraden Linie als Raumelement Abth.2*. ger. 1869. URL: http://eudml.org/doc/203056.

86 Alexander Schiftner et al. "Packing Circles and Spheres on Surfaces". In: *ACM Trans. Graph.* 28.5 (Dec. 2009), pp. 1–8. ISSN: 0730-0301. URL: https://doi.org/10.1145/1618452.1618485.

88 Ken Shoemake. "Animating rotation with quaternion curves". In: *SIGGRAPH '85*. 1985.

CHAPTER 6

CONCLUSION AND PERSPECTIVES

The unveiling climate catastrophe and the rarefaction and disappearance of the resources modern societies are built upon urge the engineer to find solutions to soften the coming blow. In that spirit, demountable edifices undoubtedly constitute an arrow in the quiver of the builder to erect sustainable cities. But as understood by Mam [58], the designer should be careful of the amount of metal used in an assembly. From a historical perspective, integral reversible joints, even if they increase the overall volume of material, are found to be a sustainable alternative to permanent fasteners which explains the interest of national and international initiatives such as DiXite in that line of research. While we can certainly leverage centuries of human trial and error to design sophisticated reversible joints, numerical technologies allow us to quickly explore the space of possible joints, and digital manufacturing holds the promise of being able to fabricate novel assemblies with strange geometrical features that are potentially even more relevant than traditional ones. This dissertation aimed at bringing a humble stone to this edifice by explaining the tools and algorithms that were developed to fully explore the space of interlocking assemblies.

We believe our main contributions to this field of research to be:

- Nor catalogue nor voxel-based: our method fully explores the space of polygonal/polyhedral assemblies while requiring very few human inputs, namely an ordered list of disassembling motions.
- To our knowledge, we are the first to generate 2D assemblies obeying rotations which opens the gate to generate a much broader range of assemblies than before. Similarly, we think to be the first able to generate 3D assemblies obeying generalised motions.
- While robust optimisation is a standard procedure in many scientific fields, we have not found in the literature the robust optimisation of integral-joints

assemblies. Our work here on opening the cone of freedom should be of great help in the field of robotic assemblies.

6.1 RESULTS AND CONTRIBUTIONS

6.1.1 NON DIRECTIONAL BLOCKING GRAPH - NDBG

For this dissertation to be self-contained, we wished to give to the reader a solid understanding of the theoretical modelling of the blocking relationships between parts in an assembly. While CHAPTER 3 does not present any finding, it sums up the state of the art on this subject. A Directional Blocking Graph, DBG, is a motion and a graph whose vertices correspond to the assembly's parts and a directed edge between two vertices indicates that one is being blocked by the other for a motion of infinitesimal magnitude. The concatenation of the DBGs for all possible motions given the Non Directional Blocking Graph, NDBG, which encapsulates the entire blocking relationships between the part. We understood that quite fortunately the assessment of the interlocking can be made by looking at a discrete and finite number of so-called base DBGs. The motions in these base DBGs corresponds to the vertices of the cones of translational and rotational freedom, $\mathcal{C}_t, \mathcal{C}_r^{ccw}, \mathcal{C}_r^{cw}$, which can easily be found by solving a linear system. If all base DBGs but one are strongly connected then the assembly is interlocked, and the DBG not strongly connected indicates which part is the key and along which motion the assembly can be disassembled.

6.1.2 2D ASSEMBLIES

Armed with our knowledge of the NDBG we decided to adopt a subtractive approach to generate multi-parts assembly: the successive parts are nested in one another by creating the next part P_{i+1} in the remaining part P_0 at the end of the previous design step. At the end of each iteration, the NDBG is calculated to ensure that the current, unfinished, assembly is interlocked, which leaves the computer free to find whichever set of parts block the motion of P_{i+1}.
Two generative algorithms were presented.
At first, we introduced a novel method to generate 2D sequential interlocking assemblies obeying translation, rotation, or a combination of both motions. It involves an obeying agent that we called `Turtle` and an instruction-giver Markov process. The Markov process randomly switches between states and for each of such states, the `Turtle` is tasked to walk by a random magnitude or to rotate by randomly sampling a carefully calculated angular interval, thus drawing a random separating polyline to partition the design domain with. One of the main hypothe-

ses of our work is the fact that the parts are polygonal, i.e. the separating curve between two parts is a polyline. Yet, as we derive the mathematical formula governing a valid assembly, one notices that these equations could readily be used to explore a broader solution space and design non-polygonal assemblies, for instance using NURBS instead of polylines to create the separating curves. A key feature of our work is the surjectivity of the mapping to the solution space. This ensures that, given enough trials and computation time, any polyline can be generated and thus that we sample homogeneously the full space of polygonal assemblies. Due to the central place we gave to randomness our approach yields surprising and novel assemblies, in particular assemblies obeying rotations, which, to our knowledge, that has not been made before. Yet, we found that the mathematical formulae driving the `Turtle` did not always have a solution, a major impediment to this method.
As a consequence we decided to use a less intuitive but more robust optimisation method, the Guided Projection Algorithm, GPA, to optimise a polyline so that it partitions the design domain into two parts. The `Turtle` approach is not entirely discarded as a simplified version instantiates the initial solution of the algorithm. The GPA proves to be a powerful and versatile tool in which the user can easily enough add constraints to guide the final solution, thus letting them choose the amount of freedom they want to have on the generated design.
At this step, the reader may have thought that the generated assemblies had strange geometrical features making manufacturing cumbersome with no proof or explanation on why they should be preferred to simpler and more traditional assemblies. Section 4.2 justify the relevance of this work by assessing the mechanical properties of the generated assemblies. Mechanical analyses are performed through a custom numerical model. The agreement between the forces calculated by our model and the displacement observed by image correlation on physical assemblies reinforces our view that our model accurately describes the stresses and strains happening in the structure. Systematic comparisons of the stresses calculated in the parts of generated assemblies and of those of a reference assembly consistently showed that a simple random sampling of the design space yields a good number of designs strictly dominating the reference, much to our surprise. This finding hints at the potential of our approach for real-life assemblies. Yet, we insist on the fact that the mechanical analysis of the generated assemblies was not the primary goal of this study and we leave to future work the confirmation of this study, e.g. by comparing the behaviours of physically built assemblies.
What we deem to be the most relevant work in this doctoral thesis has to do with the robust optimisation of assemblies with regard to fabrication imperfections and/or imprecision on the location of the operator tasked with (dis)assembling. We found that a simple two-step GPA optimisation, with and without the snap constraint ac-

tivated, let the user indirectly choose the size of the cones of rotational freedom of the part (the translation case being immediate to deal with). In turn, this opens the way to *toleranced assembly*, a major subject in robotic assembly.
We also made a quick, and crude, enquiry on the side of finite motions to find the set of possible centres of rotation to place the (dis)assembling operator.
This chapter finishes with a more mathematical touch, as the theory behind the elastic deformation of planar open curves is presented for us to be able to smoothly interpolate between two generated designs. We found that interpolated designs mechanically behave similarly to each other making it easy to find many assemblies dominating a reference one.

6.1.3 3D ASSEMBLIES

In our opinion, the main interest of CHAPTER 5 lies in the small but hopefully thorough introduction to unit dual quaternions. They are little-known mathematical objects that are particularly fit for encoding rigid body motions in 3D space. In this dissertation, the tangential velocity of the trajectory of a vertex moved by a unit dual quaternion was used as the instantaneous direction of motion in the optimisation of a triangular mesh. The rest of the chapter simply scales the tools used to generate 2D assembly to the 3D cases.

6.2 PERSPECTIVES

Several issues have been listed throughout this manuscript and should be addressed by future research to strengthen this work. The most straightforward to deal with relates to the fact that the NDBG outputs a binary answer: either an assembly is interlocked or it is not. An implicit assumption in the calculation of the base DBGs is that the coefficient of friction is infinitely high. Yet, for physically built assemblies we can feel that friction plays an important role, making assemblies theoretically interlocked disassemblable in practice. Thus the calculation of the DBGs should be more subtle, taking into account this phenomenon.
Another route would be to reverse our approach: instead of generating a part and then computing the NDBG and restarting this iteration from scratch if the NDBG does not say that the assembly is interlocked, we should follow the approach proposed by Wang *et al.* in [105] and first compute the NDBG to make it suitable and to highlights which previous parts the current one should be blocked by, then generate the separating polyline/mesh anchored in these highlighted parts.
While in 2D a mechanical analysis of the parts was performed, this work was not extended to the 3D case. On top of that, our findings regarding the fact that a random sampling of the design space often yields mechanically relevant assemblies should

be confirmed by physical tests.
Our work must of course be extended to the highly non-linear realm of finite motions.
In our opinion, the most serious weakness of this work is that we did not consider the manufacturability of the parts. This corresponds to the spirit of this dissertation: who can do more can do less. Since we can reach the entire design space of interlocking assemblies, we can in particular reach the subset of manufacturable assemblies. It should be stressed that this subset depends on the tools the manufacturer chooses to use and thus that specific constraints to curate the design space of its unfeasible solutions should be implemented for each use-case.
In 2D we were able to smoothly interpolate between designs by elastic deformation. Recent works have scaled this theory to three-dimensional objects (parametrised surfaces embedded in $\mathbb{R}^3$), [47]. While, sadly, we could not use these results in the course of this PhD due to a lack of time, we can envision a potential application: given two parts, one with desirable mechanical properties but not fabricable and the other with a poorer mechanical behaviour but manufacturable, a smooth interpolation between them could give the best of both without much computational effort. Another very promising avenue for research is close to what we showed in FIGURE 4.55: given the existing database of sound traditional assemblies, some being presented in CHAPTER 2, the exploration of the space obtained by interpolating between them would cheaply generate novel likely-to-be-sound 3D assemblies.

APPENDIX A

STEREOGRAPHIC PROJECTION

A.1 INTUITION

The stereographic projection is a conformal transformation mapping a sphere to a plane. It is defined and smooth on the entire sphere except at one particular point, the projection point. Geometrically speaking, a line is drawn between the projection point and any other point on the sphere. The intersection of this line and the plane define the projection of that point. In this manuscript, $\mathbb{R}^3$ is equipped with its usual cartesian frame $(\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z)$; the north pole refers to the point $\boldsymbol{e}_z = \begin{pmatrix} 0, 0, 1 \end{pmatrix}^T$. In this manuscript the projection point is defined as the north pole, as illustrated on FIGURE A.1.

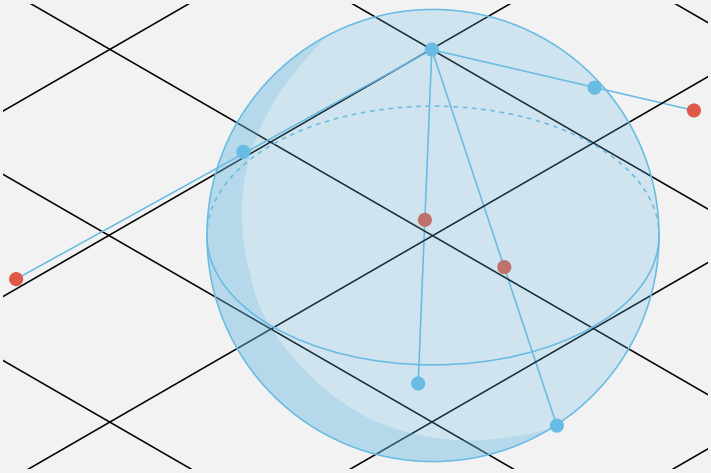


Figure A.1| The stereographic projection maps points on the sphere (in blue) to points in the plane (in red).

With such stereographic projection, points on the southern hemisphere are mapped inside the unit disk, the equator is mapped to itself, and the northern hemisphere is mapped to the rest of $\mathbb{R}^2$.

A.2 DEFINITION

Formally speaking:

- Let $\mathcal{S}^2$ be the unit sphere in $\mathbb{R}^3$:

$$\mathcal{S}^2 = \{\boldsymbol{x} \in \mathbb{R}^3, \, ||\boldsymbol{x}|| = 1\}$$

- Let:

$$\begin{array}{rccc} p & : & \mathcal{S}^2 & \longrightarrow & \mathbb{R}^2 \cup \{\infty\} \\ & & \begin{pmatrix} x \\ y \\ z \end{pmatrix} & \mapsto & \begin{pmatrix} \frac{x}{1-z} \\ \frac{y}{1-z} \end{pmatrix} \end{array}$$

  be the stereographic projection of the unit sphere from the north pole.

- Let:

$$\begin{array}{rccc} p^{-1} & : & \mathbb{R}^2 \cup \{\infty\} & \longrightarrow & \mathcal{S}^2 \\ & & \begin{pmatrix} X \\ Y \end{pmatrix} & \mapsto & \frac{1}{1+X^2+Y^2} \begin{pmatrix} 2X \\ 2Y \\ -1+X^2+Y^2 \end{pmatrix} \end{array}$$

  the inverse stereographic projection mapping the plane $\mathbb{R}^2$ onto the sphere $\mathcal{S}^2$.

A.3 PROPERTY

Property A.3.1. *The stereographic projection of a circle on the sphere is a circle or a line.*

Proof. Let $\mathcal{P}$, of equation $ax + by + cz + d = 0$, be a plane intersecting $\mathcal{S}^2$ and let $\mathcal{I}$ be the intersection. Let $\boldsymbol{X} = \begin{pmatrix} X \\ Y \end{pmatrix} \in \mathbb{R}^2$ be such that $p^{-1}(\boldsymbol{X}) \in \mathcal{I}$. By definition of the intersection $\mathcal{I}$ we have that $p^{-1}(\boldsymbol{X}) \in \mathcal{P}$. Hence $\boldsymbol{X}$ is solution of

$$\begin{aligned} & a\frac{2X}{1+X^2+Y^2} + b\frac{2Y}{1+X^2+Y^2} + c\frac{1-X^2-Y^2}{1+X^2+Y^2} + d = 0 \\ & 2aX + 2bY + c(1-X^2-Y^2) + d(1+X^2+Y^2) = 0 \\ & (d-c)(X^2+Y^2) + 2aX + 2bY + d + c = 0 \end{aligned}$$

If $(d-c) = 0$ then the above equation states that $\boldsymbol{X}$ lies on a line. Note that $d - c = 0 \iff -\boldsymbol{e}_z \in \mathcal{P}$ by plugging $-\boldsymbol{e}_z$ in the equation of the plane. Else, if $d \neq c$, then:

$$\begin{aligned} (d-c)(X^2+Y^2) + 2aX + 2bY + d + c = 0 &\iff X^2 + Y^2 + 2\frac{a}{d-c}X + 2\frac{b}{d-c} + \frac{d+c}{d-c} = 0 \\ &\iff (X + \frac{a}{d-c})^2 + (Y + \frac{b}{d-c})^2 + \frac{d+c}{d-c} - \frac{a^2+b^2}{(d-c)^2} \\ &\iff (X + \frac{a}{d-c})^2 + (Y + \frac{b}{d-c})^2 + \frac{d^2-a^2-b^2-c^2}{(d-c)^2} = 0 \end{aligned}$$

Since $\mathcal{P}$ intersects $\mathcal{S}^2$ of radius 1 we have that:

$$\frac{|d|}{\sqrt{a^2+b^2+c^2}} \leq 1$$

Hence:

$$\frac{a^2+b^2+c^2-d^2}{(d-c)^2} \geq 0$$

And thus, with $\alpha = -\frac{a}{d-c}$; $\beta = \frac{-b}{d-c}$ and $r = \sqrt{\frac{a^2+b^2+c^2-d^2}{(d-c)^2}}$ one has:

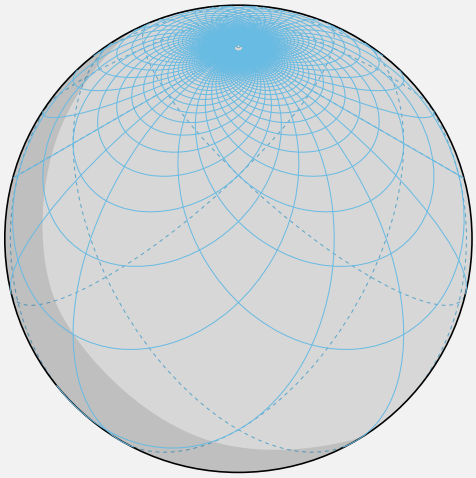


Figure A.2| Projection of a cartesian grid of edge length $\frac{1}{2}$.

$$(X + \frac{a}{d-c})^2 + (Y + \frac{b}{d-c})^2 + \frac{d^2 - a^2 - b^2 - c^2}{(d-c)^2} = 0 \Longleftrightarrow (X - \alpha)^2 + (Y - \beta)^2 - r^2 = 0$$

Which states that $\boldsymbol{X}$ lies on a circle of center (α, β) and radius r and the proof is complete. □

APPENDIX B

GRAPHS

Most of the contents of this appendix come from [10].

B.1 DEFINITIONS

At the most fundamental level a **graph** is a discrete mathematical structure aimed at representing *objects* such that some pairs are, in some sense, *related*. These objects are represented by **vertices** (often called **nodes**). A graph may be:

- **undirected**: the problem being modeled assumes that the relation between two nodes is reciprocal, for instance a cousin-cousin relation: if Bertrand is the cousin of Alice, then Alice is the cousin of Bertrand. The relation is called an **edge**.
- **directed**: the problem being modeled assumes a non reciprocity of the relationship between the nodes, for instance a parent-child relation: if Bertrand is the father of Alice, then Alice is not the mother of Bertrand. In such case the relation between two nodes is called an **arc**.

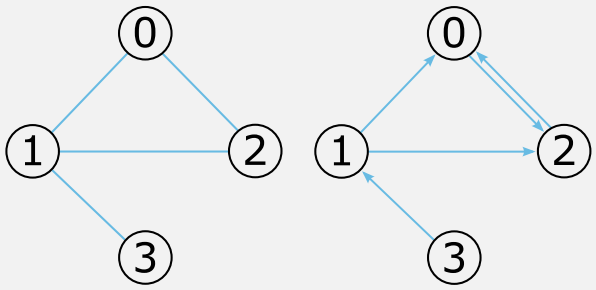


Figure B.1| Two graphs: vertices are depicted with numbered circles. Left: undirected graph, edges between vertices are represented by plain line segments. Right: directed graph, arcs between vertices are represented by arrows.

Formally, if v_p and v_q are two related vertices, $e = (v_p, v_q)$ is an edge when the order of the pair does not matter, an arc otherwise. Often a graph is called $G = (V, E)$ with V the set of vertices and E the set of edges/arcs: $e = (v_p, v_q) \in E, v_p, v_q \in V$. From now on, only directed graphs will be considered as an undirected graph can simply be obtained from a directed one by "forgetting" the arcs' orientation. A vertex in V of index p will be denoted v_p and the i^{th} arc of E from vertex v_p to vertex v_q will be denoted $e_i = (v_p, v_q)$.

B.2 CONNECTIVITY

B.2.1 CHAIN AND CYCLE

Two arcs/edges are **adjacent** when they have a vertex in common.
A **chain** of cardinality q is a sequence of q arcs $[e_1, e_2, ..., e_q]$ such as two consecutive arcs are adjacent. A chain disregards the orientation of the arcs. For instance, on FIGURE B.2 the sequence $[e_1, e_2, e_3, e_4, e_5]$ is a chain of cardinality 5 as $e_1 = (1, 0)$ shares a common vertex with $e_2 = (1, 2)$ which shares a common vertex with $e_3 = (3, 1)$ etc.

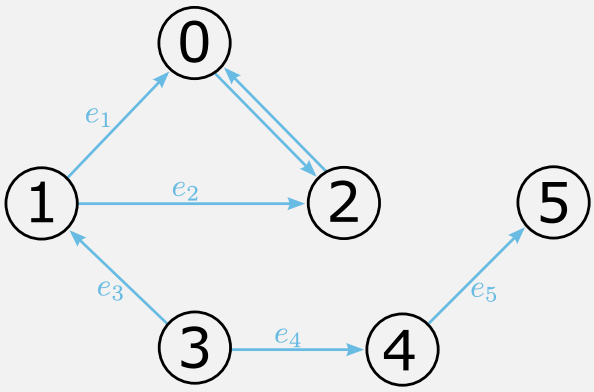


Figure B.2| The sequence of arcs $[e_1, e_2, e_3, e_4, e_5]$ is a chain of cardinality 5.

A **cycle** is a chain whose end vertices coincide.

B.2.2 PATH AND CIRCUIT

A **path** is a chain whose arcs are all directed in the same way: $[e_1, e_2, ..., e_q]$ is a path of cardinality q with the end vertex of e_i being equals to the start vertex of e_{i+1} : $e_i = (v_p, v_q)$ and $e_{i+1} = (v_q, v_r)$. For instance on FIGURE B.2, $[e_4, e_5]$ is a path of cardinality 2.
A circuit is a path whose end vertices coincide.

B.2.3 CONNECTED GRAPH

A graph (be it directed or undirected) is **connected** when a chain can be built between any pair of distinct vertices. Intuitively it means that one can travel between any two distinct vertices by walking on an edge (an arc deprived of its orientation).
The relation

$$v_p \equiv v_q = \begin{cases} \text{either } v_p = v_q \\ \text{or there exists a chain between } v_p \text{ and } v_q \end{cases}$$

is an equivalence relation:

- reflexive: $v_p \equiv v_p$
- symmetric: $v_p \equiv v_q \implies v_q \equiv v_p$
- transitive: $v_p \equiv v_q$ and $v_q \equiv v_r \implies v_p \equiv v_r$

The equivalence class that such relation induces on V partitions the set of vertices in **connected components**. Worded differently a connected component is a set of vertices from V such as there exists a chain made of arcs from E between any two nodes of that set. For instance FIGURE B.3 shows a graph with two connected components composed respectively of vertices [0, 1, 2, 3] and [4, 5].

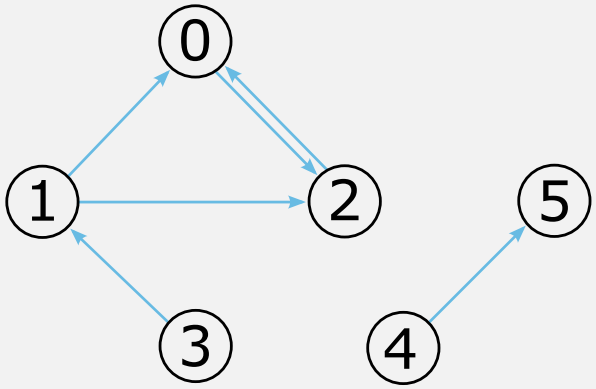


Figure B.3| A graph with two connected components

A graph with a single connected component is therefore a connected graph.

B.2.4 STRONGLY CONNECTED GRAPH

A directed graph (the concept of strong connectivity does not apply to undirected graphs) is **strongly connected** when a circuit going through any pair of distinct vertices exists. Intuitively it means that one can travel back and forth between any two distinct vertices while following the orientation of the arcs.

The relation

$$v_p \equiv v_q = \begin{cases} \text{either } v_p = v_q \\ \text{or there exists a circuit going through } v_p \text{ and } v_q. \end{cases}$$

is an equivalence relation:

- reflexive: $v_p \equiv v_p$
- symmetric: $v_p \equiv v_q \implies v_q \equiv v_p$
- transitive: $v_p \equiv v_q$ and $v_q \equiv v_r \implies v_p \equiv v_r$

The equivalence class that such relation induces on V partitions the set of vertices in **strongly connected components**. Worded differently a strongly connected component is a set of vertices from V such as there exist two paths of opposite orientation (i.e. a circuit) made of arcs from E between any two nodes of that set. For instance FIGURE B.4 shows a graph with four strongly connected components: it is possible to travel back and forth between any two nodes of the set [0, 1, 2, 3] or of the set [4, 5]. Nodes [6] and [7] are two strongly connected components reduced to single nodes.

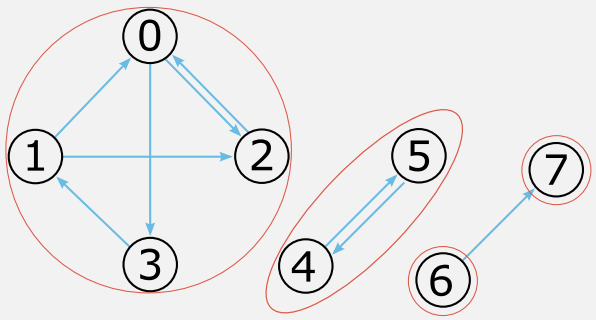


Figure B.4| A graph whose four strongly connected components are circled in red.

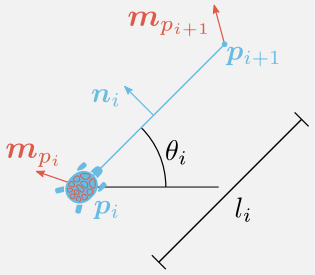
A graph with a single connected component is therefore a strongly connected graph.

APPENDIX C

INTERVALS - TURTLE IN ROTATION

C.1 WARM UP

Notations: let $\boldsymbol{p}_i = (x_i, y_i)^T \in \mathbb{R}^2$ be the current position of the `Turtle`, and let θ_i and l_i be the orientation and length of the segment drawn by the `Turtle` between $\boldsymbol{p}_i$ and its next position $\boldsymbol{p}_{i+1} = (x_{i+1}, y_{i+1})^T = \boldsymbol{p}_i + l_i(\cos\theta_i, \sin\theta_i)^T$, as shown on the inset. The goal of this section is to find in which set shall θ_i and l_i be chosen so that the line segment $[\boldsymbol{p}_i, \boldsymbol{p}_{i+1}]$ may obey a rotation around a given centre point $\boldsymbol{x} = (x, y)^T \in \mathbb{R}^2$. Assuming $\boldsymbol{p}_i \neq \boldsymbol{x}$ (a very reasonable assumption stating that the `Turtle` is not exactly on the centre of rotation) we can define the instantaneous directions of motion of point $\boldsymbol{p}_i$, $\boldsymbol{p}_{i+1}$ with respect to $\boldsymbol{x}$, $\boldsymbol{m}(\boldsymbol{p}_i, \boldsymbol{x})$ and $\boldsymbol{m}(\boldsymbol{p}_{i+1}, \boldsymbol{x})$, abbreviated $\boldsymbol{m}_{p_i}$ and $\boldsymbol{m}_{p_{i+1}}$.



