Virtual Math Models: Illustrating Fundamentals of Differential Geometry in Virtual Reality Experiences

Tal Rastopchin

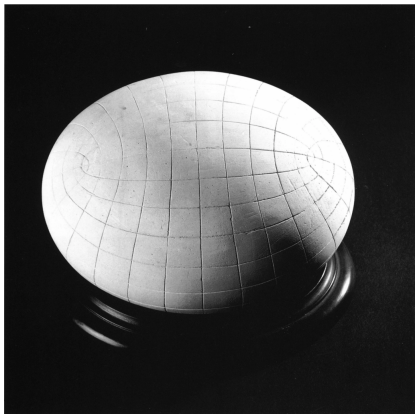
August 9, 2019

1 Introduction

In the late 19$^{\text{th}}$ century, German mathematicians started producing physical, three-dimensional models of geometric structures. They created these models in order to help them visualize what they could not using just a pen and paper [4]. One particular subset of the models made during this time were those concerned with the field of differential geometry. These models interrogated the curvature properties of surfaces; lines with special measures of curvature and points where curvature appears the same in all directions. Other models focused on the constructability of certain surfaces and the existence of minimal surfaces spanning closed boundaries [1]. Whether the insight gained was from the process of creating the models or from engaging with them as a student during lecture, these models took on an important role in how mathematicians engaged with the processes of learning and understanding mathematics.

My project is concerned with investigating the role of the mathematical models produced by German mathematicians in the late 19th century by using the affordances of modern computer graphics and virtual reality to create my own mathematically engaging experiences. I developed two virtual reality experiences, each based on a set of physical models made by German mathematicians in the late 19th century. My virtual reality experiences attempt to use the affordances of computer graphics and virtual reality to create a rich process of learning and understanding that mimics why these models were

Figure 1: Left: a plaster model three-axial ellipsoid, Fischer photo 65 [1]. Right: a string model of a hyperbolic paraboloid, Fischer photo 8 [1].



created in the first place. Specifically, I use the modern visualization techniques and physical embodiment and interaction that virtual reality offers in order to create mathematically engaging experiences.

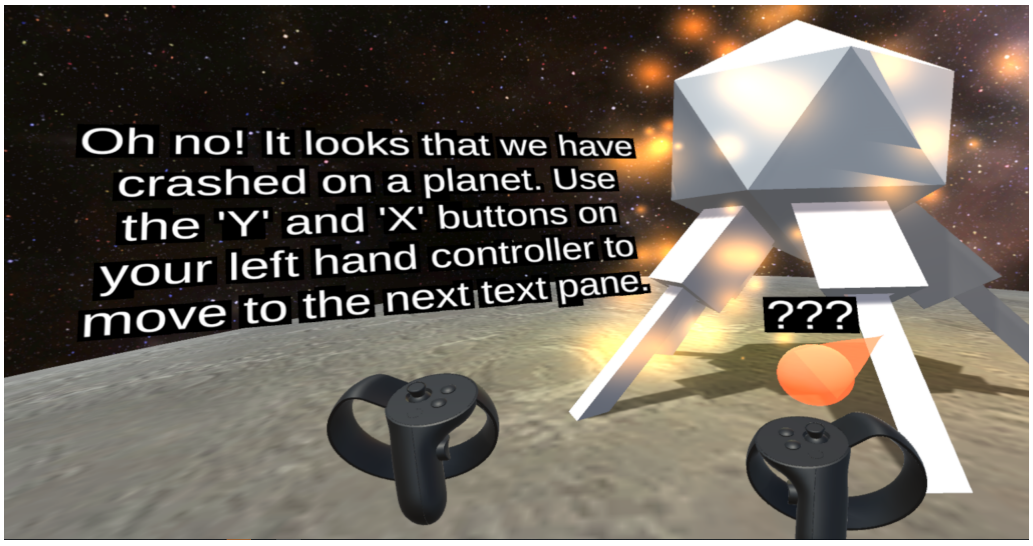
My first project is interested in the curvature properties of surfaces in $\mathbb{R}^3$ and is grounded in the models of quadric sufaces and the special properties of curvature that such models demonstrate. It is the most developed out of the two of my projects, and necessitates the differential geometrical background that I cover in this paper. My second project is interested in the construction of ruled surfaces and challenges a player to construct such surfaces using the real embodied positions of their controllers.

In the next two sections of my paper, **The Curvature Experience** and **The Ruled Surfaces Experience**, I decribe my virtual reality experiences as well as talk about my design choices in creating them. In the following section entitled **Fundamental Concepts of the Differential Geometry of Surfaces in** $\mathbb{R}^3$, I introduce the fundamental mathematics concerning the curvature properties of surfaces needed to create the Curvature Experience. In the next section entitled **Creating the Curvature Experience**, I walk through how I problem solved using the differential geometry covered in the last section to create my Curvature Experience in Unity. Lastly, in the conclusion I reflect on how I could have improved my project as well as

possible future virtual reality projects.

2 The Curvature Experience

Figure 2: A screen capture of the first frame of the curvature experience.



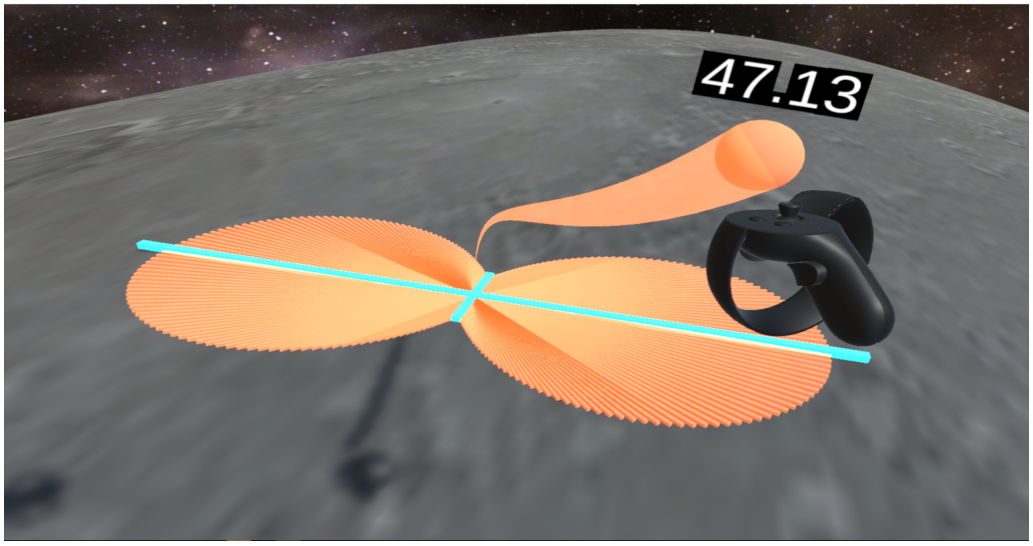
2.1 Description

My first virtual reality experience is grounded in the models of quadric sufaces and the special properties of curvature that such models demonstrate. The key concept illustrated in this experience will be that of curvature, which describes how a surface behaves locally around a given point. At each point on a surface, we can ask how the surface curves as we 'look in a certain direction'. We call points that are locally spherical "umbilical points," and it turns out that different quadric surfaces have unique configurations of such points [1].

In the first experience, our player has crash landed on an unknown planet, and they are tasked with finding all of the the umbilical points on the planet

in order to escape. This unknown planet is a three-axial ellipsoid, and it turns out that there are exactly four umbilical points the player will have to find [1]. Looking at the engraved lines of curvature on the model of the ellipsoid in Figure 1 can give an intuition of where these umbilical points are. However, if one was not told that the umbilical points are where these lines start to converge or if one was not given these lines at all, with a definition of curvature one might not easily understand where such points lie.

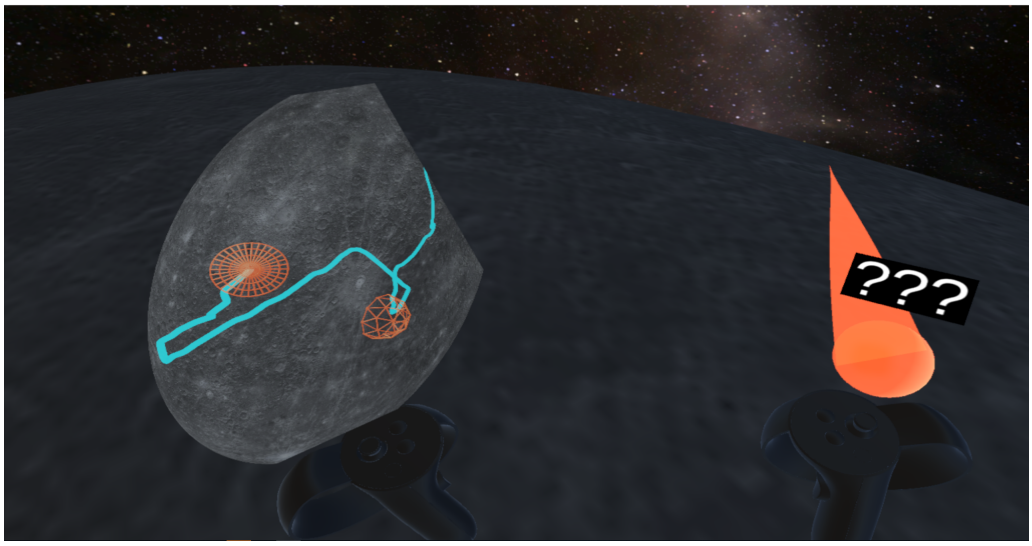
Figure 3: A screen capture of the player using the curvature compass.



My first experience uses the affordances of virtual reality to let a player 'walk' about on an ellipsoid and attempt to gain an intuition of curvature using a 'compass' and 'minimap.' The compass shows the player how the surface that they are walking on curves at a given point by visually displaying the measure of curvature in different directions. The player uses the compass by pointing their right controller at a point on the ground in front of them and pressing down the right trigger. Then, the compass displays a 'peanut shaped' array of orange tangent directions. Each of these orange tangent directions is scaled to demonstrate the 'measure of curvature' in that direction, and two cyan tangents display the most important curvatures called the principle curvatures.

In the beginning of the experience, the player is positioned on the ellipsoid where one can visually see how the surface is curving differently in two directions. To demonstrate this to the player, the instructions urge them to look at where the 'peanut shape' of the compass stretches the most and the least. The instructions ask the player if the curvature of the horizon matches up with the shape of the curvature compass. The curvature compass additionally displays a numerical measuremeant of 'how far' the player is currently away from a closest umbilical point. When the compass gets closer to zero, the player knows that they are getting close to an umbilical point, and when the player finds an umbilical point, the compass reading becomes three exclamation points and a spaceship lands.

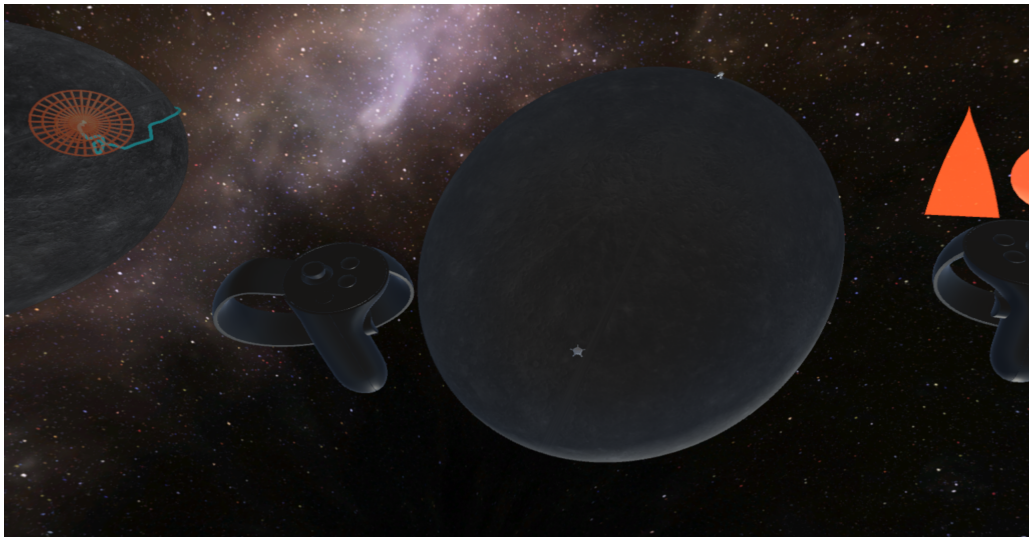
Figure 4: The minimap shows the path of the player, their previously discovered umbilical points, and a different view of the curvature of the ellipsoid.



The player is also outfitted with a minimap that allows them to keep track of their discovered umbilical points as well as their path on the ellipsoid. Additionally, the minimap is intended to give the player another way of visualizing the curvature of the ellipsoid. When I had some volunteers try out the experience to give me feedback how to improve it, some volunteers felt a bit of motion sickness. We found out that this was due to someone

feeling as if they were 'turning' forwards and backwards in the virtual reality. I thought that having the player spin and twist around in space would give them an intuition of the curvature of the surface that they are walking upon; however, I did not want my experience to make people motion sick. To solve this problem, I found out that I could make the surface a lot larger and increase the movement speed of the player, essentially making it so that where the player stands it locally looks less curved and more flat. The only problem with doing this is that it takes away from the player's ability to visually see the curvature on the horizon of the planet. However, by giving the player a minimap with a birds eye view of their position on the planet, the player can visually inspect the curvature of the surface without having to experience the motion sickness firsthand.

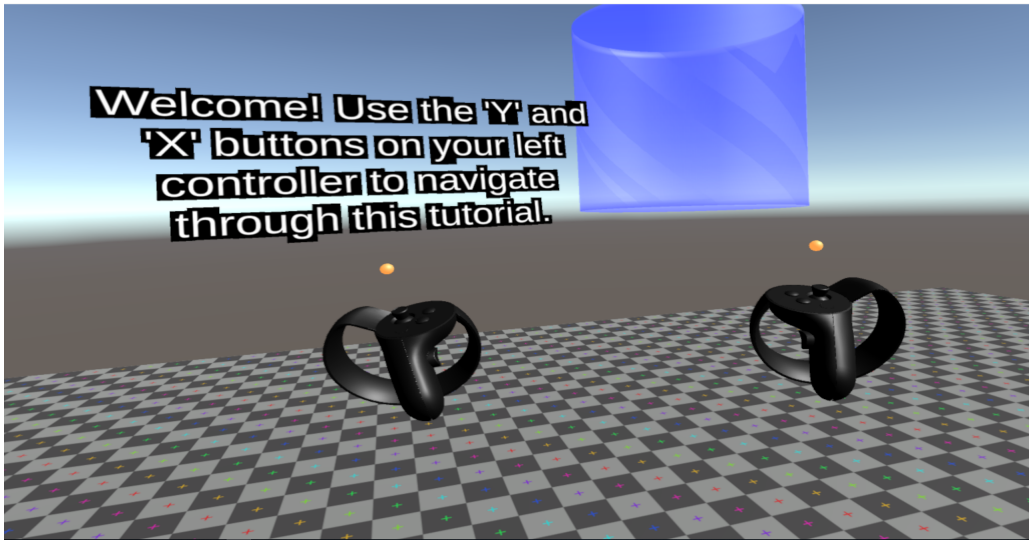
Figure 5: When they have found all the umbilical points, the player takes off and escapes the planet.



3 The Ruled Surfaces Experience

My second virtual reality experience is grounded in the models of ruled surfaces and their construction out of strings and wire. They key concept illustrated in this experience is that of how a ruled surface can be constructed out

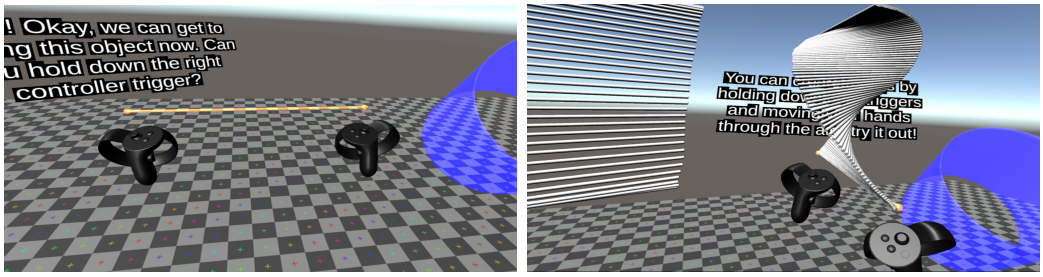
Figure 6: The introductory frame of the ruled surface experience.



of a continuous 'sweeping' of lines. The second experience, unlike the first, is a sandbox experience where players are challenged to figure out how to construct a ruled surface out of lines. I take full advantage of the affordance of embodiment in virtual reality by tasking the players to use the positions of their hands in space to draw out lines constructing four possible ruled surfaces. The experience interactively explains what a ruled surface is to the player by showing them how to construct a cylinder. The experience visualizes the challenge as a blue transparent blueprint that the player can reach out and 'grab.' The experience also demonstrates what a ruled surface is by drawing animated hints that show how one can move their hands to create the challenges. Each challenge comes in the form of a transparent 'blueprint' that the players have to fill in with their rulings and decide whether or not they have solved how to construct the ruled surfaces.

