Linear Approximations

Math 50C — Multivariable Calculus

David Arnold
David-Arnold@Eureka.redwoods.cc.ca.us

Abstract

Linear approximations of functions $g : R^2 \to R$ are explored through the visualization powers of MATLAB. A review of Taylor's theorem and its application to multivariable functions is employed to find linear approximations. *Prequisites: Taylor's theorem for functions* $g : R \to R$*. Partial derivatives. Some familiarity with* MATLAB*'s element wise operators.*

1 Introduction

This is an interactive document designed for online viewing. We've constructed this onscreen documents because we want to make a conscientious effort to cut down on the amount of paper wasted at the College. Consequently, printing of the onscreen document has been purposefully disabled. However, if you are extremely uncomfortable viewing documents onscreen, we have provided a print version. If you click on the Print Doc button, you will be transferred to the print version of the document, which you can print from your browser or the Acrobat Reader. We respectfully request that you only use this feature when you are at home. Help us to cut down on paper use at the College.

Much effort has been put into the design of the onscreen version so that you can comfortably navigate through the document. Most of the navigation tools are evident, but one particular feature warrants a bit of explanation. The section and subsection headings in the onscreen and print documents are interactive. If you click on any section or subsection header in the onscreen document, you will be transferred to an identical location in the print version of the document. If you are in the print version, you can make a return journey to the onscreen document by clicking on any section or subsection header in the print document.

Finally, the table of contents is also interactive. Clicking on an entry in the table of contents takes you directly to that section or subsection in the document.

1.1 Working with MATLAB

This document is a working document. It is expected that you are sitting in front of a computer terminal where the MATLAB software is installed. You are not supposed to read this document as if it were a short story. Rather, each time your are presented with a MATLAB command, it is expected that you will enter the command, then hit the Enter key to execute the command and view the result. Furthermore, it is expected that you will ponder the result. Make sure that you completely understand why you got the result you did before you continue with the reading.

2 Taylor's Theorem in Single Variable Calculus

You first introduction to Taylor's theorem probably occurred in your second calculus course (or in whatever course you first encountered series). Essentially, Taylor's theorem allows you to approximate a function $g : R \to R$ by an infinite series expansion.

$$g(x) = g(a) + g'(a)(x-a) + \frac{g''(a)}{2!}(x-a)^2 + \cdots + \frac{g^{(n)}(a)}{n!}(x-a)^n + \cdots \tag{1}$$

Of course, if you don't use all of the terms of the series (which in practice is impossible), you are only obtaining an approximation of the function *near* $x = a$. For example, a first order Taylor's expansion only uses terms of degree less than or equal to one.

$$g(x) = g(a) + g'(a)(x-a) \tag{2}$$

Let's use MATLAB to make this concept clear.

Example 1

Use MATLAB to approximate the function $g(x) = e^x$ with a first order Taylor's series near $x = 1$.

Solution. A first order Taylor's approximation only uses terms of degree less than or equal to one.

$$g(x) \approx g(1) + g'(1)(x-1) \tag{3}$$

Since $g(x) = e^x$ and $g'(x) = e^x$,

$$g(1) = e$$
$$g'(1) = e$$

which, when substituted into **equation 3**, yields

$$g(x) \approx e + e(x-1)$$

as the first order Taylor expansion of g near $x = 1$.

You can now use MATLAB to plot the function g together with its first order approximation $L(x) = e + e(x-1)$. The following MATLAB commands should produce an image similar to that in **Figure 1**.

```
close all
x=linspace(0,2);
g=exp(x);
L=exp(1)+exp(1)*(x-1);
plot(x,g,'b',x,L,'r',1,exp(1),'o')
grid on
```

Note that the linear approximation in **Figure 1** is very good near $x = 1$, but worsens as you move away from $x = 1$. Near $x = 2$ the approximation is not very good at all. If you wanted a first order Taylor's expansion that closely approximated the function g near $x = 2$, you would proceed as follows.