Referring to SYSTEM (3.12), one has (with $\boldsymbol{n}_i = (-\sin\theta_i, \cos\theta_i)^T$ the unit normal vector of segment $[\boldsymbol{p}_i, \boldsymbol{p}_{i+1}]$)

$$[\boldsymbol{p}_i, \boldsymbol{p}_{i+1}] \text{ obeys a rotation around } \boldsymbol{x} \iff \begin{cases} \boldsymbol{n}_i \cdot \boldsymbol{m}_{p_i} \star 0 \\ \boldsymbol{n}_i \cdot \boldsymbol{m}_{p_{i+1}} \star 0 \end{cases} \tag{C.1}$$

With $\star$ standing for $\geq$ if the rotation is counterclockwise, $\leq$ if clockwise. Let us introduce a sign $s = +1$ if the rotation is counterclockwise, -1 otherwise. Then SYSTEM (C.1) is rewritten:

$$[\boldsymbol{p}_i, \boldsymbol{p}_{i+1}] \text{ obeys a rotation around } \boldsymbol{x} \iff \begin{cases} s\boldsymbol{n}_i \cdot \boldsymbol{m}_{p_i} \geq 0 \\ s\boldsymbol{n}_i \cdot \boldsymbol{m}_{p_{i+1}} \geq 0 \end{cases} \tag{C.2}$$

$$\iff \begin{cases} s \begin{pmatrix} -\sin\theta_i \\ \cos\theta_i \end{pmatrix} \cdot \boldsymbol{m}_{p_i} \geq 0 \\ s \begin{pmatrix} -\sin\theta_i \\ \cos\theta_i \end{pmatrix} \cdot \left(\boldsymbol{m}_{p_i} + l_i \begin{pmatrix} -\sin\theta_i \\ \cos\theta_i \end{pmatrix} \right) \geq 0 \end{cases} \tag{C.3}$$

With $\boldsymbol{m}_{p_{i+1}} = \boldsymbol{m}_{p_i} + l_i(-\sin\theta_i, \cos\theta_i)^T$. For notational convenience let, in this section, $\Delta x = x_i - x$ and $\Delta y = y_i - y$ such that $\boldsymbol{m}_{p_i} = \begin{pmatrix} -\Delta y, \Delta x \end{pmatrix}^T$. With $t = \tan\frac{\theta_i}{2}$, recall the double-angle formulae:

$\cos\theta_i = \frac{1-t^2}{1+t^2}$ and $\sin\theta_i = \frac{2t}{1+t^2}$. Let us consider the first inequality:

$$\begin{aligned} s\begin{pmatrix}-\sin\theta_i\\ \cos\theta_i\end{pmatrix}\cdot\boldsymbol{m}_{p_i} \geq 0 &\iff s\begin{pmatrix}-\sin\theta_i\\ \cos\theta_i\end{pmatrix}\cdot\begin{pmatrix}-\Delta y\\ \Delta x\end{pmatrix}\geq 0\\ &\iff s\left(\Delta x\cos\theta_i + \Delta y\sin\theta_i\right)\geq 0\\ &\iff s\left(\Delta x\frac{1-t^2}{1+t^2}+\Delta y\frac{2t}{1+t^2}\right)\geq 0\\ &\iff s\left(-\Delta x t^2 + 2\Delta y t + \Delta x\right)\geq 0 \end{aligned}$$

Where $\theta_i \in]-\pi,\pi[$ for the inequalities to be defined. Let

$$\begin{array}{rccc} Q: &]-\pi,\pi[& \to & \mathbb{R}\\ & \theta & \mapsto & -\Delta x\tan^2\frac{\theta}{2}+2\Delta y\tan\frac{\theta}{2}+\Delta x \end{array}$$

And we have the equivalence:

$$s\boldsymbol{n}_i\cdot\boldsymbol{m}_{p_i}\geq 0 \iff sQ(\theta_i)\geq 0 \tag{C.4}$$

Two cases must be studied:

- If $\Delta x = 0$, $Q:\theta\mapsto 2\Delta y\tan\frac{\theta}{2}$. Hence:

$$s\boldsymbol{n}_i\cdot\boldsymbol{m}_{p_i}\geq 0 \iff \begin{cases}\theta_i\in[0,\pi[& \text{if } s\Delta y>0\\ \theta_i\in]-\pi,0] & \text{if } s\Delta y<0\end{cases} \tag{C.5}$$

- If $\Delta x\neq 0$ then $Q:\theta\mapsto Q(\theta)$ is a polynomial of degree 2 in $\tan\frac{\theta}{2}$. Its discriminant is

$$\Delta = 4||\boldsymbol{x}_i-\boldsymbol{x}||^2>0 \tag{C.6}$$

  Hence Q has two real roots:

$$\theta_{1,2} = 2\arctan\frac{\Delta y\pm||\boldsymbol{x}_i-\boldsymbol{x}||}{\Delta x} \tag{C.7}$$

Assuming $\theta_1<\theta_2$ (otherwise switch them):

$$s\boldsymbol{n}_i\cdot\boldsymbol{m}_{p_i}\geq 0 \iff \begin{cases}\theta\in[\theta_1,\theta_2] & \text{if } s\Delta x>0\\ \theta_i\in]-\pi,\theta_1]\cup[\theta_2,\pi[& \text{if } s\Delta x<0\end{cases} \tag{C.8}$$

Note the pleasant property of this first inequality: the segment length l_i and the segment orientation θ_i are independent of each other.
As for the second inequality, with again $\theta_i\in]-\pi,\pi[$:

$$\begin{aligned} s\begin{pmatrix}-\sin\theta_i\\ \cos\theta_i\end{pmatrix}\cdot\left(\boldsymbol{m}_{p_i}+l_i\begin{pmatrix}-\sin\theta_i\\ \cos\theta_i\end{pmatrix}\right)\geq 0 &\iff s\left(\sin\theta_i\Delta y + l_i\sin^2\theta_i+\cos\theta_i\Delta x+l_i\cos^2\theta_i\right)\\ &\iff s\left(\cos\theta_i\Delta x+\sin\theta_i\Delta y+l_i\right)\\ &\iff s\left(\frac{1-t^2}{1+t^2}\Delta x+\frac{2t}{1+t^2}\Delta y+l_i\right)\\ &\iff s\left((l_i-\Delta x)t^2+2\Delta y t+\Delta x+l_i\right) \end{aligned}$$

Let

$$\begin{array}{rccc} P: &]-\pi,\pi[\times[l_{min},l_{max}] & \to & \mathbb{R} \\ & (\theta,l) & \mapsto & (l-\Delta x)\tan^2\frac{\theta}{2}+2\Delta y\tan\frac{\theta}{2}+l+\Delta x \end{array}$$

And we have the equivalence:

$$s\boldsymbol{n}_i\cdot\boldsymbol{m}_{p_{i+1}}\geq 0 \iff sP(\theta_i,l_i)\geq 0 \tag{C.9}$$

Several cases must be studied:

- If $l_i=\Delta x\geq 0$, $\tilde{P}:\theta\mapsto P(\theta,l_i)$ is either constant or linear in $\tan\frac{\theta}{2}$
  - If $\Delta y=0$ then $\forall\theta\in]-\pi,\pi[\ P(\theta,l_i)=l_i+\Delta x=2l_i>0$. P is constant and positive, hence:

$$s\boldsymbol{n}_i\cdot\boldsymbol{m}_{p_{i+1}}\geq 0\Leftrightarrow s=+1 \tag{C.10}$$

  - If $\Delta y\neq 0$, $\tilde{P}:\theta\mapsto P(\theta)$ is linear in $\tan\frac{\theta}{2}$:

$$s\boldsymbol{n}_i\cdot\boldsymbol{m}_{p_{i+1}}\geq 0\Leftrightarrow\begin{cases}\theta\leq -2\arctan\frac{l_i}{\Delta y} & \text{if } s\Delta y<0\\ \theta\geq -2\arctan\frac{l_i}{\Delta y} & \text{if } s\Delta y>0\end{cases} \tag{C.11}$$

- If $l_i\neq\Delta x$, $\tilde{P}:\theta\mapsto P(\theta,l_i)$ is a second degree polynomial in $\tan\frac{\theta}{2}$ whose discriminant is:

$$\Delta=4\left(\Delta y^2+\Delta x^2-l_i^2\right) \tag{C.12}$$
$$=4\left(||\boldsymbol{x}_i-\boldsymbol{x}||^2-l_i^2\right) \tag{C.13}$$

  Hence:

$$\Delta\geq 0\Leftrightarrow||\boldsymbol{x}_i-\boldsymbol{x}||\geq l_i \tag{C.14}$$

  - If $||\boldsymbol{x}_i-\boldsymbol{x}||<l_i$:
    P has no real roots but we can derive the following equations:

$$l_i>||\boldsymbol{x}_i-\boldsymbol{x}|| \tag{C.15}$$
$$>\sqrt{\Delta x^2+\Delta y^2} \tag{C.16}$$
$$>|\Delta x| \tag{C.17}$$
$$>\Delta x \tag{C.18}$$

    Hence, $\forall\theta\in]-\pi,\pi[, ||\boldsymbol{x}_i-\boldsymbol{x}||<l_i\implies l_i-\Delta x>0\implies\tilde{P}(\theta)>0$, thus

$$s\boldsymbol{n}_i\cdot\boldsymbol{m}_{p_{i+1}}\geq 0\Leftrightarrow s=+1 \tag{C.19}$$

  - If $||\boldsymbol{x}_i-\boldsymbol{x}||\geq l_i$, $\tilde{P}$ admits two real roots θ_3 and θ_4 such that:

$$\theta_{3,4}=2\arctan\frac{-\Delta y\pm\sqrt{||\boldsymbol{x}_i-\boldsymbol{x}||^2-l_i^2}}{l_i-\Delta x} \tag{C.20}$$

    And, assuming $\theta_3<\theta_4$ (otherwise switch them), we have the equivalence:

$$s\boldsymbol{n}_i\cdot\boldsymbol{m}_{p_{i+1}}\geq 0\Leftrightarrow\begin{cases}\theta_i\in[\theta_3,\theta_4] & \text{if } s(l_i-\Delta x)<0\\ \theta_i\in]-\pi,\theta_3]\cup[\theta_4,\pi] & \text{if } s(l_i-\Delta x)>0\end{cases} \tag{C.21}$$

Summing up: When the right hand side is defined let:

$$
\begin{aligned}
\theta_0 &= 2\arctan\frac{-l_i}{\Delta y}\\
\theta_{12}^{\ominus} &= 2\arctan\frac{\Delta y - ||\boldsymbol{x}_i - \boldsymbol{x}||}{\Delta x}\\
\theta_{12}^{\oplus} &= 2\arctan\frac{\Delta y + ||\boldsymbol{x}_i - \boldsymbol{x}||}{\Delta x}\\
\theta_{34}^{\ominus} &= 2\arctan\frac{-\Delta y - \sqrt{||\boldsymbol{x}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x}\\
\theta_{34}^{\oplus} &= 2\arctan\frac{-\Delta y + \sqrt{||\boldsymbol{x}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x}\\
\theta_1 &= \min(\theta_{12}^{\ominus}, \theta_{12}^{\oplus})\\
\theta_2 &= \max(\theta_{12}^{\ominus}, \theta_{12}^{\oplus})\\
\theta_3 &= \min(\theta_{34}^{\ominus}, \theta_{34}^{\oplus})\\
\theta_4 &= \max(\theta_{34}^{\ominus}, \theta_{34}^{\oplus})
\end{aligned}
$$

C.2 A PRÉVERT-STYLE ENUMERATION OF CASES

Following the definition of the critical angles $\theta_{0,1,2,3,4}$ in SECTION C.1 regarding the `Turtle` in rotation, the crossing of the solution sets of the two inequalities of the system

$$
\begin{cases}
s\boldsymbol{n}_i \cdot \boldsymbol{m}_{p_i} \geq 0\\
s\boldsymbol{n}_i \cdot \boldsymbol{m}_{p_{i+1}} \geq 0
\end{cases}
$$

leads to a Prévert-style enumeration of cases.

1. If $l_i = \Delta x$:

(a) If $\Delta y = 0$:

- If $s = -1$: $\theta \in \emptyset$, no solution.
- If $s = +1$: $\theta_i \in]-\pi, \pi[\cap [\theta_1, \theta_2] \implies \theta_i \in [\theta_1, \theta_2]$

(b) If $\Delta y \neq 0$:

- If $s\Delta y < 0$:
  - If $s\Delta x > 0$: $\theta_i \in]-\pi, \theta_0] \cap [\theta_1, \theta_2]$:
    - If $\theta_0 < \theta_1 : \theta_i \in \emptyset$, no solution.
    - If $\theta_1 \leq \theta_0$: $\theta_i \in [\theta_1, \min(\theta_0, \theta_2)]$
  - If $s\Delta x < 0$: $\theta_i \in]-\pi, \theta_0] \cap (]-\pi, \theta_1] \cup [\theta_2, \pi[)$:
    - If $\theta_0 < \theta_2$: $\theta_i \in]-\pi, \min(\theta_0, \theta_1)]$
    - If $\theta_2 \leq \theta_0$: $\theta_i \in]-\pi, \theta_1] \cup [\theta_2, \theta_0]$
- If $s\Delta y > 0$:
  - If $s\Delta x > 0$: $\theta_i \in [\theta_0, \pi[\cap [\theta_1, \theta_2]$:
    - If $\theta_2 < \theta_0$: $\theta_i \in \emptyset$, no solution.
    - If $\theta_0 \leq \theta_2$: $\theta_i \in [\max(\theta_0, \theta_1), \theta_2]$
  - If $s\Delta x < 0$: $\theta_i \in [\theta_0, \pi[\cap (]-\pi, \theta_1] \cup [\theta_2, \pi[)$:
    - If $\theta_0 \leq \theta_1$: $\theta_i \in [\theta_0, \theta_1] \cup [\theta_2, \pi[$

- If $\theta_1 < \theta_0$: $\theta_i \in [\max(\theta_0, \theta_2), \pi[$

2. Else if $||\boldsymbol{p}_i - \boldsymbol{x}|| < l_i$:

(a) If $s = -1$: $\theta_i \in \emptyset$, no solution.

(b) If $s = +1$:

- If $\Delta x = 0$:
  - If $\Delta y < 0$: $\theta_i \in]-\pi, \pi[\cap]-\pi, 0] \implies \theta_i \in]-\pi, 0]$
  - If $\Delta y > 0$: $\theta_i \in]-\pi, \pi[\cap [0, \pi[\implies \theta_i \in [0, \pi[$
- If $\Delta x > 0$: $\theta_i \in]-\pi, \pi[\cap [\theta_1, \theta_2] \implies \theta_i \in [\theta_1, \theta_2]$
- If $\Delta x < 0$: $\theta_i \in]-\pi, \pi[\cap (]-\pi, \theta_1] \cup [\theta_2, \pi[) \implies \theta_i \in]-\pi, \theta_1] \cup [\theta_2, \pi[$

3. Else if $||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i$:

(a) If $\Delta x = 0$:

- If $s(l_i - \Delta x) < 0$:
  - If $s\Delta y < 0$: $\theta_i \in [\theta_3, \theta_4] \cap]-\pi, 0]$:
    - If $0 < \theta_3$: $\theta_i \in \emptyset$, no solution.
    - If $\theta_3 \leq 0$: $\theta_i \in [\theta_3, \min(0, \theta_4)]$
  - If $s\Delta y > 0$: $\theta_i \in [\theta_3, \theta_4] \cap [0, \pi[$
    - If $\theta_4 < 0$: $\theta_i \in \emptyset$, no solution.
    - If $0 \leq \theta_4$: $\theta_i \in [\max(0, \theta_3), \theta_4]$
- If $s(l_i - \Delta x) > 0$:
  - If $s\Delta y < 0$: $\theta_i \in (]-\pi, \theta_3] \cup [\theta_4, \pi[) \cap]-\pi, 0]$:
    - If $0 < \theta_4$: $\theta_i \in]-\pi, \min(0, \theta_3)]$
    - If $\theta_4 \leq 0$: $\theta_i \in]-\pi, \theta_3] \cup [\theta_4, 0]$
  - If $s\Delta y > 0$: $\theta_i \in (]-\pi, \theta_3] \cup [\theta_4, \pi[) \cap [0, \pi[$
    - If $0 \leq \theta_3$: $\theta_i \in [0, \theta_3] \cup [\theta_4, \pi[$
    - If $\theta_3 < 0$, $\theta_i \in [\max(0, \theta_4), \pi[$

(b) If $\Delta x \neq 0$:

- If $s(l_i - \Delta x) < 0$:
  - If $s\Delta x > 0$: $\theta_i \in [\theta_3, \theta_4] \cap [\theta_1, \theta_2]$:
    - If $\theta_4 < \theta_1$ or $\theta_2 < \theta_3$: $\theta_i \in \emptyset$, no solution.
    - If $\theta_1 \leq \theta_4$ and $\theta_3 \leq \theta_2$: $\theta_i \in [\max(\theta_3, \theta_1), \min(\theta_4, \theta_2)]$
  - If $s\Delta x < 0$: $\theta_i \in [\theta_3, \theta_4] \cap (]-\pi, \theta_1] \cup [\theta_2, \pi[)$:
    - If $\theta_3 \leq \theta_1$ and $\theta_4 < \theta_2$: $\theta_i \in [\theta_3, \min(\theta_4, \theta_1)]$
    - If $\theta_3 \leq \theta_1$ and $\theta_2 \leq \theta_4$: $\theta_i \in [\theta_3, \theta_1] \cup [\theta_2, \theta_4]$
    - If $\theta_1 < \theta_3$ and $\theta_4 < \theta_2$: $\theta_i \in \emptyset$, no solution.
    - If $\theta_1 < \theta_3$ and $\theta_2 \leq \theta_4$: $\theta_i \in [\max(\theta_3, \theta_2), \theta_4]$
- If $s(l_i - \Delta x) > 0$:
  - If $s\Delta x > 0$: $\theta_i \in [\theta_1, \theta_2] \cap (]-\pi, \theta_3] \cup [\theta_4, \pi[)$:
    - If $\theta_1 \leq \theta_3$ and $\theta_2 < \theta_4$: $\theta_i \in [\theta_1, \min(\theta_2, \theta_3)]$
    - If $\theta_1 \leq \theta_3$ and $\theta_4 \leq \theta_2$: $\theta_i \in [\theta_1, \theta_3] \cup [\theta_4, \theta_2]$
    - If $\theta_3 < \theta_1$ and $\theta_2 < \theta_4$: $\theta_i \in \emptyset$, no solution.
    - If $\theta_3 < \theta_1$ and $\theta_4 \leq \theta_2$: $\theta_i \in [\max(\theta_1, \theta_4), \theta_2]$
  - If $s\Delta x < 0$: $\theta_i \in (]-\pi, \theta_3] \cup [\theta_4, \pi[) \cap (]-\pi, \theta_1] \cup [\theta_2, \pi[)$
    - If $\max(\theta_3, \theta_1) \leq \min(\theta_4, \theta_2)$: $\theta_i \in]-\pi, \min(\theta_3, \theta_1)] \cup [\max(\theta_4, \theta_2), \pi[$

* If $\max(\theta_3, \theta_1) > \min(\theta_4, \theta_2)$: $\theta_i \in]-\pi, \min(\theta_1, \theta_3)] \cup [\min(\theta_4, \theta_2), \max(\theta_1, \theta_3)] \cup [\max(\theta_4, \theta_2), \pi[$

C.3 WHERE TO CHOOSE θ ?

This appendix refines the above enumeration and proves also partial orderings of the θ_i that are helpful to justify the fact that the `Turtle` is drawn to or repelled by $\boldsymbol{x}$ depending on the value of s.

- If $\Delta x = l_i$
  - If $\Delta y \neq 0$
    1. **If $s\boldsymbol{\Delta y} = \mathbf{0}$**
       No change are made with regards to the enumeration provided above.
    2. **If $\boldsymbol{\Delta y} \neq \mathbf{0}$**

       Prerequisite:
       For $a \in \mathbb{R}$ let:

$$f_a : \begin{array}{ccc} \mathbb{R} & \to & \mathbb{R} \\ x & \mapsto & x^2 + a^2 - x\sqrt{x^2 + a^2} \end{array}$$

$$g_a : \begin{array}{ccc} \mathbb{R} & \to & \mathbb{R} \\ x & \mapsto & x^2 + a^2 + x\sqrt{x^2 + a^2} \end{array}$$

We show that for any $a \in \mathbb{R}$ both f_a and g_a are positive on $\mathbb{R}$. f_a is trivially positive on $\mathbb{R}^-$ and g_a on $\mathbb{R}^+$. Also we remark that proving $f_a \geq 0$ on $\mathbb{R}^+$ is the same as proving $g_a \geq 0$ on $\mathbb{R}^-$. So it is enough to prove $f_a \geq 0$ on $\mathbb{R}^+$. For all $x \geq 0$ we have the equivalence:

$$\begin{aligned} f_a(x) \geq 0 &\Leftrightarrow \underbrace{x^2 + a^2}_{\geq 0} \geq \underbrace{x\sqrt{x^2 + a^2}}_{\geq 0} \\ &\Leftrightarrow (x^2 + a^2)^2 \geq \left(x\sqrt{x^2 + a^2}\right)^2 \\ &\Leftrightarrow a^2(a^2 + x^2) \geq 0 \text{ always true} \end{aligned}$$

Hence, by equivalence, as this result is true for any $x \in \mathbb{R}^+$ we have shown that $f_a \geq 0$ on $\mathbb{R}^+$ and by consequence, as outlined above, $g_a \geq 0$ on $\mathbb{R}^-$. Thus, we have successfully shown that both f_a and g_a are positive on $\mathbb{R}$ for any $a \in \mathbb{R}$.

Let:

$$\begin{aligned} a_0 &= \frac{-l_i}{\Delta y} \\ a_1 &= \frac{\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||}{l_i} \\ a_2 &= \frac{\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||}{l_i} \end{aligned}$$

such that $\theta_{0,1,2} = 2\arctan(a_{0,1,2})$ and $\theta_1 \leq \theta_2$. We prove that:

$$\Delta y < 0 \implies \theta_1 \leq \theta_2 \leq \theta_0$$
$$\Delta y > 0 \implies \theta_0 \leq \theta_1 \leq \theta_2$$

Since $x \mapsto 2\arctan(x)$ is increasing on $\mathbb{R}$ proving an ordering of the θ_i boils down to proving the same ordering for the a_i.

(a) $\Delta y < 0 \implies \theta_1 \leq \theta_2 \leq \theta_0$

Proof. As by definition of θ_1 and θ_2 we have $\theta_1 \leq \theta_2$, it is enough to prove $\Delta y < 0 \implies \theta_2 \leq \theta_0$.
Let $\Delta y \in \mathbb{R}^{-*}$. By equivalence:

$$\begin{aligned}\theta_2 \leq \theta_0 &\Leftrightarrow \frac{\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||}{l_i} \leq \frac{-l_i}{\Delta y} \\ &\Leftrightarrow \Delta y^2 + \Delta y||\boldsymbol{p}_i - \boldsymbol{x}|| \geq -l_i^2 \\ &\Leftrightarrow \Delta y^2 + l_i^2 + \Delta y\sqrt{\Delta y^2 + l_i^2} \geq 0 \\ &\Leftrightarrow g_{l_i}(\Delta y) \geq 0\end{aligned}$$

And we know that $g_{l_i} \geq 0$ on $\mathbb{R}$ so the latest inequality is true and by equivalence so is $\theta_2 \leq \theta_0$.
Thus:

$$\boxed{\Delta y < 0 \implies \theta_1 \leq \theta_2 \leq \theta_0}$$

□

(b) $\Delta y > 0 \implies \theta_0 \leq \theta_1 \leq \theta_2$

Proof. As by definition of θ_1 and θ_2 we have $\theta_1 \leq \theta_2$, it is enough to prove $\Delta y > 0 \implies \theta_0 \leq \theta_1$.
Let $\Delta y \in \mathbb{R}^{+*}$. By equivalence:

$$\begin{aligned}\theta_0 \leq \theta_1 &\Leftrightarrow \frac{-l_i}{\Delta y} \leq \frac{\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||}{l_i} \\ &\Leftrightarrow -l_i^2 \leq \Delta y^2 - \Delta y||\boldsymbol{p}_i - \boldsymbol{x}|| \\ &\Leftrightarrow \Delta y^2 + l_i^2 - \Delta y\sqrt{\Delta y^2 + l_i^2} \geq 0 \\ &\Leftrightarrow f_{l_i}(\Delta y) \geq 0\end{aligned}$$

And we know that $f_{l_i} \geq 0$ on $\mathbb{R}$ so the latest inequality is true and by equivalence so is $\theta_0 \leq \theta_1$.
Thus:

$$\boxed{\Delta y > 0 \implies \theta_0 \leq \theta_1 \leq \theta_2}$$

□

* If $s\Delta y > 0$

· If $s\Delta x < 0$
Since $\Delta x = l_i > 0$, $s\Delta x < 0 \implies s = -1$ and $s\Delta y > 0 \implies \Delta y < 0$.
Hence we know that $\theta_1 \leq \theta_2 \leq \theta_0$ and, referring to the enumeration provided

above, in such case only one of the two assertions may happen and the `Turtle` must choose θ such as:

$$\theta \in [\theta_0, \pi]$$

- If $s\Delta x > 0$
  Then $s = +1$ and $\Delta y > 0$. We know that $\theta_0 \leq \theta_1 \leq \theta_2$. Hence the `Turtle` chooses

$$\theta \in [\theta_1, \theta_2]$$

* If $s\Delta y < 0$

  - If $s\Delta x < 0$
    $s = -1$ and $\Delta y > 0$; so $\theta_0 \leq \theta_1 \leq \theta_2$ and the `Turtle` chooses:

$$\theta \in [-\pi, \theta_0]$$

  - If $s\Delta x > 0$
    $s = +1$ and $\Delta y < 0$; so $\theta_1 \leq \theta_2 \leq \theta_0$ and the `Turtle` chooses:

$$\theta \in [\theta_1, \theta_2]$$

- $||\boldsymbol{p}_i - \boldsymbol{x}|| < l_i$ No change are made with regards to the enumeration.
- $||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i$

  – If $\Delta x = 0$, let

$$a_3 = \frac{-\Delta y - \sqrt{\Delta y^2 - l_i^2}}{l_i} \tag{C.22}$$

$$a_4 = \frac{-\Delta y + \sqrt{\Delta y^2 - l_i^2}}{l_i} \tag{C.23}$$

  such that for $i \in [\![3, 4]\!]$ $\theta_i = 2\arctan(a_i)$ and $\theta_3 \leq \theta_4$. It is easy to show that:

$$\Delta y > 0 \implies \theta_3 \leq \theta_4 < 0$$
$$\Delta y < 0 \implies 0 < \theta_3 \leq \theta_4$$

  Hence the enumeration can be greatly simplified into:

  * If $s = -1$

$$\theta \in [\theta_3, \theta_4]$$

  * If $s = +1$

$$\theta \in \begin{cases} [-\pi, 0] \text{ if } \Delta y < 0 \\ [0, \pi] \text{ if } \Delta y > 0 \end{cases}$$

  – If $\Delta x \neq 0$
  Three cases may happen when $||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i$, $\Delta x \neq 0$ and $\Delta x \neq l_i$.

  1. Either $\Delta x < 0$; in which case we prove:

$$\theta_1 < \theta_3 \leq \theta_4 < \theta_2 \tag{C.24}$$

2. Or $0 < \Delta x < l_i$; we show that:

$$\begin{cases} \text{Either } \theta_3 \leq \theta_4 \leq \theta_1 \leq \theta_2 \\ \text{Or } \theta_1 \leq \theta_2 \leq \theta_3 \leq \theta_4 \end{cases} \tag{C.25}$$

3. Or $l_i < \Delta x$; we prove that:

$$\theta_3 < \theta_1 \leq \theta_2 < \theta_4 \tag{C.26}$$

In the following for $i \in [\![1, 4]\!]$ we introduce an $a_i \in \mathbb{R}$ such that $\theta_i = 2\arctan(a_i)$. Since $x \mapsto 2\arctan(x)$ is strictly increasing on $\mathbb{R}$ proving an ordering of the θ_i boils down to proving the same ordering for the a_i.

1. **If $\boldsymbol{\Delta x < 0}$**
Then, for $\Delta x < 0; \Delta y \in \mathbb{R}$ such that $\Delta x^2 + \Delta y^2 \geq l_i^2$ let:

$$\begin{aligned} a_1 &= \frac{\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x} \\ a_2 &= \frac{\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x} \\ a_3 &= \frac{-\Delta y - \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x} \\ a_4 &= \frac{-\Delta y + \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x} \end{aligned}$$

such that for $i \in [\![1, 4]\!]$ $\theta_i = 2\arctan(a_i)$ and $\theta_3 \leq \theta_4$ and $\theta_1 \leq \theta_2$. We wish to prove

$$a_1 < a_3 \leq a_4 < a_2 \tag{C.27}$$

Since by definition $a_3 \leq a_4$ we just need to show that $a_1 < a_3$ and $a_4 < a_2$.

