MATH 206, SPRING 2007, COMPUTER LAB 2

JOSHUA R. DAVIS

1. Introduction

This lab assumes all of the material from our first lab; you may need to go back and review. Here we learn a little more Mathematica, namely how to add explanatory text to our notebooks. Then we plot parametric surfaces. Then we use Mathematica's symbolic capabilities to automatically compute all of those pesky quantities such as $d\vec{x}^\top d\vec{x}$, $\partial\vec{x}/\partial u \times \partial\vec{x}/\partial v$, etc. At the end there are a few exercises to hand in (separately from your homework).

2. Writing Better Mathematica Notebooks

So far we've used only two kinds of cells in our Mathematica notebooks, namely input and output. These are the cells that do all of the work. But once you've done enough computations in Mathematica, your notebooks tend to get filled with a lot these little cells, some important and some not, some related to others and some not, etc. We need to keep our notebooks clean and understandable, in case we ever need to go back and use them again. So I've listed some suggestions here.

- Group related commands together in a single cell, so the reader understands that they're related.
- After you've figured out your problem, delete all unnecessary commands such as scratch work. Don't delete any essential commands, such as those that generate intermediate results used by later commands. But if you don't care about these intermediate results, then end the commands with a semicolon ; to suppress the output. You want to show your reader only the important stuff.
- Add text cells to explain what the Mathematica code means, how to use it, and what the output means. To add a text cell, just click the part of your notebook where you want to add the cell, go to the Format menu, go to its Style submenu, and select the Text style. When you do this, you notice that there are many styles to choose from; you can put in titles to demarcate sections and subsections, etc.

By following these suggestions, you can create a clean, self-explanatory notebook suitable for reading by others — a polished document, not just a collection of mysterious calculations. There are two ways that you might present the polished document to me (or your pals):

- You could work out all of the calculations, clean everything up, write your explanatory text, delete all of the output cells, and e-mail the notebook to me. Then I could open the notebook in Mathematica, evaluating the input cells for myself and seeing the output that they produce.
- Instead of e-mailing the notebook to me for me to run, you could go through the notebook and run it yourself, and then print out the whole thing, including the inputs, the outputs, and the explanations, on paper.

We will be using the second method in this lab. That is, I want you to submit your report on paper. More instructions can be found in the final section.

3. Visualizing Surfaces

In the first lab we plotted curves. Now we're going to plot surfaces. For starters, consider the torus of major radius a and minor radius r that you parametrized on a recent homework assignment. Your parametrization was probably something like this:

```
Clear[a, r];
torus[u_, v_] := {(a + r Cos[u]) Cos[v],
                  (a + r Cos[u]) Sin[v],
                  r Sin[u]};
```

(Notice that I clear `a` and `r` so that I'm sure they don't have any wacky values left over from previous computations. Also remember that when you define a function in Mathematica, the arguments must appear with underscores on the left-hand side.) We can plot the torus for $(u, v) \in (-\pi, \pi) \times (-\pi, \pi)$ using the same `ParametricPlot3D` command we used for curves; the only difference for surfaces is that there are two parameters, `u` and `v`:

```
a = 2;
r = 1;
ParametricPlot3D[torus[u, v], {u, -Pi, Pi}, {v, -Pi, Pi}]
```

This should give you a nice torus. Notice that Mathematica draws the surface by taking a rectangular mesh in the u-v-plane and mapping that mesh into $\mathbb{R}^3$. For our purposes this is fantastic! It gives us a visual sense of how the parametrization distorts the shape and size of little rectangles, which is exactly the area distortion phenomenon that we've discussed in class. In this case, the rectangles along the "inner rim" of the torus are pretty small, and those around the "outer rim" are pretty big. Later in the lab, we'll compute the area distortion factor to corroborate this picture.

