

Math 131/135/194, Fall 2004

Applied Calculus

Profs. Kaplan & Flath

Macalester College

Lab 4: Optimization

Goal of this lab

To understand how the derivative and its multivariate analog, the gradient, can be used to find arguments to functions that minimize the output of the function.

Mathematical Vocabulary

We will consider functions of one or more variables, $f(a, b, c, \dots)$. When we *minimize* a function, we seek to find specific numerical values of the function's arguments, a_*, b_*, c_*, \dots that make the value $f(a_*, b_*, c_*, \dots)$ as small as possible. The process of finding these values is called *minimization*. A more general concept, *optimization*, includes the possibility that what we want to find is the *maximum* value of the function. The term *extrema* (or *extreme values*) refers to either maxima or minima.

The function that we seek to minimize or maximize is called the *objective function*, referring to the fact that we seek to reach our objective by adjustment of the inputs to the function.

The smallest possible value of the output of the function is called the *minimum* of the function. It is important to distinguish between the function's minimum and the value of the arguments to the function that produce that minimum. The argument values a_*, b_*, c_*, \dots are called the *argmin* of the function, whereas the minimum is the value of the function applied to these arguments: $f(a_*, b_*, c_*, \dots)$. The argmin tells how to set the inputs of the function in order to produce the smallest possible output. The corresponding term for maximization is *argmax*.

In practical terms, maximization and minimization are completely different. For example, in economic theory, the objective function of a firm is profit. By adjusting the inputs to the firm — the mixture of materials used, the balance of capital and labor, the organization of production, the setting of prices, etc. — firms seek to maximize their profits; minimization of profits makes no sense. In evolutionary theory, the objective function is fitness for the environment. Natural selection can be seen as a kind of maximizing process; it tends to move phenotypes toward a position of maximal fitness for the environment. Species which drift toward minimal fitness would become extinct. In physics and chemistry,

systems move toward a configuration of minimal potential energy, e.g., a rock rolls downhill. Were things the other way, maximizing potential energy, rocks would shoot off to outer space on their own.

Mathematically, however, maximization and minimization are practically the same thing. The argmax of a function $f(\cdot)$ is the same as the argmin of $-f(\cdot)$. In order to simplify discussions, we'll always speak of minimization, keeping in mind that we can use the same techniques for maximization simply by changing the sign of the objective function.

Software

The software for this lab is contained in the file `optim.r` in the `Lab4` directory. You will need to “source” this file to make the software available.

We will also use some built-in software: `contour` and `nlm`.

Minimization

Consider a mass hanging from a spring, as shown in Figure 1. The overall energy of the system can be divided into two parts:

Kinetic Energy due to the motion of the system

Potential Energy due to the configuration or position of the system.

In the presence of friction, systems tend to move to a minimum of the potential energy function.

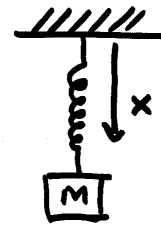


Figure 1. The geometry of the hanging spring. x is measured from the spring support. The no-force length of the spring is 1.

The potential energy of the spring system is a combination of the energy due to gravitation and the spring itself. It is a

function $E(x)$ of the position x of the mass. Mathematically, the function is

$$E(x) = -gmx + \frac{1}{2}k \cdot (x - L)^2 \quad (1)$$

where $g = 9.8m/s^2$ is the acceleration due to gravity, m is the mass, k is the constant describing the stiffness of the spring, and L is the resting length of the spring. The position x is measured from the spring's support, as shown in the figure. If x is measured in meters, then the units of $E(x)$ are Joules. This function has been implemented in R as the function `spring` setting $k = 2$ N/m,¹ $m = 1$ kg, and $L = 1$ m for simplicity; `spring` takes x as an argument and returns the potential energy of the system for that value of x .

```
> spring(0)
[1] 1
> spring(2)
[1] -18.6
```

1 Find the value of x that gives the smallest potential energy of the spring.

A simple way to do this is to make a plot of $E(x)$ versus x , and look for the argmin.