(a) $a_1 < a_3$:

Proof. We are going to reason by equivalence and show that $a_1 < a_3$ is equivalent to a partial ordering that is always true, and hence that $a_1 < a_3$ is true:

$$\begin{aligned} a_1 < a_3 &\iff \frac{\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x} < \frac{-\Delta y - \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x} \\ &\iff -\Delta y \Delta x - \Delta x\sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2} < (\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||)(l_i - \Delta x) \\ &\iff -\Delta x\sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2} < (\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||)l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||\Delta x \\ &\iff \Delta x||\boldsymbol{p}_i - \boldsymbol{x}|| - \Delta x\sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2} < (\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||)l_i \\ &\iff \underbrace{\Delta x}_{<0} \underbrace{||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \sqrt{1 - \left(\frac{l_i}{||\boldsymbol{p}_i - \boldsymbol{x}||}\right)^2}\right)}_{\geq 0} < \underbrace{(\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||)l_i}_{>0} \end{aligned}$$

As outlined, on the above inequality the left hand term is negative while the right hand term is positive. Hence this inequality is always true and by equivalence so is $a_1 < a_3$. We have thus shown:

$$\boxed{\theta_1 < \theta_3}$$

□

(b) $a_4 < a_2$:

Proof. We prove this inequality by equivalence:

$$\begin{aligned}
a_4 < a_2 &\iff \frac{-\Delta y + \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x} < \frac{\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x} \\
&\iff (l_i - \Delta x)(\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||) < -\Delta y \Delta x + \Delta x ||\boldsymbol{p}_i - \boldsymbol{x}|| \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2} \\
&\iff (\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||) l_i + ||\boldsymbol{p}_i - \boldsymbol{x}|| \Delta x < \Delta x \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2} \\
&\iff \underbrace{(\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||) l_i}_{<0} < \underbrace{\underbrace{\Delta x}_{<0} \underbrace{||\boldsymbol{p}_i - \boldsymbol{x}|| \left(-1 + \sqrt{1 - \left(\frac{l_i}{||\boldsymbol{p}_i - \boldsymbol{x}||} \right)^2} \right)}_{<0}}_{>0}
\end{aligned}$$

As outlined, on the above inequality the left hand term is negative while the right hand term is positive. Hence this inequality is always true and by equivalence so is $a_4 < a_2$. We have thus shown:

$$\boxed{\theta_4 < \theta_2}$$

□

In a nutshell we have successfully shown that:

$$\boxed{\Delta x < 0 \implies \theta_1 < \theta_3 \leq \theta_4 < \theta_2} \tag{C.28}$$

2. **If $0 < \Delta \boldsymbol{x} < l_i$**
$0 < \Delta x < l_i$ and $\Delta y \in \mathbb{R}$ such that $||\boldsymbol{p}_i - \boldsymbol{x}||^2 = \Delta x^2 + \Delta y^2 \geq l_i^2 \implies |\Delta y| \geq \sqrt{l_i^2 - \Delta x^2}$ let:

$$\begin{aligned}
a_1 &= \frac{\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x} \\
a_2 &= \frac{\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x} \\
a_3 &= \frac{-\Delta y - \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x} \\
a_4 &= \frac{-\Delta y + \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x}
\end{aligned}$$

such that for $i \in [\![1, 4]\!]$ $\theta_i = 2 \arctan(a_i)$ and $\theta_3 \leq \theta_4$ and $\theta_1 \leq \theta_2$. We are going to prove that:

$$\begin{cases} \text{either } a_3 \leq a_4 \leq a_1 \leq a_2 \\ \text{or } a_1 \leq a_2 \leq a_3 \leq a_4 \end{cases} \tag{C.29}$$

Since by definition we have $a_1 \leq a_2$ and $a_3 \leq a_4$ we just need to prove that either $a_4 \leq a_1$ or $a_2 \leq a_3$. The demonstration is in three steps:

(a) First we show that:

$$\Delta y \geq ||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i} \right) \implies a_4 < a_1$$

(b) Then that:

$$\Delta y \leq -||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right) \implies a_2 < a_3$$

(c) And lastly that:

$$\begin{cases} 0 < \Delta x < l_i \\ ||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i \end{cases} \implies |\Delta y| \geq ||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right)$$

Let's dive in!

(a) $\Delta y \geq ||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right) \implies a_4 < a_1$:

Proof. We assume that $\Delta y \geq ||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right)$, i.e. that $\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x) \geq 0$. We prove $a_4 < a_1$ by equivalence:

$$\begin{aligned} a_4 < a_1 &\Leftrightarrow \frac{-\Delta y + \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x} < \frac{\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x} \\ &\Leftrightarrow -\Delta y \Delta x + \Delta x \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2} < (\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||)(l_i - \Delta x) \\ &\Leftrightarrow \underbrace{\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x)}_{\geq 0} > \underbrace{\Delta x \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}_{\geq 0} \\ &\Leftrightarrow (\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x))^2 > \left(\Delta x \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}\right)^2 \\ &\Leftrightarrow \underbrace{2||\boldsymbol{p}_i - \boldsymbol{x}||l_i}_{>0} \underbrace{(||\boldsymbol{p}_i - \boldsymbol{x}|| - \Delta y)}_{>0} \underbrace{(l_i - \Delta x)}_{>0} > 0 \end{aligned}$$

As outlined the latest inequality is always true and, by equivalence, so is $a_4 < a_1$
We have proven what we aimed for:

$$\boxed{\Delta y \geq ||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right) \implies \theta_4 < \theta_1} \quad \text{C.30}$$

□

(b) $\Delta y \leq -||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right) \implies a_2 < a_3$:

Proof. The demonstration is similar as the one above: assume $\Delta y \leq -||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right)$, i.e. $\Delta y l_i + ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x) \leq 0$. By equivalence:

$$\begin{aligned} a_2 < a_3 &\Leftrightarrow \frac{\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x} < \frac{-\Delta y - \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x} \\ &\Leftrightarrow (\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||)(l_i - \Delta x) < \Delta x \left(-\Delta y - \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}\right) \\ &\Leftrightarrow \underbrace{\Delta x \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}_{\geq 0} < \underbrace{-\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x)}_{\geq 0} \\ &\Leftrightarrow \left(\Delta x \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}\right)^2 < (-\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x))^2 \\ &\Leftrightarrow \underbrace{2||\boldsymbol{p}_i - \boldsymbol{x}||l_i}_{>0} \underbrace{(||\boldsymbol{p}_i - \boldsymbol{x}|| + \Delta y)}_{>0} \underbrace{(l_i - \Delta x)}_{>0} > 0 \end{aligned}$$

As outlined the latest inequality is always true and, by equivalence, so is $a_2 < a_3$.
We have proven what we aimed for:

$$\boxed{\Delta y \leq -||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right) \implies \theta_2 < \theta_3} \tag{C.31}$$

□

(c) $\begin{cases} 0 < \Delta x < l_i \\ ||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i \end{cases} \implies |\Delta y| \geq ||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right)$:

Proof. We propose a proof by contradiction: recall that $||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i$ and $0 < \Delta x < l_i$ implies $|\Delta y| \geq \sqrt{l_i^2 - \Delta x^2}$

$$\begin{aligned}
|\Delta y| < ||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right) &\implies l_i^2 - \Delta x^2 < (\Delta x^2 + \Delta y^2) \left(1 - \frac{\Delta x}{l_i}\right)^2 \\
&\implies (l_i^2 - \Delta x^2)\left(1 - 1\left(1 - \Delta x^2\right)\right) < \Delta x^2 \left(1 - \frac{\Delta x}{l_i}\right)^2 \\
&\implies \frac{l_i^2 - \Delta x^2}{l_i^2} \Delta x (2l_i - \Delta x) < \frac{\Delta x^2}{l_i^2} (l_i - \Delta x)^2 \\
&\implies (l_i^2 - \Delta x^2)(2l_i - \Delta x) < \Delta x (l_i - \Delta x)^2 \\
&\implies (l_i + \Delta x)(2l_i - \Delta x) < \Delta x (l_i - \Delta x) \\
&\implies 2l_i^2 < 0 \text{ impossible}
\end{aligned}$$

Hence by contradiction:

$$\boxed{\begin{cases} 0 < \Delta x < l_i \\ ||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i \end{cases} \implies |\Delta y| \geq ||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right)}$$

□

Summing up:

We have shown that in the situation $\begin{cases} 0 < \Delta x < l_i \\ ||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i \end{cases}$ then $|\Delta y| \geq ||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right)$.

Then:

- Either $\Delta y \geq ||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right)$ in which case we proved that $\theta_4 < \theta_1$. Moreover, by definition, $\theta_3 \leq \theta_4$ and $\theta_1 \leq \theta_2$. Hence, in this situation, we have the result $\theta_3 \leq \theta_4 < \theta_1 \leq \theta_2$.
- Or $\Delta y \leq -||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right)$ in which case we proved that $\theta_2 < \theta_3$. Moreover, by definition, $\theta_3 \leq \theta_4$ and $\theta_1 \leq \theta_2$. Hence, in this situation, we have the result $\theta_1 \leq \theta_2 < \theta_3 \leq \theta_4$.

All in all we have proven:

$$\boxed{\begin{cases} 0 < \Delta x < l_i \\ ||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i \end{cases} \implies \begin{cases} \theta_3 \leq \theta_4 < \theta_1 \leq \theta_2 \text{ if } \Delta y \geq ||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right) \\ \theta_1 \leq \theta_2 < \theta_3 \leq \theta_4 \text{ if } \Delta y \leq -||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right) \end{cases}}$$

3. If $\boldsymbol{\Delta x > l_i}$

For $\Delta x > l_i$; $\Delta y \in \mathbb{R}$ let:

$$
\begin{aligned}
a_1 &= \frac{\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x} \\
a_2 &= \frac{\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x} \\
a_3 &= \frac{-\Delta y + \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x} \\
a_4 &= \frac{-\Delta y - \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x}
\end{aligned}
$$

such that for $i \in [\![1, 4]\!]$ $\theta_i = 2\arctan(a_i)$ and $\theta_3 \leq \theta_4$ and $\theta_1 \leq \theta_2$. We prove that:

$$a_3 < a_1 \leq a_2 < a_4 \tag{C.32}$$

By definition $a_1 \leq a_2$ so all we have left to prove is $a_3 \leq a_1$ and $a_2 \leq a_4$.

(a) $a_3 < a_1$:

Proof. Two cases may occur:

- Either $\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x) \geq 0$
  In which case we prove our assertion by equivalence:

$$
\begin{aligned}
a_3 < a_1 &\Leftrightarrow \frac{-\Delta y + \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x} < \frac{\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x} \\
&\Leftrightarrow (\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||)(l_i - \Delta x) < \Delta x \left(-\Delta y + \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}\right) \\
&\Leftrightarrow \underbrace{\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x)}_{\geq 0} < \underbrace{\Delta x \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}_{\geq 0} \\
&\Leftrightarrow (\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x))^2 < \left(\Delta x \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}\right)^2 \\
&\Leftrightarrow \underbrace{2||\boldsymbol{p}_i - \boldsymbol{x}||l_i}_{>0} \underbrace{(||\boldsymbol{p}_i - \boldsymbol{x}|| - \Delta y)}_{>0} \underbrace{(l_i - \Delta x)}_{<0} < 0
\end{aligned}
$$

  As outlined the latest inequality is always true and we have shown:

$$\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x) \geq 0 \implies a_3 < a_1$$

- Or $\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x) < 0$
  In which case the proposition:

$$\underbrace{\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x)}_{<0} < \underbrace{\Delta x \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}_{\geq 0}$$

  is true and as we have shown just above:

$$\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x) < \Delta x \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2} \Leftrightarrow a_3 < a_1$$

  Hence we have shown:

$$\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x) < 0 \implies a_3 < a_1$$

We have shown that the assertion $a_3 < a_1$ is true whether $\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x)$ is positive or negative. Hence it is always true:

$$\boxed{\theta_3 < \theta_1}$$

□

(b) $a_2 < a_4$:

Proof. Two cases may occur:

⋆ Either $-\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x) \geq 0$
In which case we prove our assertion by equivalence:

$$\begin{aligned}
a_2 < a_4 &\Leftrightarrow \frac{\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x} < \frac{-\Delta y - \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x} \\
&\Leftrightarrow \Delta y l_i + ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x) > -\Delta x \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2} \\
&\Leftrightarrow \underbrace{-\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x)}_{\geq 0} < \underbrace{\Delta x \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}_{\geq 0} \\
&\Leftrightarrow \left(-\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x)\right)^2 < \left(\Delta x \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}\right)^2 \\
&\Leftrightarrow \underbrace{2||\boldsymbol{p}_i - \boldsymbol{x}||}_{>0} \underbrace{l_i}_{>0} \underbrace{(||\boldsymbol{p}_i - \boldsymbol{x}|| + \Delta y)}_{>0} \underbrace{(l_i - \Delta x)}_{<0} < 0
\end{aligned}$$

As outlined the latest inequality is always true and we have shown:

$$-\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x) < 0 \implies a_2 < a_4$$

⋆ Or $-\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x) < 0$
In which case the proposition:

$$\underbrace{-\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x)}_{<0} < \underbrace{\Delta x \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}_{\geq 0}$$

is true and as we have shown just above:

$$-\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x) \leq \Delta x \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2} \Leftrightarrow a_2 < a_4$$

Hence we have shown:

$$-\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x) < 0 \implies a_2 < a_4$$

We have shown that the assertion $a_2 < a_4$ is true whether $-\Delta y l_i - ||\boldsymbol{p}_i - \boldsymbol{x}||(l_i - \Delta x)$ is positive or negative. Hence it is always true:

$$\boxed{\theta_2 < \theta_4}$$

□

In a nutshell, we have successfully shown that:

$$\boxed{\Delta x > l_i \implies \theta_3 < \theta_1 \leq \theta_2 < \theta_4} \tag{C.33}$$

Now that we have discussed the ordering of the $(\theta_i)_{i\in[\![1,4]\!]}$ which depends on the position of Δx with respect to 0 and l_i we can shed further light on the intervals in which the `Turtle` can choose a θ to orient itself. This section treats the case where $||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i$ and $\Delta x \neq 0$.

If $s(l_i - \Delta x) < 0$

* If $s\Delta x > 0$

1. If $s = -1$
If $s = -1$ then $\Delta x < 0$ to comply with both $s\Delta x > 0$ and $s(l_i - \Delta x) < 0$. Using the definition of the θ_i provided in ITEM 1 we have

$$\theta_1 < \theta_3 \leq \theta_4 < \theta_2$$

2. If $s = +1$
If $s = +1$ then $\Delta x > l_i$. Using the definition of the θ_i provided in ITEM 3 we have

$$\theta_3 < \theta_1 \leq \theta_2 < \theta_4$$

In both case ($s = +1$ and $s = -1$) we have that $\theta_1 \leq \theta_4$ and so, referring to the enumeration, the `Turtle` always finds an angle θ to orient itself:

$$\theta \in [\max(\theta_3, \theta_1), \min(\theta_4, \theta_2)]$$

* If $s\Delta x < 0$
The `Turtle` cannot enter this condition when $s = +1$ (otherwise it would mean that $\Delta x < 0$ and $\Delta x > l_i$). Hence for $s = -1$ the `Turtle` is brought here when $0 < \Delta x < l_i$. In which case, with the definition of the θ_i provided ITEM 2 we have:

$$\begin{cases} \theta_3 \leq \theta_4 < \theta_1 \leq \theta_2 \text{ if } \Delta y \geq ||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right) \\ \theta_1 \leq \theta_2 < \theta_3 \leq \theta_4 \text{ if } \Delta y \leq -||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right) \end{cases}$$

We can now see that two of the four cases enumerated are impossible. We can also see that the two remaining cases yield the same result:

- We wrote : if $\theta_3 \leq \theta_1$ and $\theta_4 < \theta_2$ then $\theta \in [\theta_3, \min(\theta_4, \theta_1)]$, but as we proved that this case is only possible when $\theta_3 \leq \theta_4 < \theta_1 \leq \theta_2$ we now realize that $\theta \in [\theta_3, \theta_4]$.
- We wrote : if $\theta_1 < \theta_3$ and $\theta_2 \leq \theta_4$ then $\theta \in [\max(\theta_3, \theta_2), \theta_4]$, but as we proved that this case is only possible when $\theta_1 \leq \theta_2 < \theta_3 \leq \theta_4$ we now realize that $\theta \in [\theta_3, \theta_4]$

Hence when the `Turtle` is brought here it must choose

$$\theta \in [\theta_3, \theta_4]$$

If $s(l_i - \Delta x) > 0$

* If $s\Delta x > 0$
The `Turtle` cannot enter this condition when $s = -1$; when $s = +1$ the `Turtle` is brought here when $0 < \Delta x < l_i$. In which case, with the definition of the θ_i provided ITEM 2 we

have:

$$\begin{cases} \theta_3 \leq \theta_4 < \theta_1 \leq \theta_2 \text{ if } \Delta y \geq ||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right) \\ \theta_1 \leq \theta_2 < \theta_3 \leq \theta_4 \text{ if } \Delta y \leq -||\boldsymbol{p}_i - \boldsymbol{x}|| \left(1 - \frac{\Delta x}{l_i}\right) \end{cases}$$

We can now see that two of the four cases enumerated are impossible. We can also see that the two remaining cases yield the same result:

- We wrote : if $\theta_1 \leq \theta_3$ and $\theta_2 < \theta_4$ then $\theta \in [\theta_1, \min(\theta_2, \theta_3)]$, but as we proved that this case is only possible when $\theta_1 \leq \theta_2 < \theta_3 \leq \theta_4$ we now realize that $\theta \in [\theta_1, \theta_2]$.
- We wrote : if $\theta_3 < \theta_1$ and $\theta_4 \leq \theta_2$ then $\theta \in [\max(\theta_1, \theta_4), \theta_2]$, but as we proved that this case is only possible when $\theta_3 \leq \theta_4 < \theta_1 \leq \theta_2$ we now realize that $\theta \in [\theta_1, \theta_2]$

Hence when the `Turtle` is brought here it must choose

$$\theta \in [\theta_1, \theta_2]$$

* If $s\Delta x < 0$

1. If $s = -1$

If $s = -1$ then $\Delta x > l_i$. Using the definition of the θ_i provided in ITEM 3 we have

$$\theta_3 < \theta_1 \leq \theta_2 < \theta_4$$

2. If $s = +1$

If $s = +1$ then $\Delta x < 0$. Using the definition of the θ_i provided in ITEM 1 we have

$$\theta_1 < \theta_3 \leq \theta_4 < \theta_2$$

In both case we have $\max(\theta_3, \theta_1) \leq \min(\theta_4, \theta_2)$. Hence, referring to the enumeration provided above we see that one of the two cases is impossible and when the `Turtle` is brought here it must choose:

$$\theta \in]-\pi, \min(\theta_3, \theta_1)] \cup [\max(\theta_4, \theta_2), \pi[$$

This appendix helps us refine the long enumeration of SECTION C.2 into a greatly simplified version presented below. Also, the intervals where $\pm\pi$ appeared were left open to ensure than $x \mapsto \tan(x)$ was defined. Yet, it can be verified that the passage to the limit still defines our expressions: the intervals can be closed.

1. If $\Delta x = l_i$

(a) If $\Delta y = 0$:

- If $s = -1$: $\theta_i \in \emptyset$, no solution.
- If $s = +1$: $\theta_i \in [\theta_1, \theta_2]$

(b) If $\Delta y \neq 0$:

- If $s = +1$: $\theta_i \in [\theta_1, \theta_2]$
- Else:
  - If $\Delta y > 0$: $\theta_i \in [-\pi, \theta_0]$
  - If $\Delta y < 0$: $\theta_i \in [\theta_0, \pi]$

2. Else if $||\boldsymbol{p}_i - \boldsymbol{x}|| < l_i$

(a) If $s = -1$: $\theta_i \in \emptyset$, no solution.

(b) If $s = +1$:

- If $\Delta x = 0$:
  - If $\Delta y < 0$: $\theta_i \in [-\pi, 0]$
  - If $\Delta y > 0$: $\theta_i \in [0, \pi]$
- If $\Delta x > 0$: $\theta_i \in [\theta_1, \theta_2]$
- If $\Delta x < 0$: $\theta_i \in [-\pi, \theta_1] \cup [\theta_2, \pi]$

3. Else if $||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i$

(a) If $\Delta x = 0$:

- If $s = -1$: $\theta_i \in [\theta_3, \theta_4]$
- If $s = +1$:
  - If $\Delta y < 0$: $\theta_i \in [-\pi, 0]$
  - If $\Delta y > 0$: $\theta_i \in [0, \pi]$

(b) If $\Delta x \neq 0$:

- If $s(l_i - \Delta x) < 0$:
  - If $s\Delta x > 0$: $\theta_i \in [\max(\theta_3, \theta_1), \min(\theta_4, \theta_2)]$
  - If $s\Delta x < 0$: $\theta_i \in [\theta_3, \theta_4]$
- If $s(l_i - \Delta x) > 0$:
  - If $s\Delta x > 0$: $\theta_i \in [\theta_1, \theta_2]$
  - If $s\Delta x < 0$: $\theta_i \in [-\pi, \min(\theta_3, \theta_1)] \cup [\max(\theta_4, \theta_2), \pi]$

C.4 PROOF THAT THE CENTRE OF ROTATION IS ATTRACTIVE WHEN $s = -1$ AND REPULSIVE WHEN $s = +1$

We now prove that the centre of rotation $\boldsymbol{x}$ acts as a repulsive (resp. attractive) point when $s = +1$ (resp. $s = -1$) in the sense that the `Turtle` moves away from (resp. towards) $\boldsymbol{x}$. Recall that $\boldsymbol{p}_i$ denotes the current position of the `Turtle` , $\boldsymbol{p}_{i+1}$ the next position, i.e. $\boldsymbol{p}_{i+1} = \boldsymbol{p}_i + l_i \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$ for a θ chosen in the correct interval as described by ENUMERATION C.3. We are going to prove that when $s = +1$ $||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| > ||\boldsymbol{p}_i - \boldsymbol{x}||$ and when $s = -1$ $||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| < ||\boldsymbol{p}_i - \boldsymbol{x}||$. We solve the equation $||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| = ||\boldsymbol{p}_i - \boldsymbol{x}||$ of unknown θ. Recall that $\Delta x = x_i - x$ and $\Delta y = y_i - y$. In the following, we assume l_i to be known and fixed.

$$
\begin{aligned}
||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| = ||\boldsymbol{p}_i - \boldsymbol{x}|| &\Leftrightarrow ||\boldsymbol{p}_{i+1} - \boldsymbol{x}||^2 = ||\boldsymbol{p}_i - \boldsymbol{x}||^2 \\
&\Leftrightarrow (\Delta x + l_i \cos\theta)^2 + (\Delta y + l_i \sin\theta)^2 = \Delta x^2 + \Delta y^2 \\
&\Leftrightarrow 2\Delta x \cos\theta + 2\Delta y \sin\theta + l_i = 0 \\
&\Leftrightarrow (l_i - 2\Delta x) \tan^2\frac{\theta}{2} + 4\Delta y \tan\frac{\theta}{2} + l_i + 2\Delta x = 0
\end{aligned}
$$

Let

$$
\begin{array}{rccc}
P: &]-\pi, \pi[& \to & \mathbb{R} \\
& \theta & \mapsto & (l_i - 2\Delta x)\tan^2\frac{\theta}{2} + 4\Delta y \tan\frac{\theta}{2} + l_i + 2\Delta x
\end{array}
$$

And we have the equivalence:

$$
||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| = ||\boldsymbol{p}_i - \boldsymbol{x}|| \Leftrightarrow P(\theta) = 0 \tag{C.34}
$$

Several cases must be studied:

- If $2\Delta x = l_i$:

  1. If $\Delta y = 0$:
  Then P is constant: $\forall\theta \;\; P(\theta) = 2l_i > 0$. Also, in this case, $||\boldsymbol{p}_i - \boldsymbol{x}|| < l_i$ which implies that the `Turtle` cannot be here when $s = -1$. Hence if the `Turtle` is exactly at the distance $\left(\frac{l_i}{2}, 0\right)$ from the centre $\boldsymbol{x}$, then $s = +1$ and since $P > 0$ then $||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| > ||\boldsymbol{p}_i - \boldsymbol{x}||$ which proves that $\boldsymbol{x}$ acts as a repulsive point and the `Turtle` moves away from it.

  2. If $\Delta y \neq 0$:
  Remark that

$$P(\theta) = 0 \Leftrightarrow \bar{P}(\theta) \equiv l_i(1 + \cos\theta) + 2\Delta y \sin\theta = 0$$

  Hence, $P(\theta) = 0 \Leftrightarrow \theta = \begin{cases} \pi \\ 2\arctan\frac{-l_i}{2\Delta y} \end{cases}$. Let $\bar{\theta} = 2\arctan\frac{-l_i}{2\Delta y}$. We first prove that when $s = +1 \; ||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| > ||\boldsymbol{p}_i - \boldsymbol{x}||$ and then that when $s = -1 \; ||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| < ||\boldsymbol{p}_i - \boldsymbol{x}||$:

  – If $s = +1$:

$$||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| > ||\boldsymbol{p}_i - \boldsymbol{x}|| \Leftrightarrow \theta \in \begin{cases}]\bar{\theta}, \pi[\text{ if } \Delta y > 0 \\ ]-\pi, \bar{\theta}[\text{ if } \Delta y < 0 \end{cases}$$

  Note that the angular opening of these two cones is strictly greater than π. We prove that whatever the sign of Δy such open cones contain the cone $[\theta_1, \theta_2]$ which is the cone in which the `Turtle` chooses a θ in such situation (see the (simplified) ENUMERATION C.3). Below we demonstrate the following implication: $\theta \in [\theta_1, \theta_2] \implies ||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| > ||\boldsymbol{p}_i - \boldsymbol{x}||$.

  (a) If $\Delta y > 0$:
  We claim that $\theta_1 > \bar{\theta}$.

  Proof. Since $\theta_2 > \theta_1$ we shall obtain $\bar{\theta} < \theta_1 < \theta_2 < \pi$ which proves the result wanted. In such case $\theta_1 = 2\arctan\frac{\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||}{l_i/2}$

$$\begin{aligned} \theta_1 > \bar{\theta} &\Leftrightarrow \frac{\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||}{l_i/2} > \frac{-l_i}{2\Delta y} \\ &\Leftrightarrow 2\Delta y^2 - 2||\boldsymbol{p}_i - \boldsymbol{x}||\Delta y + \frac{l_i^2}{2} > 0 \\ &\Leftrightarrow 2\left(||\boldsymbol{p}_i - \boldsymbol{x}||^2 - \left(\frac{l_i}{2}\right)^2\right) - 2||\boldsymbol{p}_i - \boldsymbol{x}||\Delta y + \frac{l_i^2}{2} > 0 \\ &\Leftrightarrow ||\boldsymbol{p}_i - \boldsymbol{x}|| - \Delta y > 0 \text{ always true} \end{aligned}$$

  By equivalence we have demonstrated the result looked for and so far we have proven that $s = +1$ and $\Delta y > 0 \implies \bar{\theta} < \theta_1 < \theta_2 < \pi$, which means that $\theta \in [\theta_1, \theta_2] \implies ||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| > ||\boldsymbol{p}_i - \boldsymbol{x}||$ □

  (b) If $\Delta y < 0$:
  We claim that $\theta_2 < \bar{\theta}$.