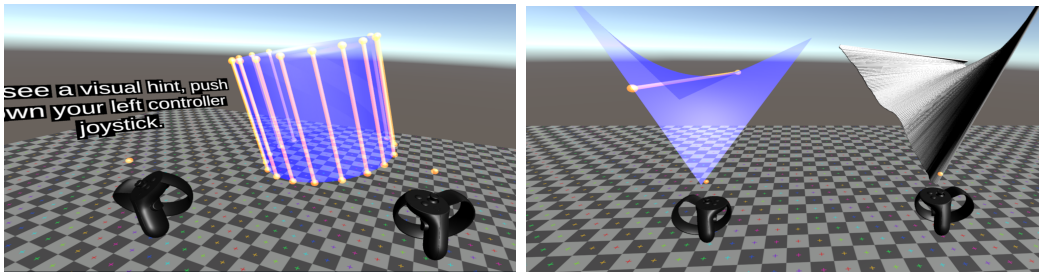
I try to give the player as much of an embodied experience as possible by not only letting the players draw in the physical space around them but also letting them use the controllers to 'grab' their drawings and blueprint visualizations. In this fashion, the players have total control over how they want to engage with the problem. Some of the people who tried out this project were crouching and walking around the virtual objects; it is a really

Figure 7: Left: demonstrating how to draw rulings. Right: drawn rulings.



engaging experience. Even though this project is not as technically engaged as the first, it uses and engages with the affordances of virtual reality in a much more tangible way.

Figure 8: Left: hints for drawing the cylinder. Right: comparing a completed hyperbolic parboloid to the blueprint.



4 Fundamental Concepts of the Differential Geometry of Surfaces in $\mathbb{R}^3$

Differential geometry extends the tools of calculus and linear algebra to analyze and investigate the properties of curves and surfaces. In particular, differential geometry allows us to define and analyze the curvature properties of a surface. In this section of my paper, I will present a brief but

comprehensive introduction to the fundamental concepts of the differential geometry of surfaces in $\mathbb{R}^3$. These fundamentals are motivated by the mathematics late 19^{th} century differential geometers were trying to illustrate with physical models as well as the problems that I needed to solve in order to create my virtual reality experiences. I have written this section in such a way anyone with a backround in single variable calculus, multivariable calculus, and linear algebra should be able to understand it. At times the reader might need to look up the definition of a term.

4.1 Curves

Before we start talking about surfaces in $\mathbb{R}^3$, we must first talk about curves in $\mathbb{R}^3$. This leads us to the following definition from do Carmo [2, pp. 2],

Definition 4.1. A *regular parameterized differentiable curve* is a differentiable map $\alpha : I \to \mathbb{R}^3$ from an open interval $I = (a, b)$ of the real line $\mathbb{R}$ into $\mathbb{R}^3$ such that $\alpha'(t) \neq 0$ for all $t \in I$.

Since α is a differentiable map, we can use the tools of calculus to look at the curve's tangent and normal vectors. Additionally, since α is *regular*, meaning $\alpha'(t) \neq 0$ for all $t \in I$, we have that a tangent line exists at every point to α. Since our goal is to talk about the curvature of surfaces, it will be necessary to analyze the curvature of curves passing through these surfaces. From multivariable calculus, recall that the arc length of a regular parameterized curve $\alpha(t)$ from the point t_0 is given by

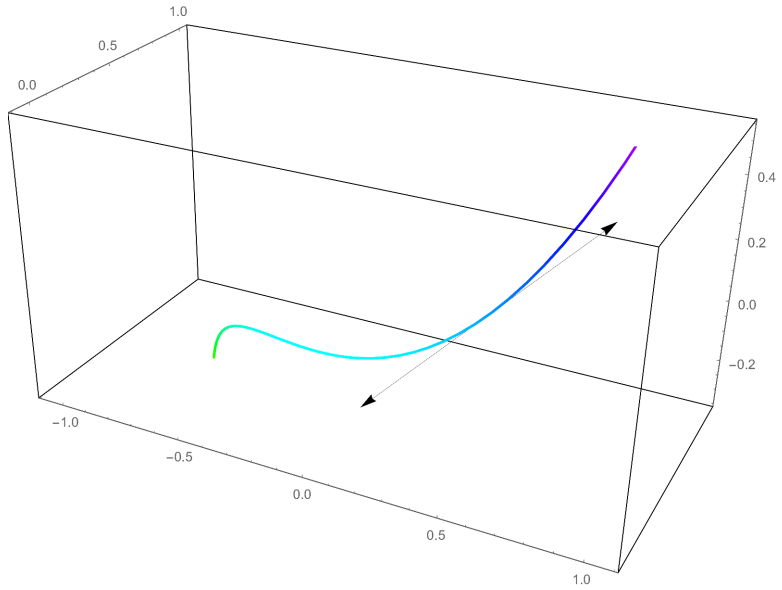
$$s(t) = \int_{t_0}^{t} |\alpha'(\tau)| \, d\tau.$$

Because α is regular, $\alpha'(t) \neq 0$ and arc length $s(t)$ is a differentiable function. By the fundamental theory of calculus, $s'(t) = |\alpha'(t)|$. To simplify their work, differential geometers often assume that the curves they are studying take on the form of an *arc length parameterization.*

Definition 4.2. Suppose that $\beta : I \to \mathbb{R}^3$ is a regular parameterized differentiable curve. We say that $\alpha(t)$ is an *arc length parameterization* of β if the parameter t is the arc length of β measured from some point.

How do we define the measure of curvature of a curve? One way we can think about curvature is by asking how much the unit tangent vector changes

Figure 9: Plot of $\alpha(t) = (t, t^2, t^3)$ with tangent line at $t = .5$.



as we move in an infintesimal direction proportional to arc length. Firstly, recall that the *unit tangent vector* $\mathbf{T}$ is the function $\alpha'(t)/|\alpha'(t)|$. Then, we have the following definition from Stewart [3, pp. 885],

Definition 4.3. The curvature of a regular parameterized differentiable curve $\alpha(t)$ is

$$k = \left|\frac{\mathrm{d}\mathbf{T}}{\mathrm{d}s}\right|.$$

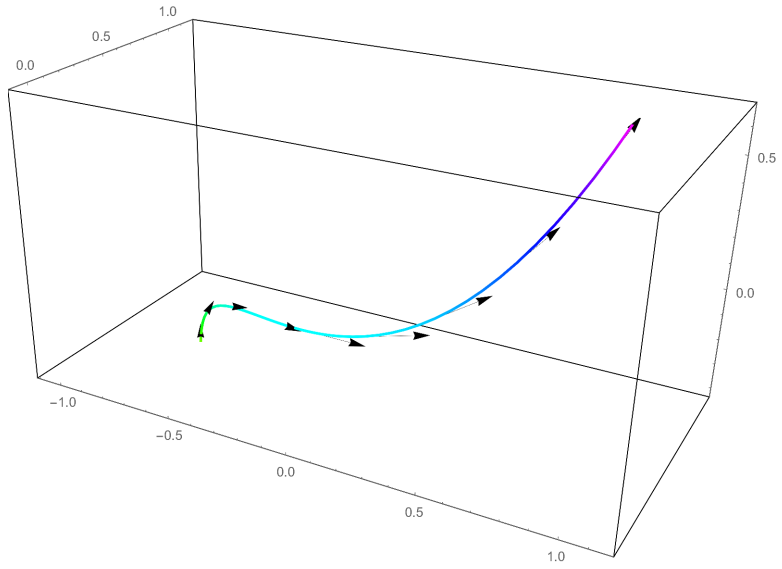
Stewart points out that by the chain rule, since arc length $s(t)$ is a function of t,

$$\frac{\mathrm{d}\mathbf{T}}{\mathrm{d}t} = \frac{\mathrm{d}\mathbf{T}}{\mathrm{d}s}\frac{\mathrm{d}s}{\mathrm{d}t}, \quad \text{and} \quad k = \left|\frac{\mathrm{d}\mathbf{T}}{\mathrm{d}s}\right| = \left|\frac{d\mathbf{T}/dt}{ds/dt}\right|.$$

Since we know that $ds/dt = s'(t) = |\alpha'(t)|$, we conclude that

$$k(t) = \frac{|\mathbf{T}'(t)|}{|\alpha'(t)|}.$$

Figure 10: Plot of $\alpha(t) = (t, t^2, t^3)$ with unit tangent vectors.



In the special case that our curve α is already parameterized by arc length, we have that $|\alpha'(t)| = 1$ and so

$$k(t) = \frac{|\mathbf{T}'(t)|}{1} = \left|\frac{\mathrm{d}}{\mathrm{d}t}\frac{\alpha'(t)}{|\alpha'(t)|}\right| = \left|\frac{\mathrm{d}}{\mathrm{d}t}\frac{\alpha'(t)}{1}\right| = |\alpha''(t)|.$$

Examples 4.4. We will compute the curvature of a parabola. Consider the parabola parameterized by the curve $\alpha(t) = (t, t^2)$. Then,

$$\alpha'(t) = (1, 2t) \quad \text{and} \quad |\alpha'(t)| = \sqrt{1 + 4t^2},$$

$$\mathbf{T}(t) = \alpha'(t)/|\alpha'(t)| = (1 + 4t^2)^{-1/2}(1, 2t),$$

$$\mathbf{T}'(t) = \left(-\frac{4t}{(1 + 4t^2)^{3/2}}, -\frac{8t^2}{(1 + 4t^2)^{3/2}} + \frac{2}{\sqrt{1 + 4t^2}} \right),$$

and so

$$k(t) = \frac{|\mathbf{T}'(t)|}{|\alpha'(t)|} = \frac{2}{(1 + 4t^2)^{3/2}}$$

Notice that at the vertex of the parabola when $t = 0$, $k(t) = 2$. As we get further from the parabola in either direction, the denominator in our equation of

Figure 11: Parabola colored by measure of curvature

curvature increases, and so the curvature decreases. This quantitative analysis is consistent with our qualtiative observation that the parabola curves most near the vertex and flattens as one moves further away.

4.2 Parameterizations

How do we describe a surface in $\mathbb{R}^3$? In multivariable calculus, we investigated graphs of functions of two variables of the form $z = f(x, y)$ where the coordinates x and y were restricted to some subset $U \subset \mathbb{R}^2$ and every point of our graph took the form $(x,\ y,\ f(x, y))$. We can consider the resulting image of $U \subset \mathbb{R}^2$ by f as a surface in $\mathbb{R}^3$. That is, one way we can describe a surface in $\mathbb{R}^3$ is by mapping an open subset of points $U \subset \mathbb{R}^2$ to a subset of points $S \subset \mathbb{R}^3$. This motivates the following definition.

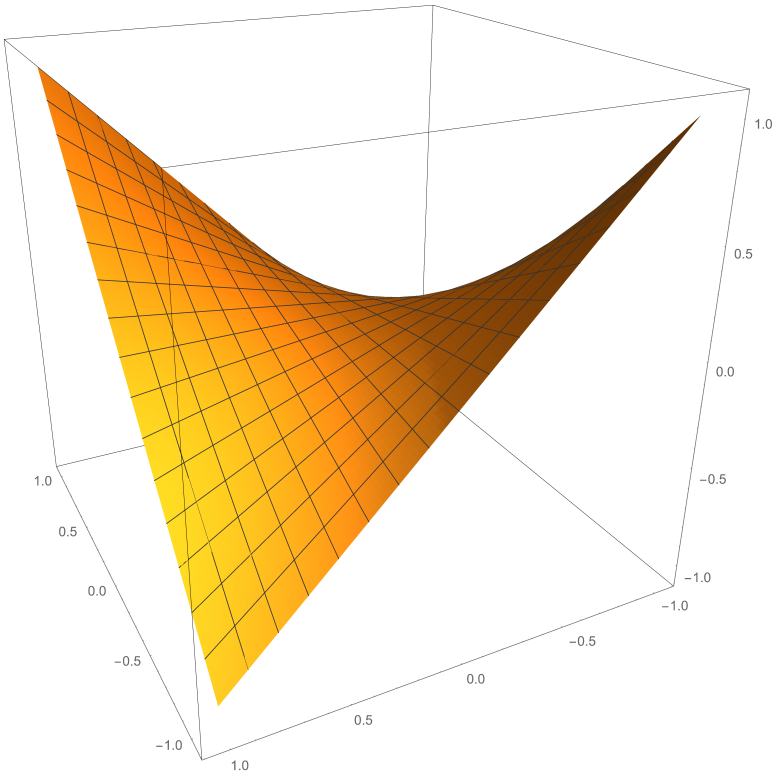
Definition 4.5. A *parameterization* of a surface $S \subset \mathbb{R}^3$ is an open subset $U \subset \mathbb{R}^2$ together with a bijective mapping $\mathbb{X} : U \subset \mathbb{R}^2 \to S \subset \mathbb{R}^3$.

The mapping $\mathbb{X}$ in our parameterization should be bijective because we want to make sure that every point in our subset $U \subset \mathbb{R}^2$ corresponds to exactly one point of $S \subset \mathbb{R}^3$ and vice versa.

Examples 4.6. We will parameterize the hyperbolic paraboloid. Recall that the hyperbolic paraboloid is a surface which can be described as the graph of the function $z = f(u, v) = uv$. Let $U = \{(u, v) \in \mathbb{R}^2 \mid -w < u < w,\ -h <$

$v < h\}$ for $w, h \in R$. Therefore, U is a rectangle centered at the origin with width $2w$ and height $2h$. Then our parameterization $\mathbb{X} : U \subset \mathbb{R}^2 \to S \subset \mathbb{R}^3$ is defined by $\mathbb{X}(u, v) = (u,\ v,\ uv)$. Notice that $\mathbb{X}$ is both one-to-one and onto, so it is a bijection. Then $\mathbb{X}$ is a parameterization of the hyperbolic paraboloid.

Figure 12: Plot parameterized hyperbolic paraboloid for $-1 < u, v < 1$.



Examples 4.7. We will parameterize the ellipsoid. In order to keep our subset $U \subset \mathbb{R}^2$ open, we will not include the 'north' and 'south' poles as well as one longitute of the ellipsoid in our parameterization. Recall that an ellipsoid is the quadratic surface given by the set of points satisfying

$$S = \left\{(x, y, z) \in \mathbb{R}^3 \mid \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1\right\}$$

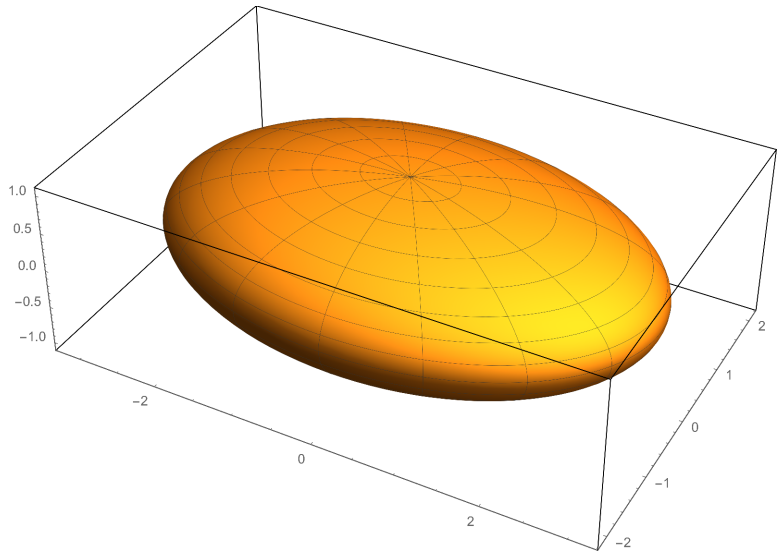
We will parameterize the ellipsoid taking inspiration from spherical coordinates. Firstly, let $U = \{(u, v) \in \mathbb{R}^2 \mid 0 < u < 2\pi,\ 0 < v < \pi\}$. Then our parameterization $\mathbb{X} : U \subset \mathbb{R}^2 \to S \subset \mathbb{R}^3$ is defined by

$$\mathbb{X}(u, v) = \mathbb{X}(x(u, v),\ y(u, v),\ z(u, v)), \text{where}$$

$$\begin{aligned} x(u, v) &= a \cos u \sin v \\ y(u, v) &= b \sin u \sin v \\ z(u, v) &= c \cos v \end{aligned}$$

Notice that the component functions of $\mathbb{X}$ satisfy $x^2/a^2 + y^2/b^2 + z^2/c^2 = 1$ and that $\mathbb{X}(U) = S$. Also, $\mathbb{X}$ is both one-to-one and onto, so it is a bijection. Then, $\mathbb{X}$ is a parameterization of the ellipsoid.

Figure 13: Plot paramterized ellipsoid.



Notice that we did not include the north and south poles $v = 0$ and $v = \pi$ as well as the longitude $u = 0$ in our parameterization, and so in a sense we are not 'fully' parameterizing the ellipsoid. If we had included those points, we would have infinitely many points mapping to the north and south poles and so $\mathbb{X}$ would not be bijective. To 'fully' parameterize the ellipsoid, we will have to use a set of parameterizations called *charts* and a set of *transition*

functions which allow us to relate sections of each chart to corresponding sections of other charts. Because I do not use charts in my project I will not cover 'fully' parameterizing a surface; however, the reader may consult do Carmo pages 55 and section 2-3.