$$g(x) = g(2) + g(2)(x-2) \tag{4}$$

Again, $g(x) = e^x$ and $g'(x) = e^x$, so

$$g(2) = e^2$$
$$g'(2) = e^2$$

which, when substituted into **equation 4**, yields

$$g(x) = e^2 + e^2(x-2)$$

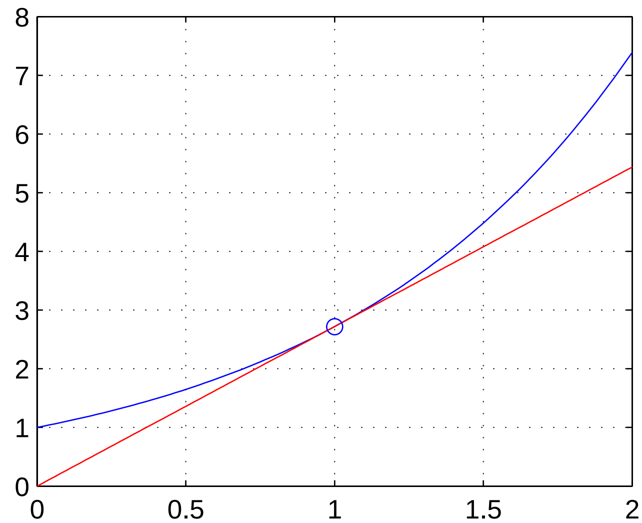

Figure 1 Linear approximation near $x = 1$.

The following MATLAB commands will plot the function g together with its linear approximation $L(x) = e^2 + e^2(x-2)$ near $x = 2$. The commands should produce an image similar to that in **Figure 2**.

```
close all
x=linspace(1,3);
g=exp(x);
L=exp(2)+exp(2)*(x-2);
plot(x,g,'b',x,L,'r',2,exp(2),'o')
grid on
```

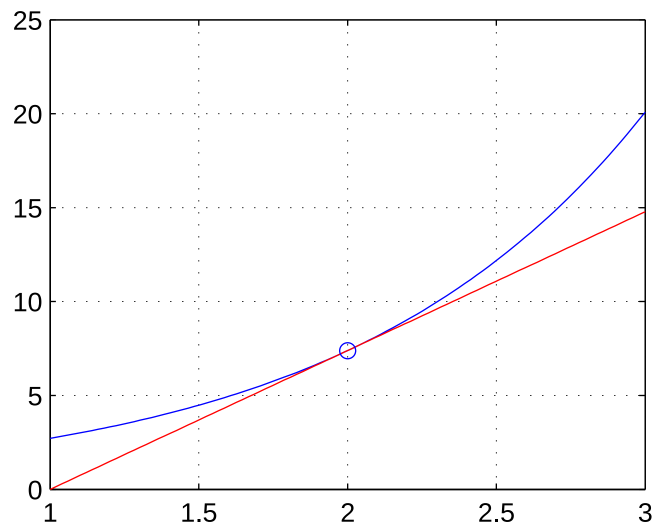

Figure 2 Linear approximation near $x = 2$.

Note that this linear approximation is excellent near $x = 2$ but deteriorates as you move away from $x = 2$.

3 Taylor's Theorem in Multivariable Calculus

Suppose that $g : R^2 \to R$. It's easy to find a first order Taylor's expansion of near the point (a, b).

$$g(x, y) \approx g(a, b) + g_x(a, b)(x - a) + g_y(a, b)(y - b) \tag{5}$$

Take a moment to note the similarities between **equation 5** and **equation 2**.

Example 2

Use MATLAB to approximate the function $g(x, y) = 4 - x^2 - y^2$ with a first order Taylor's expansion near the point $(x, y) = (1, 1)$.