  Proof. Since $\theta_1 < \theta_2$ we shall obtain $-\pi < \theta_1 < \theta_2 < \bar{\theta}$ which proves the result wanted. Here $\theta_2 = 2\arctan\frac{\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||}{l_i/2}$. Similarly as above we get the equivalence:

$$\theta_2 < \bar{\theta} \Leftrightarrow ||\boldsymbol{p}_i - \boldsymbol{x}|| + \Delta y > 0 \text{ always true}$$

□

By equivalence we have shown that $s = +1$ and $\Delta y < 0 \implies -\pi < \theta_1 < \theta_2 < \bar{\theta}$ which means that $\theta \in [\theta_1, \theta_2] \implies ||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| > ||\boldsymbol{p}_i - \boldsymbol{x}||$.

We have proven that no matter the sign of Δy, the cone $[\theta_1, \theta_2]$ is strictly contained in the open cone defined by $\bar{\theta_0}$ and $\pm\pi$. Hence, when $s = +1$, since the `Turtle` picks a $\theta \in [\theta_1, \theta_2]$, we have that $||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| > ||\boldsymbol{p}_i - \boldsymbol{x}||$ and thus the centre of rotation is repulsive.

3. If $s = -1$:

$$||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| < ||\boldsymbol{p}_i - \boldsymbol{x}|| \Leftrightarrow \theta \in \begin{cases}]\bar{\theta}, \pi[\text{ if } \Delta y < 0 \\ ]-\pi, \bar{\theta}[\text{ if } \Delta y > 0 \end{cases}$$

Given that if $||\boldsymbol{p}_i - \boldsymbol{x}|| < l_i$ the `Turtle` cannot be in this situation when $s = -1$ we just need to look at the case $||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i$. For $\Delta x = \frac{l_i}{2}$, $s = +1$ and $||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i$ then $s(l_i - \Delta x) > 0$ and $s\Delta x > 0$ and, according to the ENUMERATION C.3, the only cone in which the `Turtle` picks an angle is $[\theta_3, \theta_4]$. So we want to demonstrate that $\bar{\theta} < \theta_3 < \theta_4 < \pi$ if $\Delta y < 0$ and $-\pi < \theta_3 < \theta_4 < \bar{\theta}$ if $\Delta y > 0$ so as to imply that $\theta \in [\theta_3, \theta_4] \implies ||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| < ||\boldsymbol{p}_i - \boldsymbol{x}||$.
First we prove the following inequality:

$$2||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2 > 2|\Delta y|\sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2} \qquad \star$$

Proof.

$$\begin{aligned}
(\star) &\Leftrightarrow \left(2||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2\right)^2 > 4\Delta y^2 \left(||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2\right) \\
&\Leftrightarrow \left(2||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2\right)^2 > 4\left(||\boldsymbol{p}_i - \boldsymbol{x}||^2 - \left(\frac{l_i}{2}\right)^2\right)\left(||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2\right) \\
&\Leftrightarrow (l_i||\boldsymbol{p}_i - \boldsymbol{x}||)^2 > 0 \text{ always true}
\end{aligned}$$

As outlined, the latest inequality shows that by equivalence $(\star)$ is true. □

(a) If $\Delta y > 0$:

In such case $\theta_4 = 2\arctan\frac{-\Delta y + \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i/2}$ and:

$$\begin{aligned}
\theta_4 < \bar{\theta} &\Leftrightarrow \frac{-\Delta y + \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i/2} < -\frac{l_i}{2\Delta y} \\
&\Leftrightarrow -2\Delta y^2 + 2\Delta y\sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2} + \frac{l_i^2}{2} < 0 \\
&\Leftrightarrow -2\left(||\boldsymbol{p}_i - \boldsymbol{x}||^2 - \left(\frac{l_i}{2}\right)^2\right) + 2\Delta y\sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2} + \frac{l_i^2}{2} < 0 \\
&\Leftrightarrow 2||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2 > 2\Delta y\sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2} \\
&\Leftrightarrow (\star)
\end{aligned}$$

Since $(\star)$ is true, so is $\theta_4 < \bar{\theta}$ and we have proven what we wished for when $\Delta y > 0$.

(b) If $\Delta y < 0$:

In such case $\theta_3 = 2\arctan\frac{-\Delta y-\sqrt{||\boldsymbol{p}_i-\boldsymbol{x}||^2-l_i^2}}{l_i/2}$ and:

$$\begin{aligned}\theta_3 > \bar{\theta} &\Leftrightarrow \frac{-\Delta y-\sqrt{||\boldsymbol{p}_i-\boldsymbol{x}||^2-l_i^2}}{l_i/2} > \frac{-l_i}{2\Delta y}\\ &\Leftrightarrow 2||\boldsymbol{p}_i-\boldsymbol{x}||^2 - l_i^2 > -2\Delta y\sqrt{||\boldsymbol{p}_i-\boldsymbol{x}||^2-l_i^2}\\ &\Leftrightarrow (\star)\end{aligned}$$

Since $(\star)$ is true, so is $\theta_4 < \bar{\theta}$ and we have proven what we wished for when $\Delta y < 0$.

We have proven that no matter the sign of Δy the cone $[\theta_3, \theta_4]$ is strictly contained in the cone defined by $\bar{\theta_0}$ and $\pm\pi$ for which $||\boldsymbol{p}_{i+1}-\boldsymbol{x}|| < ||\boldsymbol{p}_i-\boldsymbol{x}||$. Hence we have proven that $\theta \in [\theta_3,\theta_4] \implies ||\boldsymbol{p}_{i+1}-\boldsymbol{x}|| < ||\boldsymbol{p}_i-\boldsymbol{x}||$. As the `Turtle` must choose an angle in $[\theta_3,\theta_4]$ when $s=-1$ we have successfully shown that centre $\boldsymbol{x}$ acts as an attractive point (the `Turtle` moves closer to it) when $s=-1$.

We have shown that if $l_i = 2\Delta x$ then the `Turtle` gets away from $\boldsymbol{x}$ when $s=+1$ and is drawn to it when $s=-1$.

- If $l_i - 2\Delta x \neq 0$:
  If $l_i - 2\Delta x \neq 0$ then P is a second order polynomial in $\tan\frac{\theta}{2}$.
  Its discriminant is:

$$\begin{aligned}\Delta &= 16\Delta y^2 - 4(l_i-2\Delta x)(l_i+2\Delta x)\\ &= 4\left(4||\boldsymbol{p}_i-\boldsymbol{x}||^2 - l_i^2\right)\end{aligned}$$

  - If $||\boldsymbol{p}_i-\boldsymbol{x}|| < \frac{l_i}{2}$:
    Then $\Delta < 0$ and P does not have real roots. Note that in such case we have in particular that $||\boldsymbol{p}_i-\boldsymbol{x}|| < l_i$ and so the `Turtle` cannot be here when $s=-1$.
    We can derive the following implications:

$$\begin{aligned}||\boldsymbol{p}_i-\boldsymbol{x}|| < \frac{l_i}{2} &\implies 2|\Delta x| < l_i\\ &\implies 2\Delta x < l_i\\ &\implies P > 0\\ &\implies ||\boldsymbol{p}_{i+1}-\boldsymbol{x}|| > ||\boldsymbol{p}_i-\boldsymbol{x}||\end{aligned}$$

    We have just proven that $\boldsymbol{x}$ is repulsive when $\Delta < 0$ (and $s=+1$, the only case possible here), i.e. when $||\boldsymbol{p}_i-\boldsymbol{x}|| < \frac{l_i}{2}$ then $P>0$, which implies $||\boldsymbol{p}_{i+1}-\boldsymbol{x}|| > ||\boldsymbol{p}_i-\boldsymbol{x}||$ and the `Turtle` moves away from $\boldsymbol{x}$.
  - If $||\boldsymbol{p}_i-\boldsymbol{x}|| \geq \frac{l_i}{2}$ P admits two real roots:

$$\theta_{56}^{\ominus} = 2\arctan\frac{-2\Delta y-\sqrt{4||\boldsymbol{p}_i-\boldsymbol{x}||^2-l_i^2}}{l_i-2\Delta x} \tag{C.35}$$

$$\theta_{56}^{\oplus} = 2\arctan\frac{-2\Delta y+\sqrt{4||\boldsymbol{p}_i-\boldsymbol{x}||^2-l_i^2}}{l_i-2\Delta x} \tag{C.36}$$

(C.37)

and let:

$$\theta_5 = \min(\theta_{56}^{\ominus}, \theta_{56}^{\oplus})$$
$$\theta_6 = \max(\theta_{56}^{\ominus}, \theta_{56}^{\oplus})$$

To prove that $\boldsymbol{x}$ is attractive or repulsive according to the sign of s we are going to prove that the interval in which the `Turtle` chooses θ [1] is contained in the interval $[\theta_5, \theta_6]$ or $[-\pi, \theta_5] \cup [\theta_6, \pi]$ (depending on whether we have $P > 0$ or $P < 0$). To that end we first prove that the intervals $[\theta_3, \theta_4]$, $[\theta_1, \theta_2]$ and $[\theta_5, \theta_6]$ are all centered around the same value (which also show that the complementary intervals $[-\pi, \theta_i] \cup [\theta_j, \pi]$ are also centered around the same value). Then we prove that the angular opening of the interval of interest (for instance $\theta_4 - \theta_3$ if the interval is $[\theta_3, \theta_4]$) is strictly less than the angular opening of the intervals bounded by θ_5 and θ_6. We then use the fact that having two cones centered around the same value such that the angular opening of one is strictly less than the one of the other essentially means that one of the cone is strictly contained in the other. In other words this will show for instance that

$$\begin{cases} [\theta_3, \theta_4] \subset [\theta_5, \theta_6] \\ \theta \in [\theta_5, \theta_6] \implies P < 0 \end{cases} \implies \boldsymbol{x} \text{ is attractive when the } \texttt{Turtle} \text{ chooses } \theta \in [\theta_3, \theta_4]$$

Proof that $[\theta_3, \theta_4]$, $[\theta_1, \theta_2]$ and $[\theta_5, \theta_6]$ are all centered around the same value

- Middle value of $[\theta_1, \theta_2]$. Let $a_{3,4}$ be such that $\theta_{3,4} = 2\arctan a_{3,4}$, i.e. $a_{3,4} = \frac{\Delta y \pm ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x}$

$$\begin{aligned} \tan\frac{\theta_2 + \theta_1}{2} &= \tan\left(\arctan a_3 + \arctan a_4\right) \\ &= \frac{a_3 + a_4}{1 - a_3 a_4} \\ &= \frac{\frac{\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x} + \frac{\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x}}{1 - \frac{\Delta y - ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x}\frac{\Delta y + ||\boldsymbol{p}_i - \boldsymbol{x}||}{\Delta x}} \\ &= \frac{\frac{2\Delta y}{\Delta x}}{1 + \frac{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - \Delta y^2}{\Delta x^2}} \\ &= \frac{\Delta y}{\Delta x} \end{aligned}$$

- Middle value of $[\theta_3, \theta_4]$. The exact same reasoning as above yield:

$$\tan\frac{\theta_3 + \theta_4}{2} = \frac{\Delta y}{\Delta x}$$

- Middle value of $[\theta_5, \theta_6]$. Idem:

$$\tan\frac{\theta_5 + \theta_6}{2} = \frac{\Delta y}{\Delta x}$$

We have shown that $\tan\frac{\theta_3+\theta_4}{2} = \tan\frac{\theta_1+\theta_2}{2} = \tan\frac{\theta_5+\theta_6}{2} =$ and so that the cones $[\theta_3, \theta_4]$, $[\theta_4, \theta_1]$ and $[\theta_5, \theta_6]$ are all centered around $2\arctan\frac{\Delta y}{\Delta x}$. We now prove that depending on the value of s the cone $[\theta_5, \theta_6]$ or $[-\pi, \theta_5] \cup [\theta_6, \pi]$ contains the cone $[\theta_i, \theta_{i+1}]$ or $[-\pi, \theta_i] \cup [\theta_{i+1}, \pi]$ for $i \in \{1, 3\}$. We first start by investigate the case $s = -1$:

1. **If $s = -1$**

Since the `Turtle` cannot be in a situation where $s = -1$ and $\frac{l_i}{2} \leq ||\boldsymbol{p}_i - \boldsymbol{x}|| < l_i$ we must

[1] so $[\theta_3, \theta_4]$ or $[\theta_1, \theta_2]$ or $[-\pi, \theta_3] \cup [\theta_4, \pi]$ or $[-\pi, \theta_1] \cup [\theta_2, \pi]$ depending on the value of $\Delta x, \Delta y, l_i$ and s

assume here that $||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i$. Looking at ENUMERATION C.3 we see that the only intervals that we must look to are either $[\theta_3, \theta_4]$ or $[-\pi, \theta_3] \cup [\theta_4, \pi]$. Also we subdivide $\mathbb{R}$ in three regions: $\Delta x < \frac{l_i}{2}$; $\frac{l_i}{2} < \Delta x < l_i$ and $\Delta x > l_i$. Before investigating each of these parts we prove the following result:

$$\theta_{56}^{\oplus} - \theta_{56}^{\ominus} > \theta_{34}^{\oplus} - \theta_{34}^{\ominus} \qquad \star$$

Where for the record:

$$\begin{aligned}
\theta_{34}^{\ominus} &= 2\arctan\frac{-\Delta y - \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x} \\
\theta_{34}^{\oplus} &= 2\arctan\frac{-\Delta y + \sqrt{||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - \Delta x} \\
\theta_{56}^{\ominus} &= 2\arctan\frac{-2\Delta y - \sqrt{4||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - 2\Delta x} \\
\theta_{56}^{\oplus} &= 2\arctan\frac{-2\Delta y + \sqrt{4||\boldsymbol{p}_i - \boldsymbol{x}||^2 - l_i^2}}{l_i - 2\Delta x}
\end{aligned}$$

To prove $(\star)$ we are going to prove $\tan\frac{\theta_{56}^{\oplus} - \theta_{56}^{\ominus}}{2} > \tan\frac{\theta_{34}^{\oplus} - \theta_{34}^{\ominus}}{2}$. Since the function $x \mapsto 2\arctan x$ is strictly increasing on $\mathbb{R}$ we will have proven the result wanted. For the sake of readability, let $h = \frac{\Delta x}{l_i}$, $u = \frac{\Delta y}{l_i}$ and $a = \frac{||\boldsymbol{p}_i - \boldsymbol{x}||}{l_i}$. Let $a_{56}^{\ominus,\oplus} = \frac{-2u \pm \sqrt{4a^2-1}}{1-2h}$, so that $\theta_{56}^{\ominus,\oplus} = 2\arctan a_{56}^{\ominus,\oplus}$

$$\begin{aligned}
\tan\frac{\theta_{56}^{\oplus} - \theta_{56}^{\ominus}}{2} &= \tan(\arctan a_{56}^{\oplus} - \arctan a_{56}^{\ominus}) \\
&= \frac{a_{56}^{\oplus} - a_{56}^{\ominus}}{1 + a_{56}^{\oplus} a_{56}^{\ominus}} \\
&= \frac{2\frac{\sqrt{4a^2-1}}{1-2h}}{1 + \frac{1-4h^2}{(1-2h)^2}} \\
&= \frac{2\frac{\sqrt{4a^2-1}}{1-2h}}{\frac{2}{1-2h}} \\
&= \sqrt{4a^2 - 1}
\end{aligned}$$

Similarly:

$$\tan\frac{\theta_{12}^{\oplus} - \theta_{12}^{\ominus}}{2} = \sqrt{a^2 - 1}$$

Since (as $||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i \implies a \geq 1$) $\sqrt{4a^2 - 1}$ is greater than $\sqrt{a^2 - 1}$ and we have proved the result wished for: the angular opening of the cone $[\theta_{56}^{\ominus}, \theta_{56}^{\oplus}]$ is greater than the one of $[\theta_{34}^{\ominus}, \theta_{34}^{\oplus}]$. We can now investigate the what that means for the `Turtle` depending on the value pf Δx.

- $\star$ $\Delta x < \frac{l_i}{2}$:
  According to ENUMERATION C.3, the `Turtle` must choose θ in the cone $[\theta_3, \theta_4]$ with $\theta_3 = \theta_{34}^{\ominus}$ and $\theta_4 = \theta_{34}^{\oplus}$. Also, since $l_i - 2\Delta x > 0$ the polynomial $\theta \mapsto P\left(\tan\frac{\theta}{2}\right)$ is negative in between its roots: $\theta \in]\theta_{56}^{\ominus}, \theta_{56}^{\oplus}[\implies P\left(\tan\frac{\theta}{2}\right) < 0 \implies ||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| < ||\boldsymbol{p}_i - \boldsymbol{x}|| \implies$ the `Turtle` is moving closer to $\boldsymbol{x}$. One can verify that in such case $\theta_5 = \theta_{56}^{\ominus}$ and $\theta_6 = \theta_{56}^{\oplus}$. We had previously proven that the intervals $[\theta_3, \theta_4]$ and $[\theta_5, \theta_6]$ are centered around the same value and we just proved that the angular opening of $[\theta_3, \theta_4]$ is strictly less than the one of $[\theta_5, \theta_6]$. Hence we have proven that

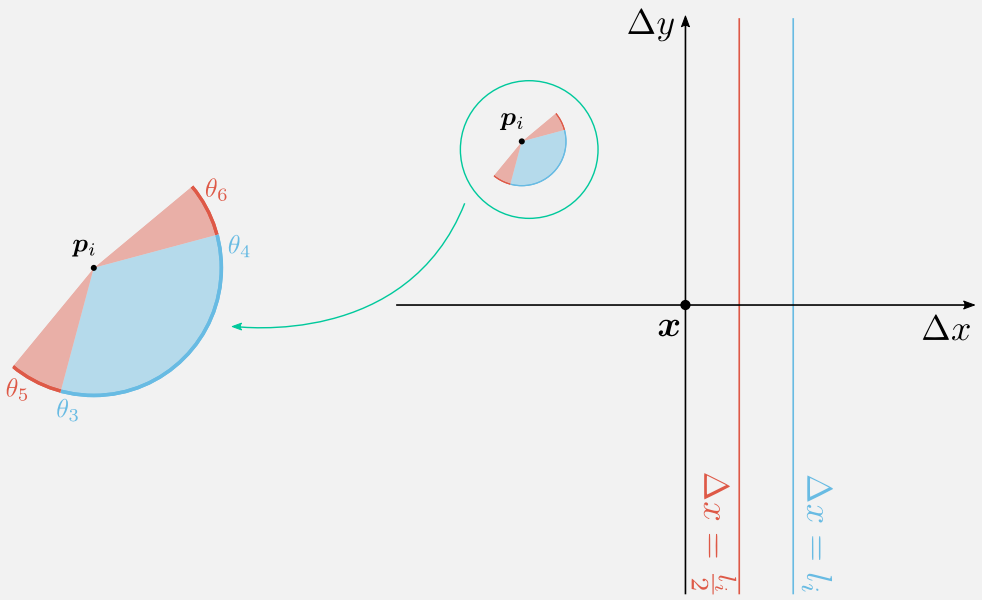


Figure C.1| Situation when $s = -1$ and $\Delta x < \frac{l_i}{2}$: the cone in blue is $[\theta_3, \theta_4]$, i.e. the range in which the `Turtle` may pick θ. Underneath in red is the cone $]\theta_5, \theta_6[$, i.e. the angular interval in which, when the `Turtle` chooses a θ in it, it moves closer to $\boldsymbol{x}$. As $[\theta_3, \theta_4] \subset]\theta_5, \theta_6[$ the `Turtle` moves towards $\boldsymbol{x}$.

$[\theta_3, \theta_4] \subset]\theta_5, \theta_6[$. Summing all up we have the result that when $s = -1$, $\Delta x < \frac{l_i}{2}$ (and necessarily $||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i$) the `Turtle` picks a $\theta \in [\theta_3, \theta_4] \subset]\theta_5, \theta_6[\implies$ the `Turtle` moves towards $\boldsymbol{x}$: $\boldsymbol{x}$ is attractive. An illustration of this case is provided FIGURE C.1.

- $\frac{l_i}{2} < \Delta x < l_i$
  Here $\theta_3 = \theta_{34}^{\ominus}$, $\theta_4 = \theta_{34}^{\oplus}$, $\theta_5 = \theta_{56}^{\oplus}$ and $\theta_6 = \theta_{56}^{\ominus}$. The `Turtle` picks $\theta \in [\theta_3, \theta_4]$. Also the cone in which P is negative is $]\theta_6, \pi] \cup [-\pi, \theta_5]$ whose angular opening is $\theta_5 - \theta_6$. These two cones being centered around the same value we have the result that the cone $[\theta_3, \theta_4] \subset]\theta_6, \pi] \cup [-\pi, \theta_5]$: when $s = -1$ and $\frac{l_i}{2} < \Delta x < l_i$ (and $||\boldsymbol{p}_i - \boldsymbol{x}|| \geq l_i$) then $\theta \in [\theta_3, \theta_4] \subset]\theta_6, \pi] \cup [-\pi, \theta_5] \implies$ the `Turtle` moves towards $\boldsymbol{x}$: $\boldsymbol{x}$ is attractive, see an illustration of this case on FIGURE C.2..
- If $\Delta x > l_i$
  This time the cone in which the `Turtle` chooses θ is $[\theta_4, \pi[\cup]-\pi, \theta_3]$ and the cone in which P is negative is $]\theta_6, \pi] \cup [-\pi, \theta_5]$. The same line of thoughts as for the two previous cases yields that in this case also the `Turtle` is drawn to $\boldsymbol{x}$, see FIGURE C.3.

2. **If $s = +1$**
We could prove in this case that the `Turtle` moves away from $\boldsymbol{x}$ by calculation, as we did when $s = +1$ but simply we observe the following: the cone $[\theta_1, \theta_2]$ or $[\theta_2, \pi] \cup [-\pi, \theta_1]$ (seen as a half-plane) is tangent to the circle of centre $\boldsymbol{x}$ and radius $||\boldsymbol{p}_i - \boldsymbol{x}||$ in $\boldsymbol{p}_i$. Indeed one can verify that $(\boldsymbol{p}_i - \boldsymbol{x}) \cdot (\boldsymbol{p}_i - \boldsymbol{p}_{i+1}) = 0$ where $\boldsymbol{p}_{i+1} = \boldsymbol{p}_i + l_i \begin{pmatrix} \cos\theta_i \\ \sin\theta_i \end{pmatrix}$ for $i \in [\![3, 4]\!]$. Hence this cone does not intersect the circle of centre $\boldsymbol{x}$ and radius $||\boldsymbol{p}_i - \boldsymbol{x}||$ anywhere except in $\boldsymbol{p}_i$: by moving on the boundary of such cone the `Turtle` is sure to increase its distance with $\boldsymbol{x}$ which therefore acts as a repulsive point in the sense that the `Turtle` moves away from it, as shown on FIGURE C.4.

In this appendix we have successfully proved that no matter the relative position of $\boldsymbol{p}_i$ with respect to $\boldsymbol{x}$

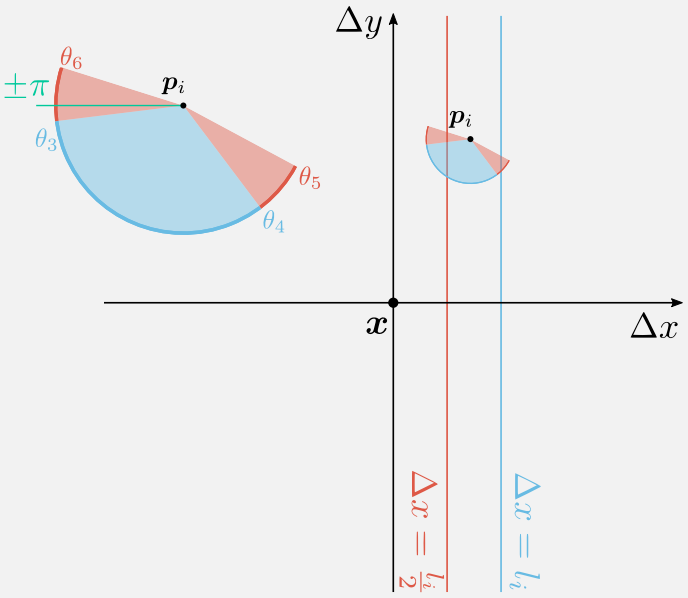


Figure C.2| Situation when $s = -1$ and $\frac{l_i}{2} < \Delta x < l_i$: see FIGURE C.1 for a caption. The point is that the `Turtle` moves towards $\boldsymbol{x}$.

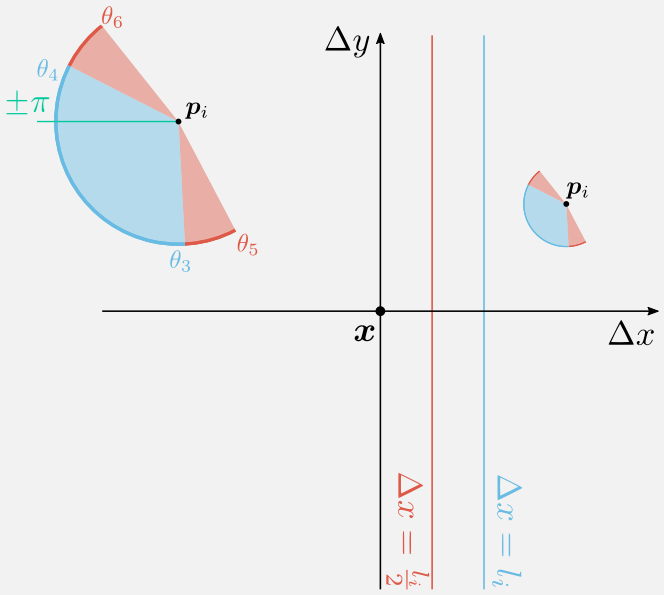


Figure C.3| Situation when $s = -1$ and $\Delta x > l_i$: see FIGURE C.1 for a caption. The point is that the `Turtle` moves towards $\boldsymbol{x}$.

