1 Applications with scalar functions

This section presents different applications of scalar functions. We begin by loading our package allowing access to several useful packages of `Julia`:

```
using CalculusWithJulia
using Plots
```

1.1 Tangent planes, linearization

Consider the case $f : R^2 \rightarrow R$. We visualize $z = f(x, y)$ through a surface. At a point (a, b), this surface, if f is sufficiently smooth, can be approximated by a flat area, or a plane. For example, the Northern hemisphere of the earth, might be modeled simplistically by $z = \sqrt{R^2 - (x^2 + y^2)}$ for some R and with the origin at the earth's core. The ancient view of a "flat earth," can be more generously seen as identifying this tangent plane with the sphere. More apt for current times, is the use of GPS coordinates to describe location. The difference between any two coordinates is technically a distance on a curved, nearly spherical, surface. But if the two points are reasonably closes (miles, not tens of miles) and accuracy isn't of utmost importance (i.e., not used for self-driving cars), then the distance can be found from the Euclidean distance formula, $\sqrt{(\Delta\text{latitude})^2 + \Delta\text{longitude})^2}$. That is, as if the points were on a plane, not a curved surface.

For the univariate case, the tangent line has many different uses. Here we see the tangent plane also does.

1.1.1 Equation of the tangent plane

The partial derivatives have the geometric view of being the derivative of the univariate functions $f(\vec{\gamma}_x(t))$ and $f(\vec{\gamma}_y(t))$, where $\vec{\gamma}_x$ moves just parallel to the x axis (e.g. $\langle t + a, b\rangle$). and $\vec{\gamma}_y$ moves just parallel to the y axis. The partial derivatives then are slopes of tangent lines to each curve. The tangent plane, should it exist, should match both slopes at a given point. With this observation, we can identify it.

Consider $f(\vec{\gamma}_x)$ at a point (a, b). The path has a tangent vector, which has "slope" $\frac{\partial f}{\partial x}$. and in the direction of the x axis, but not the y axis, as does this vector: $\langle 1, 0, \frac{\partial f}{\partial x}\rangle$. Similarly, this vector $\langle 0, 1, \frac{\partial f}{\partial y}\rangle$ describes the tangent line to $f(\vec{\gamma}_y)$ a the point.

These two vectors will lie in the plane. The normal vector is found by their cross product:

```
using SymPy
@vars f_x f_y
n = [1, 0, f_x] × [0, 1, f_y]
```

$$\begin{bmatrix} -f_x \\ -f_y \\ 1 \end{bmatrix}$$

Let $\vec{x} = \langle a, b, f(a,b)\rangle$. The tangent plane at $\vec{x}$ then is described by all vectors $\vec{v}$ with $\vec{n}\cdot(\vec{v} - \vec{x}) = 0$. Using $\vec{v} = \langle x, y, z\rangle$, we have:

$$[-\frac{\partial f}{\partial x}, -\frac{\partial f}{\partial y}, 1] \cdot [x - a, y - b, z - f(a,b)] = 0,$$

or,

$$z = f(a,b) + \frac{\partial f}{\partial x}(x - a) + \frac{\partial f}{\partial y}(y - b),$$

which is more compactly expressed as

$$z = f(a,b) + \nabla(f) \cdot \langle x - a, y - b\rangle.$$

This form would then generalize to scalar functions from $R^n \to R$. This is consistent with the definition of f being differentiable, where ∇f plays the role of the slope in the formulas.

