Differential Geometry - Exercise 8

Florian Bogner

May 29, 2020

1.

2. Let us collect a few facts:

   - The three sections are geodesics.
   - Example 4.32 can be generalized to the following: The vector field of the velocity vector turned by a constant angle within the tangent plane is parallel if the curve is a geodesic. This follows easily from Lemma 4.34.
   - It is sufficient to look at the transport of a basis, as the transport map is a linear map. The initial velocity vector $\dot{\gamma}_1(0)$ combined with itself turned by a right angle is a convenient basis for this.
   - The unit sphere is orientable. Therefore the transformation cannot be a mirroring and has to be a rotation. It therefore suffices to look at only one basis vector.

   So let us look at the initial velocity vector $\dot{\gamma}_1(0)$ and its parallel travel. [1] At the end of γ_1 it is pointing straight down. This gives us the new starting value for the parallel travel along γ_2. At the start it stands at a right angle to the new velocity vector of γ_2, so it does that too at the end of γ_2, coincidentally also pointing straight down. Finally, as the new starting vector for the travel along γ_3, it is offset by a half turn from the velocity vector, i.e. it is looking backwards along γ_3.

   Putting it all together: The vector starts pointing in the direction of γ_1 and ends pointing in the direction of γ_3. The angle between them is of course a right angle. Thus the transformation $\Pi_{\gamma_3} \circ \Pi_{\gamma_2} \circ \Pi_{\gamma_1}$ is a rotation by a right angle in the direction of γ_2.

3. We will use the parametrisation of the tractrix that we found in Exercise 3

   $$\sigma(u, v) = (u - \tanh u, \frac{1}{\cosh u} \cos v, \frac{1}{\cosh u} \sin v)$$

   As in the last exercises, we use Python and *sympy* to calculate the first and second fundamental forms, their determinants as well as the quotient which is of course the Gauss curvature. Surprisingly, it is indeed $K = -1$.

   Scaling the segment up by l also scales the resulting tractrix by l. Scaling a curve by l scales its curvature by $\frac{1}{l}$ as the curvature is the inverse of the osculating circle which gets also scaled by l. The Gauss curvature is the product of the two prinicipal curvatures, each scaled by $\frac{1}{l}$, thus the Gauss curvature gets scaled by $\frac{1}{l^2}$. Therefore $K_l = -\frac{1}{l^2}$.

4. Again with *sympy* we calculate the Gauss curvatures of σ and τ. They are

   $$K_\sigma = K_\tau = -\frac{1}{(u^2 + 1)^2}$$

[1] Technically we look at the vector field generated by it as in Theorem 4.33, but for the sake of easy language we talk about it moving.

To show that $\sigma \circ \tau^{-1}$ is not a path isometry we have to find a path γ_τ such that $\gamma_\sigma := \sigma \circ \tau^{-1} \circ \gamma_\tau$ has different length than γ_τ.

Let $u(t) := 1$ and $v(t) := t$ for $t \in [0, 2\pi]$. Let $\gamma_\tau(t) := \tau(u(t), v(t))$. Then we have

$$L(\gamma_\tau) = \int_0^{2\pi} \sqrt{\sin^2 t + \cos^2 t + 1^2}\, dt = \sqrt{8}\pi$$
$$L(\gamma_\sigma) = \int_0^{2\pi} \sqrt{\sin^2 t + \cos^2 t}\, dt = 2\pi \neq L(\gamma_\tau)$$

□

```
import sympy as sp
from sympy import diff, sqrt, sinh, cosh, tanh, cos, sin, log

class Vec3:
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z

    def __add__(self, other):
        return Vec3(self.x + other.x,
                    self.y + other.y,
                    self.z + other.z)

    def __mul__(self, scalar):
        return Vec3(scalar * self.x,
                    scalar * self.y,
                    scalar * self.z)

    def __truediv__(self, scalar):
        return Vec3(self.x / scalar,
                    self.y / scalar,
                    self.z / scalar)

    def diff(self, var):
        return Vec3(diff(self.x, var),
                    diff(self.y, var),
                    diff(self.z, var))

    def dot(self, other):
        return self.x * other.x + self.y * other.y + self.z * other.z

    def cross(self, other):
        return Vec3(self.y * other.z - self.z * other.y,
                    self.z * other.x - self.x * other.z,
                    self.x * other.y - self.y * other.x)

    def norm(self):
        return sqrt(self.x**2 + self.y**2 + self.z**2)
```

```
sp.init_printing(forecolor = 'White')
t, u, v = sp.symbols("t u v", real = True)

f = lambda t : 1 / cosh(t)
g = lambda t : t - tanh(t)


#Uncomment the right line for the different examples:

# Example 3
sig = Vec3(g(u), f(u)*cos(v), f(u)*sin(v))

# Example 4
#  sigma:
# sig = Vec3(u*cos(v), u*sin(v), log(u))
#  tau:
# sig = Vec3(u*cos(v), u*sin(v), v)

sig_u = sig.diff(u)
sig_v = sig.diff(v)

sig_uu = sig.diff(u).diff(u)
sig_uv = sig.diff(u).diff(v)
sig_vv = sig.diff(v).diff(v)

c = sig_u.cross(sig_v)

NN = c / c.norm()

E = sp.simplify(sig_u.dot(sig_u))
F = sp.simplify(sig_u.dot(sig_v))
G = sp.simplify(sig_v.dot(sig_v))

L = sp.simplify(sig_uu.dot(NN))
M = sp.simplify(sig_uv.dot(NN))
N = sp.simplify(sig_vv.dot(NN))

K = sp.simplify((L*N-M*M)/(E*G-F*F))
```