Optimization of Mechanical Systems
ME 748: Class Notes

Krishnan Suresh

Department of Mechanical Engineering
University of Wisconsin – Madison
Madison, WI 53706

If you find errors in this document (I am presuming there are plenty!), could you please email them to:
suresh@engr.wisc.edu

Version 1.18

Table of Contents

1 Introduction to Optimization

Highlights

- Continuous optimization problems can be posed in the following generic form:

$$\underset{\bar{x}}{Min}\ f(\bar{x})$$
$$h_i(\bar{x}) = 0; i = 1,2,\ldots$$
$$g_i(\bar{x}) \leq 0; i = 1,2,\ldots$$

where $\bar{x}$ are the continuous design/optimization variables, $f(\bar{x})$ is the optimization objective, $h_i(\bar{x})$ are the equality constraints, and $g_i(\bar{x})$ are the inequality constraints. The constraints may be absent in some problems.

Equation Section (Next) Optimization plays a central role in engineering; this manuscript is a brief introduction to this vast field, with a balanced approach towards theory, algorithms, tools and applications.

1.1 Typical Optimization Examples

Example 1: Consider the following example: suppose a beer manufacturing company would like to optimize a beer-can in the following sense. Assuming that the beer-can can be modeled as a perfect cylinder of radius R and height H, the objective is to arrive at an optimal value of R & H such that the surface area of the beer-can is minimized (reduces manufacturing cost) while ensuring that beer-can can hold certain given volume.

Since the area of the beer-can includes top, bottom and perimeter, mathematically, one can state the problem as follows:

$$\begin{aligned}&\underset{\{R,H\}}{Min}\ \left(2\pi R^2 + 2\pi RH\right)\\&\pi R^2 H = V\end{aligned} \tag{1.1}$$

Equation (1.1) represents a *minimization problem with an equality constraint*, a typical optimization problem.

There are numerous valid and relevant questions one can ask about an optimization problem:

- Does an optimization problem exhibit a unique solution?
- How does one determine the optimal solution?
- What are the appropriate algorithms?
- How sensitive is the optimal solution to various parameters of the problem?

We shall address most of these questions in due course. For now, let us address, perhaps, the most important question: how does one determine the optimal solution, in particular for the problem posed in Equation (1.1)?

It is 'easy' to conclude that one can eliminate the constraint in Equation (1.1) by explicitly replacing $H = V / \pi R^2$ in the objective (we shall discuss pitfalls of constraint elimination later on), simplifying the problem to:

$$\underset{\{R\}}{Min} \ \left(2\pi R^2 + \frac{2V}{R}\right) \tag{1.2}$$

Observe that we have reduced the problem from a 2-variable equality-constrained minimization problem to a *1-variable unconstrained minimization problem*. From basic calculus (see next Section), such problems can be solved by setting the derivative of the objective to 0, (again the theoretical traps of this obvious process are discussed in the next Section), i.e.,

$$\begin{aligned}
&\frac{d\left(2\pi R^2 + \dfrac{2V}{R}\right)}{dR} = 0 \Rightarrow 4\pi R - \frac{2V}{R^2} = 0 \\
&\Rightarrow R = \sqrt[3]{\frac{V}{2\pi}} \\
&H = V/(\pi R^2)
\end{aligned} \tag{1.3}$$

Now, let us do some numbers: suppose V = 16oz (i.e., 0.00047 m^3), we have:

$$\begin{aligned}
&R = \sqrt[3]{\frac{V}{2\pi}} = 0.042m = 1.65''\\
&H = V/(\pi R^2) = 0.084m = 3.3''
\end{aligned}$$

Wonderful results except that it is somewhat impractical … try carrying a 3.3 inch diameter cylinder around when you are half-drunk!

As one can now recognize (in hindsight) that proper constraints were not placed on the design variables. A reasonable modification to Equation (1.1) is possibly:

$$\begin{aligned}
&\underset{\{R,H\}}{Min} \ \left(2\pi R^2 + 2\pi RH\right)\\
&\pi R^2 H = V\\
&R \leq R_{\max}
\end{aligned} \tag{1.4}$$

We now have 2-variable minimization problem with both equality and inequality constraint. We shall later study formal methods to solve Equation (1.4), but it is fairly clear that the solution will change significantly as additional constraints are posed on an optimization problem.

Example 2: Consider now a problem where the objective is to construct the largest rectangle within a disc of radius R. We shall assume that in order to construct the largest rectangle, it must be centered with respect to the disc (prove it!). Thus, the two variables are W & H (width and height of rectangle), and the optimization problem is:

$$\begin{aligned}
&\underset{\{W,H\}}{Max} \ (WH)\\
&\left(\frac{W}{2}\right)^2 + \left(\frac{H}{2}\right)^2 = R^2
\end{aligned} \tag{1.5}$$

Observe that the above is a *maximization* problem with one equality constraint. As before, one can eliminate the equality constraint … or arguments of symmetry will lead us to the conclusion that W must be equal to H. We shall consider formal methods of solving optimization problems later on. For now, observe that Equation (1.4) is a minimization problem while Equation (1.5) is a maximization problem.

Example 3: Next, consider a typical structural problem (see Figure 1-1), where the objective is to find the optimal diameters of the individual members such that the weight of the structure is minimized, but the problem is subject to various buckling, stress, and deflection constraints. Such problems can be posed in the following form:

$$\begin{aligned}
&\underset{\{d_1,d_2,\ldots\}}{Min} \quad w \\
&d_i \geq d_{\min}, i = 1,2\ldots,M \ (buckling) \\
&\sigma_i \leq \sigma_{\max}, i = 1,2,\ldots,M \ (stress) \\
&\delta_k \leq \delta_{\max}, k = 1,2,\ldots,N \ (deflection)
\end{aligned} \tag{1.6}$$

A major difference between the structural problem and the previous two examples is that, given a particular choice for the design variables, verifying if the constraints are satisfied for the structural problem is non-trivial. Indeed, given a set of diameters for the individual members, computing the stress and deflection requires an elaborate finite element analysis.

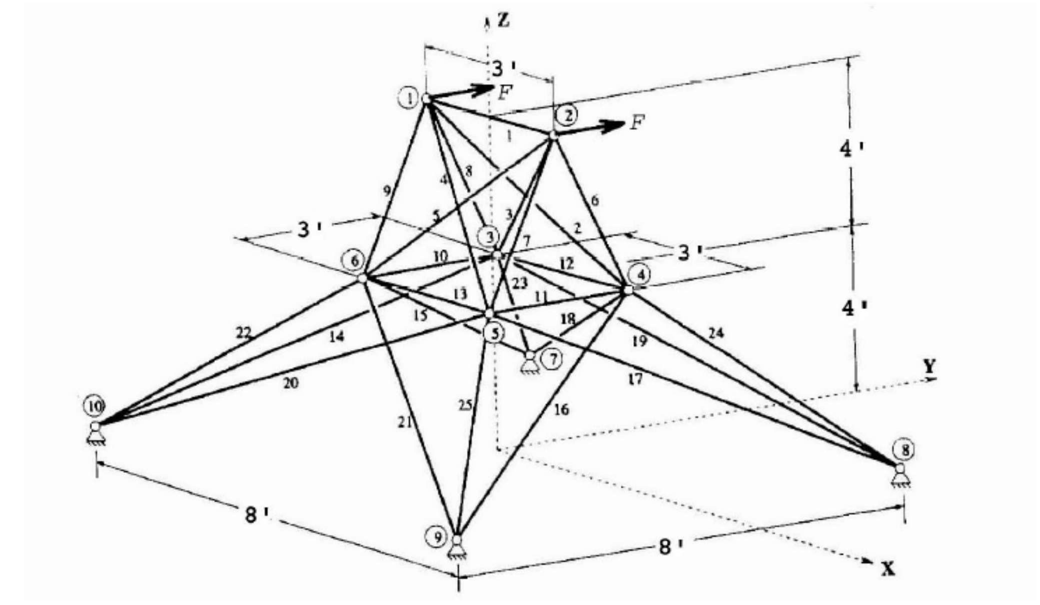


Figure 1-1: A typical load-carrying structural-frame.

1.2 Standard Formulation of Optimization Problems

From an analysis and algorithmic perspective, handling a variety of optimization problems ('minimization' versus 'maximization', '$\leq$ constraints' versus '$\geq$ constraints') quickly gets cumbersome. It is much more efficient if we can convert all such optimization problems into a standard formulation. A typical formulation that most researchers in the optimization community adopt is:

$$\begin{aligned}
&\underset{\overline{x}}{Min} \ f(\overline{x}) \\
&h_i(\overline{x}) = 0; i = 1,2,\ldots \\
&g_i(\overline{x}) \leq 0; i = 1,2,\ldots
\end{aligned} \tag{1.7}$$

Indeed, all the optimization examples discussed in the previous Section can be converted into the above standard form. For example, from Equation (1.4) we have:

$$\begin{aligned}
&\overline{x} = \{R, H\} \\
&f = \left(2\pi R^2 + 2\pi RH\right) \\
&h_1 = \pi R^2 H - V \\
&g_1 = R - R_{\max}
\end{aligned} \tag{1.8}$$

On the other hand, since Equation (1.5) is a maximization problem, it can be converted into a (standard) minimization problem as follows:

$$\begin{aligned}
&\overline{x} = \{W, H\} \\
&f = -(WH) \\
&h_1 = \left(\frac{W}{2}\right)^2 + \left(\frac{H}{2}\right)^2 - R^2
\end{aligned} \tag{1.9}$$

Finally, Equation (1.6) can be posed via:

$$\begin{aligned}
&\overline{x} = \{d_1, d_2, ...\} \\
&f = w \\
&g_i = d_{\min} - d_i ; i = 1, 2..., M \\
&g_{M+j} = \sigma_j - \sigma_{\max} ; j = 1, 2..., M \\
&g_{2M+k} = \delta_k - \delta_{\max} ; k = 1, 2..., K
\end{aligned} \tag{1.10}$$

with a total of 2M+L inequality constraints, and no equality constraints.