The following figure illustrates the above for the function $f(x,y) = 6 - x^2 - y^2$:

```
f(x,y) = 6 - x^2 -y^2
f(x)= f(x...)

a,b = 1, -1/2


# draw surface
xr = 7/4
xs = ys = range(-xr, xr, length=100)
surface(xs, ys, f, legend=false)

# visualize tangent plane as 3d polygon
pt = [a,b]
tplane(x) = f(pt) + gradient(f)(pt) ⋅ (x - [a,b])

pts = [[a-1,b-1], [a+1, b-1], [a+1, b+1], [a-1, b+1], [a-1, b-1]]
plot!(unzip([[pt..., tplane(pt)] for pt in pts])...)

# plot paths in x and y direction through (a,b)
γ_x(t) = pt + t*[1,0]
γ_y(t) = pt + t*[0,1]

plot_parametric_curve!(t -> [γ_x(t)..., (f∘γ_x)(t)], -xr-a, xr-a, linewidth=3)
plot_parametric_curve!(t -> [γ_y(t)..., (f∘γ_y)(t)], -xr-b, xr-b, linewidth=3)

# draw directional derivatives in 3d and normal
pt = [a, b, f(a,b)]
fx, fy = gradient(f)(a,b)
arrow!(pt, [1, 0, fx], linewidth=3)
arrow!(pt, [0, 1, fy], linewidth=3)
arrow!(pt, [-fx, -fy, 1], linewidth=3) # normal

# draw point in base, x-y, plane
pt = [a, b, 0]
scatter!(unzip([pt])...)
arrow!(pt, [1,0,0], linestyle=:dash)
arrow!(pt, [0,1,0], linestyle=:dash)
```

Alternate forms The equation for the tangent plane is often expressed in a more explicit form. For $n = 2$, if we set $dx = x - a$ and $dy = y - a$, then the equation for the plane becomes:

$$f(a, b) + \frac{\partial f}{\partial x} dx + \frac{\partial f}{\partial y} dy,$$

which is a common form for the equation, though possibly confusing, as ∂x and dx need to be distinguished. For $n > 2$, additional terms follow this pattern. This explicit form is helpful when doing calculations by hand, but much less so when working on the computer, say with `Julia`, as the representations using vectors (or matrices) can be readily implemented and their representation much closer to the formulas. For example, consider these two possible functions to find the tangent plane (returned as a function) at a point in 2 dimensions

```
function tangent_plane(f, pt)
  fx, fy = ForwardDiff.gradient(f, pt)
  x -> f(x...) + fx * (x[1]-pt[1]) + fy * (x[2]-pt[2])
end
```

```
tangent_plane (generic function with 1 method)
```

It isn't so bad, but as written, we specialized to the number of dimensions, used indexing, and with additional dimensions, it clearly would get tedious to generalize. Using vectors, we might have:

```
function tangent_plane(f, pt)
  ∇f = ForwardDiff.gradient(f, pt) # using a variable ∇f
  x -> f(pt) + ∇f ⋅ (x - pt)
end
```

```
tangent_plane (generic function with 1 method)
```

This is much more like the compact formula and able to handle higher dimensions without rewriting.

1.1.2 Tangent plane for level curves

Consider the surface described by $f(x, y, z) = c$, a constant. This is more general than surfaces described by $z = f(x, y)$. The concept of a tangent plane should still be applicable though. Suppose, $\vec{\gamma}(t)$ is a curve in the $x - y - z$ plane, then we have $(f \circ \vec{\gamma})(t)$ is a curve on the surface and its derivative is given by the chain rule through: $\nabla f(\vec{\gamma}(t)) \cdot \vec{\gamma}'(t)$. But this composition is constantly the same value, so the derivative is 0. This says that $\nabla f(\vec{\gamma}(t))$ is *orthogonal* to $\vec{\gamma}'(t)$ for any curve. As these tangential vectors to $\vec{\gamma}$ lie in the tangent plane, the tangent plane can be characterized by having ∇f as the normal.

This computation was previously done in two dimensions, and showed the gradient is orthogonal to the contour lines (and points in the direction of greatest ascent). It can be generalized to higher dimensions.

The surface $F(x, y, z) = z - f(x, y) = 0$ has gradient given by $\langle -\partial f/\partial x, -\partial f/\partial y, 1\rangle$, and as seen above, this vector is normal to the tangent plane, so this generalization agrees on the easier case.

For clarity:

- The scalar function $z = f(x, y)$ describes a surface, $(x, y, f(x, y))$; the gradient, ∇f is 2 dimensional and points in the direction of greatest ascent for the surface.
- The scalar function $f(x, y, z)$ *also* describes a surface, through level curves $f(x, y, z) = c$, for some *constant* c. The gradient ∇f is 3 dimensional and *orthogonal* to the surface.