Solution. A first order approximation near $(x, y) = (1, 1)$ is given by

$$g(x, y) = g(1, 1) + g_x(1, 1)(x - 1) + g_y(1, 1)(y - 1). \tag{6}$$

Since $g(x, y) = 9 - x^2 - y^2$, $g_x(x, y) = -2x$, and $g_y(x, y) = -2y$,

$$\begin{aligned} g(1, 1) &= 2 \\ g_x(1, 1) &= -2 \\ g_y(1, 1) &= -2 \end{aligned}$$

which, when substituted into **equation 6**, yields

$$g(x, y) = 2 - 2(x - 1) - 2(y - 1).$$

You can now use MATLAB to sketch the graph of $g(x, y) = 4 - x^2 - y^2$ together with its linear approximation $L(x, y) = 2 - 2(x - 1) - 2(y - 1)$. The following MATLAB commands should produce an image similar to that in **Figure 3**.

```
close all
[x,y]=meshgrid(0:.1:2);
z=4-x.^2-y.^2;
surf(x,y,z,ones(size(z)))
map=[.7 .7 .7;1 0 0];
colormap(map)
hold on
fx=-2; fy=-2;
L=2+fx*(x-1)+fy*(y-1);
mesh(x,y,L,2*ones(size(L)))
view([1,1,1])
hidden off
```

You can use the `rotate3d` command to rotate the figure into a position similar to that in **Figure 4**. You might find that other positions better show that the plane is tangent to the surface at the point $(1, 1)$.

4 Parametric Surfaces

A whole new world is opened when we allow for surfaces to be defined in terms of two parameters. Suppose for example, that we define a vector valued function with vector input as follows:

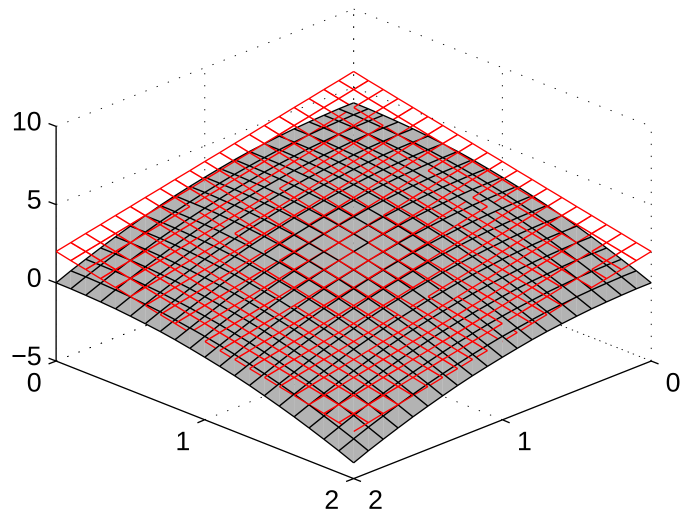

Figure 3 A linear approximation is a plane.

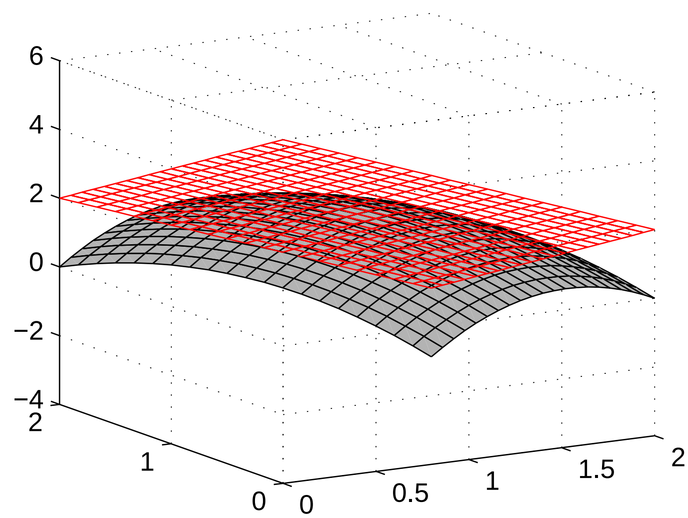

Figure 4 Rotate to another view.