The torus we've drawn is actually a little too nice, because the plot includes the values $u = \pm\pi$ and $v = \pm\pi$. To visualize the points on the torus that are missed when we don't include these values, try something like

```
b = Pi - 0.1;
ParametricPlot3D[torus[u, v], {u, -b, b}, {v, -b, b}]
```

As we saw on the homework, you need three such charts to cover the whole torus. I got my other two by phase-shifting u and v by $2\pi/3$:

```
ParametricPlot3D[torus[u + 2 Pi / 3, v + 2 Pi / 3],
                 {u, -b, b}, {v, -b, b}]
ParametricPlot3D[torus[u + 4 Pi / 3, v + 4 Pi / 3],
                 {u, -b, b}, {v, -b, b}]
```

Enough of the torus. Let's do stereographic projection:

```
stereo[u_, v_] := {(2 u)/(u^2 + v^2 + 1),
                   (2 v)/(u^2 + v^2 + 1),
                   (u^2 + v^2 - 1)/(u^2 + v^2 + 1)};
```

We can plot this on the square $(u, v) \in [-1, 1] \times [-1, 1]$ using the following command. (The PlotRange option explicitly declares that the graph is contained in the box $[-1, 1] \times [-1, 1] \times [-1, 1]$ in $\mathbb{R}^3$. When we do animations in a moment, this will help the individual plots line up correctly.)

```
a = 1;
ParametricPlot3D[stereo[u, v], {u, -a, a}, {v, -a, a},
                 PlotRange -> {{-1, 1}, {-1, 1}, {-1, 1}}]
```

By changing the value of `a` you can see more or less of the sphere. Instead of doing this by hand, let's make Mathematica's `Graphics‘Animation‘` package animate the process for us. The following commands generate a bunch of plots, with `a` running from 0.5 to 3.5 in increments of 0.1. Remember that to animate these plots, you first select them all, and then you go to the Cell menu and choose Animate Selected Graphics.

```
Clear[a];
Needs["Graphics‘Animation‘"];
Animate[ParametricPlot3D[stereo[u, v], {u, -a, a}, {v, -a, a},
                         PlotRange -> {{-1, 1}, {-1, 1}, {-1, 1}}],
        {a, 0.5, 3.5, 0.1}]
```

As `a` gets bigger and bigger, the parametrization covers more and more of the sphere. Notice that the little rectangles making up the sphere become quite small as you wander up to the north pole.

If you want help on the `Animate` command, I don't think you'll find it in Mathematica's help browser; instead, search the web for "Mathematica animation".

4. Symbolic Computation

Perhaps you are aware that computing quantities like $d\vec{x}^\top d\vec{x}$ by hand is instructive but tedious. Once you're comfortable doing it by hand, just make Mathematica do it. Here are some functions to help you.

```
Clear[u, v];
Clear[normSquared, norm, jacobian, jacTJac, areaDistortion];
normSquared[vector_] := Dot[vector, vector];
norm[vector_] := Sqrt[normSquared[vector]];
```

```
jacobian[param_] := Transpose[{D[param, u], D[param, v]}];
jacTJac[param_] := Transpose[jacobian[param]].jacobian[param];
areaDistortion[param_] := norm[Cross[D[param, u], D[param, v]]];
```

Notice that I'm calling $d\vec{x}$ the Jacobian, since that is the proper terminology. (The Jacobian is the matrix of partial derivatives, while $d\vec{x}$ is the linear transformation it represents in the standard basis.)

These commands produce the area distortion factor for the torus parametrization defined above:

```
Clear[a, r, u, v];
Simplify[areaDistortion[torus[u, v]]]
```

Mathematica gives me an answer of $\sqrt{r^2(a + r\cos u)^2}$. Assuming that $a > r$ (because otherwise the torus intersects itself in nasty ways), we can simplify the answer to $r(a + r\cos u)$. Notice that for $u \in (-\pi, \pi)$, the area distortion is largest at $u = 0$, which corresponds to the outer rim of the torus, and smallest as $u \to \pm\pi$, which corresponds to the inner rim of the torus. This explains why, in the torus plot we did earlier, the rectangular mesh was large along the outer rim and small along the inner rim.

Similarly, computing the area distortion factor for the `stereo` parametrization shows that the area gets small as (u, v) gets farther away from $(0, 0)$. Try it yourself. This corroborates the plot of `stereo` that we did earlier.