Example Let $z = f(x, y) = \sin(x)\cos(x - y)$. Find an equation for the tangent plane at $(\pi/4, \pi/3)$.

We have many possible forms to express this in, but we will use the functional description:

```
@vars x y
vars = [x, y]
f(x,y) = sin(x) * cos(x-y)
f(x) = f(x...)

gradf = diff.(f(x,y), vars)  # or use gradient(f, vars) or ∇((f,vars))

pt = [PI/4, PI/3]
gradfa = subs.(gradf, x.=>pt[1], y.=>pt[2])

f(pt) + gradfa ⋅ (vars - pt)
```

$$\left(x - \frac{\pi}{4}\right)\left(-\frac{\sqrt{2}\left(-\frac{\sqrt{6}}{4}+\frac{\sqrt{2}}{4}\right)}{2} + \frac{\sqrt{2}\left(\frac{\sqrt{2}}{4}+\frac{\sqrt{6}}{4}\right)}{2}\right) + \frac{\sqrt{2}\left(-\frac{\sqrt{6}}{4}+\frac{\sqrt{2}}{4}\right)\left(y-\frac{\pi}{3}\right)}{2} + \frac{\sqrt{2}\left(\frac{\sqrt{2}}{4}+\frac{\sqrt{6}}{4}\right)}{2}$$

Example A cylinder $f(x, y, z) = (x - a)^2 + y^2 = (2a)^2$ is intersected with a sphere $g(x, y, z) = x^2 + y^2 + z^2 = a^2$. Let V be the line of intersection. (Viviani's curve). Let P be a point on the curve. Describe the tangent to the curve.

We have the line of intersection will have tangent line lying in the tangent plane to both surfaces. These two surfaces have normal vectors given by the gradient, or $\vec{n}_1 = \langle 2(x - a), 2y, 0\rangle$ and $\vec{n}_2 = \langle 2x, 2y, 2z\rangle$. The cross product of these two vectors will lie in both tangent planes, so we have:

$$P + t(\vec{n}_1 \times \vec{n}_2),$$

will describe the tangent.

The curve may be described parametrically by $\vec{\gamma}(t) = a\langle 1+\cos(t), \sin(t), 2\sin(t/2)\rangle$. Let's see that the above is correct by verifying that the cross product of the tangent vector computed two ways is 0:

```
a = 1
gamma(t) = a * [1 + cos(t), sin(t), 2sin(t/2) ]
P = gamma(1/2)
n1(x,y,z)= [2*(x-a), 2y, 0]
n2(x,y,z) = [2x,2y,2z]
n1(x) = n1(x...)
n2(x) = n2(x...)

t = 1/2
(n1(gamma(t)) × n2(gamma(t))) × gamma'(t)
```

```
3-element Array{Float64,1}:
 0.0
 0.0
 0.0
```

