ROSE-HULMAN INSTITUTE OF TECHNOLOGY
Department of Mechanical Engineering

Introduction to Optimization

The numerical methods of optimization are very similar to those for root finding. The main difference is we are trying to find either a maximum or minimum of the function rather than $f(x) = 0$. In some cases $f(x)$ might not cross the x axis.

Let's reconsider the benchmark function.

Now let's investigate ways to find the local minima (of course pretending ignorance of the closed-form solution available by setting)—

$x = -0.8165, 0.8165.$

Given the shape of $f(x)$, we know we are looking for $x = 0.8165$ & $f(x) = -6.089$.

Let's start at $x^0 = 3$. Functions of a scalar (things that can be plotted in 2D) are really quite easy since the search direction amounts to the slope. What is not readily apparent is what should be used as a step size. For instance, the slope at $x = 3$ is 25. Since I know the answer, I'll cheat a bit and set my step size to 0.1.

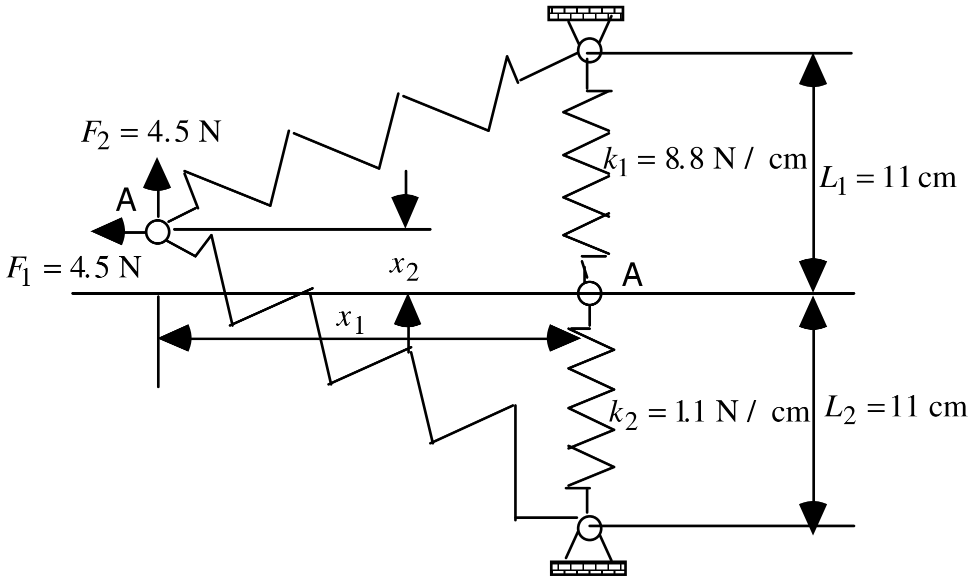
Convergence to a minimum is determined by the slope falling below a tolerance or the change in cost (in this case $f(x)$) falling below a tolerance. We test that we have found a minimum by checking that $\frac{\partial^2 f}{\partial x^2}$ is positive. The first few steps of optimization proceed as follows:

Because of the large step size I chose (0.1) the search oscillates around the optimal point a bit. Choosing step size is an important step that I will ignore until after developing the first order method in many dimensions.

Method of Steepest Descent in *n* dimensions

The really interesting problems in optimization involve multiple parameter dimensions. We'll use Example 13.2 from Magrab et al to motivate our development of multi-dimensional techniques.

Consider the two-spring system shown below. The springs are initially in a vertical, unstretched position.



After the loads are applied at joint A, the systems is deformed until it is in equilibrium. This is shown by the loaded and deformed system in the figure. We are interested in finding the equilibrium state of the loaded system--that is, the location (x_1, x_2) of joint A in the deformed system.

Equilibrium is found by deriving the PE for the system & minimizing it with respect to x_1, x_2.

$$\min_{x_1,x_2} PE(x_1,x_2) = \frac{1}{2}k_1\left(\sqrt{x_1^2 + (L_1 - x_2)^2} - L_1\right)^2 + \frac{1}{2}k_2\left(\sqrt{x_1^2 + (L_2 + x_2)^2} - L_2\right)^2 - F_1x_1 - F_2x_2$$

Functions like this (scalar function of two variables are excellent to illustrate optimization since they are easily visualized. Minima appear as valleys and maxima as peaks.

