Parameterizations in the Configuration Space and Approximations of Related Surfaces

by

DROR ATARIAH

Dissertation
Submitted for fulfillment of the Degree of
Doktor der Naturwissenschaften

Supervisor: Professor Dr. Günter Rote

Department of Mathematics and Computer Science
Freie Universität Berlin

May 2014

First Reviewer: Prof. Dr. Günter Rote
Second Reviewer: Prof. Dr. Carsten Lange
Date of Defense: 09.05.2014

This version of the dissertation was created using the sources that constitute
commit: #`06c0fd2`, 2014-05-26

To my parents, brother and wife.

Zusammenfassung

Diese Arbeit betrachtet drei Themen, die alle dem berühmten *Bewegungsplanungsproblem für Roboter* in der Ebene zugeordnet werden können. Beim Arbeiten mit Robotern in der Ebene werden diese als konvexe Polygone dargestellt und es wird ein dazugehöriger Konfigurationsraum betrachtet, in welchem jeder Punkt einer eindeutigen Platzierung des Roboters in seiner physikalischen Umgebung entspricht. Hindernisse in der Welt des Roboters werden als dreidimensionale Festkörper im Konfigurationsraum betrachtet. Die Vereinigung aller dieser Körper bilden den sogenannten verbotenen Konfigurationsraum. Jeder Punkt des verbotenen Konfigurationsraum entspricht einer nicht realisierbaren Platzierung des Roboters zwischen den Hindernissen. Der Rest des Konfigurationsraumes ist der sogenannte freie Konfigurationsraum. Das Hauptinteresse liegt genau auf der Grenze zwischen dem freien und dem verbotenen Konfigurationsraum.

Der erste Teil dieser Dissertation beinhaltet eine einfache und geometrisch motivierte Parametrisierung von jenen Oberflächen, die genau die erwähnte Grenze darstellen. Durch Benutzung dieser Parametrisierung ist es sehr einfach, unterschiedliche Visualisierungen der Grenze zu erzeugen. Zudem können tiefe Einblicke in die Differentialgeometrie der Grenze gewonnen werden, wenn Standardberechnungen unter Benutzung dieser Parametrisierung durchgeführt werden. Diese beiden Ergebnisse tragen eindeutig zur allgemeinen Arbeit an Bewegungsplanungsproblemen bei. Insbesondere wird auch gezeigt, dass die Elemente der Grenze, die Kontaktpunkte zwischen einer Kante des Roboters und einer Ecke eines Hindernisses darstellen, nicht abwickelbare Regelflächen sind — und so eine negative Gaußsche Krümmung aufweisen.

Die negativen gekrümmten Oberflächen, die als Teile der Grenze auftreten können, regten die Suche nach einer *optimalen* Triangulierung für einfachere negativ gekrümmte Oberflächen an. Insbesondere werden hyperbolische Paraboloide (auch als Sattelfläche bekannt) betrachtet. Im zweiten Teil der Dissertation werden interpolierende und nicht interpolierende Triangulierungen von allgemeinen Satteloberflächen präsentiert. Die Optimalität der erzeugten Triangulierung, die für die Approximation benutzt wird, ist durch zwei Eigenschaften charakterisiert: (i) sie garantiert eine feste Fehlerschranke und (ii) sie minimiert die Anzahl der benötigten Dreiecke um einen gegebenen Teil der Sattelfläche zu überdecken.

Der dritte Teil der Dissertation liefert eine detaillierte Beschreibung eines Beitrages für das *2D Arrangements* Paket der COMPUTATIONAL GEOMETRY ALGORITHMS LIBRARY (**CGAL**). Hierbei wird die Berechnung von Arrangements von begrenzten teilweise linearen Kurven (polygonale Kurven) in der Ebene behandelt. Das ursprüngliche in der Version 4.3. enthaltene Codepaket wurde stark verbessert. Zum einen konnte die Berechnungszeit von Arrangements von polygonalen Kurven (im Durchschnitt) um 5 Prozent beschleunigt werden. Zusätzlich ist der geänderte Code generischer und daher besser geeignet für weitere Verallgemeinerungen. Daher wurde der verbesserte Code auch dazu ausgewählt in die Bibliothek (**CGAL**) integriert zu werden und es ist geplant, ihn in die nächste Version einzubinden. Alles in allem tragen die Ergebnisse dieser Arbeit sowohl zur Arbeit an Bewegungsproblemen bei, als auch zu zahlreichen allgemeineren Zwecken.

Abstract

This work covers three topics that can all be linked to the celebrated *motion planning problem* of planar robots. When considering a planar robot, which is represented by a convex polygon, it is natural to study an associated configuration space, where each point corresponds to a unique placement of the robot in its physical world. Obstacles in the world of the robot are represented as three-dimensional solids in the associated configuration space. The union of these solids is the so-called forbidden space. Each point of the forbidden space corresponds to a placement of the robot such that its interior intersects one or more obstacles. The remainder of the configuration space is the so-called free space. Of course, the boundary between the free and forbidden spaces is of tremendous interest.

The first part of the dissertation provides a simple and geometrically motivated parameterization of the surfaces that constitute the mentioned boundary. Using this parameterization, it is easy to produce various visualizations of the boundary. Furthermore, standard computations that utilize the parameterization yield deep understanding of the differential geometry of the boundary. Clearly, these are two achievements that contribute to the general study of the motion planning problem. In particular, it is also shown that the elements of the boundary corresponding to contacts between an edge of the robot and a vertex of an obstacle are non-developable ruled surfaces — thus having a negative Gaussian curvature.

The negatively curved surfaces that emerge as portions of the discussed boundary, motivated a search for an *optimal* triangulation of simpler negatively curved surfaces. Specifically, hyperbolic paraboloids (also know as saddles) are considered. The second part of the dissertation provides both interpolating and non-interpolating triangulations of general saddle surfaces. The optimality of the yielded triangulation used for the approximation is twofold: (i) it maintains a fixed error bound, and (ii) minimizes the number of triangles needed to cover a given portion of the saddle.

Finally, the third part of the dissertation provides a detailed account of a contribution to the *2D Arrangements* package of the COMPUTATIONAL GEOMETRY ALGORITHMS LIBRARY. This part of the work considers the computation of arrangements of bounded piecewise linear curves (polylines) in the plane. More precisely, the initial package code, as shipped with version 4.3, was considerably improved in two senses. First, the computations of arrangements of polylines using the modified code improves execution time by about 5% (on average). Secondly, the modified code is much more generic and suitable for further generalizations. As a result, the improved code was accepted for integration into the library and is scheduled to be shipped with its next release. Together, the achievements presented in the dissertation can contribute to the ongoing study of the motion planning problem, as well as to numerous more general purposes.

Acknowledgments

This dissertation could not have been realized without the guidance and support of my supervisor Günter Rote. Sometimes I felt that his intuition and deep understanding were several steps ahead of mine; luckily, Dr. Rote's patience came to my rescue. I am grateful for all his efforts and contributions. Accomplishing this project would be much harder without the endless support and mentoring of Günter M. Ziegler.

Fortunately, I have many colleagues whose help made this dissertation possible. I am particularly thankful to Efi Fogel for introducing me to the world of programming. My contribution to CGAL would have never been possible without his help. Michael Hemmer and Eric Berberich encouraged me and helped me produce a better code. Many thanks also to Dan Halperin and his group (in particular, to Oren Salzman) for hosting several visits of mine in Tel-Aviv university that were extremely fruitful. Discussions with Ramsay Dayer, Sunayana Ghosh and Mathijs Wintraecken were illuminating and helpful. I received generous support from the wonderful community of the StackExchange network in general, and the members of TeX.SE in particular. Finally, I would like to thank my fellow group members in the Free university of Berlin.

I owe my deepest gratitude to Michael Stark who shared with me his priceless experience and provided me with mentoring and spiritual support.

I would like to express my gratitude to the Berlin Mathematical School and especially to its managing director Nadja Wisniewski for the administrative and financial support. Furthermore, this work is part of the project *Computational Geometric Learning*. The project CG Learning acknowledges the financial support of the Future and Emerging Technologies (FET) program within the Seventh Framework Program for Research of the European Commission, under FET-Open grant number 255827. In addition, this research was supported by the DFG Collaborative Research Center TRR 109, *Discretization in Geometry and Dynamics*.

Last but definitely not least, my family deserves special thanks: My parents and the younger brother who were understanding and patient; my sons who suffered bravely. Finally, above all, I am grateful to my wife who, by all means, stood by my side and provided me all one can ask for when writing a dissertation.

Contents

Nomenclature

T_pS	The tangent to the surface S at p, page 29
$T_S(u,v)$	The tangent to the parameterized surface S at $S(u,v)$, page 29
φ	Golden Ratio, page 56
$\mathbf{q}$	A configuration point in $\mathscr{C}$, page 10
$\langle\cdot,\cdot\rangle$	Inner/dot product, page 26
$\mathscr{P}_\lambda$	Pseudo-Euclidean transformation in $\mathbb{R}^2$, page 78
int	The interior of a set, page 12
∂	The boundary of a set, page 12
M^τ	A rational rotation matrix , see equation (2.5), page 11
R^α	Standard (counterclockwise) rotation matrix, page 11
$\mathscr{N}_pS$	The normal to the surface S at the point $p \in S$, page 25
$\mathscr{N}_S(u,v)$	The normal to the (parameterized) surface S at the point $S(u,v)$, page 26
p^z	The z coordinate of $p \in \mathbb{R}^3$. Similarly p^x and p^y, page 49
$SE(n)$	Special Euclidean group, page 12
ℓ_{pq}	The line segment connecting p and q in space, page 50
$\vec{n}_j^{O_k}$	The inward normal to the edge $E_i^{O_k}$, page 10
O_k	The k-th obstacle in $\mathscr{W}$, page 10
$E_j^{O_k}$	The j-th edge of the k-th obstacle, page 10
A	Robot; i.e. a convex planar polygon, page 9
$\vec{n}_i^A$	The inward normal to the edge E_i^A of a robot A, page 10

E_i^A	The i-th edge of A, page 10
R_0	The reference point of the robot A, page 9
a_i	The i-th vertex of the robot A and we have $a_i = r_i(\cos\alpha_i, \sin\alpha_i)$, page 9
$\mathscr{C}_{\text{forb}}$	The forbidden space, page 11
$\mathscr{C}_{\text{free}}$	The free space, page 11
$\mathscr{C}$	The configuration space, page 10
$\mathscr{W}$	The workspace, page 9
$(T, \hat{T})$	$\hat{T}$ is the lifting of a planar triangle T to a saddle S, page 60

1 Introduction

If one would like to turn a robotic vacuum cleaner into a messenger then he or she would have to teach the robot three things. First, the robovac should *learn* the layout of its world (which we call workspace); that is, obtain a description of the locations of walls, passages, pieces of furniture etc. (which we call obstacles). Next, it should be able to *decide* whether it can get from its current position (source) to some given destination (target). Naturally, this question has a positive answer if there exists at least one free path in the workspace from the current source to the provided target. Here, free path means that along the whole path the robot should not collide with any obstacle that can be found in the workspace. Finally, if the answer to the posed decision problem is positive, the robot should also be able to *find* such a free path. Together, these three tasks are often called *motion planning problem*, or the *piano movers problem*. Naturally, the motion planning problem is well established and draws the attention of scholars for at least forty years.

A simple model of the problem can be formulated by assuming that the robot is a convex polygon in the plane, while certain regions of the plane are considered as obstacles. In Figure 1.1 the robot (in green) is shown in four different placements, and the room and obstacles shown in red. Furthermore, let us assume that the robot has three *degrees of freedom*, namely, it is free to translate (two degrees) and rotate (third degree) in the plane. In this model and the assumptions given above, the problem involves the motion of a polygon that clearly has non-zero area. It is very common to analyze this model in terms of the so-called *configuration space*. Each point of the configuration space corresponds to a unique placement of the robot inside the workspace. The planar obstacles are represented as three-dimensional solids — configuration points of these solids correspond to placements where the interiors of the robot and one or more obstacles intersect. The advantage of this model is that the initial problem reduces to the motion of a point in space. The motion planning problem has been addressed by many researchers; let us mention the classical works of Schwartz and Sharir [37] and the *probabilistic roadmap (PRM)* [21], as well as the recent *Motion planning via Manifold Samples* [36] approach.

In our discussion, we assume that the robot acquired a full description of its workspace, and concentrate on issues related to the stages of decision making and

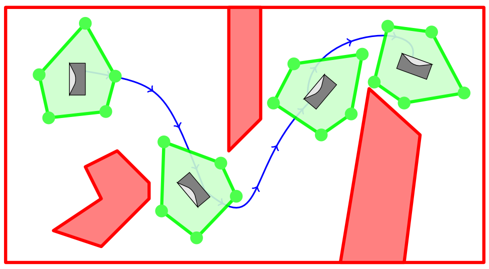


Figure 1.1: An example of a messenger robot in its workspace

free path finding. The configuration space is partitioned into two subsets. The *free* part contains configuration points that correspond to free placements of the robot such that its interior does not intersect any obstacles. The second part, so-called *forbidden* space, is the complement of the free space. It is clear that the robot cannot move from a free pose to a forbidden one without its representing point in the configuration space crossing the boundary between the free and forbidden spaces. This is why this boundary plays a crucial role. The first problem considered in this work is the study of the geometry of this boundary in the configuration space. To this end, we derive a parameterization of the various elements that constitute this boundary. Using this parameterization, we can achieve two objectives. First, we can easily visualize the boundary between the free and forbidden parts of the configuration space. This visualization provides a view inside the configuration space and serves as an important educational tool. Furthermore, once this parameterization is formalized, it is straightforward to describe the differential geometrical properties of the boundary. These two aspects taken together provide a deeper understanding of the geometrical structure of the configuration space.

When the robot moves in its workspace, it can touch obstacles. We distinguish between four types of touching (also called contacts): (i) a vertex of the robot slides on an edge of an obstacle, (ii) an edge of the robot slides on a vertex of an obstacle, (iii) a vertex of the robot touches a vertex of an obstacle and finally (iv) an edge of the robot overlaps an edge of an obstacle. Obviously, these types are not mutually exclusive, and the robot can maintain several contact types with one or more obstacles simultaneously. These contact types are strongly related to the properties of the boundary between the free and forbidden parts of the configuration space. In particular, the vertex-edge and edge-vertex types correspond to two-dimensional elements of the boundary, which we call *contact surfaces*, which are "glued" together by one-dimensional elements corresponding to the vertex-vertex and edge-edge contact types.

Once a piecewise smooth representation of the boundary between the free and forbidden spaces is obtained, using the parameterization that we discussed before, it is natural to consider its discretization. Here our interest focuses on the approximation of the boundary using a triangulated mesh. In particular, we consider the two-dimensional elements of the boundary, namely, the contact surfaces.

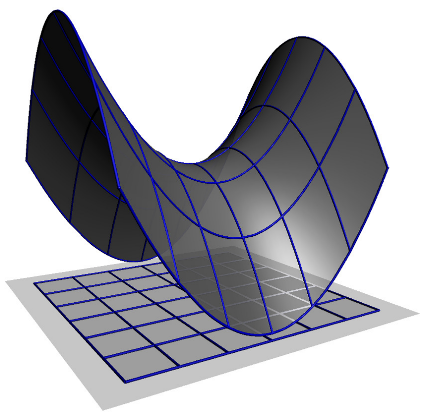


Figure 1.2: An example of a generic saddle surface.

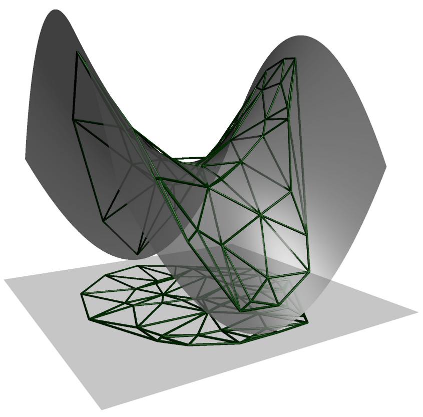


Figure 1.3: An interpolating approximation of a general saddle, which is a lifting of a planar Delaunay triangulation.

First, we observe that the Gaussian curvature of the contact surfaces of the first type, namely, those that correspond to vertex-edge contacts, vanishes. That is, surfaces of this type are *developable*. The surfaces of the second type, those that correspond to edge-vertex contacts, are more interesting from a geometrical point of view since their Gaussian curvature is negative everywhere. On the one hand, intuitively speaking, the first observation means that the contact surfaces of the first type are locally flat like the Euclidean plane. On the other hand, the surfaces of the second type, namely, those that correspond to edge-vertex contacts, are locally saddle-like. A bit more precisely, if $p \in S$ where S is a smooth and negatively curved surface, then the second order approximation of the *Monge patch* centered at p is a *hyperbolic paraboloid*, also known as a *saddle*. An example of such a saddle surface is depicted in Figure 1.2. A rigorous study of the Gaussian curvature and related topics can be found in any standard textbook on differential geometry like [12, 23].

This leads to the second part of this dissertation, namely, optimal approximation of saddle surfaces. On a broader level, optimal approximations of smooth surfaces in general and saddle surfaces in particular are of interest in their own right. In relation to the motion planning problem, the discretization of contact surfaces that correspond to edge-vertex contacts can benefit from a good approximation of saddle surfaces. As in the case of the first topic of the work, the approximation of saddle surfaces has also been intensively studied; for example, see Pottmann et al. [34]. The significant contribution presented here is an improvement of the approximation by considering *non-interpolating* triangulation.

Let us recall that a saddle surface S, as depicted in Figure 1.2, is given, up to rotations and translations, by

$$S = \left\{ (x, y, z) : z = \frac{x^2}{a^2} - \frac{y^2}{b^2} \right\}.$$

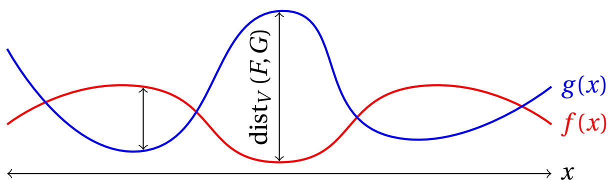


Figure 1.4: Illustration of the vertical distance in the plane.

A naive, interpolating, approximation of S using a triangulated mesh can be obtained as follows. First obtain a set $\mathscr{P}$ of points in the plane (for example randomly) and triangulate the point set (for example using the Delaunay triangulation). Next, lift the points in $\mathscr{P}$ to the saddle S. That is, given a point $p \in \mathscr{P}$ set the point $P = \left(p^x, p^y, F(p^x, p^y)\right)$ to be the lifting of p to S, where $F(x,y) = \frac{x^2}{a^2} - \frac{y^2}{b^2}$ and p^x, p^y are the x and y coordinates of p, respectively. Finally, connect two lifted points P and Q only if their preimages p and q are connected in the planar triangulation. An example of this discretization is illustrated in Figure 1.3. In this case, all the vertices of the yielded mesh are located on the surface. It is natural to expect that this approach can be improved.

Let us formulate some criteria, to make “improvement” more precise. First, let us formulate an error criterion, which is always necessary when considering approximations. To this end, we consider the *vertical distance* that we loosely define below. Intuitively, the vertical distance between two surfaces is the maximal length of a *vertical* line segment that connects point on the surfaces. Let us demonstrate this notion in the plane. Let $f, g\colon D \to \mathbb{R}$, where $D \subset \mathbb{R}^2$, be two smooth functions and let F and G be the corresponding graphs in the plane. Then $\mathrm{dist}_V(F,G) = \max_{x \in D} |f(x) - g(x)|$ as illustrated in Figure 1.4. The notion can be easily generalized for the case of surfaces, as we do in this work. We can now formulate the first criterion that we use when evaluating an approximation. In particular, given some $\varepsilon > 0$ we want all triangles $\hat{T}$ in the triangulation $\mathscr{T}$ to satisfy $\mathrm{dist}_V\left(S, \hat{T}\right) \le \varepsilon$, where S is the saddle surface. This criterion can be thought of as the *correctness* of the approximation.

Next, we want to improve the *efficiency* of the approximation. In particular, given some fixed $\varepsilon > 0$, we want to use as few triangles as possible while maintaining the correctness set by ε. For example, let $\mathscr{T}$ and $\mathscr{T}'$ be two triangulations that approximate a saddle S and assume that $\mathrm{dist}_V\left(S, \hat{T}\right) = \mathrm{dist}_V\left(S, \hat{T}'\right) \le \varepsilon$ for all $\hat{T} \in \mathscr{T}$ and $\hat{T}' \in \mathscr{T}'$. Then we say that $\mathscr{T}$ is more efficient than $\mathscr{T}'$ if it consists of *less* triangles. Obviously, the number of triangles in $\mathscr{T}$ equals the number of triangles in its vertical projection to the xy-plane. Finally, since the number of triangles is inversely proportional to their area, we can improve the efficiency of $\mathscr{T}$ by increasing the areas of the corresponding planar triangles.

These two criteria can be roughly formulated as the following problem. Let $D \subset \mathbb{R}^2$ be some domain in the plane. Find a triangulation $\mathscr{T}$ of the domain D such that for all $T \in \mathscr{T}$ its lifting to the saddle, denoted by $\hat{T}$, satisfies

$$\mathrm{dist}_V\left(S, \hat{T}\right) \leq \varepsilon.$$

Then, optimize the areas of the triangles that constitute $\mathscr{T}$ such that the vertical distance is maintained fixed. We can now consider a third criterion — the *quality* of the triangles that constitute the triangulation. More precisely, we want to improve the shape of the triangles that the triangulation comprises. The shape of a triangle can be improved in numerous ways; we choose to consider the maximization of the minimal angle.

We can now state the problem that is addressed in this work. Given a saddle surface S, defined over a finite domain, and an error bound $\varepsilon > 0$, we look for a planar triangulation $\mathscr{T}$ such that $\mathrm{dist}_V\left(S, \hat{T}\right) \leq \varepsilon$ where $\hat{T}$ is the lifting of $T \in \mathscr{T}$ to the saddle S. Furthermore, if $\mathscr{T}'$ is another planar triangulation such that the liftings of all its triangles have an ε vertical distance to S, then we show that (i) the number of triangles in $\mathscr{T}$ is smaller or equal than the number of triangles in $\mathscr{T}'$. In addition, (ii) If α and α' are the minimal angles over all the triangles in $\mathscr{T}$ and $\mathscr{T}'$, respectively, then $\alpha > \alpha'$. So far, we have considered an interpolating approximation of the saddle S. Next, we show that by allowing a *non-interpolating* triangulation of S, an improved approximation of saddles can be found. In particular, given a fixed error bound $\varepsilon > 0$, we find another planar triangulation $\mathscr{T}^\star$ such that for all $T \in \mathscr{T}^\star$ we have $\mathrm{dist}_V\left(S, \hat{T}\right) \leq \varepsilon$, where $\hat{T}$ is a special vertical lifting of T as introduced in the work. Furthermore, the areas of the triangles in the improved triangulation $\mathscr{T}^\star$ are larger than the areas of the triangles obtained in the interpolating case.

The last part of the dissertation describes the contribution to the *2D Arrangements* package of the COMPUTATIONAL GEOMETRY ALGORITHMS LIBRARY (**CGAL**).[1] Let us first briefly recall what an arrangement is. For the sake of simplicity let us consider the two-dimensional case. For a finite collection of planar geometric objects S, the arrangement $\mathscr{A}(S)$ is the subdivision of the plane into cells that are induced by the objects in S. Figure 1.5 depicts an example of an arrangement that is induced by three ellipses. In this example, the arrangement partitions the plane into thirteen bounded cells and one unbounded cell. Note that the notion of arrangement is well defined also in arbitrary dimensions. Over the past few decades, arrangements and the computation of arrangements were deeply studied. Fogel et al. [16], for example, provide further details and references.

The notion of arrangements is extremely useful in various fields. Of special interest, in our case, we note its role in the realm of robot motion. For example, as Fogel et al. [17] discuss, computations related to arrangement can solve the motion planning problem in the case of a translating robot. Thus, it comes as no surprise that while working on the other topics covered by the dissertation, the arrangement package drew some attention. In particular, the computation of arrangements of families of piecewise linear curves (which we call *polylines*) was considered. Version 4.3 of **CGAL** is shipped with an implementation that can

[1] `www.cgal.org`

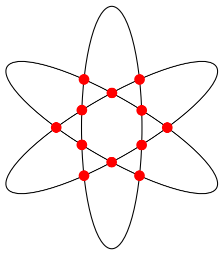


Figure 1.5: Simple example of an arrangement.

compute the arrangement of a family of *bounded* polylines in the plane. The *2D Arrangements* package supports the computation of arrangements of families of linear objects (line segments, rays and unbounded lines); however, it does not allow the computation of arrangements of families of *unbounded* polylines.

Generalizing the implementation of *2D Arrangements*, which would enable the computation of arrangements of unbounded polylines was set as the goal of this part of the work. Unfortunately, the code that is delivered with version 4.3 of **CGAL** introduced several hurdles that obstructed the fulfillment of this goal within the scope of this work. Nevertheless, a modest contribution to *2D Arrangements* was successful. The original code that handles bounded polylines was improved. First, the execution time of a computation of an arrangement of bounded polylines was reduced by 5% on average. Secondly, the updated code introduces better programming style on the one hand, and more generic implementation on the other hand. Hopefully, the second part of the contribution has paved the way towards the accomplishment of the ultimate goal.

Outline. Apart from this introduction, the dissertation consists of four chapters. The second chapter introduces in detail the motion planning problem and the related configuration space. Then, it derives the parameterization of the contact surfaces and studies their geometrical properties. In the third chapter we discuss the optimal approximation of the saddle surfaces. In the fourth chapter, we discuss the contribution made to the Arrangements on Surface package of **CGAL**. Finally, we conclude the dissertation in the fifth chapter.

2 Parameterization of Contact Surfaces

A summary of this chapter can be found in [2]. The accompanying technical report [1] gets into the details; this chapter is mostly based on these two publications.

D.A.

2.1 Introduction

The *piano movers problem* is about four decades old [37, 32] and studied intensively ever since. A fundamental part of this study is the configuration space, which is associated to the workspace at hand. A workspace that consists of a planar polygonal convex robot, which is free to rotate and translate, together with polygonal obstacles give rise to a *configuration space.* Each point in the configuration space corresponds to a unique placement or pose of the robot in its workspace, and vice versa, that is, every pose of the robot in the workspace corresponds to a unique configuration point. The presence of obstacles in the workspace translates to the partition of the configuration space into two parts, namely, the *free* and *forbidden* spaces. Configuration points in the forbidden part correspond to poses in which the interior of the robot intersects the interior of one or more obstacles.

Most studies set the solution of the *motion planning problem* as the primary goal and thus focus mainly on algorithmical aspects. Thus, the related configuration space was hardly studied from a *geometrical point of view.* In this work we focus on the geometrical properties of the configuration space, which is associated to the piano movers problem. To that end, we derive in Section 2.3 an explicit parameterization of the boundary of the forbidden space. Better understanding of this boundary can contribute, for example, to the general study of the motion planning problem. In turn, using this parameterization we study, in Section 2.5, the geometrical properties of this boundary.

In terms of visualization, most of the illustrations of the configuration space that can be found in the literature are rather simple. It is well known that for a robot that can *only* translate the boundary of the forbidden space is polygonal and can be computed using *Minkowski sums*. Thus, most of the visualizations slice the configuration space with horizontal planes. Each slice corresponds to a fixed rotation of the robot and the boundary can be computed using Minkowski sums. Finally, stacking these slices yields a discrete visualization of the obstacles as they appear in the configuration space [24]. Using the presented parameterization it is easy to visualize the boundary of the forbidden space, as can be seen in Figure 2.10 on page 23 and in [3].

Holonomic vs. non-holonomic. A robot for which *any* path in the configuration space is a valid motion in the workspace (in the absence of obstacles) is called *omnidirectional robot* or *holonomic*. On the other hand, if the robot is subject to velocity constraints, then both the constraints and the robot are called *non-holonomic*. In books like [25, 13] and [38, § 7.3.1] further details can be found. Xidias et al. [47] describes a method to tackle the motion planning problem, which is designed for both holonomic and non-holonomic robot. In this work we consider only the *holonomic* case, i.e. the controllable degrees of freedom equal the total degrees of freedom.

Previous work. The notion of configuration space was used for the first time in the seminal work of Lozano-Pérez and Wesley [27]. Surveys like [19, 46] can provide an overview at least on early related works. However, as we already pointed out, the literature aims mainly at the motion planning problem and hardly considers the boundary of the forbidden space per se, let alone parameterizing it or studying its geometrical properties.

Two interesting examples are [8, 30]. Both attempt to solve the motion planning problem itself, although they provide, as a byproduct, some idea on the geometrical nature of the boundary of the forbidden space. Yet, neither of them provide a simple, concise and geometrically driven representation of the various elements of the boundary. Another interesting example is the work presented by Varadhan et al. [44] where the boundary between the forbidden and the free spaces is approximated. The approximation obtained is guaranteed to be topologically correct and within some prescribed Hausdorff distance from the actual boundary's surface. Again, this work does not provide any geometrical insight on the nature of neither the contact surfaces nor their geometry. In his work, Brost [11] obtains both geometrical and topological information on the configuration space. Yet again, the obtained description is not straightforward.

The literature in terms of visualization of the configuration space is rather limited. Based on the parameterization presented in this paper Atariah and Rote [3] presented a scientific video that visualizes the configuration space of a planar polygonal robot. The visualization presented in [39] gives another glimpse into

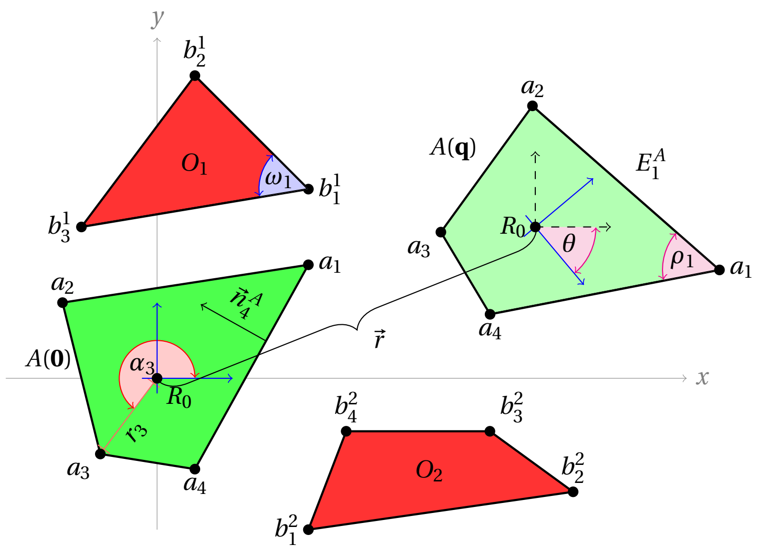


Figure 2.1: Workspace example with the robot $A(\mathbf{0})$ in its rest position (dark green) and in a configuration resulting of a translation by a vector $\vec{r}$ and rotation in angle θ (light green), that is $A(\mathbf{q})$ where $\mathbf{q} = (\vec{r}, \theta)$. Note that for $A(\mathbf{q})$ the local frame (in blue) does not align with the one of the workspace.

a configuration space. In this case the boundaries of the robot and the obstacles comprise line segments and circular arcs.

Configuration spaces were considered also in slightly different contexts. For example, Bajaj and Kim [4] consider the case of a robot with *curved* boundary that translates amid similar obstacles. The configuration space can naturally be considered as a group; see Remark 2.2 on page 12. Lysenko et al. [28] addressed the motion planning problem and other related problems as well using tools from *group morphology*. Contact surfaces are also discussed in [41].

Definitions and Notations. In this section we describe all the notations and notions that will be used throughout the chapter. Further details can be found in standard textbooks like [38, 26, 13, 25]. In addition refer to Figure 2.1 for illustrations of the various definitions.

The robot. A robot A is a convex planar polygon with n vertices denoted by $\{a_i\}_{i=1}^n$; we assume that they are given in counterclockwise order. Throughout the chapter, we follow the convention that $a_{n+1} = a_1$ and $a_0 = a_n$. The robot can translates and rotates in a *workspace* scattered with polygonal (convex) obstacles. The workspace is denoted by $\mathcal{W}$ and we take it to be $\mathcal{W} = \mathbb{R}^2$. The *reference point* of the robot is denoted by R_0, and we assume that in the *rest position* it is at the origin. Furthermore, we assume that the *local frame* of the robot aligns with the one of the workspace when in the rest position. The vertices of the robot are either

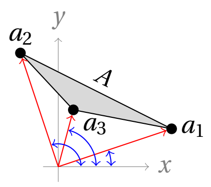


Figure 2.2: An example of a triangular robot whose vertices cannot be both counterclockwise ordered and in increasing angular order.

given with Cartesian coordinates, or with polar coordinates. For the vertex a_i we use the following notation $a_i = r_i\,(\cos\alpha_i, \sin\alpha_i)$ where $r_i := \|a_i\|$ and α_i denotes the angle with respect to the local frame of the robot. Finally, we assume that the vertices are in an increasing angular order, that is $0 \le \alpha_1 \le \ldots \le \alpha_n < 2\pi$. The two assumptions, on the order of the vertices of the robot and the monotonicity of their angles, can lead to inconsistencies, as depicted in Figure 2.2. To avoid this, we assume, in addition, that the reference point lies in the interior or on the boundary of A. Finally, we denote by ρ_i the internal angle corresponding to the i-th vertex.

Obstacle(s). Let $\{O_k\}_{k=1}^m$ be m obstacles in the workspace. The *vertices* of O_k for some k are given in counterclockwise order, and are denoted by $\{b_j^k\}$. We follow the same convention on the vertices of the obstacles, as we had for those of the robot. Finally, the interior angle at the j-th vertex will be denoted by ω_j^k. If the context introduces no confusion, then we omit the index k. We adopt the same convention regarding the cyclic ordering of the vertices as we formalized for the vertices of the robot.

Edges and their normals. An *edge* (vector) $\overline{a_i a_{i+1}}$ of the robot connecting a_i to a_{i+1} is denoted by E_i^A; that is $E_i^A = a_{i+1} - a_i$. Similarly, the j-th edge (vector) of the obstacle O_k connecting b_j^k to b_{j+1}^k is $E_j^{O_k}$. The *(inward) normals* of the edges E_i^A and $E_i^{O_k}$ are simply their 90° degrees counterclockwise rotation and denoted by $\vec{n}_i^A$ and $\vec{n}_j^{O_k}$, respectively. Note that according to this definition, the normals are not necessarily unit vectors.

Configuration space and poses. We let $\mathscr{C}$ denote the *configuration space* of the robot A in the workspace $\mathscr{W}$ that is scattered with the obstacles $\{O_k\}$. Each *configuration point*, or configuration for short, $\mathbf{q} \in \mathscr{C}$, corresponds to a unique *placement*, denoted by $A(\mathbf{q})$, of the robot in the workspace and vice versa. In other words, the placement $A(\mathbf{q})$ is the portion of the workspace that is covered by A when it assumes the configuration $\mathbf{q}$. In this work we may refer to a placement as either a *pose* or even a *configuration*, when there is no risk of confusion. We say that a configuration point $\mathbf{q}$, or the corresponding placement $A(\mathbf{q})$, is forbidden, if $A(\mathbf{q}) \cap O \neq \emptyset$ for some obstacle O in the set of obstacles $\{O_K\}$. The collection of all forbidden configuration points is called the *forbidden space* and it is denoted by $\mathscr{C}_{\text{forb}}$. Its complement is the *free space*, which is denoted by $\mathscr{C}_{\text{free}}$. Finally, $R_0(\mathbf{q}), a_i(\mathbf{q}), E_i^A(\mathbf{q})$ and $\vec{n}_i^A(\mathbf{q})$ denote respectively the position, in the workspace, of the reference point, i-th vertex, i-th edge or i-th normal of the robot.

For a configuration point $\mathbf{q} = (\vec{r}, \theta)$ we have

$$\begin{aligned} R_0(\mathbf{q}) &= \vec{r} + R_0 \\ E_i^A(\mathbf{q}) &= a_{i+1}(\mathbf{q}) - a_i(\mathbf{q}) \\ \vec{n}_i^A(\mathbf{q}) &= R^{\frac{\pi}{2}}\left(E_i^A(\mathbf{q})\right) \end{aligned}$$

where $\vec{r} = (x, y)$ is the translation component of the configuration, θ is the rotational component and R^α is the standard (counterclockwise) *rotation matrix* (cf. Figure 2.1). In order to express $a_i(\mathbf{q})$ for an arbitrary configuration $\mathbf{q}$ we have to choose a model of the configuration space. In this chapter we consider two possible models

$$\mathscr{C}^{\text{geom}} = \left\{(x, y, \theta) : (x, y) \in \mathbb{R}^2, \theta \in [0, 2\pi)\right\} \tag{2.1}$$

$$\mathscr{C}^{\text{rat}} = \left\{(x, y, \tau) : (x, y) \in \mathbb{R}^2, \tau \in \mathbb{R} \cup \{\infty\}\right\} \tag{2.2}$$

that we call the *geometrical* and *rational* models, respectively. When no confusion can arise, we simply write $\mathscr{C}$ to denote the configuration space. Note that $\mathscr{C}^{\text{geom}} = \mathbb{R}^2 \times S^1$ and $\mathscr{C}^{\text{rat}} = \mathbb{R}^2 \times \mathbb{RP}^1$. These models are related by $\tau = \tan\frac{\theta}{2}$.

For $\mathbf{q} = (\vec{r}, \theta) \in \mathscr{C}^{\text{geom}}$ we have

$$a_i(\mathbf{q}) = \vec{r} + R^\theta a_i(\mathbf{0}) = \vec{r} + R^\theta a_i. \tag{2.3}$$

On the other hand, given a configuration point $\mathbf{q}' = (\vec{r}, \tau) \in \mathscr{C}^{\text{rat}}$ with $\tau \in \mathbb{R} \cup \{\infty\}$ we have

$$a_i(\mathbf{q}') = \vec{r} + M^\tau a_i \tag{2.4}$$

where M^τ is the so-called *rational rotation matrix* given by

$$M^\tau = \frac{1}{1+\tau^2}\begin{pmatrix} 1-\tau^2 & -2\tau \\ 2\tau & 1-\tau^2 \end{pmatrix}. \tag{2.5}$$

Note that since $\lim_{\tau\to\pm\infty} M^\tau = R^\pi$ we can safely set $M^\infty = R^\pi$.

Before we continue, let us make two remarks.

Remark 2.1. When using the rational representation of the configuration space and taking rational coordinates for the translation vector $\vec{r}$ and letting $\tau \in \mathbb{Q}$, it is possible to establish *exact rational* computations. On the other hand, the geometrical representation is of more use when one is trying to visualize elements of the configuration space. This is due to the fact that the rotation component in $\mathscr{C}^{\text{geom}}$ is bounded.

Remark 2.2 (The Group Structure). The configuration space in our case, namely, the one that corresponds to a planar robot, which is free to rotate and translate, is homeomorphic to the *special Euclidean group* $SE(2)$. Indeed, the following homeomorphisms hold

$$SE(2) \cong \mathbb{R}^2 \times S^1 \cong \mathbb{R}^2 \times \mathbb{RP}^1.$$

See [26, §4.2] for further details.

Contacts and the Boundary of the Forbidden Space. For a set S we denote by ∂S and $\text{int}S$ the boundary and the interior of S, respectively. We say that $A(\mathbf{q})$ *touches* or is *in contact* with an obstacle O for a configuration $\mathbf{q}$ if

$$\partial A(\mathbf{q}) \cap \partial O \neq \emptyset \text{ and } \text{int}(A(\mathbf{q})) \cap \text{int}O = \emptyset.$$

Note that $A(\mathbf{q})$ touches O and its interior does not intersect the interior of any other obstacle in the workspace if and only if $\mathbf{q} \in \partial\mathscr{C}_{\text{forb}}$ [26, §4.3]. If only

$$\partial A(\mathbf{q}) \cap \partial O \neq \emptyset,$$

and the interiors of the robot and the obstacle either intersect or not, then we say that $A(\mathbf{q})$ *pseudo touches* or is *in pseudo contact* with the obstacle O. For illustration, see the contact between O_2 and $A(\mathbf{q})$ in Figure 2.3. If a configuration $\mathbf{q}$ corresponds to a pseudo contact but not to a contact, then we clearly have that $\mathbf{q} \in \mathscr{C}_{\text{forb}}$.

For a configuration $\mathbf{q}$, such that the A pseudo touches *or* just touches an obstacle O, one or more of the following *contact types* can hold:

Name	Notation	Definition
Vertex-Edge	(v_i-e_j)	$a_i(\mathbf{q}) \cap \text{int}E_j^O \neq \emptyset$
Edge-Vertex	(e_i-v_j)	$\text{int}E_i^A(\mathbf{q}) \cap b_j \neq \emptyset$
Vertex-Vertex	(v_i-v_j)	$a_i(\mathbf{q}) = b_j$
Edge-Edge	(e_i-e_j)	$\lvert\text{int}E_j^A(\mathbf{q}) \cap \text{int}E_j^O\rvert > 1$

Note that the contact type alone does not imply whether the interiors of the robot and the obstacle intersect or not. Note, in addition, that a robot can maintain various (pseudo) contacts with the same obstacle simultaneously. In the presence of more than one obstacle, then it can maintain multiple contacts as well. The following definitions refer to portions of $\mathscr{C}_{\text{forb}}$ that maintain a fixed (pseudo) contact with a given obstacle.

Definition 2.1 (Contact Surface)**.** The set of all configuration points that correspond to a *pseudo contact* between a vertex (or an edge) of the robot and a vertex (or an edge) of an obstacle is called a *contact surface.*

Note that a contact surface can be decomposed into two sub-surfaces; one that is contained in the interior of $\mathscr{C}_{\text{forb}}$ and the second one in $\partial\mathscr{C}_{\text{forb}}$. Indeed, this is true, because a contact surface comprises configuration points that correspond to either pseudo contacts or contacts. The following definition focuses on the configurations that realize *contacts.*

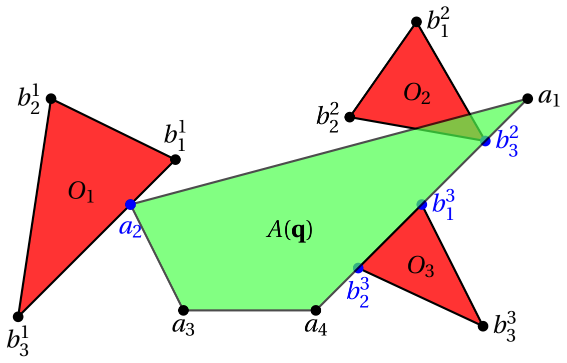


Figure 2.3: In this example, the configuration $\mathbf{q}$ corresponds to a (v_2-e_3) contact with O_1 and an (e_4-v_3) pseudo contact with O_2. In addition, with O_3 the robot maintains (e_4-v_2), (e_4-v_1) and (e_4-e_1) contacts simultaneously. This example demonstrates that $\mathbf{q}$ belongs to four contact patches and to a contact surface.

Definition 2.2 (Contact Patch)**.** The set of all configuration points that correspond to a *contact* between a vertex (or an edge) of the robot and a vertex (or an edge) of an obstacle is called a *contact patch*.

Every contact between a robot and an obstacle is also a pseudo contact, thus we have that each contact patch is a subset of a contact surface. Furthermore, it is a subset of $\partial\mathscr{C}_{\text{forb}}$. In addition, if the workspace contains a single convex obstacle, then the union of all contact patches is the boundary of the forbidden space. Finally, a contact surface that maintains either a (v-e) or an (e-v) contact is of dimension two, whereas a contact surface that maintains either a (v-v) contact or an (e-e) contact is of dimension one. This means that the boundary $\partial\mathscr{C}_{\text{forb}}$ is a union of contact patches of dimension two that are "glued" together with contact patches of dimension one. In Figure 2.3 an example of a pose that maintains various contact types is illustrated. Finally, a single obstacle is represented as a pillar-like object; an example is depicted in Figure 2.10 on page 23. The portion of $\mathscr{C}$ that is bounded inside this pillar-like object is the forbidden space, which corresponds to the considered obstacle. If the workspace contains more then one obstacle, then each one of them contribute another pillar-like object in the configuration space. The union of the interiors of the pillars is the forbidden space.

In this chapter we will formulate an explicit parameterization of the contact surfaces depending on the properties of the robot and the obstacles. Furthermore, we will find a subset of the parameter domain, of each contact surface that corresponds to the respective contact patch. Thus, we will be able to parameterize the whole boundary of the forbidden space.

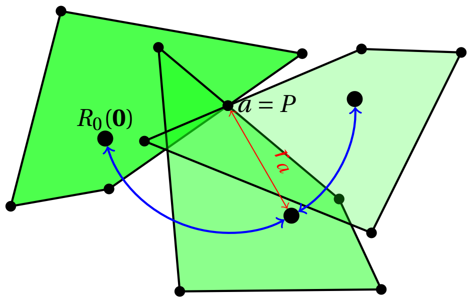


Figure 2.4: Rotating the robot around a fixed point on its boundary

2.2 Rotating the Robot

Since the robot A that we consider is holonomic, every point $P \in \mathcal{W}$ can be a center of rotation of the robot. In particular, given a configuration point $\mathbf{q} \in \mathscr{C}$, the robot can rotate about every point in the boundary $\partial A(\mathbf{q})$ of the robot. This kind of motion, which is illustrated in Figure 2.4, is the corner stone of the parameterization that we formalize in this work.

Let us, for the time being, ignore all obstacles in the workspace and assume that A can freely move in it. We set a point $P \in \mathcal{W}$ and consider a point a, which is given with respect to the local frame of the robot when it is in its rest position. Obviously, the point a can well be in either the interior, or on the boundary of the robot. For reference, see the example in Figure 2.4. According to the notations standards that we use, $a = a(\mathbf{0}), a(\mathbf{q})$ denote the position of the marked point when the robot is in either the rest position or in some pose corresponding to a configuration $\mathbf{q}$. We parameterize the set of configuration points in which the marked point a is fixed to the point P. More precisely, we want to parameterize the following set

$$P_a = \left\{\mathbf{q} \in \mathscr{C}^{\text{geom}} : a(\mathbf{q}) = P\right\}. \tag{2.6}$$

Let us stress that the model of the configuration space that we consider here is the geometric one. We will see in details, in Section 2.4, a similar parameterization that has to be used when $\mathscr{C}^{\text{rat}}$ is the model of the configuration space. Since the robot is rigid the locus $\{R_0(\mathbf{q})\}_{\mathbf{q} \in P_a}$ is a circle in the workspace with center P and radius $r_a = \|R_0 - a\|$ (cf. Figure 2.4). For $\chi \in [0, 2\pi)$, this circle is parameterized by

$$P + r_a \begin{pmatrix} \cos\chi \\ \sin\chi \end{pmatrix} \tag{2.7}$$

Based on this observation, we can prove the following lemma.