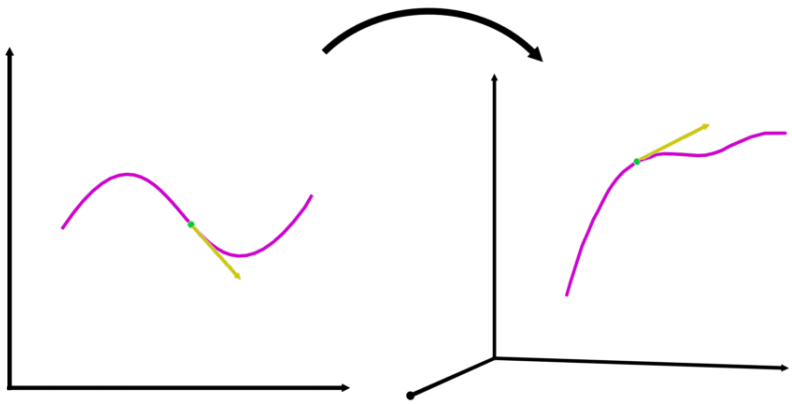
4.3 The Differential

The differential is a mapping that is essential to our understanding and treatment of the surfaces we will investigate. The differential will allow us to talk about the relationship between tangent vectors of curves in our parameterized description of our surface and the corresponding tangent vectors to the surfaces themselves.

Suppose that α is some curve lying in a subset of $U \subset \mathbb{R}^2$, and that F is a differentiable mapping from U to $\mathbb{R}^3$. Suppose that $\vec{w} = \alpha'(t)$ where $\alpha(0) = p$. After applying our mapping F to α, how do we find the new tangent vector corresponding to $\vec{w}$? To answer this question, we will consider the composition curve $\beta = F \circ \alpha$, which by the chain rule, is also differentiable.

We start with a definition from do Carmo [2, pp. 127],

Figure 14: A curve and its tangent vector in $\mathbb{R}^2$ and its image curve and corresponding tangent vector mapped to $\mathbb{R}^3$



Definition 4.8. Let $F : U \subset \mathbb{R}^n \to \mathbb{R}^m$ be a differentiable map. To each point $p \in U$, we associate a linear map $dF_p : \mathbb{R}^n \to \mathbb{R}^m$ which is called the *differential* of F at p and is defined as follows. Let $\vec{w} \in \mathbb{R}^n$ and let $\alpha : (-\epsilon, \epsilon) \to U$ be a differentiable curve such that $\alpha(0) = p$ and $\alpha'(0) = \vec{w}$. Consider the composition curve $\beta = F \circ \alpha : (-\epsilon, \epsilon) \to \mathbb{R}^m$. Then,

$$dF_p(\vec{w}) = \beta'(0).$$

Since α is some curve lying in $U \subset \mathbb{R}^n$, we have that $\beta = F \circ \alpha$ is some curve lying in $\mathbb{R}^m$. Then, we can interpret $dF_p(\vec{w}) = \beta'(0)$ as the tangent vector of β at $t = 0$ corresponding to the tangent vector $\vec{w}$. We call the differential the *pushforward* because it "pushes forward" a tangent vector from one curve to another. The following proposition and proof are taken from do Carmo [2, pp. 127].

Proposition 4.9. *The above definition of dF_p does not depend on the choice of curve which passes through p with tangent vector $\vec{w}$, and dF_p is, in fact, a linear map.*

Proof. For the purposes of the surfaces modeled in this work, we will consider the simpler case of $F : U \in \mathbb{R}^2 \to \mathbb{R}^3$. Let (u, v) be our coordinates in $\mathbb{R}^2$ and (x, y, z) be our coordinates in $\mathbb{R}^3$. Let $\vec{e_1} = (1, 0)$, $\vec{e_2} = (0, 1)$ be the canonical basis for $\mathbb{R}^2$ and $\vec{f_1} = (1, 0, 0)$, $\vec{f_2} = (0, 1, 0)$, $\vec{f_3} = (0, 0, 1)$ be the canonical basis in $\mathbb{R}^3$. Then, suppose given a curve

$$\alpha(t) = (u(t), v(t)), \quad t \in (-\epsilon, \epsilon)$$

for some differentiable functions $u(t)$ and $v(t)$;

$$\alpha'(0) = \vec{w} = u'(0)\vec{e_1} + v'(0)\vec{e_2},$$

$$F(u, v) = (x(u, v),\ y(u, v),\ z(u, v)),$$

for some differentiable functions $x(u, v), y(u, v)$, and $z(u, v)$, and

$$\begin{aligned} \beta(t) &= (F \circ \alpha)(t) \\ &= (x(u(t), v(t)),\ y(u(t), v(t)),\ z(u(t), v(t))). \end{aligned}$$

By taking the derivative at $t = 0$ and applying the multivariable chain rule, we have that

$$\beta'(0) = \left(\frac{\partial x}{\partial u}\frac{\partial u}{\partial t} + \frac{\partial x}{\partial v}\frac{\partial v}{\partial t}\right)\vec{f}_1 + \left(\frac{\partial y}{\partial u}\frac{\partial u}{\partial t} + \frac{\partial y}{\partial v}\frac{\partial v}{\partial t}\right)\vec{f}_2 + \left(\frac{\partial z}{\partial u}\frac{\partial u}{\partial t} + \frac{\partial z}{\partial v}\frac{\partial v}{\partial t}\right)\vec{f}_3$$
$$= dF_p(\vec{w}).$$

which we can express as the matrix-vector product

$$dF_p(\vec{w}) = \begin{pmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} \end{pmatrix} \begin{pmatrix} \frac{\mathrm{d}u}{\mathrm{d}t} \\ \frac{\mathrm{d}v}{\mathrm{d}t} \end{pmatrix}$$

As do Carmo states (128), this shows that dF_p is represented, with respect to the canonical basis of $\mathbb{R}^2$ and $\mathbb{R}^3$, by a matrix which depends only on the partial derivatives at p of the component functions x, y, and z of F. Thus, dF_p is a linear map, and clearly $dF_p(\vec{w})$ does not depend on the choice of α. □

Notice that the matrix in our matrix-vector product is none other than the Jacobian matrix we used when computing the change of coordinates for integrals in multivariable calculus. The tangent vector $\vec{w}$ of the curve α lying in $\mathbb{R}^2$ is a vector that itself belongs to $\mathbb{R}^2$. Since $\alpha(t) = (u(t), v(t))$, we have that $\vec{w} = \alpha'(0) = (u'(0),\ v'(0))$ and so our vector in our matrix-vector product is indeed $\vec{w}$ expressed in the canonical basis of $\mathbb{R}^2$.

The differential is crucial to our understanding of what we will call the *tangent spaces* of the surfaces that we investigate. The above matrix-vector product demonstrates that to compute the differential of some mapping applied to some tangent vector we need only compute the Jacobian matrix of our mapping and the vector representation of our tangent vector in the canonical basis of $\mathbb{R}^2$.

4.4 Regular Surfaces

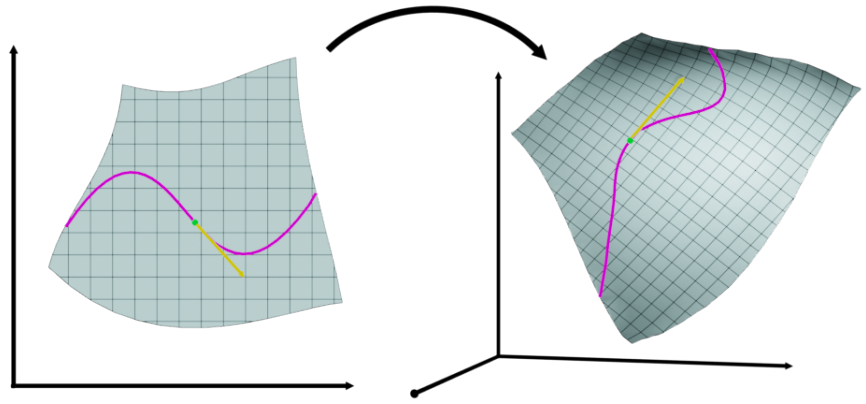
Now we have a way of describing a surface in $\mathbb{R}^3$. But can we always extend the tools of calculus and linear algebra to analyze such surfaces? Recall how we defined the regular parameterized differentiable curve (4.1) such that the component functions are differentiable and the tangent line exists at every

point to such a curve. Ideally, we would like to similarly extend the tools of calculus to the surfaces we analyze, so however we define a surface we want to make sure that it is not self intersecting and that it has no cusps. That is, we would like our surfaces to be 'nice.' The following definition from do Carmo [2, pp. 52] gives us a surface to which we can apply the tools of calculus to investigate.

Definition 4.10. A subset $S \subset \mathbb{R}^3$ is a regular surface if, for each point $p \in S$, there exists a neighborhood V in $\mathbb{R}^3$ and a map $\mathbb{X} : U \to V \cap S$ of an open set $U \subset \mathbb{R}^2$ onto $V \cap S \subset \mathbb{R}^3$ such that

1. $\mathbb{X}$ is differentiable.
2. $\mathbb{X}$ is a homeomorphism; that is, $\mathbb{X}$ is bijective, continuous, and has a continuous inverse $\mathbb{X}^{-1}$.
3. For each point $q \in U$, the differential $d\mathbb{X}_q : \mathbb{R}^2 \to \mathbb{R}^3$ is one-to-one.

Figure 15: A parameterization of a regular surface $\mathbb{X} : U \subset \mathbb{R}^2 \to S \subset \mathbb{R}^3$ with a curve in U and its image curve in S.



Let us unpack this definition. We call the mapping $\mathbb{X}$ a *parameterization* or *system of local coordinates*. Firstly, since $\mathbb{X}$ is differentiable, if we write

$$\mathbb{X}(u, v) = (x(u, v),\ y(u, v),\ z(u, v)),\ \text{for } (u, v) \in U$$

then the functions $x(u,v),\ y(u,v)$, and $z(u,v)$ have continuous partial derivatives of all orders in U.

The fact that $\mathbb{X}$ is a continuous bijective map ensures that our surface has no self intersections. do Carmo notes that this is necessary to speak about the tangent plane at a point p. It turns out that the fact that the differential $d\mathbb{X}_q : \mathbb{R}^2 \to \mathbb{R}^3$ is one-to-one guarantees the existence of a tangent plane at each point on our surface, which we will see later.

Examples 4.11. We will show that the hyperbolic paraboloid is a regular surface. Recall our parameterization from Example 4.6 where

$$U = \{(u,v) \in \mathbb{R}^2 \mid -w < u < w,\ -h < v < h\},\ \text{for } w, h \in R,$$

$$\mathbb{X} : U \subset \mathbb{R}^2 \to S \subset \mathbb{R}^3,\ \text{and}$$

$$\mathbb{X}(u,v) = (u,\ v,\ uv)$$

By computation we can show that $\mathbb{X}$ satisfies properties 1 and 2 of definition 4.10. Recall from linear algebra that one way we can show a linear map is one-to-one is by showing that the column vectors of the matrix representing the transform are linearly independent. By proposition 4.9, we know that the matrix representing $d\mathbb{X}_p$ is the Jacobian of $\mathbb{X}$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ v & u \end{pmatrix}$$

Since the upper two rows of this matrix form the identity matrix, the column vectors of the differential $d\mathbb{X}_p$ are linearly independent for all $(u,v) \in \mathbb{R}^2$ and so the differential $d\mathbb{X}_p$ is one-to-one. We conclude the hyperbolic paraboloid is a regular surface as claimed.

Examples 4.12. We will show that the ellipsoid (minus the north pole, south pole, and one longitude) is a regular surface. Recall our parameterization from Example 4.7 where

$$U = \{(u,v) \in \mathbb{R}^2 \mid 0 < u < 2\pi,\ 0 < v < \pi\},$$

$$\mathbb{X} : U \subset \mathbb{R}^2 \to S \subset \mathbb{R}^3,\ \text{and}$$

$$\mathbb{X}(u,v) = \mathbb{X}(x(u,v),\ y(u,v),\ z(u,v)),\ \text{where}$$

$$x(u,v) = a\cos u \sin v$$
$$y(u,v) = b\sin u \sin v$$
$$z(u,v) = c\ \cos v$$

By computation we can show that $\mathbb{X}$ satisfies properties 1 and 2 of definition 4.10. By proposition 4.9, we know that the matrix representing $d\mathbb{X}_p$ is the Jacobian of $\mathbb{X}$

$$\begin{pmatrix} -a\sin u \sin v & a\cos u \cos v \\ b\cos u \sin v & b\sin u \cos v \\ 0 & -c\sin v \end{pmatrix}$$

By looking at the z component of each of the column vectors, we see that that these two vectors are linearly dependent when $v = k\pi$ for $k \in \mathbb{Z}$. However, there are no points of U for which $v = 0$ or $v = \pi$. Thus, the column vectors of the Jacobian of $\mathbb{X}$ are linearly independent for all $(u, v) \in U$ and the differential $d\mathbb{X}_p$ is one-to-one. We conclude the ellipsoid (minus the poles and one longitude) is a regular surface as claimed.

4.5 The Tangent Plane

We will now show that property 3 of Definition 4.10 guarantees the existence of a tangent plane at each point of a regular surface. To do this, we must first define what it means for a vector to be tangent to a regular surface. The following definition and proposition are from do Carmo [2, pp. 83].

Definition 4.13. A *tangent vector* $\vec{w}$ to a regular surface S at a point $p \in S$ is the derivative $\alpha'(0)$ of a curve $\alpha : (-\epsilon, \epsilon) \to S$ with $\alpha(0) = p$.

A tangent vector of a regular surface S is a vector that can be expressed as the derivative of a curve α lying in S. Remember how we noticed that the differential of a mapping 'pushes forward' the tangent vector of one curve to a tangent vector of the image curve?

Proposition 4.14. *Let $\mathbb{X} : U \subset \mathbb{R}^2 \to S$ be a parameterization of a regular surface S and let $q \in U$. The vector subspace of dimension 2,*

$$d\mathbb{X}_q(\mathbb{R}^2) \subset \mathbb{R}^3$$

is equivalent to the set of tangent vectors to S at $\mathbb{X}(q)$.

Proof. We claim that $d\mathbb{X}_q(\mathbb{R}^2) = \{\vec{w} \in \mathbb{R}^3 \mid \vec{w} \text{ is tangent to } S \text{ at } \mathbb{X}(q)\}$. We proceed by double containment.

($\Longrightarrow$) We will first show that every vector belonging to the image of $\mathbb{R}^2$ by $d\mathbb{X}_q$ is a vector tangent to S at $\mathbb{X}(q)$. Let $\vec{w} = d\mathbb{X}_q(\vec{v})$ for some arbitrary vector $\vec{v} \in \mathbb{R}^2$. Consider the curve $\gamma : (-\epsilon, \epsilon) \to U$ given by

$$\gamma(t) = t\vec{v} + q, \quad t \in (-\epsilon, \epsilon)$$

Then, $\gamma(0) = q$ and $\gamma'(0) = \vec{v}$. By the definition of the differential (4.8), $\vec{w} = \alpha'(0)$, where $\alpha = \mathbb{X} \circ \gamma$. Thus, $\vec{w}$ is the tangent vector of the curve α lying in S, and so $\vec{w}$ is tangent to S at $\mathbb{X}(q)$, as claimed.

($\Longleftarrow$) We will now show that every vector tangent to S at $\mathbb{X}(q)$ belongs to the image of $\mathbb{R}^2$ by $d\mathbb{X}_q$. Let $\vec{w}$ be a tangent vector to S at $\mathbb{X}(q)$. By definition 4.13, $\vec{w} = \alpha'(0)$ where $\alpha : (-\epsilon, \epsilon) \to \mathbb{X}(U) \subset S$ is a differentiable parameterized curve with $\alpha(0) = \mathbb{X}(q)$. Consider the differentiable composition curve $\beta = \mathbb{X}^{-1} \circ \alpha : (-\epsilon, \epsilon) \to U$. By definition of the differential (4.8), we have that $d\mathbb{X}_q(\beta'(0)) = \vec{w}$. Then, $\vec{w} \in d\mathbb{X}_q(\mathbb{R}^2)$, as claimed. □

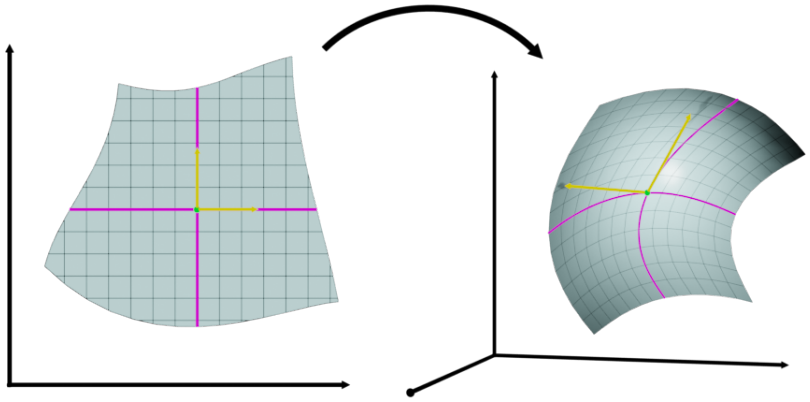
We can now talk about the set of tangent vectors at a given point on a regular surface. The above proof shows that the set of tangent vectors to a regular surface S at a point q is equivalent to the vector subspace $d\mathbb{X}_q(\mathbb{R}^2)$. How do we know that $d\mathbb{X}_q(\mathbb{R}^2)$ has dimension 2? Well, by property 3 of the definition of a regular surface (4.10) we have that for each point $q \in U$ the differential $d\mathbb{X}_q$ is one-to-one. Recall that a linear transformation is one-to-one if and only if the set of column vectors of its matrix representation are linearly independent. The image of $\mathbb{R}^2$ under $d\mathbb{X}_q$ is therefore a vector space of dimension 2.