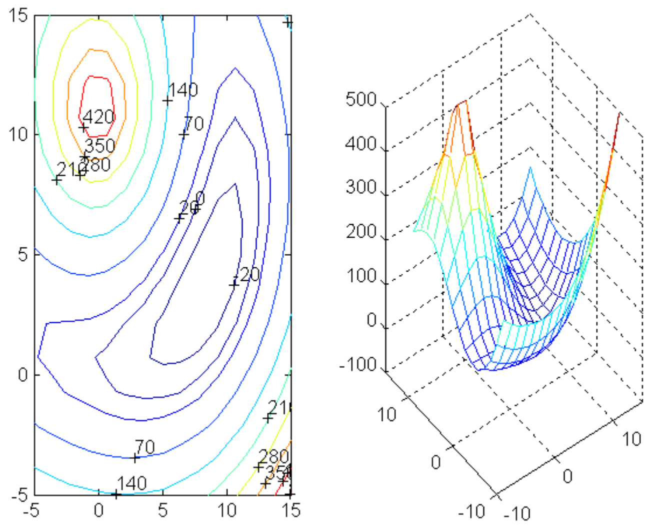


Figure 1: Potential Energy of System as a Cost Function

The following script will plot the potential energy as a function of the deflections.

```
k1 = 8.8; k2 = 1.1; L1 = 11; L2 = 11; F1 = 4.5; F2 = 4.5;
[x1,x2] = meshgrid(linspace(-5,15,15),linspace(-5,15,15));
PE1 = 1/2*k1*(sqrt(x1.^2+(L1-x2).^2)-L1).^2;

PE2 = 1/2*k2*(sqrt(x1.^2+(L2+x2).^2)-L2).^2;
PE = PE1 + PE2 - F1*x1 - F2*x2;

subplot(1,2,1)
h = contour(x1,x2,PE,[-40:20:20 70:70:490]);
clabel(h);

axis([-5 15 -5 15])
subplot(1,2,2)
mesh(x1,x2,PE);
axis([-10 15 -10 15 -100 500])
```

Now suppose we start the search at a random point on the contour. In every case, one wise choice of direction to search would be the 'steepest descent'. This approach is easily comprehended by skiers and mountain climbers. If I'm going to 'ski' this mountain, I'd like to start close to the top. (If I try to start at the top, the slope will be zero, and I won't go anywhere, unless I skate a bit first--I guess this is more like snowboarding, but I hate to admit it.) For the sake of argument, let's pick $x_1 = 0$, $x_2 = 10$. Imagine releasing a ball at zero velocity from this position on the 'mountain'. The direction the ball rolls initially is the direction of steepest descent, or 'fall line'. We want to search in this direction. Finding this direction in a space of *n* dimensions is done by finding the gradient grad*F* or

Instead of just 'left' or right, we now have *n* coefficients to describe the direction.

Occasionally, the derivatives may be difficult or impossible to find analytically. In this case, a finite difference method may be used. That is, we can approximate each partial derivative as:

Choosing the size of δx_i can be somewhat of an art, but I suggest δx_i=0.0001 for most problems.

The previous method may then be extended to a function of many variables

OR

Let's try this on the example. First find the analytic derivatives of PE:

$$\frac{\partial PE(x_1,x_2)}{\partial x_1} = \frac{k_1 x_1\left(\sqrt{x_1^2+(L_1-x_2)^2}-L_1\right)}{\sqrt{x_1^2+(L_1-x_2)^2}} + \frac{k_2 x_1\left(\sqrt{x_1^2+(L_2+x_2)^2}-L_2\right)}{\sqrt{x_1^2+(L_2+x_2)^2}} - F_1$$

$$\frac{\partial PE(x_1,x_2)}{\partial x_2} = \frac{-k_1(L_1-x_2)\left(\sqrt{x_1^2+(L_1-x_2)^2}-L_1\right)}{\sqrt{x_1^2+(L_1-x_2)^2}} + \frac{k_2(L_2+x_2)\left(\sqrt{x_1^2+(L_2+x_2)^2}-L_2\right)}{\sqrt{x_1^2+(L_2+x_2)^2}} - F_2$$

Now program the Newton method. Some notes on programming--The wise programmer will look for common factors in the expressions above and compute these first, avoiding repeating the same computation and thus speeding things up. Also, it might be good to write a *function* that evaluates the PE and derivatives of the PE given the deflections.