$$\mathbf{r}(u,v) = < u, v, 9 - u^2 - v^2 > \tag{7}$$

The mapping $\mathbf{r}$ maps points in the uv-plane to points on a surface in xyz-space. That is, $\mathbf{r} : R^2 \to R^3$.

```
[u,v]=meshgrid(0:.1:2);
subplot(121)
plot(u,v,'r')
hold on
plot(u',v','b')
axis square
xlabel('u-axis')
```

```
ylabel('v-axis')
subplot(122)
plot3(x,y,z,'r')
hold on
plot3(x',y',z','b')
grid on
hidden
view([150,30])
axis square
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
```

Indeed, this code clearly demonstrates the manner in which **r** maps points (u, v) to points (x, y, z) on the surface. In **Figure 5**, note that the blue horizontal grid lines in the uv-plane, curves with constant v-value, are mapped onto the blue grid lines on the surface in R^3. Similarly, the red vertical grid lines in the uv-plane, curves with constant u-value, are mapped onto the red grid lines on the surface. In essence, the function **r** is is defining a system of grid lines on the surface in R^3, that is, a coordinate system for the surface.

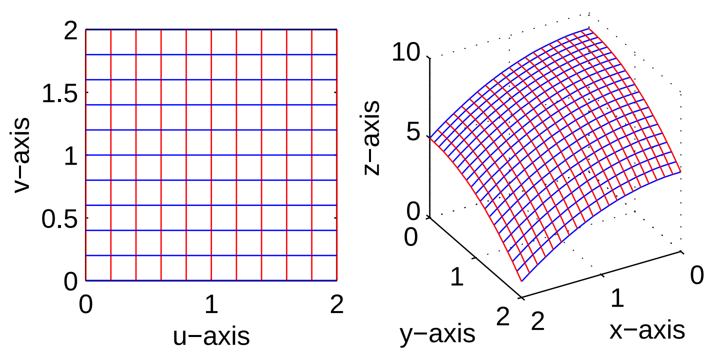

Figure 5 Mapping from the uv-plane to the surface in xyz-space.

It is a simple matter to take partial derivatives of **r** with respect to u and v.

$$\mathbf{r}_u(u, v) = < 1, 0, -2u >$$
$$\mathbf{r}_v(u, v) = < 0, 1, -2v >$$

What is not obvious is the geometric significance of these results. One would hope that all derivatives have something to do with the slopes of tangent lines, but we'll have to do a bit of snooping to understand the meaning of the partial derivatives of **r** with respect to u and v.

First, note that both $\mathbf{r}_u$ and $\mathbf{r}_v$ are *vectors*. Suppose that we evaluate each at the point $(u, v) = (1, 1)$.

$$\mathbf{r}_u(1, 1) = < 1, 0, -2 >$$
$$\mathbf{r}_v(1, 1) = < 0, 1, -2 >$$

Both derivatives are evaluated at the point $(1, 1)$, the first with v held fixed, the second with u held fixed. These fixed directions, and their images on the surface, are shown in **Figure 6**. The plot in the uv-plane is produced with this code.

```
subplot(121)
line([1,1],[0,2],'color','r')
line([0,2],[1,1],'color','b')
axis square
xlabel('u-axis')
ylabel('v-axis')
```

The surface on the right in **Figure 6** is drawn in the usual manner.

```
[u,v]=meshgrid(0:.2:2);
x=u; y=v; z=9-u.^2-v.^2;
subplot(122)
mesh(x,y,z,ones(size(z)))
hold on
map=[0.8,0.8,0.8];
colormap(map)
```

However, the next piece of code warrants some explanation. First, we use MATLAB's `find` command to "find" all points in the uv-plane having a u-value equal to 1. Then, we select the image of these points on the surface and plot them in the same color as the line $u = 1$ in the uv-plane, namely, red. Similarly, the blue curve on the surface is the image of the points (u, v) in the plane having $v = 1$.

```
k=find(u==1);
line(x(k),y(k),z(k),'color','r')
k=find(v==1);
line(x(k),y(k),z(k),'color','b')
```

Finally, a bit of formatting and labelling for that professional look.

```
axis square
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
view([150,30])
```