Our objective henceforth is to model engineering optimization problem in the standard form given in Equation (1.7), and to develop methods for analyzing, and solving such problems.

2 Single Variable Unconstrained Optimization

Highlights

- For the 1-variable unconstrained minimization problem: $\underset{x}{Min}\ f(x)$, points that satisfy $\frac{df}{dx} = 0$ are called stationary points, and further, at a stationary point if $\frac{d^2 f}{dx^2} > 0$ then the function takes a minimum at that point, and if $\frac{d^2 f}{dx^2} < 0$ then the function takes a maximum at that point.
- In general, one must rely on numerical methods to find (local) minima and stationary points.
- Methods that directly find a 'nearest' local minimum include quadratic fitting and golden section search.
- Methods that find a 'nearest' stationary point include the Newton-Raphson method.
- Important Matlab functions introduced here include `inline, fminbnd, fminunc and fsolve.`

Equation Section (Next)First, we shall focus on a special case of Equation (1.7) where there are no constraints, and there is only one design variable i.e.,

$$\underset{x}{Min}\ f(x) \tag{2.1}$$

In this Section, we shall review the theory necessary to analyze and solve Equation (2.1).

2.1 Optimality Criteria

Towards this end, recall the following theorem from basic calculus [1]:

Theorem 2-1: If x^* is the minimum of a differentiable function $f(x)$ then

$$\left.\frac{df}{dx}\right|_{x=x^*} = 0$$

Proof: We provide here an informal proof; see [2] for details. Consider the Taylor series:

$$f(x^* + \Delta x) = f(x^*) + \left.\frac{df}{dx}\right|_{x=x^*} \Delta x + \frac{1}{2!}\left.\frac{d^2 f}{dx^2}\right|_{x=x^*} (\Delta x)^2 + \frac{1}{3!}\left.\frac{d^3 f}{dx^3}\right|_{x=x^*} (\Delta x)^3 + o(\Delta x)^4$$

In particular, the first two terms:

$$f(x^* + \Delta x) \approx f(x^*) + \left.\frac{df}{dx}\right|_{x=x^*} \Delta x$$

Suppose $f(x^*)$ is a minimum, then by definition, $f(x^*)$ must be less than both $f(x^* + \Delta x)$ and $f(x^* - \Delta x)$ for sufficiently small Δx

$$f(x^*) \le f(x^* + \Delta x) \approx f(x^*) + \left.\frac{df}{dx}\right|_{x=x^*} \Delta x$$

$$\Rightarrow 0 \le \left.\frac{df}{dx}\right|_{x=x^*} \Delta x$$

Therefore,

$$f(x^*) \le f(x^* - \Delta x) \approx f(x^*) - \left.\frac{df}{dx}\right|_{x=x^*} \Delta x$$

$$\Rightarrow 0 \le -\left.\frac{df}{dx}\right|_{x=x^*} \Delta x$$

$$\Rightarrow 0 \ge \left.\frac{df}{dx}\right|_{x=x^*} \Delta x$$

Thus, we must have

$$\left.\frac{df}{dx}\right|_{x=x^*} = 0$$

♦

For example, consider the following optimization problem:

$$\underset{x}{Min} \quad x(x-1) \tag{2.2}$$

Observe that the above theorem only states that:

$$\text{If } x^* \text{ is a minimum} \Rightarrow \left.\frac{df}{dx}\right|_{x=x^*} = 0 \tag{2.3}$$

It does **not** state the inverse. In other words:

$$\left.\frac{df}{dx}\right|_{x=x^*} = 0 \not\Rightarrow x^* \text{ is a minimum} \tag{2.4}$$

Therefore, before one can 'apply' the theorem, one should know the minimum and then verify if it satisfies the above necessary condition! We shall use graphical methods to 'guess' the minimum. In Matlab, one can plot the function in the range $0 \le x \le 1$.

```
x = 0:0.01:1;
f = x.*(x-1);
plot(x,f);
grid on;
axis('equal')
```

Script 2-1: Matlab script for plotting $f(x) = x(x-1), 0 \le x \le 1$.

From the plot below, it appears that $x = 0.5$ is a minimum.

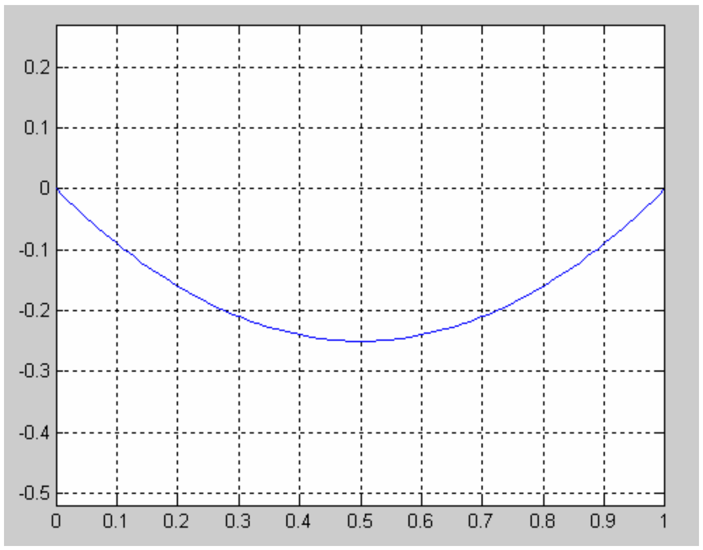


Figure 2-1: Plot of $f(x) = x(x-1)$.

Indeed, from Equation (2.2), we have:

$$\frac{df}{dx} = 2x - 1$$
$$\left.\frac{df}{dx}\right|_{x=0.5} = 0 \tag{2.5}$$

To reiterate the limitation of Theorem 2-1, if the derivative vanishes, one cannot conclude that the function takes a minimum at that point! For example, consider the function $f(x) = 16x - x^2$. Take its derivative and set it to zero, i.e.,

$$\frac{df}{dx} = 16 - 2x = 0 \Rightarrow x = 8 \tag{2.6}$$

But the function takes a maximum at x = 8, as shown below.

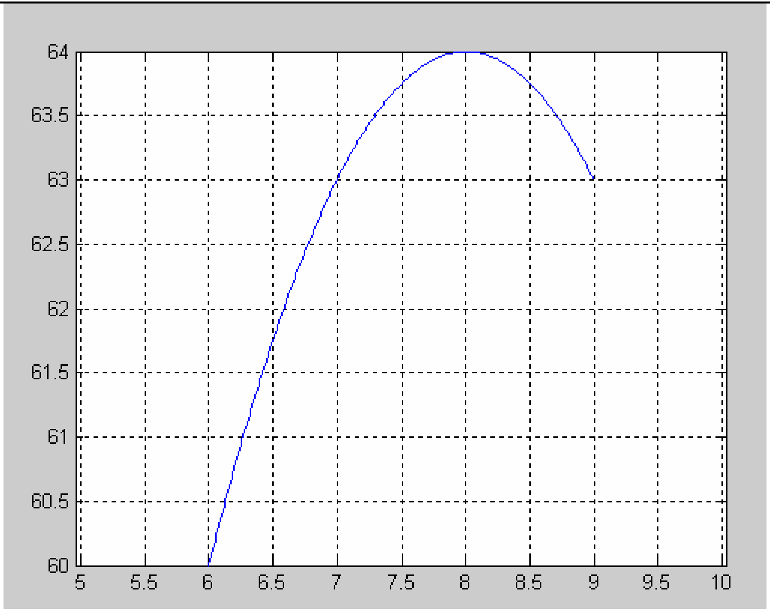


Figure 2-2: Plot of $f(x) = 16x - x^2$.

Similarly, consider $f(x) = 3x^3 - 9$; note that at x = 0, the derivative vanishes

$$\frac{df}{dx} = 9x^2\Big|_{x=0} = 0 \qquad (2.7)$$

But the function neither takes a minimum or a maximum at x = 0, as is plotted below.