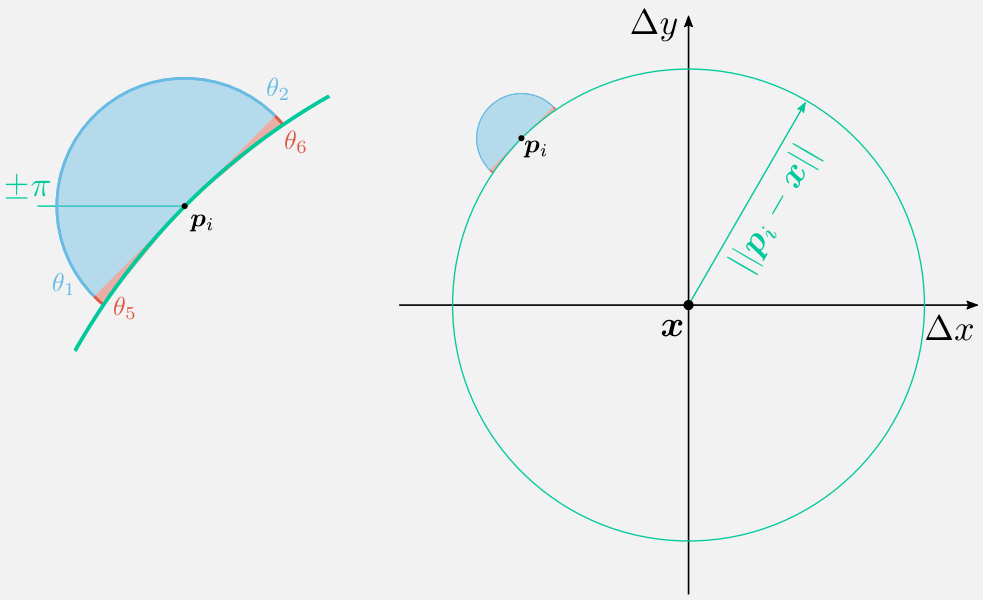


Figure C.4| Situation when $s = +1$ and $\Delta x > l_i$: see FIGURE C.1 for a caption. The point is that the `Turtle` moves away from $\boldsymbol{x}$.

the following assertion is true:

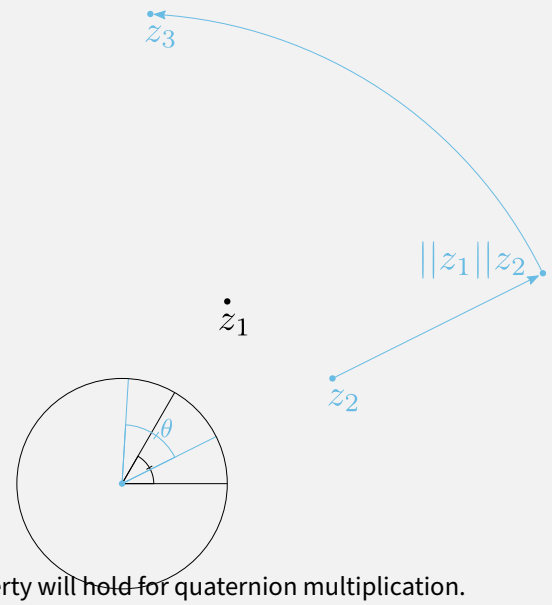
$$\begin{cases} s = -1 & \Longrightarrow ||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| < ||\boldsymbol{p}_i - \boldsymbol{x}|| \ \boldsymbol{x} \text{ is attractive} \\ s = +1 & \Longrightarrow ||\boldsymbol{p}_{i+1} - \boldsymbol{x}|| > ||\boldsymbol{p}_i - \boldsymbol{x}|| \ \boldsymbol{x} \text{ is repulsive} \end{cases}$$

APPENDIX D

QUATERNIONS AND DUAL QUATERNIONS

D.1 QUATERNIONS

D.1.1 COMPLEX MULTIPLICATION



As a small warm-up before understanding why quaternions rotate 3D objects, let us see what happens on the complex plane when one wants to multiply two points, for example $z_1 = (1, \sqrt{3}) = 1 + \sqrt{3}\boldsymbol{i}$ by $z_2 = (2, 1) = 2 + \boldsymbol{i}$. An elegant way to visualize the operation is to first scale z_2 by $||z_1||$ to get $||z_1||z_2 = 2\boldsymbol{z_2} = 4 + 2\boldsymbol{i}$ and then to rotate this new point through the angle define by the unit number $\frac{z_1}{||z_1||}$, i.e $\theta = \arg(\frac{z_1}{||z_1||}) = \frac{\pi}{3}$ to get $z_3 = (2 - \sqrt{3}, (1 + 2\sqrt{3}))$, see the inset. As one can see in this small example, if one wants to use complex algebra to simply rotate z_2 about the origin while keeping distances constant (i.e. a *pure* rotation or a rigid body motion) one must multiply z_2 by some unit complex number to avoid scaling it. This property will hold for quaternion multiplication.

D.1.2 GENERALIZING FROM 2D COMPLEX NUMBERS TO 4D QUATERNIONS

The multiplication between two quaternions q_1 and q_2 involves first scaling q_2 by the norm of q_1 and then rotating that scaled quaternion through some angle defined in q_1:

$$q_1 q_2 = \frac{q_1}{||q_1||}||q_1||q_2 \tag{D.1}$$

Where $||q_1||q_2$ represents the operation of scaling q_2 by the norm of q_1 and is the direct analogous of the complex multiplication described above, and $\frac{q_1}{||q_1||}$ corresponds to a rotation in a 4D sense applied to $||q_1||q_2$. As shown by this equation and the same as complex multiplication, the rotation (so by definition an operation keeping distances constant) must involve only unit quaternions lying on the unit hypersphere of $\mathbb{R}^4$ otherwise, depending on whether $||q_1||$ is greater or smaller than 1, the norm of the result is scaled up or down.

D.1.2.1 Beware of the pitfall !

What I believe to be the most naive way to generalize what we saw for complex rotation to quaternionic rotation is shown thereafter; we are going to work through an example to see where this intuition fails and how to rectify it to correctly express 3D rotation using quaternions.
The quaternionic representation of a vector (or a point) in $\mathbb{R}^3$, say vector $\boldsymbol{v} = (x, y, z)^T$, is to build a quaternion whose scalar part is 0 and whose imaginary part is equal to the vector: $v = [0, \boldsymbol{v}]$. Let us consider a point $\boldsymbol{p} \in \mathbb{R}^3$ that we want to rotate around an axis $\boldsymbol{q} \in \mathbb{R}^3$ through an angle θ in a clockwise manner. Introducing the quaternionic form of our 3D point as $p = [0, \boldsymbol{p}]$ and of the action "*rotate by θ around axis $\boldsymbol{q}$*" as $q = [\cos\theta, \boldsymbol{q}\sin\theta]$, a simplistic and soon-to-be-proven-wrong translation of our problem "*perform a clockwise rotation of $\boldsymbol{p}$ about $\boldsymbol{q}$ by θ*" into the quaternionic language would simply be qp (had we wanted an anticlockwise rotation, we would have written pq). Following EQUATION (D.1), we can without loss of generality assume that our axis of rotation is depicted in $\mathbb{R}^3$ as a unit vector which automatically implies that our quaternion q is of unit norm. Let us compute this product to see what happens:

$$\begin{aligned} qp &= [\cos\theta, \boldsymbol{q}\sin\theta][0, \boldsymbol{p}] \\ &= [-(\boldsymbol{q}\cdot\boldsymbol{p})\sin\theta, \boldsymbol{p}\cos\theta + (\boldsymbol{q}\times\boldsymbol{p})\sin\theta] \end{aligned}$$

The norm of qp is given by

$$\begin{aligned} ||qp||^2 &= ((\boldsymbol{q}\cdot\boldsymbol{p})\sin\theta)^2 + (\boldsymbol{p}\cos\theta + (\boldsymbol{q}\times\boldsymbol{p})\sin\theta)^2 \\ &= ((\boldsymbol{q}\cdot\boldsymbol{p})\sin\theta)^2 + (\boldsymbol{p}\cos\theta)^2 + ((\boldsymbol{q}\times\boldsymbol{p})\sin\theta)^2 \\ &= (\boldsymbol{q}\cdot\boldsymbol{p})^2\sin^2\theta + ||\boldsymbol{p}||^2\cos^2\theta + \left(||\boldsymbol{q}||^2||\boldsymbol{p}||^2 - (\boldsymbol{q}\cdot\boldsymbol{p})\right)^2\sin^2\theta \\ &= ||\boldsymbol{p}||^2\cos^2\theta + ||\boldsymbol{p}||^2\sin^2\theta \;\text{ using } ||\boldsymbol{q}|| = 1 \\ &= ||\boldsymbol{p}||^2 \end{aligned}$$

Or we could simply have written $||qp|| = ||q||||p||$ (as can be easily checked for any quaternions q and p) and use the fact that $||q|| = 1$. What we just proved is that after transforming our quaternion p by our action q we get a new quaternion qp that lies at the same distance to the origin of $\mathbb{R}^4$ as p does which is consistent with the definition of a pure rotation.

Let us also compute the angle α between the quaternions p and qp (if everything went as expected we should get $\alpha = \theta$). In order to do so one must simply recall that a quaternion is nothing else that a 4D vector and that the angle between two n-dimensional vectors $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ is given by $\arccos\frac{\boldsymbol{v_1}\cdot\boldsymbol{v_2}}{||\boldsymbol{v_1}||||\boldsymbol{v_2}||}$. With a slight abuse of notation, we will denote in this specific section both a 3D vector $\boldsymbol{v} \in \mathbb{R}^3$ and its vectorized quaternionic form $\boldsymbol{v} \in \mathbb{R}^4$ with bold font and:

$$\begin{aligned} \cos\alpha &= \frac{\boldsymbol{qp}\cdot\boldsymbol{p}}{||\boldsymbol{qp}||||\boldsymbol{p}||} \\ &= \frac{\begin{pmatrix} -(\boldsymbol{q}\cdot\boldsymbol{p})\sin\theta \\ \boldsymbol{p}\cos\theta + (\boldsymbol{q}\times\boldsymbol{p})\sin\theta \end{pmatrix} \cdot \begin{pmatrix} 0 \\ \boldsymbol{p} \end{pmatrix}}{||\boldsymbol{p}||^2} \\ &= \frac{||\boldsymbol{p}||^2\cos\theta}{||\boldsymbol{p}||^2} \\ &= \cos\theta \end{aligned}$$

We do have $\alpha = \theta$ and because we had $||qp|| = ||p||$ what we end up with is a quaternion qp at the same

distance to the origin as the quaternion p and both of them are forming an angle θ between each other. This looks exactly like the pure rotation we were asking for. Yet there is one big issue that we overlooked until now with our quaternion qp: its scalar part is non-zero (except for special cases such as $\boldsymbol{q}$ and $\boldsymbol{p}$ are orthogonal or θ is a multiple of π) which mean that we can no longer interpret our 4D quaternion as a 3D point without losing some important information: for instance, if we project qp into the 3D world by simply forgetting the scalar part, we end up with a point that is not as the same distance to the origin of $\mathbb{R}^3$ as $\boldsymbol{p}$ is, meaning that we are no longer looking at a rotated point. In other words, if we rotate a solid body (seen as a collection of points) using quaternions and then forget the scalar part to project our rotated quaternions back into the 3D world, we end up with a deformed solid not rotated like we would have wanted, which is far from being the rigid body motion initially planned: FIGURE D.1 is obtained by successively transforming each endpoint $\boldsymbol{p}$ of the unit cube on the left through a "rotation" encoded by a quaternion q, of the form qp, and displaying the imaginary part only of the result on the right.

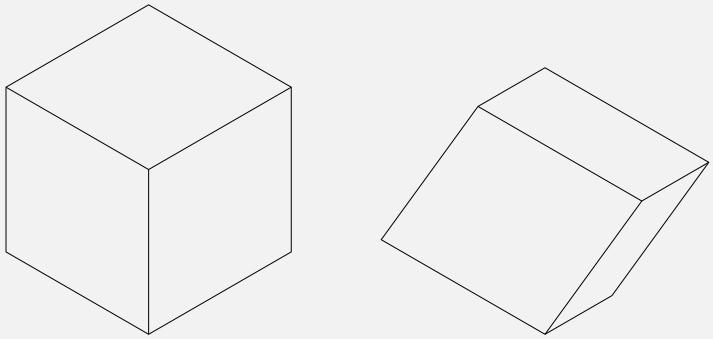


Figure D.1| Left: a cube. Right: the image of this cube through a rotation qp.

Before rectifying this naive approach and giving the correct procedure to use quaternions for spatial rotations, let us build some intuition on how and why we just morphed our 3D point $\boldsymbol{p}$ into a 4D quaternion pq that can no longer be interpreted as a point in 3D space. To do that let's rotate a 3D sphere and try to understand what would a 2D creature living in a plane see.

D.1.2.2 Getting some intuition: downgrading from 4D and 3D to 3D and 2D

A good way to understand with our 3D-wired brain what is going on in the 4D space when we use quaternions is to try to understand what a 2D creature living in the XY plane would see of a rotation in 3D. With that in mind, let us assume that we want to help a 2D creature living in the XY plane understand what rotating a 3D object looks like.

Rotating a vector in 3D is the same as unitizing this vector onto the 3D unit sphere, rotating that unit vector and then scaling it back to its original length. That being said we can without loss of generality focus on rotating points on the unit 3D sphere and this will give us the rotation of the entire 3D space.

On FIGURE D.2a, the unit sphere $\mathcal{S}^2$ is depicted and the equator, in red, is the only great circle of the sphere entirely embedded in the XY plane. Points on this equator are the only ones with a zero z-value and are hence considered by the creature as the only "*physical*" points of $\mathbb{R}^3$, same as we, as 3D creatures, only deem quaternions with a zero scalar part as being the only "*physical*" points of $\mathbb{R}^4$. In other words, this equator is the only unaltered part of the 3D sphere embedded into the plane, and points on this circle are the only ones that can be mentally represented by our hypothetical 2D creature, i.e they are the only points having some kind of physical meaning in a 2D world. Rotating this equator as a part of the 3D sphere is to the creature in 2D what rotating 3D points using quaternion is to us. So trying to understand how the creature perceives this equator as it rotates should give us intuitive insights into how we perceive the rotation of 3D points using quaternions. FIGURE D.2b lets us see what happens when the sphere is slightly rotated and we just go naively and "forget" the z component of the now-out-of-plane equator to mimic us

blindly ignoring the scalar part of the rotated quaternions. The blue ellipse is the vertical projection onto the plane of the equator in red. D.2c shows the whole sphere projected onto the plane while D.2d shows, in red, the unaltered equator and in various shades of blue the vertical projection of the equator for increasing angles of rotation of the sphere.

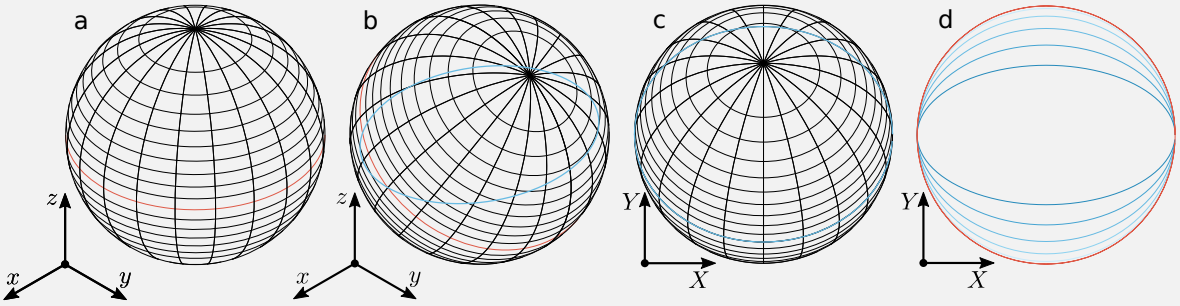


Figure D.2| What we see, versus what the 2D creatures sees.

The effect on the projected equator appears to be some kind of squishing for this particular rotation, as the images are ellipses. Obviously, the overall motion seen by the 2D creature is far from being a rigid body motion of the sphere as we see it ourselves in the 3D world.

One might object that this effect is an artefact of the projection chosen and that this particular projection (a vertical one by putting the z-value to 0) might not be the most suitable to represent a 3D sphere as two points symmetrical with respect to the plane are projected onto the same point and hence some information is necessarily lost. A somewhat better projection that uses the infinity of the XY plane to represent bijectively all points on the 3D sphere would be a stereographic projection, illustrated on FIGURE D.3. The faint lines on the backgrounds are the projection of the black latitudes and longitudes of the sphere. The equator is shown in red and its projection is in blue.

The effect of the rotation of the 3D sphere on its stereographic projection is shown in FIGURE D.4. On the left a view of how the 2D creature might perceive the sphere as it is stereographically projected onto the plane. The equator is highlighted in blue. Right: In red the original, unaltered, equator and in shades of blue the successive stereographic projections of the equator as the sphere gradually rotates. As the sphere rotates, the projection of the equator undergoes what could be best described as a translation along with some upscaling, again a transformation that is far from being the rigid body motion that we see in 3D.

D.1.3 CONCLUSION

The point of this small subsection was to build an intuitive understanding of what happens behind the scene when one multiplies the quaternionic representation of a point by a quaternion: we saw that - except in particular cases - the result quaternion has a non-zero scalar part and thus its imaginary part cannot be interpreted as the coordinates of a point after the rotation; we also showed that simply dropping the scalar component is not a solution, as distances, in particular, may not be recorded in the imaginary part only and the result of this projection into the 3D world is far from being a rigid body motion. A solid body transformed that way will most often look deformed and not rotated. To understand this phenomenon at an intuitive level, a tentative was made to explain what could a 2D creature living in the XY plane understand of the rotation of the 3D sphere, in the hope that the reader might get some insights on how to rectify his or her own perceptions on how quaternions as 4D numbers are used to rotate 3D objects.

The analogy in 2D was built around the equator of the 3D unit sphere as being the only part of the sphere fully embedded in the XY plane (so the set of unit points with a zero z-value) and that could hence be fully

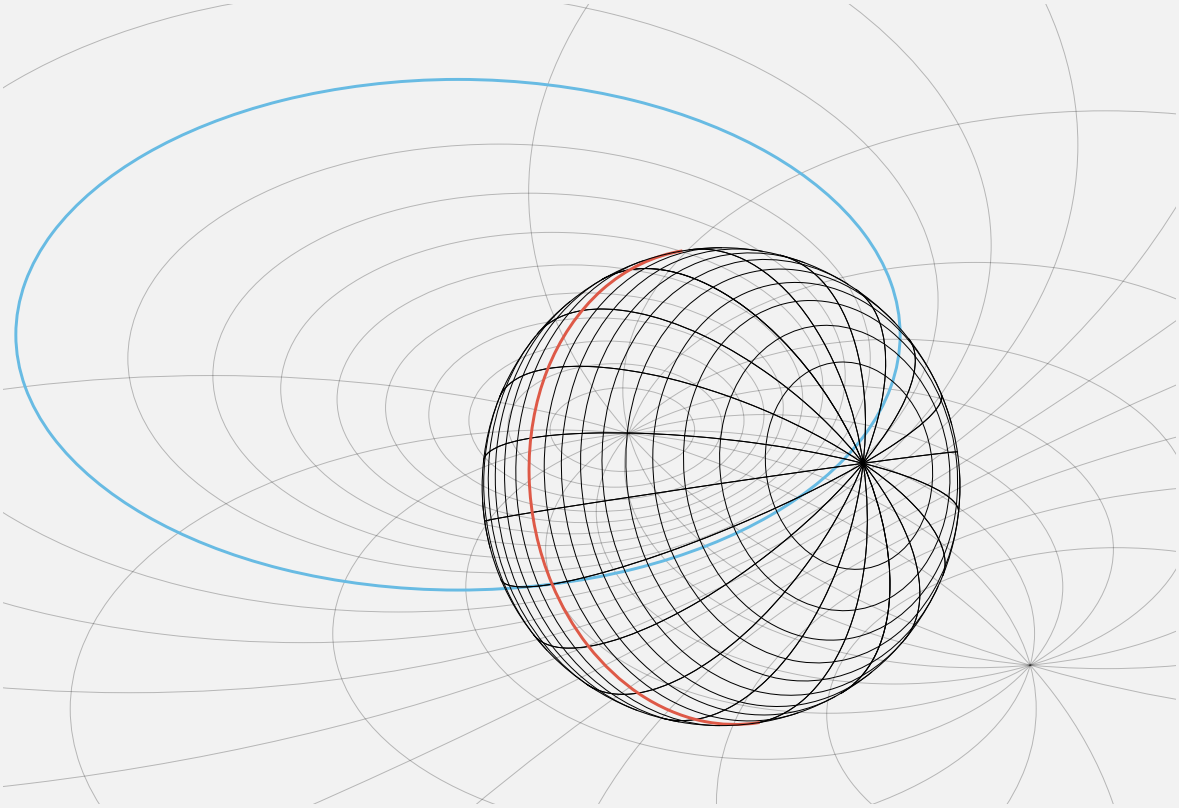


Figure D.3| In the background is the stereographic projection of the unit sphere. The circle in blue represents the original equator now rotated. The circle in orange depicts this equator under a stereographic projection onto the XY plane.

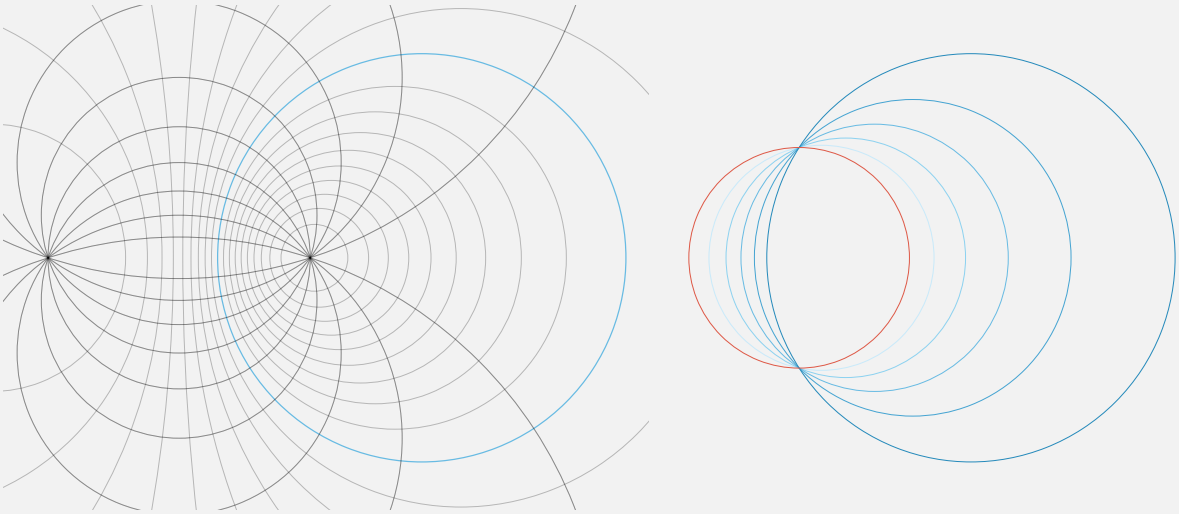


Figure D.4| Stereographic projection and its effect on the equator of rotated spheres.

understood by a hypothetical 2D creature. The counterpart of the equator for us is the set of unit quaternions having a zero scalar part and whose imaginary parts are interpreted as the coordinates of 3D points. Following how the equator is transformed after a rotation of the sphere would be to the creature what multiplying 3D points with quaternion is to us.

To have the 2D creature "see" the 3D sphere, two projections were used. The first one, termed the vertical projection, simply drops the z-component of the coordinates of points to project them onto the 2D plane,

to imitate what we did when the scalar part of the result quaternion was forgotten and we only looked at the imaginary part. The result of this projection on the rotated equator was some kind of squished circle (see FIGURE D.2) which closely mimics how the transformation of the unit cube on FIGURE D.1 was rotated but hugely deformed.
To show that this effect was not only an artefact of the chosen projection, another one was presented, the stereographic projection, which takes advantage of the infinity of the 2D plane to project bijectively the 3D sphere. There again the 2D creature could only interpret what we know to be a rigid body motion as some kind of translation motion and a scaling stretching of the equator. What the reader should bear in mind from now on is quite straightforward: if, after a rotation or any kind of transformation of a geometric object, we end up with a nonzero component on a dimension we cannot visualize (be it the z value of a points' coordinates for a 2D being or the scalar value of quaternions for us as 3D creatures) we must project this *non-physical* object back into the space that we were working with in the first place. This projection will inevitably deform in some way the geometry of the space where things happen and the result might not be to our taste.

What we have understood is twofold:

1. When we multiply the quaternionic representation $p = [0, \boldsymbol{p}]$ of a point $\boldsymbol{p} \in \mathbb{R}^3$ by a quaternion $q = [\cos\theta, \boldsymbol{u}\sin\theta]$ for some unit vector $\boldsymbol{u} \in \mathbb{R}^3$ and some angle θ we are in fact rotating p about $\boldsymbol{u}$ through θ in the **4D space**.
2. As long as the scalar part of the resulting quaternion qp (or pq) is nonzero we cannot interpret it as a 3D point without having to resort to some projection that will necessarily deform the geometry in an unwanted way.

In the next section, the correct formula to use when rotating 3D objects with quaternions will be introduced; we will also try to gain some intuitive understanding of why this formula works.