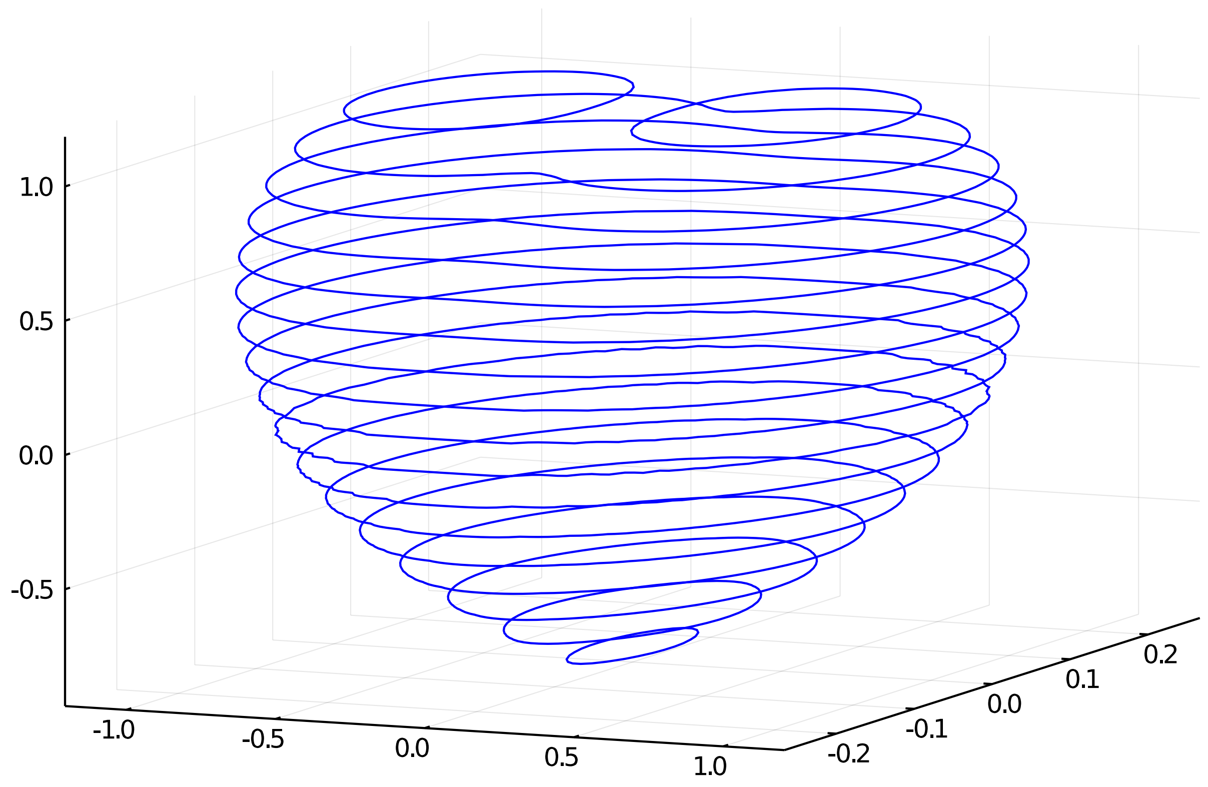
Plotting level curves of $F(x, y, z) = c$ The `wireframe` plot can be used to a surface of the type `z=f(x,y)`, as previously illustrated. However we have no way of plotting 3-dimensional implicit surfaces (of the type $F(x, y, z) = c$) as we do for 2-dimensional implicit surfaces. (However, within the `Makie` plotting framework one is provided in `CalculusWithJulia`.) The following function, which is **not** part of `CalculusWithJulia` can be used for this task to make a surface akin to the wireframe plot. The basic idea is to slice an axis, by default the z axis up and for each level plot the contours of $(x, y) \to f(x, y, z) - c$, which becomes a 2-dimensional problem. The function allows any of 3 different axes to be chosen to slice over, the default being just the z axis.

```
import Contour # installed with the Plots package, so should be available
               # import -- not using -- to avoid name collision
function plot_implicit_surface(F, c=0;
                      xlim=(-5,5), ylim=xlim, zlim=xlim,
                      nlevels=25,          # number of levels in a direction
                      slices=Dict(:z => :blue), # Dict(:x => :color, :y=>:color,
:z=>:color)
                      kwargs...            # passed to initial `plot` call
                      )

    _linspace(rng, n=150) = range(rng[1], stop=rng[2], length=n)

    X1, Y1, Z1 = _linspace(xlim), _linspace(ylim), _linspace(zlim)

    p = Plots.plot(;legend=false,kwargs...)

    if :x ∈ keys(slices)
        for x in _linspace(xlim, nlevels)
            local X1 = [F(x,y,z) for y in Y1, z in Z1]
            cnt = Contour.contours(Y1,Z1,X1, [c])
            for line in Contour.lines(Contour.levels(cnt)[1])
                ys, zs = Contour.coordinates(line) # coordinates of this line segment
                plot!(p, x .+ 0 * ys, ys, zs, color=slices[:x])
          end
        end
    end
    if :y ∈ keys(slices)
        for y in _linspace(ylim, nlevels)
            local Y1 = [F(x,y,z) for x in X1, z in Z1]
            cnt = Contour.contours(Z1,X1,Y1, [c])
            for line in Contour.lines(Contour.levels(cnt)[1])
                xs, zs = Contour.coordinates(line) # coordinates of this line segment
                plot!(p, xs, y .+ 0 * xs, zs, color=slices[:y])
            end
        end
    end

    if :z ∈ keys(slices)
        for z in _linspace(zlim, nlevels)
            local Z1 = [F(x, y, z) for x in X1, y in Y1]
            cnt = Contour.contours(X1, Y1, Z1, [c])
            for line in Contour.lines(Contour.levels(cnt)[1])
                xs, ys = Contour.coordinates(line) # coordinates of this line segment
                plot!(p, xs, ys, z .+ 0 * xs, color=slices[:z])
            end
        end
    end


    p
end
```

```
plot_implicit_surface (generic function with 2 methods)
```