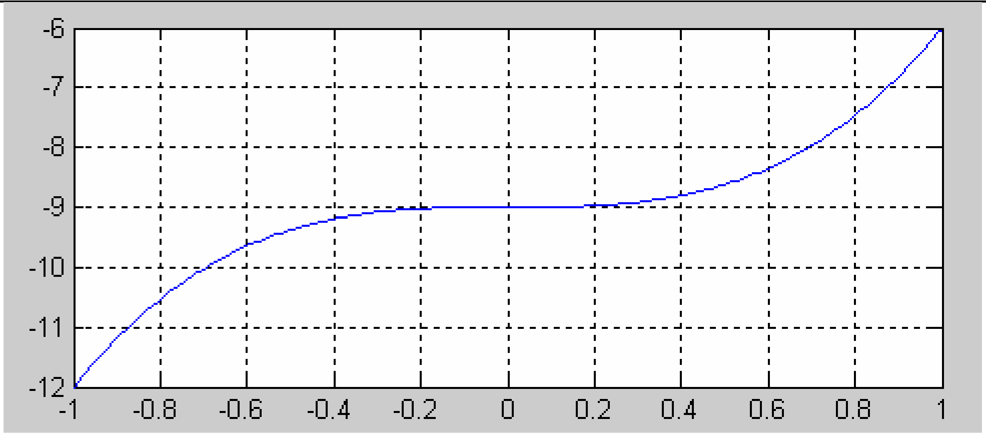


Figure 2-3: Plot of $f(x) = 3x^3 - 9$.

So, let us separate out the issues:

1. Points where the derivative of a function vanishes are called **stationary points**. Thus x = 0.5, in the first example, x = 8 in the second example and x = 0 in the last example are all stationary points.
2. A stationary point can be a minimum (as in the first example) or a maximum (as in the second example) or a cusp (as in the third example).

How do we differentiate between the three cases?

To differentiate between the three cases, we must consider the second derivative. Note that in the first example, the derivative reaches a zero at the minimum and then increases. Mathematically, we have:

$$\frac{d}{dx}\left(\frac{df}{dx}\right) = \frac{d^2f}{dx^2} > 0 \text{ at a minimum.}$$

Indeed, for the given function

$$\frac{d^2f}{dx^2} = 5 > 0$$

While, in the second example, the derivative reaches a zero at the maximum and then decreases, i.e.,

$$\frac{d}{dx}\left(\frac{df}{dx}\right) = \frac{d^2f}{dx^2} < 0 \text{ at a maximum.}$$

Indeed, for the given function

$$\frac{d^2f}{dx^2} = -6 < 0$$

Finally, one may be tempted to conclude that if the second derivative is zero at a stationary point, then it must be a cusp (neither a minimum nor a maximum). Unfortunately, this conclusion is false. For example, consider the function $f(x) = x^4 + 12$. Note that

$$\frac{df}{dx} = 4x^3 = 0 \Rightarrow x = 0$$

But the function indeed takes a minimum at x = 0, as is plotted below.

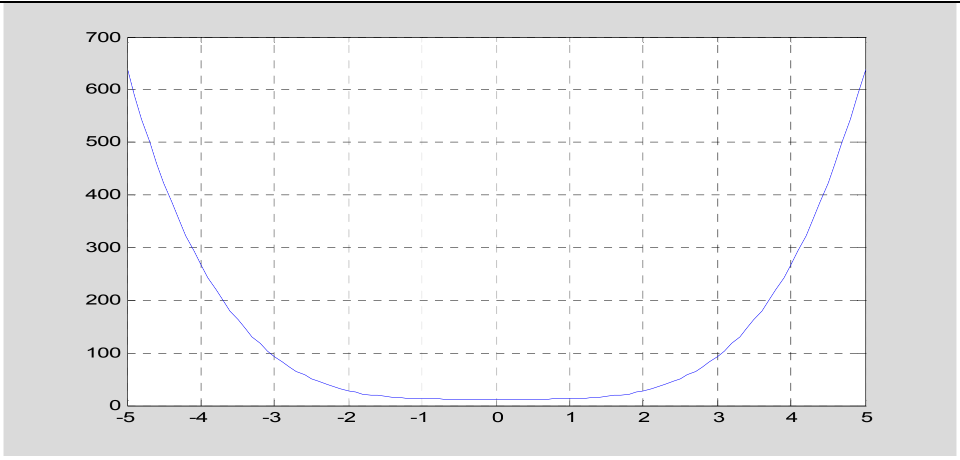


Figure 2-4: Plot of $f(x) = x^4 + 12$.

Definition: $\left.\frac{df}{dx}\right|_{x=x^*} = 0$ then x^* is a **stationary** point (and of special interest in optimization).

We now have a very powerful theorem that can be applied to find and check for minimum/ maximum.

Theorem 2-2: Consider a twice-differentiable function $f(x)$. Suppose at a stationary point x^* we have:

(a) $\left.\frac{d^2f}{dx^2}\right|_{x=x^*} > 0$ then the function takes a **minimum** at x = x*

(b) $\left.\frac{d^2f}{dx^2}\right|_{x=x^*} < 0$ then the function takes a **maximum** at x = x*

(c) $\left.\frac{d^2f}{dx^2}\right|_{x=x^*} = 0$ one cannot conclude anything about the local behavior of x.

Proof: We provide an informal proof here for part (a); see [2] for details. Consider the Taylor series:

$$f(x^* + \Delta x) = f(x^*) + \left.\frac{df}{dx}\right|_{x=x^*} \Delta x + \frac{1}{2!}\left.\frac{d^2f}{dx^2}\right|_{x=x^*} (\Delta x)^2 + \frac{1}{3!}\left.\frac{d^3f}{dx^3}\right|_{x=x^*} (\Delta x)^3 + o(\Delta x)^4$$

Since x^* is a stationary point,

$$\left.\frac{df}{dx}\right|_{x=x^*} = 0$$

we have:

$$f(x^* \pm \Delta x) \approx f(x^*) + \frac{1}{2!}\left.\frac{d^2f}{dx^2}\right|_{x=x^*} (\pm\Delta x)^2$$

Suppose:

$$\left.\frac{d^2 f}{dx^2}\right|_{x=x^*} > 0$$

then

$$\frac{1}{2!}\left.\frac{d^2 f}{dx^2}\right|_{x=x^*} (\Delta x)^2 > 0 \text{ for all } \Delta x \text{ (both positive and negative)}$$

Thus:

$$f(x^* \pm \Delta x) > f(x^*)$$

Thus $f(x^*)$ is less than both $f(x^* + \Delta x)$ and $f(x^* - \Delta x)$ for sufficiently small Δx. Part (b) the theorem can be similarly established.

♦

In fact, Theorem 2-2 can be used to find and detect multiple minima/ maxima.

For example, consider the function $f(x) = x^3 - 3x + 5$. Since

$$\frac{df}{dx} = 3x^2 - 3 = 0 \Rightarrow x = \pm 1$$

we have 2 stationary points $x^* = \pm 1$. Let us now check what happens to the 2nd derivative at these two stationary points.

$$\frac{d^2 f}{dx^2} = 6x$$

At the two stationary points, we have:

$$\left.\frac{d^2 f}{dx^2}\right|_{x^*=1} = 6 > 0$$

$$\left.\frac{d^2 f}{dx^2}\right|_{x^*=-1} = -6 < 0$$

Thus, we can conclude from the above theorem that

$x^* = +1$ is a minimum of the function, while

$x^* = -1$ is a maximum of the function.

We can confirm these findings by plotting the function below.

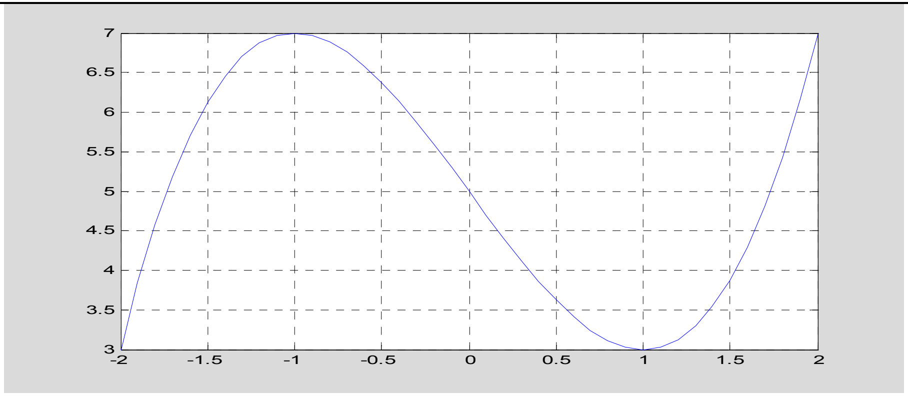


Figure 2-5: Plot of $f(x) = x^3 - 3x + 5$.

So far we have only considered polynomials. Theorem 2-2 applies to any function that is twice differentiable. Consider, for example, the function $f(x) = \sin x - x/\sqrt{2}$.

$$\frac{df}{dx} = \cos x - 1/\sqrt{2} = 0 \Rightarrow \cos x = \frac{1}{\sqrt{2}}$$

There are infinite stationary points for this function (we consider only the positive values):

$$X^* = \left\{ \pi/4 \quad 7\pi/4 \quad 9\pi/4 \quad \ldots \quad 2n\pi \pm \pi/4 \right\}$$

Let us now check what happens to the 2nd derivative at these stationary points.

$$\frac{d^2 f}{dx^2} = -\sin x$$

We have:

$$-\sin(\pi/4) < 0$$

$$-\sin(7\pi/4) > 0$$

$$-\sin(9\pi/4) < 0$$

...

Thus there are infinite minima and maxima for this function, and they alternate, as confirmed below.