D.1.4 QUATERNIONS AND SPATIAL ROTATION

Let $\boldsymbol{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3$ be a point that shall be rotated about an axis $\boldsymbol{u} = \begin{pmatrix} u \\ v \\ w \end{pmatrix} \in \mathbb{R}^3$ (with $||\boldsymbol{u}|| = 1$) through an angle $\theta \in [-\pi, \pi[$ in a *clockwise* manner. Let $p = [0, \boldsymbol{p}] = \left[0, \begin{pmatrix} x \\ y \\ z \end{pmatrix}\right]$ be the quaternion counterpart of point $\boldsymbol{p}$. Let also $q = [\cos\frac{\theta}{2}, \boldsymbol{u}\sin\frac{\theta}{2}] = \left[\cos\frac{\theta}{2}, \begin{pmatrix} u\sin\frac{\theta}{2} \\ v\sin\frac{\theta}{2} \\ w\sin\frac{\theta}{2} \end{pmatrix}\right]$ be the quaternion representing the operation of rotating about axis $\boldsymbol{u}$ through the *half angle* $\frac{\theta}{2}$. Note that by construction $q \in \mathcal{U}(\mathbb{H})$.
The claim is that the imaginary part of the quaternion $p_r = qpq^{-1} = qpq^*$ can be interpreted as the 3D point that is solution to our problem (had we wanted a rotation in an *anticlockwise* manner, the solution would be given by $q^{-1}pq$). To prove that claim, let's compute the product given by p_r. For the sake of simplicity in the following equations $\sin\frac{\theta}{2}$ is denoted as s and $\cos\frac{\theta}{2}$ as c. The definition of the multiplication between

two quaternions is presented EQUATION (5.2).

$$
\begin{aligned}
p_r &= qpq^* && \text{D.2}\\
&= [c, s\boldsymbol{u}][0, \boldsymbol{p}][c, -s\boldsymbol{u}]\\
&= [-s\boldsymbol{u}\cdot\boldsymbol{p}, c\boldsymbol{p} + s\boldsymbol{u}\times\boldsymbol{p}][c, -s\boldsymbol{u}]\\
&= [-sc\boldsymbol{u}\cdot\boldsymbol{p} - (c\boldsymbol{p} + s\boldsymbol{u}\times\boldsymbol{p})\cdot(-s\boldsymbol{u}), (-s\boldsymbol{u}\cdot\boldsymbol{p})(-s\boldsymbol{u}) + c(c\boldsymbol{p} + s\boldsymbol{u}\times\boldsymbol{p}) + (c\boldsymbol{p} + s\boldsymbol{u}\times\boldsymbol{p})\times(-s\boldsymbol{u})]\\
&= [0, s^2(\boldsymbol{u}\cdot\boldsymbol{p})\boldsymbol{u} + c^2\boldsymbol{p} + sc\boldsymbol{u}\times\boldsymbol{p} - cs\boldsymbol{p}\times\boldsymbol{u} - s^2(\boldsymbol{u}\times\boldsymbol{p})\times\boldsymbol{u}]\\
&= [0, s^2(\boldsymbol{u}\cdot\boldsymbol{p})\boldsymbol{u} + c^2\boldsymbol{p} + 2sc\boldsymbol{u}\times\boldsymbol{p} + s^2\boldsymbol{u}\times(\boldsymbol{u}\times\boldsymbol{p})]\\
&= [0, s^2(\boldsymbol{u}\cdot\boldsymbol{p})\boldsymbol{u} + c^2\boldsymbol{p} + 2sc\boldsymbol{u}\times\boldsymbol{p} + s^2\big(\boldsymbol{u}(\boldsymbol{u}\cdot\boldsymbol{p}) - \boldsymbol{p}(\boldsymbol{u}\cdot\boldsymbol{u})\big)]\\
&= [0, 2s^2(\boldsymbol{u}\cdot\boldsymbol{p})\boldsymbol{u} + c^2\boldsymbol{p} - s^2\boldsymbol{p}||\boldsymbol{u}||^2 + 2sc(\boldsymbol{u}\times\boldsymbol{p})]\\
&= [0, 2s^2(\boldsymbol{u}\cdot\boldsymbol{p})\boldsymbol{u} + (c^2 - s^2)\boldsymbol{p} + 2sc(\boldsymbol{u}\times\boldsymbol{p})] && \text{D.3}\\
&= [0, (1-\cos\theta)(\boldsymbol{u}\cdot\boldsymbol{p})\boldsymbol{u} + \cos\theta\boldsymbol{p} + \sin\theta(\boldsymbol{u}\times\boldsymbol{p})] && \text{D.4}
\end{aligned}
$$

The scalar part of p_r is equal to zero: we can safely interpret the imaginary part as the coordinates of a point $\boldsymbol{p_r} = 2s^2(\boldsymbol{u}\cdot\boldsymbol{p})\boldsymbol{u} + (c^2 - s^2)\boldsymbol{p} + 2sc\boldsymbol{u}\times\boldsymbol{p} \in \mathbb{R}^3$: $p_r = [0, \boldsymbol{p_r}]$. We must now prove that $||p_r|| = ||p||$ and that the angle between p_r and p is θ.

$$
\begin{aligned}
||p_r||^2 &= \begin{pmatrix} 0 \\ 2s^2(\boldsymbol{u}\cdot\boldsymbol{p})\boldsymbol{u} + (c^2 - s^2)\boldsymbol{p} + 2sc\boldsymbol{u}\times\boldsymbol{p} \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 2s^2(\boldsymbol{u}\cdot\boldsymbol{p})\boldsymbol{u} + (c^2 - s^2)\boldsymbol{p} + 2sc\boldsymbol{u}\times\boldsymbol{p} \end{pmatrix}\\
&= 4s^4(\boldsymbol{u}\cdot\boldsymbol{p})^2 + 2s^2(c^2 - s^2)(\boldsymbol{u}\cdot\boldsymbol{p})^2 + 0 + 2s^2(c^2 - s^2)(\boldsymbol{u}\cdot\boldsymbol{p})^2 + (c^2 - s^2)^2||\boldsymbol{p}||^2 +\\
&\quad 0 + 0 + 0 + 4s^2c^2(\boldsymbol{u}\times\boldsymbol{p})\cdot(\boldsymbol{u}\times\boldsymbol{p})\\
&= 4s^2c^2(\boldsymbol{u}\cdot\boldsymbol{p})^2 + (c^2 - s^2)^2||\boldsymbol{p}||^2 + 4s^2c^2\big((\boldsymbol{u}\cdot\boldsymbol{u})(\boldsymbol{p}\cdot\boldsymbol{p}) - (\boldsymbol{u}\cdot\boldsymbol{p})(\boldsymbol{p}\cdot\boldsymbol{u})\big)\\
&= (c^2 - s^2)^2||\boldsymbol{p}||^2 + 4s^2c^2||\boldsymbol{u}||^2||\boldsymbol{p}||^2\\
&= \left(\left(\cos^2\frac{\theta}{2} - \sin^2\frac{\theta}{2}\right)^2 + \left(2\sin\frac{\theta}{2}\cos\frac{\theta}{2}\right)^2\right)||\boldsymbol{p}||^2\\
&= (\cos^2\theta + \sin^2\theta)||\boldsymbol{p}||^2\\
&= ||\boldsymbol{p}||^2
\end{aligned}
$$

Using $||\boldsymbol{p}|| = ||p||$ we prove that

$$\boxed{||p_r|| = ||p||}$$

Now for the angle. Let us recall that for any two vectors $\boldsymbol{p} \in \mathbb{R}^d$ and $\boldsymbol{u} \in \mathbb{R}^d$, $d \geq 2$, one can decompose $\boldsymbol{p}$ into two vectors $\boldsymbol{p}_\parallel = (\boldsymbol{u}\cdot\boldsymbol{p})\boldsymbol{u}$ and $\boldsymbol{p}_\perp = \boldsymbol{p} - (\boldsymbol{u}\cdot\boldsymbol{p})\boldsymbol{u}$ that are respectively the collinear and orthogonal components of $\boldsymbol{p}$ with respect to $\boldsymbol{u}$. Rotating $\boldsymbol{p}$ about $\boldsymbol{u}$ through an angle θ is the same as rotating $\boldsymbol{p}_\perp$ in the plane whose normal is $\boldsymbol{u}$ through θ and then adding $\boldsymbol{p}_\parallel$. That means, in our case, that it is enough to check that the angle between $\boldsymbol{p_r}_\perp$ and $\boldsymbol{p}_\perp$ is θ to prove that the quaternions p_r and p form an angle θ in $\mathbb{R}^4$. First, we show that $\boldsymbol{p_r}_\parallel = \boldsymbol{p}_\parallel$:

$$
\begin{aligned}
\boldsymbol{p_r}_\parallel &= (\boldsymbol{u}\cdot\boldsymbol{p_r})\boldsymbol{u}\\
&= \big(2s^2(\boldsymbol{u}\cdot\boldsymbol{p})||\boldsymbol{u}||^2 + (c^2 - s^2)(\boldsymbol{p}\cdot\boldsymbol{u}) + 0\big)\boldsymbol{u}\\
&= \big((c^2 + s^2)\boldsymbol{u}\cdot\boldsymbol{p}\big)\boldsymbol{u}\\
&= (\boldsymbol{u}\cdot\boldsymbol{p})\boldsymbol{u}\\
&= \boldsymbol{p}_\parallel
\end{aligned}
$$

This result is expected: because $\boldsymbol{p}$ is rotated about $\boldsymbol{u}$, the collinear components $\boldsymbol{p}_{\parallel}$ and $\boldsymbol{p_r}_{\parallel}$ are the same. It implies that $||\boldsymbol{p_r}_{\perp}|| = ||\boldsymbol{p}_{\perp}||$ to keep the norm of the rotated vector constant (which is consistent with the fact that we focus on rotating $\boldsymbol{p}_{\perp}$ only, which keeps its length constant). We now compute that the angle between $\boldsymbol{p_r}_{\perp}$ and $\boldsymbol{p}_{\perp}$:

$$\begin{aligned}
\frac{\boldsymbol{p_r}_{\perp} \cdot \boldsymbol{p}_{\perp}}{||\boldsymbol{p_r}_{\perp}||||\boldsymbol{p}_{\perp}||} &= \frac{(\boldsymbol{p_r} - \boldsymbol{p_r}_{\parallel}) \cdot \boldsymbol{p}_{\perp}}{||\boldsymbol{p}_{\perp}||^2} \\
&= \frac{\big(\boldsymbol{p_r} - (\boldsymbol{u} \cdot \boldsymbol{p})\boldsymbol{u})\big) \cdot \boldsymbol{p}_{\perp}}{||\boldsymbol{p}_{\perp}||^2} \\
&= \frac{\big((2s^2 - 1)(\boldsymbol{u} \cdot \boldsymbol{p})\boldsymbol{u} + (c^2 - s^2)\boldsymbol{p} + 2sc(\boldsymbol{u} \times \boldsymbol{p})\big) \cdot \boldsymbol{p}_{\perp}}{||\boldsymbol{p}_{\perp}||^2} \\
&= \frac{\big(-\cos\theta(\boldsymbol{u} \cdot \boldsymbol{p})\boldsymbol{u} + \cos\theta\boldsymbol{p} + \sin\theta(\boldsymbol{u} \times \boldsymbol{p})\big) \cdot \boldsymbol{p}_{\perp}}{||\boldsymbol{p}_{\perp}||^2} \\
&= \frac{\big((\boldsymbol{p} - (\boldsymbol{u} \cdot \boldsymbol{p})\boldsymbol{u})\cos\theta + \sin\theta(\boldsymbol{u} \times \boldsymbol{p})\big) \cdot \boldsymbol{p}_{\perp}}{||\boldsymbol{p}_{\perp}||^2} \\
&= \frac{(\boldsymbol{p}_{\perp} \cdot \boldsymbol{p}_{\perp})\cos\theta + 0}{||\boldsymbol{p}_{\perp}||^2} \\
&= \cos\theta
\end{aligned}$$

Which proves that the angle between p_r and p is θ.
In a nutshell, we proved that the quaternion $p_r = qpq^{-1}$ has a zero scalar part and can therefore be interpreted as a 3D point $\boldsymbol{p_r}$ lying at the same distance to the origin as $\boldsymbol{p}$ does and forming an angle θ with the latter, i.e. we just rotated $\boldsymbol{p}$ through θ about $\boldsymbol{u}$.

D.2 DUAL QUATERNIONS

D.2.1 A CONCISE EXPRESSION

Following [50] and [22] we prove that any unit dual quaternion can be written down as:

$$\hat{q} = \left[\cos\frac{\hat{\theta}}{2}, \hat{\boldsymbol{u}}\sin\frac{\hat{\theta}}{2}\right]$$

where $\hat{\theta}$ is a dual angle and $\hat{\boldsymbol{v}}$ is a dual vector in $\mathbb{R}^3$.
Let $\hat{q} = q + \frac{1}{2}\epsilon\rho t q$ be a unit dual quaternion with $q = \left[\cos\frac{\theta}{2}, \boldsymbol{u}\sin\frac{\theta}{2}\right]$, $\boldsymbol{u} \in \mathbb{R}^3, \theta \in]-\pi, \pi]$ and $t = [0, \boldsymbol{t}]$, $\boldsymbol{t} \in \mathbb{R}^3$ such that $||\boldsymbol{u}|| = ||\boldsymbol{t}|| = 1$. Angle θ is the amount of rotation about axis $\boldsymbol{u}$ and ρ is the magnitude of the translation along $\boldsymbol{t}$. In the following let s denote $\sin\frac{\theta}{2}$ and $c = \cos\frac{\theta}{2}$.

Prerequisite

For a and $b \in \mathbb{R}$ we prove that:

$$\begin{aligned}
\cos(a + \epsilon b) &= \cos a - \epsilon b \sin a \\
\sin(a + \epsilon b) &= \sin a + \epsilon b \cos a
\end{aligned}$$

To that end recall that for all x:

$$\cos x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$$
$$\sin x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

Hence for a pure dual number $\hat{x} = \epsilon x$, since $\epsilon^n = 0$ when $n > 1$ one has:

$$\cos \hat{x} = 1$$
$$\sin \hat{x} = \hat{x}$$

Therefore:

$$\begin{aligned} \cos(a + \epsilon b) &= \cos(a)\cos(\epsilon b) - \sin(a)\sin(\epsilon b) \\ &= \cos a - \epsilon b \sin a \\ \sin(a + \epsilon b) &= \sin(a)\cos(\epsilon b) + \sin(\epsilon b)\cos(a) \\ &= \sin a + \epsilon b \cos a \end{aligned}$$

In a more general setting, by extending the definition of Taylor series to functions of dual numbers, it naturally follows that $f(a + \epsilon b) = f(a) + \epsilon b f'(a)$. Back to our dual quaternion:

$$\begin{aligned} \hat{q} &= q + \frac{1}{2}\epsilon\rho t q \\ &= [c, \boldsymbol{u}s] + \frac{1}{2}\epsilon\rho[-s\boldsymbol{t}\cdot\boldsymbol{u}, c\boldsymbol{t} + s\boldsymbol{t}\times\boldsymbol{u}] \\ &= \left[c - \epsilon\frac{\rho}{2}\boldsymbol{t}\cdot\boldsymbol{u}s, \boldsymbol{u}s + \epsilon\frac{\rho}{2}\left(c\boldsymbol{t} + s\boldsymbol{t}\times\boldsymbol{u}\right)\right] \end{aligned}$$

Let us rewrite the scalar part of the dual quaternion $\hat{q}$:

$$\begin{aligned} c - \epsilon\frac{\rho}{2}\boldsymbol{t}\cdot\boldsymbol{u}s &= \cos\left(\frac{\theta}{2} - \epsilon\left(\frac{\rho}{2}\boldsymbol{t}\cdot\boldsymbol{u}\right)\right) \\ &= \cos\left(\frac{\theta + \epsilon\rho\boldsymbol{t}\cdot\boldsymbol{u}}{2}\right) \end{aligned}$$

The scalar part of $\hat{q}$ invites us to introduce a dual angle $\hat{\theta} = \theta + \epsilon\rho\boldsymbol{t}\cdot\boldsymbol{u}$. Our goal is now to express the imaginary part of $\hat{q}$ using this dual angle $\hat{\theta}$. The dual part of $\hat{q}$ is:

$$\begin{aligned} \boldsymbol{u}s + \epsilon\frac{\rho}{2}(\boldsymbol{t}c + \boldsymbol{t}\times\boldsymbol{u}s) &= \boldsymbol{u}s + \epsilon\frac{\rho}{2}((\boldsymbol{t} - (\boldsymbol{t}\cdot\boldsymbol{u})\boldsymbol{u} + (\boldsymbol{t}\cdot\boldsymbol{u})\boldsymbol{u})c + \boldsymbol{t}\times\boldsymbol{u}s) \\ &= \left(s + \epsilon\frac{\rho}{2}(\boldsymbol{t}\cdot\boldsymbol{u})c\right)\boldsymbol{u} + \epsilon\frac{\rho}{2}\left((\boldsymbol{t} - (\boldsymbol{t}\cdot\boldsymbol{u})\boldsymbol{u})c + \boldsymbol{t}\times\boldsymbol{u}s\right) \\ &= \sin\frac{\hat{\theta}}{2}\boldsymbol{u} + \epsilon\frac{\rho}{2}\left((\boldsymbol{t} - (\boldsymbol{t}\cdot\boldsymbol{u})\boldsymbol{u})c + \boldsymbol{t}\times\boldsymbol{u}s\right) \end{aligned}$$

Assuming that $\theta \neq 0$ let us define:

$$\boldsymbol{u}_\epsilon = \frac{\rho}{2}\left((\boldsymbol{t} - (\boldsymbol{t}\cdot\boldsymbol{u})\boldsymbol{u})\cot\frac{\theta}{2} + \boldsymbol{t}\times\boldsymbol{u}\right)$$

so that the imaginary part of $\hat{q}$ can be rewritten as:

$$\boldsymbol{u}s + \epsilon\frac{\rho}{2}(\boldsymbol{t}c + \boldsymbol{t} \times \boldsymbol{u}s) = \sin\frac{\hat{\theta}}{2}\boldsymbol{u} + \epsilon\boldsymbol{u}_\epsilon \sin\frac{\theta}{2}$$

Remark that

$$\begin{aligned}\epsilon \sin\frac{\hat{\theta}}{2} &= \epsilon \sin\left(\frac{\theta + \epsilon\rho\boldsymbol{t}\cdot\boldsymbol{u}}{2}\right)\\ &= \epsilon\left(\sin\frac{\theta}{2} + \epsilon\frac{\rho}{2}\rho\boldsymbol{t}\cdot\boldsymbol{u}\cos\frac{\theta}{2}\right)\\ &= \epsilon\sin\frac{\theta}{2}\end{aligned}$$

Hence, denoting $\hat{\boldsymbol{u}} = \boldsymbol{u} + \epsilon\boldsymbol{u}_\epsilon$, we get:

$$\boldsymbol{u}s + \epsilon\frac{\rho}{2}(\boldsymbol{t}c + \boldsymbol{t} \times \boldsymbol{u}s) = \sin\frac{\hat{\theta}}{2}\hat{\boldsymbol{u}}$$

and:

$$\boxed{\hat{q} = \left[\cos\frac{\hat{\theta}}{2}, \hat{\boldsymbol{u}}\sin\frac{\hat{\theta}}{2}\right]}$$

D.2.2 UNIT DUAL QUATERNIONS AND PLÜCKER COORDINATES

We prove here that for a unit dual quaternion $\hat{q}$ written in the form $\hat{q} = [\cos\frac{\hat{\theta}}{2}, \hat{\boldsymbol{u}}\sin\frac{\hat{\theta}}{2}]$ with $\hat{\theta} = \theta_0 + \epsilon\theta_\epsilon$ and $\hat{\boldsymbol{u}} = \boldsymbol{u}_0 + \epsilon\boldsymbol{u}_\epsilon$, then the tuple $[\boldsymbol{u}_0, \boldsymbol{u}_\epsilon]$ defines the Plücker coordinates of a line. Bearing in mind the previous SECTION 5.1.8.3, we see that it is enough to show that

$$||\hat{q}|| = 1 \Leftrightarrow \begin{cases} ||\boldsymbol{u}_0|| = 1 \\ \boldsymbol{u}_0 \cdot \boldsymbol{u}_\epsilon = 0 \end{cases}$$

Let us develop $\hat{q}$:

$$\begin{aligned}\hat{q} &= [\cos\frac{\hat{\theta}}{2}, \hat{\boldsymbol{u}}\sin\frac{\hat{\theta}}{2}]\\ &= [\cos\frac{\hat{\theta}}{2}, (\boldsymbol{u}_0 + \epsilon\boldsymbol{u}_\epsilon)(\sin\frac{\theta_0}{2} + \epsilon\frac{\theta_\epsilon}{2}\cos\frac{\theta_0}{2})]\\ &= [\cos\frac{\theta_0}{2} - \epsilon\frac{\theta_\epsilon}{2}\sin\frac{\theta_0}{2}, \boldsymbol{u}_0\sin\frac{\theta_0}{2} + \epsilon\boldsymbol{u}_0\frac{\theta_\epsilon}{2}\cos\frac{\theta_0}{2} + \epsilon\boldsymbol{u}_\epsilon\sin\frac{\theta_0}{2}]\\ &= [\cos\frac{\theta_0}{2}, \boldsymbol{u}_0\sin\frac{\theta_0}{2}] + \epsilon[-\frac{\theta_\epsilon}{2}\sin\frac{\theta_0}{2}, \boldsymbol{u}_0\frac{\theta_\epsilon}{2}\cos\frac{\theta_0}{2} + \boldsymbol{u}_\epsilon\sin\frac{\theta_0}{2}]\\ &= q_0 + \epsilon q_\epsilon\end{aligned}$$

With $q_0 = [s_{q_0}, \boldsymbol{v}_{q_0}]$, $q_\epsilon = [s_{q_\epsilon}, \boldsymbol{v}_{q_\epsilon}]$ where $s_{q_0} = \cos\frac{\theta_0}{2}$, $\boldsymbol{v}_{q_0} = \boldsymbol{u}_0\sin\frac{\theta_0}{2}$, $s_{q_\epsilon} = -\frac{\theta_\epsilon}{2}\sin\frac{\theta_0}{2}$ and $\boldsymbol{v}_{q_\epsilon} = \boldsymbol{u}_0\frac{\theta_\epsilon}{2}\cos\frac{\theta_0}{2} + \boldsymbol{u}_\epsilon\sin\frac{\theta_0}{2}$. Recall that by definition:

$$||\hat{q}|| = 1 \Leftrightarrow \begin{cases} ||q_0|| = 1 \\ s_{q_0}s_{q_\epsilon} + \boldsymbol{v}_{q_0}\cdot\boldsymbol{v}_{q_\epsilon} = 0 \end{cases}$$

Since this system is trivially met when $\sin\frac{\theta_0}{2} = 0$ we assume it is not the case. We first prove that $||q_0|| = 1 \Leftrightarrow ||\boldsymbol{u}_0|| = 1$:

$$\begin{aligned}
||q_0|| = 1 &\Leftrightarrow s_{q_0}^2 + \boldsymbol{v}_{q_0} \cdot \boldsymbol{v}_{q_0} = 1 \\
&\Leftrightarrow \cos^2\frac{\theta_0}{2} + \boldsymbol{u}_0 \cdot \boldsymbol{u}_0 \sin^2\frac{\theta_0}{2} \underset{=}{1} \\
&\Leftrightarrow \sin^2\frac{\theta_0}{2}(\boldsymbol{u}_0 \cdot \boldsymbol{u}_0 - 1) = 0 \\
&\Leftrightarrow \boldsymbol{u}_0 \cdot \boldsymbol{u}_0 = 1 \\
&\Leftrightarrow ||\boldsymbol{u}_0|| = 1
\end{aligned}$$

Using $||\boldsymbol{u}_0|| = 1$ we now show that $s_{q_0}s_{q_\epsilon} + \boldsymbol{v}_{q_0} \cdot \boldsymbol{v}_{q_\epsilon} = 0 \Leftrightarrow \boldsymbol{u}_0 \cdot \boldsymbol{u}_\epsilon = 0$

$$\begin{aligned}
s_{q_0}s_{q_\epsilon} + \boldsymbol{v}_{q_0} \cdot \boldsymbol{v}_{q_\epsilon} = 0 &\Leftrightarrow -\theta_\epsilon \sin\frac{\theta_0}{2}\cos(\frac{\theta_0}{2}) + (\boldsymbol{u_0}\sin\frac{\theta_0}{2}) \cdot (\boldsymbol{u}_0\frac{\theta_\epsilon}{2}\cos\frac{\theta_0}{2} + \boldsymbol{u}_\epsilon \sin\frac{\theta_0}{2}) = 0 \\
&\Leftrightarrow \frac{\theta_\epsilon}{2}\sin\frac{\theta_0}{2}\cos\frac{\theta_0}{2} + \frac{\theta_\epsilon}{2}\sin\frac{\theta_0}{2}\cos\frac{\theta_0}{2}\boldsymbol{u}_0 \cdot \boldsymbol{u}_0 + \boldsymbol{u_0} \cdot \boldsymbol{u}_\epsilon \sin^2\frac{\theta_0}{2} = 0 \\
&\Leftrightarrow \boldsymbol{u}_0 \cdot \boldsymbol{u}_\epsilon = 0
\end{aligned}$$

We have proven in this section that a unit dual quaternion contains the information about a unique line whose orientation is $\boldsymbol{u_0}$ and going through the point $\boldsymbol{u_0} \times \boldsymbol{u}_\epsilon$:

$$||\hat{q}|| = 1 \Leftrightarrow [\boldsymbol{u}_0, \boldsymbol{u}_\epsilon] \text{ are the Plücker coordinates of a line}$$

APPENDIX E

GAUSSIAN PROCESS

The figures presented in this appendix are heavily inspired from [84]. The most basic kernel that can be used in a Gaussian process is the white noise kernel: each random variable is completely uncorrelated from the others, as shown on FIGURE E.1, where the kernel $\kappa(x, 0)$ is a Dirac function at 0 (top left), the covariance matrix is the identity matrix (top right), and three independent samplings are shown on the bottom.

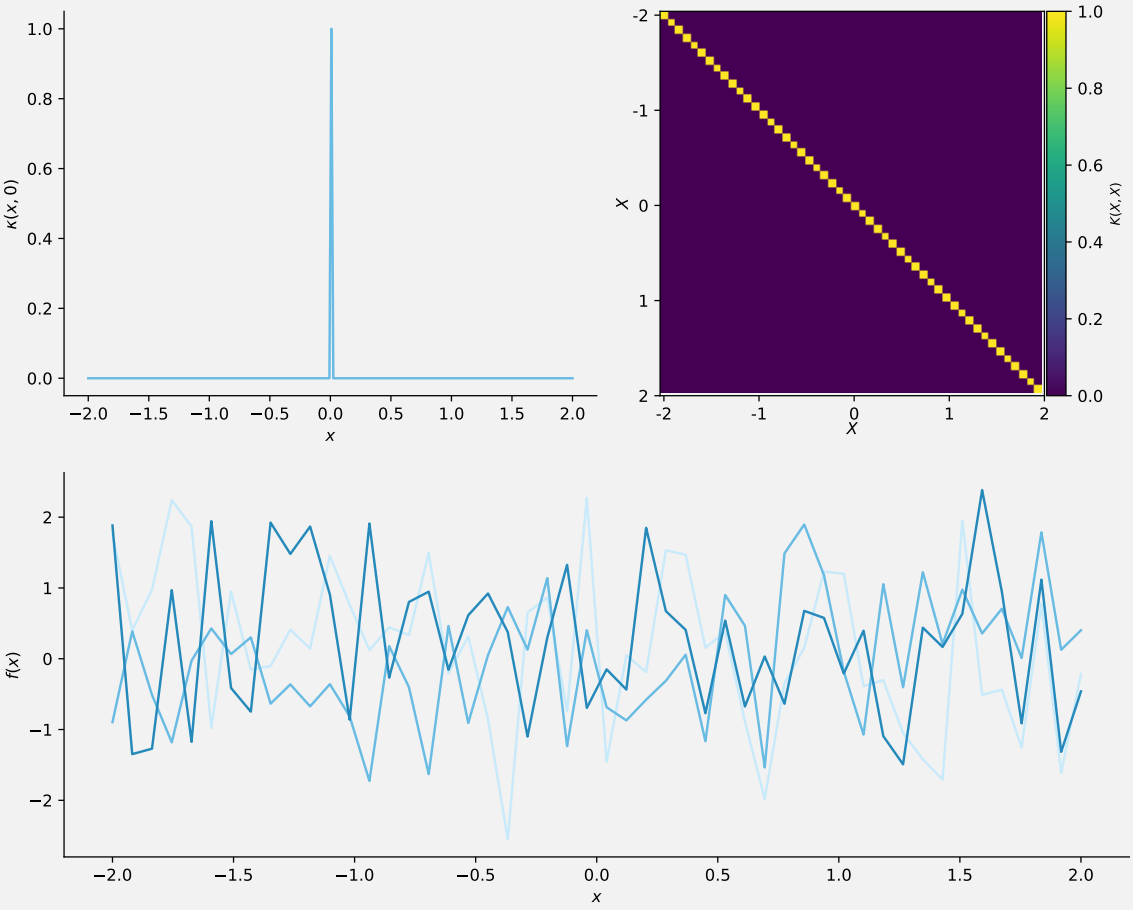


Figure E.1| White noise kernel.

The exponentiated quadratic kernel is given by:

$$\kappa(x, y) = a^2 \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right)$$

The greater σ the more correlated are far away random variables, as shown on FIGURE E.2. The amplitude parameter a simply scales the functions.