Another way to do it is to choose a candidate value x_0 which, generally, will not be at the minimum. Suppose we choose $x_0 = 0$. Calculate the derivative of the function at x_0 . Then, find a new candidate value which is a little bit different from x_0 : to the left if the derivative at x_0 is positive, or to the right if the derivative is negative. (What happens if the derivative at x_0 is exactly 0?) Call this new value x_1 . Then repeat the process, calculating the derivative at x_1 , and making a new x_2 that is a little bit different from x_1 depending on the sign of the derivative. This “little bit” is called the *step size*.

That is, let

$$x_{i+1} = x_i - s \operatorname{sign}\left(\frac{dE}{dx}\right) \quad (2)$$

where s is the step size.

This process is called *gradient descent* — it is the moral equivalent of walking downhill to find the minimum.

2 Calculate a short sequence of values x_0, x_1, x_2, x_3 following the process described above. Show numerically that the value of $E(x_{i+1})$ is smaller than $E(x_i)$.

One difficulty in gradient descent is figuring out an appropriately small step size s . If the step size is too small, the walk downhill requires many steps. If the step size is too big, we might step right by the minimum. Notice that the value of the derivative itself isn't much use. Whereas x is measured

in meters, $\frac{dE}{dx}$ has units of Joules per meter. So, it doesn't make much sense just to say something like $s = \frac{dE}{dx}$.

Remember that you can find the numerical (finite-difference) derivative of a function using the `D` operator. For instance, to evaluate the derivative of $E(x)$ at $x = 0$ you can use the command

```
> D(spring)(0)
[1] -11.8
```

Note the strange looking use of parentheses. This is because `D(spring)` is the derivative function and `D(spring)(0)` evaluates that function at 0.

The second derivative at $x = 0$ can be found in a similar way:

```
> D(D(spring))(0)
[1] 2.000011
```

3 Keeping in mind that x is measured in meters and $E(x)$ in Joules, what are the units of $\frac{d^2}{dx^2}E$? Now consider the number

$$\frac{\frac{d}{dx}E(x_i)}{\frac{d^2}{dx^2}E(x_i)} \quad (3)$$

where both of the derivatives are to be evaluated at the point x_i . What are the units of this ratio of the first to the second derivative?

A powerful way to find minima involves setting the step size s in gradient descent to be equal to the quantity in Equation 3, that is

$$x_{i+1} = x_i - \frac{\frac{d}{dx}E(x_i)}{\frac{d^2}{dx^2}E(x_i)} \quad (4)$$

Starting with $x_0 = 0$, iterate Equation 4 for several steps. Write down the sequence of x_i you find. Is this method faster or slower than the method of Equation 2?

The smarts about minimization have been packaged into a nice software program that calculates the derivatives etc. for you. This is called `nlm` — short for nonlinear minimization. To use it, give two arguments: (1) the function to be minimized and (2) an initial guess for the argmin. The guess does not have to be very precise. Try

```
> nlm(spring, 0)
```

The `nlm` program returns a variety of information such as the value of the minimum and of the argmin. This information is packaged together into a list, which has named components. The component named “minimum” contains the smallest value of the function that was found by the program. The component named “estimate” gives the argmin.

¹Since forces are measured in Newtons (N), the spring constant in the equation $F = k(x - L)$ has units of Newtons per meter (N/m).

In the R-language, the components of a list are referred to by giving a `$` followed by the component name. So, the component containing the minimum is denoted `$minimum` and the component containing the argmin is denoted `$estimate`.

Functions of Multiple Variables

For a function of a single variable, it is comparatively easy to find the minimum graphically; the iteration methods described above essentially automate the process in an efficient way.

For functions of multiple variables, graphs will not be so effective. But, when there are only two variables, we can still make a graph.