How do we geometrically describe the 2 dimensional vector space $d\mathbb{X}_q(\mathbb{R}^2)$? Well, the span of two linearly independent vectors in $\mathbb{R}^3$ determines a plane. Thus, as we claimed, property 3 of Definition 4.10 guarantees the existence of a tangent plane at each point of a regular surface.

Definition 4.15. Let $\mathbb{X} : U \subset \mathbb{R}^2 \to S$ be a parameterization of a regular surface S, $q \in U$, and $p = \mathbb{X}(q)$. The *tangent plane* to S at the point $p = \mathbb{X}(q)$ denoted by $T_p(S)$ is the 2-dimensional vector subspace $d\mathbb{X}_q(\mathbb{R}^2) \subset \mathbb{R}^3$. We also call this vector space the *tangent space* of S at p.

Definition 4.16. The *pushforward* is the linear map $d\mathbb{X}_q : T_q(U) \to T_p(S)$. The *pullback* is inverse of the pushforward $d\mathbb{X}_q^{-1} : T_p(S) \to T_q(U)$.

Figure 16: Because the differential $d\mathbb{X}_p$ is one to one, $d\mathbb{X}_p(\vec{e}_1)$ and $d\mathbb{X}_p(\vec{e}_2)$ are linearly independent and define a tangent plane.



The pushforward takes a tangent vector in $T_q(U)$ and 'pushes it forward' to its corresponding tangent vector in $T_p(S)$. The above definition uses Proposition 4.14 to reinterpret the differential as a linear map from one tangent space to another. Since the pushforward is a bijection, we know that it has an inverse mapping.

Notice that the choice of the parameterization $\mathbb{X}$ determines a tangent space basis $\{\frac{\partial \mathbb{X}}{\partial u}(q),\ \frac{\partial \mathbb{X}}{\partial v}(q)\}$ of $T_p(S)$. It is easier to refer to these vectors as $\mathbb{X}_u$ and $\mathbb{X}_v$. This means that at every point p of a regular surface S we have a tangent plane defined by two linearly independent vectors. Therefore, we have a normal vector at each point p of S. This leads us to the following definition.

Definition 4.17. Let $\mathbb{X} : U \subset \mathbb{R}^2 \to S$ be a parameterization of a regular surface S and let $q \in U$. The *unit normal vector field* is a mapping $N : \mathbb{X}(U) \to \mathbb{R}^3$ that assigns to each point $p \in \mathbb{X}(U)$ a unit normal vector to S by the rule

$$N(q) = \frac{\mathbb{X}_u \times \mathbb{X}_v}{|\mathbb{X}_u \times \mathbb{X}_v|}(q).$$

Just as how we showed how to compute the pushforward of a tangent vector in Proposition 4.9, we will show how to express a tangent vector

in the tangent space basis for some given parameterization. The following example is from do Carmo [2, pp. 84].

Examples 4.18. Let $\vec{w}$ be some arbitrary tangent vector belonging to $T_p(S)$ associated to a parameterization $\mathbb{X}$. By Definition 4.13, we have that $\vec{w} = \alpha'(0)$ where $\alpha = \mathbb{X} \circ \beta$ where $\beta : (-\epsilon, \epsilon) \to U$ is given by $\beta(t) = (u(t),\ v(t))$ with $\beta(0) = q = \mathbb{X}^{-1}(p)$. Then,

$$\begin{aligned}\alpha'(0) &= \frac{\mathrm{d}}{\mathrm{d}t}(\mathbb{X} \circ \beta)(0) = \frac{\mathrm{d}}{\mathrm{d}t}(\mathbb{X}(u(t),\ v(t))(0) \\ &= \mathbb{X}_u(q)u'(0) + \mathbb{X}_v v'(0) = \vec{w}\end{aligned}$$

Now that we have

1. A definition of a surface that we can extend the tools of calculus and linear algebra to analyze,
2. a linear map that lets us pushforward tangent vectors in our parameterizations of such surfaces, and
3. an understanding of the relationship between the tangent spaces of the subsets in our parameterization,

we can start to systematically develop the tools of differential geometry that we will use to analyze and investigate surfaces in $\mathbb{R}^3$.

4.6 Measuring Angles and Distances

What is a mathematical structure that we can use to model a 'space' in which we can meaningfully take measurements of angles and distances? We will start with a motivating example.

Examples 4.19. The vector space $\mathbb{R}^3$ equipped with the standard dot product is a mathematical structure in which we can meaningfully take measurements of angles and distances. Recall that for all vectors $\vec{w}_1$ and $\vec{w}_2 \in \mathbb{R}^3$,

1. $\vec{w}_1 \cdot \vec{w}_2 = |\vec{w}_1||\vec{w}_2| \cos\theta$, and
2. $\vec{w}_1 \cdot \vec{w}_1 = |\vec{w}_1|^2$

where θ is the angle between $\vec{w}_1$ and $\vec{w}_2$. Notice that two vectors $\vec{w}_1, \vec{w}_2 \in \mathbb{R}^3$ are orthogonal if and only if $\vec{w}_1 \cdot \vec{w}_2 = 0$.

We will need to abstract what it means to measure angles and distances in order to take such measurements on the surfaces in $\mathbb{R}^3$ that we will investigate. The space $\mathbb{R}^n$ with the standard dot product motivates the following definitions.

Definition 4.20. An *inner product space* is a vector space V equipped with a mapping $\langle \cdot\ ,\ \cdot \rangle : V \times V \to R$ called the *inner product* such that for all vectors $\vec{u}, \vec{v}, \vec{w} \in V$ and scalars $c \in \mathbb{R}$,

1. (symmetry) $\langle \vec{u},\ \vec{v} \rangle = \langle \vec{v},\ \vec{u} \rangle$
2. (billinearity) $\langle \vec{u} + \vec{v},\ \vec{w} \rangle = \langle \vec{u},\ \vec{w} \rangle + \langle \vec{v},\ \vec{w} \rangle$ and $\langle c\vec{u},\ \vec{v} \rangle = c\langle \vec{u},\ \vec{v} \rangle$
   $\langle \vec{u},\ \vec{w} + \vec{v} \rangle = \langle \vec{u},\ \vec{w} \rangle + \langle \vec{u},\ \vec{v} \rangle$ and $\langle \vec{u},\ c\vec{v} \rangle = c\langle \vec{u},\ \vec{v} \rangle$
3. (positive definite) $\langle \vec{u},\ \vec{u} \rangle > 0$ for $\vec{u} \neq 0$

Definition 4.21. Two nonzero vectors $\vec{w}_1$ and $\vec{w}_2$ are *orthogonal* if and only if $\langle \vec{w}_1\ ,\ \vec{w}_2 \rangle = 0$.

By this definition, $\mathbb{R}^3$ is an inner product space when equipped with the standard dot product. Therefore, an inner product space is precisely the mathematical structure we can use to model a 'space' in which we can meaningfully take measurements of angles and distances.

Because our inner products will be motivated by the standard dot product of $\mathbb{R}^3$, differentiating dot products of vector functions will be useful.

Proposition 4.22. *Let $r : I \to \mathbb{R}^3$ and $s : I \to \mathbb{R}^3$ be two vector functions on the open interval $I = (a, b) \in \mathbb{R}$ where $r(t) = (r_1(t), r_2(t), r_3(t))$ and $s(t) = (s_1(t), s_2(t), s_3(t))$. Then,*

$$r(t) \cdot s(t) = r'(t) \cdot s(t) + r(t) \cdot s'(t).$$

Proof. Let the vector functions r and s be defined as above. Then,

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}(r(t) \cdot s(t)) &= \frac{\mathrm{d}}{\mathrm{d}t}((r_1(t), r_2(t), r_3(t)) \cdot (s_1(t), s_2(t), s_3(t))) \\
&= \frac{\mathrm{d}}{\mathrm{d}t}(r_1(t)s_1(t) + r_2(t)s_2(t) + r_3(t)s_3(t)) \\
&= \sum_{i=1}^{3} r_i'(t)s_i(t) + r_i(t)s_i'(t) \\
&= r'(t) \cdot s(t) + r(t) \cdot s'(t).
\end{aligned}$$

as claimed. □

Note that since all of our inner products will be derived from the standard dot product, we will be able to apply this product rule to any of our inner products.

4.7 The First Fundamental Form

How to we measure the angle between two tangent vectors on a regular surface? Well, since the tangent space of a regular surface is a subspace of $\mathbb{R}^3$, we can use the dot product to measure angles between tangent vectors.

Definition 4.23. Let S be a regular surface and consider the tangent plane $T_p(S)$. We endow the vector space $T_p(S)$ with the inner product $\langle \cdot\ ,\ \cdot \rangle_p$ defined by

$$\langle \vec{w}_1\ ,\ \vec{w}_2 \rangle_p = \vec{w}_1 \cdot \vec{w}_2, \quad \vec{w}_1,\ \vec{w}_2 \in T_p(U) \subset \mathbb{R}^3$$

where $\cdot$ is the standard dot product of $\mathbb{R}^3$.

Now, suppose that we are working with some parameterization $\mathbb{X}$ of some regular surface S and we want to measure the angles of tangent vectors in U. To do this, we cannot simply measure the angle between two tangent vectors using the usual dot product of $\mathbb{R}^2$, because doing so would completely disregard our mapping $\mathbb{X}$. We are interested in vectors in S, not in U. We are only using vectors in U to represent vectors in S, so the inner product in U does not tell us about the geometry of S. This motivates the following definition.

Definition 4.24. Let $\mathbb{X} : U \subset \mathbb{R}^2 \to S$ be a parameterization of a regular surface S, $q \in U$, and $p = \mathbb{X}(q)$. We endow the vector space $T_q(U)$ with the inner product $\langle \cdot\ ,\ \cdot \rangle_q$ defined by

$$\langle \vec{w}_1, \vec{w}_2 \rangle_q = \langle d\mathbb{X}_q(\vec{w}_1),\ d\mathbb{X}_q(\vec{w}_2) \rangle_p$$

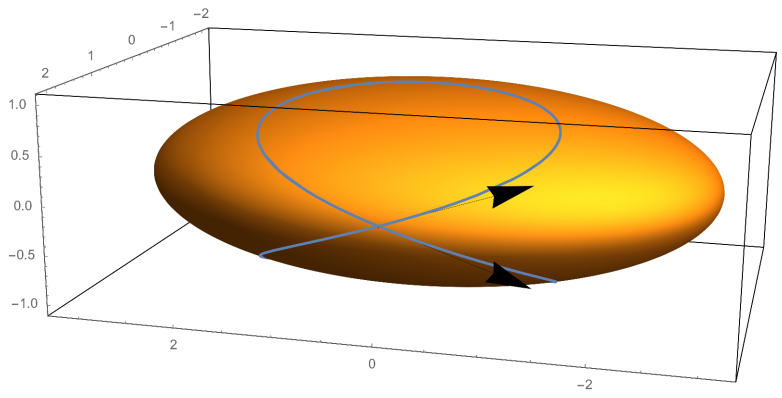
Examples 4.25. Recall our regular surface parameterization of the ellipsoid from Examples 4.7 and 4.12. We would like to determine whether or not the two tangent vectors of S corresponding to $\vec{w}_1 = (1, 1)$ and $\vec{w}_2 = (1, -1)$ at the point $q = (\pi/2, \pi/2)$ are orthogonal. Notice that $(1, 1) \cdot (1, -1) = 0$, which tells us that $(1, 1)$ and $(1, -1)$ are orthogonal in U. By Definition 4.24,

we have that

$$\begin{aligned}\langle \vec{w}_1, \vec{w}_2 \rangle_q &= \langle d\mathbb{X}_q(\vec{w}_1),\ d\mathbb{X}_q(\vec{w}_2) \rangle_p \\ &= \langle (-a, 0, -c),\ (-a, 0, c) \rangle_p \\ &= (-a, 0, -c) \cdot (-a, 0, c) \\ &= a^2 - c^2\end{aligned}$$

By Definition 4.21, we have that $\vec{w}_1$ and $\vec{w}_2$ are orthogonal if and only if $a^2 = c^2$. This is interesting because while $\vec{w}_1$ and $\vec{w}_2$ are orthogonal in the subset $U \subset \mathbb{R}^2$, they are not orthogonal for most choices of a and c.

Figure 17: For choice of $a = 3, b = 2$ and $c = 1$, the two vectors that are orthogonal in U are not orthogonal in S.



Instead of directly using the tangent space inner product to measure angles and lengths of vectors, mathematicians have chosen to 'capture' these metric quantities in a *quadratic form*. Though it does seem arbitrary at first, by representing the tangent space inner product with a quadratic form we can introduce some functions that vastly simplify our algebra moving forward. The following definition and example are from do Carmo [2, pp. 92].

Definition 4.26. The *quadratic form* $I_p : T_p(S) \to R$ given by

$$I_p(\vec{w}) = \langle \vec{w},\ \vec{w} \rangle_p = |\vec{w}|^2 \geq 0$$

is called the *first fundamental form* of the regular surface $S \subset \mathbb{R}^3$ at $p \in S$.

Notice that $I_p(\vec{w})$ is just the square of the length of $\vec{w}$. In order to show how to compute the first fundamental form at a point on our ellispsoid, we will first show how to express first fundamental form as a quadratic form.

Examples 4.27. Suppose that $\mathbb{X}(u, v)$ is a parameterization of S at p. We will write the first fundamental form in terms of the tangent space basis $\{\mathbb{X}_u, \mathbb{X}_v\}$. Let $\vec{w} \in T_p(S)$ be an arbitrary tangent vector to S at p. By Definition 4.13, $\vec{w} = \alpha'(0)$ where $\alpha = \mathbb{X}(u(t), v(t)), t \in (-\epsilon, \epsilon)$ is a parameterized curve with $p = \alpha(0) = \mathbb{X}(u_0, v_0)$. Then,

$$\begin{aligned} I_p(\vec{w}) &= I_p(\alpha'(0)) \\ &= \langle \alpha'(0),\ \alpha'(0) \rangle_p \\ &= \langle \mathbb{X}_u u' + \mathbb{X}_v v',\ \mathbb{X}_u u' + \mathbb{X}_v v' \rangle_p \\ &= \langle \mathbb{X}_u,\ \mathbb{X}_u \rangle_p (u')^2 + \langle \mathbb{X}_u,\ \mathbb{X}_v \rangle_p u'v' + \langle \mathbb{X}_v,\ \mathbb{X}_v \rangle_p (v')^2 \\ &= E \cdot (u')^2 + 2F \cdot u'v' + G \cdot (v')^2 \end{aligned}$$

where

$$\begin{aligned} E &= \langle \mathbb{X}_u,\ \mathbb{X}_u \rangle_p \\ F &= \langle \mathbb{X}_u,\ \mathbb{X}_v \rangle_p = \langle \mathbb{X}_v,\ \mathbb{X}_u \rangle_p \\ G &= \langle \mathbb{X}_v,\ \mathbb{X}_v \rangle_p \end{aligned}$$

We can therefore compute the first fundamental form I_p as a quadratic form in the scalars representing $\vec{w}$ in the canonical basis of $\mathbb{R}^2$ with coefficients E, F, and G.

Noticing that the coefficients E, F, and G are indeed functions of $p = \mathbb{X}(u, v)$, we obtain the following definition.

Definition 4.28. The *coefficients of the first fundamental form* are the differentiable functions $E(u, v), G(u, v)$, and $F(u, v)$ defined above.

It is important to note that the coefficients of the first fundamental form are computed with respect to a particular choice of parameterization $\mathbb{X}$. Now, to compute the first fundamental form applied to a tangent vector all we need to do is compute the coefficents of the first fundamental form and represent our desired tangent vector in the canonical $\mathbb{R}^2$ basis. For an arbitrary tangent vector, computing the first fundamental form comes down to finding its coefficient functions.

Examples 4.29. We will compute the coefficients of the first fundamental form of the ellipsoid. Recall our parameterization of the ellipsoid from Example 4.7 and the Jacobian of our parameterization from Example 4.12. Then,

$$\begin{aligned} E(u,v) &= \langle \mathbb{X}_u,\ \mathbb{X}_u \rangle_p = (\sin^2 v)(a^2 \sin^2 u + b^2 \cos^2 u) \\ F(u,v) &= \langle \mathbb{X}_u,\ \mathbb{X}_v \rangle_p = (b^2 - a^2)(\sin u \sin v \cos u \cos v) \\ G(u,v) &= \langle \mathbb{X}_v,\ \mathbb{X}_v \rangle_p = (\cos^2 v)(a^2 \cos^2 u + b^2 \sin^2 u) \end{aligned}$$

It turns out that we can use the first fundamental form to measure the lengths of curves, and areas of regions on our surfaces. Much of my project has the first fundamental form working in the background, and for that reason I will not explicity illustrate these two metric applications of the first fundamental form. Since the first fundamental form lets us measure lengths of tangent vectors, curves, and areas of regions all residing in our surfaces, we say that the first fundamental form is *intrinsic* to the geometry of our surface. That is, someone who could take measuremeants involving curves or regions that are entirely in the surface would have access to that feature. One might ask if we lose information by 'compressing' the tangent space inner product into the first fundamental form, but it turns out that we can use the properties of the inner product to fully 'recover' the tangent inner product.