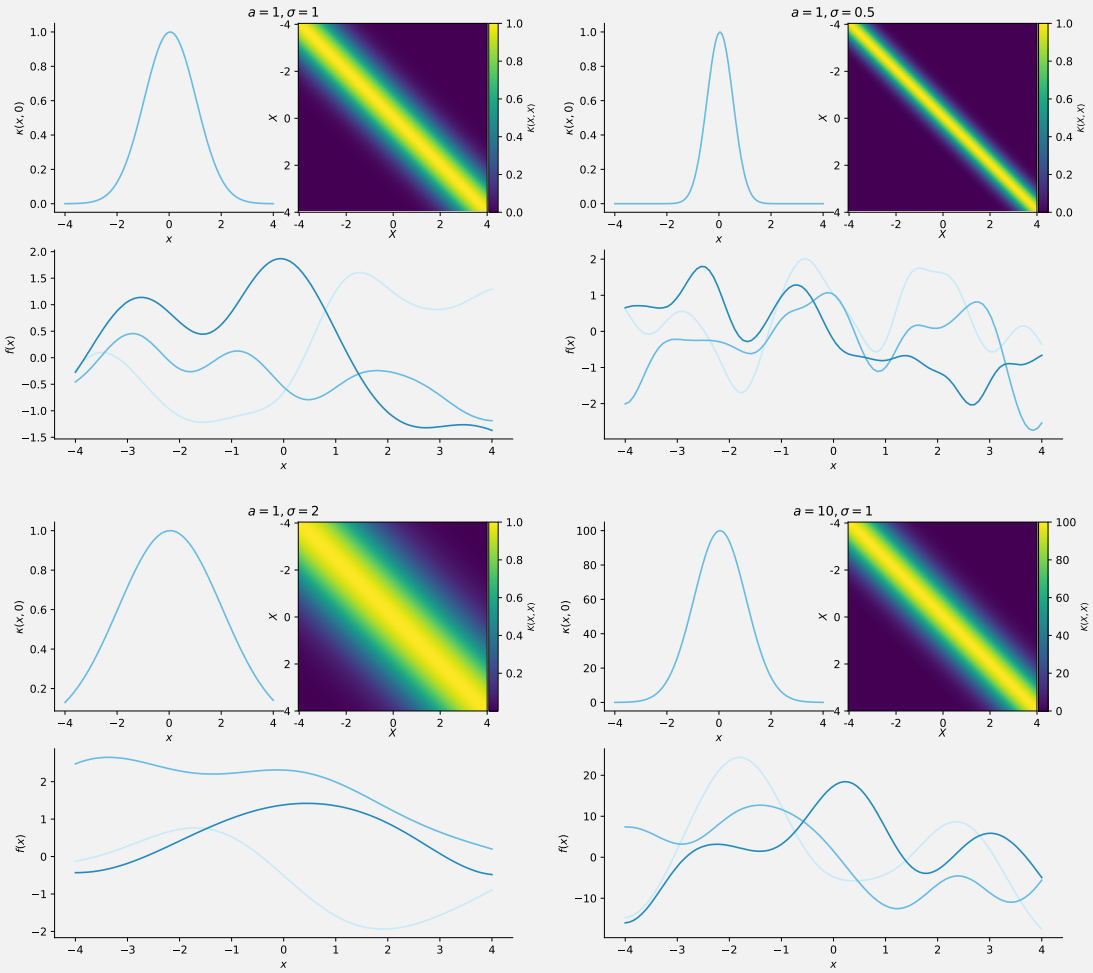
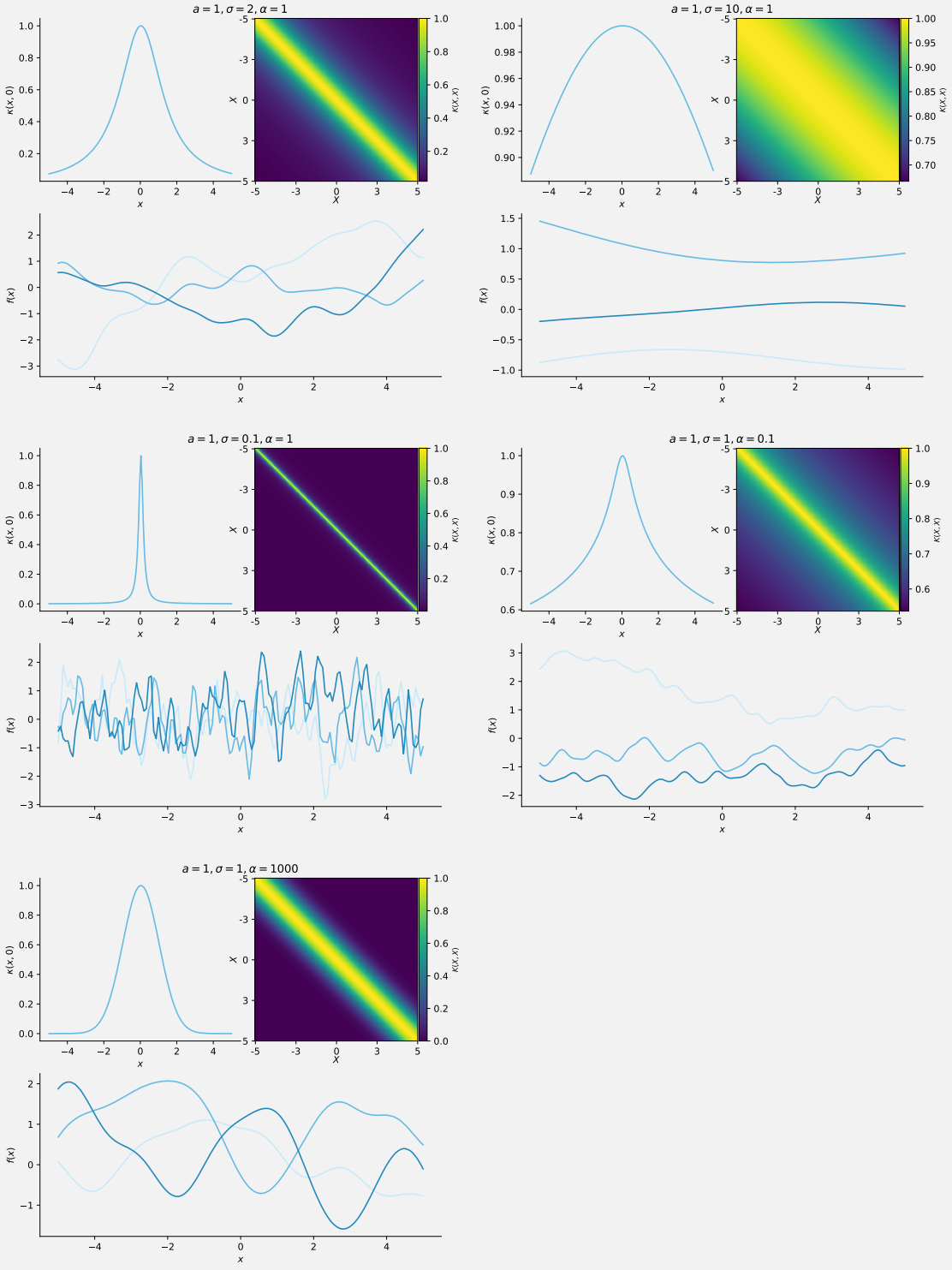


Figure E.2| Several exponentiated quadratic kernels.

The rational quadratic kernel is given by

$$\kappa(x, y) = a^2 \left(1 + \frac{||x - y||^2}{2\alpha\sigma^2}\right)^{-\alpha}$$

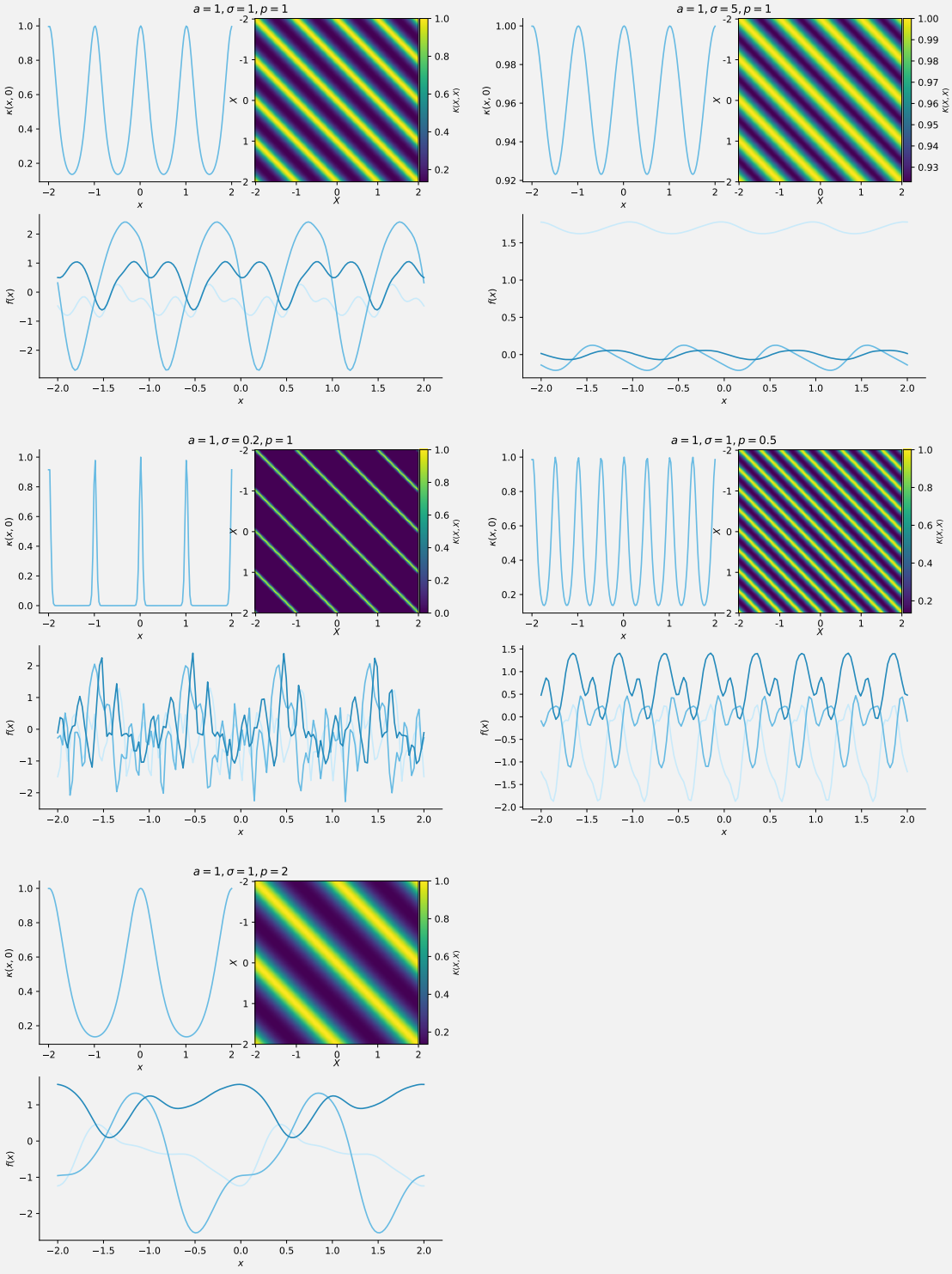
Parameter σ influences the magnitude of the correlation between far away values while parameter α influences the steepness of the correlation.



Periodic kernels may also be defined. For instance:

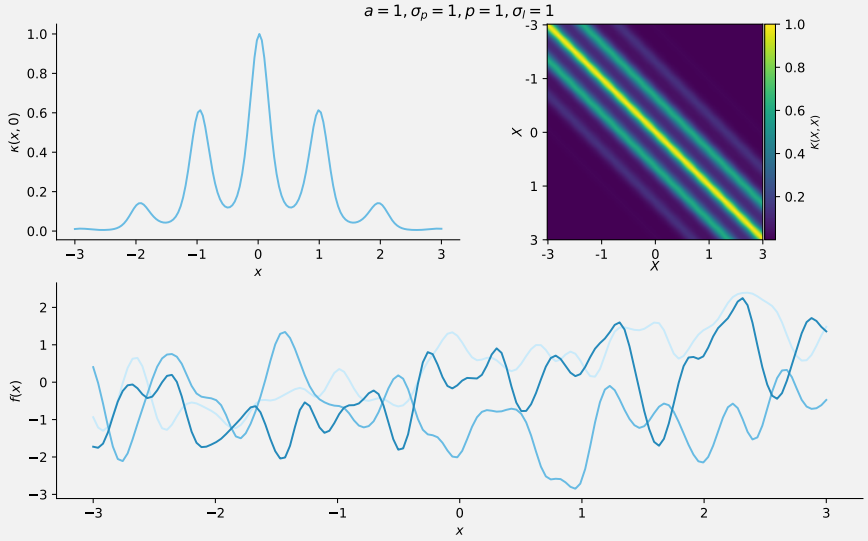
$$\kappa(x, y) = a^2 \exp\left(\frac{-2}{\sigma^2} \sin^2\left(\pi \frac{|x - y|}{p}\right)\right)$$

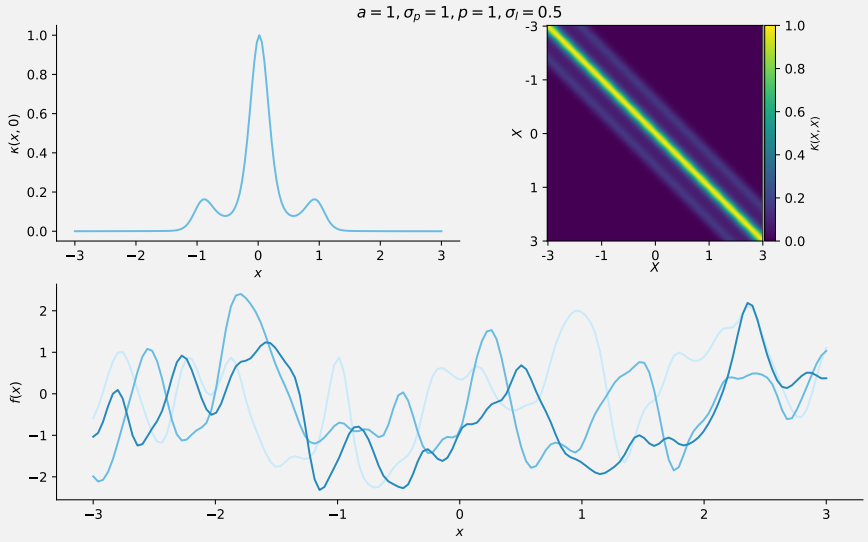
Parameter p plays on the period, i.e. the distance between two highly correlated random values.

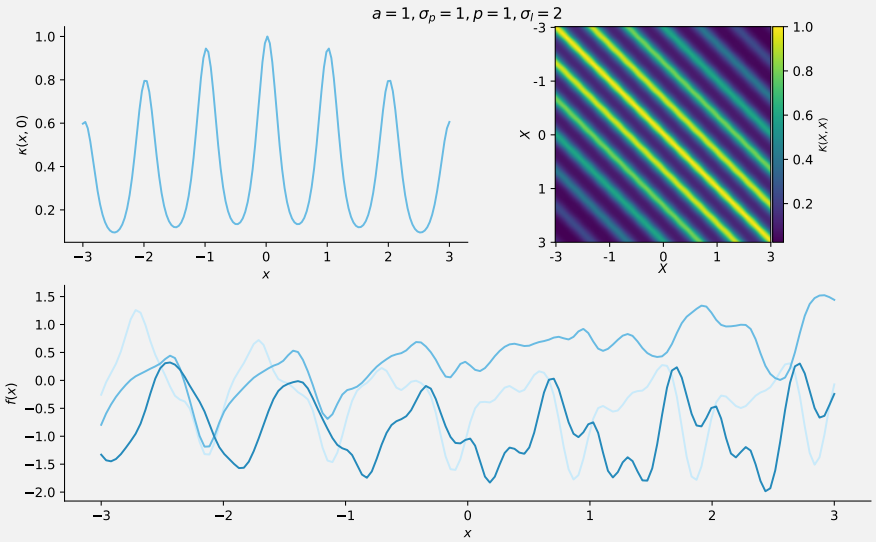


Finally it is also possible to combine kernels. Here we define a damped periodic kernel by multiplying a periodic kernel with and exponentiated quadratic one:

$$\kappa(x, y) = a^2 \exp\left(\frac{-2}{\sigma_p^2} \sin^2\left(\pi \frac{|x - y|}{p}\right)\right) \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right)$$


$a = 1, \sigma_p = 1, p = 1, \sigma_l = 1$
$\kappa(x, 0)$
x
X
$\kappa(X, X)$
$f(x)$


$a = 1, \sigma_p = 1, p = 1, \sigma_l = 0.5$
$\kappa(x, 0)$
x
X
$\kappa(X, X)$
$f(x)$


$a = 1, \sigma_p = 1, p = 1, \sigma_l = 2$
$\kappa(x, 0)$
x
X
$\kappa(X, X)$
$f(x)$

APPENDIX F

CONES OF FREEDOM OF IMPERFECT MESHES

On FIGURES F.1, F.2 and F.3 the small figures on the top left displays the generating rays of $\mathcal{C}_{\mathcal{M}^\epsilon}$ in the 15 cartesian planes, same as FIGURE 5.26. On the top middle is shown the 2D PCA performed on the rays, as well as the convex hull of the projection in red. Bottom left and middle: views of the mesh $\mathcal{M}^\epsilon$ with in green the prescribed motion (pure rotation) and in colours the unit dual quaternions defining the cone; the brighter the red the more $\theta_0 > \theta_\epsilon$, the deeper the blue the more $\theta_\epsilon > \theta_0$. Right: plots showing the evolution of the area and volume of the convex hull of the cone as well as the number of vertices, in both standard and log scale. They were calculated for various family of $\mathcal{M}^\epsilon$, for various number of prescribed snap face-vertex pairs. The highlighted red points correspond to the mesh on the figure. It shows that for low value of snaps (3 or less), the cones of freedom of the imperfect meshes $\mathcal{C}_{\mathcal{M}^\epsilon}$ vary wildly: there is a relatively high spread of the values of the three measured metrics. For more than 3 snaps, the variation is of much lower magnitude, implying that even the imperfect meshes all have a rather small cone of freedom.

These three figures, combined with FIGURE 5.34, show that the evolution of the shape of the cones $\mathcal{C}_{\mathcal{M}^\epsilon}$ greatly depends on the geometrical features of the perfect mesh $\mathcal{M}$. While the cones abruptly shrink in size for the mesh with four dents on FIGURE F.1 and the random one on F.3, the shrinking is slower of the meshes with one and two dents on FIGURES 5.34, F.2. This difference in behaviour imply that we cannot give any guidelines on what should be the number of face-vertex pairs prescribed to snap to reduce the cones to a small size: it depends too much on the geometry of the mesh.
On FIGURE F.1, the prescribed $\hat{q}$ encodes a vertical screw motion, shown on the right of FIGURE 5.24.
On FIGURE F.2, the prescribed $\hat{q}$ encodes a pure rotation, shown on the left of FIGURE 5.24.
On FIGURE F.3, the prescribed $\hat{q}$ encodes a pure rotation, shown on the left of FIGURE 5.24. The mesh was randomly initialised.

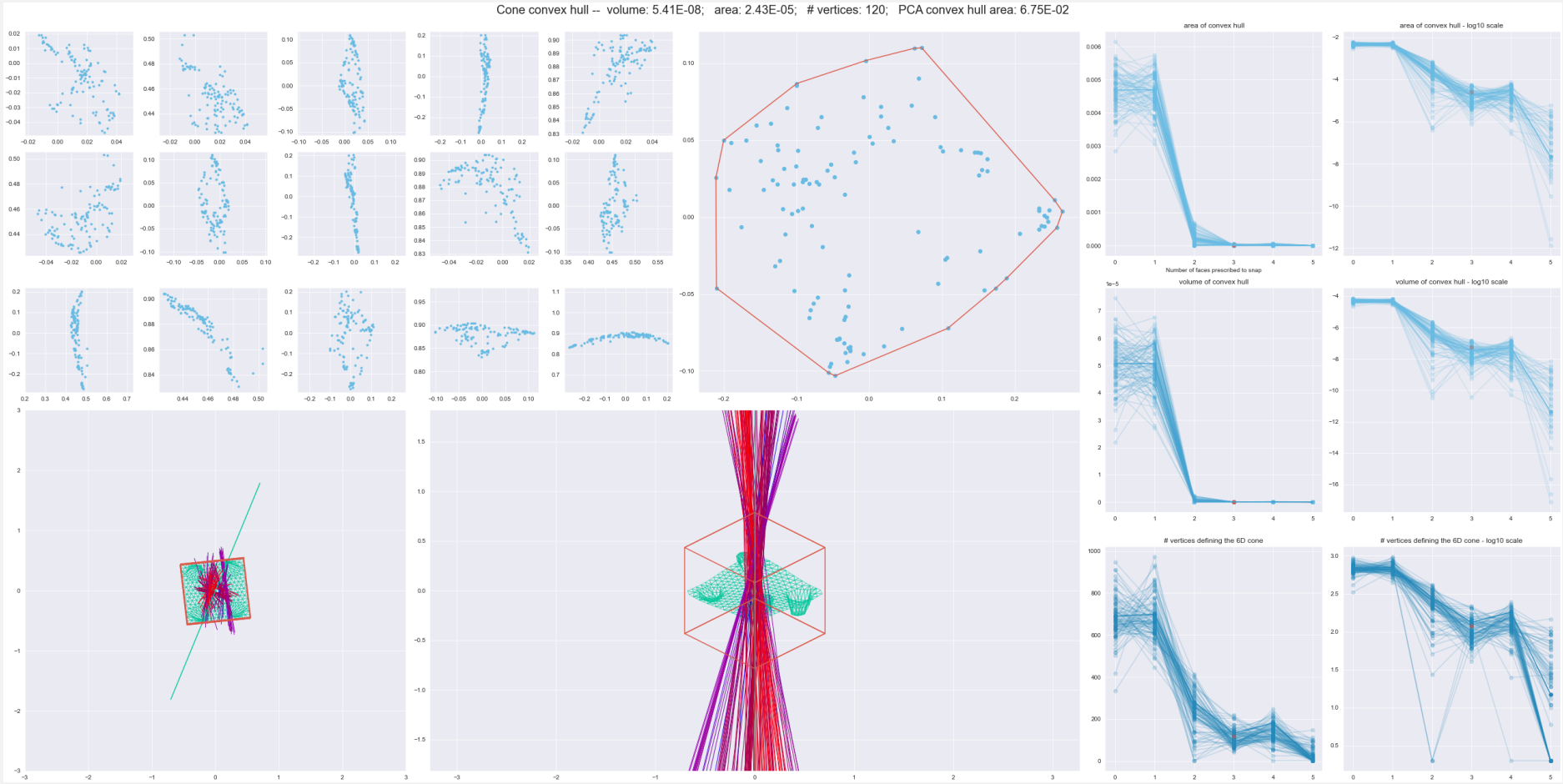


Figure F.1| See text for a description.

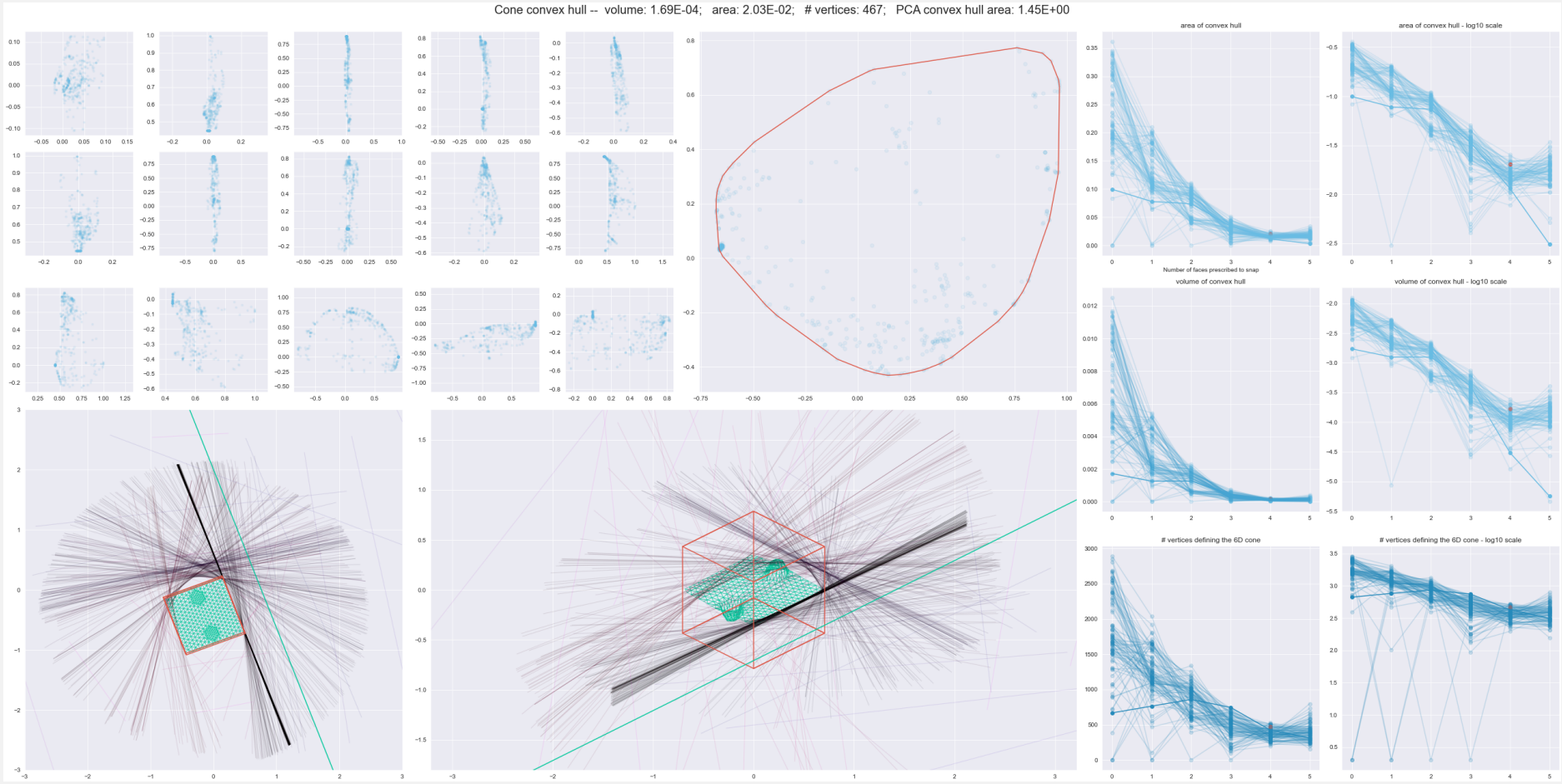


Figure F.2| See text for a description.