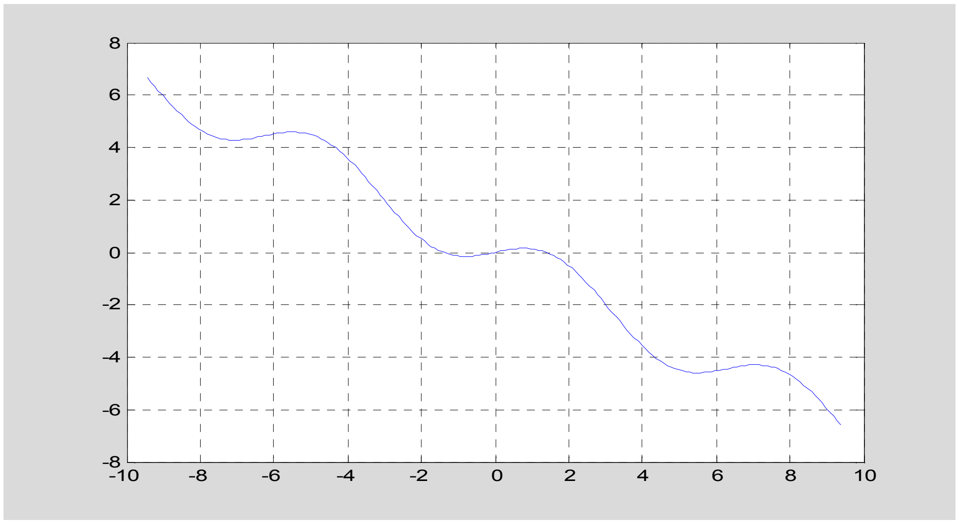


Figure 2-6: Plot of $f(x) = \sin x - x/\sqrt{2}$.

Note: The above example also illustrates a very important concept that Theorem 2-2 only helps us find **local** minimum and maximum. None of the minimum/ maximum is a **global** minimum/ maximum.

Thus one must be contended with finding local minimums and maximums.The next example illustrates that even finding the local minimum and maximum is hard using the above theorem.

Consider the function $f(x) = 3\sin x - x + 0.1x^2 + 0.1\cos(2x)$

$$\frac{df}{dx} = 3\cos x - 1 + 0.2x - 0.2\sin(2x)$$

Theorem 2-2 states that one must find the stationary points where the derivative vanishes. But solving for x in

$$\frac{df}{dx} = 3\cos x - 1 + 0.2x - 0.2\sin(2x) = 0$$

is non-trivial.

2.2 Numerical Methods for Computing Minima in 1-Dimension

For most optimization problems, one must rely on numerical methods for finding stationary points, and classifying them. Broadly, there are two broad classes of numerical methods to find stationary points:

(1) Directly find the minima of $f(x)$

(2) Solve $\frac{df}{dx} = 0$ using a non-linear solver

All numerical algorithms require either a range or an initial guess. Depending on the range/starting points, algorithms will converge to different stationary point/minima. In particular, we will consider the function $f(x) = 3\sin x - x + 0.1x^2 + 0.1\cos(2x)$ as an example; the function is plotted below.

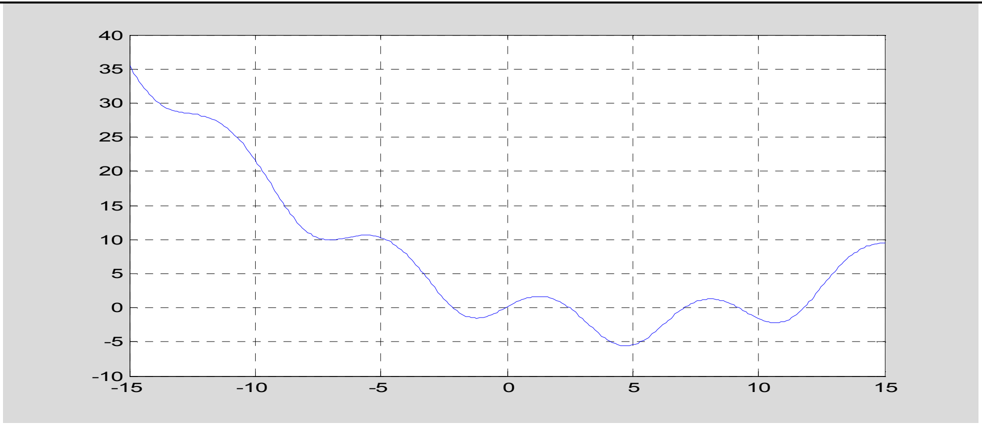


Figure 2-7: Plot of $f(x) = 3\sin x - x + 0.1x^2 + 0.1\cos(2x)$.

2.2.1 MATLAB 'fminbnd' Minimization Routine

First, we will use the MATLAB 'fminbnd' function to find the minimal points by solving:

$$\underset{x}{Min}\ f(x) \tag{2.8}$$

The fminbnd function is called a zeroth order minimizer in that it only requires the function (and *not* its derivative). However, it requires a range within which it will search for a minimum. The Matlab script below demonstrates the use of fminbnd for 3 different starting ranges. The computed minima are -1.19, 0, and 4.73 (Question: Why did the function return '0' in the second case?)

```
f = inline('3*sin(x)-x+0.1*x^2+0.1*cos(2*x)');
fminbnd(f,-2,-1) % range [-2,-1]
fminbnd(f,0,2) % range [0,2]
fminbnd(f,4,5) % range [4,5]
```

Script 2-2: Matlab script for finding minima via fminbnd.

To find a maximum, we only need to negate the function as below.

```
fN=inline('(-1)*(3*sin(x)-x+0.1*x^2+0.1*cos(2*x))');
fminbnd(fN,0,2) % range [0,2]
```

Script 2-3: Matlab script for finding maxima via negation.

We can call matlab files instead of inline functions via fminbnd. For example, create a '.m' file named 'fExample.m' with the following contents.

```
function [f] = fExample(x)
f=3*sin(x)-x+0.1*x^2+0.1*cos(2*x);
```

Script 2-4: Matlab file fExample.m.

Then, the following script shows how fminbnd may now be used.

```
fminbnd(@(x)fExample(x),-2,-1) % range [-2,-1]
```

Script 2-5: Matlab script for finding minimum via fminbnd.

The matlab file allows you to include additional parameters. For example the following file contains parameter as part of the function definition.

```
function [f] = fExample2(x,p)
f=3*sin(x)-x+0.1*x^2+p*cos(2*x);
```

Script 2-6: Matlab file fExample.m.

Then, the following script show how a parameter (of value 0.2) may be passed.

```
fminbnd(@(x)fExample2(x,0.2),-2,-1) % range [-2,-1]
```

Script 2-7: Matlab script for finding minimum via fminbnd.

Having understood, how to *use* 'fminbnd', let us see what algorithms can be used to implement fminbnd? There are many possibilities; we shall consider three, namely: binary search, golden section search, quadratic fitting.

Binary Search

The simplest method is the binary search. Suppose the interval $[x_0, x_1]$ is given within which we must seek a minimum. We sample $f(x)$ at three points, namely $x_0, x_2 \,\&\, x_1$, where x_2 is the mid-point:

$$x_2 = (x_0 + x_1)/2 \tag{2.9}$$

Let the sampled values be $f_0, f_2 \,\&\, f_1$. First, check if the x_{tol} or f_{tol} is met. If so, we are done, else we proceed as follows. We consider $(f_0 + f_2)$ and $(f_2 + f_1)$. If the former is small, then we search again in the interval $[x_0, x_2]$ (via a while loop or a recursive call), else we search in the interval $[x_2, x_1]$. While the binary search is simple, it is slow. (One could speed things up via additional checks. For example, suppose, f_2 is greater than both f_0 and f_1, then, we can assume there cannot exist an internal minimum, and return one of the extremes.)

Golden Section

To speed things up, we can use the golden section method (a highly popular zeroth order method). As before, suppose the interval $[x_0, x_1]$ is given within which we must seek a minimum. We sample $f(x)$ at three points, namely $x_0, x_2 \,\&\, x_1$, where x_2 is NOT the mid-point, but defined via:

$$\frac{x_1 - x_2}{x_2 - x_0} = \alpha \tag{2.10}$$

where $\alpha > 1$ will be determined optimally as described below. The idea behind breaking up the interval into unequal halves is to discard as much as possible, as quickly as possible.

Now, since the interval $[x_2, x_1]$ is larger of the two intervals, by choice, we sample inside $[x_2, x_1]$ at x_3, where:

$$\frac{x_1 - x_3}{x_3 - x_2} = \alpha \tag{2.11}$$

Let the sampled values be f_0, f_2, f_3 & f_1. Now, if $f_3 < f_2$ we repeat the process in the interval $[x_2, x_1]$, else in the interval $[x_0, x_3]$. As in the binary search, the interval shrinks during each iteration.