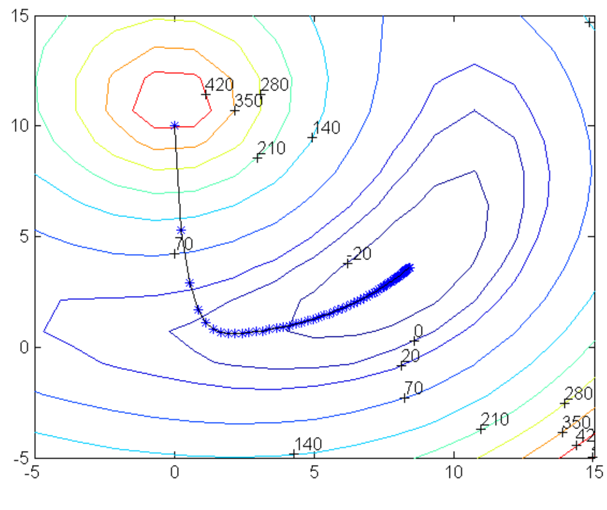


Figure 2: First one hundred iterations of steepest descent.

Cubic Line Search

As we saw in the 1D case, step size is critical to rapid convergence. That is, we need to specify how far the 'skier' will descend along a certain direction before she again checks the slope.

In general, evaluating the cost (PE in this example) is much easier than evaluating the slope. The current example illustrates this well.

One approach to determining the best step size is the cubic line search. In this method, we determine the search direction, and then build a one-dimensional profile of the cost along this direction. Graphically, we can think of this as 'slicing' through the mountain resulting, for our current example in the following 1D profile:

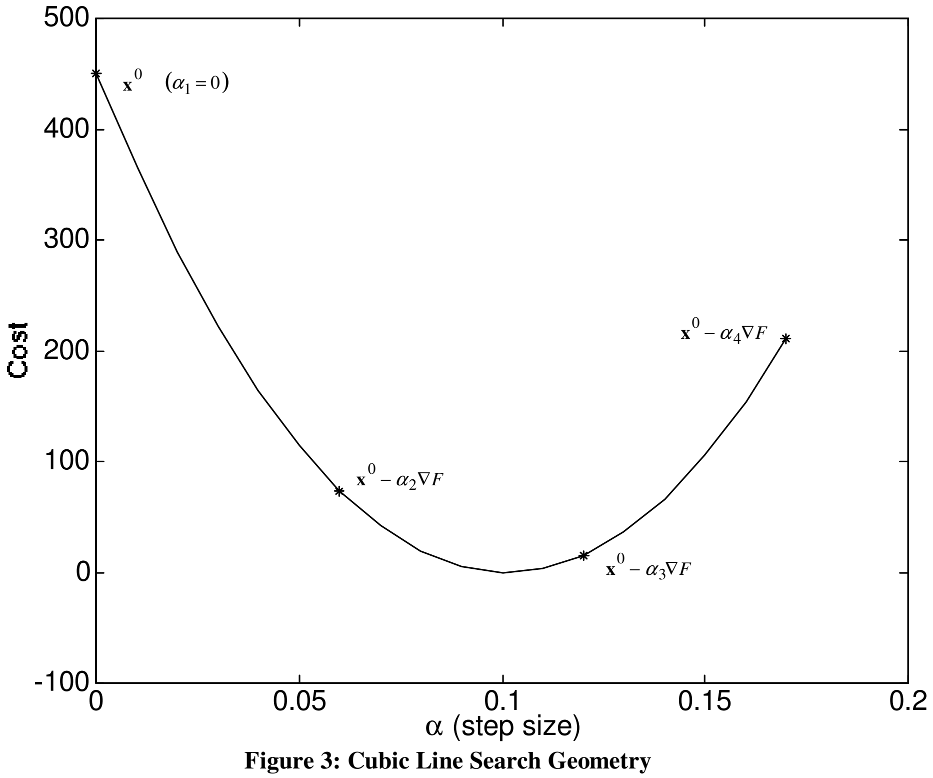


Figure 3: Cubic Line Search Geometry

Now we search along the prescribed direction for at least four points. In the best case, these points will capture a local minimum.