Proposition 4.30. *If we know the value of the first fundamental form applied to an arbitrary tangent vector $\vec{w}$, then we can compute the tangent space inner product applied to any two arbitrary tangent vectors $\vec{w}_1$ and $\vec{w}_2$.*

Proof. Suppose that for all $\vec{v} \in T_p(S)$ we know $I_p(\vec{v})$. We claim that we can compute $\langle \vec{v},\ \vec{w} \rangle_p$ for all $\vec{v}, \vec{w} \in T_p(S)$. Let $\vec{v}, \vec{w} \in T_p(S)$ be arbitrary and notice that by billinearity and symmetry of the inner product,

$$\begin{aligned} I_p(\vec{v} + \vec{w}) &= \langle \vec{v} + \vec{w},\ \vec{v} + \vec{w} \rangle_p \\ &= \langle \vec{v},\ \vec{v} + \vec{w} \rangle_p + \langle \vec{w},\ \vec{v} + \vec{w} \rangle_p \\ &= \langle \vec{v},\ \vec{v} \rangle_p + \langle \vec{v},\ \vec{w} \rangle_p + \langle \vec{w},\ \vec{v} \rangle_p + \langle \vec{w},\ \vec{w} \rangle_p \\ &= I_p(\vec{v}) + I_p(\vec{w}) + 2\langle \vec{v},\ \vec{w} \rangle_p \end{aligned}$$

Notice that since we can compute $I_p(\vec{v}+\vec{w}), I_p(\vec{v})$, and $I_p(\vec{w})$ we have a way of determining the quantity $2\langle \vec{v},\ \vec{w} \rangle_p$. Since $\vec{v}$ and $\vec{w} \in T_p(S)$ were arbitrary, we conclude that we can compute $\langle \vec{v},\ \vec{w} \rangle_p$ for all $\vec{v}, \vec{w} \in T_p(S)$, as claimed. □

In this fashion, the first fundamental form I_p completely captures the structure of the tangent space inner product $\langle \cdot\ ,\ \cdot \rangle_p$. Since the tangent space inner product describes all metric quantities that can be computed from within a surface, we conclude that the first fundamental form does the same.

Moving forward we will no longer denote exactly which kind of inner product we are using. That is, we will not specify whether the inner product is at a point in U or $T_p(U)$ or $\mathbb{R}^3$. We will use the notation $\langle \cdot\ ,\ \cdot \rangle$ to represent each of these inner products, and specify when necessary which one we are using.

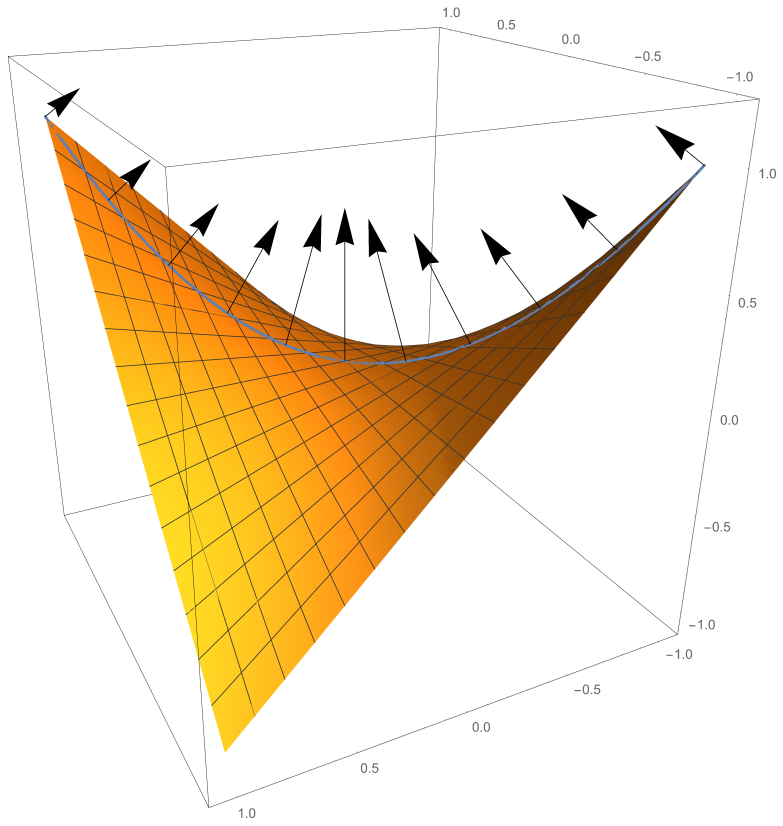
4.8 The Second Fundamental Form

One concept that we grappled with in multivariable calculus was that of the *directional derivative*. In computing the directional derivative of some function of two variables, we were interested in finding the rate of change of that function of two variables as we moved in some 2-dimensional direction. We may similarly measure the curvature at a point of a surface in a particular direction by looking at the curvature of a curve passing through that point with tangent vector in that same direction. This leads us to the following definition adapted from do Carmo and Fischer [2, pp. 141], [1, pp. 150].

Definition 4.31. Let $p \in S$ and $\vec{v} \in T_p(S)$ be a unit length tangent vector. Consider the plane E passing through p which contains both $\vec{v}$ and the normal vector of S at p. Then, consider the plane curve α which is the intersection of E and S. The *normal curvature* at p in the direction of $\vec{v}$ is the curvature of the plane curve α at the point p. We denote the normal curvature k_n and assign to it a positive sign if α is above the tangent plane $T_p(S)$ in some neighborhood of p and a negative sign if α is below the tangent plane $T_p(S)$ in some neighborhood of p.

Examples 4.32. Consider the hyperbolic paraboloid parameterized in Example 4.11. Looking at the differential of our parameterization mapping $d\mathbb{X}_p$, we know that at the point $\mathbb{X}(0,0) = (0,0,0)$ the tangent space basis is $\{(1,0,0),(0,1,0)\}$. Then, $(2/\sqrt{2})(1,1,0) \in T_p(S)$. We will compute the normal curvature at the point $p = \mathbb{X}(0,0) = (0,0,0)$ in the direction $\vec{v} = (2/\sqrt{2})(1,1,0)$. To do this, we must first find the plane E that contains the normal vector $N(p) = (1,0,0) \times (0,1,0) = (0,0,1)$ and the vector $\vec{v}$.

Figure 18: A plot of the acceleration vectors of our curve $\alpha(t) = (t, t, t^2)$ lying in the hyperbolic paraboloid.



Since the normal vector of our surface is $(0, 0, 1)$, our plane E is the plane defined by the line spanned by $\vec{v}$ in the $uv-$plane. Since the span of $\vec{v}$ is equivalent to the line $u = v$ in the $uv-$plane, we have that the intersection curve α is none other than the mapping of the curve $u = v$. Therefore we can parameterize our curve in U such that $u = t = v$, which tells us that our intersection curve α is the curve $\alpha(t) = \mathbb{X}(t, t) = (t, t, t^2)$. Now, we just have to find the curvature of α at $t = 0$. Since our tangent vector is

$$\alpha'(t) = (1, 1, 2t),$$

we have that the unit tangent vector is

$$T(t) = (2 + 4t^2)^{-1/2}(1, 1, 2t),$$

and by the product rule the derivative of the unit tangent is

$$T'(t) = \left(\frac{-4t}{(2+4t^2)^{3/2}}, \ \frac{-4t}{(2+4t^2)^{3/2}}, \ \frac{2}{\sqrt{2+4t^2}} + 2t\frac{-4t}{(2+4t^2)^{3/2}} \right).$$

Then,

$$|T'(0)| = |(0, 0, \sqrt{2})| = \sqrt{2}.$$

We lastly need to compute the length of the tangent vector

$$|\alpha'(0)| = |(1, 1, 0)| = \sqrt{2}.$$

Since α is above the tangent plane $T_p(S)$ in a small neighborhood, we say that the sign of our normal curvature is positive and conclude that the normal curvature at p in the direction $\vec{v}$ is $\sqrt{2}/\sqrt{2} = 1$.

Thinking about the curvature at a point on a surface in a given direction as the curvature of a curve passing through that point in the given direction is geometrically intuitive. However, as we just saw it can be tricky to compute. If the intersection of the plane E with the surface S had not been as simple as it was in the above example, finding the parameterization of a curve passing through the desired point with the specified direction could be tricky. To simplify this computation and express it in terms of our description of the surface without a curve itself, differential geometers have investigated how the unit normal vector field changes as we move in some direction. That is, starting at some point $p \in S$, we might ask how dN_p changes as we move infinitisimally in some tangent direction $\vec{v} \in T_p(S)$. This suggests the following definition from do Carmo [2, pp. 141].

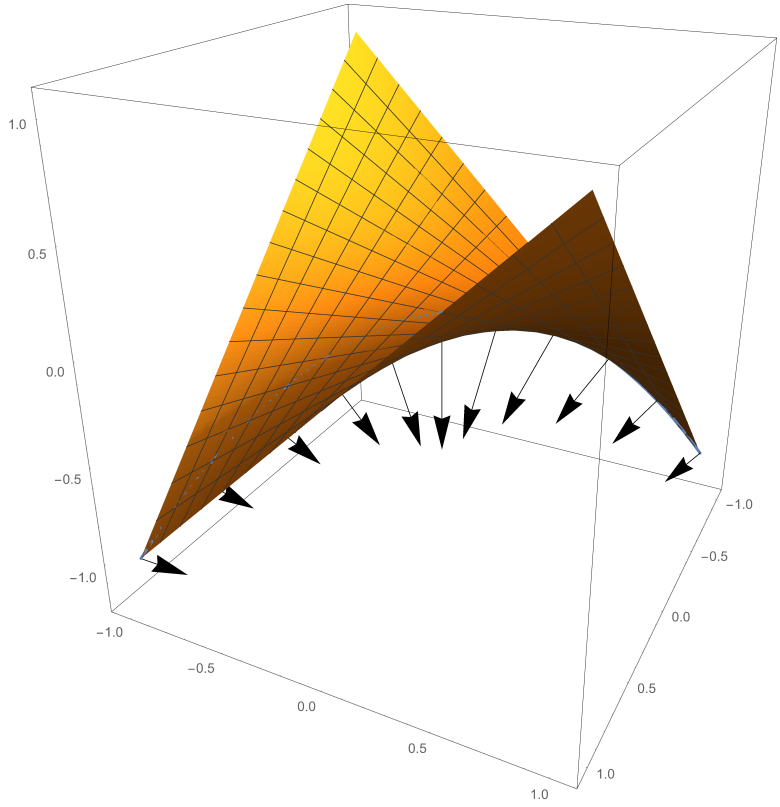
Definition 4.33. The quadratic form $II_p : T_p(S) \to \mathbb{R}$ defined by

$$II_p(\vec{v}) = -\langle dN_p(\vec{v}), \ \vec{v} \rangle$$

is called the *second fundamental form* of S at p.

Let us break down this definition. For the purposes of my project and that of simplicity, we will assume that $\vec{v} \in T_p(S)$ is a tangent vector of unit length. What exactly is $dN_p(\vec{v})$? As with the differential, let us think of $\vec{v}$ as the tangent vector of some curve $\alpha(s) : (-\epsilon, \epsilon) \to S$. Then we can consider the restriction of the unit normal vector field N to the curve α as $N \circ \alpha(s) = \overline{N}(s)$. By the chain rule, $\overline{N}'(0) = dN_p(\alpha'(0))$. We can think of

Figure 19: If we had alternatively asked for the curvature in the direction $(1, -1, 0)$, we would have parameterized our curve $\alpha(t) = (t, -t, t^2)$, our acceleration vectors would points downards, and we would have a normal curvature of -1.



$dN_p(\vec{v})$ as the vector pointing in the 'tangent direction' of the unit normal vector field starting at the point p as one travels in the direction $\vec{v}$.

When we take the inner product of the 'tangent vector' of the unit normal vector field with the direction $\vec{v}$ itself, since $\vec{v}$ is a unit vector, this quantity measures the length of the projection of $dN_p(\vec{v})$ onto $\vec{v}$. That is, we are measuring 'how much' of the 'tangent vector' of the unit normal vector field lies in the direction of $\vec{v}$. If this seems confusing at first, do not worry about it, because we are just about to show a relationship between the second fundamental form and normal curvature. We will start with a lemma followed

by a proposition adapted from [2, pp. 141].

Lemma 4.34. *Suppose that $\alpha(t)$ and $\beta(t)$ are two differentiable vector valued functions such that $\langle \alpha(t),\ \beta(t)\rangle = c$ for some constant c. Then,*

$$\langle \alpha'(t),\ \beta(t)\rangle = -\langle \alpha(t),\ \beta'(t)\rangle.$$

Proof. By Proposition 4.22, differentiating both sides of $\langle \alpha(t),\ \beta(t)\rangle = c$ yields $\langle \alpha'(t),\ \beta(t)\rangle + \langle \alpha(t),\ \beta'(t)\rangle = 0$. Then,

$$\langle \alpha'(t),\ \beta(t)\rangle = -\langle \alpha(t),\ \beta'(t)\rangle$$

as claimed. □

Proposition 4.35. *The value of the second fundamental form II_p for a unit tangent vector $\vec{v} \in T_p(S)$ is equal to the normal curvature at p in the direction of $\vec{v}$.*

Proof. Let $\alpha(s)$ be the arc length parameterization of plane curve α specified to be the intersection of the plane E and the surface S in Definition 4.31. We will denote the restriction of the unit normal vector field N to the curve α as $\overline{N}(s) = N \circ \alpha(s)$. Then, since the tangent vector $\alpha'(s)$ is orthogonal to the normal vector $\overline{N}(s)$, we have that $\langle \overline{N}(s),\ \alpha'(s)\rangle = 0$. By Lemma 4.34, we have that

$$\langle \overline{N}(s),\ \alpha''(s)\rangle = -\langle \overline{N}'(s),\ \alpha'(s)\rangle.$$

Therefore,

$$\begin{aligned} II_p(\alpha'(0)) &= -\langle dN_p(\alpha'(0)),\ \alpha'(0)\rangle && \text{(Definition 4.33)} \\ &= -\langle \overline{N}'(0),\ \alpha'(0)\rangle && \text{(Chain rule)} \\ &= \langle \overline{N}(0),\ \alpha''(0)\rangle && \text{(Above computation)} \end{aligned}$$

Now, since $\alpha(s)$ is an arc length parameterization, we have that $|\alpha'(s)| = c$ for some constant c. Then, since $|\alpha'(s)|^2 = \langle \alpha'(s),\ \alpha'(s)\rangle$, by Lemma 4.34 we have that

$$\langle \alpha''(s),\ \alpha'(s)\rangle = -\langle \alpha'(s),\ \alpha''(s)\rangle$$

and billinearity and symmetry of the inner product tells us that

$$\langle \alpha'(s),\ \alpha''(s)\rangle = 0$$

which means that $\alpha'(s)$ and $\alpha''(s)$ are orthogonal. Since $\alpha''(s)$ is orthogonal to $\alpha'(s)$, $\overline{N}(s)$ is orthogonal to $\alpha'(s)$, and all three vectors $\alpha'(s), \alpha''(s)$, and $\overline{N}(s)$ belong to the plane E, we must have that $\overline{N}(s)$ is parallel to $\alpha''(s)$. Because $\overline{N}(s)$ is parallel to $\alpha''(s)$ and $\overline{N}(s)$ is unit length, we can say that

$$\begin{aligned}\langle \overline{N}(0),\ \alpha''(0)\rangle &= |\alpha''(0)| \cdot \langle \overline{N}(0),\ \alpha''(0)/|\alpha''(0)|\rangle \\ &= |\alpha''(0)| \cdot 1 \\ &= k_n.\end{aligned}$$

Then, the value of the second fundamental form II_p for a unit tangent vector $\vec{v} \in T_p(S)$ is equal to the normal curvature at p in the direction of $\vec{v}$, as claimed. □

We will now compute the normal curvature as we did in Example 4.32 using the second fundamental form in place of the direct definition.