5. The Assignment

Don't hand in any of the work done above. Just hand in the stuff in this section, in a polished form. I mean polished in the sense of Section 2. There shouldn't be anything unnecessary; outputs that I don't care about should be suppressed using semicolons; related commands should probably be put together into a single cell; before and/or after each such group of commands, you should add a text cell to explain the commands and/or the output.

You might want to work everything out in one notebook, make a copy, and edit that copy down to be handed in.

5.1. **Southern Hemisphere.** Earlier in the lab we plotted a big chunk of the unit sphere using the `stereo` parametrization. Now, use this same parametrization to plot exactly the southern hemisphere of the sphere — no more, no less. Do this not by changing the definition of `stereo`, but by changing the arguments to `stereo` when it is used in the `ParametricPlot3D` command. If you do it as I expect you to, then the mesh that Mathematica draws on the surface will be different from the one it drew earlier for $(u, v) \in (-1, 1) \times (-1, 1)$.

Hand in the Mathematica code, the plot, and a brief description (in a text cell) of how the mesh is different from the one before.

5.2. **Area Distortion.** In the `areaDistortion` function above, we compute the area distortion factor as $|\partial\vec{x}/\partial u \times \partial\vec{x}/\partial v|$. In class we discussed another way to write it. This other way uses $d\vec{x}^{\top} d\vec{x}$, which is computed

by the function `jacTJac` above. Define a new Mathematica function called `areaDistortionNew` that computes the area distortion factor using `jacTJac`. Check that your new function produces the same results as `areaDistortion` for the `torus` and `stereo` parametrizations.

Hand in the Mathematica code for your `areaDistortionNew`.

5.3. **Unit Normal.** Write a function `normal` that takes in a parametrization `param` and outputs the unit normal vector $\vec{N}$ determined by that parametrization (by crossing $\partial\vec{x}/\partial u$ and $\partial\vec{x}/\partial v$ and normalizing). Test your function on `stereo` using this command:

```
Simplify[normal[stereo]]
```

The simplification might not go as far as you like. Explain why the normal really is the inward-pointing unit normal to the sphere.

Hand in the code for your `normal` function and the explanation. (You may write the explanation by hand, if you like; alternatively, you can type it in a text cell, if it doesn't require a lot of mathematical notation.)

5.4. **Differential of the Gauss Map of the Torus.** Any parametrization $\vec{x}$ induces a unit normal field $\vec{N}$ as in the preceding exercise, and this is essentially the Gauss map. We can understand its differential $d\vec{N}$ by its effect on the basis $\{\partial\vec{x}/\partial u, \partial\vec{x}/\partial v\}$. As we've discussed in class,

$$\begin{aligned} d\vec{N}\left(\tfrac{\partial\vec{x}}{\partial u}\right) &= \tfrac{\partial}{\partial u}\vec{N} &:= \tfrac{\partial}{\partial u}\left(\vec{N}\circ\vec{x}\right), \\ d\vec{N}\left(\tfrac{\partial\vec{x}}{\partial v}\right) &= \tfrac{\partial}{\partial v}\vec{N} &:= \tfrac{\partial}{\partial v}\left(\vec{N}\circ\vec{x}\right). \end{aligned}$$

Using the `torus` parametrization, the function `normal` from the preceding exercise, and a little differentiation, compute these six quantities for the torus in Mathematica:

$$\vec{N}, \quad d\vec{N}\left(\frac{\partial\vec{x}}{\partial u}\right), \quad d\vec{N}\left(\frac{\partial\vec{x}}{\partial v}\right),$$

$$\left\langle \vec{N}, d\vec{N}\left(\frac{\partial\vec{x}}{\partial u}\right)\right\rangle, \quad \left\langle d\vec{N}\left(\frac{\partial\vec{x}}{\partial u}\right), d\vec{N}\left(\frac{\partial\vec{x}}{\partial v}\right)\right\rangle, \quad \left\langle d\vec{N}\left(\frac{\partial\vec{x}}{\partial v}\right), \vec{N}\right\rangle.$$

Simplify each one using `Simplify`. The three dot products should simplify to 0.

Hand in the Mathematica commands needed for all of this, along with their simplified output. Also explain in a text cell why the first and third dot products should be zero. (We'll discuss the behavior of the second dot product later.)