Now, what should be the value of α for the method to converge as quickly as possible? Observe that, in each iteration, one may end-up choosing $[x_2, x_1]$ or $[x_0, x_3]$ with equal probability. The best case therefore is to make these intervals equal (else the algorithm mau suffer if all choices turned out to be unfavorable), i.e.,

$$x_1 - x_2 = x_3 - x_0 \tag{2.12}$$

Without a loss in generality, let the initial interval $[x_0, x_1]$ be $[0,1]$. Thus, from Equation (2.10), we have:

$$\begin{aligned} 1 - x_2 &= \alpha(x_2) \\ x_2 &= \frac{1}{(1+\alpha)} \end{aligned} \tag{2.13}$$

Similarly, from Equation (2.11), we have:

$$\begin{aligned} 1 - x_3 &= \alpha(x_3 - x_2) \\ x_3 &= \frac{1 + \alpha x_2}{1+\alpha} = \frac{1 + \dfrac{\alpha}{1+\alpha}}{1+\alpha} = \frac{1+2\alpha}{(1+\alpha)^2} \end{aligned} \tag{2.14}$$

Substituting Equations (2.14) and (2.13) in Equation (2.12), we have:

$$\begin{aligned} &1 - \frac{1}{1+\alpha} = \frac{1+2\alpha}{(1+\alpha)^2} \\ &\Rightarrow \\ &\alpha = \frac{1+2\alpha}{(1+\alpha)} \Rightarrow \alpha^2 - \alpha - 1 = 0 \\ &\alpha = \left(1+\sqrt{5}\right)/2 = 1.61803 \quad \text{(Golden ratio)} \end{aligned} \tag{2.15}$$

Thus, $\alpha = 1.61803$, decides how one divides the intervals optimally, hence the name golden section method. (One could speed things up via additional checks. For example, suppose, f_2 is greater than both f_0 and f_1, then, we can assume there cannot exist an internal minimum, and return one of the extremes, and so forth.)

Quadratic Fitting

The idea behind quadratic fitting is to exploit analytical results to speed-up further. The idea is as follows. Given a range $[x_0, x_1]$ within which we seek a minimum, we sample $f(x)$ at three points, namely x_0, x_{mid} & x_1, where $x_{mid} = (x_0 + x_1)/2$. Then, we fit a quadratic polynomial $q(x) = Ax^2 + Bx + C$ (a simple exercise in linear algebra) to these points. Note that the stationary point of the quadratic can be determined by setting the derivative to zero:

$$\frac{dq}{dx} = 2Ax + B = 0$$
$$\Rightarrow x^* = -B/(2A)$$

Further, the second derivative is:

$$\frac{d^2q}{dx^2} = 2A$$

If $A < 0$, then the quadratic (and the function) has a minimum at one of the extremes of the x-range, else the minimum is estimated at:

$$x_{\min} = -B/(2A)$$

If $x_{\min}$ lies in the range $[x_0, x_{mid}]$, then we again search for the minimum within this interval repeat the process (until convergence). Else we search in $[x_{mid}, x_1]$ and repeat the process. The (rudimentary) code below is an implementation of this algorithm.

```
function [xmin, fmin, nIterations] = …
fminViaQuadFit(f,x0,x1,xTol,fTol,maxIterations)
% A rudimentary code for finding the minimum of a 1-D function.
% Via quadratic polynomial fitting.
% Example:
% f = inline('3*sin(x)-x+0.1*x^2+0.1*cos(2*x)');
% [xmin,fmin,nIterations]=fminViaQuadFit(f,-2,-1,1e-5,1e-5,100)
% Warning: the code has not been tested thoroughly
nIterations = 1;
while (1)
    xmid = (x0+x1)/2;
    f0 = feval(f,x0);
    fmid = feval(f,xmid);
    f1 = feval(f,x1);
    % Check if x-range or f-range is within tolerance
    fVec = [f0 fmid f1];
    if (abs(x0-x1) < xTol)||((max(fVec) - min(fVec))< fTol)
        xmin = xmid; fmin = fmid;
        break;
    end
    % If not try fitting a quadratic
    p = polyfit([x0,xmid,x1],[f0,fmid,f1],2);
    if (p(1) < 0) % maxima
       if (f0 < f1)
           xmin = x0; fmin = f0;
       else
           xmin = x1; fmin = f1;
       end
       break;
    end
    % Estimate minimum via fit
    xmin = -p(2)/(2*p(1));
    fmin = feval(f,xmin);
    if (xmin < xmid)
        x1 = xmid; %right limit replaced
    else
        x0 = xmid;%left limit replaced
    end
    nIterations = nIterations +1;
    if (nIterations > maxIterations)
```

```
        xmin = xmid; fmin = fmid;
        disp('Maxiterations exceeded ...unsuccessful termination');
        break;
    end
end
```

Script 2-8: Matlab script fminViaQuadFit.

Below is an example that uses fminViaQuadFit. As before, one can pass Matlab filenames rather than inline functions.

```
f = inline('3*sin(x)-x+0.1*x^2+0.1*cos(2*x)');
fminViaQuadFit(f,-2,-1,1e-6,1e-6,100)
fminViaQuadFit(f,0,2,1e-6,1e-6,100)
fminViaQuadFit(f,4,5,1e-6,1e-6,100)
```

Script 2-9: Matlab script for finding minima via quadratic fit.

2.2.2 MATLAB 'fminunc' Minimization Routine

Next, we shall consider the MATLAB 'fminunc' function to find the minima by solving:

$$\underset{x}{Min}\ f(x) \tag{2.16}$$

via the fminunc Matlab function. fminunc is a *first-order* minimizer in that it (technically) requires that the user provide a derivative. If the user does not provide a derivative, the function internally uses finite differences to compute the derivative. In addition, it requires a starting point (rather than a range). The Matlab script below demonstrates how minima may be computed for 3 different starting points. The computed minima are -1.19, and 4.73. Observe that both are minima.

```
f = inline('3*sin(x)-x+0.1*x^2+0.1*cos(2*x)');
fminunc(f,-1) % initial guess of -1
fminunc(f,1) % initial guess of 1
fminunc(f,5) % initial guess of 5
```

Script 2-10: Matlab script for finding minima via fminunc.

In the above example, we did not provide any gradient information. One can do so as shown below, thereby dramatically improving the convergence (for complex problems). For example, we can create an '.m' file called fWithGrad with contents shown below.

```
function [f,g] = fWithGrad(x)
f = 3*sin(x)-x+0.1*x^2+0.1*cos(2*x);
g = 3*cos(x)-1+0.2*x-0.2*sin(2*x);
```

Script 2-11: Matlab file that returns function and gradient.

Then, we can set the options for gradient usage and call fminunc.

```
op = optimset('GradObj','on','LargeScale','off');
fminunc(@(x) fWithGrad(x),-1,op)
```

Script 2-12: Matlab script for finding minima with gradient.

Having understood the basic syntax of fminunc, let us now understand the logic behind it. Note that, in 1-D, given a function and its derivative, we can move in the direction of decrease to reduce the function. The question is how far can we move?

The problem is referred to as called *step-length determination*. There are various 'rules/ conditions' for step-length determination: Armijo rule, Wolfe conditions, etc[2]. However, a simple and effective 'back-tracking' idea of step-length determination is as follows: take a big-step forward (somewhat arbitrary), and check if the function value has reduced. If so stop, and sample derivative at that point, and repeat. If not, reduce the step-length (back-track), and check. Repeat sampling until the function value has reduced. Implementation of a robust step-length determination algorithm is tricky; further details may be found in [2].

2.2.3 MATLAB 'fsolve' Non-Linear Solver

Next, we will consider the MATLAB 'fsolve' function to find the stationary points, by solving the non-linear equation:

$$\frac{df}{dx} = 0 \tag{2.17}$$

The user must provide the first deritvative of the function and the second derivative. If the second derivative is not provided, internally, fsolve uses finite differences to estimate the second derivative. The Matlab script below demonstrates how stationary points may be computed for 3 different starting points. The computed stationary points are -1.19, 1.28 and 4.73. Observe that all 3 stationary points, but only the 1st and 3rd are minima.

```
dfdx = inline('3*cos(x)-1+0.2*x-0.2*sin(2*x)');
fsolve(dfdx,-1) % initial guess of -1
fsolve(dfdx,1) % initial guess of 1
fsolve(dfdx,5) % initial guess of 5
```

Script 2-13: Matlab script for finding stationary points.

Most non-linear solvers have their roots in the famous Newton-Raphson method. Observe that Equation (2.17) is, in general, a non-linear equation (with multiple solutions), that may be expressed in the following form:

$$G(x) = 0 \tag{2.18}$$

For example, suppose $f(x) = 3\sin x - x + 0.1x^2 + 0.1\cos(2x)$, then the corresponding non-linear equation is:

$$3\cos x - 1 + 0.2x - 0.2\sin(2x) = 0 \tag{2.19}$$

Recall that the Newton-Raphson method for solving such non-linear equations proceeds as follows. Suppose, we start with an initial guess x_0 that will most likely not satisfy Equation (2.18). In other words,

$$G(x_0) \neq 0 \tag{2.20}$$

The question then is, what should be the next guess x_1 such that it comes closer to satisfying Equation (2.18). Let $x_1 = x_0 + \Delta x$; thus, we want:

$$G(x_0 + \Delta x) \approx 0 \tag{2.21}$$

Exploiting the familiar Taylor series expansion, we have:

$$G(x_0 + \Delta x) \approx G(x_0) + \Delta x \left.\frac{dG}{dx}\right|_{x_0} \approx 0 \tag{2.22}$$

Therefore, the step size is given by:

$$\Delta x = -\left(\left.\frac{dG}{dx}\right|_{x_0}\right)^{-1} G(x_0) \tag{2.23}$$

In other words, the next 'guess' is

$$x_1 = x_0 - \left(\left.\frac{dG}{dx}\right|_{x_0}\right)^{-1} G(x_0) \tag{2.24}$$

Generalizing this, we have the recurrence equation:

$$x_{n+1} = x_n - \left(\left. \frac{dG}{dx} \right|_{x_n} \right)^{-1} G(x_n) \tag{2.25}$$

Equation (2.25) is the famous Newton-Raphson formula for finding the roots of $G(x)$. Observe that, at each step, we need $G(x)$ and its derivative. But $G(x)$ itself is the derivative of $f(x)$. Thus the Newton-Raphson method is a 2nd order method.