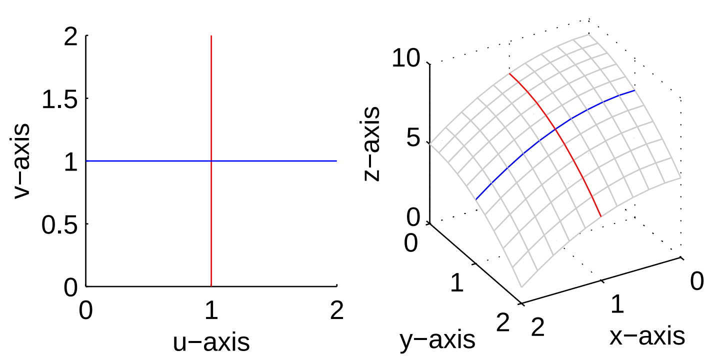

Figure 6 The partial derivatives are vectors tangent to the surface in the u and v-directions.

4.1 Tangent Vectors

Now, what about these vectors?

$$\mathbf{r}_u(1,1) = < 1, 0, -2 >$$
$$\mathbf{r}_v(1,1) = < 0, 1, -2 >$$

MATLAB's `quiver3` command is used to draw vectors in 3-space. The syntax `quiver3(x,y,z,dx,dy,dz,s)` is simple enough. At the points (x, y, z) are drawn vectors with displacement $< dx, dy, dz >$.[1] What we want to do is draw the vectors $\mathbf{r}_u(1,1)$ and $\mathbf{r}_v(1,1)$ at the point $(1, 1, 9 - 1^2 - 1^2)$, or $(1, 1, 7)$.

```
quiver3([1,1],[1,1],[7,7],[1,0],[0,1],[-2,-2],0.5)
```

This command produces the vectors shown in **Figure 7**. Notice that they are both tangent to the surface at the point $(1, 1, 7)$. Further, note that $\mathbf{r}_u(1,1) = < 1, 0, -2 >$ is pointing in the "direction" of the curve that is the image of the line in the uv-plane where v is held fixed (the blue curve). We might say that $\mathbf{r}_u(1,1)$ is the derivative *in the u-direction* at the point $(1, 1, 7)$. Similarly, the vector $\mathbf{r}_v(1,1) = < 0, 1, -2 >$ is pointing in the direction of the curve that is the image of the line in the uv-plane where u is held fixed (the red curve). We say that this is the derivative in the u-direction.

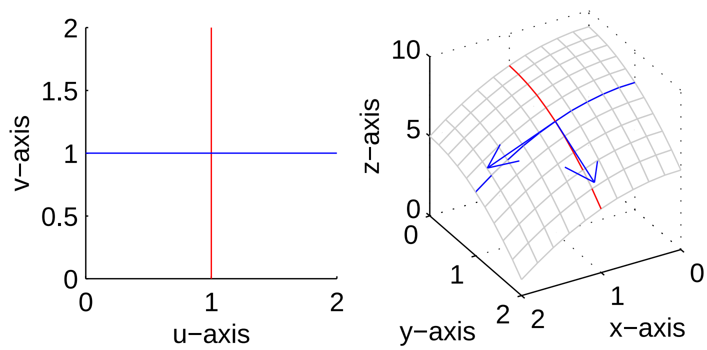

Figure 7 The partial derivatives of $\mathbf{r}$ with respect to u and v are vectors tangent to the surface in the u and v-directions, respectively.

4.2 The Vector Normal to the Surface

Of course, if we take the cross product of the vectors $\mathbf{r}_u(1,1)$ and $\mathbf{r}_v(1,1)$, we should get a vector that is *normal* or orthogonal to the surface at $(1, 1, 7)$.

$$\mathbf{r}_u(1,1) \times \mathbf{r}_v(1,1) = \begin{pmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 1 & 0 & -2 \\ 0 & 1 & -2 \end{pmatrix} = 2\mathbf{i} + 2\mathbf{j} + \mathbf{k},$$

or, equivalently, $< 2, 2, 1 >$. But is this vector normal to the surface at $(1, 1, 7)$?