Deciding what step size to use in finding these points is very much an art. You might have noticed that the largest step size tested above is 0.17. I did this on purpose to make the plot look pretty, but in general, we will not have this much insight into the problem. If the cost remains bounded, then we can use

.

However, even if the cost remains bounded over this interval, the quality of information may be poor if no minimum is straddled, or if the cost gets very high in the search direction. Occasionally, additional cost evaluations and logic may be required to improve the quality of information.

Once we have four points (including the current guess) we form a *cubic polynomial* model of the cost along the current search direction, that is

(1)

With four points, and four unknown constants, the curve fit is done by solution of a Vandermonde system. (In practice, use the **polyfit** function, just thought I'd show you the inner workings here)

Solving the linear system above gives us the c_is in the cubic model of Eqn 1.

Once we have the cubic model, we can find the minimum along that direction by taking the derivative and setting it (the slope) equal to zero. Given the form of the cubic in Eqn 1, the derivative will always be the following quadratic polynomial:

(2)

So the roots can be found by the quadratic formula:

The root corresponding to the minimum is determined by testing the second derivative:

```
if
   minimum is at $\alpha_1^*$
else
   minimum is at $\alpha_2^*$
```

Now set the step size according to the logic above.

Thus, using cubic line search, the steepest descent method is modified as follows:

Determine the search direction

Build the cubic model along
Solve (2) for possible minima
Test

and select a step size
Update **x** by

Repeat until convergence

Here we try it on the example from Magrab:

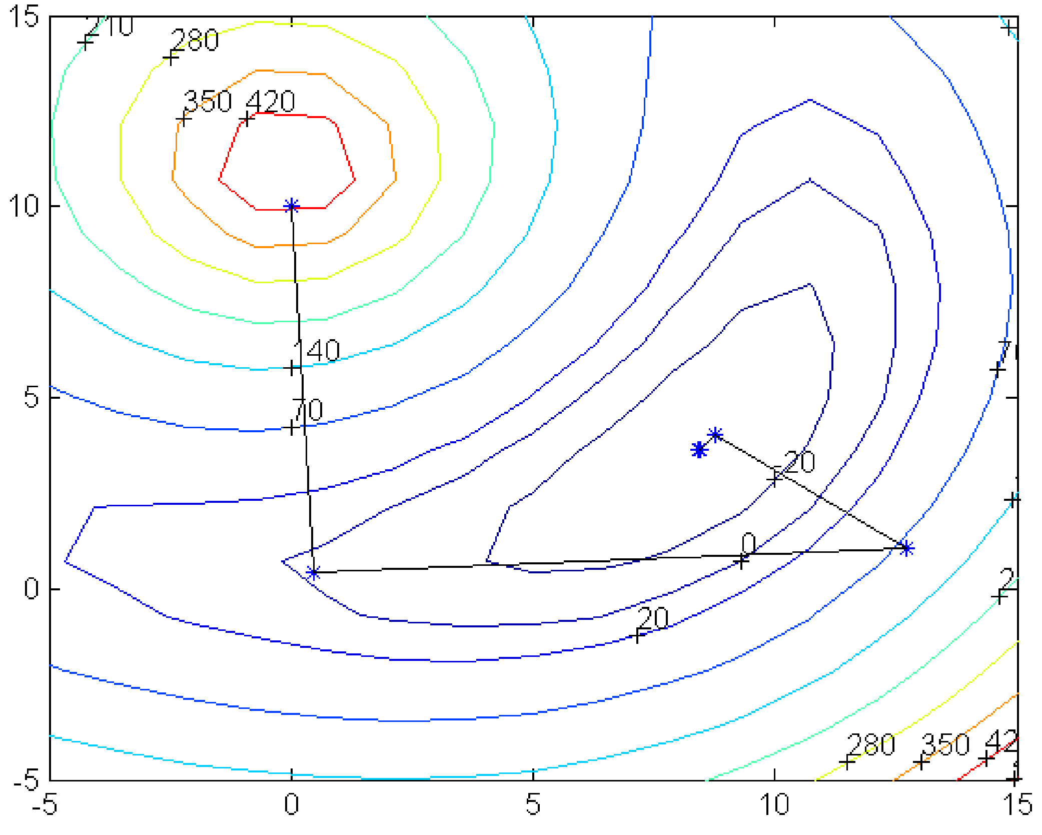


Figure 4: First ten iterations of Steepest Descent Method with Cubic Line Search

Second order Newton Method in Multiple Dimensions

Just as we sought a parabolic model of the non-linear equations in order to quickly iterate to a solution, we can create a quadratic (paraboloid) model of any non-linear cost function in many dimensions. The model will be based on multi-dimensional Taylor Series, and will have the form:

(3)

Where **c** is the gradient vector:

H is the Hessian Matrix

x is the current guess and **x*** is the optimum solution. When **x** is optimized the first derivative of Eqn 3 will be zero:

Solving for **x***:

(4)

Equation 4 gives the update sequence for a Second Order Newton Method in many dimensions. Essentially, the **H** matrix has the effect of steering the search direction toward the minima of the paraboloid model, rather than in the steepest descent direction. Once the search direction is determined, we would determine the step size from a cubic line search just as we did in the first order case.

In this case, the test for a minima is that the gradient (slope) is very small and the Hessian matrix **H** is *positive definite*. The second order Newton method can only be applied in a region where the **H** matrix is positive definite. This places serious limitations on the use of this algorithm for the current problem. Positive definite **H** means the contour looks like a bowl sitting 'right side up', meaning the contents cannot escape by gravity alone. The region that clearly looks like a peak has a non positive definite **H.**

This method is rarely advantageous as the second derivatives in the **H** matrix are usually expensive to compute. The attached Maple spreadsheet gives the needed second derivatives for our potential energy example as a case in point. Also, I must initialize in a region where **H** is positive definite, so I won't be making a valid comparison with the previous two methods. I will conclude this tome by reporting whether or not I am successful at programming those equations in the next hour.

Victory!! I was able to finish the program in about 30 minutes. The top-level function is very similar to the one for steepest descent with cubic line search. Only a few lines are modified to provide for computing the Hessian matrix and determining the search direction to be $\mathbf{H}^{-1}\mathbf{c}$.
The final plot shows that we approach the optimum point almost directly, and arrive in just a few iterations.

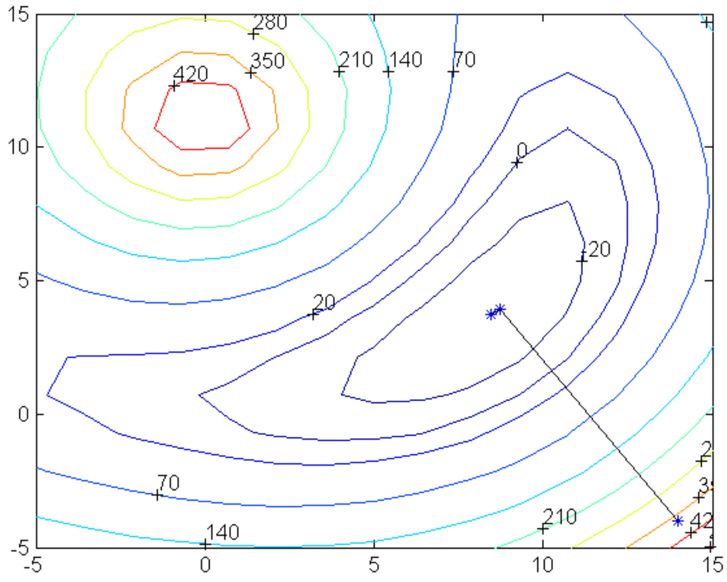


Figure 5: Result of Second Order Newton Search

References

Magrab, E. B., et al, *An Engineer's Guide to Matlab*, Prentice-Hall, Upper Saddle River, NJ, 2000.

Strang, G., *Introduction to Applied Mathematics,* Wellesley-Cambridge Press, Wellesley, MA, 1986.

Matlab Optimization Toolbox Documentation:
http://www.mathworks.com/access/helpdesk/help/pdf_doc/optim/optim_tb.pdf