The user must therefore supply the 1st derivative and (ideally) the 2nd derivative of $f(x)$ to fsolve. Below is a Matlab script where the 2nd derivative is also supplied to fsolve.

```
G = inline('3*cos(x)-1+0.2*x-0.2*sin(2*x)');
dGdx = inline('-3*sin(x)+0.2-0.4*cos(2*x)');
op = optimset('Jacobian','on','LargeScale','off');
F = {G dGdx}; %
fsolve(F,-1,op) % initial guess of -1
fsolve(dfdx,1,op) % initial guess of 1
fsolve(dfdx,5,op) % initial guess of 5
```

Script 2-14: Matlab script for finding stationary points.

However, if the 2nd derivative is not provided (see earlier script), fsolve relies on sophisticated finite difference methods to estimate the 2nd derivative.

3 Multivariable Unconstrained Minimization

Highlights

- The Taylor series of a N-variable function, up to two terms, is given by:

  $$f(p_0 + \Delta p) \approx f(p_0) + \Delta p^T \nabla f(p_0) + \frac{1}{2}\Delta p^T \nabla^2 f(p_0) \Delta p$$

  $$\nabla f = \begin{Bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{Bmatrix} \text{ ... called the gradient of the function}$$

  $$\nabla^2 f = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x \partial y} \\ \dfrac{\partial^2 f}{\partial x \partial y} & \dfrac{\partial^2 f}{\partial y^2} \end{bmatrix} \text{ ... called the Hessian of the function}$$

- For an N-variable unconstrained minimization problem: $\underset{\bar{x}}{Min}\ f(\bar{x})$, points that satisfy $\nabla f = 0$ are called stationary points. Further, at a stationary point, if all the eigenvalues of the Hessian $H = \nabla^2 f$ are greater than zero then $f(\bar{x})$ takes a minimum at that point.
- In general, one must rely on numerical methods to find (local) minima & stationary points.
- The line-search concept underlies most numerical methods.
- For quadratic functions exhibiting a minimum: (a) performing a line-search along the N eigenvectors of its Hessian, will yield the exact minimum in N steps, and (b) performing a line-search along any N conjugate directions will yield the exact minimum in N steps.
- For non-quadratic functions: (a) the Powell method of line-search is a popular 0th order line search technique that finds a 'nearest' local minimum, (b) conjugate-gradient method and BFGS are two popular 1st order line search techniques that find a 'nearest' local minimum, and (c) the Newton-Raphson method of line-search is a popular 2nd order line search technique that finds a 'nearest' stationary point.
- Important Matlab functions introduced include `eig, fminunc and fsolve.`

Equation Section (Next) In this Chapter, we will focus on the theory underlying multi-variable unconstrained minimization problems of the form:

$$\underset{\bar{x}}{Min}\ f(\bar{x}) \tag{3.1}$$

Just as 1-D Taylor series played a critical role in the previous Chapter, higher order Taylor series plays an important role in this Chapter.

3.1 Examples of Multi-Dimensional Functions

We will come across a variety of multi-dimensional functions, but a few examples of functions include:

$$\begin{aligned}
&f_1(x,y) = x^2 + y^2 \\
&f_2(x,y) = x^2 - 2xy + 4y^2 - 2x - 4y - 5 \\
&f_3(u,v) = -4u^2 - 6v^2 \\
&f_4(\alpha,\beta) = (\alpha-1)^2 + (\beta^2 - \alpha)^4 \\
&f_5\left(x_1,x_2,x_3\right) = 4x_1^2 + 4x_2^2 + 4x_3^2 - x_1x_2 - x_2x_3 + 10x_3 \\
&f_6(u,v) = \frac{1}{2}100\left(\sqrt{u^2 + (v+1)^2} - 1\right)^2 + \frac{1}{2}500\left(\sqrt{u^2 + (v-1)^2} - 1\right)^2 - (10u + 8v) \\
&f_7\left(x_1,x_2,x_3,x_4\right) = \sin(x_1 + 2x_2 + 3x_2 + 4x_4) \\
&f_8(\alpha,\beta) = u^2 + 3v^2, where\ u\ \&\ v \text{ satisfy:} \begin{bmatrix} 3+\alpha^2 & -1+\alpha \\ -1+\alpha & \sin(\beta) \end{bmatrix} \begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{Bmatrix} 1 \\ -1 \end{Bmatrix}
\end{aligned} \tag{3.2}$$

We shall classify functions by two means:

- *Dimension*: Functions 1, 2, 3, 4, 6 & 8 are of two dimensions, i.e., they involve 2 variables, or N=2 for such functions. On the other hand N = 3, for functions 5, while N = 4 for function 6.
- *Whether a function is quadratic or not*: A function is quadratic if it is a polynomial, and the highest power with respect to all variables is no more than 2. Functions 1, 2, 3 & 5 are quadratic, while the rest are not. Note that, in function 8, although the function is quadratic with respect to $u\ \&\ v$ it is not quadratic with respect to the design variables $\alpha\ \&\ \beta$, hence function-8 is *not* quadratic.

3.1.1 Quadratic Functions

Quadratic functions play a very important role in optimization for at least two reasons: (1) they are easy to analyze from a theoretical perspective, and one can therefore predict the behavior of optimization algorithms on such functions (they are often used as test-cases during code development), and (2) non-quadratic functions start behaving like quadratic functions as one approaches a minimum or maximum (more on this later).

It is convenient to express quadratic functions in a compact form. For example, consider

$$f_2(x,y) = x^2 - 2xy + 4y^2 - 2x - 4y - 5 \tag{3.3}$$

As one can verify, the function can be expressed as:

$$f_2(x,y) = \frac{1}{2}\begin{Bmatrix} x & y \end{Bmatrix} \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix} + \begin{Bmatrix} x & y \end{Bmatrix} \begin{Bmatrix} -2 \\ -4 \end{Bmatrix} - 5 \tag{3.4}$$

Thus, in a compact form

$$f_2(x,y) = \frac{1}{2}\begin{Bmatrix} x & y \end{Bmatrix} Q \begin{Bmatrix} x \\ y \end{Bmatrix} + \begin{Bmatrix} x & y \end{Bmatrix} b + c \tag{3.5}$$

where

$$Q = \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix}; b = \begin{Bmatrix} -2 \\ -4 \end{Bmatrix} c = -5 \tag{3.6}$$

One can verify that the entries of Q are given by:

$$Q = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x \partial y} \\ \dfrac{\partial^2 f}{\partial x \partial y} & \dfrac{\partial^2 f}{\partial y^2} \end{bmatrix} \tag{3.7}$$

The linear and constant terms are easy to extract by observation. More generally, any quadratic function can be expressed as:

$$f(\overline{x}) = \frac{1}{2}\overline{x}^T Q \overline{x} + \overline{x}^T b + c \tag{3.8}$$

Thus, for example, for $f_5(x_1, x_2, x_3) = 4x_1^2 + 4x_2^2 + 4x_3^2 - x_1x_2 - x_2x_3 + 10x_3$

$$f_5 = \frac{1}{2}\begin{Bmatrix} x_1 & x_2 & x_3 \end{Bmatrix} \begin{bmatrix} 8 & -1 & 0 \\ -1 & 8 & -1 \\ 0 & -1 & 8 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} + \begin{Bmatrix} x_1 & x_2 & x_3 \end{Bmatrix} \begin{Bmatrix} 0 \\ 0 \\ 10 \end{Bmatrix} + 0$$

3.2 Taylor Series in Higher Dimensions

Recall the 2-dimensional Taylor series (we will typically need only two terms) [1]:

$$f(x_0 + \Delta x, y_0 + \Delta y) = \begin{cases} f(x_0, y_0) + \left(\left.\dfrac{\partial f}{\partial x}\right|_{(x_0,y_0)} \Delta x + \left.\dfrac{\partial f}{\partial y}\right|_{(x_0,y_0)} \Delta y \right) \\ + \dfrac{1}{2}\left(\left.\dfrac{\partial^2 f}{\partial x^2}\right|_{(x_0,y_0)} (\Delta x)^2 + 2\left.\dfrac{\partial^2 f}{\partial x \partial y}\right|_{(x_0,y_0)} (\Delta x)(\Delta y) + \left.\dfrac{\partial^2 f}{\partial y^2}\right|_{(x_0,y_0)} (\Delta y)^2 \right) \\ + o\left(\left|\Delta x, \Delta y\right|\right)^3 \end{cases} \tag{3.9}$$

To generalize this to higher dimensions, we shall write this in a vector form as:

$$f(p_0 + \Delta p) \approx f(p_0) + \Delta p^T \nabla f(p_0) + \frac{1}{2}\Delta p^T \nabla^2 f(p_0) \Delta p \tag{3.10}$$

where

$$p_0 = \begin{Bmatrix} x_0 \\ y_0 \end{Bmatrix}; \Delta p = \begin{Bmatrix} \Delta x \\ \Delta y \end{Bmatrix}$$

$$\nabla f^T = \begin{Bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{Bmatrix} \text{ ... called the gradient of the function} \tag{3.11}$$

$$\nabla^2 f = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x \partial y} \\ \dfrac{\partial^2 f}{\partial x \partial y} & \dfrac{\partial^2 f}{\partial y^2} \end{bmatrix} \text{ ... called the Hessian of the function}$$

Example: Find the 2nd order Taylor series of $f(x,y) = x^2 - 2xy + 4y^2 - 2x - 4y - 5$ at $(0,0)$.