Lemma 2.3. *Let $P \in \mathcal{W}$ be a point in the workspace. Furthermore, let*

$$a = r_a \begin{pmatrix} \cos\alpha_a \\ \sin\alpha_a \end{pmatrix}$$

be a point that is given with respect to the local frame of the robot when it is in its rest position. Then, the set P_a, as defined in Equation (2.6)*, is parameterized by*

$$\mathbf{q}_a(\phi) = \begin{pmatrix} \vec{r}_a(\phi) \\ \theta_a(\phi) \end{pmatrix} = \begin{pmatrix} P - R^{\phi} a \\ \phi \end{pmatrix} \tag{2.8}$$

for $\phi \in [0, 2\pi)$. That is $\mathbf{q} \in P_a$ if and only if $\mathbf{q} = \mathbf{q}_a(\phi)$ for some $\phi \in [0, 2\pi)$.

Proof. On the one hand, we have

$$\begin{aligned} a(\mathbf{q}_a(\phi)) &= \vec{r}_a(\phi) + R^{\theta_a(\phi)} a \\ &= P - R^{\phi} a + R^{\phi} a = P \end{aligned}$$

This shows, that for every ϕ, the point $a(\mathbf{q}_a(\phi))$ is fixed to P.

Conversely, given $\mathbf{q} = (\vec{r}, \theta) \in P_a$ we have

$$P = a(\mathbf{q}) = \vec{r} + R^{\theta} a$$

Thus, $\vec{r} = P - R^{\theta} a$. Finally, for $\phi = \theta$ we have that $\mathbf{q} = \mathbf{q}_a(\phi)$. □

Note that for $\phi = 0$ the parameterization given in Equation (2.8) is merely a translation. In other words, the local frame that is assigned to the robot for $\mathbf{q}_a(0)$ is aligned with the global frame of the workspace. Finally, note that Equation (2.8) is a parameterization of a *helix* in the configuration space. We conclude this section with the following remark.

Remark 2.3 (Covering $\mathscr{C}$-Space with Helices). Instead of taking $a \in \partial A$, we can generalize the idea and consider an arbitrary linear combination of the vertices of the robot, $a = \sum_{i=1}^{n} \lambda_i a_i$, and some point $P \in \mathscr{W}$. The set of configurations that correspond to a rotation of the robot such that a is fixed to P is again a helix. As a matter of fact, every configuration point $\mathbf{q} \in \mathscr{C}$ is contained in infinitely many helices of this form.[1]

2.3 Parameterizing Contact Surfaces

In this section we consider the robot A and *one convex* obstacle O. Later, an arbitrary obstacle can be decomposed into convex subsets and each sub-obstacle can be treated in a similar way. Given a (pseudo) contact type of A and O we will derive an explicit parameterization of the corresponding contact surfaces and patches.

[1]This remark has been contributed by S. Ghosh.

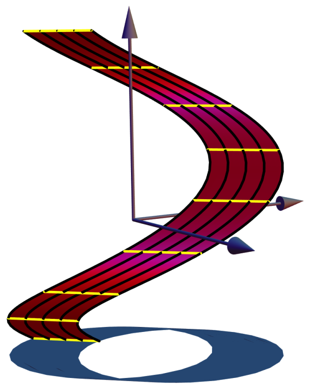


Figure 2.5: Example of a generic (v-e) contact *surface* in $\mathscr{C}$-space. The black curves are helices that correspond to $S(t_0,\phi)$ for $t_0 \in \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$. The yellow lines are the rulings of the surface; corresponding to fixed ϕ and varying $t \in [0,1]$.

2.3.1 Vertex-Edge Contact

A vertex-edge contact occurs when a vertex a_i of the robot lies in the interior of an edge E_j^O of the obstacle in a free manner, i.e. such that the interiors do not intersect (see O_1 and $A(\mathbf{q})$ in Figure 2.3 for an example). In this section, based on the parameterization obtained in Section 2.2, we will provide an explicit parameterization of the contact surface and the contact patch in the configuration space that correspond to the prescribed (v$_i$-e$_j$) pseudo contact and contact.

Let $P(t) = (1-t)b_j + tb_{j+1}$ be an arbitrary point in the interior of E_j^O. For $t \in (0,1)$ and $\phi \in [0,2\pi)$ the configurations yielding the poses where a_i is fixed to $P(t)$ and rotating about it, are derived from Equation (2.8) by replacing P with $P(t)$ and setting $a = a_i$

$$\begin{aligned} S(t,\phi) &= \begin{pmatrix} P(t) - R^{\phi}a_i \\ \phi \end{pmatrix} \\ &= \begin{pmatrix} b_j - R^{\phi}a_i \\ \phi \end{pmatrix} + t\begin{pmatrix} b_{j+1} - b_j \\ 0 \end{pmatrix} \qquad (2.9) \\ &= c(\phi) + t\vec{r}(\phi) \end{aligned}$$

Clearly, this surface is a *ruled surface* with directrix $c(\phi)$ and $\vec{r}(\phi) \neq 0$ as the vector field. Note that since $\frac{d}{d\phi}\vec{r}(\phi) = 0$ we have that S is a *cylindrical* ruled surface and thus *developable*. Note that for every $t' \in (0,1)$ we have that $S(t',\phi)$ is a helix. Furthermore, for $(0,1) \ni t'' \neq t'$ the helix $S(t'',\phi)$ is congruent to $S(t',\phi)$. Finally, for $t \in \{0,1\}$ the parameterization reduces to two helices, which correspond to the two pseudo contacts (v$_i$-v$_j$) and (v$_i$-v$_{j+1}$), respectively (cf. Section 2.3.3). Clearly, as ϕ varies in the interval $[0,2\pi)$, the configuration points on S represent both free *and* forbidden contact. In Figure 2.5, such a surface is illustrated.

Remark 2.4. If we fix a vertex a_i of the robot and generate all possible vertex-edge contact surfaces with all edges of the obstacle, then *helices* contained in each of

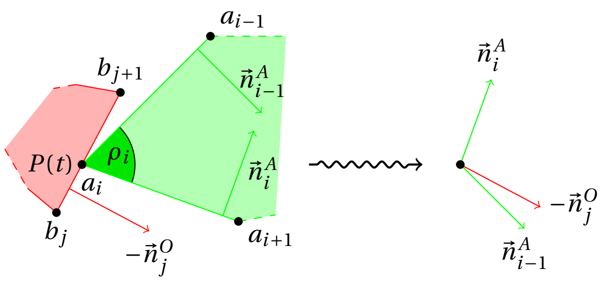


Figure 2.6: Generic setting of a (v-e) contact and the corresponding arrangement of the edge normals. Robot in green, and obstacle in red.

these contact surfaces are congruent copies of each other. Note that if the obstacles are regular polygons then for a fixed vertex of the robot the contact surfaces are just congruent copies of each other. If, in addition, the robot is a regular polygon as well, then *all the vertex-edge contact surfaces* are congruent copies of each other.[2]

Our next goal is to find a sub-domain $\Phi \subset [0, 2\pi)$ such that $S(t, \phi)_{|\phi\in\Phi}$ will be the *contact patch* that is contained in S. In Section 2.3.1.1 we analyze the domain $[0, 2\pi)$ of ϕ and find the sub-domain Φ. Section 2.A.1 provide a similar analysis, using a different technique; this alternative approach has some limitations and advantages, which are discussed.

2.3.1.1 Vertex Edge Angle Range Analysis

By now, we are given two indices i, j we derived the parameterization of the *contact surface* that corresponds to the (v_i-e_j) contact type. In particular, the configuration points on S correspond to all pseudo contacts between a_i and E^O_j. Let S be this surface, as given in Equation (2.9). Our next goal is to find a *contact patch* $S' \subset S \in \mathscr{C}$, such that for all $\mathbf{q} \in S'$ we will have that $a_i(\mathbf{q})$ *touches* the edge of the obstacle.

In particular, for each $t_0 \in (0, 1)$ we have to find a sub-domain $\Phi \subset [0, 2\pi)$ such that every point $\mathbf{q} \in S(t_0, \phi)\ |_{\phi\in\Phi}$ will yield a contact. Since we assume that both the robot and the obstacle are convex, the sub-domain Φ is independent of t_0. This can be seen in Figures 2.3 and 2.6. For some fixed $t_0 \in (0, 1)$ let $\mathbf{q}_i(\phi) = S(t_0, \phi)$ be a helix in S, which correspond to the pseudo contact between the vertex a_i and the point $P = P(t_0)$ in the interior of the edge E^O_j. Note that $a_i(\mathbf{q}_i(\phi)) = P$ for all ϕ. Finally, the sub-domain Φ can be determined by finding two values:

- $\phi_{\min}$: The angle for which $a_{i+1}(\mathbf{q}_i(\phi_{\min}))$ lies on the line containing the edge E^O_j and $a_{i-1}(\mathbf{q}_i(\phi_{\min}))$ lies to the right of this edge.
- ϕ_{range}: The range of rotation that maintains the contact of a_i with the point P. In practice this means that we want to have that $a_{i-1}((\mathbf{q}_i(\phi_{\min} + \phi_{\text{range}}))$

[2]This observation has been contributed by S. Ghosh.

will lie on the line containing E_j^O such that $a_{i+1}(\mathbf{q}_i(\phi_{\min}+\phi_{\text{range}}))$ will lie to its right. See Figure 2.6 for an illustration of the setting we consider.

Let $\phi_{\max} = \phi_{\min} + \phi_{\text{range}}$ and define

$$\Phi = \begin{cases} [\phi_{\min}, \phi_{\max}] & \text{if } \phi_{\max} < 2\pi \\ [\phi_{\min}, 2\pi) \cup [0, \phi_{\max} - 2\pi] & \text{if } \phi_{\max} \geq 2\pi \end{cases} \tag{2.10}$$

In turn, for all $\phi \in \Phi$ we have $a_i(\mathbf{q}_i(\phi)) = P$. Conversely, if for some configuration $\mathbf{q}$ there is a *contact* between a_i and the point P on the edge E_j^O, then there exists some $\phi_0 \in \Phi$ such that $\mathbf{q} = \mathbf{q}_i(\phi_0)$. Note that Φ as defined above is maximal only if P is an interior point of E_j^O. Otherwise, the sub-domain Φ is larger. This, however, is the case of vertex-vertex contact that we consider in Section 2.3.3.

We now compute the values of $\phi_{\min}$ and ϕ_{range}. The latter is straightforward to find, and depends on the interior angle at the vertex a_i of A, namely,

$$\phi_{\text{range}} = \pi - \rho_i.$$

Computing $\phi_{\min}$. We want to find ϕ such that for $\mathbf{q}_i(\phi) \in \mathscr{C}$ the following will hold:

$$P - \|E_i^A\| \frac{E_j^O}{\|E_j^O\|} = a_{i+1}(\mathbf{q}_i(\phi)),$$

where E_j^O is consider as the vector from b_j to b_{j+1} and $\|\cdot\|$ denotes the length of an edge. Solving this equation for ϕ is equivalent to solving

$$-\|E_i^A\| \frac{E_j^O}{\|E_j^O\|} = M \cdot \left(x, y\right)^T,$$

where $x = \cos\phi$, $y = \sin\phi$ and

$$M = \left(E_i^A, R^{\frac{\pi}{2}} \cdot E_i^A\right)^T. \tag{2.11}$$

Since $\det M = \|E_i^A\|^2 \neq 0$, this system has a unique solution, denoted by $\left(x_0, y_0\right)^T$. We define $\{\phi_i\}_{i=1}^4$ as follows

$$\begin{aligned} \{\phi_1, \phi_2\} &= \arccos(x_0) \cap [0, 2\pi) \\ \{\phi_3, \phi_4\} &= \arcsin(y_0) \cap [0, 2\pi) \end{aligned}$$

Note that since $\left(x_0, y_0\right)$ is a unit vector, we have that $\{\phi_1, \phi_2\} \cap \{\phi_3, \phi_4\}$ contains exactly one element. As $\phi_{\min}$ should lie in $[0, 2\pi)$, it satisfies

$$\phi_{\min} = \{\phi_1, \phi_2\} \cap \{\phi_3, \phi_4\}.$$

For any combination of signs of x_0 and y_0 Table 2.1 suggests in which interval $\phi_{\min}$ is, and using the definition of the ϕ_i's it can be easily found. Finally, the red patches in Figure 2.10 on page 23 correspond to all possible vertex-edge contacts between triangular robot and obstacle.

x_0	y_0	$\phi_{\min} \in$
≥ 0	≥ 0	$[0, \frac{\pi}{2}]$
< 0	≥ 0	$[\frac{\pi}{2}, \pi]$
< 0	< 0	$[\pi, \frac{3\pi}{2}]$
≥ 0	< 0	$[\frac{3\pi}{2}, 2\pi]$

Table 2.1: The interval containing $\phi_{\min}$, depending on signs of x_0 and y_0 for the vertex-edge and edge-vertex contact types.

2.3.1.2 On the Exactness

From the practical point of view, it is important to point that the evaluation of $\phi_{\min}$ involves trigonometric functions, thus it cannot be expressed in an *exact* manner. In particular, the matrix M in Equation (2.11) involves the trigonometric functions as well, and thus cannot be represented in an exact manner. In turn, this means that x_0 and y_0 above cannot be computed exactly in the first place. In Section 2.A.1 we find the sub-domain Φ without using trigonometrical functions, and thus it is represented exactly. However, this comes with a price; in order to exploit this advantage, one has to impose the restriction that all the vertices of the robot should lie on a circle of some fixed radius. This, obviously, is a very restrictive assumption. Finally, since we consider $\mathscr{C}^{\text{geom}}$ as the model of the configuration space, the rotation of the robot as discussed in Section 2.2 cannot yield exact representations. In turn, indeed, the parameterization of the contact surface and patch itself, given in Equation (2.9), involves the trigonometric functions and thus cannot be computed in an exact manner.

2.3.2 Edge-Vertex Contact

Recall that Equation (2.8) parameterizes a rotation of the robot around a point P such that a boundary point $a \in \partial A$ is fixed to P. For any $t \in (0,1)$ we let

$$a_{i,t} = (1-t)a_i + ta_{i+1} \tag{2.12}$$

denote the point on the edge E_i^A of the robot which is to be in pseudo contact with b_j. Next, in Equation (2.8), we replace P with b_j and a with $a_{i,t}$ and obtain

$$\begin{aligned} S(t,\phi) &= \begin{pmatrix} b_j - R^{\phi} a_{i,t} \\ \phi \end{pmatrix} \\ &= \begin{pmatrix} b_j - R^{\phi} a_i \\ \phi \end{pmatrix} + t \begin{pmatrix} -R^{\phi} \cdot E_i^A \\ 0 \end{pmatrix} \\ &= c(\phi) + t\vec{r}(\phi). \end{aligned} \tag{2.13}$$

for $t \in (0,1)$ and $\phi \in [0, 2\pi)$. $S(t,\phi)$ is the contact surface corresponding to the (e_i-v_j) pseudo contact. Again, like the parameterization in Equation (2.9), we

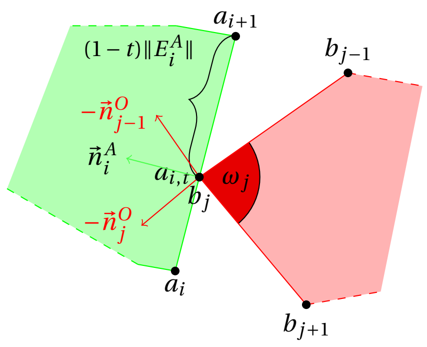


Figure 2.7: Generic setting of an (e-v) contact and the corresponding arrangement of the edge normals.

obtain a ruled surface. We study its geometry further in Section 2.5.1. We stress that configuration points on this surface lie both in $\partial\mathscr{C}_{\text{forb}}$ and in the interior of $\mathscr{C}_{\text{forb}}$, since this is a contact *surface*.

In order to find the contact *patch*, which is contained in S, as before, we have to find a sub-domain $\Phi \subset [0, 2\pi)$ for which $S(t,\phi)_{|\phi\in\Phi}$ is a collection of configuration points that corresponds to contacts and *not* to pseudo contacts. Again, as can be seen in Figure 2.7, the sub-domain Φ does not depend on t.

Remark 2.5. In contrast to the case of (v-e) (pseudo) contacts, here in the edge-vertex case we have that each contact surfaces is a collection of helices that are *not* congruent since their radii depend on $a_{i,t}$. This suggests, as we establish in Section 2.5.1, that the (e-v) contact surfaces are *not developable*.

Edge-Vertex Angle Range Analysis. Next, we find a sub-domain $\Phi \subset [0, 2\pi)$ such that $S(t,\phi)_{|\phi\in\Phi}$ is the *contact patch*, which is contained in the contact surface S. Like before, we provide two methods; one general that involves computations of trigonometric functions and thus inexact. The other, discussed in Section 2.A.2, is more accurate but at the same time more restrictive. Let us start with the general approach. To that end, we compute $\phi_{\min}, \phi_{\text{range}}$ and $\phi_{\max} = \phi_{\min} + \phi_{\text{range}}$ and we set Φ as defined in Equation (2.10). As $\phi_{\min}$ and ϕ_{range} depend only on the indices i and j but not on t, we fix some $t_0 \in (0,1)$ and let $\mathbf{q}_i(\phi) = S(t_0, \phi)$. In this case, we have that $a_{i+1}(\mathbf{q}_i(\phi_{\min}))$ has to lie on the line containing E_{j-1}^O, such that the interiors of the robot and the obstacle do not intersect. Similarly, $a_i(\mathbf{q}_i(\phi_{\max}))$ has to lie on the line segment containing E_j^O (cf. Figure 2.7). Clearly, we have that

$$\phi_{\text{range}} = \pi - \omega_j.$$

It is left to find the value of $\phi_{\min}$.

Computing $\phi_{\min}$. In the case of (e-v) contact, the minimal angle of rotation $\phi_{\min}$ is the one for which the following will hold

$$\left.\begin{aligned} a_{i+1}(\mathbf{q}_i(\phi)) - b_j \parallel b_j - b_{j-1} \\ a_{i,t_0}(\mathbf{q}_i(\phi)) = b_j \end{aligned}\right\} \tag{2.14}$$

See Figure 2.7 for reference. First note that

$$\forall \mathbf{q} \in \mathscr{C} \quad \| a_{i+1}(\mathbf{q}) - a_{i,t}(\mathbf{q}) \| = (1-t)\, \| E_i^A \|.$$

Thus, in order to find ϕ such that the conditions in (2.14) will hold we have to solve the following equation

$$b_j - (1-t_0)\, \| E_i^A \| \, \frac{E_{j-1}^O}{\| E_{j-1}^O \|} = a_{i+1}(\mathbf{q}_i(\phi)),$$

with ϕ as the unknown. Like in the previous computation of $\phi_{\min}$, the last equation can be rewritten as

$$(t_0 - 1) \| E_i^A \| \frac{E_{j-1}^O}{\| E_{j-1}^O \|} = M \cdot \left(x, y\right)^T$$

with $x = \cos\phi$, $y = \sin\phi$ and

$$M = \left(a_{i+1} - a_{i,t_0}, R^{\frac{\pi}{2}} \cdot (a_{i+1} - a_{i,t_0})\right)^T, \tag{2.15}$$

similarly to the matrix in Equation (2.11). Since $\det M = (1-t_0)^2 \, \| E_i^A \|^2 \neq 0$, the system above has a unique solution, namely,

$$(x_0, y_0)^T = (t_0 - 1) \frac{\| E_i^A \|}{\| E_{j-1}^O \|} M^{-1} \cdot E_{j-1}^O.$$

Note that since the matrix in Equation (2.15) contains trigonometric function and the solutions x_0, y_0 are obtained using inverse trigonometric functions, the values that we compute for $\phi_{\min}$ using this method cannot be represented exactly. In Section 2.A.2 we present a method, analogous to the one shown in Section 2.A.1, that can yield exact representation of Φ. Similarly to the case of the vertex-edge contact, due to the model of the configuration space in use and to the trigonometric nature of the computations, the contact surfaces (or patches) also in the case of edge-vertex contacts cannot be represented exactly. Finally, we refer to Figure 2.10 on page 23, where the green patches correspond to the edge-vertex contacts.

2.3.3 Vertex-Vertex and Edge-Edge Contacts

Motions of the robot that maintain either a vertex-edge or an edge-vertex contact have two degrees of freedom. In the previous sections we parameterized these motions, and, indeed, the developed parameterization yielded contact surfaces (and

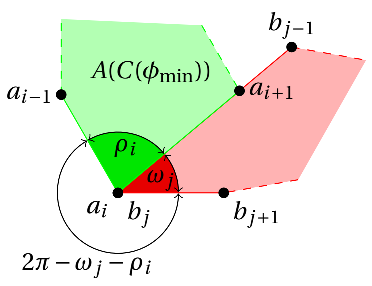


Figure 2.8: Illustration of a vertex-vertex contact with $C(\cdot)$ and $\phi_{\min}$ as defined in Section 2.3.3.1.

patches) of dimension two. In order to complete the picture we have to consider the configurations that correspond to vertex-vertex and edge-edge (pseudo) contacts. Motions maintaining these contacts have only *one* degree of freedom. We recall that the boundaries of the two-dimensional contact patches that we derived are exactly the one-dimensional contact patches. Indeed, we use the parameterization that we obtained and express explicitly the parameterizations of the vertex-vertex and edge-edge contacts.

2.3.3.1 Vertex-Vertex Contact

For two indices i and j, the corresponding (v$_i$-v$_j$) pseudo contact is parameterized by

$$C(\phi) = S(0,\phi)$$

for $\phi \in [0, 2\pi)$ and $S(\cdot,\cdot)$ as given in Equation (2.9). Note that for $\phi \in [0, 2\pi)$ we obtain a contact *surface* (actually a single helix), since it corresponds to (v$_i$-v$_j$) pseudo contacts that are not contacts. In order to find the interval Φ, which correspond to the (v-v) *contacts* alone, we compute $\phi_{\min}$ that corresponds to the (v$_i$-e$_{j-1}$) contact. For the pose that corresponds to $C(\phi_{\min})$ we have that a_i coincides with b_j, a_{i+1} lies on the line containing E^O_{j-1} and a_{i-1} lies to the right of this edge. See Figure 2.8 for an illustration. For Φ as defined in Equation (2.10) on page 18 with $\phi_{\max} = \phi_{\min} + 2\pi - \omega_j - \rho_i$ we get the sub-helical-arc $C(\phi)_{|\phi\in\Phi}$ that corresponds to the (v-v) contacts alone.

2.3.3.2 Edge-Edge Contact

Parameterizing the configurations that correspond to an (e$_i$-e$_j$) contact can be again obtained using the parameterization of the corresponding (v$_i$-e$_j$) contact. Let us set

$$C(t) = S(t, \phi_{\min})$$

where $S(\cdot,\cdot)$ is given in Equation (2.9) on page 16 and $\phi_{\min}$ is the one defined in Section 2.3.1.1. In this case we have that $C(0)$ corresponds to a (v$_i$-v$_j$) contact

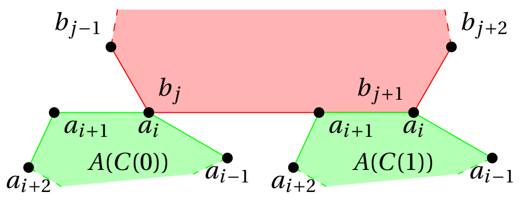


Figure 2.9: Setting of an edge-edge contact where $C(\cdot)$ is given in Section 2.3.3.2.

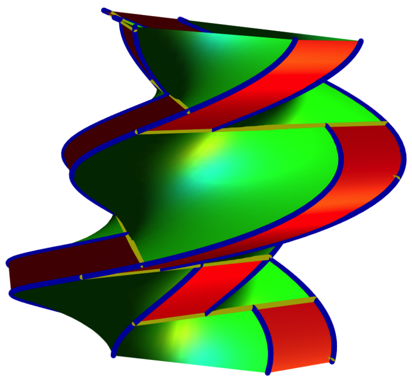


Figure 2.10: An example of all possible contact patches for a given robot and one obstacle in $\mathscr{C}$. In red and green are the (v-e) and (e-v) contact patches, respectively. The blue helical arcs correspond to (v-v) contacts and the yellow (straight) line segments correspond to (e-e) contacts.

and $C(1)$ corresponds to a (v_i-v_{j+1}) contact. In both cases a_{i+1} lies on the line containing E^O_j. In Figure 2.9 note that for $t \in [0,1)$ we do not obtain the whole (e_i-e_j) contact. If we take $t \in \left(0, 1+\frac{\|E^A_i\|}{\|E^O_j\|}\right)$, then the whole ($\mathrm{e}_i$-$\mathrm{e}_j$) contact is obtained; in particular for $t=0$ the vertex a_i coincides with b_j and for $t = 1+\frac{\|E^A_i\|}{\|E^O_j\|}$ the vertex a_{i+1} coincides with b_{j+1}.

2.3.4 Conclusion

In this section we derived, based on the fundamental motion described in Section 2.2, the parameterization of all the elements of the boundary of the forbidden space, which correspond to a single obstacle. Each obstacle in the workspace contributes a pillar-like object, similar to the one depicted in Figure 2.10. Given an obstacle O, the portion of $\mathscr{C}$ bounded "inside" the corresponding pillar-like object is the forbidden space related to O. The portion of the configuration space bounded inside all the pillars, is the forbidden space.

We will conclude the section by noting that most of the patches that we visualize in Figure 2.10 are bounded from all sides by contact patches of dimension one. However, some of them do not; this is due to the fact that we visualize $\mathscr{C}^{\text{geom}} \cong \mathbb{R}^2 \times S^1$. Cases where we had to split Φ (cf. Equation (2.10) on page 18) into two connected components correspond to contact patches that are visualized as disconnected. This is obviously merely an artifact of the visualization, as we have to restrict ourselves to a Euclidean space.

2.4 Rational Parameterization

So far, in our discussion, we considered $\mathscr{C}^{\text{geom}}$ as the model of the configuration space. In other words, we used $\mathbb{R}^2 \times S^1$ to model the configuration space $\mathscr{C}$, such that the first two coordinates correspond to the translation component of the configuration points and the third coordinate corresponds to the *angle* of a rotation. That representation, of the configuration space, provides a strong and direct geometrical intuition, which can be utilized for visualization purposes as we saw earlier. However, this representation introduces one prominent drawback. Namely, even if the coordinates of the vertices of the robot can be represented exactly, the contact surfaces that we derive *cannot* be represented exactly as their parameterization, cf. Equations (2.8), (2.9) and (2.13), involves trigonometric functions. This fact means, for example, that intersecting a contact patch with a line segment in the configuration space cannot yield *exact* results when the geometrical model $\mathscr{C}^{\text{geom}}$ is in use.

By switching to $\mathscr{C}^{\text{rat}}$ as the model of the configuration space, we can overcome this problem, and obtain a rational-based parameterization of the contact surfaces and patches. In other words, we can parameterize the contact surfaces using rational functions. Recall that given a configuration point $\mathbf{q} = (\vec{r}, \theta) \in \mathscr{C}^{\text{geom}}$ with $\theta \in [0, 2\pi)$ we have that

$$\begin{aligned} a_i(\mathbf{q}) &= \vec{r} + R^{\theta} a_i \\ &= \vec{r} + M^{\tau} a_i = a_i(\mathbf{q}') \end{aligned}$$

where $\mathbf{q}' = (\vec{r}, \tau) \in \mathscr{C}^{\text{rat}}$, M^{τ} is the rational rotation matrix and

$$(-\infty, \infty] \ni \tau = \tau(\theta) = \begin{cases} \tan\frac{\theta}{2} & \text{if } \theta \neq \pi \\ \infty & \text{otherwise} \end{cases} \tag{2.16}$$

Similarly to the study in the case where we used $\mathscr{C}^{\text{geom}}$, we obtain a parameterization in $\mathscr{C}^{\text{rat}}$ of a rotation of the robot A such that $a \in \partial A$ is fixed to a point P in the workspace that is given by

$$\mathbf{k}_a(\psi) = (\vec{r}_a(\psi), \tau_a(\psi)) \tag{2.17}$$

with $\vec{r}_a(\psi) = P - M^{\psi} a$ and $\tau_a(\psi) = \psi$. Finally, by either letting $P = P(t) = (1-t)b_j + t b_{j+1}$ vary along an edge of the obstacle or letting $a = a_{i,t} = (1-t)a_i + t a_{i+1}$ vary along an edge of the robot, we obtain two *rational* parameterizations

$$S(t,\psi) = \begin{pmatrix} P(t) - M^{\psi} a_i \\ \psi \end{pmatrix} \tag{2.18}$$

$$S(t,\psi) = \begin{pmatrix} b_j - M^{\psi} a_{i,t} \\ \psi \end{pmatrix} \tag{2.19}$$

with $\psi \in (-\infty,\infty]$ and $t \in (0,1)$ of the (v$_i$-e$_j$) and (e$_i$-v$_j$) contact *surfaces*, respectively.

Next, we want to find a sub-domain $\Psi \subset (-\infty,\infty]$ that will correspond to the contact patches in the configuration space. To that end, let $\phi_{\min}$ and $\phi_{\max}$ be the angels that correspond to either a vertex-edge or edge-vertex case as discussed in Section 2.3.1 and Section 2.3.2, respectively. Using Equation (2.16) we can find the corresponding $\psi_{\min}$ and $\psi_{\max}$. In turn, Ψ can be defined as follows:

$$\Psi = \begin{cases} [\psi_{\min},\psi_{\max}] & \text{if } 0 \le \phi_{\min},\phi_{\max} \le \pi \vee \pi < \phi_{\min},\phi_{\max} < 3\pi \\ [\psi_{\min},\infty] \cup (-\infty,\psi_{\max}] & \text{if } \phi_{\min} \in [0,\pi] \wedge \phi_{\max} \in (\pi,2\pi) \end{cases}$$

Finally, in a similar approach to the one taken when considering the geometrical model of the configuration space, the missing one-dimensional contact patches can be easily determined.

2.5 Differential Geometry of Contact Surfaces

Once one obtains the parameterization of the contact surfaces and patches, it is natural and interesting to study their geometric properties. Better understanding of these properties can be of help in the future to study further the configuration space in general and in the context of the motion planning problem in particular. For example, establishing a discretization of the contact surfaces and patches can prove to be useful. We first, in Section 2.5.1, consider the surfaces and patches embedded in $\mathscr{C}^{\text{geom}}$ and later, in Section 2.5.2, we study the geometry of the boundary of the forbidden space when it is embedded in $\mathscr{C}^{\text{rat}}$.

2.5.1 Geometrical Model

For a single obstacle O, the union of the corresponding contact patches constitute the boundary between the free and forbidden space. In other words, for a configuration point $\mathbf{q} \in S$, for some contact patch S, we have that $A(\mathbf{q})$ touches the obstacle O. Let $\mathbf{q}_1(\varepsilon)$ and $\mathbf{q}_2(\varepsilon)$ be the intersection points of the normal line $\mathbf{q} + t\mathscr{N}_{\mathbf{q}}S$, with $t \in \mathbb{R}$, and the sphere of radius $\varepsilon > 0$ centered at $\mathbf{q}$. Then, there exists some small $\varepsilon_0 > 0$ such that either $\mathbf{q}_1(\varepsilon_0) \in \mathscr{C}_{\text{free}}$ and $\mathbf{q}_2(\varepsilon_0) \in \mathscr{C}_{\text{forb}}$ or vice versa. Here $\mathscr{N}_{\mathbf{q}}S$ denotes the *normal* to the surface S at the point $\mathbf{q} \in S$. Recall, that for a surface $S(t,\phi)$, its unit normal is given by $\frac{\partial_t S \times \partial_\phi S}{\|\partial_t S \times \partial_\phi S\|}$, where $\partial_t S$ and $\partial_\phi S$ are the partial derivatives of S. We however, for the sake of simplicity of the expressions, consider here

$$\mathcal{N}_S(t,\phi) = \partial_t S \times \partial_\phi S$$

without normalization. The following two lemmas can be directly proved.

Lemma 2.4 (Vertex-edge case)**.** *If $S(t,\phi)$ is a vertex-edge contact patch, then we have*

$$\mathcal{N}_S(t,\phi) = -\begin{pmatrix} \vec{n}_j^O \\ \langle E_j^O, R^\phi a_i \rangle \end{pmatrix}$$

Lemma 2.5 (Edge-vertex case)**.** *If $S(t,\phi)$ is a edge-vertex contact patch, then we have*

$$\mathcal{N}_S(t,\phi) = \begin{pmatrix} R^{\frac{\pi}{2}+\phi}(a_{i+1}-a_i) \\ \langle a_{i,t}, a_{i+1}-a_i \rangle \end{pmatrix} = \begin{pmatrix} R^\phi \vec{n}_i^A \\ \langle a_{i,t}, E_i^A \rangle \end{pmatrix}$$

Note that in both cases the normals point towards the free portion of the configuration space. More precisely, if the workspace contains only one obstacle, then the configuration point

$$S(t,\phi) + \varepsilon \mathcal{N}_S(t,\phi),$$

for all $\varepsilon > 0$, is in $\mathcal{C}_{\text{free}}$. The information on the normals of the contact patches can be helpful when one studies the configuration space and address the motion planning problem (cf. [44, 43]).

As we saw in Section 2.3.1, the contact surfaces that correspond to (v-e) contacts are rather simple, namely, *developable* surfaces. Therefore their geometry is rather simple as well – we thus leave their study to Appendix 2.B. Thus, in the remaining of this section we will study the geometrical properties of the (e_i-v_j) contact surfaces. For the sake of simplicity we assume that $P = b_j$ is at the origin. As before, let us assume that $R_0(\mathbf{0})$ is at the origin as well. For $t \in (0,1)$ and $\phi \in [0,2\pi)$ and using Equation (2.13) the contact surface is given by

$$\begin{aligned} S(t,\phi) &= \begin{pmatrix} -R^\phi a_{i,t} \\ \phi \end{pmatrix} \\ &= \begin{pmatrix} -R^\phi a_i \\ \phi \end{pmatrix} + t \begin{pmatrix} -R^\phi \cdot E_i^A \\ 0 \end{pmatrix} \\ &= c(\phi) + t\vec{r}(\phi) \end{aligned}$$

where $a_{i,t} = (1-t)a_i + t a_{i+1}$ as before. Note that $\vec{r}(\phi), \frac{d}{d\phi}\vec{r}(\phi) \neq 0$; this means that the ($e_i$-$v_j$) contact surface is a *non-cylindrical ruled surface* [12].

We start our study of the geometrical properties of the contact surface by computing its first (denoted E, F, G) and second (denoted e, f, g) fundamental forms.

$$E = \|a_i - a_{i+1}\|^2 \qquad e = 0$$
$$F = \det(a_i, a_{i+1}) \qquad f = -\frac{\|a_i - a_{i+1}\|^2}{v}$$
$$G = 1 + \|a_{i,t}\|^2 \qquad g = -\frac{\det(a_i, a_{i+1})}{v}$$

where $v = v(t) = \sqrt{EG - F^2}$. It is easy to verify that $v \neq 0$ for all $t \in \mathbb{R}$, and thus, all expressions are well defined.

Gaussian and mean curvatures. Next, we can easily express the *Gaussian curvature*

$$\begin{aligned} K(t) = K(t,\phi) &= \frac{eg - f^2}{EG - F^2} \\ &= -\frac{\|a_i - a_{i+1}\|^4}{v^4} = -\frac{E^2}{v^4} \end{aligned} \tag{2.20}$$

Note that the Gaussian curvature depends *only* on t (and not on ϕ) as v depends on t, that is the point of contact along E_i^A. Similarly, the *mean curvature* is given by

$$\begin{aligned} H(t) = H(t,\phi) &= \frac{eG - 2fF + gE}{2(EG - F^2)} \\ &= \frac{\|a_i - a_{i+1}\|^2 \det(a_i, a_{i+1})}{2v^3} = \frac{EF}{2v^3} \end{aligned} \tag{2.21}$$

The mean curvature only depends on t, similarly to the case of the Gaussian curvature. Furthermore, if the two vertices, a_i and a_{i+1}, considered as vectors with bases at the origin, are linearly dependent, then the mean curvature vanishes; thus the surface is a *minimal surface*. This yield the following lemma.

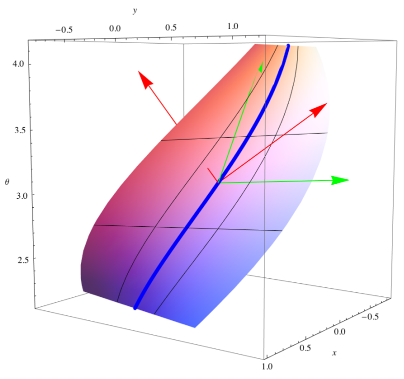
Lemma 2.6. *If the line through $E_i^A(\mathbf{0})$ contains the reference point $R_0(\mathbf{0})$, then the corresponding (e_i-v_j) contact surface is a* minimal surface.

Extrema of curvatures and the striction curve. Both $\frac{d}{dt}K$ and $\frac{d}{dt}H$ vanish for the same single value of t, as can be easily verified, and it is given by

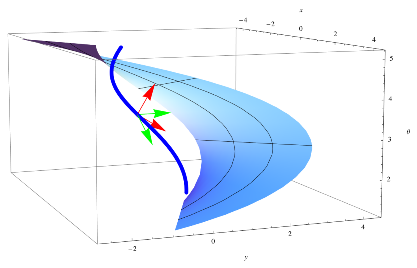
$$t_\star = \frac{\langle a_i, a_i - a_{i+1} \rangle}{E}.$$

We recall that the *striction curve* of a ruled surface can be characterized as the locus of points where the Gaussian curvature attains its extremum [35]. Since $\lim_{t\to\pm\infty} K(t) = 0$ and $K(t) \neq 0$ for all t, we have that $K(t_\star)$ is a global extremum of the Gaussian curvature. Thus we have that the curve $S(t_\star, \phi)$ is the *striction curve* of the contact surface.[3] Next, we can state the following theorem.

[3]The striction curve can also be computed directly, following, for example, the definitions given in [12].



(a) Striction curve is on the patch.



(b) Striction curve is off the patch.

Figure 2.11: Two examples of an (e-v) contact surface. The green arrows are the *asymptotic directions* and the red ones are the *principal curvature directions* for some points on the striction curve. In solid blue is the striction curve.

Theorem 2.7. *For any edge-vertex contact surface the extremum of the Gaussian curvature is*

$$\sup_{t\in\mathbb{R}} |K(t)| = 1$$

and independent of the properties of the robot.

Proof. Since $\sup_{t\in\mathbb{R}} |K(t)| = |K(t_\star)|$, it is enough to compute $K(t_\star)$. To that end, it is sufficient to show that $v(t_\star)^2 = E$; we omit the tedious verification. □

It is easy to verify that the point $a_{t_\star} = (1-t_\star)a_i + t_\star a_{i+1} \in \ell$ is the closest point to the origin among all the points on the line ℓ that contains E_i^A. Thus, it is easy to see that $t_\star$ may not lie in the interval $[0,1]$. That is, the striction curve of the (e-v) contact surface is not necessarily contained in the surface itself. See for example Figures 2.11(a) and 2.11(b).

We conclude this paragraph, with the following observation. In contrast to the Gaussian curvature, the extremum of the mean curvature depends on the properties of the robot. In particular, we have the following

$$\sup_{t\in\mathbb{R}} H(t) = H(t_\star) = \frac{d}{2},$$

where d is the distance of ℓ from the origin and is given by

$$d = \frac{\det(a_i, a_{i+1})}{\|a_{i+1} - a_i\|} = \frac{F}{\sqrt{E}}.$$

The normal curvature and friends. Next, we compute the *normal curvature* of the surface, denoted by κ_N. In general, for a given unit tangent vector $\varepsilon \in T_{\mathbf{q}}S$, the normal curvature in its direction is given by $\kappa_N(\varepsilon) = II(\varepsilon, \varepsilon)$, where $II(\cdot,\cdot)$ denotes the *second fundamental form.* The first step we take is to set an orthonormal coordinate system in the tangent space $T_{\mathbf{q}}S$. Let

$$\begin{aligned} E_1 &= \frac{1}{\sqrt{E}}\frac{\partial S}{\partial t} \\ E_2 &= \frac{1}{\sqrt{E(EG-F^2)}}\left(E\frac{\partial S}{\partial \phi} - F\frac{\partial S}{\partial t}\right) \end{aligned} \tag{2.22}$$

be an orthonormal frame in $T_{\mathbf{q}}S$. Note that E_1 points in the direction of the ruling of the ruled surface S. Next, for each point $\mathbf{q} \in S$ and given some angle ξ we obtain a unit tangent vector

$$\varepsilon(\xi) = E_1 \cdot \cos\xi + E_2 \cdot \sin\xi$$

in $T_{\mathbf{q}}S$. For the sake of completeness, let us note that $\varepsilon(\xi)$ can be expressed in terms of the standard coordinate system $\{\frac{\partial S}{\partial t}, \frac{\partial S}{\partial \phi}\}$ as:

$$\varepsilon(\xi) = \left(\frac{\sqrt{EG-F^2}\cos\xi - F\sin\xi}{\sqrt{EG-F^2}\sqrt{E}}\right)\frac{\partial S}{\partial t} + \frac{\sqrt{E}\sin\xi}{\sqrt{EG-F^2}}\frac{\partial S}{\partial \phi}.$$

Therefore, the normal curvature at $\mathbf{q} \in S(t,\phi)$ in direction ξ, with respect to the orthonormal coordinate system of the tangent plane T_pS, is given by [18]

$$\kappa_N(\xi) = \frac{E\sin\xi(F\sin\xi - 2v\cos\xi)}{v^3}. \tag{2.23}$$

Recall that v depends on t and does not vanish for all $t \in \mathbb{R}$, and therefore Equation (2.23) is well defined. It is easy to note that the normal curvature depends only on the direction in the tangent plane, given by ξ, and the position on the ruling given by t.

Singularities and Extrema of the Normal Curvature. Next, we want to find the *asymptotic directions* and the *principal curvature directions* of the contact surface. In other words we want to find the values of ξ for which the normal curvature either vanishes or attains an extremum. Furthermore, we find the extrema values of the normal curvature, that is, expressions of the *principal curvatures.*

As the rulings on the surface are straight lines, the normal curvature in their directions should vanish, and indeed for $\xi \in \{0, \pi\}$ the normal curvature vanishes. The other direction where it vanishes corresponds to

$$\xi = \arctan\frac{2v}{\det(a_i, a_{i+1})} = \arctan\frac{2v}{F} \tag{2.24}$$

That is, $\varepsilon(0)$ and $\varepsilon(\arctan\frac{2v}{F})$ point in the asymptotic directions of the surface. The green arrows in Figure 2.11 point in the directions where the normal curvature vanishes. Note, that one of the arrows in each figure is parallel to the ruling's direction.

Remark 2.6. Note that if S is a minimal surface (and thus also a helicoid), then the denominator in Equation (2.24) vanishes. Thus, we have that the second direction for which the normal curvature vanishes corresponds to $\xi = \pi/2$. This coincide with the fact that for the helicoid the asymptotic directions are orthogonal [20].

We now want to find the principal curvature directions and to that end we have to derive $\kappa_N(\xi)$ with respect to ξ. This yields

$$\frac{\partial}{\partial \xi}\kappa_N(\xi) = \frac{2E}{v^3}(F \sin\xi \cos\xi - v \cos 2\xi).$$

In turn, it can be verified that that for

$$\xi_1 = \frac{1}{2}\arctan\frac{2v}{F}$$

the normal curvature attains an extremum. Since the principal curvature directions are orthogonal, we have that it attains its other extremum for $\xi_2 = \xi_1 + \frac{\pi}{2}$. In Figure 2.11 the red arrows point in the principal curvature directions. Finally, the principal curvatures are

$$\kappa_1 = \kappa_N(\xi_1) \quad , \quad \kappa_2 = \kappa_N(\xi_2) \tag{2.25}$$

We can simplify the expressions of the principal curvatures as follow

$$\begin{aligned}\kappa_1 &= -\frac{2E}{v\left(F + \sqrt{F^2 + 4v^2}\right)}\\ \kappa_2 &= \frac{E(F + \sqrt{F^2 + 4v^2})}{2v^3}.\end{aligned}$$

One can easily verify that $K = \kappa_1\kappa_2$ as in Equation (2.20) and $H = \frac{\kappa_1+\kappa_2}{2}$ as in Equation (2.21).

Remark 2.7. The principal curvatures could have been computed directly using formulas that base on the fundamental forms. We took a longer path as we wanted to obtain expressions for the principal curvature directions as well.

More general Point of View. In Section 2.3.2 we derived a parameterization of an (e-v) contact surface. We can assume, without loss of generality, that the edge E_i^A, when in the rest position, is horizontal and at distance d from the origin. Let ℓ be the line that contains the edge. Furthermore, let $a_t = (t, d)$ be a varying point on the line ℓ. Using this more general point of view, we can show that the properties of the contact surface depend on the distance of the edge of the robot from the origin. For an illustration see Figure 2.12. In Equation (2.8) we can replace a with a_t, and set $P = (0,0)$, thus yielding the following parameterization in $\mathscr{C}$

$$S(t,\phi) = \begin{pmatrix} -R^{\phi} a_t \\ \phi \end{pmatrix} \tag{2.26}$$

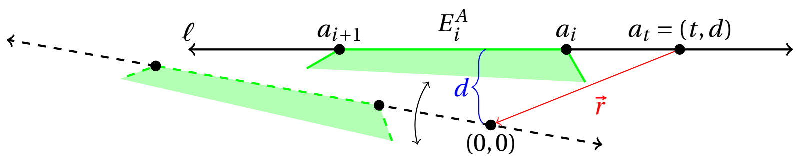


Figure 2.12: An infinite edge with a vertex a_t on it in solid and its translated and rotated pose where a_t coincides with the origin.