We demonstrate with an example from a February 14, 2019 article in the New York Times. It shows an equation for a "heart," as the graphic will illustrate:

```
a,b = 1,3
f(x,y,z) = (x^2+((1+b)*y)^2+z^2-1)^3-x^2*z^3-a*y^2*z^3

plot_implicit_surface(f, xlim=(-2,2), ylim=(-1,1), zlim=(-1,2))
```



1.2 Linearization

The tangent plane is the best ”linear approximation” to a function at a point. ”Linear” refers to mathematical properties of the tangent plane, but at a practical level it means easy to compute, as it will involve only multiplication and addition. ”Approximation” is useful in that if a bit of error is an acceptable tradeoff for computational ease, the tangent plane may be used in place of the function. In the univariate case, this is known as linearization, and the tradeoff is widely used in the derivation of theoretical relationships, as well as in practice to get reasonable numeric values.

Formally, this is saying:

$$f(\vec{x}) \approx f(\vec{a}) + \nabla f(\vec{a}) \cdot (\vec{x} - \vec{a}).$$

The explicit meaning of $\approx$ will be made clear when the generalization of Taylor's theorem is to be stated.

Example: Linear approximation The volume of a cylinder is $V = \pi r^2 h$. It is thought a cylinder has $r = 1$ and $h = 2$. If instead, the amounts are $r = 1.01, h = 2.01$, what is the difference in volume?

That is, if $V(r, h) = \pi r^2 h$, what is $V(1.01, 2.01) - V(1, 2)$?

We can use linear approximation to see that this difference is *approximately* $\nabla V \cdot \langle 0.01, 0.01 \rangle$. This is:

```
V(r, h) = pi * r^2 * h
V(v) = V(v...)
a = [1,2]
dx = [0.01, 0.01]
ForwardDiff.gradient(V, a) ⋅ dx    # or use ∇(V)(a)
```

```
0.15707963267948966
```

The exact difference can be computed:

```
V(a + dx) - V(a)
```

```
0.15833941133357854
```

Example Let $f(x, y) = \sin(\pi xy^2)$. Estimate $f(1.1, 0.9)$.

Using linear approximation with $dx = 0.1$ and $dy = -0.1$, this is

$$f(1, 1) + \nabla f(1, 1) \cdot \langle 0.1, -0.1 \rangle,$$

where $f(1, 1) = sin(\pi) = 0$ and $\nabla f = \langle y^2 \cos(\pi xy^2), \cos(\pi xy^2)2y \rangle = \cos(\pi xy^2)\langle x, 2y \rangle$. So, the answer is:

$$0 + \cos(\pi)\langle 1, 2 \rangle \cdot \langle 0.1, -0.1 \rangle = (-1)(0.1 - 2(0.1)) = 0.1.$$

Example A piriform is described by the quartic surface $f(x, y, z) = x^4 - x^3 + y^2 + z^2 = 0$. Find the tangent line at the point $\langle 2, 2, 2 \rangle$.