[1] The argument `s` holds a scaling factor. Seldom do we want to draw the vectors at their actual length. However, we do want the vectors to have the correct *relative* lengths. Set $s = 0$ to turn off all scaling.

```
quiver3(1,1,7,2,2,1,0.5)
view([140,-10])
axis tight
grid off
```

It's necessary to use the `rotate` tool to obtain a view that make the cross product appear to be orthogonal to the surface at the point $(1, 1, 7)$. The command `view([140,-10])` provides such an orientation. The result is shown in **Figure 8**.

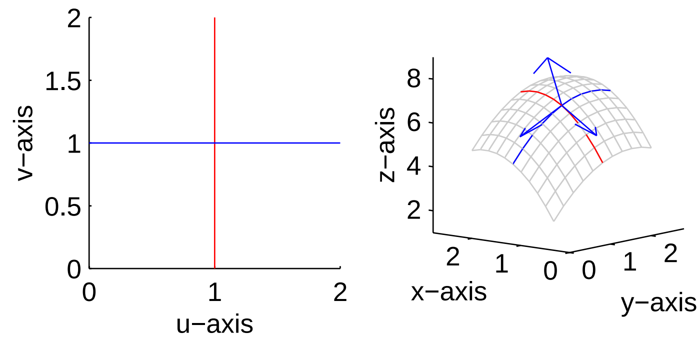

Figure 8 Adding $\mathbf{r}_u(1, 1) \times \mathbf{r}_v(1, 1)$.

4.3 The Tangent Plane

Of course, if we have a point on the surface, and a normal vector at that point, the equation of the plane tangent to the surface at that point is easily obtained.

$$\begin{aligned} < x - 1, y - 1, z - 7 > \cdot < 2, 2, 1 > &= 0 \\ 2x - 2 + 2y - 2 + z - 7 &= 0 \\ 2x + 2y + z - 11 &= 0 \end{aligned}$$

Solve this last equation for z.

$$z = 11 - 2x - 2y$$

The plane is easily parametrized in terms of u and v.

$$\begin{aligned} x &= u \\ y &= v \\ z &= 11 - 2u - 2v \end{aligned}$$

In this manner, we can use the same values of u and v currently stored in MATLAB's workspace. We need only add the plane to our graph in **Figure 8**.

```
x=u;
y=v;
z=11-2*u-2*y;
mesh(x,y,z,ones(size(z)))
hidden off
```

The results are shown in **Figure 9**. Clearly, the plane is tangent to the surface at the point $(1, 1, 7)$.

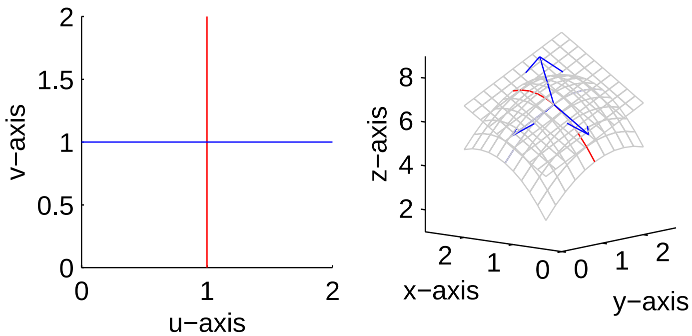

Figure 9 Adding the tangent plane.

5 Homework Exercises

1. Use Taylor's Theorem to find and sketch the plane tangent to the surface defined by

$$z = \sqrt{4 - x^2 - y^2}$$

at the point $(1, 1, \sqrt{2})$.

2. First, parametrize the surface in Exercise 1 by rotating the curve

$$\begin{aligned} y &= 2\cos u \\ z &= 2\sin u, \end{aligned}$$

$0 \le u \le \pi/2$, about the z-axis. Obtain a parametrization in terms of u and v, where v is the angle of rotation about the z-axis. Once you've obtained the parametrization, follow the lead in the narrative to sketch the surface, tangent and normal vectors, and the tangent plane at the point $(1, 1, \sqrt{2})$.