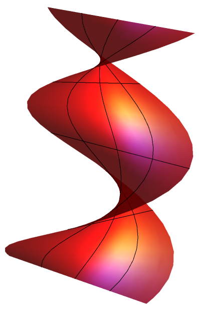


Figure 2.13: An example of a general edge-vertex contact surface for $d = 0.5$.

with $\phi \in [0, 2\pi)$ and $t \in \mathbb{R}$. In Figure 2.13 an example of such generalized edge-vertex contact surface is plotted.

We conclude this section with the study of the geometry of the general contact surface that we introduced in Equation (2.26). In particular, we derive expressions of the Gaussian, mean and normal curvatures, as well as the principal curvatures directions and their magnitudes. For a *fixed* distance d we have the following expressions

$$\begin{aligned} K &= -\frac{1}{(1+t^2)^2} \\ H &= -\frac{d}{2\sqrt{(1+t^2)^3}}, \end{aligned}$$

for the Gaussian and mean curvatures, respectively. As before, the normal curvature in the direction of a unit tangent vector $\varepsilon(\xi)$ is

$$\kappa_N = -\frac{\sin\xi(2\sqrt{1+t^2}\cos\xi + d\sin\xi)}{\sqrt{(1+t^2)^3}}$$

where ξ parameterizes the direction in the tangent space of the unit vector $\varepsilon(\xi)$. It comes as no surprise that all the curvatures depends only on t. In terms of the motion of the robot, it means that the curvatures of an (e-v) contact surface depends only on the position of a_t on the line ℓ, and *not* on the angle that the line forms with respect to the x-axis. When thinking of the origin as a vertex of an obstacle, then the above means that the curvature depends only on the point of contact on the edge of the robot and not on the angle of the contact. Furthermore, the Gaussian curvature does not even depend on the distance of the edge from the origin; i.e. it does not depend on the properties of the robot. Note that for $d = 0$ the mean curvature vanishes; that is, the corresponding surface is a minimal one. Recall that $d = 0$ means that the general edge is passing through the origin and this coincides with the result of Lemma 2.6.

We have that the normal curvature vanishes for

$$\xi_0 = 0 \quad , \quad \xi_1 = \arctan\left(-\frac{2\sqrt{1+t^2}}{d}\right).$$

and its derivative with respect to ξ vanishes for

$$\xi_2 = \frac{1}{2}\xi_1 \quad , \quad \xi_3 = \frac{1}{2}\xi_1 + \frac{\pi}{2}.$$

Therefore at the point $S(t,\phi) \in S$, the vectors $\varepsilon(\xi_0)$ and $\varepsilon(\xi_1)$ point in the asymptotic directions and $\varepsilon(\xi_2)$, $\varepsilon(\xi_3)$ point in the principal curvature directions. Finally, the principal curvatures are

$$\kappa_1 = \kappa_N(\xi_2) = \frac{-d + \sqrt{4 + d^2 + 4t^2}}{2\left(1+t^2\right)^{3/2}}$$

$$\kappa_2 = \kappa_N(\xi_3) = -\frac{d + \sqrt{4 + d^2 + 4t^2}}{2\left(1+t^2\right)^{3/2}}$$

Let us conclude by pointing out that both the Gaussian and the mean curvatures attain their extrema for $t = 0$. For the Gaussian curvature we have that its extremum is -1 and the extremum of the mean curvature is $\frac{d}{2}$. The striction curve of the general edge-vertex contact surface is attained when fixing $t = 0$ where the point a_t with $t = 0$ is the closest point of the line ℓ to the origin. All these results coincide with previous discussions in this section.

2.5.2 Rational Model

In this section we will derive expression of the Gaussian and mean curvatures of edge-vertex contact surfaces that arise when using the rational model for the configuration space. The case of vertex-edge contact is simpler and for the sake of brevity is omitted.

From Equation (2.19) we have that a general edge-vertex contact surface is given by

$$S(t,\psi) = \begin{pmatrix} b_j - M^{\psi} \cdot a_{i,t} \\ \psi \end{pmatrix}$$

when modeling the configuration space using the rational model. It is easily shown that the partial derivatives are given by

$$\frac{\partial S}{\partial t} = \begin{pmatrix} -M^{\psi} \cdot (a_{i+1} - a_i) \\ 0 \end{pmatrix} \quad , \quad \frac{\partial S}{\partial \psi} = \begin{pmatrix} -\frac{d}{d\psi} M^{\psi} \cdot a_{i,t} \\ 1 \end{pmatrix}$$

In turn, the fundamental forms are given by

$$
\begin{aligned}
E &= \|a_i - a_{i+1}\|^2 & e &= 0 \\
F &= \frac{2}{1+\psi^2}\det(a_i, a_{i+1}) & f &= -\frac{2}{1+\psi^2}\frac{\|a_i - a_{i+1}\|^2}{v} \\
G &= 1 + \frac{4}{(1+\psi^2)^2}\|a_{i,t}\|^2 & g &= -\frac{4}{(1+\psi^2)^2}\frac{\det(a_i, a_{i+1}) + \psi\,\langle a_i - a_{i+1}, a_{i,t}\rangle}{v}
\end{aligned}
$$

Here, $v = v(t,\psi) = \sqrt{EG - F^2}$. Note that in contrast to the case discussed in Section 2.5.1, here v depends on both t and ψ. Now, using the standard formulas for the Gaussian and mean curvatures, we can obtain the following expressions:

$$
\begin{aligned}
K(t,\psi) &= -\frac{4}{(1+\psi^2)^2}\frac{E^2}{v^4} \\
H(t,\psi) &= \frac{2E}{(1+\psi^2)v^3}\left(\frac{1}{2}F - \frac{\psi}{1+\psi^2}\langle a_i - a_{i+1}, a_{i,t}\rangle\right)
\end{aligned}
$$

In this case, when using the rational model, both curvatures depend on *both* parameters (cf. Equations (2.20) and (2.21) when using the geometric model). Note that

$$
\begin{aligned}
\lim_{t\to\pm\infty} K(t,\psi) &= \lim_{\psi\to\pm\infty} K(t,\psi) = 0 \\
\lim_{t\to\pm\infty} H(t,\psi) &= \lim_{\psi\to\pm\infty} H(t,\psi) = 0
\end{aligned}
$$

2.6 Conclusion

In this chapter we discussed prominent aspects of the geometry of contact surfaces in the configuration space that arises when a planar convex polygon is free to translate and rotate amid planar polygonal obstacles. We considered two models of the configuration space, namely, the geometrical one, which is more intuitive, and the rational one, which is more suitable when machine-precision is needed. We started the discussion by providing explicit and concise parameterizations for the surfaces that correspond to *vertex-edge* and *edge-vertex* contacts. Once the parameterizations were formulated we carried out standard differential geometry computations and derived compact expressions of important notions like the Gaussian and mean curvatures, principal curvatures and principal curvature directions, etc.

Application. Using the parameterization described in this chapter we produced a short video [3], which visualizes the configuration space of a convex polygonal robot that moves amid convex polygonal obstacles in the plane. The video serves as an educational tool when studying the notion of configuration space.

Future work. The computations of the contact *patches* heavily rely on the assumption that the robot A is convex. Eliminating this restriction is of interest.

Other interesting extensions of the parameterization would be to consider a robot with a boundary that consists of non-linear edges; for example circular arcs (cf. [30]).

Once the contact surfaces and patches can be explicitly parameterized, it is natural to consider their discretizations. An approximated version of the configuration space can be used to address the general problem of *motion planning*. Furthermore, given either the smooth or discrete representations of the contact surfaces or patches, it is possible to study their mutual intersections and their intersections with other elements in the configuration space, as done for example in [36]. This study can be of help in the investigation of the arrangement of the contact surfaces in the configuration space.

Note that the vertex-edge contact surface, see Equation (2.13), reduces to a vertical plane if a_i is at the origin; or in other words, if the robot's vertex under consideration coincides with the reference point, which is assumed to be at the origin. Furthermore, the edge-vertex contact surface reduces to a minimal surface if the reference point lies on the line containing the edge of the robot; in particular, if the reference point coincides with one of the vertices of the edge. On the other hand, the reference point can be chosen arbitrarily. These observations suggest that by perturbing the reference point one could reduce the geometrical complexity of the contact surfaces. However, for each such choice of the reference point there corresponds a configuration space. Given a contact type, one can set the reference point such that the corresponding contact surface would be as simple as possible. The problem in this case is that one has to deal with many copies of the configuration space and understand how are they related.

2.A Angle Range Analysis Using Normals

In Section 2.3, given a contact type, we first parameterized the corresponding contact surface. Then, we found a sub-domain Φ for the parameter ϕ such that $S(t,\phi)_{|\phi\in\Phi}$ for $t \in (0,1)$ yielded the corresponding contact patch. As we stressed, the described computations cannot yield an exact representation of Φ. In this section, we pose further assumptions on the setting of the robot, and in turn we will be able to find the desired sub-domain without involving trigonometric functions. Thus, providing an exact representation of Φ. We discuss in this section both the vertex-edge and edge-vertex cases.

2.A.1 Vertex-Edge Case

Let i, j be two indices and consider the contact type (v_i-e_j) and let $S(t,\phi)$ be the corresponding contact surface as given in Equation (2.9) on page 16 with $t \in (0,1)$ and $\phi \in [0,2\pi)$. We want to find $\Phi \subset [0,2\pi)$ such that for all $\phi \in \Phi$ we will have that $A(\mathbf{q}_{a_i}(\phi))$ *touches* E^O_j. As we already saw, the sub-domain Φ does not depend on the point of contact on the edge E^O_j of the obstacle. Thus, for some fixed $t_0 \in (0,1)$ let $\mathbf{q}_i(\phi) = S(t_0,\phi)$ be the helix that is contained in S and that corresponds to a pseudo contact between a_i and the point

$$P = P(t_0) = (1-t_0)b_j + t_0 b_{j+1}$$

on the edge E^O_j of the obstacle O. We will find Φ for this single helix. More explicitly, for every ϕ we have that $a_i(\mathbf{q}_i(\phi))$ pseudo touches the fixed point P on the edge E^O_j. We want to find the values of ϕ for which $a_i(\mathbf{q}_i(\phi))$ will *touch* P.

In order to obtain the bounds on the angle ϕ for which $a_i(\mathbf{q}_i(\phi))$ merely touches P, we have to check the angles between the (inward) normals $\vec{n}^A_i, \vec{n}^A_{i-1}$ of the robot and $\vec{n}^O_j$ of the obstacle; cf. Figure 2.6 on page 17. As both the robot and the obstacle are convex and we assume a counterclockwise order of their vertices, we can establish the following observation.

Observation 2.8. The vertex $a_i(\mathbf{q}_i(\phi)$ touches the point $P \in E^O_j$ if and only if when traversing the circle of direction counterclockwise then the normals are ordered as follow $\vec{n}^A_{j-1}$, $-\vec{n}^O_j$ and $\vec{n}^A_j$.

The condition in the observation can be expressed as the following system of inequalities:

$$\begin{cases} \det\left(\vec{n}^A_{i-1}(\mathbf{q}_i(\phi)), -\vec{n}^O_j\right) \geq 0 \\ \det\left(-\vec{n}^O_j, \vec{n}^A_i(\mathbf{q}_i(\phi))\right) \geq 0 \end{cases} \tag{2.27}$$

Recall that $\det(u,v)$ denotes the determinant of the 2×2 matrix formed by the two vectors u and v. Note that u, v need not be of unit length as we are merely interested in the sign of the determinant. This systems means that $\vec{n}^A_{i-1}(\mathbf{q}_i(\phi))$ and $-\vec{n}^O_j$ form a *right-turn* and the normal $-\vec{n}^O_j$ forms a *right-turn* with the normal $\vec{n}^A_i(\mathbf{q}_i(\phi))$ as well (see Figure 2.6).

For each $\phi \in [0, 2\pi)$ we have, using Equation (2.3), the following

$$a_{i\pm1}(\mathbf{q}_i(\phi)) = P - R^{\phi} a_i + R^{\phi} a_{i\pm1}.$$

In turn, the edge normals of the robot satisfy

$$\begin{aligned} \vec{n}^A_{i-1}(\mathbf{q}_i(\phi)) =& R^{\frac{\pi}{2}}(a_i(\mathbf{q}_i(\phi)) - a_{i-1}(\mathbf{q}_i(\phi)) \\ \vec{n}^A_i(\mathbf{q}_i(\phi)) =& R^{\frac{\pi}{2}}(a_{i+1}(\mathbf{q}_i(\phi)) - a_i(\mathbf{q}_i(\phi)) \end{aligned}$$

Since the normal $-\vec{n}^O_j$ is stationary, and does not depend on the configuration space we refer to it as a constant vector, given by $-\vec{n}^O_j = (\cos\beta_j, \sin\beta_j)$ for some fixed $\beta_j \in [0, 2\pi)$. The system of inequalities from Equation (2.27) reduces to the following system

$$\begin{cases} r_{i-1}\cos(\phi + \alpha_{i-1} - \beta_j) - r_i \cos(\phi + \alpha_i - \beta_j) \geq 0 \\ -r_i \cos(\phi + \alpha_i - \beta_j) + r_{i+1}\cos(\phi + \alpha_{i+1} - \beta_j) \geq 0 \end{cases} \tag{2.28}$$

Although we obtained here a rather compact system of inequalities, it is, in general, hard to analytically reduce it further, and obtain the desired domain for ϕ. However, assuming $r_j =$ const for all j's, that is all the vertices of the robot lie on a circle of a fixed radius, we can use the *cosine difference* formula and reduce the system of inequalities further to obtain

$$\begin{cases} \sin\left(\dfrac{2\phi - 2\beta_j + \alpha_{i-1} + \alpha_i}{2}\right)\sin\left(\dfrac{\alpha_{i-1} - \alpha_i}{2}\right) \leq 0 \\ \sin\left(\dfrac{2\phi - 2\beta_j + \alpha_{i+1} + \alpha_i}{2}\right)\sin\left(\dfrac{\alpha_{i+1} - \alpha_i}{2}\right) \leq 0 \end{cases} \tag{2.29}$$

In this last system of inequalities both

$$\sin\left(\frac{\alpha_{i-1} - \alpha_i}{2}\right) \text{ and } \sin\left(\frac{\alpha_{i+1} - \alpha_i}{2}\right)$$

depend only on the properties of the robot and their sign can be easily found. Indeed, for $1 < i < n$ we have that $0 \leq \alpha_{i-1} \leq \alpha_i < 2\pi$ and in turn $\sin\frac{\alpha_{i-1}-\alpha_i}{2} \leq 0$. On the other hand, since $0 < \alpha_i \leq \alpha_{i+1} < 2\pi$ we have that $\sin\frac{\alpha_{i+1}-\alpha_i}{2} \geq 0$. Thus, the inequality in (2.29) holds if ϕ satisfies the following constraints.

Condition 2.9 ($1 < i < n$)**.**

$$\begin{cases} \beta_j - \dfrac{\alpha_{i-1} + \alpha_i}{2} \leq \phi \leq \pi + \beta_j - \dfrac{\alpha_{i-1} + a_i}{2} \\ \pi + \beta_j - \dfrac{\alpha_{i+1} + \alpha_i}{2} \leq \phi \leq 2\pi + \beta_j - \dfrac{\alpha_{i+1} + \alpha_i}{2} \end{cases}$$

The cases $i = 1$ and $i = n$ have to be treated separately. That is, depending on the signs of $\sin\frac{\alpha_{i-1}-\alpha_i}{2}$ and $\sin\frac{\alpha_{i+1}-\alpha_i}{2}$ the inequalities can be reduced into a condition similar to the one obtained for $1 < i < n$.

Let us stress again that the compact and exact representation of the range of ϕ as described in Condition 2.9 is valid only if the three vertices a_{i-1}, a_i and a_{i+1} of the robot have the same distance from the reference point. In order to obtain a complete description of the contact patches using this approach, all the vertices of the robot have to lie on a circle of some fixed radius. If the vertices do not lie on a circle, then the approach described in Section 2.3.1 can be used.

Before ending this section, let us consider Equation (2.28) again. The condition that we posed in order to solve that system, namely, $r_j = \text{const}$ for all j's, is very restrictive. Note, that for a given (v$_i$-e$_j$) contact it is enough to assume that $r_{i-1} = r_i = r_{i+1}$. Or, in other words, we can assume that a_{i-1}, a_i and a_{i+1} lie on a circle centered at R_0. However, since the reference point can be chosen arbitrarily, we can set it at the center of the circle defined by a_{i-1}, a_i and a_{i+1}. The problem with this approach is that the configuration space depends on the choice of the reference point. Differently said, this approach yields a domain of ϕ that *depends* on the specially chosen reference point. Applying this approach to a different contacts yields a domain of ϕ that will be correct only for the corresponding configuration space.

2.A.2 Angle Range Analysis Using Normals - Edge-Vertex

Next, let us introduce an obstacle O and we rotate the robot A such that its boundary point $a_{i,t} = (1-t)a_i + a_{i+1}$ is fixed to the obstacle's vertex b_j. We will now find the range of ϕ for which this pseudo contact is merely a contact. Formally, for some fixed t_0 this means that we will find the ϕ's for which $\mathbf{q}_i(\phi) = S(t_0, \phi)$ is in $\mathscr{C}_{\text{free}}$ and $S(\cdot,\cdot)$ is the surface defined in Equation (2.13) on page 19. Like before, due to the consistent ordering of the vertices and the convexity of the objects the desired free configuration is obtained when $\vec{n}_i^A$ is between $-\vec{n}_{j-1}^O$ and $-\vec{n}_j^O$. Equivalently, when $-\vec{n}_{j-1}^O$ forms a *right-turn* with $\vec{n}_i^A$ and $\vec{n}_i^A$ forms another *right-turn* with $-\vec{n}_j^O$ (see Figure 2.7 on page 20). Thus, in order to maintain a *contact* between E_i^A and the vertex b_j of the obstacle, the angle parameter ϕ has satisfy the following system of inequalities:

$$\begin{cases} \det\left(-\vec{n}_{j-1}^O, \vec{n}_i^A(\mathbf{q}_i(\phi))\right) \geq 0 \\ \det\left(\vec{n}_i^A(\mathbf{q}_i(\phi)), -\vec{n}_j^O\right) \geq 0 \end{cases}$$

For the sake of simplicity, we assume that $-\vec{n}_{j-1}^O = \left(\cos\beta_{j-1}, \sin\beta_{j-1}\right)$ and $-\vec{n}_j^O = \left(\cos\beta_j, \sin\beta_j\right)$ for some fixed β_{j-1} and β_j. In addition, we have that

$$\vec{n}_i^A(\mathbf{q}_i(\phi) = R^{\phi+\frac{\pi}{2}} \cdot (a_{i+1} - a_i).$$

In turn, the system of inequalities reduces to

$$\begin{cases} r_i \cos(\alpha_i - \beta_{j-1} + \phi) - r_{i+1} \cos(\alpha_{i+1} - \beta_{j-1} + \phi) \geq 0 \\ - r_i \cos(\alpha_i - \beta_j + \phi) + r_{i+1} \cos(\alpha_{i+1} - \beta_j + \phi) \geq 0 \end{cases}$$

Next, assuming that $r_i = r_{i+1} > 0$, the last system of inequalities reduces to:

$$\begin{cases} \sin(\phi - \beta_{j-1} + \frac{\alpha_i + \alpha_{i+1}}{2}) \sin \frac{\alpha_i - \alpha_{i+1}}{2} \leq 0 \\ \sin(\phi - \beta_j + \frac{\alpha_i + \alpha_{i+1}}{2}) \sin \frac{\alpha_{i+1} - \alpha_i}{2} \leq 0 \end{cases}$$

Depending on i, the signs of $\sin \frac{\alpha_i - \alpha_{i+1}}{2}$ and $\sin \frac{\alpha_{i+1} - \alpha_i}{2}$ can be determined. In turn, the system of inequalities can be reduced and an expression for the range of ϕ can be obtained in a similar way to the one detailed in Section 2.A.1.

2.B Differential Geometry of V-E Contact Surfaces

In general a vertex-edge contact surface is given by the parameterization in Equation (2.9) on page 16. One can easily compute the first fundamental forms of this surface, namely,

$$\begin{aligned} E &= \|E_j^O\|^2 \\ F &= -\langle E_j^O, R^{\phi + \frac{\pi}{2}} a_i \rangle \\ G &= 1 + \|a_i\|^2 \end{aligned} \tag{2.30}$$

Next, the surface normal is given by

$$\mathcal{N}_S(t, \phi) = \frac{1}{\sqrt{\|b_{j+1} - b_j\|^2 + \langle \cos\phi\, a_i + \sin\phi\, R^{\frac{\pi}{2}} a_i, b_{j+1} - b_j \rangle}} \begin{pmatrix} R^{-\frac{\pi}{2}} (b_{j+1} - b_j) \\ - \langle \cos\phi\, a_i + \sin\phi\, R^{\frac{\pi}{2}} a_i, b_{j+1} - b_j \rangle \end{pmatrix}$$

In turn, we can find expressions of the second fundamental forms, namely,

$$\begin{aligned} e &= 0 \\ f &= 0 \\ g &= \frac{\langle \cos\phi\, R^{\frac{\pi}{2}} a_i - \sin\phi\, a_i, E_j^O \rangle}{\sqrt{\|E_j^O\|^2 + \langle \cos\phi\, a_i + \sin\phi\, R^{\frac{\pi}{2}} a_i, E_j^O \rangle^2}} \end{aligned} \tag{2.31}$$

It is now easy, using the formula given in Equation (2.20) on page 27, to verify that the Gaussian curvature of a vertex-edge contact surface vanishes; that is such a surface is developable. One can next use the standard formula of the mean curvature, as can be found in Equation (2.21). The resulting expression is rather complicated, and of little interest, however, its numerator can be expressed as follows

$$\|E_j^O\|^2 \langle E_j^O, R^{\phi+\frac{\pi}{2}} a_i\rangle \, .$$

Since the edges of the obstacles are not degenerated, we can conclude that the mean curvature is identically zero if and only if $a_i = \vec{0}$. In this case, it is easily seen, from the parameterization of the contact surface directly, that the surface is a plane.

3 Optimal Triangulation of Hyperbolic Paraboloids

3.1 Introduction

In this chapter we consider the problem of local triangulation of hyperbolic paraboloid surfaces. These surfaces are given by $\{(x, y, z) : z = F(x, y)\}$, where $F(x, y)$ is a bivariate polynomial as given in Equation (3.2) that satisfies the conditions expressed in (3.3). By "local" we mean that given an arbitrary point on such a surface, our goal is to find a corresponding collection $\mathscr{T}$ of triangles that forms a triangulation that is homeomorphic to a neighborhood of the given point. In order to turn this problem into a well-posed one, we have to impose further restrictions on the desired family $\mathscr{T}$. In particular, we establish a local triangulation of such surfaces that is optimal in two senses. First, we look for optimality with respect to the so-called *vertical distance* criterion. To that end, let $f : D \to \mathbb{R}$ be a piecewise linear function that defines $\mathscr{T}$, where $D \subset \mathbb{R}^2$ is its domain. Given some $\varepsilon > 0$ we demand that

$$\max_{(x,y)\in D} |F(x, y) - f(x, y)| \le \varepsilon.$$

In other words, the vertical distance (cf. Section 3.4) between each triangle $T \in \mathscr{T}$ and the surface has to be bounded by ε. In addition, we want to minimize the number of triangle in D that $\mathscr{T}$ comprises while maintaining the first constraint. Note that this number is inversely proportional to the area of the triangles in $\mathscr{T}$. Thus, in other words, we want to maximize the area of the triangles in $\mathscr{T}$. Together, these two constraints form a well-posed problem that we study in this chapter. Furthermore, we shall consider both an interpolating triangulation and a non-interpolating one. In particular, the vertices of the triangles that constitute $\mathscr{T}$ could either lie on the saddle surface, and form an *interpolating* triangulation. Or, the vertices could lie in space, thus forming a *non-interpolating* triangulation. Note, that in either of the cases we still want to maintain the two conditions, namely, optimality with respect to the vertical distance and maximization of the area of the triangles.

Motivation. The original motivation of this work is the piecewise linear approximation of a *negatively curved* smooth surface in three dimensions, with respect to the *Hausdorff distance.* We would like to find an approximation with the minimum number of triangular patches for a given error bound, or minimize the error subject to a bound on the number of triangles. As a secondary criterion, we might consider the shape of the triangles we are using.

Locally, around a surface point p, a smooth surface can be approximated (up to second order) as the graph of a quadratic function. This is achieved by translating p to the origin, rotating the coordinate system to make the tangent plane horizontal, and a second order Taylor expansion.

In particular, an arbitrary point p on a smooth surface S has a unit normal $\mathcal{N}_p S$ defined. Applying an appropriate Euclidean motion, $\mathcal{T}$, on the space will yield a surface $\tilde{S}$ such that $\mathcal{T}(p)$ will be at the origin and the corresponding normal, $\mathcal{N}_{\mathcal{T}(p)}\tilde{S}$ will point in a vertical direction. The map $\mathcal{T}$ consists of a translation component $\mathcal{T}_1$ for which $\mathcal{T}_1(x) = x - p$ for all $x \in \mathbb{R}^3$. The second part, is a rotation, $\mathcal{T}_2$, which maps the normal $\mathcal{N}_{\mathcal{T}_1(p)}\mathcal{T}_1(S)$ to a vertical pointing normal $\pm(0,0,1)$.

This means that when considering a local approximation of a smooth surface S around a point $p \in S$, it is safe to assume that p is at the origin, the tangent plane at p is horizontal and its intrinsic neighborhood in S is a graph of a bivariate function. More explicitly, this means that there exists a domain $D \subset \mathbb{R}^2$ and a function $F_1\colon D \to \mathbb{R}$, such that the graph of F_1 over the domain D coincides with the surface S. Note that the smoothness of the surface S implies the smoothness of the function F_1. Therefore, we can use the *Taylor expansion* up to the second order terms of F_1, and obtain a quadratic approximating function $F\colon U \to \mathbb{R}$, for some neighborhood $U \subset D$, of the function F_1. In particular, this approximating function is a quadratic function of the following form

$$F(x,y) = a_1x^2 + 2a_2xy + a_3y^2. \tag{3.1}$$

Note that F has no linear terms since the normal at the origin is aligned with the vertical direction. Furthermore, it has no constant term, since it satisfies $F(0,0) = 0$. The *quadratic surface* $z = F(x,y)$ yields a second order approximation of the surface S in a neighborhood of the origin. Finally, a local approximation of the graph of a quadratic function in two variables can be transformed, using the inverse of the transformation $\mathcal{T}$, into a local approximation around an arbitrary point of a smooth surface.

Since the tangent plane to a quadratic surface, as above, at the origin is horizontal, the distance to the closest point on an approximating piecewise linear surface (which determines the *Hausdorff distance*) is close to vertical. In fact, the *vertical* distance between the graph of $z = F(x,y)$ and an approximating surface is always an upper bound for the Hausdorff distance. The quality of this approximation depends on the normal to the surface at the point of interest. We shall use the tangent plane at the origin to approximate the surface in a neighborhood of this point. Therefore, our investigations provide a good local model for the original approximation problem.

The choice of the vertical distance as a distance measure has a crucial advantage. With respect to vertical distance approximation, *all points of a quadratic surface $z = F(x, y)$ look equal,* in the follow sense: For every two points p and q on the surface, there is an affine transformation that (i) maps p to q, (ii) maps the surface to itself, (iii) maps vertical lines vertical to vertical lines, and (iv) leaves vertical distances unchanged. This is further discussed in Section 3.4. Therefore, analysing the local situation around one point, the origin, is sufficient to analyze the whole surface. As a result, in the case of an interpolating approximation we are able to completely *characterize the optimal triangulations of quadratic surfaces* defined over the whole plane $D = \mathbb{R}^2$. Here we mean optimality with respect to the error. Furthermore, in this work we consider a non-interpolating scheme for the triangulation, which yields an improvement of the interpolating one. More precisely, by allowing the vertices of the triangles to lie in space, we can obtain *larger* triangles, while still maintaining the vertical distance constraint.

It turns out that there is a one-parameter family of optimal triangulations; this is in addition to two parameters for translating the mesh in the plane, which does not change the (planar) shape of the triangles (see Lemma 3.9 on page 60). Of course, this result cannot be applied directly to a triangulation over a domain D, but it is relevant as a local result, when the boundary of D is far away. The freedom of choosing one shape parameter could be used to facilitate the adaptation and grading of the triangulation over a larger surface area with varying curvature parameters. This will, however, be left for future work. We also investigate the choice of the parameter that yields the optimal shape of the triangles, in the sense of maximizing the smallest angle in the plane projection (See Section 3.5.1).

The optimization problem that we pose is to use the smallest number of triangles for a given bound ε on the vertical approximation error. Since the number of triangles for triangulating a domain is inversely proportional to the average area of a triangle, the problem boils down to finding the *largest-area triangle* subject to the error constraint. This triangle and its reflected copy can be used to tile the plane. Since the situation is invariant under rigid motions in the plane (as aforementioned and described in more detail in Section 3.5.1 below), this triangular mesh will give an optimal triangulation of the whole surface, defined over $\mathbb{R}^2$, in terms of the number of triangles per area.

Previous work. The study of approximation and triangulation of surfaces in general is a fundamental research topic in fields like computer graphics, computational geometry, etc. Thus, it comes as no surprise that the approximation and triangulation of saddle surfaces was addressed in the past. In this section we survey the literature and focus on works that provide theoretical results with respect to the quality of the yielded approximation.

A general approximation framework is the so-called *data depended triangulation.* Here data points are sampled from the domain $\mathbb{R}^d$ together with their corresponding values. Then a triangulation of the data points is found such that the corresponding triangulation of the hypersurface is optimal with respect to a given criteria. In two dimensions, Dyn et al. [15] provided an algorithm that triangulates the data points depending on their corresponding data values. Melissaratos [29] applied the same framework to the (convex) case of hyper-paraboloids and proved that the Delaunay triangulation (of the data points) is the optimal triangulation with respect to the L_p norm. Desnoguès and Devillers [14] provided a locally optimal triangulation of the hyperbolic paraboloid with respect to the L^2 norm. The work of Pottmann et al. [34] treats all possible quadratic functions and considers the L^∞ norm as the error criterion. Pottmann et al. conjecture that interpolating approximation is mandatory for an optimal piecewise linear approximation; in Section 3.6 we invalidate this conjecture. In contrast to Pottmann et al., in the presented work we employ a strong geometric intuition, which yields, nevertheless, similar results.

Outline of the Chapter. In Sections 3.2 to 3.4 we introduce the fundamental notions that we shall consider in this chapter. In particular we review some elementary notions and properties of conic sections and quadratic surfaces. Furthermore, we present in details the notion of *vertical distance* that we use as the fundamental criteria of our approximation. The core of the chapter can be found in Section 3.5 where we first introduce the local approximation of a simple saddle is first established. The discussion is followed by the local approximation of the canonical and general saddles as well. In Section 3.6 we allow a non-interpolating triangulation, which results in an improved error. Discussion and conclusions can be found in Section 3.7.

3.2 Remarks on Conic Sections

In this section we recall some notations and fundamental facts related to *conic sections*; that is, planar curves that are the zero sets of the form $F(x,y)=0$ for some bivariate polynomial

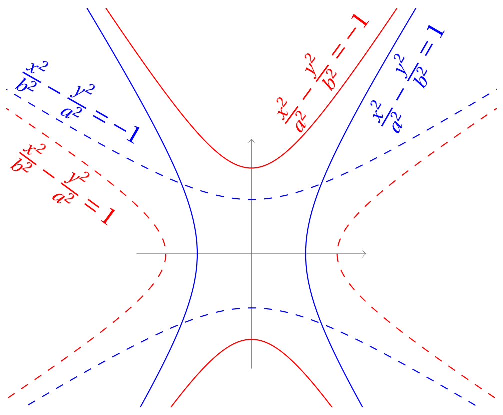
$$F(x,y) = a_{11}x^2 + 2a_{12}xy + a_{22}y^2 + 2a_{13}x + 2a_{23}y + a_{33}. \tag{3.2}$$

In this paper, we are mainly interested in the case of *hyperbolic* conic sections (called *hyperbolas* for short), which are those that correspond to polynomials whose coefficients satisfy the following conditions

$$D = \begin{vmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{vmatrix} < 0 \quad , \quad A = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{vmatrix} \neq 0. \tag{3.3}$$

Each hyperbola has two asymptotes that intersect at a point called the *center* of the hyperbola. The coordinates of the center, (x_c, y_c), of a generic hyperbola are given by

$$x_c = -\frac{1}{D}\begin{vmatrix} a_{13} & a_{12} \\ a_{23} & a_{22} \end{vmatrix} \quad , \quad y_c = -\frac{1}{D}\begin{vmatrix} a_{11} & a_{13} \\ a_{12} & a_{23} \end{vmatrix}. \tag{3.4}$$



$\frac{x^2}{b^2} - \frac{y^2}{a^2} = -1$
$\frac{x^2}{b^2} - \frac{y^2}{a^2} = 1$
$\frac{x^2}{a^2} - \frac{y^2}{b^2} = -1$
$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$

Figure 3.1: Two hyperbolas and their conjugates for some $a < b$.

The image of the asymptotes is the zero set of $F(x, y) - \frac{A}{D}$. Together, D and A are invariants of the hyperbola under translations and rotations (the above notation was inspired by [22]).

A principal case of hyperbolas is the *canonical hyperbola*. Two constants a, b can define two canonical hyperbolas as follows

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = \pm 1, \tag{3.5}$$

which are said to be conjugated. The first one is *east-west* opened, and the second one is *north-south* opened. Interchanging the roles of a and b yields a $\frac{\pi}{2}$-rotated pair of hyperbolas; see Figure 3.1 for an example. For $a = b$, the corresponding hyperbolas are called *rectangular*. Finally, taking $a = b = \sqrt{2}$ and rotating the hyperbolas by 45° yields two so-called *simple hyperbolas*, which are the zero set of $xy = \pm 1$. Note that every generic hyperbola can be transformed into a simple one using an affine transformation.

We shall now study in details how to transform a generic hyperbola $\mathfrak{h}$ to a canonical one. Let $\mathfrak{h}$ be the hyperbola which is the zero set of the polynomial given in Equation (3.2), such that its coefficients satisfy the conditions given in (3.3). First, we apply a translation transformation $\mathscr{T}_1 : \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$\mathscr{T}_1(x, y) \mapsto \begin{pmatrix} x + x_c \\ y + y_c \end{pmatrix}. \tag{3.6}$$

We let $\mathfrak{h}' = \mathscr{T}_1(\mathfrak{h})$ be the image of $\mathfrak{h}$ under the translation $\mathscr{T}_1$. The resulting hyperbola $\mathfrak{h}'$ is congruent to $\mathfrak{h}$ and centered at the origin.

It is left to rotate $\mathfrak{h}'$ so its symmetry axes will align with the x and y-axes. Let Φ be an angle between the positive x axis and either of the axes of symmetry of the hyperbola $\mathfrak{h}'$. A rotation of $\mathfrak{h}'$ about the origin in an angle Φ yields a canonical hyperbola. Note that the resulting hyperbola might be either east-west or north-south opened depending on the choosing of Φ and the orientation of the rotation.

We set $\mathscr{T}_2 : \mathbb{R}^2 \mapsto \mathbb{R}^2$ to be this rotation, defined as follows:

$$\mathscr{T}_2\left(x, y\right) \mapsto \begin{pmatrix} \cos\Phi & -\sin\Phi \\ \sin\Phi & \cos\Phi \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}. \tag{3.7}$$

Then, $\mathfrak{h}'' = \mathscr{T}_2(\mathfrak{h}')$ is a canonical hyperbola that is congruent to $\mathfrak{h}$. Next we find expressions for $\cos\Phi$ and $\sin\Phi$. Note that [22]

$$\tan 2\Phi = \frac{2a_{12}}{a_{11} - a_{22}} = \frac{2\tan\Phi}{1 - \tan^2\Phi}. \tag{3.8}$$

Thus, we have that

$$\tan\Phi = \frac{a_{22} - a_{11} \pm \sqrt{(a_{11} - a_{22})^2 + 4a_{12}^2}}{2a_{12}}.$$

If $a_{12} = 0$, then we have that $\mathfrak{h}$ is aligned with the coordinate axes in the first place, and thus $\mathscr{T}_2$ is the identity transformation. Thus, we can safely assume that $a_{12} \neq 0$. We can now choose one value for $\tan\Phi$

$$\tan\Phi = \frac{a_{22} - a_{11} + \sqrt{(a_{11} - a_{22})^2 + 4a_{12}^2}}{2a_{12}}. \tag{3.9}$$

Choosing the other possible value for $\tan\Phi$ would yield a 90° rotated hyperbola. Since $\Phi \notin \{0, \pi\}$, we can, using elementary trigonometry, obtain the following expressions for $\sin\Phi$ and $\cos\Phi$:

$$\sin\Phi = \frac{\tan\Phi}{\sqrt{1 + \tan^2\Phi}} \quad , \quad \cos\Phi = \frac{1}{\sqrt{1 + \tan^2\Phi}},$$

that can be explicitly expressed as follow:

$$\begin{aligned} \sin\Phi &= \frac{a_{22} - a_{11} + \sqrt{4a_{12}^2 + (a_{11} - a_{22})^2}}{\sqrt{4a_{12}^2 + \left(a_{22} - a_{11} + \sqrt{4a_{12}^2 + (a_{11} - a_{22})^2}\right)^2}} \\ \cos\Phi &= \frac{2a_{12}}{\sqrt{4a_{12}^2 + \left(a_{22} - a_{11} + \sqrt{4a_{12}^2 + (a_{11} - a_{22})^2}\right)^2}} \end{aligned} \tag{3.10}$$

Figure 3.2 illustrates the various steps and the following lemma summarizes the discussion.

Lemma 3.1. *Let $\mathfrak{h}$ be an arbitrary, non-degenerated, hyperbola as in Equation* (3.2) *where its coefficients satisfy the conditions expressed in* (3.3)*. In addition, let $\left(x_c, y_c\right)$ be the coordinates of the center of $\mathfrak{h}$ (cf. Equation* (3.4)*). Furthermore, let $\mathscr{T}_1$ be a translation as defined in Equation* (3.6) *and $\mathscr{T}_2$ be a rotation as given in Equation* (3.7) *were* $\sin\Phi$ *and* $\cos\Phi$ *are as given in Equation* (3.10)*. Then*

$$\tilde{\mathfrak{h}} = \mathscr{T}_2\left(\mathscr{T}_1\left(\mathfrak{h}\right)\right)$$

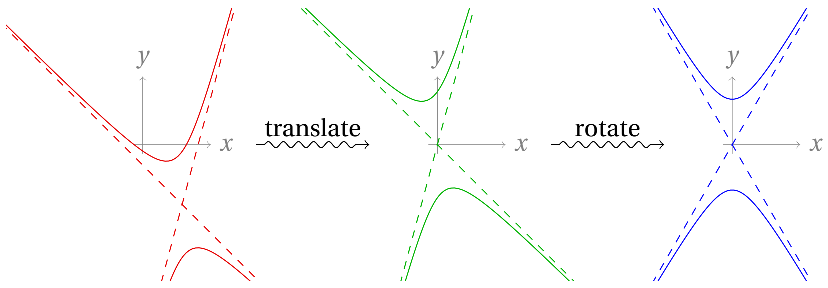


Figure 3.2: On the left a generic hyperbola $\mathfrak{h}$. At the middle the same hyperbola centered at the origin, namely, $\mathscr{T}_1(\mathfrak{h})$. Finally, on the right, a canonical hyperbola $\tilde{\mathfrak{h}}$ that is congruent to $\mathfrak{h}$.

is a canonical *hyperbola given by the zero set of the bivariate polynomial*

$$\tilde{F}(x,y) = ax^2 + by^2 - 1,$$

such that $a \cdot b < 0$.

Proof. Direct computation yields that $\mathscr{T}_2(\mathscr{T}_2(\mathfrak{h}))$ is a canonical hyperbola, which is the zero set of the following bivariate polynomial

$$\bar{F}(x,y) = \bar{a}x^2 + \bar{b}y^2 + \bar{c}$$

with $\bar{c} = \frac{A}{D} \neq 0$. Thus we can obtain the representation in the statement of the lemma. □

3.3 Remarks on Quadratic Surfaces

As we saw in Section 3.1, a key ingredient in the local approximation of smooth surfaces is the study of *quadratic surfaces*. In particular we are interested in the quadratic surfaces $z = F(x,y)$ where $F(x,y)$ is a bivariate polynomial of degree two as given in Equation (3.1). The quadratic surface $z = F(x,y)$ can be brought into one of four *normal forms*, by a linear transformation of the (x,y)-plane and a scaling of the z-axis:

$$z = 0 \tag{3.11}$$
$$z = x^2 \tag{3.12}$$
$$z = x^2 + y^2 \tag{3.13}$$
$$z = x^2 - y^2 \tag{3.14}$$

These transformations do not change the approximation problem. We ignore the cases with zero Gaussian curvature: case (3.12), the parabolic cylinder, and case (3.11). The convex case (3.13) is well-known [34]: optimal triangulations are formed by regular triangles. Since these can be rotated, it comes as no surprise that there is a one-parameter family of optimal triangles. We concentrate in this work on the non-convex case of a *saddle surface*, case (3.14), namely, where the Gaussian curvature is negative.

The most general quadratic surface is $z = F(x, y)$ where $F(x, y)$ is a bivariate polynomial as in Equation (3.2). If, in addition, the coefficients of the polynomial satisfy the conditions given in (3.3) then the resulting surface is a *general* saddle, which is also called *hyperbolic paraboloid*; we use both names interchangeably. If $a_{11}, a_{22} \neq 0$ and otherwise $a_{ij} = 0$, then we call the surface a *canonical saddle* and it is given explicitly as the set

$$\{(x, y, z) : z = ax^2 + by^2 \text{ and } ab < 0\}.$$

A canonical saddle for which $a_{11} = a_{22}$ is called *rectangular saddle*. Finally, if $a_{12} = 1/2$ and otherwise $a_{ij} = 0$ we call the surface a *simple saddle* and it is given by

$$\{(x, y, z) : z = xy\}.$$

Note that a 45° rotation of a simple saddle around the origin yields a canonical saddle.

For each fixed z_0, we obtain a hyperbola by intersecting the saddle surface with a plane at height z_0. The centers of all hyperbolas, which are generated in this way, lie on a common vertical line. The intersection point of the locus of the centers and the surface S is called the *center* of the hyperbolic paraboloid and its coordinates are (x_c, y_c, h) where x_c and y_c are the centers of the various hyperbolas (cf. Equation (3.4)). In addition the *height* of the surface S is $h = F(x_c, y_c) = \frac{A}{D}$ where A and D are as given in (3.3).

Remark 3.1 (Hyperbolas and their conjugates and asymptotes). Note that the hyperbola that is the intersection of a saddle S and the plane at $h + z_0$ is conjugate to the one obtained by the intersection of S and the plane at height $h - z_0$. In turn, for an arbitrary hyperbola, which is the zero set of $F(x, y) = 0$, it is easy to see that its conjugate is given by the zero set $F(x, y) - 2\frac{A}{D} = 0$. Finally, the intersection curve of the general saddle and a horizontal plane at height $z = h$ yields a *degenerated* hyperbola; namely, straight lines, which are the asymptotes of the hyperbolas that correspond to the saddle. Figure 3.3 illustrates notions mentioned in this section and remark.

3.4 Vertical Distance

The *vertical distance* that was briefly introduced in the introduction and is to be discussed in this section, is nothing but the L_∞ norm. However, in this work we emphasize the geometrical nature of the approximation problem and its solution, thus we favor the notion “vertical distance”, which is more intuitive. Although,

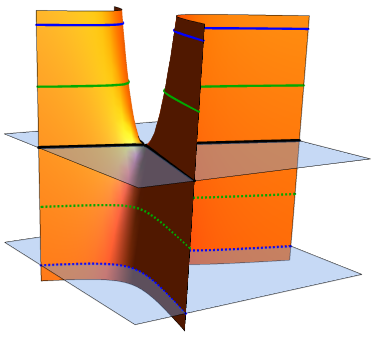


Figure 3.3: A generic saddle surface. The upper plane is passing through the surface's center, and the center is the intersection of the black lines, which are the asymptotes of all hyperbolas associated with the surface. The solid green and blue curves are two hyperbolas and the corresponding dashed curves are their conjugates.

in general, the Hausdorff distance is more suitable for evaluating the quality of approximations we consider the vertical distance due to its nice properties in the context of our discussion. Furthermore, the vertical distance provides (locally) an upper bound on the Hausdorff distance. The definition of the vertical distance is designed to measure the maximal vertical distance between a graph of a function $f\colon \mathbb{R}^2 \to \mathbb{R}$ and various geometric objects.

Definition 3.2 (Vertical Distance)**.** Let $D_1, D_2 \subset \mathbb{R}^2$ be two intersecting domains and let $f\colon D_1 \to \mathbb{R}$ and $g\colon D_2 \to \mathbb{R}$ be two functions. The *vertical distance* between f and g is

$$\mathrm{dist}_V\left(f,g\right) = \sup_{(x,y)\in D_1\cap D_2} |f(x,y) - g(x,y)|.$$

More generally, for two sets $S_1, S_2 \subset \mathbb{R}^3$ we let $\mathrm{dist}_V\left(S_1,S_2\right)$ be the supremum of the vertical distance between $s_1 \in S_1$ and $s_2 \in S_2$ such that s_1 and s_2 have the same (x,y)-coordinates. In practice we consider a function f, which is defined over the whole plane $\mathbb{R}^2$, and g can be either a point, line or line segment, piecewise linear surface patch, another function over $\mathbb{R}^2$ etc. In the context at hand, we let f be a bivariate polynomial, and g is either a line segment connecting two points in space or a piecewise linear surface. It follows naturally, that given two points $p, q \in \mathbb{R}^3$, which lie on the same vertical line, we have

$$\mathrm{dist}_V\left(p,q\right) = |p^z - q^z|,$$

where p^z and q^z are the z-coordinates of p and q, respectively.[1]

Given a general quadratic surface S, which is the graph of a bivariate polynomial of degree at most two, we like to compute the vertical distance between a line

[1]Similarly, p^x and p^y are used to denote the x and y coordinates of p, respectively.