Here, ∇f describes a *normal* to the tangent plane. The description of a plane may be described by $\hat{N} \cdot (\vec{x} - \vec{x}_0) = 0$, where $\vec{x}_0$ is identified with a point on the plane (the point $(2, 2, 2)$ here). With this, we have $\hat{N} \cdot \vec{x} = ax + by + cz = \hat{N} \cdot \langle 2, 2, 2 \rangle = 2(a + b + c)$. For ths problem, $\nabla f(2, 2, 2) = \langle a, b, c \rangle$ is given by:

```
f(x,y,z) = x^4 -x^3 + y^2 + z^2
a, b,c = ∇(f)(2,2,2)
"$a x + $b y  + $c z = $([a,b,c] ⋅ [2,2,2])"
```

```
"20 x + 4 y  + 4 z = 56"
```

1.2.1 Newton's method to solve $f(x, y) = 0$ and $g(x, y) = 0$.

The level curve $f(x, y) = 0$ and the level curve $g(x, y) = 0$ may intersect. Solving algebraically for the intersection may be difficult in most cases, though the linear case is not. (The linear case being the intersection of two lines).

To elaborate, consider two linear equations written in a general form:

$$ax + by = u \tag{1}$$
$$cx + dy = v \tag{2}$$

A method to solve this by hand would be to solve for y from one equation, replace this expression into the second equation and then solve for x. From there, y can be found. A more advanced method expresses the problem in a matrix formulation of the form $Mx = b$ and solves that equation. This form of solving is implemented in `Julia`, through the "backslash" operator. Here is the general solution:

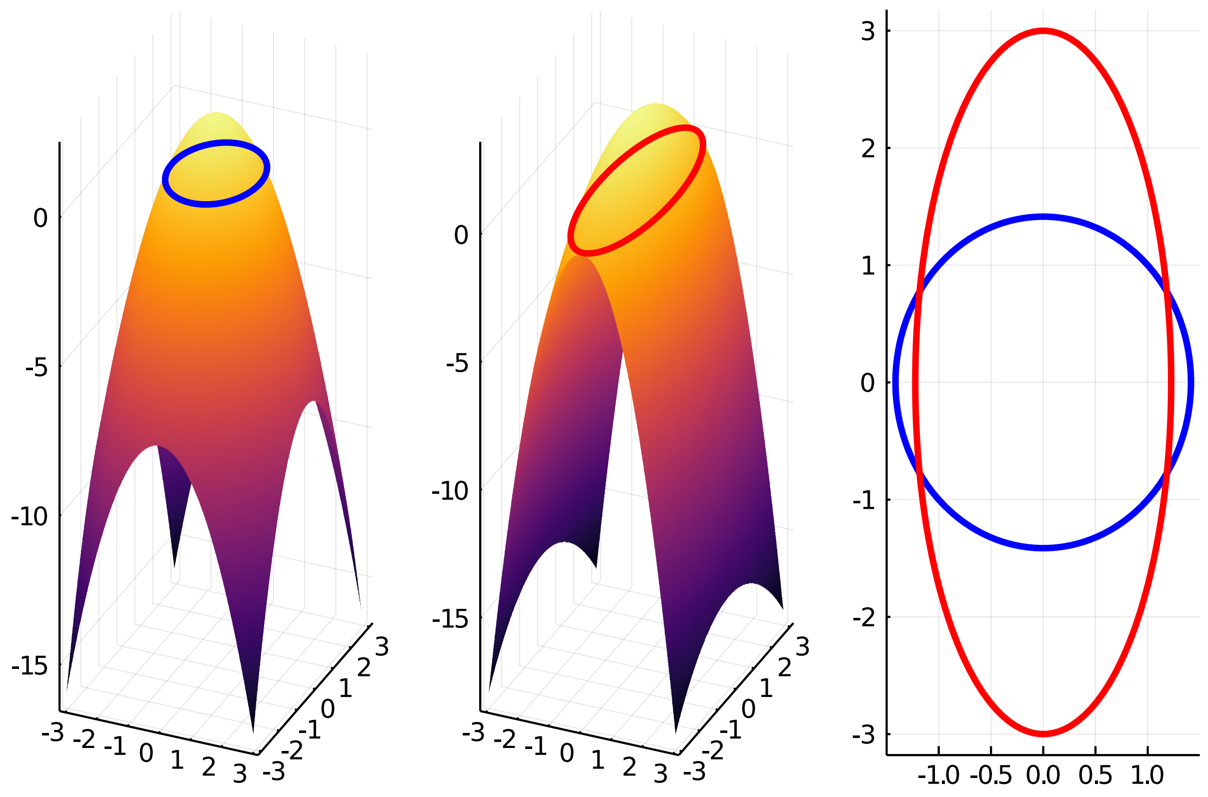
```
@vars a b c d u v
M = [a b; c d]
B = [u, v]
M \ B .|> simplify
```