Examples 4.36. Recall the computation of normal curvature for the hyperbolic paraboloid from Example 4.32. We will compute the normal curvature at the point $p = (0,0,0) = \mathbb{X}(0,0)$ in the direction $\vec{v} = \sqrt{2}/2(1,1,0)$. By Proposition 4.35, we have that our desired normal curvature is the quantity

$$II_p(\vec{v}) = -\langle dN_p(\vec{v}),\ \vec{v}\rangle.$$

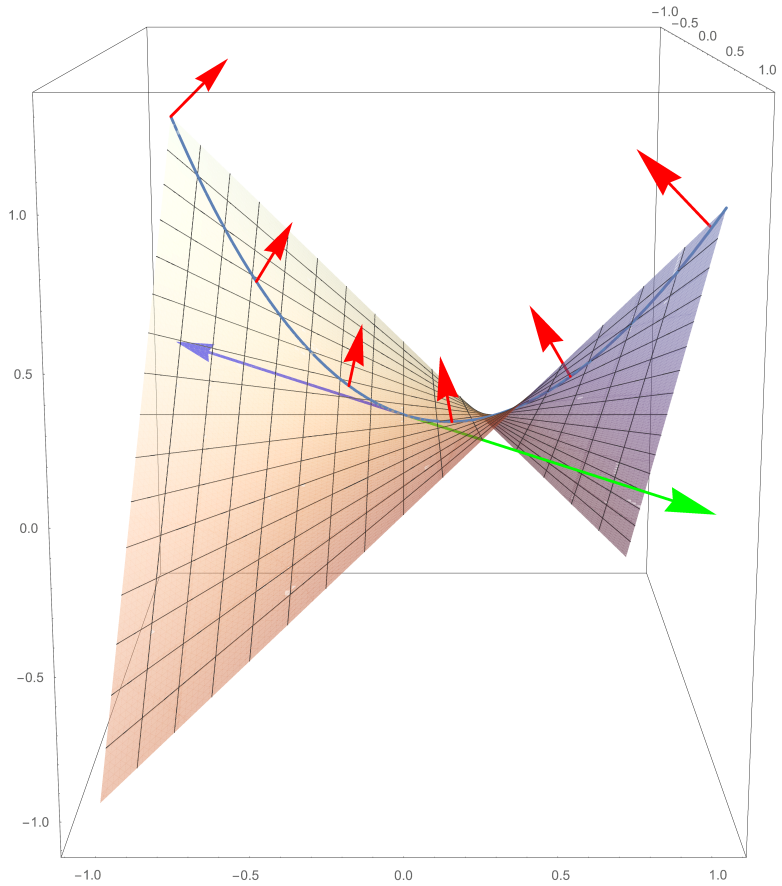
We shall first compute the matrix representing dN_p. To do this, we use Definition 4.17 to define a unit normal vector field

$$\begin{aligned}N(q) &= \frac{\mathbb{X}_u \times \mathbb{X}_v}{|\mathbb{X}_u \times \mathbb{X}_v|}(q) \\ &= \frac{(1,0,v) \times (0,1,u)}{|(1,0,v) \times (0,1,u)|} \\ &= (1+u^2+v^2)^{-1/2}(-v,-u,1).\end{aligned}$$

Then, the Jacobian matrix of $N(q)$ is

$$\begin{pmatrix} \frac{uv}{(1+u^2+v^2)^{3/2}} & \frac{v^2}{(1+u^2+v^2)^{3/2}} - \frac{1}{\sqrt{1+u^2+v^2}} \\ \frac{u^2}{(1+u^2+v^2)^{3/2}} - \frac{1}{\sqrt{1+u^2+v^2}} & \frac{uv}{(1+u^2+v^2)^{3/2}} \\ -\frac{u}{(1+u^2+v^2)^{3/2}} & -\frac{v}{(1+u^2+v^2)^{3/2}} \end{pmatrix}$$

Figure 20: Red: the unit normal vectors along the curve (t, t, t^2). Green: the unit length tangent vector $\vec{v}$. Blue: $dN_p(\vec{v})$. In this case, $\vec{v}$ is an eigenvector of dN_p.



This looks complex, but at $(u, v) = (0, 0)$ the matrix simplifies to

$$\begin{pmatrix} 0 & -1 \\ -1 & 0 \\ 0 & 0 \end{pmatrix}$$

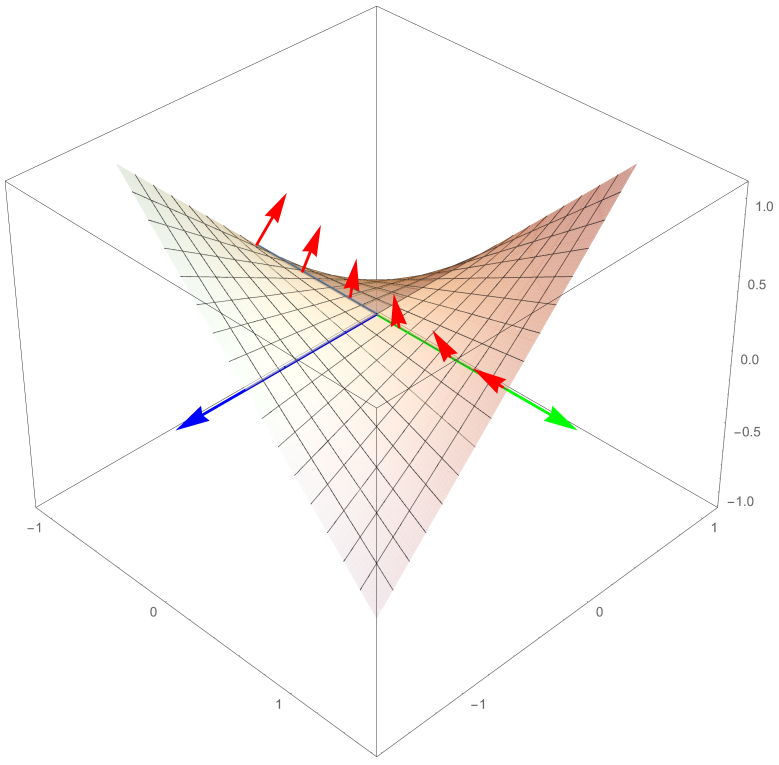
To compute $dN_p(\vec{v})$, recall that we must express $\vec{v}$ in the tangent space basis. Since we know that the tangent space basis is $\{(1, 0, 0), (0, 1, 0)\}$, we can express $\vec{v}$ as $\sqrt{2}/2(1, 1)$. Then, $dN_p(\vec{v}) = \sqrt{2}/2(-1, -1)$, which back in $\mathbb{R}^3$

corresponds to the vector $\sqrt{2}/2(-1,-1,0)$. We lastly compute

$$\begin{aligned} -\langle dN_p(\vec{w}),\ \vec{w}\rangle &= \langle \sqrt{2}/2(-1,-1,0),\ \sqrt{2}/2(1,1,0)\rangle \\ &= -(1/2)((-1,-1,0)\cdot(1,1,0)) \\ &= 1 \end{aligned}$$

which coincides with the value of normal curvature which we computed in Example 4.32.

Figure 21: Red: the unit normal vectors along the curve $(t,0,0)$. Green: the unit length tangent vector $\vec{v}$. Blue: $dN_p(\vec{v})$. In this case, $dN_p(\vec{v})$ is orthogonal to $\vec{v}$ and so the normal curvature of S at p in the direction $\vec{v}$ is 0.



To finish this section, we will lastly demonstrate how to compute the coefficients of the second fundamental form given a parameterization of a

regular surface. The following example is taken with commentary from do Carmo [2, pp. 154].

Examples 4.37. Let $\mathbb{X}(u,v)$ be a paramterization at a point $p \in S$ of a regular surface S, and let $\alpha(t) = \mathbb{X}(u(t), v(t))$ be a parameterized curve on S with $\alpha(0) = p$. By the chain rule, we have that the tangent vector to $\alpha(t)$ at p is

$$\alpha' = \mathbb{X}_u u' + \mathbb{X}_v v', \quad \text{and}$$
$$dN_p(\alpha') = N'(u(t), v(t)) = N_u u' + N_v v'.$$

The expression of the second fundamental form in the basis $\{\mathbb{X}_u, \mathbb{X}_v\}$ is given by

$$\begin{aligned} II_p(\alpha') &= -\langle dN_p(\alpha'),\ \alpha' \rangle = -\langle N_u u' + N_v v',\ \mathbb{X}_u u' + \mathbb{X}_v v' \rangle \\ &= -\langle N_u,\ \mathbb{X}_u \rangle (u')^2 - \langle N_u,\ \mathbb{X}_v \rangle u'v' - \langle N_v,\ \mathbb{X}_u \rangle v'u' - \langle N_v,\ \mathbb{X}_v \rangle (v')^2 \end{aligned}$$

Now, we said that the second fundamental form is a quadratic form, so we should have that $\langle N_u,\ \mathbb{X}_v \rangle = \langle N_v,\ \mathbb{X}_u \rangle$. To show this, recall that the tangent space basis vectors $\mathbb{X}_u$ and $\mathbb{X}_v$ are both orthogonal to the unit normal vector N. Then, by taking the partial derivatives with respect to u and v and using Proposition 4.22, we have that

$$\begin{aligned} \frac{\partial}{\partial u}\langle N,\ \mathbb{X}_u \rangle = 0 \quad &\Longrightarrow \quad \langle N,\ \mathbb{X}_{uu} \rangle = -\langle N_u,\ \mathbb{X}_u \rangle, \\ \frac{\partial}{\partial v}\langle N,\ \mathbb{X}_u \rangle = 0 \quad &\Longrightarrow \quad \langle N,\ \mathbb{X}_{uv} \rangle = -\langle N_v,\ \mathbb{X}_u \rangle, \qquad (1) \\ \frac{\partial}{\partial u}\langle N,\ \mathbb{X}_v \rangle = 0 \quad &\Longrightarrow \quad \langle N,\ \mathbb{X}_{vu} \rangle = -\langle N_u,\ \mathbb{X}_v \rangle, \qquad (2) \\ \frac{\partial}{\partial v}\langle N,\ \mathbb{X}_v \rangle = 0 \quad &\Longrightarrow \quad \langle N,\ \mathbb{X}_{vv} \rangle = -\langle N_v,\ \mathbb{X}_v \rangle. \end{aligned}$$

Then, since our mapping $\mathbb{X}$ is differentiable, we have that by the equality of mixed partial derivatives (Clairaut's theorem) $\mathbb{X}_{uv} = \mathbb{X}_{vu}$ which means that $\langle N,\ \mathbb{X}_{uv} \rangle = \langle N,\ \mathbb{X}_{vu} \rangle$. Because of this, we can combine equations (1) and (2) in the above computations as well as use the fact that the inner product is symmetric to write $\langle N_u,\ \mathbb{X}_v \rangle = \langle N_v,\ \mathbb{X}_u \rangle$. With all of this, we can continue where we left off above and say that

$$\begin{aligned} II_p(\alpha') &= -\langle N_u,\ \mathbb{X}_u \rangle (u')^2 - \langle N_u,\ \mathbb{X}_v \rangle u'v' - \langle N_v,\ \mathbb{X}_u \rangle v'u' - \langle N_v,\ \mathbb{X}_v \rangle (v')^2 \\ &= e(u')^2 + 2fu'v' + g(v')^2 \end{aligned}$$

where

$$\begin{aligned} e &= -\langle N_u,\ \mathbb{X}_u\rangle = \langle N,\ \mathbb{X}_{uu}\rangle \\ f &= -\langle N_u,\ \mathbb{X}_v\rangle = -\langle N_v,\ \mathbb{X}_u\rangle = \langle N,\ \mathbb{X}_{uv}\rangle = \langle N,\ \mathbb{X}_{vu}\rangle \\ g &= -\langle N_v,\ \mathbb{X}_v\rangle = \langle N,\ \mathbb{X}_{vv}\rangle \end{aligned}$$

Definition 4.38. The *coefficients of the second fundamental form* are the differentiable functions $e(u,v), g(u,v)$, and $f(u,v)$ defined above.

Just as with the coefficients of the first fundamental form, it is important to notice that they are computed with respect to a particular choice of parameterization $\mathbb{X}$.

Examples 4.39. We will compute the coefficients of the second fundamental form for the hyperbolic paraboloid continuing with the parameterization that we used in Example 4.36. Recall that

$$\mathbb{X}_u = (1,0,v) \quad \text{and} \quad \mathbb{X}_v = (0,1,u)$$

and our choice of unit normal vector field was such that

$$N(q) = (1+u^2+v^2)^{-1/2}(-v,-u,1).$$

Instead of differentiating our unit normal vector field like we did in the previous example, according to Definition 4.38 we can compute the dot product of the second partials of our mapping $\mathbb{X}$. Then,

$$\mathbb{X}_{uu} = (0,0,0),\ \mathbb{X}_{uv} = (0,0,1),\ \text{and } \mathbb{X}_{vv} = (0,0,0)$$

so

$$\begin{aligned} e(u,v) &= \langle N,\ \mathbb{X}_{uu}\rangle = 0 \\ f(u,v) &= \langle N,\ \mathbb{X}_{uv}\rangle = (1+u^2+v^2)^{-1/2} \\ g(u,v) &= \langle N,\ \mathbb{X}_{vv}\rangle = 0 \end{aligned}$$

Notice that expressing the second fundamental form in the tangent space basis using the coefficients of the second fundamental form makes computing normal curvature much easier than before. To compute the normal curvature

at the point $p = (0,0,0) = \mathbb{X}(0,0)$ in the direction $\vec{v} = \sqrt{2}/2(1,1,0)$ we now need only compute

$$\begin{aligned} II_p(\vec{v}) &= e(0,0)(u')^2 + 2f(0,0)u'v' + g(0,0)(v')^2 \\ &= 2f(0,0)(\sqrt{2}/2)(\sqrt{2}/2) \\ &= 1 \end{aligned}$$

That was so much easier than our first or second computation!

As we have seen, the second fundental form is a tool which allows us to measure the curvature of a surface at a point along a given direction.

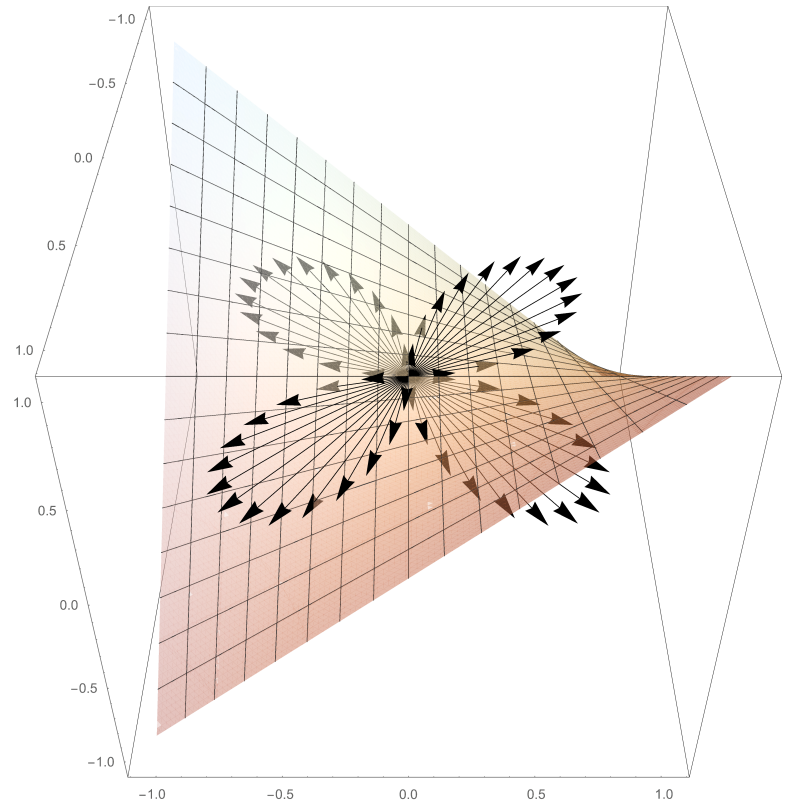
4.9 More On Curvature

Recalling that the differential is a linear transformation, we might wonder if dN_p has eigenvectors and if so what the eigenvectors and corresponding eigenvalues tell us about how our surfaces are curving. Because the linear algebra that lets us investigate the eigenvalues and eigenvectors of dN_p is not relevant to my project, I will state the results without proof. These results help give us a really solid understanding of the local curvature properties of surfaces.

According to do Carmo, one can show that $\langle dN_p(\vec{w}),\ \vec{v}\rangle = \langle \vec{w},\ dN_p(\vec{v})\rangle$ when $\vec{v}$ and $\vec{w}$ are orthogonal. This implies that the matrix for dN_p with respect to an orthonormal basis is symmetric, which implies its eigenvectors are orthogonal. Suppose that $\{\vec{e_1}, \vec{e_2}\}$ is an orthonormal basis for dN_p. Then, the corresponding eigenvalues $dN_p(e_1) = -k_1 e_1$ and $dN_p(e_2) = -k_2 e_2$ are the maximum and minimum values of the second fundamental form restricted to the unit circle of $T_p(S)$. In this fashion, we know that the maximum and minimum values of the second fundamental form restricted to the tangent space unit circle are the maximum and minimum normal curvatures as well as correspond to orthogonal tangent vectors. This is intruiging because it means that at every point on a regular surface, the directions corresponding to the maximum and minumum normal curvatures are orthogonal. Furthermore, these eigenvectors and eigenvalues describe the normal curvature of every unit length tangent vectors. This insight from linear algebra leads us to the following definition from do Carmo [2, pp. 144].

Definition 4.40. The maximum normal curvature k_1 and the minimum normal curvature k_2 are called the *principle curvatures* at p. The corresponding

Figure 22: On the hyperbolic paraboloid, we can visualize the principle curvatures by looking at each of the unit length tangent vectors scaled by their corresponding measure of normal curvature.



directions, that is, the directions given by the eigenvectors e_1 and e_2 are called the *principle directions* at p.

Recall that the sign of the normal curvature k_n along the direction of a unit tangent vector v at a point p tells us whether the surface curves upwards or downwards as we travel along the direction v. When the sign of the normal curvature k_n is negative, we say that S behaves like a "downwards parabola" while moving along v; when the sign of k_n is positive, we say that S behaves like an "upwards parabola" while moving along v. Since the principle directions and principle curvatures at a point completely describe a surface locally, we can use their values to classify the different types of points of

regular surfaces. This leads us to the following definitions adapted from do Carmo [2, pp. 146].