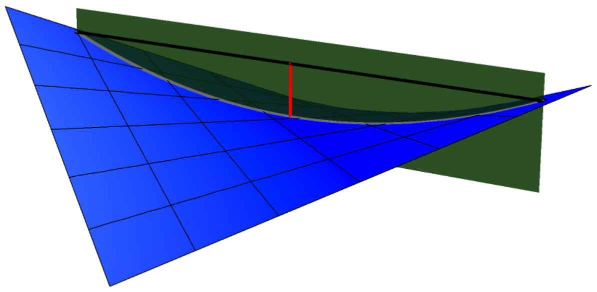


Figure 3.4: The red segment realizes the vertical distance between a line segment that interpolates points on a simple saddle and the saddle itself. The intersection curve between the vertical plane and the saddle is a parabola.

segment connecting two points $p, q \in S$ and the surface itself. Since $\left(x, y, F(x, y)\right) \in S$ for every $\left(x, y\right) \in \mathbb{R}^2$, we have that $\text{dist}_V\left(p, S\right)$ is finite for all p in space, and in particular it is equal to the (Euclidean) distance from p to its vertical projection on S.

Lemma 3.3. *Let S be a quadratic surface $S = \left\{\left(x, y, z\right) : z = F(x, y)\right\}$, where $F(x, y)$ is a bivariate polynomial as in Equation* (3.2)*. Furthermore, let p, q be two points on S and let ℓ_{pq} be the line segment connecting them. Then*

$$\text{dist}_V\left(\ell_{pq}, S\right) = \frac{1}{4}\left|a_{11}\Delta_x^2 + 2a_{12}\Delta_x\Delta_y + a_{22}\Delta_y^2\right|,$$

where $\Delta_x = p^x - q^x$ and $\Delta_y = p^y - q^y$. Furthermore, the vertical distance between ℓ_{pq} and S is attained at the mid-point of the line segment.

Proof. Let $\ell(t) = (1-t)p + tq$ be a parameterization of ℓ_{pq}, then for all $t \in [0, 1]$ we have

$$\begin{aligned}\text{dist}_V\left(\ell(t), S\right) &= |\ell(t)^z - F(\ell(t)^x, \ell(t)^y)| \\ &= |t(1-t)| \cdot \left|a_{11}(\Delta_x)^2 + 2a_{12}(\Delta_x)(\Delta_y) + a_{22}(\Delta_y)^2\right|\end{aligned}$$

Given $t \in [0, 1]$, we have that this functional attains its maximum at $t = \frac{1}{2}$, and this proves the lemma. Figure 3.4 illustrates the case discussed in this lemma. □

The last lemma shows that the vertical distance between a line segment interpolating two points on a quadratic surface and the surface itself depends only on relative position of the vertical projections of the points to the plane. In addition, the quadratic surfaces have a more general property with respect to vertical distance, which is stated in the following lemma.

Lemma 3.4. *For every point $p \in S$, where $S = \left\{\left(x, y, z\right) : z = F(x, y)\right\}$, there exists an affine transformation $\mathcal{T}_p : \mathbb{R}^3 \to \mathbb{R}^3$ given by*

$$\mathcal{T}_p(x, y, z) \mapsto \begin{pmatrix} x + u \\ y + v \\ z + ax + by + c \end{pmatrix},$$

which satisfies the following. First, it maps the point p to the origin. In addition, it maps S to a quadratic surface $\tilde{S}$, which is the graph of a polynomial of the form $\tilde{F}(x,y) = a_{11}x^2 + 2a_{12}xy + a_{22}y^2$. Finally, for any pair of points $q, r \in \mathbb{R}^3$, which lie on a vertical line, we have

$$\operatorname{dist}_V\left(q,r\right) = \operatorname{dist}_V\left(\mathscr{T}_p(q), \mathscr{T}_p(r)\right).$$

Proof. Let $p = \left(\xi, \eta, F(\xi,\eta)\right) \in S$ be a point on the given general saddle. Next, set the parameters u, v, a, b and c in the desired transformation $\mathscr{T}_p$, as follows:

$$\begin{aligned} u &= -\xi & a &= -2(a_{13} + a_{11}\xi + a_{12}\eta) \\ v &= -\eta & b &= -2(a_{23} + a_{12}\xi + a_{22}\eta) \\ & & c &= a_{11}\xi^2 + a_{22}\eta^2 + 2a_{12}\xi\eta - a_{33} \end{aligned}$$

where a_{ij} are the coefficients of the polynomial $F(x,y)$. We can now verify directly that $\mathscr{T}_p(p) = (0,0,0)$. Furthermore, a general point $q = \left(x, y, F(x,y)\right) \in S$ is mapped to

$$\mathscr{T}_p(q) = \begin{pmatrix} x-\xi \\ y-\eta \\ a_{11}(x-\xi)^2 + 2a_{12}(x-\xi)(y-\eta) + a_{22}(y-\eta)^2 \end{pmatrix}.$$

This shows that $\mathscr{T}_p$ maps S to a quadratic surface $\tilde{S}$, which is given by the polynomial $\tilde{F}(x,y) = a_{11}x^2 + 2a_{12}xy + a_{22}y^2$. Note that the tangent plane $T_{\mathscr{T}_p(p)}\tilde{S}$ is horizontal. Finally, the vertical distance is invariant under $\mathscr{T}_p$ for points in $\mathbb{R}^3$ that lie on the same vertical line. □

In other words, this lemma formalizes explicitly the vague statement from the introduction, that all points of a quadratic surface "look equal". Furthermore, this last lemma assures that we can consider, without loss of generality, the canonical saddle surfaces with center at the origin when we investigate the vertical distance. Finally, in Figure 3.5 the effect of the map $\mathscr{T}$ is visualized; a given (red) arbitrary saddle along with a (blue) point on it is mapped to a (green) saddle such that the given point is mapped to the origin, and its tangent plane is horizontal. We conclude this section and establish the relation between the vertical distance and the *Hausdorff distance.*

Lemma 3.5 (Bounding the Hausdorff distance)**.** *Let $A, B \subset \mathbb{R}^3$ be two sets such that their projection to the plane is identical. Then the following holds*

$$\operatorname{dist}_H(A,B) \leq \operatorname{dist}_V(A,B)$$

where $\operatorname{dist}_H(A,B)$ *is the* Hausdorff distance *between the two sets.*

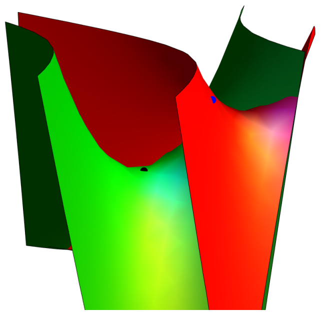


Figure 3.5: In red, a general quadratic surface S is given and an arbitrary point $p \in S$ is marked in blue. In green the transformed surface $\tilde{S}$ (cf. Lemma 3.4) is plotted, with a black point at the origin, which is the image of p.

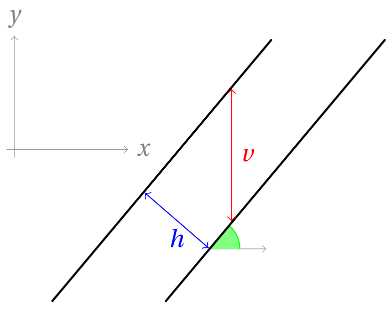


Figure 3.6: The vertical and Hausdorff distances between two parallel lines illustrate that the former can be arbitrarily large, depending on the slope, while the latter is fixed.

Proof. Recall that $\operatorname{dist}_H(A,B) = \max\{\operatorname{dist}(A,B), \operatorname{dist}(B,A)\}$ where $\operatorname{dist}(A,B)$ is the distance from the set A to the set B, that is

$$\operatorname{dist}(A,B) = \sup_{a \in A} \operatorname{dist}(a,B).$$

But, for all $a \in A$ we have $\operatorname{dist}(a,B) \le \operatorname{dist}(a, \pi_B(a))$ where $\pi_B(a)$ is the vertical projection of a on B. In turn, $\operatorname{dist}(a,\pi_B(a)) \le \operatorname{dist}_V(A,B)$ for all $a \in A$, which means that $\operatorname{dist}(A,B) \le \operatorname{dist}_V(A,B)$. Similar argument holds also for the distance $\operatorname{dist}(B,A)$ from B to A. $\square$

Note that this bound can be arbitrarily bad. For example, consider two parallel planes with distance $\varepsilon > 0$ between them. In this case the Hausdorff distance is fixed and the vertical distance can be arbitrarily big depending on the normals of the planes. In Figure 3.6 this idea is illustrated in a two-dimensional example. Nevertheless, at least locally, the vertical distance can still serve as a criteria to measure the quality of a local approximation.

3.5 Local Interpolating Triangulation of Saddle Surfaces

In this section we develop an optimal local approximation of saddle surfaces with respect to the vertical distance. The approximation derived here is similar to the one obtained by Pottmann et al. [34]; the approach we take, however, is more geometrical and intuitive. Here, the meaning of *optimality* is twofold. On the one hand, given a tolerance constant $\varepsilon > 0$, we assert that the piecewise linear patch $\mathscr{T}$, which approximates the neighborhood of a point $p \in S$, maintains $\mathrm{dist}_V(S,T) \le \varepsilon$ for all $T \in \mathscr{T}$. On the other hand, each triangle in $\mathscr{T}$ is maximal; that is, if $T \in \mathscr{T}$ is bigger, then the vertical distance will increase and if it is smaller the distance could decrease. We will first, in Section 3.5.1, provide an optimal triangulation of the simple saddle. Then, in Section 3.5.2, we generalize the result and provide an optimal triangulation of a canonical saddle, which can be used to obtain an approximation of a general saddle as well.

3.5.1 Simple Saddle

In this section we want to obtain a triangulation of a neighborhood of an arbitrary point located on a simple saddle surface

$$S = \left\{(x,y,z) \in \mathbb{R}^3 : z = xy\right\}. \tag{3.15}$$

The obtained triangulation should be topologically correct, that is homeomorphic to a disc. Furthermore, we want to assure that the *maximal* vertical distance between the triangulation and the surface will be fixed.

Vertical Distance to a Triangle. By Lemma 3.3 we have that the maximal vertical distance between the line segment ℓ_{pq} connecting two points $p,q \in S$ and the surface S itself is given by

$$\mathrm{dist}_V\left(S,\ell_{pq}\right) = \frac{1}{4}\cdot\left|v^x v^y\right|,$$

where $\vec{v} = (v^x, v^y, v^z) = p - q$ is the vector from q to p. Recall that the simple saddle is a ruled surface, and in turn if p and q lie on a ruling, then the vertical distance from the line connecting them to S vanishes. Finally, recall, the maximal vertical distance is attained at the midpoint of ℓ_{pq}.

Let us now consider the case of a triangle with vertices $p_0, p_1, p_2 \in S$, denoted by $T = \triangle p_0 p_1 p_2$, for which we want to compute the vertical distance to S, denoted by $\mathrm{dist}_V(S,T)$.

Lemma 3.6. *Given three points $p_0, p_1, p_2 \in S$, where S is a simple saddle surface, then $\mathrm{dist}_V(S,T)$ is attained for a point $p \in T$ which is a midpoint of one of the triangle's edges.*

Proof. Let p be an interior point of T and $p' = \pi(p) \in S$ be it vertical projection on S. Since S is a saddle surface it bends at p' both towards the triangle T and away from it. Therefore there exists a point $p'' \in S$ in a neighborhood of p' such that

$$\| \pi^{-1}(p'') - p'' \| > \| p - \pi(p) \|,$$

where $\pi^{-1}(p'')$ is the vertical projection of $p'' \in S$ back to the triangle T. Therefore, the point $p \in T$ for which the vertical distance is attained cannot be an interior point of T. Thus, it has to lie on an edge of T, and in turn (cf. Lemma 3.3) it has to be a midpoint of one of the edges of T. □

Together, Lemmas 3.3 and 3.6 establish the following lemma.

Lemma 3.7. *Let $S = \{(x, y, z) \in \mathbb{R}^3 : z = xy\}$ be a simple saddle. If a triangle T with three vertices $p_0, p_1, p_2 \in S$ is given, then the vertical distance* $\mathrm{dist}_V(S, T)$ *is given by*

$$\mathrm{dist}_V(S, T) = \max_{i \in \mathbb{Z}_3} \frac{1}{4} \left| e_i^x e_i^y \right|,$$

where $\vec{e}_i = p_{i+1 \bmod 3} - p_i$ for $i \in \mathbb{Z}_3$.

This last lemma can be used to characterize the triangles with vertices on a simple saddle that maintain a prescribed vertical distance. In particular, given a point $p_0 \in S$ we use this lemma to find two more points $p_1, p_2 \in S$ such that the triangle $T = \triangle p_0 p_1 p_2$ will satisfy

$$\mathrm{dist}_V(S, T) \le \varepsilon$$

for some given $\varepsilon > 0$. Let us denote the triangle's edge vectors by $\vec{e}_i = p_{i+1 \bmod 3} - p_i$ (cf. Figure 3.7). For the distance bound to hold, as we saw in Lemma 3.7, the edge vectors have to satisfy

$$|e_i^x e_i^y| \le 4\varepsilon \tag{3.16}$$

for all $i \in \{0, 1, 2\}$, where $\vec{e}_i = \left(e_i^x, e_i^y, e_i^z\right)$. The geometrical interpretation of this last condition is that the heads of the projections of the edge vectors to the plane should lie *inside* the region

$$\mathscr{R} = \left\{(x, y) \in \mathbb{R}^2 : |xy| \le 4\varepsilon\right\},$$

or on its boundary. The region $\mathscr{R}$ is the one bounded by the solid hyperbolas in Figure 3.8. This means that in order to bound the vertical distance between a triangle with vertices lying on a simple saddle surface and the surface itself, we have to consider the planar problem of arranging three edge vectors, which realize a triangle in the plane. The heads of the vectors should be contained in a region bounded by a hyperbola and its conjugate. Since we are interested in the arrangement of the edge vectors we can assume that p_0 is at the origin. This can also be seen by recalling Lemma 3.4.

Let us now consider the planar problem that we discussed. We say that a planar triangle T with vertices p_0, p_1, p_2 is a *valid triangle* if its edge vectors satisfy the

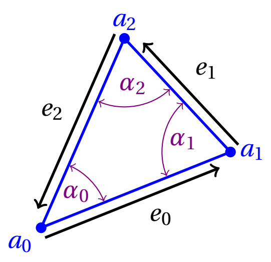


Figure 3.7: Triangle with its edge vectors.

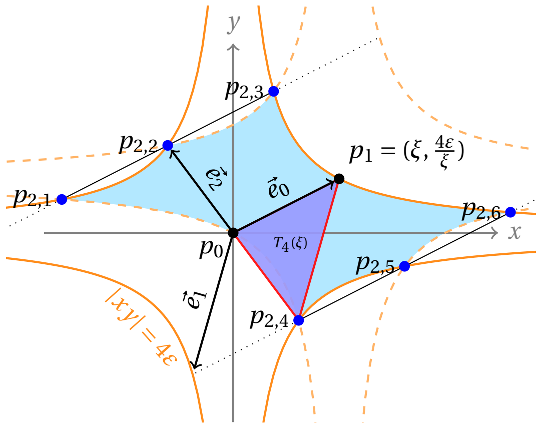


Figure 3.8: The region where the third vertex p_2 of a valid triangle can be placed is shaded in blue.

inequality in (3.16). Since we are only interested in the properties of the edge vectors, we have that valid triangles are invariant under translations. Thus, we can assume, without loss of generality, that p_0 is at the origin. For a planar triangle T with p_0 at the origin, we have that its edge vectors are given by:

$$\begin{aligned}
\vec{e}_0 &= \left({p_1}^x, {p_1}^y\right) \\
\vec{e}_1 &= \left({p_2}^x - {p_1}^x, {p_2}^y - {p_1}^y\right) \\
\vec{e}_2 &= -\left({p_2}^x, {p_2}^y\right)
\end{aligned}$$

Thus, T is a valid triangle if the following system of inequalities is satisfied:

$$\begin{cases}
|p_1^x p_1^y| \le 4\varepsilon \\
|({p_2}^x - {p_1}^x)({p_2}^y - {p_1}^y)| \le 4\varepsilon \\
|{p_2}^x {p_2}^y| \le 4\varepsilon
\end{cases}$$

Clearly, this set of conditions is too vague and we have to impose further conditions in order to narrow down the family of valid triangles.

Maximize the Area of a Valid Triangle. Recall that ultimately we want to lift the valid triangles back to the simple saddle in order to obtain an optimal local triangulation. As we discussed, triangles of largest area are desired. Therefore, the first step is to optimize the area of the valid triangles. Let T be a valid triangle with p_0 at the origin. As we want to maximize the area of T, we can assume that at least one of its edge vectors satisfies Equation (3.16) as an equality, because otherwise the triangle can be scaled up and in turn increase its ares. Without loss of generality we assume that this edge is the edge $\vec{e}_0$, or equivalently, $|{p_1}^x {p_1}^y| = 4\varepsilon$. Let us set

$$p_1 = \left(\xi, \frac{4\varepsilon}{\xi}\right), \quad \xi > 0 \tag{3.17}$$

Symmetrically, as the branches of the hyperbola are symmetric with respect to the axes, we could also set p_1 to be either $\left(-\xi, -\frac{4\varepsilon}{\xi}\right)$ or $\left(\pm\xi, \mp\frac{4\varepsilon}{\xi}\right)$. As long as the case under consideration is the one of the simple saddle, all these possibilities are (almost) the same. However, once the hyperbolas are not symmetric we should pay attention to the way we choose the branch on which p_1 is to be located. See Remark 3.2 on page 60 for further details and what is meant by "almost".

We now have to find where to put p_2 in order to maximize the area. The feasible region where p_2 can be located is given by the following two inequalities

$$\begin{cases} |{p_2}^x {p_2}^y| \le 4\varepsilon \\ |({p_2}^x - {p_1}^x)({p_2}^y - {p_1}^y)| \le 4\varepsilon \end{cases}$$

This is the region bounded by the intersection of a hyperbola, its translated copy and their conjugates, as seen in Figure 3.8. The area of the triangle T is proportional to the distance between p_2 and the line through p_0 and p_1. This distance can only be maximized if p_2 is located on an intersection of two hyperbolas; otherwise, if p_2 lies in the interior of the intersection region or on the boundary of a single hyperbola, it cannot be optimal, since the region is concave. Finally, ${p_2}^x$ and ${p_2}^y$ should therefore satisfy the following system of equations

$$\begin{cases} |{p_2}^x {p_2}^y| = 4\varepsilon \\ |({p_2}^x - \xi)({p_2}^y - \frac{4\varepsilon}{\xi})| = 4\varepsilon \end{cases}$$

Making a case distinction for the absolute value leads to four different systems of two quadratic equations. Since the quadratic terms match, each system has two solutions, which leads to a total of eight solutions. Two solutions are imaginary, and there are six real solutions:

$$\begin{aligned} p_{2,1} &= \left(-\varphi\xi, \frac{1}{\varphi}\frac{4\varepsilon}{\xi}\right) & p_{2,2} &= \left(-\frac{1}{\varphi}\xi, \varphi\frac{4\varepsilon}{\xi}\right) & p_{2,3} &= \left(\frac{1}{\varphi^2}\xi, \varphi^2\frac{4\varepsilon}{\xi}\right) \\ p_{2,4} &= \left(\frac{1}{\varphi}\xi, -\varphi\frac{4\varepsilon}{\xi}\right) & p_{2,5} &= \left(\varphi\xi, -\frac{1}{\varphi}\frac{4\varepsilon}{\xi}\right) & p_{2,6} &= \left(\varphi^2\xi, \frac{1}{\varphi^2}\frac{4\varepsilon}{\xi}\right) \end{aligned} \tag{3.18}$$

where $\varphi = (1+\sqrt{5})/2$ is the *golden ratio*.

For $\varepsilon > 0$ and $\xi \in \mathbb{R}\setminus\{0\}$ this yields six one-parameter families of triangles, which we denote by $T_i(\xi) = \triangle p_0 p_1 p_{2,i}$. See Figure 3.9 for an illustration of some area optimal planar triangles. Note that the triple of planar points $\{p_{2,1}, p_{2,2}, p_{2,3}\}$ is colinear and similarly, the triple $\{p_{2,4}, p_{2,5}, p_{2,6}\}$ is colinear as well. Finally the lines containing the triples above are parallel to $\vec{e}_0$ (cf. Figure 3.8). Therefore, the areas of the triangles $\{T_i(\xi)\}_{i=1}^6$ are all equal to $2\varepsilon\sqrt{5}$ independently of ξ. These triangles are all, by the construction, valid triangles of maximal area, that is, *all* their edge vectors satisfy Equation (3.16) as an equality. For example, $T_4(\xi)$ is illustrated in Figure 3.8 along with all the other elements that we discussed. Before we continue, we note that for all ξ the following congruences hold

$$T_1(\xi) \cong T_6(\xi),\ T_2(\xi) \cong T_5(\xi),\ T_3(\xi) \cong T_4(\xi). \tag{3.19}$$

This can be shown by computing the lengths of the triangles' edges.

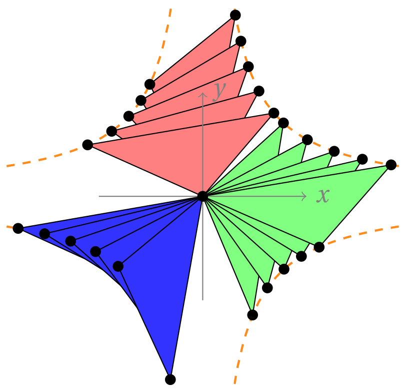


Figure 3.9: In red, green and blue samples of the optimal planar trinagles T_1, T_4 and T_3 for several values of ξ, respectively. All have one vertex at the origin and the other two located on the hyperbola $|xy| = 4\varepsilon$.

Optimize the Shape of the Valid Triangles. By now, we have six one-parameter families of valid triangles such that all have a maximal area. We can impose yet another restriction on these families that will yield better shaped triangles on the one hand and will narrow the set of candidates triangles further on the other hand. In particular, our next step is to optimize the *quality* of the valid triangles $T_i(\xi)$'s. The literature is rich with quality measures of triangles; see, for example, [31]. We consider in this work one of the simplest measures, namely, the smallest angle. More precisely, our goal is to maximize the smallest angle of the valid triangles $T_i(\xi)$'s. Due to the congruence relation, expressed in Equation (3.19), it is enough to maximize the minimal angles of the first three families. Let us compute the case for $i = 1$ in detail; the other cases can be handled in a similar way.

Let us now consider the triangle $T_1(\xi)$, and find for which value of ξ the minimal angle is maximized, given that its vertices are p_0, p_1 and $p_{2,1}$ as defined previously in Equations (3.17) and (3.18). Let $\alpha_0(\xi), \alpha_1(\xi)$ and $\alpha_2(\xi)$ be the angle functions corresponding to the vertices p_0, p_1 and p_{21} of $T_1(\xi)$, respectively. These functions are plotted in Figure 3.10 and their explicit expressions are given in Appendix 3.A. Let $\xi_0^c, \xi_1^c, \xi_2^c$ and ξ_3^c be the values of ξ for which the three functions intersect. One can easily verify that the minimal angle, namely, the lower envelope of the three functions, is maximized for

$$\xi_0 = \frac{2}{\varphi}\sqrt{\varepsilon},$$

which equals to ξ_2^c and lies between ξ_1^c and ξ_3^c. In Appendix 3.A detailed computations can be found. Note that ξ_0 depends on ε. By symmetry we obtain that $T_1(\pm\xi_0)$ are valid triangles of maximal area and with the minimal angle maximized. By the congruency of the triangles $T_i(\xi)$, we obtain that $T_6(\pm\xi_0)$ are optimal triangles as well.

Similarly, we can easily find that for $\xi_1 = 2\varphi\sqrt{\varepsilon}$ we have that $T_3(\pm\xi_1)$ and $T_4(\pm\xi_1)$

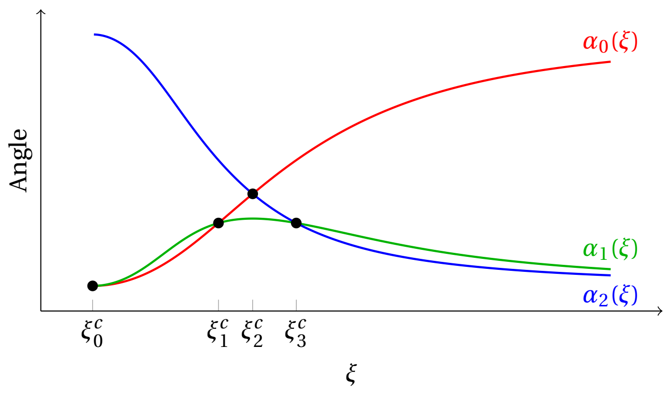


Figure 3.10: Plot of the three angle functions of the triangle $T_1(\xi)$, all depending on ξ.

are the optimal triangles we are looking for, once ε is given.[2] Finally, for every $\varepsilon > 0$, the following six triangles

$$
\begin{aligned}
T_1^{opt} &= T_1\left(\frac{2}{\varphi}\sqrt{\varepsilon}\right) & T_2^{opt} &= T_1\left(-\frac{2}{\varphi}\sqrt{\varepsilon}\right) \\
T_3^{opt} &= T_3\left(2\varphi\sqrt{\varepsilon}\right) & T_4^{opt} &= T_3\left(-2\varphi\sqrt{\varepsilon}\right) \\
T_5^{opt} &= T_4\left(2\varphi\sqrt{\varepsilon}\right) & T_6^{opt} &= T_4\left(-2\varphi\sqrt{\varepsilon}\right)
\end{aligned}
\tag{3.20}
$$

triangulate a neighborhood of the origin in the (x, y)-plane, as can be seen in the lower right part of Figure 3.12. Direct evaluation shows that all these triangles are isosceles with one edge of length $2\sqrt{2\varepsilon}$ and the two other of length $2\sqrt{3\varepsilon}$ each. In addition the minimal angle equals $\arccos\frac{2}{3} = \frac{\pi}{c}$ where $c \approx 3.735$.

If we were to take the conjugate choice of p_1 in Equation (3.17), then the triangles obtained in Equation (3.20) would provide a triangulation of a neighborhood of the origin rotated by 90°. Figure 3.11 illustrates the effect of this alternation. In general, any triangle, together with its negative mirror image, can be used to tile the plane. But, up to translations, there are just two tilings with optimal shape: the one given in Figure 3.12 bottom right, and the one shown in Figure 3.11, which is a reflection at the x-axis of the first one.[3] We will see in Section 3.5.2 that once the hyperbolas (or the corresponding saddles) are not simple, the two possibilities are no longer equally optimal. See Figures 3.25 and 3.26 for an example.

Local Triangulation of Simple Saddle. We are now ready to generate a triangulation of a neighborhoods of an arbitrary point p given on a simple saddle surface S,

[2]Note that similar computation can show that the families $T_2(\pm\xi_2)$ and $T_5(\pm\xi_2)$ attain an optimal shape for $\xi_2 = 2\sqrt{\varepsilon}$. However, as can be seen in Figure 3.12 they do not contribute to the triangulation, and thus we discard them.

[3]Instead of this reflection we could either reflect with respect to the y-axis, or rotation by 90°. Either transformation can be used.

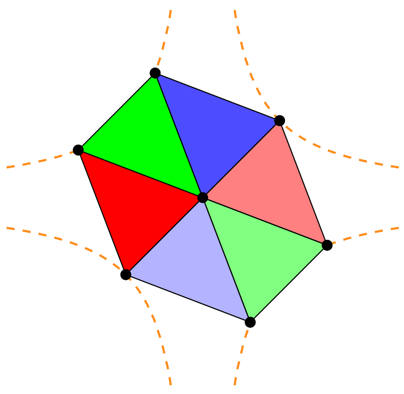


Figure 3.11: Planar optimal triangles of the conjugate case, namely, choosing p_1 initially on a different branch. Compare to the case depicted in the lower right corner of Figure 3.12.

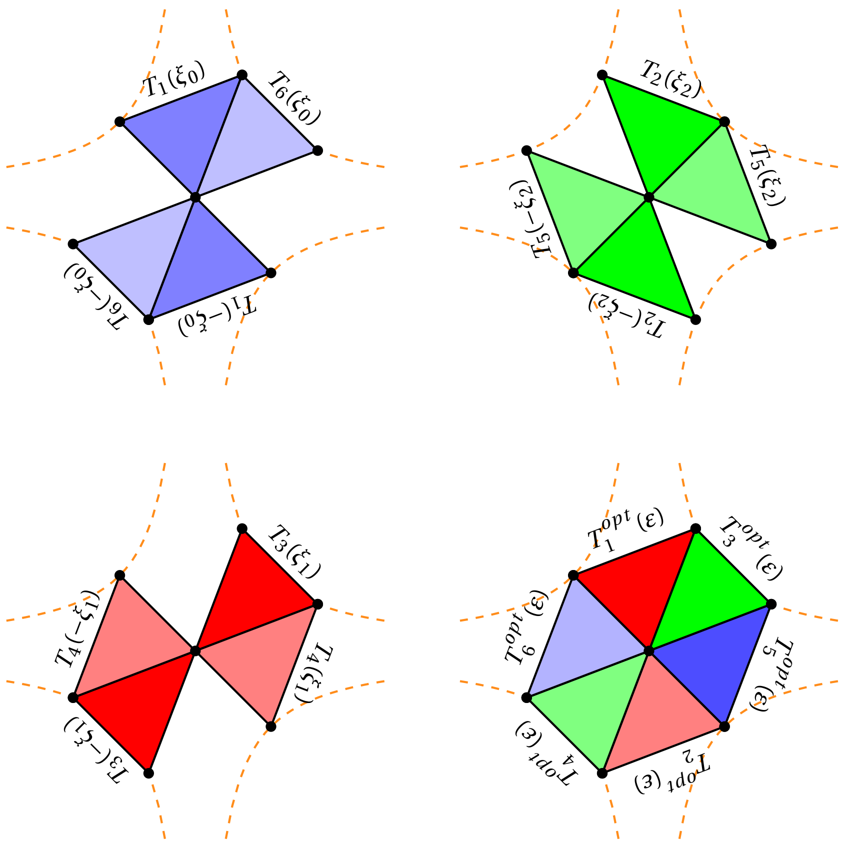


Figure 3.12: The best shaped triangles (up to translations) that satisfy the condition (3.16). On the lower right corner a fundamental domain for tiling the plane is shown. Note that $T_3(\pm\xi_1) = T_6(\pm\xi_0)$

as given in Equation (3.15) on page 53. In particular, we want to obtain a triangulation that is optimal in the following sense. First, and most important, it should meet a prescribed maximal vertical distance. More explicitly, for a given $\varepsilon > 0$, the maximal vertical distance for all the triangles $T \in \mathscr{T}$ to the surface S, where $\mathscr{T}$ is the collection of triangles in the triangulation, has to satisfy

$$\operatorname{dist}_V(S, T) \leq \varepsilon \quad \forall T \in \mathscr{T}. \tag{3.21}$$

In addition, we want the triangulation to be both *efficient* and *nice*. By *efficient triangulation* we refer to a triangulation where *few* triangles cover *large* area, while still maintaining the maximal vertical distance optimality as expressed in Equation (3.21). In addition, a triangulation is *nicely shaped* if the smallest angles of all the triangles that constitute the triangulation are as large as possible while still meeting the optimality of the vertical distance.

As we saw in Lemma 3.7, for a triangle with vertices on S, the vertical distance depends on its projection to the (x, y)-plane. In Equation (3.20) we characterized the planar triangles that satisfy, given some $\varepsilon > 0$, the edge vectors condition and have maximal area and optimal (planar) shape. Furthermore, for a given $\varepsilon > 0$, *all* the edges of the triangles in $\{T_i^{opt}\}_{i=1}^{6}$ satisfy an equality in (3.16). Note that this also hold for any translated copy of the family. We use this family of triangles in order to obtain a local triangulation that will satisfy the desired properties aforementioned.

Notation 3.8 (Lifting a triangle)**.** Given a planar triangle T and a saddle S, we denote by $\hat{T}$ the vertical lifting of T to S. In particular

$$\hat{T} = \operatorname{conv}\{\pi(a_i)\}_{i=0}^{2},$$

where $\pi(a_i)$ is the vertical lifting of the vertex a_i of T to the saddle S.

Lemma 3.9. *Let S be a simple saddle and $\varepsilon > 0$. In addition let $T_i^{opt}(\vec{v})$ be a $\vec{v} \in \mathbb{R}^2$ translation of T_i^{opt}. Then, the collection $\mathscr{T} = \{\hat{T}_i^{opt}(\vec{v})\}$ is an interpolating triangulation of a neighborhood of the point $(\vec{v}^x, \vec{v}^y, \vec{v}^x\vec{v}^y)$ on S, with maximal vertical distance equals ε. That is, for all i, $\operatorname{dist}_V\left(\hat{T}_i^{opt}(\vec{v}), S\right) = \varepsilon$.*

Proof. Direct computation proves that all the edge vectors of all the triangles in $\mathscr{T}$ satisfy

$$|e_i^x e_i^y| = 4\varepsilon,$$

and thus by Lemma 3.7 we have $\operatorname{dist}_V\left(S, \hat{T}_i^{opt}(\vec{v})\right) = \varepsilon$ for all $i \in \{1, 2, \ldots, 6\}$. $\square$

Remark 3.2. In Equation (3.17) we set $p_1 = \left(\xi, \frac{4\varepsilon}{\xi}\right)$. As pointed out already, setting $p_1 = \left(-\xi, -\frac{\varepsilon}{\xi}\right)$ yields the same triangulation. Both cases produce the same triangles, which can be found in Figure 3.12. However, when choosing $p_1 = \left(\pm\xi, \mp\frac{\varepsilon}{\xi}\right)$ we can obtain a triangulation of the conjugate hyperbolas, as can be seen in Figure 3.11. The resulting triangulation of the saddle in this case is a conjugate version as well.

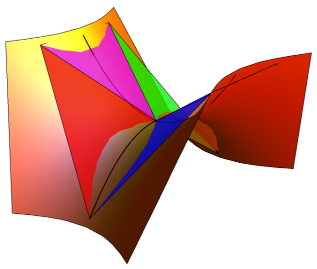


Figure 3.13: An example of a triangulation of a simple saddle surface with optimal shaped triangles of maximal area.

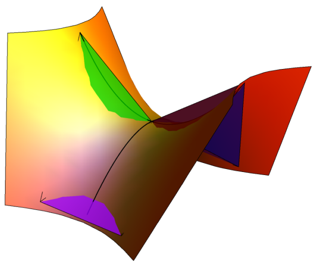


Figure 3.14: A triangulation that is conjugate to the one in Figure 3.13, which results from different choice of p_1.

In Figures 3.13 and 3.14 we can see two triangulations of S, corresponding to the first and second choice of p_1. In the first case each lifted triangle has two edges above S and one edge below. On the other hand, in the conjugate case each triangle has two edges below S and one above. However, when the saddle at hand is not simple, the choosing of p_1 has a bigger role as can be seen in Figures 3.25 and 3.26 where one choice yields larger smallest angles than the other one. We address this subtle issue later in Section 3.5.2.

On the Angles of the Optimal Triangulation. Once the planar triangles are lifted to the saddle surface, not all of them are kept isosceles anymore. In particular, when $(\nu,\mu) = (0,0)$, that is no translation is involved, $\hat{T}_3^{opt}$ and $\hat{T}_4^{opt}$ are both isosceles with minimal angles equal to

$$\arccos\left(\frac{2+4\varepsilon}{3+4\varepsilon}\right).$$

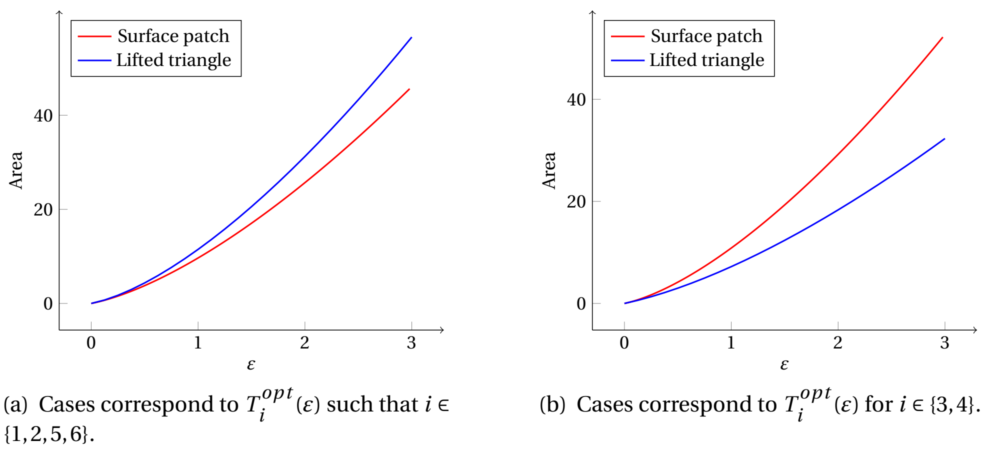
The other four triangles are all congruent with a minimal angle that equals

$$\arccos\left(\frac{2(1+4\varepsilon)}{\sqrt{(3+4\varepsilon)(3+16\varepsilon)}}\right).$$

In both cases, the minimal angles is given by a decreasing function, and thus the smaller ε is the bigger the minimal angle is. Note that as $\varepsilon \to 0$ both smallest angles converge to $\arccos\frac{2}{3} = \frac{\pi}{c}$ for $c \approx 3.735$, which is the same smallest angle value that we obtained in the planar case.

Furthermore, once translation takes place, that is when trying to triangulate a neighborhood of an arbitrary point $(\nu,\mu,\nu\mu) \in S$, then the minimal angle decreases whenever the magnitude of the translation $(\nu,\mu) \in \mathbb{R}^2$ increases.

Covered Area. In this paragraph we consider the area of the approximating triangles and the corresponding surface patches. First, when considering a triangulation



(a) Cases correspond to $T_i^{opt}(\varepsilon)$ such that $i \in \{1,2,5,6\}$.

(b) Cases correspond to $T_i^{opt}(\varepsilon)$ for $i \in \{3,4\}$.

Figure 3.15: The surface area of a patch of the simple saddle (in red) compared to the area of the approximating triangle (in blue) as functions of ε. Horizontal axis represents $\varepsilon \in [0,3]$ and the vertical axes is the area.

of a neighborhood of the origin, namely, the case where no translation is applied, a direct computation of the area of the optimal triangles $\hat{T}_i^{opt}$ yields that

$$\text{area}(\hat{T}_3^{opt}) = \text{area}(\hat{T}_4^{opt}) = 2\varepsilon\sqrt{5+8\varepsilon}, \tag{3.22}$$

and otherwise

$$\text{area}(\hat{T}_i^{opt}) = 2\varepsilon\sqrt{5+28\varepsilon} \tag{3.23}$$

for $i \in \{1,2,5,6\}$. In this case, we can provide a numerical approximation of the area of the simple saddle itself that corresponds to an optimal triangle T_i^{opt}. As we can see in Figure 3.15 in both case the area of the approximating triangle (in green) converges to the are of the corresponding surface patch (in red). An analytic expression of the area of the surface patch is not provided, and thus the convergence rate cannot be analyzed.

On the other hand, we can consider the case where we allow translations of the planar optimal triangles before lifting them to the simple saddle. This case corresponds to the local triangulation of an arbitrary point of S. As before, let $T_i^{opt}(\vec{v})$ be a translation of T_i^{opt} by a vector $\vec{v} \in \mathbb{R}^2$ and let $\hat{T}_i^{opt}(\vec{v})$ be its lifting. In this case, we have that

$$\text{area}(\hat{T}_i^{opt}(\vec{v})) \geq 2\varepsilon\sqrt{5} \quad \forall \vec{v} \in \mathbb{R}^2.$$

In particular, for each $i \in \{1,\dots,6\}$ there exists $\vec{v}_i \in \mathbb{R}^2$ such that $\text{area}(\hat{T}_i^{opt}(\vec{v}_i)) = 2\varepsilon\sqrt{5}$. Recall that this is the area of the planar optimal triangles. Furthermore, the area of $\hat{T}_i^{opt}(\vec{v})$ increases depending on the distance of $\vec{v}$ from $\vec{v}_i$.

Gaussian Curvature. Given a point $p \in S$, then one can directly compute the Gaussian curvature $K(p)$ at p

$$K(p) = -\frac{1}{(1+x^2+y^2)^2}.$$

In addition, let $\mathscr{T}_p$ be the optimal triangulation obtained in Lemma 3.9, then we can compute the *discrete Gaussian curvature* at p. In [33, 40] the (total) Gaussian curvature is defined as follows:

$$K_p = 2\pi - \sum_{T \in \mathscr{T}_p} \alpha_T(p)$$

where $\alpha_T(p)$ is the inner angle of the triangle T at the common vertex p. In our case, if p is at the origin, then we can find an explicit expression of K_p, namely,

$$K_p = 2\pi - 2\arccos\left(\frac{2+4\varepsilon}{3+4\varepsilon}\right) - 4\arccos\left(\frac{\sqrt{2}\varepsilon(1-4\varepsilon)}{\sqrt{\varepsilon(1+2\varepsilon)}\sqrt{\varepsilon(3+4\varepsilon)}}\right).$$

However, it turns out that $\lim_{\varepsilon\to 0} K_p = 0$ and it does not converge to the correct value of the Gaussian curvature of the simple saddle at the origin.

Alternatively, one can consider the definition of a discrete Gaussian curvature as given by [9, 10], namely,

$$K_p = \frac{3}{A(p)}\left(2\pi - \sum_{T \in \mathscr{T}_p} \alpha_T(p)\right),$$

where $A(p)$ is the total area of the triangles in the discrete patch $\mathscr{T}_p$. Using the expressions of the areas of the triangulating triangles (Equations (3.22) and (3.23)) we can obtain a new expression for the discrete Gaussian curvature. This time it is easy to show that for p at the origin $\lim_{\varepsilon\to 0} K_p = -1$, and indeed converges to the Gaussian curvature of the smooth surface.

From Local to Global Triangulation. As we saw, the local planar triangulation we obtained is invariant under translations. Therefore, we can first tesselate the plane with optimal hexagon, similar to those found in Figure 3.12, and then lift the tessellation to the saddle S. For the simple saddle surface case studied in this section, the planar tessellation can be found in Figure 3.16, and the corresponding surface triangulation is plotted in Figure 3.17. All the triangles in this triangulation maintain a fixed maximal vertical distance throughout the domain, but the shape of the triangles themselves degenerates depending on their remoteness from the origin.

3.5.2 Canonical Saddle

Our next task is to generalize the results that we obtained for the simple saddle and generate and optimal approximation of a *canonical saddle*. Without loss of

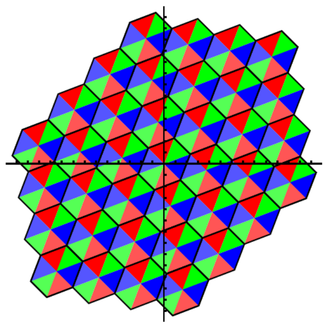


Figure 3.16: A tessellation of the plane with optimal planar triangles, which correspond to a simple saddle surface.

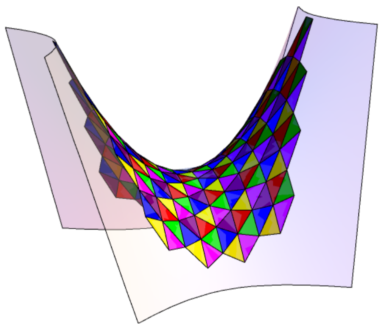


Figure 3.17: The triangles from the tessellation in Figure 3.16 lifted to the simple saddle itself.

generality, we recall (cf. Section 3.3) that a canonical saddle is

$$S = \{(x, y, z) : z = ax^2 - by^2\},$$

such that $a \cdot b > 0$. For the sake of simplicity, we can assume, without loss of generality, that $a = 1$ and $b > 0$. We start the discussion in a similar approach to the one employed in Section 3.5.1. Given a generic planar triangle T our goal is to maximize its area while maintaining the vertical distance constraint, namely,

$$\text{dist}_V(S, \hat{T}) \leq \varepsilon,$$

where $\hat{T}$ is the lifting of T to the canonical saddle S and $\varepsilon > 0$. First, recall from the discussion on the vertical distance (cf. Section 3.4), that it is invariant under translations of T in the plane. Therefore, we can assume that one of the vertices of T is at the origin. Indeed, we denote the vertices of T by

$$p_0 = (0,0) \quad p_1 = (x_1, y_1) \quad p_2 = (x_2, y_2).$$

In addition, like in the simple case, the vertical distance cannot be attained at an interior point of $\hat{T}$. Thus, we can use Lemma 3.3 and we have to solve the following system:

$$\begin{cases} \frac{1}{4}|x_1^2 - by_1^2| = \varepsilon \\ \frac{1}{4}|(x_1 - x_2)^2 - b(y_1 - y_2)^2| = \varepsilon \\ \frac{1}{4}|x_2^2 - by_2^2| = \varepsilon \end{cases} \tag{3.24}$$

From the first equation we have that p_1 can either be located on the hyperbola $\mathfrak{h}$ given by $x^2 - by^2 = 4\varepsilon$, or on its conjugate counterpart, namely, $x^2 - by^2 = -4\varepsilon$, which is denoted by $\mathfrak{h}'$. In Figure 3.18 the former hyperbola is plotted in red and

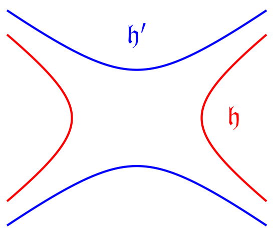


Figure 3.18: An example of the hyperbolas $\mathfrak{h}$ and $\mathfrak{h}'$.