Solution: $f(p_0 + \Delta p) \approx f(p_0) + \nabla f^T(p_0)\Delta p + \dfrac{1}{2}\Delta p^T \nabla^2 f(p_0)\Delta p$

$$p_0 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}; \Delta p = \begin{Bmatrix} \Delta x \\ \Delta y \end{Bmatrix}$$

$$\nabla f = \begin{Bmatrix} 2x - y - 2 \\ -2x + 8y - 4 \end{Bmatrix}$$

$$\nabla f^T(p_0) = \begin{Bmatrix} -2 & -4 \end{Bmatrix}$$

$$\nabla^2 f = \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix}$$

$$\nabla^2 f(p_0) = \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix}$$

Resulting in:

$$f(p_0 + \Delta p) \approx 5 + \begin{Bmatrix} \Delta x & \Delta y \end{Bmatrix} \begin{Bmatrix} -2 \\ -4 \end{Bmatrix} + \frac{1}{2} \begin{Bmatrix} \Delta x & \Delta y \end{Bmatrix} \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix} \begin{Bmatrix} \Delta x \\ \Delta y \end{Bmatrix}$$

Observe that, for quadratic functions, 2-term Taylor series is sufficient to capture the function exactly.

♦

Now consider a function in N-dimensions $f(x_1, x_2, ..., x_N)$. As before, let:

$$p^0 = \begin{Bmatrix} x_1^0 \\ x_2^0 \\ \cdots \\ x_N^0 \end{Bmatrix} \text{ and } p = \begin{Bmatrix} x_1 \\ x_2 \\ \cdots \\ x_N \end{Bmatrix} \tag{3.12}$$

$$\nabla f = \left\{ \frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \dots \quad \frac{\partial f}{\partial x_N} \right\}^T \quad \text{... the gradient of the function} \tag{3.13}$$

$$\nabla^2 f = H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_N} \\ \dots & \dots & \dots & \dots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \frac{\partial^2 f}{\partial x_N \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix} \text{... the Hessian of the function} \tag{3.14}$$

Then the Taylor series in a vector form is as before:

$$f(p_0 + \Delta p) = f(p_0) + \Delta p^T \nabla f(p_0) + \frac{1}{2} \Delta p^T \nabla^2 f(p_0) \Delta p + o(\Delta p)^3 \tag{3.15}$$

Question: Find the 2nd order Taylor series of the function:

$f(\{x_1, x_2, x_3, x_4\}) = \sin(x_1 + 2x_2 + 3x_2 + 4x_4)$ at the origin.

Taylor series is the perhaps the most valuable mathematical tool for proving theorems and understanding/ developing robust algorithms in optimization.

3.3 Optimality Criteria in Multi-Dimensions

Recall that the stationary point played a critical role in 1-dimension. We now have the generalization of this concept to higher dimensions.

Definition: Consider a function $f(\{x_1, x_2, ..., x_N\})$. Suppose at a point $p^* = \{x_1^*, x_2^*, ..., x_N^*\}$ $\nabla f(p^*) = 0$, then p^* is a **stationary point**.

Note that $\nabla f(p^*) = 0$ is a vector equation with N-components, i.e., all N partial derivatives must vanish at a stationary point.

Theorem 3-1: Consider a function $f(\{x_1, x_2, ..., x_N\})$ that is twice differentiable at $p^* = \{x_1^*, x_2^*, ..., x_N^*\}$, where p^* is a stationary point, and let H be the Hessian defined in Equation (3.14), then

(a) If all the eigen values of H at p^* are positive then p^* is a **minimum.**

(b) If all the eigen values of H at p^* are negative then p^* is a **maximum.**

(c) If some the eigen values of H at p^* are positive, while others are negative then p^* is a **saddle point.**

(d) If some of the eigen values at p^* are zero (while the rest of the eigen-values are of the same sign) then the data is **inconclusive**.

Proof: Observe that the matrix H is symmetric. Therefore, all its eigen-values are real and mutually orthogonal [Reference]. Further, suppose $\{\rho_i, v_i\}_{i=1,N}$ are the eigen-pairs of H , i.e.,

$$Hv_i = \rho_i v_i$$

Then any vector Δp can be expressed as a linear combination of v_i, i.e.,

$$\Delta p = \sum_i a_i v_i$$

Now consider the expression:

$$f(p^* + \Delta p) \approx f(p^*) + \Delta p^T \nabla f(p^*) + \frac{1}{2}\Delta p^T H \Delta p$$

Since p^* is a stationary point, the 2nd term vanishes. Further:

$$H\Delta p = H\sum_i a_i v_i = \sum_i a_i H v_i = \sum_i a_i H v_i = \sum_i a_i \rho_i v_i$$

Thus, the 3rd term reduces to:

$$\frac{1}{2}\Delta p^T H \Delta p = \frac{1}{2}\left(\sum_i a_i v_i\right)^T H\Delta p = \frac{1}{2}\left(\sum_i a_i v_i\right)^T \left(\sum_i a_i \rho_i v_i\right)$$

Further, since the eigen-vectors are orthonormal, i.e., $v_j^T v_i = 0, i \neq j$, and $v_i^T v_i = 1$

$$\frac{1}{2}\Delta p^T H \Delta p = \frac{1}{2}\sum_i a_i^2 \rho_i$$

leading to:

$$f(p^* \pm \Delta p) \approx f(p^*) + \frac{1}{2}\sum_i a_i^2 \rho_i$$

From above follows part (a) and part (b).

♦

3.4 Illustrative Examples

Example: Find all the stationary points of $f(x,y) = x^2 - 2xy + 4y^2 - 2x - 4y - 5$, and classify these points as minimum/maximum/saddle-point, etc.

Solution: First compute the gradient

$$\nabla f = \begin{Bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{Bmatrix} = \begin{Bmatrix} 2x - 2y - 2 \\ -2x + 8y - 4 \end{Bmatrix}$$

Set both terms to zero and solve for the stationary points.

$$\begin{Bmatrix} 2x - 2y - 2 \\ -2x + 8y - 4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

$$\Rightarrow \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix}\begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{Bmatrix} 2 \\ 4 \end{Bmatrix}$$

This is a linear system of equations and can be solved, for example in Matlab, as shown below.

```
A = [2 -2; -2 8];
b = [2 4]';
A\b
```

Script 3-1: Matlab script for solving a 2x2 system.

Thus, the stationary point is $p^* = (x^*, y^*) = (2,1)$. Let us next find the Hessian of $f(x,y) = x^2 - 2xy + 4y^2 - 2x - 4y - 5$

$$H = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x \partial y} \\ \dfrac{\partial^2 f}{\partial x \partial y} & \dfrac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix}$$

Since the Hessian is a constant, the eigen values of this matrix are 1.39 & 8.6, and are computed below in Matlab:

```
H = [2 -2; -2 8];
eig(H)
```

Script 3-2: Matlab script for finding the eigen- values of the Hessian.

Since both eigenvalues are positive, we conclude that the function takes a (local) minimum at $p* = (x*, y*) = (2,1)$. One can indeed confirm this by plotting the function around this point in Matlab.

```
f = 'x^2-2*x*y+8*y^2 -2*x-4*y+5';
ezmesh(f,[-1.5 -1.1 1.1 1.5]);
```

Script 3-3: Matlab script for ezmesh.

The resulting plot is shown below.

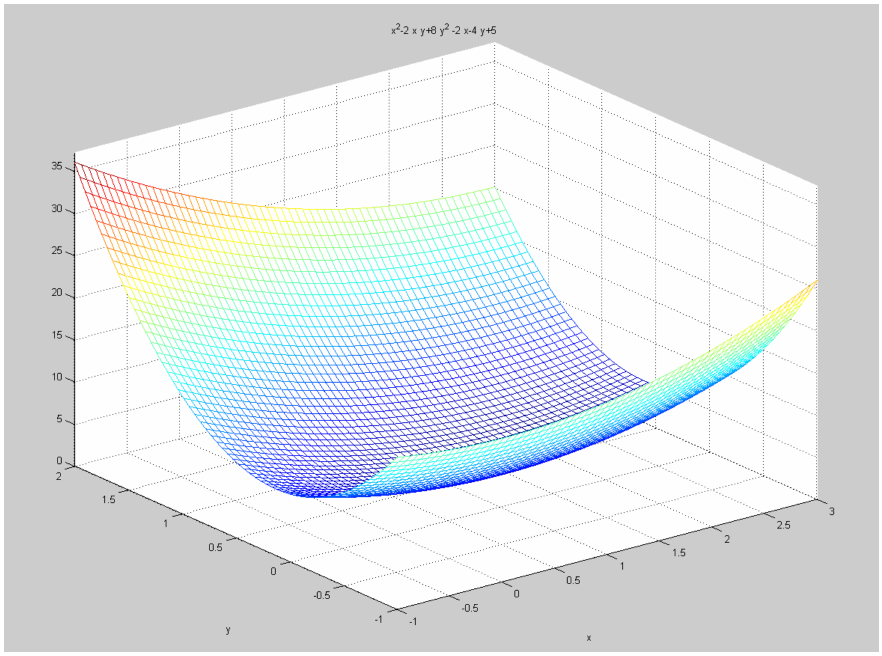


Figure 3-1: Plot of $f(x, y) = x^2 - 2xy + 4y^2 - 2x - 4y - 5$.