The function `springs` (Note the 's' at the end.) gives the potential energy for a chain of springs, with each spring constant, mass, and length set as in the single-spring case. As in Figure 1, the coordinates are the length of each link in the chain. So, for example, a chain with link lengths of 1 and 2 has potential energy

```
> springs(c(1,2))
[1] -38.2
```

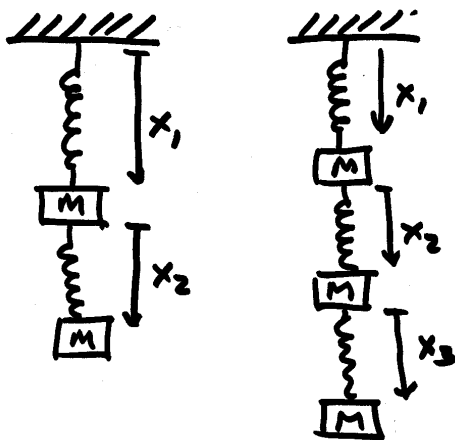


Figure 2. The geometry of a chain of $N = 2$ springs and a chain of $N = 3$ springs. x_i is distance between masses i and $i - 1$. (We consider the support to be $i = 0$.) The no-force length of each spring is 1. The potential energy of this system is given by

$$E(x_1, x_2, \dots, x_N) = \sum_{i=1}^N \left(\frac{1}{2} k (x_i - 1)^2 - gm \sum_{j=1}^i x_j \right)$$

Although this expression looks somewhat daunting, it is really just the common-sense extension of Equation 1 written in compact mathematical notation.

We can make a contour plot of the potential energy of the two-spring system. To do this, we need to specify a set of lengths for the first spring and for the second spring, then evaluate the energy for each combination of these sets of lengths. Follow this example carefully. Note that you do not want too

many points produced by the `seq` command, since the total number of points will be the product of the number of points in the two input variables:

```
> x1 = seq(0,20,length=50)
> x2 = seq(0,20,length=50)
> z = evalAtGrid(x1, x2,springs)
> contour(x1, x2, z)
```

- 4 Make the contour plot and find the argmin, that is, the values of x_1 and x_2 at which the potential energy is minimized. Note that the optimal value of x_2 is smaller than the optimal value of x_1 . Does this make sense in terms of the physical situation depicted in Figure 1?

We can also apply the ideas we discussed in class of gradient descent and Newton's method to functions of multiple variables. For example, here is a function that is the gradient of the multiple-springs system:

```
> g = grad(springs)
```

We can evaluate the gradient at any point we choose. For example

```
> g(c(8,8))
      [,1]
[1,] -5.6
[2,]  4.2
```

Note that this will only work if you previously define the value of `g`.

To help you visualize the gradient function, your diligent instructors provide you with the `plotarrow` function. Once you have a contour plot in the graphics window and have defined the gradient, then use `plotarrow` like this

```
> plotarrow(g, c(8,8))
```

Note that the first argument is the *gradient of the objective function*, NOT the objective function itself. The second argument is the point at which you want the gradient evaluated. `plotarrow` draws a nice arrow in the direction of the gradient. Walking in the *opposite* direction will bring you to the objective function's argmin.

- 5 Plot out the gradient vector in several spots on your contour plot and include this in your write-up. What is the gradient vector at the argmin of the function?

Automated methods for finding minima of functions of several variables use various combinations and modifications of gradient descent and Newton's method. The `nlm` function takes care of all this for us:

```
> nlm(springs, c(0,0))
$minimum
[1] -149.45
$estimate
[1] 10.8  5.9
$gradient
[1] 1.098973e-05 5.554277e-06
```

In fact, we can use `nlm` for functions of more than two variables, e.g., a chain with 10 links:

```
> nlm(springs, c(0,0,0,0,0,0,0,0,0,0))
$minimum
[1] -9782.85
$estimate
 [1] 50.0 45.1 40.2 35.3 30.4 25.5 20.6 15.7 10.8  5.9
$gradient
 [1] 5.045e-05 4.537e-05 4.031e-05 3.529e-05 3.021e-05
 [6] 2.525e-05 2.022e-05 1.517e-05 1.010e-05 4.932e-06
```

6 Explain in physical terms why the argmin of the

`springs` function looks the way it does, with links of shorter and shorter length as we move from the top of the chain to the bottom.

Why does the gradient have 10 components? Why are the values small? What should they be, ideally, at the argmin?

7 Explain why the expressions `nlm(sin,0)` and `nlm(sin,2)` give different results.