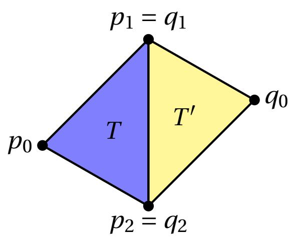


Figure 3.19: A triangle T and its point reflection T' with respect to the midpoint of the segment connecting p_1 and p_2.

the latter in blue. We first treat the former case, namely, assume that p_1 lies on $\mathfrak{h}$. From the first equation in (3.24) we have that

$$x_1 = \pm\sqrt{by_1^2 + 4\varepsilon}$$

for all $y_1 \in \mathbb{R}$. Without loss of generality we can choose p_1 to lie on the right branch of $\mathfrak{h}$, that is $p_1 = (\xi, \eta)$ where $\xi = \sqrt{b\eta^2 + 4\varepsilon}$ and $\eta \in \mathbb{R}$. Given this parameterization of p_1, we can now, using the second and third equations in (3.24), find expressions for x_2 and y_2 as functions of η. Indeed

$$\begin{aligned}
p_{21} &= \left(\frac{-\xi^2 + \sqrt{5b}\xi\eta}{2\xi}, \frac{-b\eta + \sqrt{5b}\xi}{2b}\right)\\
p_{22} &= \left(\frac{3\xi^2 + \sqrt{5b}\xi\eta}{2\xi}, \frac{3b\eta + \sqrt{5b}\xi}{2b}\right)\\
p_{23} &= \left(\frac{\xi^2 + \sqrt{5b}\xi\eta}{2\xi}, \frac{b\eta + \sqrt{5b}\xi}{2b}\right)\\
p_{24} &= \left(\frac{\xi^2 - \sqrt{5b}\xi\eta}{2\xi}, \frac{b\eta - \sqrt{5b}\xi}{2b}\right)\\
p_{25} &= \left(-\frac{\xi^2 + \sqrt{5b}\xi\eta}{2\xi}, -\frac{b\eta + \sqrt{5b}\xi}{2b}\right)\\
p_{26} &= \left(\frac{3\xi^2 - \sqrt{5b}\xi\eta}{2\xi}, \frac{3b\eta - \sqrt{5b}\xi}{2b}\right)
\end{aligned}$$

In other words, similarly to the case of the simple saddle, we obtain six one-parameter families of triangles, which we denote by $T_i(\eta) = \triangle p_0 p_1 p_{2i}$. For all $i \in \{1, \ldots, 6\}$ we have that the area of $T_i(\eta)$ equals $\sqrt{5/b}\,\varepsilon$, independently from η. In Figure 3.20 the triangles $T_i(\eta)$'s are plotted for some fixed value of η. It is easy to verify that if $\hat{T}_i(\eta)$ is the lifting of $T_i(\eta)$ to the canonical saddle S, then we have $\text{dist}_V\left(S, \hat{T}_i(\eta)\right) = \varepsilon$ for all i and for all $\eta \in \mathbb{R}$.

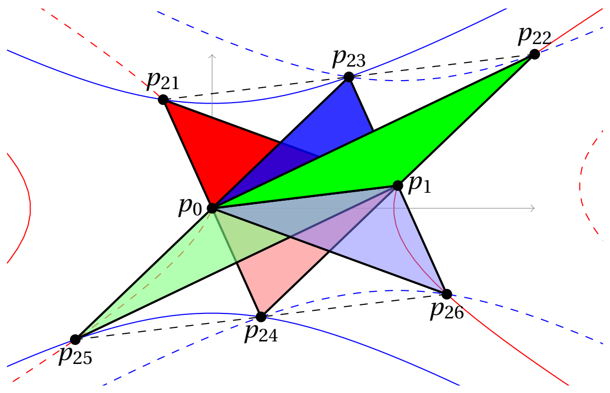


Figure 3.20: Six area maximal triangles obtained for a canonical case $z = x^2 - by^2$. All the triangles are parameterized by the y-value of p_1, which is located on the hyperbola $x^2 - by^2 = 4\varepsilon$. Note that the dashed hyperbolas are centered at p_1.

Before we continue, let us point out that the vertical distance is invariant under point reflections of the corresponding planar triangles. More explicitly, let $\hat{T}$ be an arbitrary triangle with vertices on some saddle and let T be its projection to the xy-plane. If T' is a point reflection of T with respect to the midpoint of one of its edges, then

$$\text{dist}_V\left(S, \hat{T}\right) = \text{dist}_V\left(S, \hat{T}'\right)$$

where $\hat{T}'$ is the lifting of the triangle T' to the same arbitrary saddle. The following lemma proves this.

Lemma 3.10. *Let T and T' be two planar triangles, such that the latter is a point reflection of the former with respect to the midpoint of one of its edges. Then,*

$$\text{dist}_V\left(S, \hat{T}\right) = \text{dist}_V\left(S, \hat{T}'\right),$$

where $\hat{T}$ and $\hat{T}'$ are the liftings of T and T' to some saddle S, respectively.

Proof. Let T be a planar triangle with vertices p_0, p_1 and p_2. Furthermore, let $\hat{T}$ be the lifting of T to the saddle S. Since S is a saddle, we have that the vertical distance $\text{dist}_V\left(S, \hat{T}\right)$ is attained on the boundary of the triangle. Thus, by Lemma 3.3, we have that $\text{dist}_V\left(S, \hat{T}\right) = \max_{i,j} \text{dist}_V\left(S, \ell_{\hat{p}_i\hat{p}_j}\right)$, such that $0 \le i < j \le 2$ and $\hat{p}_i$ and $\hat{p}_j$ are the vertices of $\hat{T}$. Without loss of generality, let T' be the point reflection of T with respect to the midpoint of the edge connecting p_1 and p_2 (cf. Figure 3.19). In particular, the vertices of T' are q_0, q_1 and q_2 such that $q_1 = p_1, q_2 = p_2$ and $q_0 = p_2 + p_1 - p_0$. We have that

$$\begin{aligned}
\text{dist}_V\left(S, \ell_{\hat{p}_0\hat{p}_1}\right) &= \text{dist}_V\left(S, \ell_{\hat{q}_0\hat{q}_2}\right)\\
\text{dist}_V\left(S, \ell_{\hat{p}_1\hat{p}_2}\right) &= \text{dist}_V\left(S, \ell_{\hat{q}_1\hat{q}_2}\right)\\
\text{dist}_V\left(S, \ell_{\hat{p}_0\hat{p}_2}\right) &= \text{dist}_V\left(S, \ell_{\hat{q}_0\hat{q}_1}\right)
\end{aligned}$$

since $p_2 - q_0 = p_0 - p_1$ and $p_1 - q_0 = p_0 - p_2$. Thus the lemma is proved. □

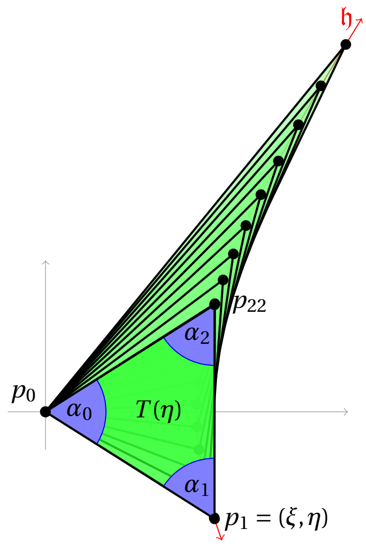


Figure 3.21: The triangle $T_2(\eta)$, which we denote by $T(\eta)$, plotted for various values of η.

We can conclude that the vertical distance is invariant under translations in the xy-plane and point reflections. Therefore, for some fixed η and $i \in \{1,\ldots,6\}$, the triangle $T_i(\eta)$ and its translations and point reflections (with respect to the midpoints of its edges) can be used to tile the plane. Lifting this tiling to the canonical saddle yields an approximation of the surface for which the maximal vertical distance equals ε. We now observe (cf. Figure 3.20) that it is sufficient to consider only one one-parameter family of triangles. We first note that $T_4(\eta)$ is a point reflection of $T_3(\eta)$ with respect to the midpoint of the line connecting p_0 and p_1. Moreover, $T_3(\eta)$ is a point reflection with respect to the midpoint of a triangle that has two vertices on $\mathfrak{h}'$ and is treated later; thus we can, at this point, ignore these two families (cf. Remark 3.4 on page 70). Furthermore, $T_1(\eta)$ and $T_5(\eta)$ are point reflections, with respect to the midpoint of the line segment $\overline{p_0 p_1}$, of $T_6(\eta)$ and $T_2(\eta)$, respectively. In addition, if

$$\eta_2 = \frac{1}{2}\left(3\eta_1 - \sqrt{5}\sqrt{\eta_1^2 + \frac{4\varepsilon}{b}}\right),$$

then $T_2(\eta_2) = T_6(\eta_1)$ for all $\eta_1 \in \mathbb{R}$. Therefore, we can, without loss of generality, focus on $T_2(\eta)$, which we denote by $T(\eta)$. The last step to take, is to optimize the shape of $T(\eta)$, and obtain an optimal approximation of the canonical saddle, similar to the one we obtained in Section 3.5.1. Once again, given two triangles, then we say that the shape of the first is better than the shape of the second if the minimal angle of the first is larger than the minimal angle of the second.

Let $\alpha_0(\eta), \alpha_1(\eta)$ and $\alpha_2(\eta)$ be the functions of the angles of $T(\eta)$ at the vertices p_0, p_1 and p_{22}, respectively. See, for reference, Figure 3.21 where $T(\eta)$ is plotted for several values of η. By their definition, the angle functions are smooth. In the next lemma we establish four properties of the angle function.

Lemma 3.11. *The angle function $\alpha_0(\eta), \alpha_1(\eta)$ and $\alpha_2(\eta)$ of the parameterized triangle $T(\eta)$ satisfy the following properties.*

Property 1: The following limits hold:

$$\lim_{\eta\to\pm\infty}\alpha_0(\eta)=0$$
$$\lim_{\eta\to-\infty}\alpha_1(\eta)=\lim_{\eta\to\infty}\alpha_2(\eta)=0$$
$$\lim_{\eta\to\infty}\alpha_1(\eta)=\lim_{\eta\to-\infty}\alpha_2(\eta)=\pi$$

Property 2: The functions $\alpha_1(\eta)$ and $\alpha_2(\eta)$ are monotonically increasing and decreasing, respectively.

Property 3: $\alpha_0(\eta)$ has a global maximum *for $\eta_0=-\sqrt{\varepsilon/b}$.*

Property 4: Finally, $\alpha_1(\eta_0)=\alpha_2(\eta_0)$.

Proof. The slopes of the edges of $T(\eta)$ as a function of η are given by (see cell `In[30]` in Appendix 3.C):

$$\frac{\eta}{\sqrt{b\eta^2+4\varepsilon}}$$
$$\frac{b^2\eta^3+4b\eta\varepsilon+3\sqrt{5}\varepsilon\sqrt{b\left(b\eta^2+4\varepsilon\right)}}{b\sqrt{b\eta^2+4\varepsilon}\left(b\eta^2+9\varepsilon\right)}$$
$$\frac{b^2\eta^3+4b\eta\varepsilon-\sqrt{5}\varepsilon\sqrt{b\left(b\eta^2+4\varepsilon\right)}}{b\left(b\eta^2-\varepsilon\right)\sqrt{b\eta^2+4\varepsilon}}$$

It is easy to verify that for $\eta\to\infty$ all the limits of all the slopes is $\frac{1}{\sqrt{b}}$, which is also the slope of the asymptote of the hyperbola. Therefore we have that $\lim_{\eta\to\infty}\alpha_1(\eta)=\pi$. In turn, for $\eta\to\infty$ we have that the limit of the other two angle functions is zero. Similarly, it is easy to show the case where $\eta\to-\infty$. This proves the first property.

When η traverses the interval $(-\infty,\infty)$, the edge from p_0 to p_1 turns left and the edge from p_1 to p_{22} turns right when observed from p_1. Therefore, the angle function $\alpha_1(\eta)$ is monotonically increasing. Similar argument shows that $\alpha_2(\eta)$ is monotonically decreasing. This establishes the second property.

Direct computation yields that the derivative of $\alpha_0(\eta)$ has a single critical value $\eta_0=-\sqrt{\frac{\varepsilon}{b}}$. In particular, $\alpha_0'(\eta_0)=0$ and the second derivative satisfies $\alpha_0''(\eta_0)<0$; see cell `In[34]` in Appendix 3.C. Thus, $\alpha_0(\eta)$ has, indeed, a global maximum for η_0. Finally, it is easy to verify the last property. $\square$

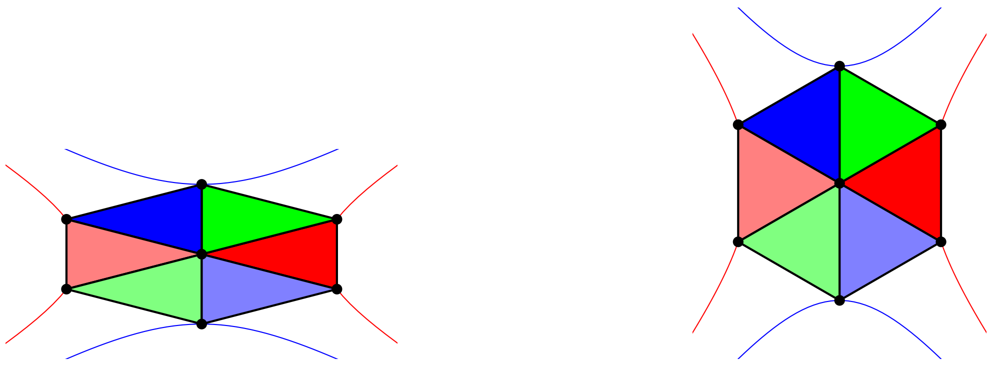


Figure 3.22: Optimal triangulation of the neighborhood of the origin using copies of T from Equation (3.25). Note that each of the red triangles has *two* vertices on $\mathfrak{h}$.

Remark 3.3. The proof of the last lemma can be verified easily using symbolic computations, as we present in Appendix 3.C. However, as proposed by G. Rote, one can obtain a more elegant parameterization of the triangle in question. Using this parameterization the complexity of the computations reduces significantly.

Finally, let $\alpha = \min\{\alpha_0(\eta_0), \alpha_1(\eta_0)\}$ be the minimal angle of $T(\eta_0)$. Given the properties above, for all $\eta \neq \eta_0$ we have that the minimal angle of the corresponding $T(\eta)$ is smaller than α. Therefore we can state the following:

Claim. *The triangle $T = T(\eta_0)$ for $\eta_0 = -\sqrt{\varepsilon/b}$ is of optimal shape among all the triangles in the one-parameter family of triangles $T(\eta)$. In particular*

$$T = \operatorname{conv}\left\{(0,0), \left(\sqrt{5\varepsilon}, -\sqrt{\frac{\varepsilon}{b}}\right), \left(\sqrt{5\varepsilon}, \sqrt{\frac{\varepsilon}{b}}\right)\right\} \tag{3.25}$$

It is interesting to point that for $b = 3/5$ the triangle T is *equilateral*. Finally, we can triangulate the neighborhood of the origin using T and its point reflections (with respect to the midpoints of its edges) and translations; as can be seen in Figure 3.22. In turn, we can lift this local triangulation to the given canonical saddle and obtain a triangulation. Next, we have to treat the case where p_1 is on the conjugate hyperbola, namely, cf. $\mathfrak{h}'$ in Figure 3.18.

We now assume that p_1 is on the upper hyperbola $\mathfrak{h}'$, and thus, from the first equation in (3.24), can without loss of generality be parameterized by its x-value as follow

$$p_1(\xi) = \left(\xi, \sqrt{\frac{\xi^2 + 4\varepsilon}{b}}\right).$$

As before, next we obtain six one-parameter families of triangles $T_i'(\xi)$, $i \in \{1,\ldots,6\}$, such that all have the same area and for which

$$\operatorname{dist}_V\left(S, \hat{T}_i'(\xi)\right) = \varepsilon$$

where $\hat{T}_i'(\xi)$ is the lifting of $T_i'(\xi)$ to the canonical saddle S. Once again, the area of the triangles is $\sqrt{5/b}\,\varepsilon$, like in the previous case. We can now choose, without loss of generality, one of the families. For the sake of convenience, we choose a family for which p_1 and p_2 lie on the hyperbola $\mathfrak{h}'$. In particular, we set $T'(\xi)$ to be the triangle with vertices

$$\left\{ \begin{array}{c} (0,0), \left(\xi, \sqrt{\dfrac{\xi^2+4\varepsilon}{b}}\right), \\ \left(\dfrac{1}{2}\left(3\xi - \sqrt{5(\xi^2+4\varepsilon)}\right), -\dfrac{\sqrt{5b}\xi - 3\sqrt{b\left(\xi^2+4\varepsilon\right)}}{2b}\right) \end{array} \right\}$$

Similarly to the case where p_1 is located on $\mathfrak{h}$, we want to maximize the minimal angle; in other words find the value of ξ for which the minimal angle of $T'(\xi)$ is maximized. Again, we can determine that the for $\xi_0 = \sqrt{\varepsilon}$ the minimal angle is maximized. Once again, the resulting triangle is an isosceles. Let us set $T' = T'(\xi_0)$, which is explicitly given by

$$T' = \operatorname{conv}\left\{(0,0), \left(\sqrt{\varepsilon}, \sqrt{\frac{5\varepsilon}{b}}\right), \left(-\sqrt{\varepsilon}, \sqrt{\frac{5\varepsilon}{b}}\right)\right\} \tag{3.26}$$

Note that in this case, for $b = 5/3$ the corresponding triangle is equilateral.

Remark 3.4 (Relation between the cases). We considered here two cases depending on the initial choice of p_1. In particular, p_1 can lie either on $\mathfrak{h}$ or on $\mathfrak{h}'$. In the first case, cf. Figure 3.20, we parameterize six families of triangles and we denoted them by $T_i(\eta)$. Similarly, in the second case we obtain six one-parameter families of triangles as well, and we denote them by $T_i^c(\xi)$. The details on the second case can be found in Appendix 3.C. One can verify that if

$$\eta = \frac{1}{2}\left(\frac{\sqrt{5}\xi}{\sqrt{b}} - \frac{\sqrt{b(\xi^2+4\varepsilon)}}{b}\right),$$

then $T_3(\eta) = T_5^c(\xi)$ for $\xi \in \mathbb{R}$.[4]

The Role of the parameter *b*. By now, given a canonical saddle S of the form $z = x^2 - by^2$ for some $b > 0$, we obtained two optimal triangles T and T' as can be found in Equations (3.25) and (3.26), respectively. Either of the triangles can

[4] For reference, in Appendix 3.C, $T_i(\eta)$ and $T_i^c(\xi)$ are named `optTs` and `conjOptTs`, respectively.

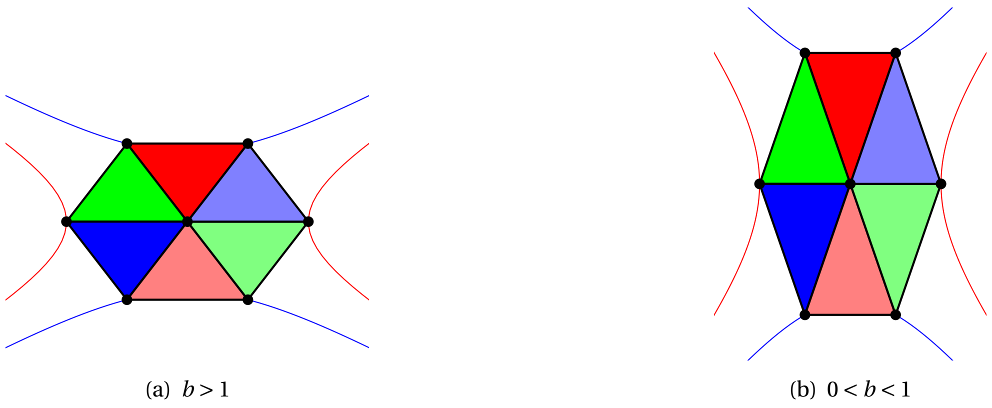

(a) $b > 1$ (b) $0 < b < 1$

Figure 3.23: Similar triangulation as the one obtained in Figure 3.22, only here T' from Equation (3.26) is used. Note that here the each of the red triangles has two vertices on $\mathfrak{h}'$.

be used to generate an approximation of the saddle S. Furthermore, the areas of the triangles is the same, and thus, in this perspective, the two possibilities are equally good. However, as can be seen in Figures 3.22 and 3.23, depending on the parameter b, the quality of the triangles changes. In particular, the images suggest that for $0 < b < 1$ the triangle T has to be chosen, and if $b > 1$ then it should be T'. The following lemma establishes the above observation and summarizes the discussion.

Lemma 3.12. *Let S be a saddle surface given by $z = x^2 - by^2$, with $b > 0$ and let $\varepsilon > 0$. Set*

$$T^\dagger = \begin{cases} T & \text{if } 0 < b < 1 \\ T' & \text{if } b \geq 1 \end{cases}$$

where T and T' are given in Equations (3.25) *and* (3.26)*, respectively. In addition, let $\hat{T}^\dagger$ be the lifting of $T^\dagger$ to the saddle S. Finally, let τ and $\hat{\tau}$ be an arbitrary planar triangle and its lifting to S, respectively. Then the following hold:*

1. $\operatorname{dist}_V\left(S, \hat{T}^\dagger\right) = \varepsilon$.
2. *If* $\operatorname{dist}_V(S, \hat{\tau}) = \varepsilon$*, then the area of τ is bounded from above by the area of $T^\dagger$.*
3. *If* $\operatorname{dist}_V(S, \hat{\tau}) = \varepsilon$ *and* $\operatorname{area}(\tau) = \operatorname{area}(T^\dagger)$*, then the minimal angle of τ is bounded from above by the the minimal angle of $T^\dagger$.*

Proof. Let us first verify that the construction indeed yields $\operatorname{dist}_V\left(S, \hat{T}^\dagger\right) = \varepsilon$. To that end, we have to compute the vertical distance between the edges of $\hat{T}^\dagger$ and S. This is directly done, by considering the vertices of the planar triangles T from Equation (3.25) and T' from Equation (3.26) and verifying the vertical distance using Lemma 3.3.

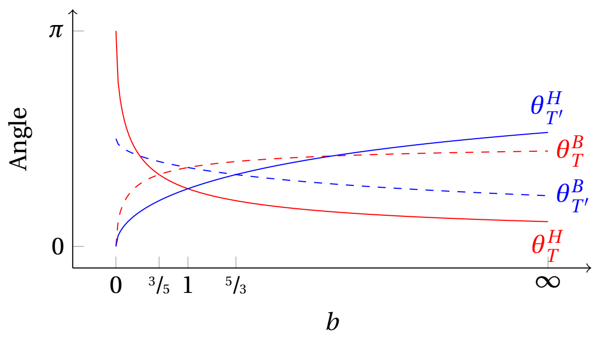


Figure 3.24: The angles of the triangles T and T' as functions of the parameter b. Note that for $b = 3/5$ the triangle T is equilateral, and for $b = 5/3$ the triangle T' is equilateral.

The second part of the lemma is guaranteed by the construction of the planar triangles T and T'. It is left to prove the third part. To that end, we have to consider the angles of the triangles T and T'. In particular, we have to show that the minimal angle of T is larger then the minimal angle of T' if $0 < b < 1$ and vice versa for $b \geq 1$.

Indeed, a direct computation yields that the head and base angles of T are

$$\theta^H_T = \arccos\left(\frac{5b-1}{1+5b}\right) \text{ and } \theta^B_T = \arccos\left(\frac{1}{\sqrt{1+5b}}\right),$$

respectively. Similarly, the head and base angles of T' are

$$\theta^H_{T'} = \arccos\left(\frac{5-b}{5+b}\right) \text{ and } \theta^B_{T'} = \arccos\left(\sqrt{\frac{b}{5+b}}\right),$$

respectively. In Figure 3.24 these angles are plotted as functions of b. An elementary analysis of these functions establishes the third part of the lemma. $\square$

We can conclude, that using the triangle $T^\dagger$ we can obtain the desired triangulation of a corresponding saddle surface, which is given by $z = x^2 - by^2$ with $b > 0$. More precisely, using translations and point reflections of $T^\dagger$ we can obtain a triangulation of some domain $D \subset \mathbb{R}^2$. In turn, we can lift this planar triangulation to the given saddle and obtain an optimal approximation of the surface.

Relation to the simple case. Next we want to establish two relations between the canonical case discussed in this section and the simple one that we discussed in Section 3.5.1. In particular, we discuss how one can derive an optimal approximation of the simple saddle from the canonical one and vice versa. The first direction is obvious; the case of the simple saddle is a special case of the canonical one. In particular, for $b = 1$ the saddle S is a rotation of the simple saddle. Indeed, in this case, we have that the minimal angles of T and T' are the same (cf. Figure 3.24), and thus they are equally good for the sake of an optimal approximation. In other words, the two triangulation that we discuss in Remark 3.2 on 60 are nothing but rotated images of T and T' from the current section.

The second relation, however, is less trivial. In Remark 3.2 we discuss two possible triangulations of the simple saddle that can be obtained. One when setting initially $p_1 = \left(\xi, \frac{4\varepsilon}{\xi}\right)$ and the second when setting $p_1 = \left(\xi, -\frac{4\varepsilon}{\xi}\right)$. Let $T_{\bar{S}}$ and $T'_{\bar{S}}$ be one of the triangles that we obtain in the first and second choices (when considering the *simple* saddle), respectively. Furthermore, let $\hat{T}_{\bar{S}}$ and $\hat{T}'_{\bar{S}}$ be their liftings to the simple saddle, which we denote here by $\bar{S} = \left\{(x, y, z) : z = xy\right\}$. In addition, let us denote two transformations. First, let $\mathscr{R}\colon \mathbb{R}^3 \to \mathbb{R}^3$ be a $\pi/4$ clockwise rotation around the z-axis. Then, $\mathscr{R}(\bar{S}) = \left\{(x, y, z) : z = \frac{1}{2}x^2 - \frac{1}{2}y^2\right\}$ is a rectangular saddle. Secondly,

$$S = \mathscr{A}(\mathscr{R}(\bar{S})) = \left\{(x, y, z) : z = x^2 - by^2\right\},$$

where $\mathscr{A} : \mathbb{R}^3 \to \mathbb{R}^3$ is an anisotropic map that is given by

$$(x, y, z) \mapsto \left(\frac{1}{\sqrt{2}}x, \frac{1}{\sqrt{2b}}y, z\right).$$

In other words, using the maps $\mathscr{R}$ and $\mathscr{S}$, we can transform the simple saddle $\bar{S}$ to a canonical one of the form we discuss in this section. Note that $\mathscr{R}$ and $\mathscr{S}$ are strongly related to the maps discussed in Section 3.2. Finally, one can easily verify that $\hat{T} = \mathscr{A}(\mathscr{R}(\hat{T}_{\bar{S}}))$ and $\hat{T}' = \mathscr{A}(\mathscr{R}(\hat{T}'_{\bar{S}}))$, where $\hat{T}$ and $\hat{T}'$ are the liftings to the canonical surface S of the triangle T that is defined in Equation (3.25) and T' from Equation (3.26), respectively.

The second relation shows that it is possible to use the triangles that we obtained in the case of the simple saddle for the sake of optimal triangulation of a canonical saddle. However, this is not straightforward and one has to be careful. Although both possible triangulations in the simple case are equally good, due to the symmetry of the simple saddle, it is no longer the case when they are transformed to the canonical saddle. Thus, when using the maps $\mathscr{A}$ and $\mathscr{R}$ one has to choose carefully, depending on the properties of the canonical saddle at hand, the triangulation from the simple case. Given the correct choice, the composition of the maps mentioned above maps optimal triangles to optimal triangles; here optimality stands both for the error bound and the shape of the triangles. This observation is surprising since the composition of the maps *changes* the shape of the triangles.

The conjugate case and General Saddles. In this section we handled merely the saddle which is given by $z = x^2 - by^2$, such that $b > 0$. The conjugate surface, which is given by $z = -x^2 + by^2$, can be approximated using the same triangles T and T'; one simply has to lift them to the conjugate surface. Note that lifting a 90° rotation of either T or T' around the origin yields an approximation of the surface that is given by $z = bx^2 - y$.

Finally, let us briefly outline how to obtain an optimal triangulation of a generic saddle which is given by $z = \alpha x^2 - \beta y^2$ such that $\alpha \cdot \beta > 0$ and given some fixed $\varepsilon' > 0$. Let us set $\varepsilon = \varepsilon'/\alpha$ and $b = \beta/\alpha$; assume that $0 < b < 1$. Let us now obtain the planar triangle T from Equation (3.25). In particular we have the following planar triangle

$$\Theta = \operatorname{conv}\left\{(0,0), \left(\sqrt{\frac{5\varepsilon}{\alpha}}, -\sqrt{\frac{\varepsilon}{\beta}}\right), \left(\sqrt{\frac{5\varepsilon}{\alpha}}, \sqrt{\frac{\varepsilon}{\beta}}\right)\right\} \tag{3.27}$$

Let us lift this triangle to the saddle $z = x^2 - by^2$ and obtain $\hat{\Theta}$ that attains a maximal vertical distance ε to the saddle. The map $\mathscr{S}_\alpha : \mathbb{R}^3 \to \mathbb{R}^3$ given by

$$\mathscr{S}_\alpha((x,y,z)) = (x,y,\alpha z),$$

maps the vertices of $\hat{\Theta}$ to the generic canonical saddle $z = \alpha x^2 - \beta y^2$. In particular, we have

$$\mathscr{S}_\alpha\left(\hat{\Theta}\right) = \operatorname{conv}\left\{(0,0,0), \left(\sqrt{\frac{5\varepsilon}{\alpha}}, -\sqrt{\frac{\varepsilon}{\beta}}, 4\varepsilon\right), \left(\sqrt{\frac{5\varepsilon}{\alpha}}, \sqrt{\frac{\varepsilon}{\beta}}, 4\varepsilon\right)\right\}.$$

It is easy to verify that the maximal vertical distance between $\mathscr{S}_\alpha\left(\hat{\Theta}\right)$ and the generic canonical saddle we started with equals ε'. Naturally, if $b > 1$, then we should obtain Θ as derived from T' in Equation (3.26) as discussed in Lemma 3.12. This concludes the discussion on the optimal (interpolating) triangulation of canonical saddles.

Remark 3.5 (Approximate General Saddles). We end the discussion by pointing that a general saddle can be obtained by first rotating a canonical saddle and then translating it in space. Thus, one can use the triangulation that we obtain in this section also for the approximation of general saddles. In Figure 3.25 an example of the local, optimal and interpolating approximation of a *general saddle* surface is plotted. Note that in the plotted example, the center of the triangulation is *not* the origin. In contrast, in Figure 3.26 we plot the triangulation that is obtained when using the conjugate planar triangulation. In particular, compare the quality of the planar triangles in both cases.

3.6 Non-interpolating Triangulation

In this section we improve the local triangulation that we obtained in Section 3.5. While the approach described in the former section yields triangles that *interpolate* the smooth saddle, in this section we will allow *non-interpolating* triangles. By doing so, we increase the freedom to choose the approximating triangles. This, in turn, yields an improvement of the approximation. The improved triangulation presented in this section invalidates the conjecture of Pottmann et al. [34] that we discussed in the introduction. For the sake of simplicity, we consider in this section the simple saddle.

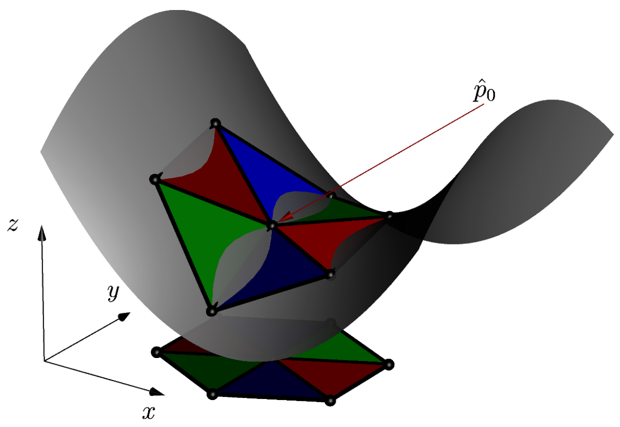


Figure 3.25: An optimal planar triangulation together with its lifting to a general saddle surfaces. In this example, p_0 is not at the origin and a translation of the planar triangulation is applied. As a result, the central vertex of the approximating patch, denoted by $\hat{p}_0$, is not at the origin.

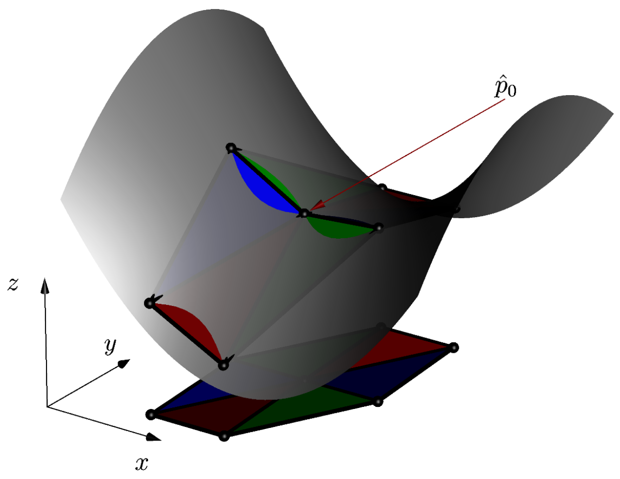


Figure 3.26: Approximation of the same saddle as in Figure 3.25 using the conjugate planar triangles.

We follow a similar path as the one we took in Section 3.5.1. Our goal is to find a planar triangle $T_\star$ with vertices p_0, p_1 and p_2 such that

$$\operatorname{dist}_V\left(S, \hat{T}_\star\right) \leq \varepsilon,$$

where $\hat{T}_\star$ is the lifting of $T_\star$ to the *offset saddle*

$$S_\alpha = \left\{(x, y, z) : z = xy + \alpha\right\} \tag{3.28}$$

for $\alpha > 0$. Note that for $\alpha > \varepsilon$ we have that $\operatorname{dist}_V\left(S, \hat{T}_\star\right) > \varepsilon$. Thus, we can safely assume that $\varepsilon \geq \alpha \geq 0$. Similarly to Lemma 3.3, let $\ell_i(t)$ be a parameterization of the i-th edge of $\hat{T}_\star$. In turn, it is easy to find the vertical distance from $\ell_i(t)$ to S parameterized by $t \in [0,1]$. It is easy to establish that the extrema of the vertical distance functions are attained either at the midpoints or the endpoints of the edges. Note that at the endpoints the vertical distance from an edge to the surface is α. Therefore, since we want to bound the distance by ε and we assume that $\varepsilon \geq \alpha$ we have for $i \in \{0,1,2\}$

$$\operatorname{dist}_V(S, \ell_i) = \left|\frac{1}{4}(p_i^x - p_k^x)(p_i^y - p_k^y) + \alpha\right| \quad \text{s.t. } k = i+1 \mod 3 \tag{3.29}$$

From this, we can derive the following lemma.

Lemma 3.13 (Translation Invariant)**.** *Given a planar triangle T with vertices p_0, p_1 and p_2 and another triangle T', which is a translation of T by a vector $\vec{v}$, we have that*

$$\operatorname{dist}_V\left(S, \hat{T}\right) = \operatorname{dist}_V\left(S, \hat{T}'\right),$$

where $\hat{T}$ and $\hat{T}'$ are the liftings of T and T' to the offset saddle S_α, respectively.

Proof. Similarly to Equation (3.29), for all $i \in \{0,1,2\}$ we have that

$$\begin{aligned}\operatorname{dist}_V(S, \ell_i) &= \left|\frac{1}{4}(p_i^x - p_k^x)(p_i^y - p_k^y) + \alpha\right| \\ &= \left|\frac{1}{4}\left((p_i^x + v^x) - (p_k^x + v^x)\right)\left((p_i^y + v^y) - (p_k^y + v^y)\right) + \alpha\right| \\ &= \operatorname{dist}_V\left(S, \ell_i'\right).\end{aligned}$$

Here ℓ_i and ℓ_i' are the i-th edges of $\hat{T}$ and $\hat{T}'$, respectively. Therefore we have,

$$\begin{aligned}\operatorname{dist}_V\left(S, \hat{T}\right) &= \max_{i\in\{0,1,2\}} \operatorname{dist}_V(S, \ell_i) \\ &= \max_{i\in\{0,1,2\}} \operatorname{dist}_V\left(S, \ell_i'\right) \quad = \operatorname{dist}_V\left(S, \hat{T}'\right).\end{aligned}$$

This concludes the proof. □

In turn, we can assume, without loss of generality, that p_0 is at the origin. It is our next goal to determine the locations for the vertices p_1 and p_2, such that $\operatorname{dist}_V(S, \ell_i) \leq \varepsilon$ and the area of the corresponding planar triangle $T_\star$ will be maximal.

Theorem 3.14. *Let $\varepsilon > 0$ be fixed and set $T_\star \subset \mathbb{R}^2$ to be the triangle with the following vertices:*

$$\begin{aligned} p_0 &= (0,0) \\ p_1 &= 2\sqrt{\varepsilon}\left(1+\frac{1}{\sqrt{3}}, 1-\frac{1}{\sqrt{3}}\right) \\ p_2 &= 2\sqrt{\varepsilon}\left(1-\frac{1}{\sqrt{3}}, 1+\frac{1}{\sqrt{3}}\right) \end{aligned}$$

If $\hat{T}_\star$ is the lifting of the triangle $T_\star$ to the offset saddle S_{α_0} for $\alpha_0 = \frac{\varepsilon}{3}$, then the following hold

1. Vertical distance: $\operatorname{dist}_V\left(S, \hat{T}_\star\right) = \varepsilon$
2. Area of projection to the plane: $\operatorname{area}(T_\star) = \frac{8\varepsilon}{\sqrt{3}} \approx 4.6188\varepsilon$

Proof. Plugging in the coordinates of $T_\star$ into the expressions for the verticals distances from the edges of $\hat{T}_\star$ to the simple saddle, cf. Equation (3.29), yields directly that $\operatorname{dist}_V\left(S, \hat{T}_\star\right) = \varepsilon$. Similarly, direct evaluation yields the area of $T_\star$. □

Before discussing the technical details that yields the "right" coordinates in the last theorem, we discuss its significance. Recall that in Section 3.5.1, where we considered the interpolating approximation of a simple saddle, we obtained that the area of the planar projection of the approximating triangles is $2\sqrt{5}\varepsilon \approx 4.4721\varepsilon$ and smaller than $\operatorname{area}(T_\star)$. Thus, the triangle $T_\star$ defined in Theorem 3.14 allows us to use fewer triangles in order to approximate a simple saddle and, in the meanwhile, maintain the same vertical distance, namely, ε. This result presents a counterexample to the conjecture presented in [34].

Next we discuss the finding of the coordinates p_1 and p_2 as given in Theorem 3.14. As we want to maximize the area of the triangle $T_\star$, we have to solve the following system of equations

$$\begin{cases} \left|\frac{1}{4}p_1^x p_1^y + \alpha\right| = \varepsilon \\ \left|\frac{1}{4}(p_1^x - p_2^x)(p_1^y - p_2^y) + \alpha\right| = \varepsilon \\ \left|\frac{1}{4}p_2^x p_2^y + \alpha\right| = \varepsilon \end{cases} \tag{3.30}$$

which is derived from Equation (3.29). Let us assume that p_1 lies in the first quadrant; thus, by setting $p_1^x = \xi$ we can obtain from the first equation above the following parameterization

$$p_1(\xi,\alpha) = \left(\xi, \frac{4(\varepsilon-\alpha)}{\xi}\right), \tag{3.31}$$

for $\xi > 0$ and $\varepsilon \geq \alpha \geq 0$. Next, from the last two equations, we obtain *six* possible (real) solutions for p_2, all parameterized by ξ and α as well. Let us denote the resulting triangles by $T_\star^i(\xi,\alpha)$ for $i \in \{1,\dots,6\}$. For the sake of brevity, the explicit

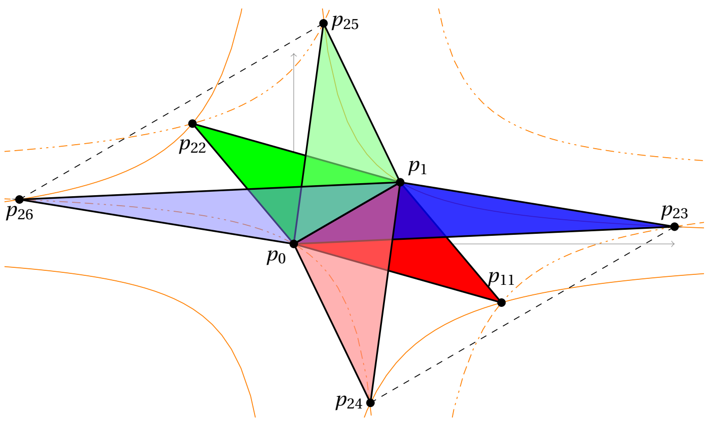


Figure 3.27: Generic setting of the family $T^i_\star(\xi,\alpha)$ for some fixed parameters, such that $\alpha > 0$.

expressions of the third coordinates can be found in Appendix 3.B. A generic instance, for some fixed parameters ε, α and ξ, is illustrated in Figure 3.27. It is easy to verify, using the explicit coordinates of the vertices p_{2i}, that the lines $\overline{p_{23}p_{24}}$, $\overline{p_{25}p_{26}}$ and $\overline{p_0p_1}$ are all parallel. Thus the corresponding triangles have the same area, which is, since we assume that $\alpha > 0$, *larger* than the area of $T^1_\star(\xi,\alpha)$ and $T^2_\star(\xi,\alpha)$. In addition we note that the areas of all the triangles depend only on ε and α, but *not* on ξ. Finally, the vertical distance between $\hat{T}^i_\star(\xi,\alpha)$ and the simple saddle is fixed ε. Lemmas 3.13 and 3.15 show that the vertical distance is invariant under translations and point reflections of the planar triangles, even when they are lifted to an offset saddle. Thus, we can use one triangle together with its point reflections (with respect to the midpoints of its edges) and translations to form a local triangulation. Therefore we can choose, without loss of generality, one of the four largest triangles. Let $T_\star(\xi,\alpha)$ denote $T^5_\star(\xi,\alpha)$. As the area of $T_\star(\xi,\alpha)$ does not depend on ξ, our next goal is to eliminate this parameter.

Indeed, by employing the so-called *pseudo-Euclidean transformation*, we can optimize the shape of $T_\star(\xi,\alpha)$ and eliminate the parameter ξ. A map $\mathscr{P}_\lambda : \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$\mathscr{P}_\lambda(x,y) \mapsto \left(\lambda x, \frac{1}{\lambda} y\right) \tag{3.32}$$

where $\lambda > 0$ is called *pseudo-Euclidean transformation*. In Figure 3.28 the effect of such transformations is illustrated. Note that if one of the vertices of a triangle T is at the origin and the other two lie on the same hyperbola, then all images of T under pseudo-Euclidean transformations lie on that hyperbola. Furthermore, in this case, we have also that $\operatorname{dist}_V\left(S,\hat{T}\right) = \operatorname{dist}_V\left(S,\widehat{\mathscr{P}_\lambda(T)}\right)$, such that $\hat{T}$ and $\widehat{\mathscr{P}_\lambda(T)}$ are the liftings of the triangles T and $\mathscr{P}_\lambda(T)$, respectively, to S_α, for all parameters λ. Finally, It is easy to verify that $\operatorname{area}(T) = \operatorname{area}(\mathscr{P}_\lambda(T))$ for all planar triangles.

Since one vertex of $T_\star(\xi,\alpha)$ is at the origin and the other two lie on the same hyperbola, we have that all triangles $\mathscr{P}_\lambda(T_\star(\xi,\alpha))$ are of the same area and maintain

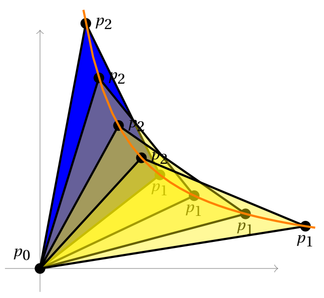


Figure 3.28: A triangle (in blue) and its three images under different pseudo-Euclidean transformations.

the same fixed vertical distance when lifted to S_α. Therefore, we can assume that $T_\star(\xi,\alpha)$ is symmetric with respect to the line $y=x$. In particular, for

$$\xi_0 = \sqrt{2}\sqrt{3\varepsilon-\alpha+\sqrt{(5\varepsilon-3\alpha)(\alpha+\varepsilon)}}$$

we have that $T_\star(\xi_0,\alpha)$ is such a triangle (cf. Figure 3.29).

We can now easily find that the area of $T_\star(\xi_0,\alpha)$ satisfies

$$\operatorname{area}(T_\star(\xi_0,\alpha)) = 2\sqrt{(5\varepsilon-3\alpha)(\alpha+\varepsilon)}.$$

In turn, for $\alpha_0 = \frac{\varepsilon}{3}$ we have that the area of $T_\star(\xi_0,\alpha)$ is maximized. Note that $T_\star(\xi_0,\alpha_0)$ is the same triangle as $T_\star$ from the statement of Theorem 3.14.

Remark 3.6. The triangle $T_\star$ is equilateral. In the case of a simple hyperbolic paraboloid we obtain that optimal *interpolating* triangles are equilaterals (cf. [34]). In the case at hand, of a simple hyperbolic paraboloid we obtain, again, optimality for equilateral triangles, however this time the yielded approximation is not interpolating the smooth surface. Furthermore, it is easy to verify that the triangle $T_\star(\xi_0,0)$ is identical to $T_3^{opt}(\varepsilon)$ from Equation (3.20). As expected, in this case, where $\alpha=0$, $T_\star(\xi_0,0)$ has to be lifted to $S_0=S$. In other words the discussion in Section 3.5.1 is merely a special case with $\alpha=0$ of the current one.

If we want to use $T_\star$ as a building block for the approximation of the simple saddle S, we miss one more element. Namely, we have to verify that the vertical distance is invariant under point reflections of triangles also in the case where the triangles are lifted to an offset saddle (cf. Lemma 3.10).

Lemma 3.15 (Reflection Invariant). *Let T be a planar triangle with vertices p_0, p_1 and p_2, and let T' be its point reflection with respect to the midpoint of one of its edges. Then, we have that*

$$\operatorname{dist}_V\left(S,\hat{T}\right) = \operatorname{dist}_V\left(S,\hat{T}'\right),$$

where $\hat{T}$ and $\hat{T}'$ are the liftings of T and T' to the offset saddle S_α, respectively.

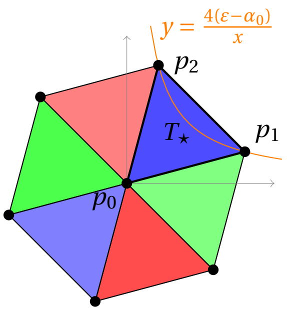


Figure 3.29: The triangle $T_\star$ from Theorem 3.14 along with five of its images under translations and point reflections. Together, the six triangles form a local triangulation of a neighborhood of the origin. Lifting them to the offset saddle S_{α_0} yields a local *non-interpolating* approximation of the simple saddle S, which maintain a vertical distance equals ε.