$$\left[\begin{array}{c}\frac{-bv+du}{ad-bc}\\\frac{av-cu}{ad-bc}\end{array}\right]$$

The term $\det(M) = ad - bc$ term is important, as evidenced by its appearance in the denominator of each term. When this is zero there is not a unique solution, as in the typical case.

Using Newton's method to solve for intersection points, uses linearization of the surfaces to replace the problem to the intersection of level curves for tangent planes. This is the linear case that can be readily solved. As with Newton's method for the univariate case, the new answer is generally a better *approximation* to the answer, and the process is iterated to get a *good enough* approximation, as defined through some tolerance.

Consider the functions $f(x, y) = 2 - x^2 - y^2$ and $g(x, y) = 3 - 2x^2 - (1/3)y^2$. These graphs show their surfaces with the level sets for $c = 0$ drawn and just the levels sets, showing they intersect in 4 places.



We look to find the intersection point near $(1, 1)$ using Newton's method

We have by linearization:

$$f(x, y) \approx f(x_n, y_n) + \frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial y}\Delta y \tag{3}$$

$$g(x, y) \approx g(x_n, y_n) + \frac{\partial g}{\partial x}\Delta x + \frac{\partial g}{\partial y}\Delta y, \tag{4}$$

where $\Delta x = x - x_n$ and $\Delta y = y - y_n$. Setting $f(x, y) = 0$ and $g(x, y) = 0$, leaves these two linear equations in Δx and Δy:

$$\frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial y}\Delta y = -f(x_n, y_n) \tag{5}$$

$$\frac{\partial g}{\partial x}\Delta x + \frac{\partial g}{\partial y}\Delta y = -g(x_n, y_n). \tag{6}$$

One step of Newton's method defines (x_{n+1}, y_{n+1}) to be the values (x, y) that make the linearized functions about (x_n, y_n) both equal to $\vec{0}$.

As just described, we can use `Julia`'s \ operation to solve the above system of equations, if we express them in matrix form. With this, one step of Newton's method can be coded as follows:

```julia
function newton_step(f, g, xn)
    M = [ForwardDiff.gradient(f, xn)'; ForwardDiff.gradient(g, xn)']
    b = -[f(xn), g(xn)]
    Delta = M \ b
    xn + Delta
end
```

```
newton_step (generic function with 1 method)
```

We investigate what happens starting at $(1, 1)$ after one step:

```
f(x,y) = 2 - x^2 - y^2
g(x,y) = 3 - 2x^2 - (1/3)y^2
f(v) = f(v...); g(v) = g(v...)
x0 = [1,1]
x1 = newton_step(f, g, x0)
```

```
2-element Array{Float64,1}:
 1.2
 0.8
```

The new function values are

```
f(x1), g(x1)
```

```
(-0.08000000000000007, -0.09333333333333327)
```

We can get better approximations by iterating. Here we hard code 4 more steps:

```
x2 = newton_step(f, g, x1)
x3 = newton_step(f, g, x2)
x4 = newton_step(f, g, x3)
x5 = newton_step(f, g, x4)
x5, f(x5), g(x5)
```

```
([1.1832159566199232, 0.7745966692414834], 0.0, 1.6653345369377348e-16)
```

We see that at the new point, `x5`, both functions are basically the same value, 0, so we have approximated the intersection point.

For nearby initial guesses and reasonable functions, Newton's method is *quadratic*, so should take few steps for convergence, as above.