Definition 4.41. Let $p \in S$ and let $dN_p : T_p(S) \to T_p(S)$ be the differential of the unit normal vector field. The determinant of dN_p or the product of the principle curvatures is the *Gaussian curvature* of S at p, denoted $K(p)$.

Suppose that we have a point p on a surface S for which the Gaussian curvature is positive. Since the Gaussian curvature k is equivalent to the product of the principle curvatures $k_1 \cdot k_2$, we have that either both of k_1 and k_2 are positive or both of k_1 and k_2 are negative. Because of both the facts that

1) The sign of a normal curvature tells us whether or not a surface curves upwards or downwards as we travel along a given direction, and

2) The principle diretions along with the principle curvatures completely determine the normal curvature values for any other given tangent vector,

we have that around the point p the surface S either completely "curves upwards" or "curves downwards." Locally, these points look like points belonging to a stretched sphere, also known as an ellipsoid. For this reason, we call such points with positive Gaussian curvature *elliptical points*.

On the other hand, suppose that we have negative Gaussian curvature. Then, one of k_1 and k_2 is positive and the other must be negative. Assume that k_1 is positive and k_2 is negative. In this fashion, we have that around the point p the surface S "curves upwards" along the direction vector associated with k_2 and "curves downwards" along the direction vector associated with k_1. Locally, these points look like the points belonging to a saddle shape, also known as a hyperboloid. For this reason, we call such points with negative Gaussian curvature *hyperbolic points*.

Now, suppose that we have zero Gaussian curvature. Then, one of k_1 and k_2 is zero. Suppose that k_1 is zero and k_2 is nonzero. Then, we have that around the point p the surface S "curves upwards" or "curves downwards" along the direction vector associated with k_2 and does not curve at all along the direction vector associated with k_1. Locally, these points look like the points beloinging to an extruded parabola, where along one direction the surface curves upwards or downwards and along the other the surface remains flat. For this reason, we call such points with negative Gaussian curvature

parabolic points. Lastly, if both the principle curvatures are zero, then our surface S does not curve at all around the point p. Locally, these points look like points belonging to planes. For this reason, we call such points *planar points*. This classification leads us to the following definitions.

Definition 4.42. A point p of a surface S is called

1. *Elliptic* if $K(p) > 0$
2. *Hyperbolic* if $K(p) < 0$
3. *Parabolic* if $K(p) = 0$ with one of k_1 or k_2 nonzero
4. *Planar* if $K(p) = 0$ with $k_1 = k_2 = 0$

Now, suppose that at a point p on a surface S we have that the principle curvatures are equivalent; that is, we have that $k_1 = k_2$. We call such points *umbilical points*.

Definition 4.43. A point p of a surface S is called an *umbilical point* if the principle curvatures are equivalent; that is, if $k_1 = k_2$.

Following the two facts outlined in the classification of points (namely the interpretation of the sign of the normal curvature and how the principle directions and curvatures completely determine every possible normal curvature), we know that umbilical points locally behave like points of a sphere or a plane. That is, standing at an umbilical point, measuring the normal curvature reveals it is the same no matter which direction that one looks.

4.10 Parallel Transport

We have investigated and classified the curvature of points of regular surfaces, but one last problem that we will have to solve is that of movement on our surfaces. What does it mean to move in a 'straight path' on a curved surface? We will have to generalize what we mean by a 'straight path'. In order to do that, we will have to introduce something that we call the *covariant derivative*. We start with the following definitions from do Carmo [2, pp. 238].

Definition 4.44. A *vector field* in an open set $U \subset S$ of a regular surface S is a correspondence w that assigns to each $p \in U$ a vector $w(p) \in T_p(S)$. The vector field w is *differentiable* at p if, for some parameterizatoin $\mathbb{X}(u, v)$ in p, the components a and b of $w = a\mathbb{X}_u + b\mathbb{X}_v$ in the tangent space basis $\{\mathbb{X}_u, \mathbb{X}_v\}$ are differentiable functions at p.

We will use the covariant derivative to generalize what we mean by a 'line' on surfaces. Suppose that you are standing at the north pole of a sphere with your arm pointing right out in front of you. In this scenario, you will be a 'curve' lying in the sphere and your arm will be some 'vector' in the tangent space of that sphere where you are standing. Now, suppose that you walked forward until you reached the equator, keeping your arm in the same position the entire time. From a local perspective, you have walked forward in a 'line' of sorts, and your arm has been pointing in the same direction the whole time. However, from a global perspective, your arm was originally pointing in some direction tangent to the $xy-$plane, wheareas now it is pointing down parallel to the normal vector of that $xy-$plane.

We can interpret the motion of your arm along the path you traveled as that of a vector field restricted to a curve on a surface. What is special about the motion of your arm / tangent vector as you moved along that curve? This leads us to the following definition from do Carmo [2, pp. 238].

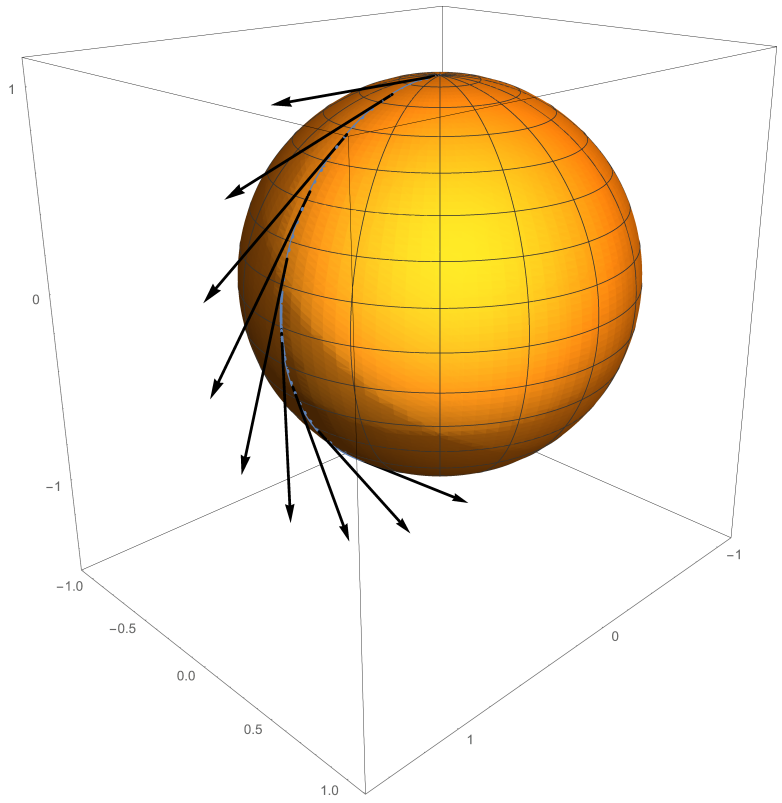
Definition 4.45. Let w be a differentiable vector field in an open set $U \subset S$ and $p \in U$. Let $y \in T_p(S)$. Consider a parameterized curve

$$\alpha : (-\epsilon, \epsilon) \to U,$$

with $\alpha(0) = p$ and $\alpha'(0) = y$, and let $w(t), t \in (-\epsilon, \epsilon)$ be the restriction of the vector field w to the curve α. The vector obtained by the normal projection of $(dw/dt)(0)$ onto the plane $T_p(S)$ is called the *covariant derivative* at p of the vector field w relative to the vector y. We denote this covariant derivative by $(Dw/dt)(0)$.

By walking from the north pole in a 'straight path' towards the equator, you traveled along a *great circle*, which is the intersection of some plane centered at the origin with the sphere. A great circle is just some 2-dimensional circle embedded in 3-dimensional space. Notice that the acceleration component of a curve parameterizing a circle points towards the center of that circle. That is, the acceleration component of a circle is normal to that circle at every point. This leads us to the following definition from do Carmo [2, pp. 238].

Figure 23: Demonstration of the parallel transport of your hand as a tangent vector while 'walking in a straight path' on the sphere.



Definition 4.46. A vector field w along a parameterized curve $\alpha : I \to S$ is said to be parallel if $Dw/dt = 0$ for all $t \in I$.

When the covariant derivative Dw/dt is 0, we have that the acceleration component of our vector field w along the curve exists only in the normal component of our surface. In this fashion, a 'straight path' lying in a surface is a path that has an acceleration vector parallel to the normal vector of the tangent space. We might be more formal and say that

Definition 4.47. Let $\alpha : I \to S$ by a parameterized curve and $w_0 \in T_{\alpha(t_0)}(S), t \in I$. Let w be the parallel vector field along α, with $w(t_0) = w_0$. The vector $w(t_1), t_1 \in I$ is called the *parallel transport* of w_0 along α at the point t_1.

Essentially, if we want to parallel transport a tangent vector some curve, we need to create or find a vector field that has our specified tangent vector as an initial condition where the vector field is parallel along our curve.

5 Creating the Curvature Experience

I wanted to create a virtual reality experience that allows a player to walk on the surface of an ellipsoid and use a compass and a minimap to gain an intuition of curvature and find umbilical points. In order to do this, I needed to figure out how to render the ellipsoid, position the player, move the player, compute curvature, render the compass, render the minimap, and create a tutorial that would make the experience meaningful an interesting to the player.

5.1 Rendering the Surface

The first problem I had to solve was that of rendering the ellipsoid I wanted the player to walk on. In computer graphics, one approach to modeling an object is that of using a *tessellation* to construct a triangle mesh representation of that object. Tessellation makes a lot of sense because we can use a parameterization (Definition 4.5) that describes the ellipsoid in order to construct triangle meshes approximating the surface. A tessellation essentially takes the domain of some parameterization $\mathbb{X} : U \subset \mathbb{R}^2 \to S \subset \mathbb{R}^3$, 'slices' it up into a bunch of rectangles comprised of triangles, and uses $\mathbb{X}$ to map each triangle to its corresponding location on the surface of S.

The tessellations that I use in my project take the form of a parameterization $\mathbb{X} : U \subset \mathbb{R}^2 \to S \subset \mathbb{R}^3$ where

$$U = \{(u, v) \in \mathbb{R}^2 \mid u_{min} \leq u \leq u_{max},\ v_{min} \leq v \leq v_{max}\}$$

is a closed rectangular subset of $\mathbb{R}^2$. To tessellate the surface, one specifies how many 'slices' we would like to take in the u and v direction. Because increasing the number of slices increases the quality of the approximation of the surface, we call the number of u and v slices $uRes$ and $vRes$ respectively. We use these two numbers to subdivide the rectangular region U with a total number of $(uRes + 1) \cdot (vRes + 1)$ vertices. Since this subdivided rectangular region is composed of $uRes \cdot vRes$ smaller rectangles, and each rectangle is composed of two triangles, we will create $2 \cdot uRes \cdot vRes$ triangles in total.

The pseudocode explaining the tessellation of my algorithm implemented in *ParametricSurfaceTessellation.cs* is as follows.

foreach *rectangle in the subdivided rectangular region* **do**
    determine the corresponding four points on the surface;
    determine the two triangles comprising this rectangle;
    determine the normal of each vertex in this rectangle;
    write the rectangle vertices, normals, and triangles to the mesh;
end

Algorithm 1: Tessellation

I attatch the *TessellationObject.cs* script to an empty GameObject with a MeshFilter and MeshRenderer component to create a mesh representation of the ellipsoid just before any frame update function is called.

5.2 Positioning and Orienting the Player

After rendering the ellipsoid, the next problem I had to solve was that of positioning and moving around the player on the surface of the ellipsoid. The first thing that I had to to was create an abstract class representing a regular surface. In *RegularSurface.cs* I have defined such an abstract class, and in *ParametricSurfaceMapping.cs* I have defined a class which uses a *RegularSurface* to keep track of a system of 'local' and 'global' coordinates for a single point. Then, I define and extend the Ellipsoid class from the RegularSurface class in *RegularSurfaces.cs*. Because I will be using a parameterization of the ellipsoid to keep track of the player, I will use the phrases 'local point,' 'local tangent,' or 'local coordinates' when talking about the player's position and orientation in U, and I will use the phrases 'global point,' 'global tangent,' or 'global coordinates' when talking about the player's position and orientation in S.

Suppose that I want to position and orient my player at global point $p = \mathbb{X}(q)$ corresponding to the local point $q = (u_0, v_0)$. In order to do this, my abstract class *RegularSurface* should contain a method *Mapping(localPoint)* that is essentially the parameterization function $\mathbb{X}$ and a method *Pushforward(localPoint, localTangent)* that allows one to apply the pushforward (Definition 4.16) to a local tangent vector. We can use the *Mapping* method to compute p in terms of $q = (u_0, v_0)$, and we can use the *Pushforward* method compute the global tangent space basis vectors in order to find the

unit normal vector $N(p)$. To compute the global tangent space basis vectors, we can apply the pushforward to the canonical tangent space basis of $U, \{(1,0),(0,1)\}$. Instead of manually computing a formula for the unit normal vector field for each of our surfaces, we can have our program normalize the cross product of the basis of $T_p(S)$.

To keep track of all of this for a specific local point, the *ParametricSurfaceMapping* needs to keep track of a Vector2 *localPoint*, and the canonical tangent space basis of U in the form of Vector2 *localUTangent* = new Vector2(1, 0), and Vector2 *localVTangent* = new Vector2(0, 1). Then, the *ParametricSurfaceMapping* class uses the *Mapping* and *Pushforward* methods in an *UpdateGlobalCoordinates* method to update the Vector3 *globalPoint*, Vector3 *globalUTangent*, Vector3 *globalVTangent*, and Vector3 *globalNormal* fields respectively. This gives us the following algorithm.

globalPoint = mapping(localPoint);
globalUTangent = Pushforward(localPoint, localUTangent);
globalVTangent = Pushforward(localPoint, localVTangent);
globalNormal = Normalize(Cross(globalUTangent, globalVTangent));

Algorithm 2: Position and Orientation

With this functionality, we can set a local point and call *UpdateGlobalCoordinates* to have access to all of the information we need to orient a player at a given point on a parameterized surface. In the *PlayerController.cs* script, we position and orient the player by calling the

transform.SetPositionAndRotation(position, orientation)

command passing to the position parameter *globalPoint* and to the orientation

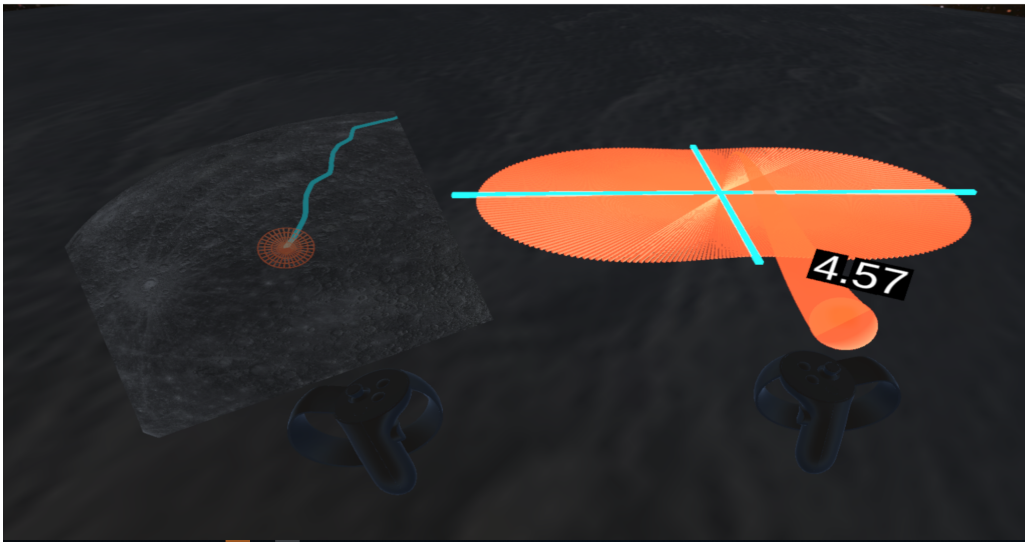
Quaternion.LookRotation(forwards, updwards),

where the forwards parameter is *globalUTangent* and the upwards parameter is *globalVTangent*.

In order to position and orient the player, we can attach the *PlayerController.cs* script to an empty GameObject and make the [CameraRig] prefab a child of the PlayerController. At this point, when we run our program the [CameraRig] will be initially positioned and oriented at a point on TessellationObject, and so the player will be able to locally walk and look around.

5.3 Moving the Player

Figure 24: The minimap shows the player's traveled path.



Now that we have a way of positioning and orienting the player given a local point, the next problem I had to solve was that of moving around the player. I could not simply map the controller joystick horizontal and vertical input to local point of the player, because then when the player is facing a certain direction and they press the joystick forward they are not guaranteed to move in the direction they are looking. How can we make it so that the player moves in the direction they are looking?