Proof. Without loss of generality, let T' be the point reflection of T with respect to the midpoint of the edge connecting p_1 and p_2 (cf. Figure 3.19). In particular, the new vertex of T', denoted by q_0, is given by $q_0 = p_2 + p_1 - p_0$. In turn, from Equation (3.29), we have that

$$\begin{aligned}\operatorname{dist}_V\left(S,\ell'_0\right) &= \operatorname{dist}_V\left(S,\ell_2\right)\\ \operatorname{dist}_V\left(S,\ell'_1\right) &= \operatorname{dist}_V\left(S,\ell_1\right)\\ \operatorname{dist}_V\left(S,\ell'_2\right) &= \operatorname{dist}_V\left(S,\ell_0\right)\end{aligned}$$

Thus we have $\operatorname{dist}_V\left(S,\hat{T}\right) = \operatorname{dist}_V\left(S,\hat{T}'\right)$. □

Finally, using Lemmas 3.13 and 3.15 we can triangulate a neighborhood of the origin, as depicted in Figure 3.29. Lifting the obtained triangles to S_{α_0} yields a local approximation of the simple saddle S. Furthermore, by applying further translations and point reflections, we can obtain a triangulation of $\mathbb{R}^2$, which yields, when lifted to S_{α_0}, a global and non-interpolating approximation $\mathcal{T}$ of the simple saddle S. For all triangles $T \in \mathcal{T}$ we have $\operatorname{dist}_V(S,T) = \varepsilon$.

The Case Where $\alpha < 0$. So far we considered the case where $\alpha \geq 0$ and smaller than ε. We now review the case where $\alpha < 0$. In this case (cf. Figure 3.30), the triangles $T^1_\star(\xi,\alpha)$ and $T^2_\star(\xi,\alpha)$ are of largest area among all six possible triangles. In particular, their areas are given by

$$2\sqrt{(\varepsilon-\alpha)(3\alpha+5\varepsilon)}$$

as functions of α. Furthermore, it is easy to verify that this functions attains a global extremum for

$$\alpha_0^- = -\frac{\varepsilon}{3}.$$

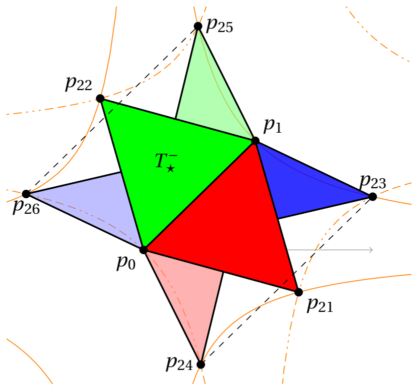


Figure 3.30: The family $T_\star^i(\xi_0^-, \alpha_0^-)$ of triangles, such that $\alpha < 0$. Note that the triangles that correspond to the vertices p_{21} and p_{22} are equilateral triangles.

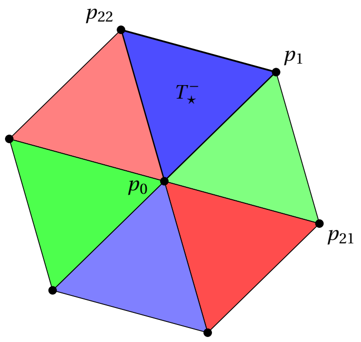


Figure 3.31: A triangulation of a neighborhood of the origin generated by $T_\star^-$. Lifting this planar triangulation to the offset saddle $S_{\alpha_0^-}$ yields an optimal non-interpolating approximation of the simple saddle.

In this case, it follows that the area of the planar triangles is $\frac{8\varepsilon}{\sqrt{3}}$, like in the case of $\alpha > 0$ that we discuss in Theorem 3.14. Finally, for

$$\xi_0^- = 4\sqrt{\frac{\varepsilon}{3}}$$

we have that the triangles are equilaterals. Without loss of generality, we set (cf. Figure 3.30)

$$T_\star^- = T_\star^2(\xi_0^-, \alpha_0^-),$$

and it is easy to verify that $\text{dist}_V\left(S, \hat{T}_\star^-\right) = \varepsilon$ where $\hat{T}_\star^-$ is the lifting of $T_\star^-$ to the offset saddle $S_{\alpha_0^-}$. We can now tile the plane using translations and point reflections of $T_\star^-$; see Figure 3.31 and compare to Figure 3.29. In turn, lifting the tiling to the offset saddle $S_{\alpha_0^-}$ yields and optimal non-interpolating approximation of the simple saddle.

We conclude this paragraph by considering the *rotation* of $T_\star$, which is defined in Theorem 3.14. Let $T_\star'$ be a *rotation* of $T_\star$ by 90° around the origin. Next, we can verify that

$$\text{dist}_V\left(S, \hat{T}_\star'\right) = \varepsilon,$$

where $\hat{T}_\star'$ is the lifting of $T_\star'$ to the offset saddle $S_{\alpha_0^-}$.

3.7 Conclusion

In this chapter we considered the problem of approximating a saddle surfaces. In particular, we found a triangular mesh that approximates a given saddle surface with respect to the so-called *vertical distance* (cf. Section 3.4), which served as the error measure. As a matter of fact, we derived two possible meshes. The first has its vertices lying *on* the surface; this is what we call an *interpolating* approximation — this case is discussed in Section 3.5. The second mesh has its vertices lying on a saddle which is an offset of the given surface. In particular, in this latter case, the vertices of the mesh do not lie on the given saddle surface, and thus yield a *non-interpolating* approximation. In Section 3.6 we discussed this case while considering the simple saddle.

Furthermore, we showed that the obtained approximations are optimal in the following sense. First, let us focus on the interpolating case. Let S be a general surface, given by $\{(x, y, z) : z = \alpha x^2 - \beta y^2\}$. In addition, let T be some planar triangle such that $\operatorname{dist}_V(S, \hat{T}) \le \varepsilon$, where $\hat{T}$ is a lifting of T to the saddle surface S and $\varepsilon > 0$. Then, we established that

$$\operatorname{area}(T) \le \varepsilon \sqrt{\frac{5}{\alpha\beta}} = \operatorname{area}(\Theta),$$

where Θ is given in Equation (3.27).[5] Moreover, if $\operatorname{area}(T) = \operatorname{area}(\Theta)$, then the minimal angle of T is bounded from above by the minimal angle of Θ.

When considering the non-interpolating case, we have a similar notion of optimality. Let S be a simple saddle and S_α be an α-offset of S for some $\alpha > 0$. Again, let T be an arbitrary planar triangle and $\hat{T}$ be its lifting to S_α. Thus, if $\operatorname{dist}_V(S, \hat{T}) \le \varepsilon$ for some $\varepsilon > 0$, then

$$\operatorname{area}(T) \le \frac{8\varepsilon}{\sqrt{3}} = \operatorname{area}(T_\star),$$

where $T_\star$ is defined in Theorem 3.14. Furthermore, since $T_\star$ is an equilateral triangle, we have that the minimal angle of T is at most as large as the (minimal) angle of $T_\star$.

Let us conclude this short summary with the following observation. If ℓ is an edge of an optimal triangle, then it can never be contained in a ruling of the saddle (in the interpolating case) nor can its projection to the saddle be contained in a ruling in the non-interpolating case. In addition, the triangles do align with the principal curvature directions of the saddle. More precisely, if we consider the neighborhood of the origin and focus on a triangle that has two vertices on the *same* hyperbola, then this edge points in one principal curvature direction and its $90°$ rotation points in the second one. For reference see Figures 3.11, 3.12, 3.22, 3.23 and 3.29.

Future work. In this chapter two interesting issues were left untreated. First, let us stress that the optimal approximations yielded in this chapter are of local

[5] The triangle Θ in Equation (3.27) is defined under the assumption that $0 < \beta/\alpha < 1$. If $\beta/\alpha > 1$, then we have to define Θ by taking T' from Equation (3.26). However, regardless of α and β, the area of Θ is constant.

nature. In particular, if T is an optimal triangle obtained in this chapter and $T_{\vec{v}}$ is a translated copy of T by a vector $\vec{v} \in \mathbb{R}^2$, then the lifted triangle $\hat{T}_{\vec{v}}$ degenerates (w.r.t. the minimal angle of the triangle) depending on the magnitude of $\vec{v}$ while still maintaining the same vertical distance from the saddle surface. However, as we discussed in Section 3.4, the vertical distance is no longer a good upper bounded on the Hausdorff distance once T is translated away from the origin. Therefore, it is interesting to obtain several (optimal) local approximation of a saddle, centered at different points of the saddle, and glue them together into one approximating mesh. To that end it is worthy to mention the work of Bertram et al. [7], where they considered the stitching of local triangulations into a global one. In addition, one should keep in mind that the optimal triangles presented in this dissertation are members of one-parameter families of triangles. This parameter could be useful when addressing the issue we present here.

The second interesting issue deals with the non-interpolating case. In this chapter we considered a single lifting scheme; each vertex of a planar triangle is vertically lifted either to a given saddle surface or to an offset of the given saddle. Other lifting schemes were not considered, thus it is left open whether one can obtain an improved triangulation by considering some different lifting.

3.A Explicit Computation of the Angles of $T_1(\xi)$

When optimizing the shape of the planar triangles, in the case of the simple saddle, in Section 3.5.1, we analyze the angle functions of the triangle $T_1(\xi)$ as functions of ξ. For the sake of fluent reading, the expressions are presented in this appendix. First, we obtain the functions of the three angles of the triangle as follow:

$$\begin{aligned}
\alpha_0(\xi) &= \arccos\left(\frac{32\varepsilon^2-(3+\sqrt{5})\xi^4}{\sqrt{2}\sqrt{(16\varepsilon^2+\xi^4)(32\varepsilon^2+(7+3\sqrt{5})\xi^4)}}\right)\\
\alpha_1(\xi) &= \arccos\left(\frac{8(\sqrt{5}-1)\varepsilon^2+(2+\sqrt{5})\xi^4}{\sqrt{(16\varepsilon^2+\xi^4)(8(3-\sqrt{5})\varepsilon^2+(9+4\sqrt{5})\xi^4)}}\right)\\
\alpha_2(\xi) &= \arccos\left(\frac{16(1-\sqrt{5})\varepsilon^2+(11+5\sqrt{5})\xi^4}{\sqrt{512(3-\sqrt{5})\varepsilon^4+96(7+3\sqrt{5})\varepsilon^2\xi^4+2(123+55\sqrt{5})\xi^8}}\right).
\end{aligned}$$

In Figure 3.10 the angle functions of $T_1(\xi)$ are plotted, as functions of ξ.[6] Next we can easily find the ξ values of the intersections between all functions. In particular, $\alpha_0(\xi)$ and $\alpha_1(\xi)$ intersect for

$$\begin{aligned}
\xi_0^c &= 0\\
\xi_1^c &= 2\sqrt{\sqrt{5}-2}\sqrt{\varepsilon} \approx 0.971737\sqrt{\varepsilon}
\end{aligned}$$

Similarly, $\alpha_0(\xi)$ and $\alpha_2(\xi)$ intersect for

$$\xi_2^c = \frac{2}{\varphi}\sqrt{\varepsilon} \approx 1.23607\sqrt{\varepsilon}.$$

Finally, $\alpha_1(\xi)$ and $\alpha_2(\xi)$ intersect for

$$\xi_3^c = \sqrt{2\left(\sqrt{5}-1\right)}\sqrt{\varepsilon} \approx 1.5723\sqrt{\varepsilon}.$$

Since the functions are continuous, in each interval $[\xi_i^c, \xi_{i+1}^c)$, for $i \in \{0,1,2,3\}$, we can determine which function is the minimal.[7] Thus, we can determine the lower envelope and in particular find that its maximum in the interval $[0,\infty)$ is attained for $\alpha_1(\xi_0)$ where $\xi_0 = \xi_2^c = \frac{2}{\varphi}\sqrt{\varepsilon}$; cf. Figure 3.10. Therefore, the triangle $T_1(\xi_0)$ has the maximal minimal angle in the one-parameter family $T_1(\xi)$.

[6]In the illustrated example we chose fixed $\varepsilon = 1$.

[7]Here we let $\xi_4^c = \infty$.

3.B Expression for the Coordinates of $T^i_\star(\xi,\alpha)$

In this section we provide the expressions for the coordinates of the triangles $T^i_\star(\xi,\alpha)$, which were derived in Section 3.6. For the sake of completeness, recall that $p_0 = (0,0)$ and

$$p_1(\xi,\alpha) = \left(\xi, -\frac{4(\varepsilon-\alpha)}{\xi}\right).$$

Let $p_{2i}(\xi,\alpha)$ denote the third vertex of $T^i_\star(\xi,\alpha)$; in particular, the solutions of the last two equations in Equation (3.30) yield

$$p_{21}(\xi,\alpha) = \begin{pmatrix} \frac{\xi(\alpha-\varepsilon-\sqrt{(\varepsilon-\alpha)(5\varepsilon+3\alpha)})}{2(\alpha-\varepsilon)} \\ -\frac{2(\alpha-\varepsilon+\sqrt{(\varepsilon-\alpha)(5\varepsilon+3\alpha)})}{\xi} \end{pmatrix}$$

$$p_{22}(\xi,\alpha) = \begin{pmatrix} \frac{\xi(\alpha-\varepsilon+\sqrt{(\varepsilon-\alpha)(5\varepsilon+3\alpha)})}{2(\alpha-\varepsilon)} \\ -\frac{2(\alpha-\varepsilon-\sqrt{(\varepsilon-\alpha)(5\varepsilon+3\alpha)})}{\xi} \end{pmatrix}$$

$$p_{23}(\xi,\alpha) = \begin{pmatrix} \frac{\xi(\alpha-3\varepsilon-\sqrt{(\varepsilon+\alpha)(5\varepsilon-3\alpha)})}{2(\alpha-\varepsilon)} \\ -\frac{2(\alpha-3\varepsilon+\sqrt{(\varepsilon+\alpha)(5\varepsilon-3\alpha)})}{\xi} \end{pmatrix}$$

$$p_{24}(\xi,\alpha) = \begin{pmatrix} \frac{\xi(\alpha+\varepsilon-\sqrt{(\varepsilon+\alpha)(5\varepsilon-3\alpha)})}{2(\alpha-\varepsilon)} \\ -\frac{2(\alpha+\varepsilon+\sqrt{(\varepsilon+\alpha)(5\varepsilon-3\alpha)})}{\xi} \end{pmatrix}$$

$$p_{25}(\xi,\alpha) = \begin{pmatrix} \frac{\xi(\alpha-3\varepsilon+\sqrt{(\varepsilon+\alpha)(5\varepsilon-3\alpha)})}{2(\alpha-\varepsilon)} \\ -\frac{2(\alpha-3\varepsilon-\sqrt{(\varepsilon+\alpha)(5\varepsilon-3\alpha)})}{\xi} \end{pmatrix}$$

$$p_{26}(\xi,\alpha) = \begin{pmatrix} \frac{\xi(\alpha+\varepsilon+\sqrt{(\varepsilon+\alpha)(5\varepsilon-3\alpha)})}{2(\alpha-\varepsilon)} \\ -\frac{2(\alpha+\varepsilon-\sqrt{(\varepsilon+\alpha)(5\varepsilon-3\alpha)})}{\xi} \end{pmatrix}$$

3.C Symbolic Computations Transcript

In this appendix the transcript of the symbolic computations related to the case of the canonical saddle can be found. The computations were carried out using `Mathematica`.

Optimal (Local) Interpolatin1g Approximation of Canonical Saddle Surfaces

In this notebook we consider a canonical saddle surface of the form:

$z = x^2 - by^2$

such that b>0.

Let us define the functions that describe the surface, parameterized by the parameter b.

```
In[1]:= ClearAll[F, conjF]
        F[x_, y_] :=  x^2 - b y^2;
        F[p_ /; Length[p] == 2] := F[p[[1]], p[[2]]];
        conjF[x_, y_] := -F[x, y];
        conjF[p_ /; Length[p] == 2] := conjF[p[[1]], p[[2]]];
```

A generic planar triangle, with one vertex at the origin, which we will optimize.

```
In[6]:= ClearAll[t]
        t = {{0, 0}, {x1, y1}, {x2, y2}};
```

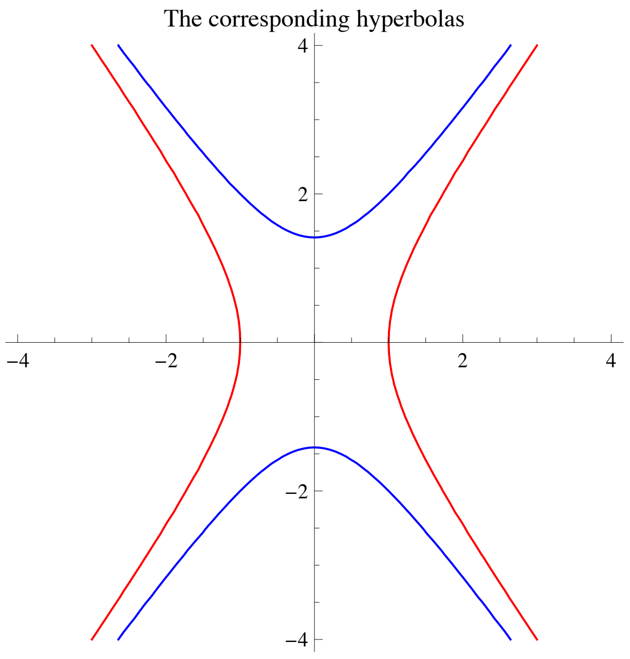
We shall use the following for plots of triangles in the plane.

```
In[8]:= ClearAll[colors]
        colors = {Red, Green, Blue, Darker[Red],
            Darker[Green], Darker[Blue]};
```

Under the above assumptions, we have that the hyperbola that corresponds to F(x,y)=1 is east-west open (in red) and its conjugate is north-south open (in blue)

```
In[10]:= Block[{b = .5, bnd = 4},
          Show[
           ContourPlot[F[x, y] == 1, {x, -bnd, bnd},
            {y, -bnd, bnd}, ContourStyle -> Red],
           ContourPlot[conjF[x, y] == 1, {x, -bnd, bnd},
            {y, -bnd, bnd}, ContourStyle -> Blue],
           ImageSize → 300, Frame → False, Axes → True,
           PlotLabel → "The corresponding hyperbolas"]
         ]
```

Out[10]=



We treat two cases, namely where $p_1 = (x_1, y_1)$ is located.

Red case:

Parameterize the vertices and obtain parameterized planar triangles

Here we assume that $p_1$is located on the red hyperbola.

```
In[11]:= $Assumptions = b > 0 && x1 ∈ Reals && y1 ∈ Reals && ϵ > 0;
```

Find the relation between x_1 and y_1.

```
In[12]:= ClearAll[p1Rule]
         p1Rule = Solve[1 / 4 F[t[[2]]] == ϵ, x1]
```

$$\text{Out[13]=}\quad \left\{\left\{x1 \to -\sqrt{b\, y1^2 + 4\, \epsilon}\right\}, \left\{x1 \to \sqrt{b\, y1^2 + 4\, \epsilon}\right\}\right\}$$

Without loss of generality we can choose$x_1$to be positive.

```
In[14]:= p1Rule = p1Rule[[2]];
```

Find the relation between x_2 and y_2. **Note that two solutions are imaginary!** Pick the real solutions.

```
In[15]:= ClearAll[p2Rule];
         p2Rule =
          Solve[1 / 4 Abs[F[t[[3]]]] == ϵ &&
             1 / 4 Abs[F[t[[3]] - t[[2]] /. p1Rule]] == ϵ,
            {x2, y2}] // Simplify
         p2Rule = p2Rule[[3 ;; 8]];
         (*Verify in the output of the previous line
          which entries are real!*)
```

$$\text{Out[16]=}\quad \left\{\left\{x2 \to \frac{b\, y1^2 + 4\, \epsilon + \sqrt{3}\, y1 \sqrt{-b\, (b\, y1^2 + 4\, \epsilon)}}{2 \sqrt{b\, y1^2 + 4\, \epsilon}},\right.\right.$$

$$\left. y2 \to \frac{b\, y1 + \sqrt{3}\, \sqrt{-b\, (b\, y1^2 + 4\, \epsilon)}}{2\, b}\right\},$$

$$\left\{x2 \to \frac{b\, y1^2 + 4\, \epsilon - i \sqrt{3}\, y1 \sqrt{b\, (b\, y1^2 + 4\, \epsilon)}}{2 \sqrt{b\, y1^2 + 4\, \epsilon}},\right.$$

$$\left. y2 \to \frac{b\, y1 - i \sqrt{3}\, \sqrt{b\, (b\, y1^2 + 4\, \epsilon)}}{2\, b}\right\},$$

$$\left\{x2 \to \frac{-b\, y1^2 - 4\, \epsilon + \sqrt{5}\, y1 \sqrt{b\, (b\, y1^2 + 4\, \epsilon)}}{2 \sqrt{b\, y1^2 + 4\, \epsilon}},\right.$$

$$\left. y2 \to \frac{-b\, y1 + \sqrt{5}\, \sqrt{b\, (b\, y1^2 + 4\, \epsilon)}}{2\, b}\right\},$$

$$\left\{x2 \to \frac{3\,b\,y1^2 + 12\,\epsilon + \sqrt{5}\;y1\,\sqrt{b\,\left(b\,y1^2 + 4\,\epsilon\right)}}{2\,\sqrt{b\,y1^2 + 4\,\epsilon}},\right.$$

$$\left. y2 \to \frac{3\,b\,y1 + \sqrt{5}\;\sqrt{b\,\left(b\,y1^2 + 4\,\epsilon\right)}}{2\,b}\right\},$$

$$\left\{x2 \to \frac{b\,y1^2 + 4\,\epsilon + \sqrt{5}\;y1\,\sqrt{b\,\left(b\,y1^2 + 4\,\epsilon\right)}}{2\,\sqrt{b\,y1^2 + 4\,\epsilon}},\right.$$

$$\left. y2 \to \frac{b\,y1 + \sqrt{5}\;\sqrt{b\,\left(b\,y1^2 + 4\,\epsilon\right)}}{2\,b}\right\},$$

$$\left\{x2 \to \frac{b\,y1^2 + 4\,\epsilon - \sqrt{5}\;y1\,\sqrt{b\,\left(b\,y1^2 + 4\,\epsilon\right)}}{2\,\sqrt{b\,y1^2 + 4\,\epsilon}},\right.$$

$$\left. y2 \to \frac{b\,y1 - \sqrt{5}\;\sqrt{b\,\left(b\,y1^2 + 4\,\epsilon\right)}}{2\,b}\right\},$$

$$\left\{x2 \to -\frac{b\,y1^2 + 4\,\epsilon + \sqrt{5}\;y1\,\sqrt{b\,\left(b\,y1^2 + 4\,\epsilon\right)}}{2\,\sqrt{b\,y1^2 + 4\,\epsilon}},\right.$$

$$\left. y2 \to -\frac{b\,y1 + \sqrt{5}\;\sqrt{b\,\left(b\,y1^2 + 4\,\epsilon\right)}}{2\,b}\right\},$$

$$\left\{x2 \to \frac{3\,b\,y1^2 + 12\,\epsilon - \sqrt{5}\;y1\,\sqrt{b\,\left(b\,y1^2 + 4\,\epsilon\right)}}{2\,\sqrt{b\,y1^2 + 4\,\epsilon}},\right.$$

$$\left. y2 \to \frac{3\,b\,y1 - \sqrt{5}\;\sqrt{b\,\left(b\,y1^2 + 4\,\epsilon\right)}}{2\,b}\right\}\}$$

Six one-parameter families of triangles

```
In[18]:= tOpts = t /. p1Rule /. p2Rule;
```

All having the same area:

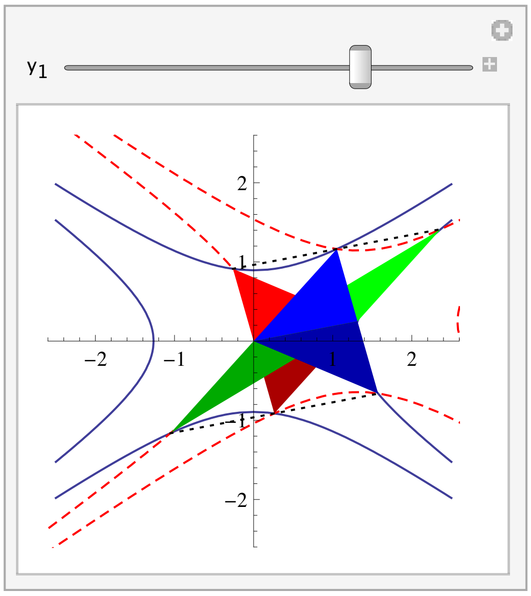
```
In[19]:= 1 / 2 Abs [Det [#[[2 ;; 3]]]] & /@ tOpts //
           FullSimplify // DeleteDuplicates
```

Out[19]= $\left\{\frac{\sqrt{5}\ \epsilon}{\sqrt{b}}\right\}$

Visualize the six one-parameter families of triangles

```
In[20]:= Manipulate[
          Block[{ϵ = 0.4, y1 = y10, b = 2, bnd = 2.5},
           Show[
            ContourPlot[Abs[F[x, y]] == 4 ϵ, {x, -bnd, bnd},
             {y, -bnd, bnd}],
            ContourPlot[
             Abs[F[{x, y} - ({x1, y1} /. p1Rule)]] == 4 ϵ,
             {x, -5, 5}, {y, -4, 4},
             ContourStyle -> {Red, Dashed}],
            Graphics[MapThread[List,
              {colors, Polygon[#] & /@ tOpts}]],
            Graphics[
             {Dotted,
              Line[{tOpts[[1]][[3]], tOpts[[3]][[3]],
                 tOpts[[2]][[3]]}]}],
            Graphics[
             {Dotted,
              Line[{tOpts[[5]][[3]], tOpts[[4]][[3]],
                 tOpts[[6]][[3]]}]}],
            (*Locus of the mid points of the base of
             the green triangle*)
            ParametricPlot[
             {(5 b y^2 + 20 ϵ + Sqrt[5] y Sqrt[b (b y^2 + 4 ϵ)]) /
                 (4 Sqrt[b y^2 + 4 ϵ] 5 y) / 4 +
               1 / 4 Sqrt[5] Sqrt[y^2 + (4 ϵ) / b]}, {y, -3, 3},
             PlotStyle -> {Dotted, Magenta}],
            Axes -> True, Frame -> False, ImageSize → 200
           ]],
          {{y10, 0.25, "y1"}, -2, 1}
         ]
```

Out[20]=



Verify collinearity of the vertices connected with dashed lines in the image above

```
In[21]:= {Det[{Append[tOpts[[1]][[3]], 1],
              Append[tOpts[[2]][[3]], 1],
              Append[tOpts[[3]][[3]], 1]}],
          Det[{Append[tOpts[[4]][[3]], 1],
              Append[tOpts[[5]][[3]], 1],
              Append[tOpts[[6]][[3]], 1]}]} // FullSimplify

Out[21]= {0, 0}
```

Study a single one-parameter family of triangles

Select a single one-parameter family of triangles (which has two vertices on the same hyperbola)

```
In[22]:= tCandidate = tOpts[[2]]
```

Out[22]= $\left\{\{0,0\},\left\{\sqrt{b\,y1^2+4\,\epsilon},\,y1\right\},\left\{\frac{3\,b\,y1^2+12\,\epsilon+\sqrt{5}\,y1\,\sqrt{b\,(b\,y1^2+4\,\epsilon)}}{2\,\sqrt{b\,y1^2+4\,\epsilon}},\frac{3\,b\,y1+\sqrt{5}\,\sqrt{b\,(b\,y1^2+4\,\epsilon)}}{2\,b}\right\}\right\}$

Find for which value of $y_1$the selected triangle is symmetric with respect to the x-axis

```
In[23]:= tSymmetryRule =
          Simplify[
            Solve[
             tCandidate[[2]][[2]] == -tCandidate[[3]][[2]] &&
              tCandidate[[2]][[1]] == tCandidate[[3]][[1]],
             y1, Reals]] // Flatten
```

Out[23]= $\left\{y1\to-\sqrt{\frac{\epsilon}{b}}\right\}$

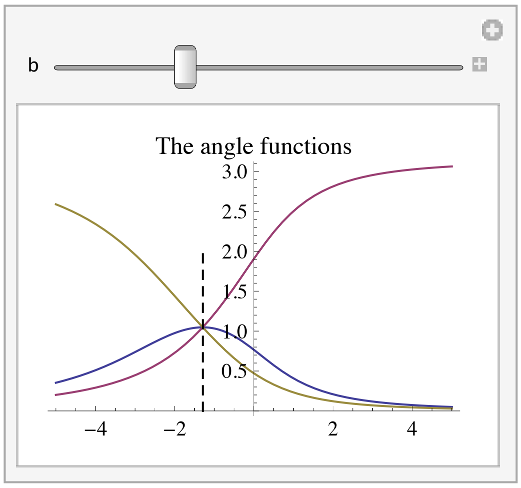
In order to optimize its shape we next study the angles of the selected triangle

```
In[24]:= tCandidateEdges =
           tCandidate - RotateLeft[tCandidate];
         tCandidateCosAngles = {
             Dot[-tCandidateEdges[[1]], tCandidateEdges[[3]]]
              (Norm[tCandidateEdges[[1]]]
                Norm[tCandidateEdges[[3]]]),
             Dot[tCandidateEdges[[1]],
               -tCandidateEdges[[2]]] /
              (Norm[tCandidateEdges[[1]]]
                Norm[tCandidateEdges[[2]]]),
             Dot[tCandidateEdges[[2]],
               -tCandidateEdges[[3]]] /
              (Norm[tCandidateEdges[[2]]]
                Norm[tCandidateEdges[[3]]])} //
            FullSimplify;
         tCandidateAngles = ArcCos[tCandidateCosAngles];
```

For fixed values of ϵ and b the angle functions are parameterized by y_1. We plot the angles depending on the value of b

```
In[27]:= Manipulate[
          Block[
           {ε = 1, functions = tCandidateAngles /. b → b0},
           Show[
            Plot[functions, {y1, -5, 5}],
            Graphics[
             {Dashed,
              Line[{{y1, 0}, {y1, 2}} /. tSymmetryRule /.
                 b → b0]}],
            PlotLabel → "The angle functions",
            ImageSize → 200
           ]
          ], {{b0, 3 / 5, "b"}, 0, 2}]
```

Out[27]=



Compute the behavior of the angle functions for $y_1 \to \pm\infty$.

```
In[28]:= ArcCos[Limit[#, y1 -> Infinity] & /@
            tCandidateCosAngles] // FullSimplify
         ArcCos[Limit[#, y1 -> -Infinity] & /@
            tCandidateCosAngles] // FullSimplify

Out[28]= {0, π, 0}

Out[29]= {0, 0, π}
```

Another approach, to establish the limits, is to consider the slopes of the edges of the candidate triangle

In[30]:= `tCandidateEdgeSlopes = Table[FullSimplify[tCandidateEdges[[i]][[2]] / tCandidateEdges[[i]][[1]]], {i, 3}]`

Out[30]=
$$\left\{\frac{y1}{\sqrt{b\,y1^2+4\,\epsilon}},\ \frac{b^2\,y1^3+4\,b\,y1\,\epsilon-\sqrt{5}\,\epsilon\,\sqrt{b\,(b\,y1^2+4\,\epsilon)}}{b\,(b\,y1^2-\epsilon)\,\sqrt{b\,y1^2+4\,\epsilon}},\ \frac{b^2\,y1^3+4\,b\,y1\,\epsilon+3\,\sqrt{5}\,\epsilon\,\sqrt{b\,(b\,y1^2+4\,\epsilon)}}{b\,\sqrt{b\,y1^2+4\,\epsilon}\,(b\,y1^2+9\,\epsilon)}\right\}$$

In[31]:=
```
Table[Limit[tCandidateEdgeSlopes[[i]],
  y1 → Infinity], {i, 3}]
Table[Limit[tCandidateEdgeSlopes[[i]],
  y1 → -Infinity], {i, 3}]
```

Out[31]=
$$\left\{\frac{1}{\sqrt{b}},\ \frac{1}{\sqrt{b}},\ \frac{1}{\sqrt{b}}\right\}$$

Out[32]=
$$\left\{-\frac{1}{\sqrt{b}},\ -\frac{1}{\sqrt{b}},\ -\frac{1}{\sqrt{b}}\right\}$$

Verify that the first angle function (colored in BLUE) has a global maximum.

In[33]:= `D[tCandidateAngles[[1]], y1] // FullSimplify`

Out[33]=
$$-\left(4\,(1+b)\,\epsilon\,\left(5\,b\,y1^2+10\,\epsilon+3\,\sqrt{5}\,y1\,\sqrt{b\,(b\,y1^2+4\,\epsilon)}\right)\right)\Big/\left(\sqrt{b\,y1^2+4\,\epsilon}\,\left((1+b)\,y1^2+4\,\epsilon\right)\left(7\,b^2\,y1^2+10\,\epsilon+3\,\sqrt{5}\,y1\,\sqrt{b\,(b\,y1^2+4\,\epsilon)}+b\left(7\,y1^2+18\,\epsilon+3\,\sqrt{5}\,y1\,\sqrt{b\,(b\,y1^2+4\,\epsilon)}\right)\right)\right)$$

```
In[34]:= D[tCandidateAngles[[1]], y1] // FullSimplify
         FullSimplify[Solve[% == 0, y1, Reals]]
         FullSimplify[D[tCandidateAngles[[1]], {y1, 2}] /.
           %]
```

Out[34]=
$$-\left(4\ (1+b)\ \epsilon\left(5\,b\,y1^2+10\,\epsilon+3\sqrt{5}\ y1\sqrt{b\left(b\,y1^2+4\,\epsilon\right)}\right)\right)\Big/\left(\sqrt{b\,y1^2+4\,\epsilon}\ \left((1+b)\ y1^2+4\,\epsilon\right)\left(7\,b^2\,y1^2+10\,\epsilon+3\sqrt{5}\ y1\sqrt{b\left(b\,y1^2+4\,\epsilon\right)}+b\left(7\,y1^2+18\,\epsilon+3\sqrt{5}\ y1\sqrt{b\left(b\,y1^2+4\,\epsilon\right)}\right)\right)\right)$$

Out[35]=
$$\left\{\left\{y1\to-\sqrt{\frac{\epsilon}{b}}\right\}\right\}$$

Out[36]=
$$\left\{-\left(16\,b\ (1+b)\left(\sqrt{b\,\epsilon}+20\sqrt{b^3\,\epsilon}+150\sqrt{b^5\,\epsilon}+500\sqrt{b^7\,\epsilon}+625\sqrt{b^9\,\epsilon}\right)\right)\Big/\left(\sqrt{5}\ (1+5\,b)^6\,\epsilon^{3/2}\right)\right\}$$

Indeed, for $y_1 = -\sqrt{\frac{\epsilon}{b}}$ the derivative vanishes and the second order derivative is *negative*. Thus, the first angle function has a global maximum at $y_1 = -\sqrt{\frac{\epsilon}{b}}$.

Next, we verify that the second angle function is monotonically increasing:

```
In[37]:= D[tCandidateAngles[[2]], y1] // FullSimplify;
         Reduce[% > 0, y1] // FullSimplify

Out[38]= True
```

Finally, we verify that the third angle function is monotonically decreasing. For the sake of simplicity of the computation, we consider the third angle as the complement of the first two.

```
In[39]:= D[π - tCandidateAngles[[1]] - tCandidateAngles[[2]],
            y1] // FullSimplify;
         Simplify[
          Reduce[% < 0 && b > 0 && x1 ∈ Reals && y1 ∈ Reals && ϵ > 0,
           y1]]

Out[40]= True
```

It is now clear the the symmetric triangle is of optimal shape, as its minimal angle is maximized in the chosen one-parameter family.

```
In[41]:= tSymmetric = tCandidate /. tSymmetryRule //
           FullSimplify
```

Out[41]= $\left\{\{0, 0\}, \left\{\sqrt{5}\,\sqrt{\epsilon}, -\sqrt{\frac{\epsilon}{b}}\right\}, \left\{\sqrt{5}\,\sqrt{\epsilon}, \sqrt{\frac{\epsilon}{b}}\right\}\right\}$

When is it equilateral?

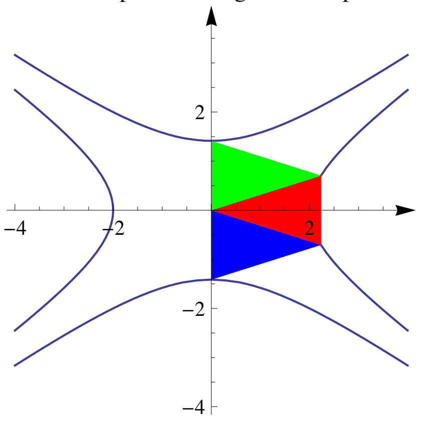
```
In[42]:= Solve[b > 0 && Norm[tSymmetric[[2]]] ==
              Norm[tSymmetric[[2]] - tSymmetric[[3]]], b] //
           FullSimplify // Flatten
```

Out[42]= $\left\{b \to \frac{3}{5}\right\}$

In[43]:=
```
Block[{ϵ = 1, x1 = 1, b = 2},
 Show[
  ContourPlot[Abs[F[x, y]] == 4 ϵ, {x, -4, 4},
   {y, -4, 4}],
  Graphics[{Red, Polygon[tSymmetric]}],
  Graphics[{Blue, Polygon[
      {{0, 0}, tSymmetric[[2]],
       tSymmetric[[3]] + tSymmetric[[1]] -
        tSymmetric[[3]] + tSymmetric[[2]] -
        tSymmetric[[3]]}
    ]}],
  Graphics[{Green, Polygon[
      {{0, 0}, tSymmetric[[3]],
       tSymmetric[[2]] + tSymmetric[[1]] -
        tSymmetric[[2]] + tSymmetric[[3]] -
        tSymmetric[[2]]}
    ]}],
  Frame -> False, Axes -> True,
  AxesStyle -> Arrowheads[0.05],
  PlotLabel →
   "Use the optimal triangles in the plane",
  ImageSize → 200
 ]]
```

Out[43]=

Use the optimal triangles in the plane



Verify

Finally, let us verify that when lifting the triangles to the saddle surface, they indeed maintain a maximal vertical distance which equals ϵ

```
In[44]:= tSymmetricLifted =
          Append[#, F[#[[1]], #[[2]]]] & /@ tSymmetric //
           FullSimplify
```

Out[44]= $\left\{\{0,0,0\},\left\{\sqrt{5}\sqrt{\epsilon},-\sqrt{\frac{\epsilon}{b}},4\epsilon\right\},\left\{\sqrt{5}\sqrt{\epsilon},\sqrt{\frac{\epsilon}{b}},4\epsilon\right\}\right\}$

Parameterize the edges of the lifted triangle

```
In[45]:= tSymmetricLiftedEdges =
           (1 - λ) tSymmetricLifted +
             λ RotateLeft[tSymmetricLifted] // FullSimplify
```

Out[45]= $\left\{\left\{\sqrt{5}\sqrt{\epsilon}\,\lambda,-\sqrt{\frac{\epsilon}{b}}\,\lambda,4\epsilon\lambda\right\},\right.$
$\left\{\sqrt{5}\sqrt{\epsilon},\sqrt{\frac{\epsilon}{b}}\,(-1+2\lambda),4\epsilon\right\},$
$\left.\left\{-\sqrt{5}\sqrt{\epsilon}\,(-1+\lambda),-\sqrt{\frac{\epsilon}{b}}\,(-1+\lambda),-4\epsilon\,(-1+\lambda)\right\}\right\}$

Define the vertical distance functions from the triangle’s edges to the saddle surface

```
In[46]:= tSymmetricLiftedVdistsFuncs =
          F[#[[1]], #[[2]]] - #[[3]] & /@
            tSymmetricLiftedEdges // FullSimplify
```

Out[46]= $\{4\epsilon\,(-1+\lambda)\,\lambda,-4\epsilon\,(-1+\lambda)\,\lambda,4\epsilon\,(-1+\lambda)\,\lambda\}$

Find the critical value of the vertical distance functions

```
In[47]:= Solve[D[tSymmetricLiftedVdistsFuncs, λ] == 0,
          λ]
```

Out[47]= $\left\{\left\{\lambda\to\frac{1}{2}\right\}\right\}$

Evaluate the vertical distance (up to +/- sign)

```
In[48]:= tSymmetricLiftedVdistsFuncs /. λ -> 1 / 2 //
          FullSimplify
```

Out[48]= $\{-\epsilon,\epsilon,-\epsilon\}$

Blue case:

Parameterize the vertices and obtain parameterized planar triangles

Like before only this time p_1 will be located on the conjugate hyperbola

```
In[49]:= ClearAll[conjP1Rule];
         conjP1Rule = Solve[1 / 4 F[t[[2]]] == -ϵ, y1]
         conjP1Rule = conjP1Rule[[2]];
```

Out[50]= $\left\{\left\{y1 \to -\frac{\sqrt{x1^2 + 4\,\epsilon}}{\sqrt{b}}\right\}, \left\{y1 \to \frac{\sqrt{x1^2 + 4\,\epsilon}}{\sqrt{b}}\right\}\right\}$

In turn, find the functions for the coordinates of p_2

```
In[52]:= ClearAll[conjP2Rule];
         conjP2Rule =
          Solve[1 / 4 Abs[F[t[[3]]]] == ϵ &&
             1 / 4 Abs[F[t[[3]] - t[[2]]]] == ϵ, {x2, y2}] //
           Simplify
         conjP2Rule =
          Assuming[x1 > 0,
           FullSimplify[conjP2Rule[[3 ;; 8]]]];
         (*Verify in the output of the previous line
          which entries are real!*)
```

Out[53]= $\Big\{\Big\{x2 \to \Big(x1^4 - b\, x1^2\, y1^2 + y1 \sqrt{b\,(x1^2 - b\, y1^2)\,(x1^2 - b\, y1^2 + 16\,\epsilon)}\; \text{Abs}[x1]\Big) \Big/ \Big(2\,(x1^3 - b\, x1\, y1^2)\Big),\ y2 \to \frac{x1^2\, y1 - b\, y1^3 + \sqrt{\frac{(x1^2 - b\, y1^2)\,(x1^2 - b\, y1^2 + 16\,\epsilon)}{b}}\; \text{Abs}[x1]}{2\,(x1^2 - b\, y1^2)}\Big\},$

$\Big\{x2 \to \Big(x1^4 - b\, x1^2\, y1^2 - y1 \sqrt{b\,(x1^2 - b\, y1^2)\,(x1^2 - b\, y1^2 + 16\,\epsilon)}\; \text{Abs}[x1]\Big) \Big/ \Big(2\,(x1^3 - b\, x1\, y1^2)\Big),\ y2 \to$

$$-\frac{-\mathrm{x1}^2\,\mathrm{y1}+\mathrm{b}\,\mathrm{y1}^3+\sqrt{\frac{(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2)\,(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2+16\,\epsilon)}{\mathrm{b}}}\ \mathrm{Abs}[\mathrm{x1}]}{2\,(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2)}\Big\}$$

$$,\ \Big\{\mathrm{x2}\to\Big(\mathrm{x1}^4-\mathrm{b}\,\mathrm{x1}^2\,\mathrm{y1}^2+$$
$$\mathrm{y1}\,\sqrt{\mathrm{b}\,(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2)\,(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2-16\,\epsilon)}\ \mathrm{Abs}[\mathrm{x1}]\Big)\Big/$$
$$\left(2\,(\mathrm{x1}^3-\mathrm{b}\,\mathrm{x1}\,\mathrm{y1}^2)\right),\ \mathrm{y2}\to$$
$$\frac{\mathrm{x1}^2\,\mathrm{y1}-\mathrm{b}\,\mathrm{y1}^3+\sqrt{\frac{(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2)\,(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2-16\,\epsilon)}{\mathrm{b}}}\ \mathrm{Abs}[\mathrm{x1}]}{2\,(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2)}\Big\},$$

$$\Big\{\mathrm{x2}\to\Big(\mathrm{x1}^4-\mathrm{b}\,\mathrm{x1}^2\,\mathrm{y1}^2-$$
$$\mathrm{y1}\,\sqrt{\mathrm{b}\,(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2)\,(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2-16\,\epsilon)}\ \mathrm{Abs}[\mathrm{x1}]\Big)\Big/$$
$$\left(2\,(\mathrm{x1}^3-\mathrm{b}\,\mathrm{x1}\,\mathrm{y1}^2)\right),\ \mathrm{y2}\to$$
$$-\frac{-\mathrm{x1}^2\,\mathrm{y1}+\mathrm{b}\,\mathrm{y1}^3+\sqrt{\frac{(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2)\,(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2-16\,\epsilon)}{\mathrm{b}}}\ \mathrm{Abs}[\mathrm{x1}]}{2\,(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2)}\Big\}$$

$$,\ \Big\{\mathrm{x2}\to\Big(\mathrm{x1}^4-\mathrm{b}\,\mathrm{x1}^2\,\mathrm{y1}^2+8\,\mathrm{x1}^2\,\epsilon+\mathrm{y1}$$
$$\sqrt{\mathrm{b}\,(\mathrm{x1}^4-2\,\mathrm{b}\,\mathrm{x1}^2\,\mathrm{y1}^2+\mathrm{b}^2\,\mathrm{y1}^4+64\,\epsilon^2)}\ \mathrm{Abs}[\mathrm{x1}]\Big)\Big/$$
$$\left(2\,\mathrm{x1}^3-2\,\mathrm{b}\,\mathrm{x1}\,\mathrm{y1}^2\right),\ \mathrm{y2}\to\Big(\mathrm{b}\,\mathrm{y1}\,(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2+8\,\epsilon)+$$
$$\sqrt{\mathrm{b}\,(\mathrm{x1}^4-2\,\mathrm{b}\,\mathrm{x1}^2\,\mathrm{y1}^2+\mathrm{b}^2\,\mathrm{y1}^4+64\,\epsilon^2)}\ \mathrm{Abs}[\mathrm{x1}]\Big)\Big/$$
$$\left(2\,\mathrm{b}\,(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2)\right)\Big\},$$

$$\Big\{\mathrm{x2}\to\Big(\mathrm{x1}^4-\mathrm{b}\,\mathrm{x1}^2\,\mathrm{y1}^2-8\,\mathrm{x1}^2\,\epsilon+\mathrm{y1}$$
$$\sqrt{\mathrm{b}\,(\mathrm{x1}^4-2\,\mathrm{b}\,\mathrm{x1}^2\,\mathrm{y1}^2+\mathrm{b}^2\,\mathrm{y1}^4+64\,\epsilon^2)}\ \mathrm{Abs}[\mathrm{x1}]\Big)\Big/$$
$$\left(2\,\mathrm{x1}^3-2\,\mathrm{b}\,\mathrm{x1}\,\mathrm{y1}^2\right),\ \mathrm{y2}\to\Big(\mathrm{b}\,\mathrm{y1}\,(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2-8\,\epsilon)+$$
$$\sqrt{\mathrm{b}\,(\mathrm{x1}^4-2\,\mathrm{b}\,\mathrm{x1}^2\,\mathrm{y1}^2+\mathrm{b}^2\,\mathrm{y1}^4+64\,\epsilon^2)}\ \mathrm{Abs}[\mathrm{x1}]\Big)\Big/$$
$$\left(2\,\mathrm{b}\,(\mathrm{x1}^2-\mathrm{b}\,\mathrm{y1}^2)\right)\Big\},$$

$$\Big\{\mathrm{x2}\to\Big(\mathrm{x1}^4-\mathrm{b}\,\mathrm{x1}^2\,\mathrm{y1}^2+8\,\mathrm{x1}^2\,\epsilon-\mathrm{y1}$$
$$\sqrt{\mathrm{b}\,(\mathrm{x1}^4-2\,\mathrm{b}\,\mathrm{x1}^2\,\mathrm{y1}^2+\mathrm{b}^2\,\mathrm{y1}^4+64\,\epsilon^2)}\ \mathrm{Abs}[\mathrm{x1}]\Big)\Big/$$

```
    (2 x1^3 - 2 b x1 y1^2), y2 → (b y1 (x1^2 - b y1^2 + 8 ∈) -
        Sqrt[b (x1^4 - 2 b x1^2 y1^2 + b^2 y1^4 + 64 ∈^2)] Abs[x1]) /
    (2 b (x1^2 - b y1^2))},
  {x2 → (x1^4 - b x1^2 y1^2 - 8 x1^2 ∈ - y1
        Sqrt[b (x1^4 - 2 b x1^2 y1^2 + b^2 y1^4 + 64 ∈^2)] Abs[x1]) /
    (2 x1^3 - 2 b x1 y1^2), y2 → (b y1 (-x1^2 + b y1^2 + 8 ∈) +
        Sqrt[b (x1^4 - 2 b x1^2 y1^2 + b^2 y1^4 + 64 ∈^2)] Abs[x1]) /
    (2 b (-x1^2 + b y1^2))}}
```