Here is a simplistic method to iterate n steps:

```
function nm(f, g, x, n=5)
    for i in 1:n
      x = newton_step(f, g, x)
    end
    x
end
```

```
nm (generic function with 2 methods)
```

Example Consider the bicylinder the intersection of two perpendicular cylinders of the same radius. If the radius is 1, we might express these by the functions:

$$f(x,y) = \sqrt{1-y^2}, \quad g(x,y) = \sqrt{1-x^2}.$$

We see that $(1,1)$, $(-1,1)$, $(1,-1)$ and $(-1,-1)$ are solutions to $f(x,y)=0$, $g(x,y)=0$ *and* $(0,0)$ is a solution to $f(x,y)=1$ and $g(x,y)=1$. What about a level like $1/2$, say?

Rather than work with $f(x,y)=c$ we solve $f(x,y)^2=c^2$, as that will be avoid issues with the square root not being defined. Here is one way to solve:

```
c = 1/2
f(x,y) = 1 - y^2 - c^2
g(x,y) = (1 - x^2) - c^2
f(v) = f(v...); g(v) = g(v...)
nm(f, g, [1/2, 1/3])
```

```
2-element Array{Float64,1}:
 0.8660254037844386
 0.8660254037935468
```

That $x=y$ is not so surprising, and in fact, this problem can more easily be solved analytically through $x^2=y^2=1-c^2$.

1.3 Implicit differentiation

Implicit differentiation of an equation of two variables (say x and y) is performed by *assuming* y is a function of x and when differentiating an expression with y, use the chain rule. For example, the slope of the tangent line, dy/dx, for the general ellipse $x^2/a+y^2/b=1$ can be found through this calculation:

$$\frac{d}{dx}(\frac{x^2}{a}+\frac{y^2}{b}) = \frac{d}{dx}(1),$$

or, using $d/dx(y^2)=2ydy/dx$:

$$\frac{2x}{a}+\frac{2y\frac{dy}{dx}}{b}=0.$$

From this, solving for dy/dx is routine, as the equation is linear in that unknown: $dy/dx = -(b/a)(x/y)$

With more variables, the same technique may be used. Say we have variables x, y, and z in a relation like $F(x,y,z)=0$. If we assume $z=z(x,y)$ for some differentiable function (we mention later what conditions will ensure this assumption is valid for some open set), then we can proceed as before, using the chain rule as necessary.

For example, consider the ellipsoid: $x^2/a + y^2/b + z^2/c = 1$. What is $\partial z/\partial x$ and $\partial z/\partial y$, as needed to describe the tangent plane as above?

To find $\partial/\partial x$ we have:

$$\frac{\partial}{\partial x}(x^2/a + y^2/b + z^2/c) = \frac{\partial}{\partial x}1,$$

or

$$\frac{2x}{a} + \frac{0}{b} + \frac{2z\frac{\partial z}{\partial x}}{c} = 0.$$

Again the desired unknown is within a linear equation so can readily be solved:

$$\frac{\partial z}{\partial x} = -\frac{c}{a}\frac{x}{z}.$$

A similar approach can be used for $\partial z/\partial y$.

Example Let $f(x, y, z) = x^4 - x^3 + y^2 + z^2 = 0$ be a surface with point $(2, 2, 2)$. Find $\partial z/\partial x$ and $\partial z/\partial y$.

To find $\partial z/\partial x$ we have:

```
Z = SymFunction("Z")
@vars x y
solve(diff(x^4 -x^3 + y^2 + Z(x,y)^2, x), diff(Z(x,y),x))
```

$$\left[\ \frac{x^2(3-4x)}{2Z(x,y)}\ \right]$$

Similarly, for $\partial z/\partial y$:

```
Z = SymFunction("Z")
@vars x y
solve(diff(x^4 -x^3 + y^2 + Z(x,y)^2, y), diff(Z(x,y),y))
```

$$\left[\ -\frac{y}{Z(x,y)}\ \right]$$

1.4 Optimization

For a continuous univariate function $f : R \to R$ over an interval I the question of finding a maximum or minimum value is aided by two theorems:

- The Extreme Value Theorem, which states that if I is closed (e.g, $I = [a, b]$) then f has a maximum (minimum) value M and there is at least one value c with $a \leq c \leq b$ with $M = f(x)$.