We can approach this by determining what local tangent vector the player's camera local forward orientation vector in the global tangent space corresponds to. We can compute our movement in U by adding this local tangent vector to our current local point. To do this, we can project the camera eye's forward vector into the global tangent plane, use the cross product with the normal to determine the player's 'right' tangent vector, and use the pullback of at the current local point to determine the corresponding local tangent vectors. This means that we will have to add a *Pullback* method to our abstract RegularSurface class. Then, we can linearly combine the pullback of these local forward and local right tangent vectors with the horizontal

and vertical components of the joystick input to determine the direction vector of our movement. Lastly, we can add this direction vector to the local point in order to move our player. This suggests the following algorithm.

forwardProjection = The projection of *cameraEye.forward* onto the plane defined by the normal vector globalNormal;
rightProjection = -Cross(forwardProjection, globalNormal);

forwardPullback = *Pullback(localPoint, forwardProjection)*;
rightPullback = *Pullback(localPoint, forwardProjection)*;

newLocalPoint = inputH $\cdot$ *rightPullback* + inputV $\cdot$ *forwardPullback*;
localPoint = newLocalPoint;

Algorithm 3: Movement I

In practice, this algorithm works except at the singular points of our parameterization. Because the tangent plane does not exist, when we use our positioning and orienting algorithm too close to a singular point we will not have a unit normal vector to orient the player. In the parameterization of the ellipsoid, we said that the 'north pole' $v = 0$ and 'south pole' $v = \pi$ as well as longitude $u = 0$ were irregular. One way that we can make sure a player does not walk through or enter an irregular point is by only updating the local point if the new local point does not reside in a small region bounding the irregular point. This tells us that our abstract RegularSurface should also have some method *InRegularSurface(localPoint)* that we can use to determine whether or not a player movement is valid. This suggests the following improvement to the movement algorithm, continuing just after we compute the new local point.

newLocalPoint = inputH $\cdot$ *rightPullback* + inputV $\cdot$ *forwardPullback*;

if *Not (InSingularRegion(newLocalPoint))* **then**
  localPoint = newLocalPoint;
end

Algorithm 4: Movement II

For the parameterization of the ellipsoid, it turns out that we only need

to make sure the player does not get to close to the north or south pole. Because Sine and Cosine are periodic, by not restricting $0 < u < 2\pi$ in our subset U our mapping is no longer a bijection but still gives us the correct position and tangent space information. In this case, instead of making sure our (u, v) coordinates are restricted to the specified open rectangular subset $U \subset \mathbb{R}^2$, we can take advantage of the periodicity of Sine and Cosine in order to only restrict v. This means that the implementation of the *InSingularRegion(localPoint)* method need only return true when v or $\pi - v$ is less than some delta threshold.

After this change to our movement algorithm, moving around the ellipsoid feels okay. However, experimentally as I walked around on the ellipsoid when its parameters made it a sphere, I did not move as I expected. Near the south pole, as I walked forward I would get 'dragged' to either my left or right. My advisor realized that even though we were being updated to the right position in each frame when we used our movement algorithm, we were not explicitly accounting for the parallel transport of the vectors defining the player's tangent plane. That is, after the player's position would be updated, their new orientation was not dependent on their previous orientation. Therefore, they were not locally 'moving in a straight path.'

In order to capture the parallel transport of the player's orientation, ideally we would like to parallel transport the player's tangent vector. However, it turns out that if we parallel transport a basis vector of a 2-dimensional tangent space, we also parallel transport a corresponding orthogonal tangent space basis vector. This means that parallel transporting any one vector of the tangent space can tell us how the entire tangent space parallel transports. I do not explicitly cover this in my section on parallel transport, but intuitively the reader can imagine that in the 'walking with your arm out' example, if you were to hold out your other arm to the right or left and walk in a 'straight path,' both your arms would remain orthogonal. To incorporate this into our program, instead of changing our movement algorithm, we can modify our position and orientation algorithm to keep track of and parallel transport the forward vector of the PlayerController GameObject.

Okay, but how exactly do we parallel transport a tangent vector? Before each movement, we can keep track of the forward orientation vector of the PlayerController. Then, after we compute where we move to, we want to know what the new forward orientation vector is at that point. Since at each frame we are updating our position in U by such a small change, we can approximate the parallel transport by projecting the old global forward

vector into the new tangent plane. Then, we can reorient the player based on this new global forward vector. This suggests the following unified algorithm, given a *localPoint* to initially position and orient the player.

Algorithm 2: Position and Orientation;
globalForward = *globalUTangent*;
orientation = Quaternion.LookRotation(*globalForward*, *globalNormal*);
SetPositionAndOrientation*(globalPoint, orientation)*;

while *moving* **do**
    forwardProjection = The projection of *cameraEye.forward* onto the plane defined by the normal vector globalNormal;
    rightProjection = -Cross(forwardProjection, globalNormal);

    forwardPullback = *Pullback(localPoint, forwardProjection)*;
    rightPullback = *Pullback(localPoint, forwardProjection)*;

    newLocalPoint = inputH · *rightPullback* + inputV · *forwardPullback*;
    localPoint = newLocalPoint;
    newLocalPoint = inputH · *rightPullback* + inputV · *forwardPullback*;

    if *Not (InSingularRegion(newLocalPoint))* **then**
        localPoint = newLocalPoint;
        Algorithm 2: Position and Orientation;
        globalForward = The projection of *globalForward* onto the plane defined by the normal vector *globalNormal*;
        orientation = Quaternion.LookRotation(*globalForward*, *globalNormal*);
        SetPositionAndOrientation*(globalPoint, orientation)*;
    end
end

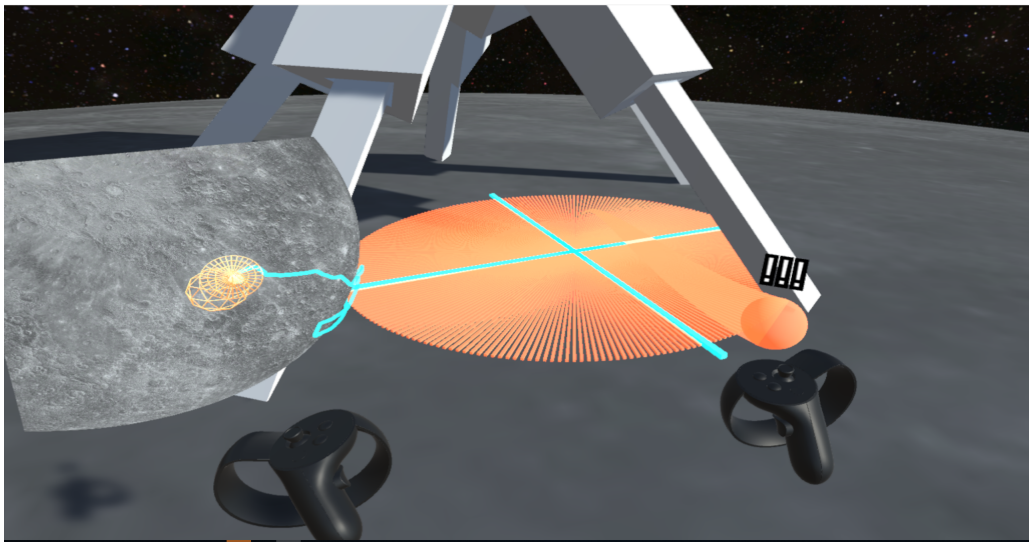
Algorithm 5: Movement III

With this algorithm, our player can now look and move in a direction such that the PlayerController's forward vector gets parallel transported. In this

fashion, when our player looks in a constant direction and moves forward they walk in a 'straight path,' and so our player can naturally move around on the ellipsoid.

5.4 Computing Curvature

Figure 25: A screencapture of the player discovering an umbilical point.



Now that our player can move naturally on the surface of the ellipsoid, the next problem I had to solve was that of constructing the curvature compass. Because normal curvature is a natural way of talking about the curvature of surfaces, I wanted my compass to let the viewer visualize an array of tangential normal curvatures as well as the principle directions at each point. To find the principle directions, I originally tried using Mathematica to compute the eigenvalues and eigenvectors of the matrix representing the differential of the unit normal vector field dN_p, but I found that quickly became complicated. If I want to make sure that my curvature compass works for an arbitrary implementation of the abstract RegularSurface class, precomputing the principle directions of a parameterized surface is a viable option but requires lots of computational time and memory. However, it turns out that

we can get away with not precomputing these values. Recall from Proposition 4.35 that the second fundamental form applied to a unit length tangent vector is the normal curvature in that direction. Also, recall from Section 4.9 that the maximum and minumum values of the second fundamental form applied to the tangent space unit circle are the principle curvatures. These facts together suggest that if our abstract RegularSurface class has methods that capture the coefficents second fundamental form, we can numerically compute our normal curvatures and approxiamate the principle directions.

We first add a method that computes the second fundamental form applied at a local point applied to a tangent vector called *Form2(localPoint, globalTangent)*. Then, to separate our curvature computations we define the RegularSurfaceCurvature class. An instance of RegularSurfaceCurvature keeps a reference to the RegularSurface object we are currently using as well as a ParametricSurfaceMapping object. For now, we can assume that the ParametricSurfaceMapping object is that of the PlayerController, and so its local point is that of the player. For a given local point, the RegularSurfaceCurvature object will determine the principle curvatures and directions by first computing the normal curvatures of a high quantity of evenly spaced tangent vectors in the tangent space unit circle. While we do this, we can keep track of the maximum and minimum normal curvatures. After we compute all of our normal curvatures, we know that the tangent vectors corresponding to the maximum and minimum normal curvatures are the principle directions. This suggests the following algorithm, given a positive integer *numDirections* for the number of tangent directions we would like to sample.

```
minNormalCurvature = 0;
maxNormalCurvature = 0;
deltaAngle = 360.0f / numDirections;
unitTangentVector = Normalize(globalUTangent);

for i = 0 to numDirections do
    angle = i · deltaAngle ;
    currentTangent = RotateAboutBy(unitTangentVector,
      globalNormal, angle);
    currentUnitTangent = Normalize(currentTangent);
    normalCurvature = NormalCurvature(currentUnitTangent);

    if i == 0 then
        minNormalCurvature = normalCurvature;
        maxNormalCurvature = normalCurvature;
    else
        if normalCurvature > maxNormalCurvature then
            maxNormalCurvature = normalCurvature;
        end

        if normalCurvature < minNormalCurvature then
            minNormalCurvature = normalCurvature;
        end
    end
end
```

Algorithm 6: Curvature Computation

In the above algorithm, if we store each of our (currentTangent, normalCurvature) pairs in an array and use two indices to keep track of which normalCurvature corresponds to the maximum and minimum normal curvatures, we then know the principle directions. I implement this exact algorithm in *RegularSurfaceCurvature.cs*. In my program, I use the difference of the principle curvatures as a way of telling how far the player is from a closest umbilical point. When the difference of the principle curvatures is less than a specified threshold, I say that the player has discovered an umbilical point.

5.5 The Curvature Compass

Now, given a RegularSurface and a ParametricSurfaceMapping corresponding to a point on that surface, we can compute the normal curvatures and principle curvatures. However, the curvature compass is designed so that the player can point it in any direction on the ground in front of them. To determine where we want to measure these normal curvatures, we initiate a raycast out of the player controller associated to the curvature compass towards the TessellationObject mesh. However, this raycast returns a 3-dimensional point on the surface of the ellipsoid; that is, it is a global point and not a local point. Because our curvature computations require a local point, we need to find a local point corresponding to the global point determined by the raycast in our choice of parameterization $\mathbb{X}$. We can take inspiration from multivariable calculus and do this by using an iterated tangent plane approximation. This suggests the following algorithm, given some global destination point globalDestination, some desired resulting minimum distance from our destination distThreshold, and a maximum number of tangent plane appriximations maxIterations.

```
compassMapping = new
 ParametricSurfaceMapping(playerMapping.localPoint,
 playerMapping.regularSurface);

iteration = 0;
dist = Dist(compassMapping.globalPoint, globalDestination);

while dist > distThreshold and iteration < maxIterations do
    globalDestinationDirection = globalDestination -
     compassMapping.globalPoint;
    globalDestinationProjection = The projection of
     globalDestinationDirection onto the plane defined by the normal
     vector compassMapping.globalNormal;

    localDestinationTangent = Pullback(compassMapping.localPoint,
     globalDestinationProjection);
    newLocalPoint = compassMapping.localPoint +
     localDestinationTangent;

    compassMapping.UpdateGlobalCoordinates(newLocalPoint);

    dist = Dist(compassMapping.globalPoint, globalDestination);
    iteration++;
end
```

Algorithm 7: Tangent Plane Pursuit

The tangent plane pursuit algorithm iteratively updates a ParametricSurfaceMapping to reflect the most recent tangent plane approximation of our destination, and bails out early if we get within a desired distance of our destination. This algorithm is the last peice needed to create all of the desired functionality of the curvature compass.

6 Conclusion

Both of the Curvature Experience and Ruled Surfaces Experience are finished projects; however, in this section I offer some ideas for fucture work.

With the Curvature Experience, I originally envisioned the player finding

umibilical points on two quadric surfaces other than the ellipsoid. Fischer explains that the two-sheeted hyperboloid has four umbilical points, and that the elliptic paraboloid has two. While working on the project I designed my program and class structure in such a way that it would not be complicated to implement the functionality for other parameterizations of regular surfaces. However, I did not end up having the time to implement them.

One other extension is a reconfiguration of my RegularSurface class. Ideally, we would not want to take advantage of the periodicity of a function and make sure that players do not walk into singular regions by restricting where the player can walk to. The definition of a regular surface (4.10) suggests how we might adapt our RegularSurface class and ParametricSurfaceMapping class to allow the player to walk across the the entirety of a regular surface.

With the Ruled Surface Experience, one problem that I was not able to solve was that of determining whether or not the player has 'correctly' drawn out a ruled surface to solve a challenge. Even though many of the people who gave feedback on my experiences enjoyed trying to create the challenges, sometimes players would just have fun creating their own ruled surface drawings, straying away from the game component of the experience. I do think that if there was a were to quantify and reward the players when they figured out how to construct a particular ruled surface, the players would know for sure when they have solved the problems.

As virtual math models, I found that working on and playtesting my project had me engaging with mathematics in similar ways to those of the physical models my projects were grounded in. In creating the curvature experience, I really had to sit down and gain an understanding of how mathematical quantities like the pushforward of a tangent vector and the expression of the second fundamental form in the tangent space basis are computed. The coding in my project forced me to think critically about vectors and the various ways that we represent them.

The ruled surface experience actually gave me an insight into the equations for the family of lines parameterizing the third challenge, Cayley's ruled surface. When I was working on creating the visual hints for the ruled surfaces challenges, I had to find parameterizations of the rulings of these surfaces so that I could have Unity draw these out in virtual space. While determining the rulings of the cylinder, hyperbolic paraboloid, helicoid, and one-sheeted hyperboloid is fairly straightforward, because Cayley's ruled surface is expressed as a cubic surface taking the form $3z - 3xy + x^3 = 0$, there

is no obvious set of rulings. When I was playtesting the ruled surfaces experience, I was trying to create Cayley's ruled cubic and I realized that to do so I needed to move my hands in such a way that looked like I was projecting a family of horizontal lines in the $xy-$plane onto my surface. When my advisor and I were trying to find the parameterization of the ruling the surface, the insight that I got from the embodied virtual reality experience ended up being the actual set of rulings for the surface. This is important because it demonstrates how the affordances of virtual reality can be used to offer insights into solutions of mathematical problems.

With some of the framework that I have produced, I can imagine extending the functionality of the curvature experience to explore surfaces with negative gaussian curvature. It turns out that because I capture parallel transport in the experience, when a player walks forward in a constant direction they actually trace out a curve called a *geodesic*, which is another fundamental curve in differential geometry. Geodesics are the 'straight paths' of curved surfaces, and my functionality allows one to trace out geodesics on parameterized regular surfaces. I imagine a future virtual reality experience where players have to walk in or draw geodesic curves in order to solve mazes or puzzles. Furthermore, such an experience might demonstrate how gaussian curvature affects the convergence and divergence of parallel geodesics.

In between the curvature experience and the ruled surfaces experience, I imagine another virtual reality experience where players can investigate how two different surfaces locally have the same inner product. That is, these players will investigate surfaces that have a bijective map between them that preserves the first fundamental form. Such a map is called an isometry, and being able to isometrically manipulate one surface into another seems like a good starting point for a project.

As we have seen, the affordances of virtual reality allow us to engage with mathematical models in similar but different ways to those of German mathematicians in the late 19^{th} century. Virtual reality allows our mathematical models to become mathematical experiences that are a lot more engaging and facilitate a conversation between the player and mathematics. My virtual reality projects explored this mathematical experience and demonstrated both how creating mathematical models can still be used as a process of mathematical learning but also how engaging with mathematical models can still be used for teaching purposes.

7 Acknowledgements

I would like to thank my advisor Professor French for supporting me with my initial ideas and iterative changes to each of my virtual reality experiences during the course of this project; thank Libby Farrell for giving me feedback on the mathematics portion of my paper; thank my friends that encouraged me at every stage of the MAP; thank the volunteers that gave some of their time to try out my experiences and give feedback on them; and lastly, I would like to thank Grinnell College for the funding that allowed me to work at the college this summer.

References

[1] Barth, W., Böhm, J., do Carmo, M. P., Fischer, G., Knörrer, H., Leiterer, J., Pinkall, U., Quaisser, E., and Reckziegel, H. *Mathematical models*. Friedr. Vieweg & Sohn, Braunschweig, 1986. Translated from the German.

[2] do Carmo, M. P. *Differential geometry of curves and surfaces*. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1976. Translated from the Portuguese.

[3] Stewart, J. *Calculus*, 4 ed. Brooks/Cole Publishing Company, C.A., 1999.

[4] Upara, C. Mathematical models, 1870-1930.