Given the functions of the coordinates of p_1 and p_2 we can obtain, again, six one-parameter families of triangles

```
In[55]:= ClearAll[conjTOpts];
         conjTOpts = t /. conjP2Rule /. conjP1Rule //
            FullSimplify;
```

All have the same area (which in turn equals to the area of the triangles in the RED case)

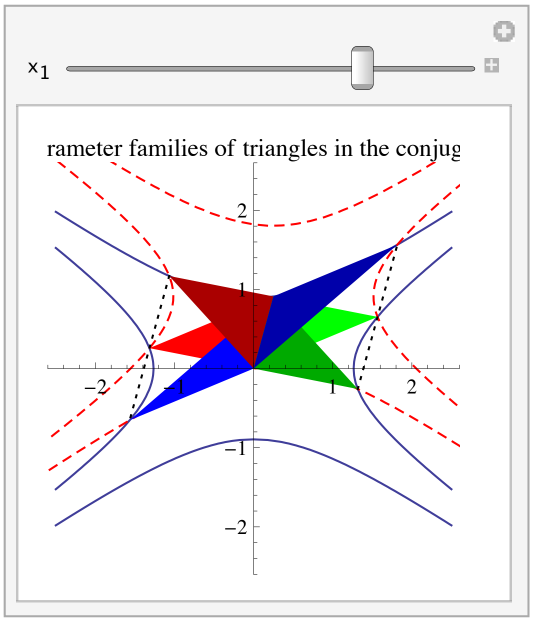
```
In[57]:= 1 / 2 Abs[Det[#[[2 ;; 3]]]] & /@ conjTOpts //
           FullSimplify // DeleteDuplicates
```

Out[57]= $\left\{\frac{\sqrt{5}\,\epsilon}{\sqrt{b}}\right\}$

Visualize generic triangles

```
In[58]:= Manipulate[
          Block[{ϵ = 0.4, x1 = x10, b = 2, bnd = 2.5},
           Show[
            ContourPlot[Abs[F[x, y]] == 4 ϵ, {x, -bnd, bnd},
             {y, -bnd, bnd}],
            ContourPlot[
             Abs[F[{x, y} - ({x1, y1} /. conjP1Rule)]] == 4 ϵ,
             {x, -5, 5}, {y, -4, 4},
             ContourStyle -> {Red, Dashed}],
            Graphics[MapThread[List,
              {colors, Polygon[#] & /@ conjTOpts}]],
            (*Graphics[Polygon[tOpts[[2]]]],*)
            Graphics[
             {Dotted,
              Line[{conjTOpts[[5]][[3]],
                conjTOpts[[2]][[3]],
                conjTOpts[[6]][[3]]}]}],
            Graphics[
             {Dotted,
              Line[{conjTOpts[[4]][[3]],
                conjTOpts[[1]][[3]],
                conjTOpts[[3]][[3]]}]}],
            (*Locus of the mid points of the base of
             the green triangle*)
            ParametricPlot[
             {(5 b y^2 + 20 ϵ + Sqrt[5] y Sqrt[b (b y^2 + 4 ϵ)]) /
                 (4 Sqrt[b y^2 + 4 ϵ] 5 y) / 4 +
               1 / 4 Sqrt[5] Sqrt[y^2 + (4 ϵ) / b]}, {y, -3, 3},
             PlotStyle -> {Dotted, Magenta}],
            PlotLabel →
             "One-parameter families of triangles in
               the conjugate case",
            Axes -> True, Frame -> False, ImageSize → 200
           ]],
          {{x10, 0.25, "x₁"}, -2, 1}
         ]
```

Out[58]=



Verify collinearity:

In[59]:=
```
{Det[{Append[conjTOpts[[1]][[3]], 1],
     Append[conjTOpts[[4]][[3]], 1],
     Append[conjTOpts[[3]][[3]], 1]}],
 Det[{Append[conjTOpts[[6]][[3]], 1],
     Append[conjTOpts[[2]][[3]], 1],
     Append[conjTOpts[[5]][[3]], 1]}]} //
  FullSimplify
```

Out[59]=
```
{0, 0}
```

Study a single one-parameter family of triangles

Let us pick the red (4th) one-parameter family of triangles

```
In[60]:= ClearAll[conjTCandidate];
         conjTCandidate = conjTOpts[[4]]
```

Out[61]=
$$\left\{\{0,0\},\left\{x1,\sqrt{\frac{x1^2+4\,\epsilon}{b}}\right\},\left\{\frac{1}{2}\left(3\,x1-\sqrt{5}\,\sqrt{x1^2+4\,\epsilon}\right),-\frac{\sqrt{5}\,\sqrt{b}\,x1-3\,\sqrt{b\,\left(x1^2+4\,\epsilon\right)}}{2\,b}\right\}\right\}$$

and determine for which value of $x_1$is it symmetric with respect to the y-axis

```
In[62]:= ClearAll[conjTSymmetryRule];
         conjTSymmetryRule =
          Simplify[
            Solve[
             conjTCandidate[[2]][[2]] ==
               conjTCandidate[[3]][[2]] &&
              conjTCandidate[[2]][[1]] ==
               -conjTCandidate[[3]][[1]], x1, Reals]] //
           Flatten
```

Out[63]= $\left\{x1\to\sqrt{\epsilon}\right\}$

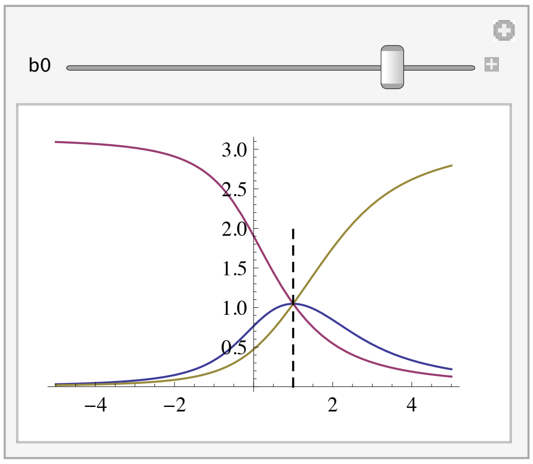
Again, in order to find the family member of optimal shape we have to consider the angles of the triangles in the family:

```
In[64]:= conjTCandidateEdges =
           conjTCandidate - RotateLeft[conjTCandidate];
         conjTCandidateCosAngles = {
             Dot[-conjTCandidateEdges[[1]],
               conjTCandidateEdges[[3]]] /
              (Norm[conjTCandidateEdges[[1]]]
                Norm[conjTCandidateEdges[[3]]]),
            Dot[conjTCandidateEdges[[1]],
               -conjTCandidateEdges[[2]]] /
              (Norm[conjTCandidateEdges[[1]]]
                Norm[conjTCandidateEdges[[2]]]),
             Dot[conjTCandidateEdges[[2]],
               -conjTCandidateEdges[[3]]] /
              (Norm[conjTCandidateEdges[[2]]]
                Norm[conjTCandidateEdges[[3]]])} //
            FullSimplify;
         conjTCandidateAngles =
           ArcCos[conjTCandidateCosAngles];
```

The angle are functions of $x_1$and are parameterized by b

```
In[67]:= Manipulate[
           Block[
            {ϵ = 1, functions =
               conjTCandidateAngles /. b → b0},
            Show[
             Plot[functions, {x1, -5, 5}],
             Graphics[
              {Dashed,
               Line[{{x1, 0}, {x1, 2}} /. conjTSymmetryRule /.
                  b → b0]}],
             ImageSize → 200
            ]
           ], {{b0, 5 / 3}, 0, 2}]
```

Out[67]=



Study the behavior of the angle functions at +∞ and -∞

```
In[68]:= ArcCos[Limit[#, x1 -> Infinity] & /@
             conjTCandidateCosAngles] // FullSimplify
         ArcCos[Limit[#, x1 -> -Infinity] & /@
             conjTCandidateCosAngles] // FullSimplify

Out[68]= {0, 0, π}

Out[69]= {0, π, 0}
```

Verify the global maximum of the first angle function

```
In[70]:= D[conjTCandidateAngles[[1]], x1] // FullSimplify;
         Solve[% == 0, x1, Reals] // FullSimplify
         D[conjTCandidateAngles[[1]], {x1, 2}] /. % //
          FullSimplify
```

Out[71]= $\{\{x1 \to \sqrt{\epsilon}\}\}$

Out[72]= $\left\{-\frac{16\sqrt{b}\,(1+b)}{\sqrt{5}\,(5+b)^2\,\epsilon}\right\}$

Next, verify that the second function is monotonically decreasing

```
In[73]:= D[conjTCandidateAngles[[2]], x1] // FullSimplify;
         Reduce[% < 0, x1] // FullSimplify
```

```
Out[74]= True
```

And finally, verify that the third angle function is monotonically increasing

```
In[75]:= D[π - conjTCandidateAngles[[1]] -
              conjTCandidateAngles[[2]], x1] // Simplify;
         Simplify[Reduce[% > 0 && b > 0 && ϵ > 0, x1]]
```

```
Out[76]= True
```

We can now set the symmetric triangle

```
In[77]:= conjTSymmetric =
          conjTCandidate /. conjTSymmetryRule // FullSimplify
```

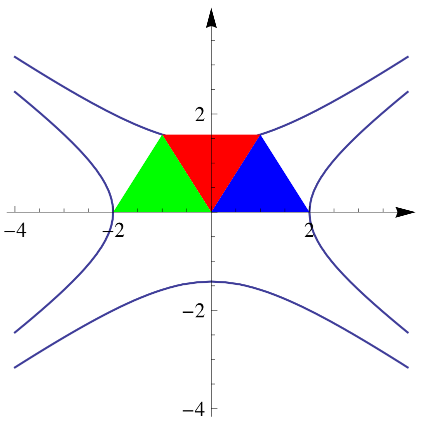
Out[77]= $\left\{\{0, 0\}, \left\{\sqrt{\epsilon}, \sqrt{5}\sqrt{\frac{\epsilon}{b}}\right\}, \left\{-\sqrt{\epsilon}, \frac{\sqrt{5}\,\epsilon}{\sqrt{b\,\epsilon}}\right\}\right\}$

and find when is it equilateral?

```
In[78]:= Reduce[
          Norm[conjTSymmetric[[2]]] ==
            Norm[conjTSymmetric[[2]] -
              conjTSymmetric[[3]]] && ϵ > 0 && b > 0, b]
```

Out[78]= $\epsilon > 0 \;\&\&\; b == \frac{5}{3}$

In[79]:=
```
Block[{ϵ = 1, x1 = 1, b = 2},
 Show[
  ContourPlot[Abs[F[x, y]] == 4 ϵ, {x, -4, 4},
   {y, -4, 4}],
  Graphics[{Red, Polygon[conjTSymmetric]}],
  Graphics[{Blue, Polygon[
     {{0, 0}, conjTSymmetric[[2]],
      conjTSymmetric[[3]] + conjTSymmetric[[1]] -
       conjTSymmetric[[3]] + conjTSymmetric[[2]] -
       conjTSymmetric[[3]]}
    ]}],
  Graphics[{Green, Polygon[
     {{0, 0}, conjTSymmetric[[3]],
      conjTSymmetric[[2]] + conjTSymmetric[[1]] -
       conjTSymmetric[[2]] + conjTSymmetric[[3]] -
       conjTSymmetric[[2]]}
    ]}],
  Frame -> False, Axes -> True,
  AxesStyle -> Arrowheads[0.05], ImageSize → 200
 ]]
```

Out[79]=


Verify

Like before, verify that the lifting of the symmetric triangle yields an ϵ maximal vertical distance

```
In[80]:= conjTSymmetricLifted =
          Append[#, F[#[[1]], #[[2]]]] & /@ conjTSymmetric //
           FullSimplify
```

Out[80]= $\left\{\{0, 0, 0\}, \left\{\sqrt{\epsilon}, \sqrt{5}\sqrt{\frac{\epsilon}{b}}, -4\epsilon\right\}, \left\{-\sqrt{\epsilon}, \frac{\sqrt{5}\,\epsilon}{\sqrt{b\,\epsilon}}, -4\epsilon\right\}\right\}$

```
In[81]:= conjTSymmetricLiftedEdges =
          (1 - λ) conjTSymmetricLifted +
            λ RotateLeft[conjTSymmetricLifted] //
           FullSimplify
```

Out[81]= $\left\{\left\{\sqrt{\epsilon}\,\lambda, \sqrt{5}\sqrt{\frac{\epsilon}{b}}\,\lambda, -4\,\epsilon\,\lambda\right\}, \left\{\sqrt{\epsilon}\,(1-2\lambda), \frac{\sqrt{5}\,\epsilon}{\sqrt{b\,\epsilon}}, -4\epsilon\right\}, \left\{\sqrt{\epsilon}\,(-1+\lambda), -\frac{\sqrt{5}\,\epsilon\,(-1+\lambda)}{\sqrt{b\,\epsilon}}, 4\,\epsilon\,(-1+\lambda)\right\}\right\}$

```
In[82]:= conjTSymmetricLiftedVdistsFuncs =
          F[#[[1]], #[[2]]] - #[[3]] & /@
            conjTSymmetricLiftedEdges // FullSimplify
```

Out[82]= $\{-4\,\epsilon\,(-1+\lambda)\,\lambda, 4\,\epsilon\,(-1+\lambda)\,\lambda, -4\,\epsilon\,(-1+\lambda)\,\lambda\}$

```
In[83]:= Solve[D[conjTSymmetricLiftedVdistsFuncs, λ] == 0,
          λ]
```

Out[83]= $\left\{\left\{\lambda \to \frac{1}{2}\right\}\right\}$

```
In[84]:= conjTSymmetricLiftedVdistsFuncs /. λ -> 1 / 2 //
          FullSimplify
```

Out[84]= $\{\epsilon, -\epsilon, \epsilon\}$

Red or Blue?

Recall that the two triangles from the previous two sections are:

In[85]:=
```
tSymmetric
conjTSymmetric
```

Out[85]= $\left\{\{0, 0\}, \left\{\sqrt{5}\,\sqrt{\epsilon}, -\sqrt{\frac{\epsilon}{b}}\right\}, \left\{\sqrt{5}\,\sqrt{\epsilon}, \sqrt{\frac{\epsilon}{b}}\right\}\right\}$

Out[86]= $\left\{\{0, 0\}, \left\{\sqrt{\epsilon}, \sqrt{5}\,\sqrt{\frac{\epsilon}{b}}\right\}, \left\{-\sqrt{\epsilon}, \frac{\sqrt{5}\,\epsilon}{\sqrt{b\,\epsilon}}\right\}\right\}$

and their angles are given by

In[87]:=
```
tAngles = {
    ArcCos[Dot[tSymmetric[[2]] - tSymmetric[[1]],
        tSymmetric[[3]] - tSymmetric[[1]]] /
       (Norm[tSymmetric[[2]] - tSymmetric[[1]]]
         Norm[tSymmetric[[3]] - tSymmetric[[1]]])],
    ArcCos[
     Dot[tSymmetric[[1]] - tSymmetric[[2]],
        tSymmetric[[3]] - tSymmetric[[2]]] /
       (Norm[tSymmetric[[1]] - tSymmetric[[2]]]
         Norm[tSymmetric[[3]] - tSymmetric[[2]]])]
   } // Simplify
Limit[%, b → Infinity]
```

Out[87]= $\left\{\mathrm{ArcCos}\left[\frac{-1+5\,b}{1+5\,b}\right], \mathrm{ArcCos}\left[\frac{1}{\sqrt{1+5\,b}}\right]\right\}$

Out[88]= $\left\{0, \frac{\pi}{2}\right\}$

```
In[89]:= conjTAngles = {
             ArcCos[
              Dot[conjTSymmetric[[2]] - conjTSymmetric[[1]],
                 conjTSymmetric[[3]] - conjTSymmetric[[1]]] /
                (Norm[conjTSymmetric[[2]] -
                     conjTSymmetric[[1]]]
                  Norm[conjTSymmetric[[3]] -
                     conjTSymmetric[[1]]])],
             ArcCos[
              Dot[conjTSymmetric[[1]] - conjTSymmetric[[2]],
                 conjTSymmetric[[3]] - conjTSymmetric[[2]]] /
                (Norm[conjTSymmetric[[1]] -
                     conjTSymmetric[[2]]]
                  Norm[conjTSymmetric[[3]] -
                     conjTSymmetric[[2]]])]
           } // Simplify
         Limit[%, b → Infinity]
```

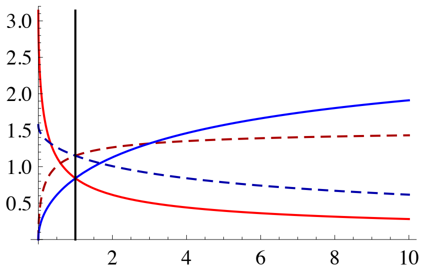
Out[89]= $\left\{\text{ArcCos}\left[\frac{5-b}{5+b}\right],\ \text{ArcCos}\left[\sqrt{\frac{b}{5+b}}\right]\right\}$

Out[90]= $\{\pi,\ 0\}$

as functions of b.

```
In[91]:= Show[
           Plot[#, {b, 0, 10},
               PlotStyle → {Red, {Darker[Red], Dashed},
                 Blue, {Darker[Blue], Dashed}}] &[
             Join[tAngles, conjTAngles]],
           Graphics[Line[{{1, 0}, {1, π}}]],
           ImageSize → 200
         ]
```

Out[91]=



First, verify that behavior of the functions.

```
In[92]:= Reduce[D[tAngles[[1]], b] < 0, b] // Simplify
         Reduce[D[tAngles[[2]], b] > 0, b] // Simplify
         Reduce[D[conjTAngles[[1]], b] > 0, b] // Simplify
         Reduce[D[conjTAngles[[2]], b] < 0, b] // Simplify

Out[92]= True

Out[93]= True

Out[94]= True

Out[95]= True
```

For 0≤b≤1 the first angle of the conjugate triangle is the minimal one:

```
In[96]:= Solve[Cos[tAngles[[1]]] == Cos[conjTAngles[[1]]] &&
           b ≥ 0, b]
         Solve[Cos[tAngles[[2]]] == Cos[conjTAngles[[1]]] &&
           b ≥ 0, b]
         Solve[
          Cos[conjTAngles[[2]]] == Cos[conjTAngles[[1]]] &&
           b ≥ 0, b]

Out[96]= {{b → 1}}

Out[97]= {{b → 0}, {b → 3}}
```

Out[98]= $\left\{\left\{b \to \frac{5}{3}\right\}\right\}$

Indeed, this function does not intersect any other function in the interior of the interior of the interval.
For b>1 the first angle function of the original triangle is the minimal.

Relation Between the Red and Blue

In[99]:=
```
T = tOpts[[3]]
conjT = conjTOpts[[5]]
```

Out[99]=
$$\left\{\{0,0\},\left\{\sqrt{b\,y1^2+4\,\epsilon},\ y1\right\},\left\{\frac{b\,y1^2+4\,\epsilon+\sqrt{5}\ y1\ \sqrt{b\left(b\,y1^2+4\,\epsilon\right)}}{2\sqrt{b\,y1^2+4\,\epsilon}},\ \frac{b\,y1+\sqrt{5}\ \sqrt{b\left(b\,y1^2+4\,\epsilon\right)}}{2\,b}\right\}\right\}$$

Out[100]=
$$\left\{\{0,0\},\left\{x1,\ \sqrt{\frac{x1^2+4\,\epsilon}{b}}\right\},\left\{\frac{1}{2}\left(-x1+\sqrt{5}\ \sqrt{x1^2+4\,\epsilon}\right),\ \frac{\sqrt{5}\ \sqrt{b}\ x1-\sqrt{b\left(x1^2+4\,\epsilon\right)}}{2\,b}\right\}\right\}$$

In[101]:=
```
redBlueRule =
  Solve[T[[3]] == conjT[[2]] && T[[2]] == conjT[[3]],
    y1][[1]][[1]]
```

Out[101]=
$$y1\rightarrow\frac{1}{2}\left(\frac{\sqrt{5}\ x1}{\sqrt{b}}-\frac{\sqrt{b\left(x1^2+4\,\epsilon\right)}}{b}\right)$$

In[102]:=
```
FullSimplify[(T[[2]] - conjT[[3]]) /. redBlueRule]
```

Out[102]=
$$\left\{\frac{1}{2}\left(x1-\sqrt{5}\ \sqrt{x1^2+4\,\epsilon}+\sqrt{6\,x1^2+20\,\epsilon-2\sqrt{5}\ x1\ \sqrt{x1^2+4\,\epsilon}}\right),\ 0\right\}$$

```
In[103]:= FullSimplify[(x1 - √5 √(x1² + 4 ϵ))² -
             (6 x1² + 20 ϵ - 2 √5 x1 √(x1² + 4 ϵ))]

Out[103]= 0
```

General Canonical Saddle

Here we show how to obtain an optimal triangulation of a generic canonical saddle from the one we obtained which corresponds to a saddle of the form $z = x^2 - \mathrm{by}^2$.

```
In[104]:= base2general[x_, y_, z_] := {x, y, α z};
          base2general[p_] :=
            base2general[p[[1]], p[[2]], p[[3]]] /;
             Length[p] == 3;
```

Let $S_G$be a generic canonical saddle given by $z = \alpha x^2 - \beta y^2$ and set $b = \frac{\beta}{\alpha}$. Then $\frac{z}{a} = x^2 - b\, y^2$ is a canonical saddle as we studied which is only scaled in the z-direction. Next, consider the symmetric triangle:

```
In[106]:= τBase = tSymmetric /. ϵ → ϵ/α /. b → β/α
```

$$\text{Out[106]=}\ \left\{\{0,0\},\left\{\sqrt{5}\sqrt{\frac{\epsilon}{\alpha}},-\sqrt{\frac{\epsilon}{\beta}}\right\},\left\{\sqrt{5}\sqrt{\frac{\epsilon}{\alpha}},\sqrt{\frac{\epsilon}{\beta}}\right\}\right\}$$

```
In[107]:= τBaseLifted = Append[#, #[[1]]² - β/α #[[2]]²] & /@ τBase
```

$$\text{Out[107]=}\ \left\{\{0,0,0\},\left\{\sqrt{5}\sqrt{\frac{\epsilon}{\alpha}},-\sqrt{\frac{\epsilon}{\beta}},\frac{4\,\epsilon}{\alpha}\right\},\left\{\sqrt{5}\sqrt{\frac{\epsilon}{\alpha}},\sqrt{\frac{\epsilon}{\beta}},\frac{4\,\epsilon}{\alpha}\right\}\right\}$$

```
In[108]:= τBaseLiftedEdges =
            (1 - λ) τBaseLifted + λ RotateLeft[τBaseLifted] //
             FullSimplify
```

Out[108]= $\left\{\left\{\sqrt{5}\sqrt{\frac{\epsilon}{\alpha}}\,\lambda,\ -\sqrt{\frac{\epsilon}{\beta}}\,\lambda,\ \frac{4\,\epsilon\,\lambda}{\alpha}\right\},\ \left\{\sqrt{5}\sqrt{\frac{\epsilon}{\alpha}},\ \sqrt{\frac{\epsilon}{\beta}}\,(-1+2\,\lambda),\ \frac{4\,\epsilon}{\alpha}\right\},\ \left\{-\sqrt{5}\sqrt{\frac{\epsilon}{\alpha}}\,(-1+\lambda),\ -\sqrt{\frac{\epsilon}{\beta}}\,(-1+\lambda),\ -\frac{4\,\epsilon\,(-1+\lambda)}{\alpha}\right\}\right\}$

In[109]:= τBaseLiftedDistFuncs =
  $\#[[1]]^2 - \frac{\beta}{\alpha}\,\#[[2]]^2 - \#[[3]]$ & /@ τBaseLiftedEdges //
   FullSimplify

Out[109]= $\left\{\frac{4\,\epsilon\,(-1+\lambda)\,\lambda}{\alpha},\ -\frac{4\,\epsilon\,(-1+\lambda)\,\lambda}{\alpha},\ \frac{4\,\epsilon\,(-1+\lambda)\,\lambda}{\alpha}\right\}$

```
In[110]:= Solve[D[τBaseLiftedDistFuncs, λ] == 0, λ]
```

Out[110]= $\left\{\left\{\lambda \to \frac{1}{2}\right\}\right\}$

```
In[111]:= τBaseLiftedDistFuncs /. λ → 1 / 2
```

Out[111]= $\left\{-\frac{\epsilon}{\alpha},\ \frac{\epsilon}{\alpha},\ -\frac{\epsilon}{\alpha}\right\}$

```
In[112]:= τGenericLifted = base2general[#] & /@ τBaseLifted
```

Out[112]= $\left\{\{0,\ 0,\ 0\},\ \left\{\sqrt{5}\sqrt{\frac{\epsilon}{\alpha}},\ -\sqrt{\frac{\epsilon}{\beta}},\ 4\,\epsilon\right\},\ \left\{\sqrt{5}\sqrt{\frac{\epsilon}{\alpha}},\ \sqrt{\frac{\epsilon}{\beta}},\ 4\,\epsilon\right\}\right\}$

In[113]:= $\alpha\,\#[[1]]^2 - \beta\,\#[[2]]^2 == \#[[3]]$ & /@ τGenericLifted

```
Out[113]= {True, True, True}
```

```
In[114]:= τGenericLiftedEdges =
            (1 - λ) τGenericLifted + λ RotateLeft[τGenericLifted]
```

Out[114]=
$$\left\{\left\{\sqrt{5}\sqrt{\frac{\epsilon}{\alpha}}\,\lambda,\ -\sqrt{\frac{\epsilon}{\beta}}\,\lambda,\ 4\,\epsilon\,\lambda\right\},\right.$$
$$\left\{\sqrt{5}\sqrt{\frac{\epsilon}{\alpha}}\,(1-\lambda)+\sqrt{5}\sqrt{\frac{\epsilon}{\alpha}}\,\lambda,\right.$$
$$\left.-\sqrt{\frac{\epsilon}{\beta}}\,(1-\lambda)+\sqrt{\frac{\epsilon}{\beta}}\,\lambda,\ 4\,\epsilon\,(1-\lambda)+4\,\epsilon\,\lambda\right\},$$
$$\left.\left\{\sqrt{5}\sqrt{\frac{\epsilon}{\alpha}}\,(1-\lambda),\ \sqrt{\frac{\epsilon}{\beta}}\,(1-\lambda),\ 4\,\epsilon\,(1-\lambda)\right\}\right\}$$

```
In[115]:= τGenericLiftedDistFuncs =
            α #[[1]]^2 - β #[[2]]^2 - #[[3]] & /@ τGenericLiftedEdges
```

Out[115]=
$$\left\{-4\,\epsilon\,\lambda+4\,\epsilon\,\lambda^2,\right.$$
$$-4\,\epsilon\,(1-\lambda)-4\,\epsilon\,\lambda+\alpha\left(\sqrt{5}\sqrt{\frac{\epsilon}{\alpha}}\,(1-\lambda)+\sqrt{5}\sqrt{\frac{\epsilon}{\alpha}}\,\lambda\right)^2-$$
$$\left.\beta\left(-\sqrt{\frac{\epsilon}{\beta}}\,(1-\lambda)+\sqrt{\frac{\epsilon}{\beta}}\,\lambda\right)^2,\ -4\,\epsilon\,(1-\lambda)+4\,\epsilon\,(1-\lambda)^2\right\}$$

```
In[116]:= Solve[D[τGenericLiftedDistFuncs, λ] == 0, λ]
```

Out[116]=
$$\left\{\left\{\lambda\to\frac{1}{2}\right\}\right\}$$

In[117]:= τGenericLiftedDistFuncs /. $\lambda\to\frac{1}{2}$

```
Out[117]= {-∈, ∈, -∈}
```

4 Revising Computation of Arrangement of Polylines

4.1 Background and Motivation

The Computational Geometry Algorithms Library (**CGAL**[1]) aims at providing easy access to efficient and reliable geometric algorithms in the from of a C++ library. One of the oldest packages shipped as part of **CGAL** is the *2D Arrangements* package. This package supports the robust construction and maintenance of arrangements of curves embedded on certain orientable two-dimensional parametric surfaces in three-dimensional space[45, 17, 5, 6], and robust operations on them, e.g., overlay computation.[2]

The implementation of the various algorithms that construct and manipulate arrangements is generic, as it is independent on the type of curves and embedding surface they handle. All steps of the algorithms are enabled by a minimal set of geometric primitives, such as comparing two points on the embedding surface in parametric lexicographic order and computing intersection points. Each set of primitives that handle a particular family of curves are gathered in a dedicated module called *geometry-traits*. Different geometry-traits modules are provided in the *2D Arrangements* package to handle various families of curves.

Before we continue let us introduce some definitions and notions that we use throughout the chapter. A parametric planar *curve* is a piecewise continuous map $C(t)\colon I \to \mathbb{R}^2$, where I is an interval. Without loss of generality, I can be close or open-close or open unit interval. Next, an n-tuple of curves $(s_i)_{i=1}^{n}$ is called a *polycurve*; see Figure 4.2. Each curve s_i is called a *segment*, and in particular, a polycurve is also a curve. The endpoints of the constituting segments (if they exist) are called the *vertices* of the polycurve. Note that both a curve and a polycurve can be disconnected. In addition, two different permutations of the segments yields two different polycurves that have the same (geometrical) image. An *unbounded*

[1]`www.cgal.org`

[2]Arrangements on surfaces are supported in **CGAL** as of version 3.4, albeit not documented yet.

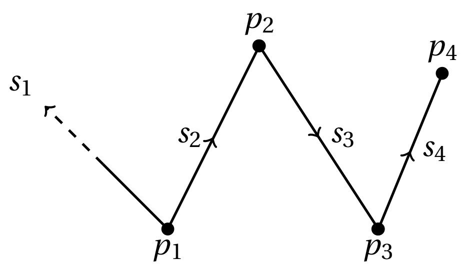


Figure 4.1: An unbounded (x-monotone) polyline $\mathscr{P}$ that consists of four segments and four vertices. In this example s_1 is a ray.

polyline is a polycurve such that all its segments are linear; namely, each s_i is either a line or a ray or a bounded (linear) segment.[3] Finally, if all the segments of an unbounded polyline are bounded (i.e. finite), then we call it a *bounded polyline*. In cases where there is no risk of confusion, we simply use the term polyline. A special kind of curves are the so-called x-monotone curves defined next.

Definition 4.1 (x-monotone curve)**.** In the context of the *2D Arrangements* package, a parametric curve $C(t) = (X(t), Y(t))$, where $X(t), Y(t) : I \to \mathbb{R}$, is *$x$-monotone* if the following hold

- $C(t)$ is continuous and injective.
- For $0 \le t_1 < t_2 \le 1$ we have that $C(t_1)$ is lexicographically smaller than $C(t_2)$.
- If $X(t_1) = X(t_2) = q$ for some $0 \le t_1 < t_2 \le 1$, then $X(t) = q$ for all $t \in I$. In this case we have that $C(t)$ is a *vertical* x-monotone curve.

In practice, the *2D Arrangements* package computes the arrangements of x-monotone curves. When a general curve is inserted into an arrangement, it is first decomposed into x-monotone sub-curves. Note that when a general polycurve is decomposed then the generated x-monotone sub-curves are not necessarily the segments that constitute the given polycurve. A polyline $\mathscr{P}$ that is also x-monotone is called *x-monotone polyline*. An x-monotone polyline can comprise either a *single* line, or any number of bounded segments and *at most* two rays; see Figure 4.1.

The Arr_polyline_traits_2<SegmentTraits> traits class is a prominent component of the package, which computes the arrangement of (bounded) polylines in the plane. This traits class, in its current implementation as can be found in version 4.3, relies on the Arr_segment_traits_2<Kernel> traits class, which handles the case of an arrangement of bounded line segments. The package also supports the computations of arrangements of linear objects (namely, lines, rays and line segments), using the Arr_linear_traits_2<Kernel> traits class. However, it is not possible to compute the arrangement of *unbounded* polylines. The ultimate goal of

[3]Be careful not to confuse between a segment of a polyline and a linear (bounded) segment.

this work was to extend the package and provide support for the computation of an arrangement of unbounded polylines.

A natural approach to achieve this goal would be to rely on the Arr_linear_-traits_2<K> traits class and generalize the existing traits class, which handles bounded polylines, so it could support unbounded ones as well. Unfortunately, the implementation of the computation of arrangement of polylines shipped with version 4.3 of **CGAL** has some limitations. For example, it is only possible to construct a polyline by providing a range of points as the input. In turn, the constructed polyline is the concatenation of the input points and it has an orientation determined by the order they are given.[4] Another deficiency lies in the iteration over the elements of an existing polyline; currently the user can only iterate over its *vertices*. Finally, the existing code is several years old and fails to meet nowadays standards of **CGAL**. These limitations, among others, make it difficult to generalize the functionality of the traits class directly — for example in order to enable the computation of an arrangement of *unbounded* polylines. Let us demonstrate one of the possible issues. A priori, it is impossible to define an unbounded polyline given merely a sequence of points. One cannot construct, without further details, the first and last segments of such a polyline.

Due to the aforementioned limitations, it turned out that the polylines traits class had to be first revised, before it could be generalized. Unfortunately, due to the lack of time and resources, the ultimate goal, namely implementing the support for computation of arrangements of unbounded polylines, was not accomplished and only the code upgrading was successful. This upgrade lays the foundation for future generalization of the class. In addition, by following an advanced coding style, as we discuss below, we also manage to introduce an improvement of at most 6.77% in the performance of arrangements of polylines (see Table 4.2 in Appendix 4.B for details). This chapter summarizes the contribution to the *2D Arrangements* package of **CGAL**. Throughout the chapter we employ the typesetting conventions that are described in Table 4.1. Furthermore, the implementation that can be found in version 4.3 is referred to as the *current implementation.* The implementation that is the result of this work is referred to as the *upgraded implementation* and it is expected to be part of the next public release (of version 4.4).

Motivation. One obvious motivation is extending the *2D Arrangements* package and allowing the treatment of more general families of polylines (or polycurves). Another motivation is to carry out a benchmark suggested by Dan Halperin as we describe next. In order to utilize the work of Salzman [36] one has to be able to compute the arrangement of rational curves. Here, rational curves are the zero sets of rational functions with two variables in the plane. To this end, the *2D Arrangements* package includes a traits class that handles rational curves in the plane and enables the construction of arrangements induced by such curves; it was developed by

[4]By *orientation* here we refer to a notion of direction of both the constituting segments and in turn the polyline itself.

Typeface	Meaning
foo.bar	file name
ConceptName	concept (note the capitalization and the absence of space)
Model_name	model
`Some_type`	type
`My_functor`	function object (also known as *functor*)

Table 4.1: Typesetting conventions

Oren Salzman and Michael Hemmer. Obviously, the biggest advantage of this traits class is the proven correctness of its output. On the other hand, in some cases, the computational overload introduced by the need of sophisticated number types can result in lengthy running time. It was proposed by Dan Halperin to investigate this issue, and in particular compare two ways for the computation of the arrangement of rational curves. One computation should simply use the dedicated traits class. The second, on the other hand, should practice an approximation approach. That is, first provide a piecewise linear (PL) approximation of the rational curves, and then compute the arrangement of the approximating PL curves using the generalized Arr_polyline_traits_2<SegmentTraits> traits class. Since rational curves are unbounded, the computation of an arrangement of unbounded polylines is mandatory in order to practice the second approach.

These kinds of generalizations are natural given the parameterized nature of CGAL. In particular, Arr_polyline_traits_2<SegmentTraits> is parameterized with the template parameter SegmentTraits.[5] In the existing implementation it is only possible to instantiate Arr_polyline_traits_2<ST> with an instance of either Arr_segment_traits_2<Kernel> or Arr_non_caching_segment_traits_-2<Kernel>. Thus, allowing merely the computation of arrangements of families of bounded polylines. By generalizing the code of Arr_polyline_traits_2<ST>, it is desired to allow its instantiation with models of SegmentTraits that are not necessarily bounded linear segments. For example, use Arr_circular_arcs_traits_2 traits class as the segment traits. This will allow the computation of an arrangement of piecewise-circular-arcs curves in the plane. More generally, Arr_polyline_traits_-2<SegmentTraits> can be the building block of a more general traits class that will handle *polycurves*. Here, given a class of planar curves, a polycurves is a curve such that each of its segments is a planar curve from the given class.

4.2 Existing Implementation

In this section we provide an overview on the current implementation related to the computation of arrangements of polylines. A general introduction to CGAL can be found in [42] and detailed reference to the *2D Arrangements* package can be found

[5]For the sake of brevity we occasionally write ST instead of SegmentTraits.

in [17]. **CGAL** follows the generic programming paradigm; one of the fundamental building blocks of the implementation is the set of the various traits classes. In this chapter we mainly consider two class templates

- Arr_segment_traits_2<Kernel>, and
- Arr_polyline_traits_2<SegmentTraits>.

The first class template is parameterized by a geometric kernel; namely, when Arr_segment_traits_2<Kernel> is instantiated, the template parameter Kernel, or simply K for short, is substituted with a type that must model the concept Kernel of **CGAL**. The class template Arr_segment_traits_2<K> itself models the concepts

- ArrangementTraits_2,
- ArrangementLandmarkTraits_2 and
- ArrangementDirectionalXMonotoneTraits_2.

Note that the class template Arr_non_caching_segment_traits_2 is also a geometry traits class that supports planar segments in the plane. It posses characteristics different than the Arr_segment_traits_2, but syntactically, they are exchangeable.

When the second class, Arr_polyline_traits_2<ST>, is instantiated, the template parameter ST has to be substituted with a type that models the concept ArrangementTraits_2 and can handle line segments. The class template Arr_polyline_traits_2<SegmentTraits> itself currently models the following concepts:

- ArrangementTraits_2
- ArrangementLandmarkTraits_2

Its implementation can be found in *Arr_polyline_traits_2.h*. In the current implementation users can substitute the template parameter with Arr_segment_traits_2<K>. Since Arr_polyline_traits_2<ST> models the concept ArrangementTraits_2, it has two nested types, namely `Curve_2` and `X_monotone_curve_2`, which we denote by `Polyline` and `X-Polyline`, respectively.[6] These types are used to represent polylines and x-monotone polylines, respectively. Similarly, the template parameter Arr_segment_traits_2<K> has two nested types `Curve_2` and `X_monotone_curve_2`, which are used to represent bounded line segments. However, since a bounded line segment is always x-monotone, in this case the two types are virtually the same, and we denote an object of either by `Segment`. In summary, a general bounded polyline can be represented as an object of type `Polyline` and its segments are all objects of type `Segment`. Arr_polyline_traits_2<SegmentTraits>, together with its nested curve types, is the component of the *2D Arrangements* package of **CGAL** that handles the construction and computations of arrangements of polylines. The actual implementations of `Polyline` and `X-Polyline` can be found in *Polyline_2.h*.

4.3 Revising the Implementation

Next, we study in depth the various issues that the current implementation exhibits. In particular, we discuss non-optimal implementation details, design problems and

[6]In addition, it has also the nested type `Point_2`.

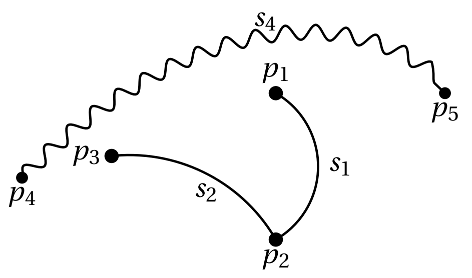


Figure 4.2: A bounded and disconnected polycurve that comprises three segments. Note that its first segment is *not* x-monotone.

limitation that they yield. Furthermore, we describe the measures taken to correct and improve the implementation. One of the fundamental flaws in the design of the existing implementation is the lack of separation between the geometric types and their traits classes. Thus, all cases where functors from traits classes were used inside `Polyline` and `X-Polyline` were removed. In addition, in order to simplify the package, we defined default segment traits and kernel to be used for the computation of arrangements of polylines in case the user does not specify either of them (see Section 4.5).

4.3.1 Non x-monotone Segments

The current implementation assumes, as aforementioned, that the types `Curve_2` and `X_monotone_curve_2` nested in the traits class that substitutes the template parameter SegmentTraits are the same. This assumption obviously holds when the parameter in use is the class template Arr_segment_traits_2<K>. However, having in mind the possibility of allowing the construction of polycurves, this assumption no longer holds (see, for example Figure 4.2, where the segment s_1 is not x-monotone). The first step towards this generalization is to make the following distinction: The segments of `Polyline` can be either `SegmentTraits::Curve_-2` *or* `SegmentTraits::X_monotone_curve_2`, but the segments of `X-Polyline` can only be of the latter type. This reflects the idea that a general polycurve can comprise general curves as its segments and an x-monotone polycurve can only comprise x-monotone curves.

4.3.2 Construction of the Constituting Segments

It is important to note that in general the construction of either an x-monotone segment or general one is not necessarily well defined given its endpoints; for example, consider a circular arc. On the other hand, line segments are well-defined given their endpoints. Indeed, the current implementation exploits this fact. Once the segments provided by SegmentTraits cannot be constructed given their endpoints, the functor Make_x_monotone_2 will break (cf. Section 4.3.4). This issue was not addressed in the presented work, and is left open.

4.3.3 Construction of Polylines

It is safe to say that a fundamental hurdle that the current implementation introduces, when one is after its generalization, is the construction of the geometric objects Polyline and X-Polyline nested in Arr_polyline_traits_2<ST>. The current implementation relies on their vertices and it is *only* possible to construct them from a given range of *points*. This, obviously, introduces a substantial limitation on the possible extensions of the class. For example, one cannot construct an unbounded polyline given a range of points. A far more natural construction should rely on the segments that constitute the Polyline or X-Polyline.

The construction issue was addressed as follows. First, the *construction* of Polyline and X-Polyline was altered, so it can *only* obtain a range of Segments as an input. Secondly, two *construction functors* were defined in the corresponding traits class, Arr_polyline_traits_2<ST>, namely,

- Construct_curve_2, and
- Construct_x_monotone_curve_2.

The signatures of operator() of the first functor are:[7]

```
Curve_2 operator()(const Point_2& p, const Point_2& q) const
Curve_2 operator()(const Segment_2& seg) const
Curve_2 operator()(ForwardIterator begin,
                   ForwardIterator end) const
```

In the first case, the resulting polyline is the segment defined by the two end points.[8] Similarly, the second case yields a polyline, which consists of only one given segment. In this case, if the segments that SegmentTraits can represent are not necessarily x-monotone, then, a priori, the segment that the resulting polyline comprises can be either x-monotone or not. The third overload is the most important one; its input is a range of either points or (general) segments. In the former case, the segments of the constructed polyline connects consecutive points in the range. Here again we employ the assumption that the segments are uniquely defined by their endpoints. The latter case, exhibit an interesting behavior. Since the segments in the input, in general, are not necessarily x-monotone, it is impossible to assert that the constructed polyline will be connected, let alone well oriented (cf. Section 4.3.4). As a matter of fact, we cannot determine the endpoints of a curve that is not x-monotone. This is because the concept of the functors Construct_min_vertex_2 and Construct_max_vertex_2 can only handle, by their definition, x-monotone curves.

[7]In C++, a class that overloads its function call operator, by defining an operator() member function, is called a *function object* or *functor*.

[8]Note that in this case the assumption that a segment can be uniquely constructed given its *two* endpoints is used.

Similar overloads of `operator()` were implemented also for the construction of `X-Polyline` in `Construct_x_monotone_curve_2`. They differ in two senses. First, obviously, their return value is `X-Polyline` instead of `Polyline`. The second difference is that input segments have to be instances of ST::X_monotone_curve_-2 instead of ST::Curve_2. Therefore, it is possible to assert that the constructed x-monotone polyline will be connected and well oriented.