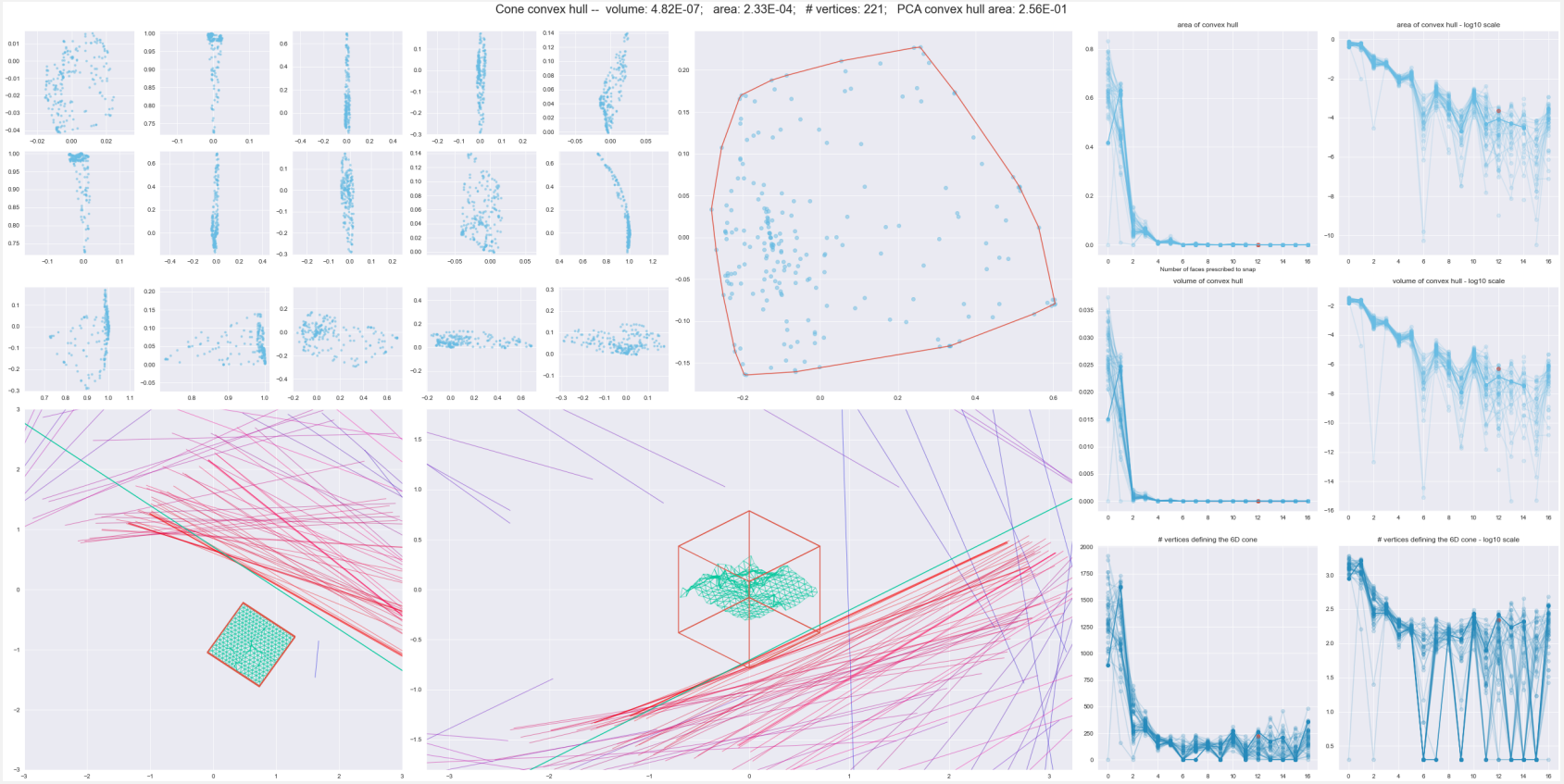


Figure F.3| See text for a description.

BIBLIOGRAPHY

1 C'est pas sorcier - France 3. *C'est pas sorcier - Le Grand Palais épate la galerie*. Youtube. 2013. URL: https://youtu.be/9huxiit6bME?t=1189.

2 3Blue1Brown. *Visualizing quaternions (4d numbers) with stereographic projection*. Youtube. 2018. URL: https://www.youtube.com/watch?v=d4EgbgTm0Bg.

3 Lior Aharoni, Ido Bachelet, and Josephine Carstensen. "Topology optimization of rigid interlocking assemblies". In: *Computers & Structures* 250 (July 2021), p. 106521.

4 Ergun Akleman et al. "Generalized Abeille Tiles: Topologically Interlocked Space-Filling Shapes Generated Based on Fabric Symmetries". In: *Computers and Graphics* 89 (May 2020).

5 Ergun Akleman et al. "Generalized Abeille tiles: Topologically interlocked space-filling shapes generated based on fabric symmetries". In: *Computers & Graphics* 89 (2020), pp. 156–166. ISSN: 0097-8493. URL: https://www.sciencedirect.com/science/article/pii/S0097849320300674.

6 Aleksandra Apolinarska et al. "Robotic assembly of timber joints using reinforcement learning". In: *Automation in Construction* 125 (May 2021), p. 103569.

7 *Approvisionnement pétrolier futur de l'Union Européenne: état des réserves et perspectives de production des principaux pays fournisseurs*. The Shift Project. May 2021. URL: https://theshiftproject.org/wp-content/uploads/2021/05/Approvisionnement-petrolier-futur-de-lUE_Shift-Project_Mai-2021_RAPPORT-COMPLET.pdf.

8 Matthieu Auzanneau. "Métaux critiques, charbon, gaz, pétrole : nous entrons dans les récifs". In: *Le Monde - Oil Man (blog)* (Sept. 12, 2021). URL: https://www.lemonde.fr/blog/petrole/2021/10/12/metaux-critiques-charbon-gaz-petrole-nous-entrons-dans-les-recifs/#more-13013.

9 Marine Bagneris et al. "Structural Morphology Issues in Conceptual Design of Double Curved Systems". In: *International Journal of Space Structures* 23 (June 2008), pp. 79–87.

10 Claude Berge. *Graphes et hypergraphes*. Dunod, 1970.

11 Karthik Bharath and Sebastian Kurtek. "Analysis of shape data: From landmarks to elastic curves". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 12 (Jan. 2020).

12 Thomas Bock and Thomas Linner. *Robot-Oriented Design: Design and Management Tools for the Deployment of Automation and Robotics in Construction*. Cambridge University Press, 2015.

13 A.S. Bolles. *Industrial History of the United States: From the Earliest Settlements to the Present Time : Being a Complete Survey of American Industries ... : Together with a Description of Canadian Industries*. Gilded age. Henry Bill Publishing Company, 1881. URL: https://books.google.fr/books?id=Nmnn08iu3JYC.

14 Jorge M. Branco and Thierry Descamps. "Analysis and strengthening of carpentry joints". In: *Construction and Building Materials* 97 (2015). Special Issue: Reinforcement of Timber Structures, pp. 34–47. ISSN: 0950-0618. URL: https://www.sciencedirect.com/science/article/pii/S0950061815006029.

15 Jorge Manuel Branco, Maurizio Piazza, and Paulo J.S. Cruz. "Experimental evaluation of different strengthening techniques of traditional timber connections". In: *Engineering Structures* 33.8 (2011), pp. 2259–2270. ISSN: 0141-0296. URL: https://www.sciencedirect.com/science/article/pii/S0141029611001490.

16 Matthias Braun et al. "Calibration and Validation of a Linear-Elastic Numerical Model for Timber Step Joints Based on the Results of Experimental Investigations". In: *Materials* 15.5 (2022), p. 1639.

17 M. Brocato and L. Mondardini. "A new type of stone dome based on Abeille's bond". In: *International Journal of Solids and Structures* 49.13 (2012), pp. 1786–1801.

18 Edvard P. G. Bruun et al. "Human–robot collaboration: a fabrication framework for the sequential design and construction of unplanned spatial structures". In: *Digital Creativity* 31 (2020), pp. 320–336.

19 Guy Chazan. "German companies halt production to cope with rising energy prices". In: *Financial Times* (Mar. 10, 2022). URL: https://www.ft.com/content/d0d46712-6234-4d24-bbed-924a00dd0ca9.

20 Paolo Cignoni et al. "Field-Aligned Mesh Joinery". In: *ACM Trans. Graph.* 33.1 (Feb. 2014). ISSN: 0730-0301. URL: https://doi.org/10.1145/2537852.

21 Clifford. "Preliminary Sketch of Biquaternions". In: *Proceedings of The London Mathematical Society* (1873), pp. 381–395.

22 Konstantinos Daniilidis. "Hand-Eye Calibration Using Dual Quaternions". In: *The International Journal of Robotics Research* 18.3 (Mar. 1999), pp. 286–298. ISSN: 0278-3649, 1741-3176. URL: http://journals.sagepub.com/doi/10.1177/02783649922066213 (visited on 01/09/2020).

23 *Déchets chiffres-clés (Waste, key figures)*. Accessed: 2021-04-09. 2017. URL: https://www.ademe.fr/sites/default/files/assets/documents/dechets-chiffrescles-edition2020-3-010692.pdf.

24 Edsger W Dijkstra et al. "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1 (1959), pp. 269–271.

25 *DiXite - Digital Construction Site*. I-SITE FUTURE. URL: https://dixite.future-isite.fr/.

26 Lee Djumas et al. "Deformation mechanics of non-planar topologically interlocked assemblies with structural hierarchy and varying geometry". In: *Scientific Reports* 7 (Sept. 2017).

27 Arcady Dyskin et al. "Fracture Resistant Structures Based on Topological Interlocking with Non-planar Contacts". In: *Advanced Engineering Materials* 5 (Mar. 2003), pp. 116–119.

28 Yuri Estrin, Arcady Dyskin, and E. Pasternak. "Topological Interlocking as a Material Design Concept". In: *Materials Science and Engineering: C* 31 (Aug. 2011), pp. 1189–1194.

29 Ursula Frick, Tom Van Mele, and Philippe Block. "Decomposing Three-Dimensional Shapes into Self-supporting, Discrete-Element Assemblies". In: (2015). Ed. by Mette Ramsgaard Thomsen et al., pp. 187–201. URL: http://link.springer.com/10.1007/978-3-319-24208-8_16 (visited on 10/17/2019).

30 Chi-Wing Fu et al. "Computational Interlocking Furniture Assembly". In: *ACM Transactions on Graphics* 34.4 (July 2015). ISSN: 0730-0301.

31 Paul A Gagniuc. *Markov Chains: From Theory to Implementation and Experimentation*. ISBN: 978-1-119-38759-6 978-1-119-38755-8. Hoboken, NJ, USA: John Wiley & Sons, Inc., June 22, 2017. (Visited on 09/17/2021).

32 J. Gallon. *Machines et inventions approuvées par l'Académie royale des Sciences*. Vol. 1. *Mémoire concernant la voûte plate inventée par M. Abeille*. Académie royale des Sciences, 1735, pp. 159–162.

33 J. Gallon. *Machines et inventions approuvées par l'Académie royale des Sciences*. Vol. 1. *Voûte plate inventée par le père Sébastien de l'Académie Royale des Sciences*. Académie royale des Sciences, 1735, pp. 163–165.

34 B. Gfeller et al. "Wood bonding by vibrational welding". In: *Journal of Adhesion Science and Technology* 17.11 (2003), pp. 1573–1589. eprint: https://doi.org/10.1163/156856103769207419. URL: https://doi.org/10.1163/156856103769207419.

35 S. Giedion, Reserve Affairs United States. Office of the Assistant Secretary of Defense (Manpower, and Logistics). *Space, Time and Architecture: The Growth of a New Tradition*. Charles Eliot Norton Lectures: The Charles Eliot Norton Lectures. Harvard University Press, 1967. ISBN: 9780674830400. URL: https://books.google.fr/books?id=ZHZnmKxkGMwC.

36 Pierre Gilibert, Romain Mesnil, and Olivier Baverel. "Rule-based generative design of translational and rotational interlocking assemblies". In: *Automation in Construction* 135 (2022), p. 104142. ISSN: 0926-5805. URL: https://www.sciencedirect.com/science/article/pii/S0926580522000152.

37 Elsé Glahn. "Chinese building standards in the 12th century". In: *Scientific American* 244 (1981), pp. 162–173.

38 Julien Glath et al. "Thinking and Designing Reversible Structures with Non-sequential Assemblies". In: Sept. 2022, pp. 249–259. ISBN: 978-3-031-13248-3.

39 Tristan Gobin et al. "Robot-oriented design for complex timber structures". In progress. PhD thesis. 2022.

40 H. Greeley. *The Great Industries of the United States: Being an Historical Summary of the Origin, Growth, and Perfection of the Chief Industrial Arts of this Country*. J.B. Burr & Hyde, 1872. URL: https://books.google.fr/books?id=KSEaAAAAYAAJ.

41 D. Halperin, J.-C. Latombe, and R. H. Wilson. "A General Framework for Assembly Planning: The Motion Space Approach". In: *Algorithmica* 26.3 (Mar. 1, 2000), pp. 577–601. ISSN: 0178-4617, 1432-0541. URL: http://link.springer.com/10.1007/s004539910025 (visited on 11/08/2019).

42 Volker Helm et al. "Additive robotic fabrication of complex timber structures". In: 2015.

43 Yasua Nakahara Hideo Sato. *The Complete Japanese Joinery*. Hartley and Marks Publishers, 2000.

44 Kristian Hildebrand, Bernd Bickel, and Marc Alexa. "crdbrd: Shape Fabrication by Sliding Planar Slices". In: *Computer Graphics Forum* 31 (May 2012), pp. 583–592.

45 Dunham Jackson. "The Instantaneous Motion of a Rigid Body". In: *The American Mathematical Monthly* (Dec. 1942), 661 to 667.

46 Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. "Robust Inside-Outside Segmentation Using Generalized Winding Numbers". In: *ACM Trans. Graph.* 32.4 (July 2013). ISSN: 0730-0301. URL: https://doi.org/10.1145/2461912.2461916.

47 Ian Jermyn et al. "Elastic Shape Analysis of Three-Dimensional Objects". In: *Synthesis Lectures on Computer Vision* 7 (Sept. 2017), pp. 1–185.

48 Pablo Jimenez. "Survey on assembly sequencing: A combinatorial and geometrical perspective". In: *Journal of Intelligent Manufacturing* 24 (Apr. 2011), pp. 1–16.

49 Craig Jones and Geoffrey Hammond. "Embodied energy and carbon in construction materials". In: *Proceedings of The Ice - Energy* 161 (Jan. 2008), pp. 87–98.

50 Ladislav Kavan et al. "Dual Quaternions for Rigid Transformation Blending". In: (2006), p. 10.

51 Anja Kunic et al. "Design and assembly automation of the Robotic Reversible Timber Beam". In: *Automation in Construction* 123 (2021), p. 103531.

52 A.J.D. Lambert. "Disassembly sequencing: A survey". In: *International Journal of Production Research* 41 (Nov. 2003), pp. 3721–3759.

53 Maria Larsson et al. "Tsugite: Interactive Design and Fabrication of Wood Joints". In: UIST '20 (2020). pp. 317–327.

54 *Le chantier du grand palais - dossier pédagogique du grand palais n°2*. Accessed: 2022-08-03. 2013. URL: https://www.grandpalais.fr/sites/default/files/user_images/30/dossier_pedago_chantier_grand_palais.pdf.

55 Kui-Yip Lo, Chi-Wing Fu, and Hongwei Li. "3D Polyomino Puzzle". In: *ACM Trans. Graph.* 28.5 (Dec. 2009), pp. 1–8. ISSN: 0730-0301. URL: https://doi.org/10.1145/1618452.1618503.

56 Vianney Loing et al. "Free-form structures from topologically interlocking masonries". In: *Automation in Construction* 113 (2020), p. 103117. ISSN: 0926-5805. URL: https://www.sciencedirect.com/science/article/pii/S0926580519310957.

57 Linjie Luo et al. "Chopper: Partitioning Models into 3D-Printable Parts". In: *ACM Transactions on Graphics* 31.6 (Nov. 2012). ISSN: 0730-0301.

58 Koliann Mam. "Exploration structurelle et environnementale des ouvrages en bois de grande portée". Theses. École des Ponts ParisTech, Nov. 2021. URL: https://pastel.archives-ouvertes.fr/tel-03572887.

59 J.M. McCarthy. *An Introduction to Theoretical Kinematics*. The MIT Press, 1990.

60 David I. Armstrong McKay et al. "Exceeding 1.5°C global warming could trigger multiple climate tipping points". In: *Science* 377.6611 (2022), eabn7950. eprint: https://www.science.org/doi/pdf/10.1126/science.abn7950. URL: https://www.science.org/doi/abs/10.1126/science.abn7950.

61 Romain Mesnil et al. "Form finding of nexorades using the translations method". In: *Automation in Construction* (2018).

62 Abhijeet Mishra et al. "Land use change and carbon emissions of a transformation to timber cities". In: *Nature Communications* 13 (2022).

63 Camilo Mora et al. "Global risk of deadly heat". In: *Nature Climate Change* 7 (June 2017).

64 Julieta Moradei et al. "Structural Characterization of Traditional Moment-Resisting Timber Joinery". In: (July 2018).

65 *Natural gas supply statistics*. eurostats - statistics explained. 2022. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Natural_gas_supply_statistics.

66 *Neue Geometrie des Raumes gegründet auf die Betrachtung der geraden Linie als Raumelement Abth.2*. ger. 1869. URL: http://eudml.org/doc/203056.

67 *Neutralité & Bâtiment (neutrality and construction)*. Accessed: 2021-04-09. 2016. URL: https://ile-de-france.ademe.fr/sites/default/files/neutralite-carbone-batiment.pdf.

68 *Optimizing structural building elements in metal by using additive manufacturing*. Accessed: 2021-12-07. 2015. URL: https://www.ingentaconnect.com/content/iass/piass/2015/00002015/00000002/art00016.

69 Robin Oval et al. "Feature-based Topology Finding of Patterns for Shell Structures". In: *Automation in Construction* (Feb. 2019).

70 Stefana Parascho et al. "Cooperative Fabrication of Spatial Metal Structures". In: 2017.

71 Stefana Parascho et al. "Robotic vault: a cooperative robotic assembly method for brick vault construction". In: 2020.

72 Stefana Parascho et al. "LightVault: A Design and Robotic Fabrication Method for Complex Masonry Structures". In: 2021.

73 Tsvetana Paraskova. “Oil Major Total Sees 10 Million Bpd Supply Gap In 2025”. In: *Oilprice* (Feb. 21, 2021). URL: https://oilprice.com/Energy/Energy-General/Oil-Major-Total-Sees-10-Million-Bpd-Supply-Gap-In-2025.html.

74 Dr. Mary Neighbour Parent. *JAANUS - Japanese Architecture and Art Net Users System*. Internet. 2003. URL: https://www.aisf.or.jp/~jaanus/.

75 Maria Parisi and Maurizio Piazza. “Mechanics of Plain and Retrofitted Traditional Timber Connections”. In: *Journal of Structural Engineering-asce - J STRUCT ENG-ASCE* 126 (Dec. 2000).

76 *Pathway to critical and formidable goal of net-zero emissions by 2050 is narrow but brings huge benefits, according to IEA special report*. International Energy Agency. May 18, 2021. URL: https://www.iea.org/news/pathway-to-critical-and-formidable-goal-of-net-zero-emissions-by-2050-is-narrow-but-brings-huge-benefits.

77 Ercüment Erman Ph.D. “A Survey on Structural Timber Joint Classifications and a Proposal Taxonomy”. In: *Architectural Science Review* 42.3 (1999), pp. 169–180. eprint: https://doi.org/10.1080/00038628.1999.9696874. URL: https://doi.org/10.1080/00038628.1999.9696874.

78 Gilibert Pierre, Mesnil Romain, and Baverel Olivier. *Rule-based generative design of translational and rotational interlocking assemblies*. Version v0. Aug. 2021. URL: https://doi.org/10.5281/zenodo.5158465.

79 *Prospectives 2035 et 2050 de consommation de matériaux pour la construction neuve et la rénovation énergétique BBC*. Accessed: 2022-09-13. 2019. URL: https://librairie.ademe.fr/urbanisme-et-batiment/439-prospectives-2035-et-2050-de-consommation-de-materiaux-pour-la-construction-neuve-et-la-renovation-energetique-bbc.html.

80 Reuters. “Spain’s electric steel mills cut output amid soaring power prices”. In: *Reuters* (Mar. 10, 2022). URL: https://www.reuters.com/business/acerinox-arcelormittal-celsa-cut-steel-output-spain-due-high-electricity-prices-2022-03-10/.

81 Reuters/ABC. “India could run out of coal soon. So why is a country with such big reserves facing shortages?” In: *ABC* (Oct. 5, 2021). URL: https://www.abc.net.au/news/2021-10-05/india-facing-coal-shortage-could-run-out-of-power-explainer/100516332.

82 Reuters/ABC. “Solar Power May Be the Next Victim of China’s Coal Shortage”. In: *Bloomberg* (Sept. 29, 2021). URL: https://www.bloomberg.com/news/articles/2021-09-29/china-slashes-silicon-output-signaling-higher-solar-panel-costs.

83 Christopher Robeller and Yves Weinand. “Interlocking Folded Plate – Integral Mechanical Attachment for Structural Wood Panels”. In: *International Journal of Space Structures* 30 (2015), pp. 111–122.

84 Peter Roelants. *Gaussian process, Python notebooks*. Internet. 2019. URL: https://peterroelants.github.io/posts/gaussian-process-kernels/.

85 Bruce Romney et al. “An Efficient System for Geometric Assembly Sequence Generation and Evaluation”. In: Sept. 1995, pp. 699–712.

86 Alexander Schiftner et al. “Packing Circles and Spheres on Surfaces”. In: *ACM Trans. Graph.* 28.5 (Dec. 2009), pp. 1–8. ISSN: 0730-0301. URL: https://doi.org/10.1145/1618452.1618485.

87 Yuliy Schwartzburg and Mark Pauly. “Fabrication-aware Design with Intersecting Planar Pieces”. In: *Computer Graphics Forum* 32 (May 2013).

88 Ken Shoemake. “Animating rotation with quaternion curves”. In: *SIGGRAPH ’85*. 1985.

89 Peng Song, Chi-Wing Fu, and Daniel Cohen-Or. “Recursive Interlocking Puzzles”. In: *ACM Transactions on Graphics* 31.6 (Nov. 2012). ISSN: 0730-0301.

90 Peng Song et al. "Printing 3D objects with interlocking parts". In: *Computer Aided Geometric Design* 35-36 (Mar. 2015), pp. 137–148.

91 Paul E. Sprague. "The Origin of Balloon Framing". In: *Journal of the Society of Architectural Historians* 40.4 (Dec. 1981), pp. 311–319. ISSN: 0037-9808. eprint: https://online.ucpress.edu/jsah/article-pdf/40/4/311/172938/989648.pdf. URL: https://doi.org/10.2307/989648.

92 Anuj Srivastava and Eric Klassen. *Functional and Shape Data Analysis*. Jan. 2016. ISBN: 978-1-4939-4018-9.

93 Anuj Srivastava et al. "Shape Analysis of Elastic Curves in Euclidean Spaces". In: *IEEE transactions on pattern analysis and machine intelligence* (Sept. 2010).

94 Sai Ganesh Subramanian et al. "Delaunay Lofts: A biologically inspired approach for modeling space filling modular structures". In: *Computers & Graphics* 82 (2019), pp. 73–83. ISSN: 0097-8493. URL: https://www.sciencedirect.com/science/article/pii/S0097849319300822.

95 *Sustainable steel - Indicators 2020 and steel applications*. wordsteel. 2019. URL: https://worldsteel.org/media-centre/press-releases/2020/sustainable-steel-indicators-2020-and-steel-applications/.

96 Chengcheng Tang et al. "Form-Finding with Polyhedral Meshes Made Simple". In: *ACM Trans. Graph.* 33.4 (July 2014). ISSN: 0730-0301. URL: https://doi.org/10.1145/2601097.2601213.

97 Robert Tarjan. "Depth-First Search and Linear Graph Algorithms". In: *SIAM Journal on Computing* 1.2 (1972), pp. 146–160. eprint: https://doi.org/10.1137/0201010. URL: https://doi.org/10.1137/0201010.

98 Willy Tegel et al. "Early Neolithic Water Wells Reveal the World's Oldest Wood Architecture". In: *PLoS ONE* 7 (Dec. 2012), e51374.

99 Romain Testuz, Yuliy Schwartzburg, and Mark Pauly. "Automatic Generation of Constructable Brick Sculptures". In: *Eurographics 2013 - Short Papers*. Ed. by M.- A. Otaduy and O. Sorkine. The Eurographics Association, 2013.

100 *The Share of Wood-Framed Homes Increased in 2021*. National Association of Home Builders. URL: https://eyeonhousing.org/2022/07/the-share-of-wood-framed-homes-increased-in-2021/.

101 Gengo Matsui Torashichi Sumiyoshi. *Wood Joints in Classical Japanese Architecture*. Kajima Institute Publishing, 1991.

102 Alicia Valero, Antonio Valero, and Guiomar Calvo. "Material Limits of the Energy Transition". In: *The Material Limits of Energy Transition: Thanatia*. Cham: Springer International Publishing, 2021, pp. 147–187. ISBN: 978-3-030-78533-8. URL: https://doi.org/10.1007/978-3-030-78533-8_6.

103 Guan Wang et al. "Statistical analysis and modeling of the geometry and topology of plant roots". In: *Journal of Theoretical Biology* 486 (2020), p. 110108. ISSN: 0022-5193. URL: https://www.sciencedirect.com/science/article/pii/S0022519319304771.

104 Jianliang Wang et al. "A review of physical supply and EROI of fossil fuels in China". In: *Petroleum Science* 14 (2017), pp. 806–821.

105 Ziqi Wang, Peng Song, and Mark Pauly. "DESIA: A General Framework for Designing Interlocking Assemblies". In: *ACM Transactions on Graphics* 37.6 (Dec. 2018). ISSN: 0730-0301.

106 Ziqi Wang, Peng Song, and Mark Pauly. "MOCCA: Modeling and Optimizing Cone-Joints for Complex Assemblies". In: *ACM Trans. Graph.* 40.4 (July 2021). ISSN: 0730-0301. URL: https://doi.org/10.1145/3450626.3459680.

107 Ziqi Wang, Peng Song, and Mark Pauly. "State of the Art on Computational Design of Assemblies with Rigid Parts". In: *Computer Graphics Forum* 40 (May 2021), pp. 633–657.

108 Ziqi Wang et al. “Design and Structural Optimization of Topological Interlocking Assemblies”. In: *ACM Transactions on Graphics* 38.6 (2019), p. 13.

109 Michael Weizmann, Oded Amir, and Jacob Grobman. “Topological interlocking in buildings: A case for the design and construction of floors”. In: *Automation in Construction* 72 (June 2016), pp. 18–25.

110 Daniel Welsby et al. “Unextractable fossil fuels in a 1.5°C world.” In: *Nature* 597 7875 (2021), pp. 230–234.

111 Emily Whiting, John Ochsendorf, and Frédo Durand. “Procedural Modeling of Structurally-Sound Masonry Buildings”. In: *ACM Trans. Graph.* 28.5 (Dec. 2009), pp. 1–9. ISSN: 0730-0301. URL: https://doi.org/10.1145/1618452.1618458.

112 R.H. Wilson and T. Matsui. “Partitioning An Assembly For Infinitesimal Motions In Translation And Rotation”. In: 2 (1992), pp. 1311–1318. URL: http://ieeexplore.ieee.org/document/594555/ (visited on 11/08/2019).

113 Randall H. Wilson and Jean-Claude Latombe. “Geometric reasoning about mechanical assembly”. In: *Artificial Intelligence* 71.2 (Dec. 1994), pp. 371–396.

114 G.E. Woodward and F.W. Woodward. *Woodward’s Country Homes*. Woodward’s country homes vol. 1. Woodward, 1866. URL: https://books.google.fr/books?id=8FFDAQAAMAAJ.

115 Shiqing Xin et al. “Making Burr Puzzles from 3D Models”. In: *ACM Transactions on Graphics* 30.4 (July 2011). ISSN: 0730-0301.

116 Yong-Liang Yang, Jun Wang, and Niloy J. Mitra. “Reforming Shapes for Material-Aware Fabrication”. In: *Proceedings of the Eurographics Symposium on Geometry Processing*. SGP ’15. pp. 53–64. Graz, Austria: Eurographics Association, 2015.

117 Jiaxian Yao et al. “Interactive Design and Stability Analysis of Decorative Joinery for Furniture”. In: *ACM Transactions on Graphics* 36.4 (Mar. 2017). ISSN: 0730-0301.

118 Klaus Zwerger. *Wood and Wood Joints - Building Traditions of Europe, Japan and China*. Basel: Birkhäuser Verlag GmbH, 2011.