Let us conclude with few remarks. First, although, as we just saw, the construction of disconnected polylines is possible, they are in practice useless and their arrangement cannot be computed. Therefore, it is the responsibility of the user to verify that constructed general, i.e. not necessarily x-monotone, polylines are continuous. Secondly, the construction of degenerated polylines is no longer supported.[9] This change has no significant ramifications since neither of the available traits classes, which could substitute the template parameter SegmentTraits, support the construction of degenerated segments. Lastly, the user should be encouraged to use the construction functors from Arr_polyline_traits_2<ST> and not the direct constructor of the corresponding types. The reason for that is that these functors preform validity tests of the input. In particular, they verify (when possible) that the input yields a valid concatenation of segments and an x-monotone polyline when applicable.

4.3.4 Well Oriented Polylines

Internally, the *2D Arrangements* package compute the arrangement of x-monotone curves. Namely, when it has to process a curve that is *not* x-monotone, it first breaks the curve into x-monotone pieces. This is done using the functor `Make_-x_monotone_2` that is defined in the concept ArrangementTraits_2. Although it is clear from a mathematical point of view, that an x-monotone polyline is nothing but a set of segments that concatenate well (see Figures 4.3(a) and 4.3(b)), it is valuable to store them in some consistent way. Following this motivation, indeed, the curve type nested in the segments traits class, `Segment`, has the member functions `source()` and `target()`. These member functions provide a notion of a direction of a segment; i.e. a segment s is directed (or oriented) from its `s.source()` to its `s.target()`. Note that these functions are not defined in any of the concepts that Arr_segment_traits_2<K> models. This fact poses no problem, as long as these functions are used solely by the corresponding traits class. However, the implementation of `Polyline` and `X-Polyline` uses these functions in order to provide a consistent, lexicographically increasing, storing of x-monotone polylines. This obviously contradicts the generic programming paradigm; Arr_polyline_-traits_2<K>, and its nested types, may only rely on the functionality provided by the concepts that SegmentTraits has to model — see Section 4.2.

However, since the notion of direction of x-monotone polylines is desirable the followings solution was implemented. First, let us we recall that the concept

[9]We say that a polyline is degenerated if it has one or more segments of length zero.

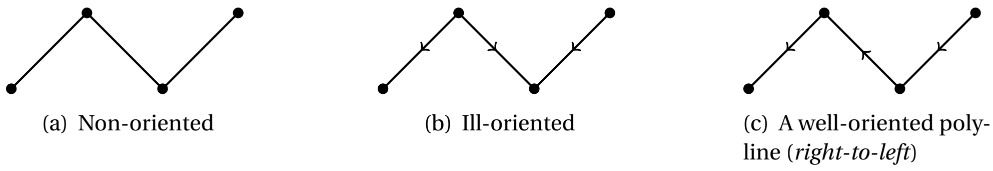

(a) Non-oriented (b) Ill-oriented (c) A well-oriented polyline (*right-to-left*)

Figure 4.3: General x-monotone polyline

ArrangementDirectionalXMonotoneTraits_2 provides two functors:

- `Compare_endpoints_xy_2` and
- `Construct_opposite_2`.

Together with the functors

- `Construct_min_vertex_2` and
- `Construct_max_vertex_2`,

which are provided by the concept ArrangementBasicTraits_2, it is possible to implement the notion of a direction of an x-monotone curve. In particular, since Arr_segment_traits_2<K> models both concepts, we can replace the calls to `source()` and `target()` by combinations of the functors aforementioned. For example, if comparing the end points of a segment s yields `SMALLER`, then `s.source()` can be constructed using the `Construct_min_vertex_2` functor.

When imposing the requirement that the template parameter SegmentTraits should be a model of ArrangementDirectionalXMonotoneTraits_2 as well we guarantee that the constituting x-monotone segments will be oriented. In turn we can implement the notion of a *well-oriented* x-monotone polyline. In particular an x-monotone polyline is well-oriented if the target of its i-th segment is the source of its $i+1$-th segment. Note that in the original implementation an x-monotone polyline could only be stored in an increasing lexicographical order, i.e. well-oriented from left-to-right. Now the x-monotone polylines have to be merely well-oriented and thus in turn allow more generality. See Figure 4.3(c) for an example. Since most of the functors of Arr_polyline_traits_2<ST> assumed that x-monotone polylines are directed from left-to-right, many adaptions had to be made to the code once this invariant was removed. One prominent example is the functor `Intersect_2`, that, according to its concept definition, has to return the objects of the intersection in an ascending xy-lexicographical order.

Remark 4.1. Note that the discussion in this section treated x-monotone polylines alone. It has no effect on the general polylines, and in particular `Polyline` can be ill-oriented or even discontinuous as we already mentioned.

Finally, once posing the condition that SegmentTraits has to be a model of ArrangementDirectionalXMonotoneTraits_2 it is easy to assert that Arr_polyline_-traits_2<ST> will model this concept as well. Indeed, it is rather straight forward to implement the missing two functors that are needed according to the definition of this concept.

4.3.5 Iteration Over Polylines

A fundamental part of the generic programming paradigm is the use of *iterators*. Thus, it comes as no surprise that **CGAL** implements iterators of various kinds. In turn an iterator over the vertices of an existing `Polyline` or `X-Polyline` is implemented in the current implementation in order to inspect the polylines. But the behavior of such an iterator is not always well defined; for example, when the polyline has no first (or last) vertex, like in the case of an unbounded polyline. Therefore, the iterators over the vertices that are currently defined for `Polyline` and `X-Polyline` were replaced with iterators over the segments of the polyline. Note that the previously provided iterators were implemented manually. However, internally a `Polyline` holds the segments in a standard container.[10] Thus, in turn, iterating over the segments of a polyline can be done by simply using the iterators provided by the container. This approach is obviously simpler and more robust. Furthermore, this natural iterator can be used regardless whether the polyline is bounded or not.

4.3.6 Augmentation of Polylines

Augmenting an existing polyline is natural and necessary, thus available in the existing implementation. However, this implementation is limiting on the on hand and introduces bad style on the other. More precisely, it considers the vertices of a polyline as its building blocks, rather then its segments. Thus, it supports the insertion of a vertex at “the end” of an existing polyline. However, in general, finding “the end” of a general polyline might not be possible; for example if the last segment is an unbounded ray. Therefore, for example, it is not possible using the original implementation to augment a `Polyline` that contains a ray. In addition, the augmenting was implemented as a member functions of the classes `Polyline` and `X-Polyline`. This approach does not meet the modern design pattern of **CGAL**.

The issues mentioned were solved by introducing a new functor `Push_back_2` in Arr_polyline_traits_2<ST>. In Code snippet 1 we list all signatures of the overloads of `operator()` of this functor. Note that an `X-Polyline` can only be augmented by an x-monotone segment while a general polyline can obviously be augmented by a general segment. In the first case in Code snippet 1 a new `Segment` that connects the free point of the last segment of `Polyline` and the given point will

[10]Here by “standard container” we refer to one of the implementations of containers as provided by the STANDARD TEMPLATE LIBRARY.

```
    void operator()(Curve_2& cv, const Point_2& p) const
    void operator()(Curve_2& cv, const Segment_2& seg) const
    void operator()(X_monotone_curve_2& xcv,
                    const Point_2& p) const
    void operator()(X_monotone_curve_2& xcv,
                    const X_monotone_segment_2& seg) const
```

Code Snippet 1: The signatures of the various overloads of `operator()` in the functor `Push_back_2`. Here `Curve_2` and `X_monotone_curve_2` are `Polyline` and `X-Polyline`, respectively. Similarly, `Segment_2` and `X_monotone_segment_2` are instances of `SegmentTraits::Curve_2` and `SegmentTraits::X_monotone_curve_2`, respectively.

be added. On the other hand, in the second case, since the input polyline is general, the given segment is simply appended; this can result in a disconnected polyline. The result in the last two cases is more structured and yields an x-monotone polyline as the output. For the sake of safety and robustness, the code validates the input in the case of an `X-Polyline`, and makes sure that the resulting polyline is indeed an x-monotone one. Finally, in order to improve the usability of the traits class, a counterpart functor, `Push_front_2`, was implemented. In contrast to `Push_back_2`, this second functor inserts a point (and the corresponding segment) or a segment at the front of an existing polyline.

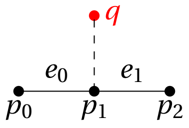
Remark 4.2. Let s be the last segment of some input polyline. Note that in the first overload in Code snippet 1 the given polyline is not necessarily x-monotone. Thus, s can either be x-monotone or not. In the latter case it is impossible to determine the endpoints of s, let alone its free vertex. Therefore, in general it is impossible to append a vertex at the end of a general polyline. However, currently the template parameter SegmentTraits can be replaced by classes that represent x-monotone segments alone. Due to this fact we can assume that s is x-monotone and in turn safely augment the polyline. Nevertheless, this is a tricky issue that has to be addressed. One way to tackle this problem is to check, using meta-programming whether the class that substitutes the template parameter distinguishes between x-monotone curves or not. In case it does distinguish the discussed overload of the `operator()` of the functor `Push_back_2` should be disabled. Otherwise, it can be kept as it is in the upgraded implementation.

4.3.7 Bug Fix in the Location Functions

Due to the lifting of the left-to-right invariant of the x-monotone polylines, the functions `locate()` and `locate_side()` had to be corrected.[11] During the correction a bug was found in the original implementation. Note that, by its defi-

[11]In the current implementation these functions are called `_locate()` and `_locate_side()` (note the additional underscore).

nition, `_locate(seg_tr,c,p)` should return the index of a segment of the polyline `c` that contains the point `p` in its x-range. Similarly, `_locate_side(seg_tr,c,q,bool)` should return the index of the segment of the polyline `c` that is defined to the left (or to the right) of the query point `p` if `bool=TRUE` (if `bool=FALSE`). Set $p_0 = (-1,0)$, $p_1 = (0,0)$ and $p_2 = (1,0)$ and let c be the x-monotone polyline that connects them. Next, let $q = (0,1)$ be the query point.



In the original implementation `_locate(seg_tr,c,q)` would return the (correct) index $i = 0$. Then, if we invoke `_locate_side(seg_tr, c, q, true)` the returned value would be $i = 0$, although it should be $i = 1$, since e_1 is to the right of q. This happens because neither of the following tests

```
if (equal(segment_traits_2()->
            construct_min_vertex_2_object()(cv[i]), q))
if (equal(segment_traits_2()->
            construct_max_vertex_2_object()(cv[i]), q))
```

returns `true`. Note that both functions should return the index of the segment that contains the query point in its x-range. The bug described above was fixed, and the implementation of the two functions was improved.

4.4 Testing the Code and Benchmark

In order to test the correctness of the code two steps were taken. First, the test suite was executed on the upgraded implementation. This step is crucial as it validates the output of the code in degenerated and pathological cases. However, the test suite of the arrangement of polylines presented two problems. First, it did not test cases where `X-Polyline`s were directed from right-to-left. Secondly, it turned out that the tests were not exhaustive enough and some cases were left untested in the original test suite. Therefore, the test suite was revised and updated so it will consider all the possible cases that involve `Polyline`s and `X-Polyline`s (such that the latter could be directed either from left-to-right or from right-to-left). For example, originally there were only 63 testings of the functor `Intersect_2` and now there are about 1700 of them. Similarly, we now have 692 test cases for the functors `Are_mergeable_2` and `Merge_2` versus only 11 cases in the original test suite.

The second step was computing the arrangement of a polyline that is constructed from a random range of points. The arrangement of sets of randomly generated points were computed twice; once with the original code and once with the upgraded one. This step achieved two goals. First, by comparing the resulting arrangement it was possible to confirm the correctness of the computations.

Secondly, and equally important, this test benchmarked the performances of the two implementations. Table 4.2 in Appendix 4.B summarizes the results of the benchmarks. Given the results, it is clear that the new code does not impair the performance of the package. The benchmark showed that the usage of advanced coding resulted in an improvement of 4.8% on average in the time needed for the computations. The code that can be found in Appendix 4.A was used for the benchmark. Note that this code uses the *deprecated* construction of polylines — this was done in order to run the same benchmark both on the original implementation and on the upgraded one.

4.5 Summary

Let us summarize the most important changes that the upgraded code introduces. First, it is now possible to instantiate Arr_polyline_traits_2<ST> without specifying the SegmentTraits. If the template parameter is not provided then, by default, the Arr_segment_traits_2<K> is used with the kernel Exact_predicates_exact_-constructions_kernel. Secondly, polylines (and x-monotone polylines) can and should be constructed using the construction functors that are now provided in the traits class. Furthermore, x-monotone polylines can now be directed either from left-to-right or from right-to-left. Thirdly, one can now iterate over the segments that constitute a polyline instead of iterating over its vertices.

4.6 Future Work

There are two interesting directions for the generalization of the polyline traits class. First, the original goal of this work, namely, computation of the arrangement of unbounded polylines. The second direction, which is even more general, is to implement a traits class for polycurves. It is probably best to base this traits class on the upgraded implementation. Due to time constraints the initial goal has not been completely accomplished. Fortunately however, the code in its new state is much more suitable for further development in the directions mentioned above. The ultimate goal would be to have a class that can compute the arrangement of general, not necessarily bounded, polycurves. In the remaining of the section some useful hints and ideas that could be useful when addressing either of the goals above are provided.

Unbounded Polylines. A fundamental missing step towards the computation of an arrangement of unbounded polylines is modeling the concept Arrangement-OpenBoundaryTraits_2. The functors provided by this concept returns all the needed information about the behavior of the curves near the boundaries.

Polycurves. A significant obstacle in this direction is that the code still assumes that any segment can be uniquely defined given two end points. This condition is satisfied for geodesics on a unit sphere, but not for circular arcs for instance. Thus, it might be possible to compute the arrangement of poly-geodesics on a sphere without removing this assumption. However, in general, it is probably a good practice to remove this assumption and treat polycurves only through their constituting segments.

Some General Remarks. Finally, during the work on this project it was easy to map several issues that have to be addressed. These issues vary from pure technicalities that have to be addressed to issues that involve design decisions that have to be taken.

- The dependency of the output of `Make_x_monotone_2` on the input. Let us start with the example of a triangle given by the vertices $p_1 = (0,1)$, $p_2 = (-1,0)$ and $p_3 = (1,0)$. The triangle can be considered as a (closed) polyline given by the 4-tuple (p_1, p_2, p_3, p_1) or by the tuple (p_2, p_3, p_1, p_2). In the former case, the functor `Make_x_monotone_2` breaks the polyline into the three segments that constitute the triangle. In the latter case, the functor breaks the polyline into *two* segments, namely, (p_2, p_3) and (p_3, p_1, p_2). Therefore, computing the arrangement of this single polyline (i.e. triangle) yields either three vertices and edges and two faces in the latter case or two vertices edges and faces in the former. Note that the number of faces is correct in both cases, but otherwise, the output is different. It is also important to point that since the package computes the arrangements of x-monotone curves, the arrangement of a simple closed curve will always yield at least two vertices corresponding to the leftmost and rightmost points of the curve. It is desirable that the output of the functor will be the set of maximally x-monotone polylines.

- Augmenting a non x-monotone polyline. See Remark 4.2 for further details.

- On the merging of curves. The functor `AreMergeable_2` determines, according to its definition, when two x-monotone polylines are mergable:

  > "(two x-monotone curves) `xc1` and `xc2` are mergeable if their underlying curves are identical, they share a common endpoint, and they do not bend to form a non- x-monotone curve."
  >
  > Taken from [42]

  This definition is rather restrictive and rules out curves that intuitively speaking should be mergeable. For example, the case of overlapping curves is *not* considered mergeable according to this definition. In turn, the current implementation of `Merge_2` deals merely with concatenation of `X-Polyline`'s. A priori, allowing the merging of overlapping curves can save redundant computations. It is therefore recommended to revise the definitions of the concepts `AreMergeable_2` and `Merge_2`. This kind of change potentially influences all elements of *2D Arrangements* package, and the implementations of all models of these functors will have to be revised.

4.A Code for Benchmark

Following is the code that was used to benchmark the upgrade of the code for the computation of an arrangement of polylines in the plane. The code can also be found in the file *bench_random_arr_polylines.cpp*.

```
#include <list>
#include <boost/timer.hpp>
#include <boost/lexical_cast.hpp>
#include <CGAL/Exact_predicates_exact_constructions_kernel.h>
#include <CGAL/point_generators_2.h>
#include <CGAL/Arr_segment_traits_2.h>
#include <CGAL/Arr_polyline_traits_2.h>
#include <CGAL/Arrangement_2.h>

typedef CGAL::Exact_predicates_exact_constructions_kernel Kernel;
typedef CGAL::Arr_segment_traits_2<Kernel>                Segment_traits_2;
typedef CGAL::Arr_polyline_traits_2<Segment_traits_2>     Traits_2;
typedef Traits_2::Point_2                                 Point_2;
typedef Segment_traits_2::Curve_2                         Segment_2;
typedef Traits_2::Curve_2                                 Polyline_2;
typedef CGAL::Arrangement_2<Traits_2>                     Arrangement_2;

int main(int argc, char* argv[])
{
  if (argc < 2) {
    std::cout << "Usage: " << argv[0] << " <number of points> [seed]"
    << std::endl;
    return -1;
  }
  unsigned int number_of_points(boost::lexical_cast<unsigned int>(argv[1]));
  std::list<Point_2> pts;
  unsigned int seed;
  if (argc == 3) {
    seed = boost::lexical_cast<unsigned int>(argv[2]);
    CGAL::Random rnd(seed);
    CGAL::Random_points_in_square_2<Point_2> g(10, rnd);
    for (unsigned int i = 1; i < number_of_points; ++i) pts.push_back(*g++);
  }
  else {
    CGAL::Random rnd;
    seed = rnd.get_seed();
    CGAL::Random_points_in_square_2<Point_2> g(10, rnd);
    for (unsigned int i = 1; i < number_of_points; ++i) pts.push_back(*g++);
  }
  std::cout << "Seed to be used: " << seed << std::endl;
  Polyline_2 poly(pts.begin(), pts.end());
  Arrangement_2 arr;
  boost::timer timer;
  insert(arr, poly);
  double secs = timer.elapsed();

  std::cout << "Arrangement computation took: " << secs << std::endl;
  std::cout << "The arrangement size:" << std::endl
  << "   V = " << arr.number_of_vertices()
  << ",  E = " << arr.number_of_edges()
  << ",  F = " << arr.number_of_faces() << std::endl;

  return 0;
}
```

4.B Summary of benchmark tests

Seed	N	V	E	F	Timer (U)	Timer (I)	Change in %	Mean change in %
1368347036	50	326	615	291	0.017041	0.017389	2.0012652	
1368347050	50	259	485	228	0.013991	0.01423	1.6795502	
1368347062	50	254	471	219	0.013594	0.014035	3.1421446	2.27432
1368347141	500	29357	58378	29023	1.50945	1.65587	8.8424816	
1368347165	500	26395	52464	26071	1.36537	1.44331	5.4000873	
1368347184	500	27592	54847	27257	1.4248	1.50637	5.4150043	6.5525244
1368347214	1000	117252	233848	116598	6.13624	6.39283	4.0137154	
1368347267	1000	115255	229844	114591	6.05129	6.32789	4.3711253	
1368347320	1000	122911	245155	122246	6.41939	6.78416	5.3767895	4.5872101
1371453603	2000	453785	906218	452435	24.2019	25.3598	4.5658877	
1371453661	2000	472734	944122	471390	25.1665	26.2028	3.9549208	
1371453708	2000	450377	899397	449022	23.8502	24.9693	4.4819038	4.3342374
1371453752	2500	710089	1418556	708469	37.8684	39.9537	5.2192913	
1371453814	2500	739811	1477940	738131	39.4883	41.7421	5.3993450	
1371453926	2500	730482	1459272	728792	40.0593	40.8858	2.0214842	4.2133735
1371453996	3000	1041705	2081422	1039719	56.5583	60.2639	6.1489548	
1371454074	3000	1024104	2046202	1022100	55.2771	61.2115	9.6949103	
1371454144	3000	1011434	2020912	1009480	54.5034	57.0535	4.4696644	6.7711765

Table 4.2: Benchmark results. "Seed" is the seed used for the random generation of the vertices. "N" is the number of random samples. "V,E,F" are the number of vertices, edges and faces in the generated arrangement, respectively. "Time (U)" is the running time using the upgraded code and "Timer (I)" is the running time when using the code shipped with version 4.3

5 Conclusion

This dissertation consists of three principal topics: First, (i) the parameterization of contact surfaces. Secondly, (ii) optimal approximation of saddle surfaces, and finally (iii) improvement of the implementation that compute arrangements of polylines in **CGAL**. The first part of this chapter summarizes the work and its second part discusses possible future directions of research.

Summary. The three principal issues treated in this dissertation, which we mentioned above, are all related, one way or another, to the celebrated motion planning problem. The motion planning problem itself is not addressed in the presented work, rather we consider derived issues whose solutions are of general interest and can contribute to the study of the problem. In Chapter 2 we consider the boundary between the free and forbidden spaces in the configuration space of a planar convex polygonal robot. More precisely, we formulate a concise and geometrically motivated parameterization of this boundary. Let us first define the model that we consider. A physical robot is modeled as a convex polygon in the plane, which we denote by A, that is free to rotate and translate amid polygonal obstacles that are scattered in its workspace. In particular, the robot has three degrees of freedom; two degrees determine the translation component of a placement and one degree determines its rotational component. We assume that the description of the robot is known; in particular, its vertices, denoted by $\{a_i\}_{i=1}^{n}$, are given (in a counterclockwise order) with respect to a local frame centered at the so-called *reference point*, which is denoted by R_0. Furthermore, we say that the robot is in its *rest position* if R_0 is at the origin of the workspace and the local frame of the robot aligns with the global one of the workspace. In Figure 2.1 on page 9 the model is illustrated and the robot is plotted in its rest position and in some arbitrary *placement*. A displacement of the robot from its rest position to some placement in the workspace is determined by a translation vector $\vec{r} \in \mathbb{R}^2$ and a scalar θ that corresponds to the rotation of the robot around its reference point. The pair $\mathbf{q} = (\vec{r}, \theta)$ is called a *configuration point* and the collection of all configuration points is the so-called *configuration space* that we denote by $\mathcal{C}$. It is easy to see that there exists a bijection between configuration points in $\mathscr{C}$ and placements of the robot in the workspace. For a configuration point $\mathbf{q}$, we denote with $A(\mathbf{q})$ its corresponding placement of the robot.

Next, we consider the obstacles that are scattered in the workspace of the robot. Once again, we assume that their descriptions are given. In general, given a *source* and a *target* placements of the robot, the motion planning problem has to solve two questions. First, it has to be determined whether a free path from the source to the target exists. In this context, a path is a continuous curve in the configuration space, such that the placements that correspond to its endpoints are the given source and target. A path $c \subset \mathscr{C}$ is free if for all configuration points $\mathbf{q} \in c$ of the path, the corresponding placement $A(\mathbf{q})$ is free, namely, $A(\mathbf{q})$ does not encroach any obstacle. Secondly, if it is determined that a free path exists, then the next task is to construct one.

Clearly, once obstacles are scattered in the workspace there are configuration points that correspond to placements in which the robot intersects one or more obstacles. The collection of such points is called the *forbidden space* and its complement is the *free space*. Note that a path between given source and target placements is free if and only if it is fully contained in the free space. A path in the configuration space between a configuration point that corresponds to a free placement of the robot and a point that corresponds to a forbidden placement must intersect the boundary between the free and forbidden spaces. In this work we consider this boundary and formulate a parameterization of its components.

We say, in this summary, that $A(\mathbf{q})$ touches an obstacle if their boundaries (and possibly also their interiors) intersect.[1] Note that if $\mathbf{q}$ belongs to the boundary between the free and forbidden spaces, then $A(\mathbf{q})$ touches an obstacle. Let O be a convex polygonal obstacle with vertices $\{b_j\}_{j=1}^{m}$ given in a counterclockwise order (see Figure 2.1 for reference). Let b_j and b_{j+1} be two consecutive vertices of the obstacle and let a_i be a vertex of the robot. Then, for all $t \in [0,1]$ and $\phi \in [0, 2\pi)$ we have that the configuration point

$$S_{a_i}^{E_j^O}(t,\phi) = \begin{pmatrix} (1-t)b_j + tb_{j+1} - R^{\phi} a_i \\ \phi \end{pmatrix} \in \mathscr{C}$$

corresponds to a contact of A and O, such that the vertex a_i of the robot lies on the edge of the obstacle that connects b_j and b_{j+1}, which is denoted by E_j^O; cf. Figure 5.1. See Equation (2.9) and Section 2.3.1 for further details. Similarly, if a_i and a_{i+1} are two consecutive vertices of A and b_j is a vertex of O, then for all $t \in [0,1]$ and $\phi \in [0, 2\pi)$, we have that the configuration point

$$S_{E_i^A}^{b_j}(t,\phi) = \begin{pmatrix} b_j - R^{\phi} \cdot ((1-t)a_i + ta_{i+1}) \\ \phi \end{pmatrix} \in \mathscr{C}$$

corresponds to a contact of A and O such that the edge from a_i to a_{i+1} of the robot, denoted by E_i^A, contains the vertex b_j of the obstacle (cf. Equation (2.13)

[1] More precisely, as discussed in Section 2.1, we distinguish between *contacts* and *pseudo contacts*.

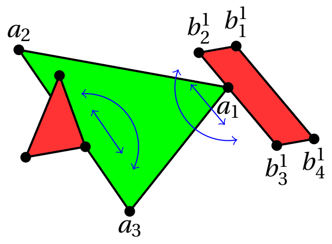


Figure 5.1: Example of a vertex-edge and an edge-vertex contact types. The vertex a_1 of the robot maintains a vertex-edge contact with the right obstacle and the edge from a_2 to a_3 of the robot maintains an edge-vertex contact with the left obstacle. Bare in mind that the obstacles are stationary.

and Figure 5.1). The complete discussion in this case can be found in Section 2.3.2. Clearly, when the robot maintains either of these two possible contacts it has two degrees of freedom (as can be seen in Figure 5.1), and in turn the corresponding geometric objects in the configuration space are surfaces. This observation, obviously, coincides with the parameterizations formulated above. Note that one can consider one-dimensional contacts, where a vertex of the robot touches a vertex of the obstacle, or an edge of A and an edge of O overlap; these cases are discussed in Section 2.3.3 but omitted in this summary.

It is easy to see that a configuration point that lies on the surfaces parameterized above can belong either to the free space or to the forbidden one. The boundary between the free and forbidden spaces is a union of *subsets* of the two (and one)-dimensional surfaces (and curves) that we parameterize. In the respective sections we determine these subsets by finding the corresponding subsets in the parameter domains of the surfaces (and curves). This discussion yields a complete description of the boundary as we illustrate in Figure 2.10 on page 23. In addition, this parameterization allows us to produce visualizations of the configuration space; see [3] for an example.

Using the parameterization that we formalize, it is easy to study the differential geometry of the boundary under discussion. This is done in details in Section 2.5. Let us mention here two key outcomes of this discussion. First, the surfaces that correspond to both vertex-edge and edge-vertex contacts are ruled surfaces. Secondly, the Gaussian curvature of the former is zero everywhere and of the latter is negative. This last result leads us to the second part of the work, Chapter 3, where we consider optimal approximations of saddle surfaces.

It is easy to show that the second order approximation of a smooth negatively curved surface is the so-called *saddle surface*. This is briefly discussed in the introduction (Chapter 1) and in Section 3.1. More details can be found in standard textbooks on differential geometry like [12, 23]. Therefore, studying approximations of saddle surfaces could be helpful when addressing the problem of approximation and discretization of contact surfaces of the second type, namely, those that correspond to edge-vertex contacts of a robot and an obstacle.

In what follows, S is a generic saddle that is given, up to translations and rotations, by

$$S = \left\{ (x, y, z) : z = \frac{x^2}{a^2} - \frac{y^2}{b^2} \right\}.$$

Let $D \subset \mathbb{R}^2$ be some closed domain, over which the saddle S is defined. One can naively obtain an approximation of S over D by first triangulating its domain and then lifting this triangulation.[2] The obtained triangulation can be considered as an approximation of S. An approximation obtained using this approach is most likely useless. In any case, when dealing with approximations, we have to set an error measure. Probably the most crucial and integral element of an approximation problem is the error measure; namely, a criterion that allows the evaluation of the correctness of the yielded approximations. In this work we consider the so-called *vertical distance* (see Section 3.4).

Intuitively, the vertical distance between two *graphs* that are defined over the same domain is the maximal vertical distance between corresponding points. When considering the approximation of saddle surfaces, this error measure has two advantages, First, it is simple to handle. Secondly, the vertical distance between a line segment and a saddle is invariant under translations of the segment. More precisely, if ℓ and ℓ' are two planar line segments such that one is the translation of the other, then

$$\text{dist}_V \left(S, \hat{\ell}\right) = \text{dist}_V \left(S, \hat{\ell}'\right),$$

where $\text{dist}_V (S, \cdot)$ is the vertical distance from the saddle S to some other geometric object and $\hat{\ell}, \hat{\ell}'$ are the liftings of ℓ and ℓ' to the saddle, respectively. Finally, let us point that the vertical distance is an upper bound on the *Hausdorff distance.*

Next, we show that for two planar triangles T and T' such that one is a translation of the other, we have that

$$\text{dist}_V \left(S, \hat{T}\right) = \text{dist}_V \left(S, \hat{T}'\right)$$

where $\hat{T}$ and $\hat{T}'$ are the liftings of T and T' to the saddle, respectively. Same property holds also if T' is a point reflection of T with respect to one of the midpoints of its edges. In this work, given some $\varepsilon > 0$, we find a planar triangle T with $\text{dist}_V \left(S, \hat{T}\right) = \varepsilon$. We can then tessellate the plane with copies of T and obtain a triangulation of S. Let $\mathscr{T}$ be the collection of these triangles, and let $\hat{\mathscr{T}}$ be its lifting to S.

Our next objective is to optimize the yielded triangulation $\hat{\mathscr{T}}$. In particular, we increase the area of the corresponding planar triangles, while maintaining the same prescribed $\varepsilon > 0$ vertical distance. Indeed, we find a one-parameter family of triangles, denoted by $T(\xi)$, such that the vertical distance between the saddle and $\hat{T}(\xi)$ is ε, where $\hat{T}(\xi)$ is the lifting of $T(\xi)$ to the saddle S. Furthermore, we show

[2]Here we say that lifting of a planar triangle T to the saddle S, is the triangle $\hat{T}$ such that its vertices are the vertical liftings of the vertices of T to the given saddle.

that the area of the triangles in this family is fixed. Finally, we prove that if a planar triangle T has an area larger than the area of $T(\xi)$ then $\mathrm{dist}_V\left(S, \hat{T}\right)$ is larger than ε. In other words, the members of the family $T(\xi)$ yield optimal triangulation of the saddle S. Finally, we find the best shaped member of this family. Here, "best shaped" refers to the triangle where the minimal angle is maximized. These results are discussed in detail in Section 3.5.

Note that so far we considered liftings of planar triangles to the saddle S; that is, a lifted triangle $\hat{T}$ has its vertices *on* the saddle. This can be called an *interpolating* approach. In Section 3.6 we consider *non-interpolating* variant. More explicitly, given a planar triangle T, we let $\hat{T}$ be a vertical lifting of T such that its vertices are not necessarily on the saddle. By allowing this additional degree of freedom, we show how the approximation of S can be improved. In particular, we find a planar triangle $T_\star$ such that $\mathrm{dist}_V\left(S, \hat{T}_\star\right) = \varepsilon$, where $\hat{T}_\star$ is a special lifting of $T_\star$, and the area of $T_\star$ is *larger* than the area of $T(\xi)$ that we discuss in the interpolating case. Once again, using translations and point reflections of $T_\star$ we obtain a tessellation of the domain D, and in turn a non-interpolating approximation of S. This result invalidates a conjecture made by Pottmann et al. [34].

The last part of the work, which is considered in Chapter 4, details the contribution made to the *2D Arrangements* package in the COMPUTATIONAL GEOMETRY ALGORITHMS LIBRARY (**CGAL**). In particular, the computation of arrangements of families of polylines was considered. The contribution of this part is twofold. First, the running time of the computation of arrangements of families of polylines was improved by 5% on average. Secondly, the new version of the code, which will be shipped with the next public release of **CGAL** (version 4.4), is far more general and practices higher standards of coding.

Future Work. We conclude the chapter by reviewing the most important and interesting issues that were left untreated in this dissertation.

Parameterization in the Configuration Space. Probably *the* most significant limitation of the parameterization formulated in Chapter 2, is the assumption that the robot has to be convex. The challenge in the case of non-convex robot is to determine the sub-domains of the parameter spaces that correspond to *contact patches.*

Optimal Approximation of Saddle Surfaces. In both the interpolating approach (cf. Section 3.5) and the non-interpolating one (cf. Section 3.6) we find a planar (optimal) triangle T and we tessellate the domain of the saddle using its translations and point reflections. This yields a global approximation of the saddle such that the maximal vertical distance from the lifted triangles to the saddle is ε, for some fixed $\varepsilon > 0$. However, the optimality of the approximation deteriorate for triangles that are far from the center of the saddle. For example, their minimal angles decrease.

An interesting possible direction of research is to obtain a global approximation of the saddle that does not deteriorate. Roughly speaking, obtain many local approximating meshes and combine them together. Let us outline a suggested approach. Let p be an arbitrary point on a saddle surface S and let S_p be a translation and rotation of S such that p is mapped to the origin and the corresponding tangent plane is horizontal. Let M_p be a second order approximation of S_p in a neighborhood of the origin; in particular, M_p is again a (different) saddle surface.

Next, obtain a global approximation (centered at the origin) of the original saddle as we discussed in this work (for example see Figure 3.17 on page 64). Set a quality threshold and purge all triangles in the mesh that do not meet it. For example, set a lower bound on the minimal angle of the lifted triangles and remove from the mesh all the triangles whose minimal angle is smaller than the bound. Let $\mathcal{T}_0$ be the resulting mesh and let $p \in S$ be a point that is not “covered” by $\mathcal{T}_0$. Now, obtain a global approximation of M_p and discard all triangles that do not meet the quality threshold. Using translations and rotations transform the yielded mesh back to the saddle S and denote it by $\mathcal{T}_1$. The meshes $\mathcal{T}_0$ and $\mathcal{T}_1$ are two local (optimal) approximations of S around the origin and around p. The process can now be iterated until a sequence of meshes that covers the domain of interest is obtained. Next, obtain a single mesh $\mathcal{T}$ using the obtained meshes that will be a global approximation of S. Furthermore, it is of interest to use a similar approach and obtain optimal approximations of negatively curved surfaces.

When the non-interpolating approach was employed, we considered a single lifting scheme of the planar triangles. Namely, we lifted the planar triangles to an offset saddle S_α; cf. Equation (3.28). However, another interesting direction is to consider different schemes. It is possible that different lifting schemes can yield improved approximations. If this is not the case, it is interesting to prove that the non-interpolating triangulation suggested in Section 3.6 is optimal.

Arrangements of Polylines in CGAL. As we discussed in Chapter 4, the initial goal of allowing the computation of arrangements of *unbounded* polylines was not accomplished. Thus, naturally, this issue is left open. Furthermore, the updated code, which is a product of this work, is far more generic then the initial version. Therefore, an even more general goal can be set: allow the computation of arrangements of general *polycurves*. Further details can be found in Section 4.6.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit ohne fremde Hilfe und nur mit den angegebenen Quellen verfasst habe. Die Stellen, die ich dem Wortlaut oder dem Sinn nach anderen Werken entnommen habe, sind durch Angabe der Quellen kenntlich gemacht.

Berlin,

Bibliography

[1] D. Atariah, S. Ghosh, and G. Rote. *On the Parameterization and the Geometry of the Configuration Space of a Single Planar Robot.* Tech. rep. Freie Universität Berlin, 2013.

[2] D. Atariah, S. Ghosh, and G. Rote. "On the Parameterization and the Geometry of the Configuration Space of a Single Planar Robot". In: *Journal of WSCG* 21.1 (2013), pp. 11–20.

[3] D. Atariah and G. Rote. "Configuration space visualization". In: *Proceedings of the 2012 symposium on Computational Geometry.* SoCG '12. Chapel Hill, North Carolina, USA: ACM, 2012, pp. 415–416. DOI: 10.1145/2261250.2261313.

[4] C. Bajaj and M.-S. Kim. "Generation of Configuration Space Obstacles: Moving Algebraic Surfaces". In: *The International Journal of Robotics Research* 9.1 (1990), pp. 92–112. DOI: 10.1177/027836499000900104.

[5] E. Berberich, E. Fogel, D. Halperin, M. Kerber, and O. Setter. "Arrangements on Parametric Surfaces II: Concretizations and Applications". In: *Mathematics in Computer Science* 4 (1 2010), pp. 67–91. DOI: 10.1007/s11786-010-0043-4.

[6] E. Berberich, E. Fogel, D. Halperin, K. Mehlhorn, and R. Wein. "Arrangements on Parametric Surfaces I: General Framework and Infrastructure". In: *Mathematics in Computer Science* 4 (1 2010), pp. 45–66. DOI: 10.1007/s11786-010-0042-5.

[7] M. Bertram, J. C. Barnes, B. Hamann, K. I. Joy, H. Pottmann, and D. Wushour. "Piecewise optimal triangulation for the approximation of scattered data in the plane". In: *Computer Aided Geometric Design* 17.8 (2000), pp. 767–787. DOI: 10.1016/S0167-8396(00)00026-1.

[8] J.-D. Boissonnat and F. Avnaim. *Polygon placement under translation and rotation.* Rapport de recherche RR-0889. INRIA, 1988. URL: http://hal.inria.fr/inria-00075665/en/.

[9] V. Borrelli, F. Cazals, and J.-M. Morvan. “On the angular defect of triangulations and the pointwise approximation of curvatures”. In: *Computer Aided Geometric Design* 20.6 (2003), pp. 319–341. DOI: 10.1016/S0167-8396(03)00077-3.

[10] V. Borrelli, F. Cazals, and J.-M. Morvan. *On the angular defect of triangulations and the pointwise approximation of curvatures*. Tech. rep. RR-4590. INRIA, Oct. 2002. URL: ftp://ftp-sop.inria.fr/abs/fcazals/papers/LocalCurv1.pdf.

[11] R. Brost. “Computing metric and topological properties of configuration space obstacles”. In: *Proceedings of IEEE International Conference on Robotics and Automation*. Vol. 1. 1989, pp. 170–176. DOI: 10.1109/ROBOT.1989.99985.

[12] M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

[13] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents series)*. The MIT Press, May 2005.

[14] P. Desnoguès and O. Devillers. “A Locally Optimal Triangulation of the Hyperbolic Paraboloid”. In: *Canadian Conference on Computational Geometry*. 1995, pp. 49–54. URL: http://hal.archives-ouvertes.fr/docs/00/41/32/29/PDF/cccg.pdf.

[15] N. Dyn, D. Levin, and S. Rippa. “Data Dependent Triangulations for Piecewise Linear Interpolation”. In: *IMA Journal of Numerical Analysis* 10.1 (1990), pp. 137–154. DOI: 10.1093/imanum/10.1.137.

[16] E. Fogel, D. Halperin, L. Kettner, M. Teillaud, R. Wein, and N. Wolpert. “Arrangements”. In: *Effective Computational Geometry for Curves and Surfaces*. Ed. by J.-D. Boissonnat and M. Teillaud. Mathematics and Visualization. Springer, 2007. Chap. 1, pp. 1–66.

[17] E. Fogel, D. Halperin, and R. Wein. *CGAL Arrangements and Their Applications - A Step-by-Step Guide*. Vol. 7. Geometry and computing. Springer, 2012, pp. I–XIX, 1–293. DOI: 10.1007/978-3-642-17283-0.

[18] A. Gray. *Modern Differential Geometry of Curves and Surfaces*. CRC Press, 1993.

[19] Y. K. Hwang and N. Ahuja. “Gross motion planning – A survey”. In: *ACM Comput. Surv.* 24.3 (Sept. 1992), pp. 219–291. DOI: 10.1145/136035.136037.

[20] S. Hyde, Z. Blum, T. Landh, S. Lidin, B. Ninham, S. Andersson, and K. Larsson. *The Language of Shape: The Role of Curvature in Condensed Matter: Physics, Chemistry and Biology*. Elsevier Science, 1996.

[21] L. Kavraki, P. Svestka, J.-c. Latombe, and M. Overmars. “Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces”. In: *IEEE International Conference On Robotics And Automation.* 1996, pp. 566–580.

[22] G. Korn and T. Korn. *Mathematical Handbook for Scientists and Engineers: Definitions, Theorems, and Formulas for Reference and Review.* Dover Civil and Mechanical Engineering Series. Dover Publications, 2000.

[23] W. Kühnel. *Differential Geometry: Curves - Surfaces - Manifolds.* American Mathematical Society, 2005.

[24] J.-C. Latombe. *Robot Motion Planning.* 3rd. Kluwer Academic Publishers, 1993.

[25] J.-P. Laumond, ed. *Robot Motion Planning and Control.* Lectures Notes in Control and Information Sciences. Springer, 1998.

[26] S. M. LaValle. *Planning Algorithms.* Cambridge, U.K.: Cambridge University Press, 2006. URL: `http://planning.cs.uiuc.edu/`.

[27] T. Lozano-Pérez and M. A. Wesley. “An algorithm for planning collision-free paths among polyhedral obstacles”. In: *Commun. ACM* 22.10 (Oct. 1979), pp. 560–570. DOI: `10.1145/359156.359164`.

[28] M. Lysenko, S. Nelaturi, and V. Shapiro. “Group morphology with convolution algebras”. In: *Symposium on Solid and Physical Modeling.* 2010, pp. 11–22.

[29] E. Melissaratos. *L_p Optimal d Dimensional Triangulations for Piecewise Linear Interpolation: A New Result on Data dependent Triangulations.* Tech. rep. Utrech University, 1993.

[30] V. Milenkovic, E. Sacks, and S. Trac. “Robust Complete Path Planning in the Plane”. In: *Algorithmic Foundations of Robotics X.* Ed. by E. Frazzoli, T. Lozano-Perez, N. Roy, and D. Rus. Vol. 86. Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, 2013, pp. 37–52. DOI: `10.1007/978-3-642-36279-8_3`.

[31] P. P. Pébay and T. J. Baker. “Analysis of Triangle Quality Measures”. In: *Math. Comput.* 72.244 (Oct. 2003), pp. 1817–1839. DOI: `10.1090/S0025-5718-03-01485-6`.

[32] A. V. Pogorelov. *Extrinsic Geometry of Convex Surfaces (Translations of mathematical monographs Volume 35).* American Mathematical Society, Dec. 1973.

[33] K. Polthier and M. Schmies. “Straightest geodesics on polyhedral surfaces”. In: *SIGGRAPH ’06: ACM SIGGRAPH 2006 Courses.* Boston, Massachusetts: ACM, 2006, pp. 30–38. DOI: `10.1145/1185657.1185664`.

[34] H. Pottmann, R. Krasauskas, B. Hamann, K. Joy, and W. Seibold. “On Piecewise Linear Approximation of Quadratic Functions”. In: *Journal for Geometry and Graphics* 4.1 (2000), pp. 9–31.

[35] H. Pottmann and J. Wallner. *Computational Line Geometry.* Printed chapter 5. Springer-Verlag New York, Inc., 2001.

[36] O. Salzman. “Motion Planning via Manifold Samples”. MA thesis. Tel-Aviv University, 2011.

[37] J. Schwartz and M. Sharir. “On the piano movers’ problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers”. In: *Communications on Pure and Applied Mathematics* 36 (1983), pp. 345–398.

[38] B. Siciliano and O. Khatib, eds. *Springer Handbook of Robotics.* 1st ed. Springer, June 2008.

[39] J. Stoecker and V. Milenkovic. “Interactive visualization of 3D configuration spaces”. In: *Proceedings of the twenty-ninth annual symposium on Computational geometry.* SoCG ’13. Rio de Janeiro, Brazil: ACM, 2013, pp. 341–342. DOI: 10.1145/2462356.2462358.

[40] J. Sullivan. “Curvatures of Smooth and Discrete Surfaces”. English. In: *Discrete Differential Geometry.* Ed. by A. I. Bobenko, J. M. Sullivan, P. Schröder, and G. M. Ziegler. Vol. 38. Oberwolfach Seminars. Birkhäuser Basel, 2008, pp. 175–188. DOI: 10.1007/978-3-7643-8621-4_9.

[41] K. Tang and Y.-J. Liu. “A geometric method for determining intersection relations between a movable convex object and a set of planar polygons”. In: *Robotics, IEEE Transactions on* 20.4 (Aug. 2004), pp. 636–650. DOI: 10.1109/TRO.2004.829479.

[42] The CGAL Project. *CGAL User and Reference Manual.* 4.3. CGAL Editorial Board, 2013. URL: http://doc.cgal.org/4.3/Manual/packages.html.

[43] G. Varadhan. “Accurate sampling-based algorithms for surface extraction and motion planning”. PhD thesis. Chapel Hill, NC, USA: University of North Carolina at Chapel Hill, 2005.

[44] G. Varadhan, Y. J. Kim, S. Krishnan, and D. Manocha. “Topology Preserving Approximation of Free Configuration Space”. In: *International Conference on Robotics and Automation.* 2006, pp. 3041–3048.

[45] R. Wein, E. Fogel, B. Zukerman, and D. Halperin. “2D Arrangements”. In: CGAL *User and Reference Manual.* 4.1. http://www.cgal.org/Manual/4.1/doc_html/cgal_manual/packages.html#Pkg:Arrangement2. CGAL Editorial Board, 2012.

[46] K. D. Wise and A. Bowyer. “A Survey of Global Configuration-Space Mapping Techniques for a Single Robot in a Static Environment”. In: *The International Journal of Robotics Research* 19.8 (2000), pp. 762–779. DOI: 10.1177/02783640022067157.

[47] E. K. Xidias, P. N. Azariadis, and N. A. Aspragathos. “Path Planning of Holonomic and Non-Holonomic Robots Using Bump- Surfaces”. In: *Computer-Aided Design and Applications* 5 (2008), pp. 497–507. URL: http://www.cadanda.com/CAD_5_1-4__497-507.pdf.

Index