For 2-D problems, one can also visualize the contours instead of the surface height. Given a 2-D function f, a contour is the set of (xy) points where the function takes a constant value, i.e., $f(x, y) = c$, for some value of c. For different values of c, one obtains different contours. For example, consider the function $f(x, y) = x^2 + y^2$. It is clear that for positive values of c, one obtains concentric circles (contours). We shall use the plotContour routine below for plotting contours; the details needs of this code is not essential at this point.

```
function plotContour(varargin)
% Routine for plotting contours
% Usage 1: For quadratic functions
% plotContour(Q,b,c,xRange,yRange);
% Example: plotContour([1 0; 0 1],[0 0],0,[-1 1],[-1 1]);
%
% Usage 2: For more general functions
% plotContour(f,xRange,yRange,OptionalParams);

if (isreal(varargin{1})) % usage 1
    type = 1; % Quadratic
    Q = varargin{1};b = varargin{2}(:); c = varargin{3};
    xRange = varargin{4}; yRange = varargin{5};
else % usage 2
    type = 2; % NonQuadratic
    f = varargin{1};
    xRange = varargin{2}; yRange = varargin{3};
end
N = 10; % subdivision and number of contours
x = xRange(1):(xRange(2)-xRange(1))/N:xRange(2);
y = yRange(1):(yRange(2)-yRange(1))/N:yRange(2);
[X,Y] = meshgrid(x,y);
Z = zeros(size(X));
% since f may not be vectorized, we have to loop thru
for i = 1:size(X,1)
    for j = 1:size(X,2)
        xBar = [X(i,j); Y(i,j)];
        if (type==1)
            Z(i,j) = 0.5*xBar'*Q*xBar+xBar'*b+c;
        else
            Z(i,j) = feval(f,xBar,varargin{4:end});
        end
    end
end
[c,h] = contour(X,Y,Z,N); clabel(c,h);axis('equal')
grid on;
```

Script 3-4: Matlab script plotContour.

The above function may be used as follows.

```
Q = [2 0; 0 2];
b = [0 0];
c = 0;
plotContour(Q,b,c,[-1 1],[-1 1])
```

Script 3-5: Contour plotting via plotContour.

The resulting contour plot is shown below.

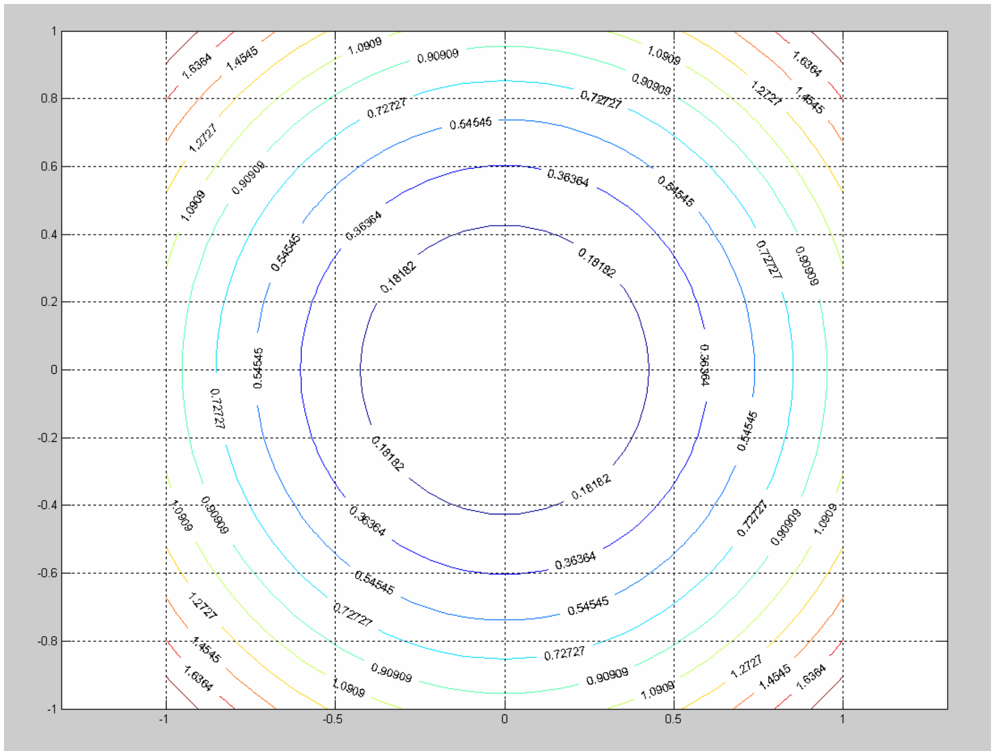


Figure 3-2: Contour plot of $f(x,y) = x^2 + y^2$.

Note that: (1) there are infinite contours for a function,(2) for every point (x_0, y_0) there exists a unique contour passing through that point, however, for a given value of the right hand side value c, a contour may not exist, and (3) no two contours may cross each other, and (4) at every point (x_0, y_0), the gradient of the function, i.e., ∇f (a 2-dimensional vector), is perpendicular to the contour passing through that point, and points in the direction of increasing values of f.

Returning back to our example where $f(x,y) = x^2 - 2xy + 4y^2 - 2x - 4y - 5$, observe that:

$$Q = \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix}$$

$$b = \begin{bmatrix} -2 \\ -4 \end{bmatrix}$$

$$c = 5$$

its contour is shown below.

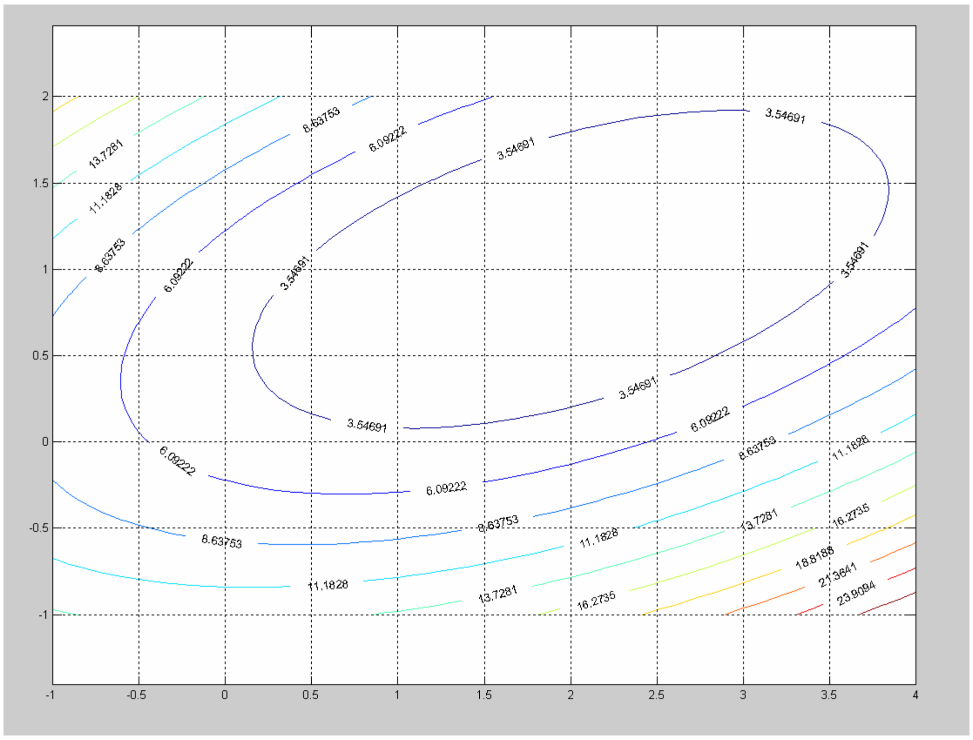


Figure 3-3: Contour plot of $f(x,y) = x^2 - 2xy + 4y^2 - 2x - 4y - 5$.

Example: Find the stationary point and determine its type for the function $f(x,y) = -4x^2 - 6y^2$

Solution: As before, we arrive at the following Matlab script:

```
Q = [-8 0; 0 -12];
b = [0 0]';
c = 0;
Q\b
eig(Q)
plotContour(Q,b,c,[-1 1],[-1 1]);
```

Script 3-6: Matlab script for $f(x,y) = -4x^2 - 6y^2$.

The stationary point is $(0,0)$ and it is a **maximum** point since the eigen values are -8 & -12. The 2-D contour is shown below; as one can observe that the contour values increase towards the origin, i.e., the function takes a maximum.

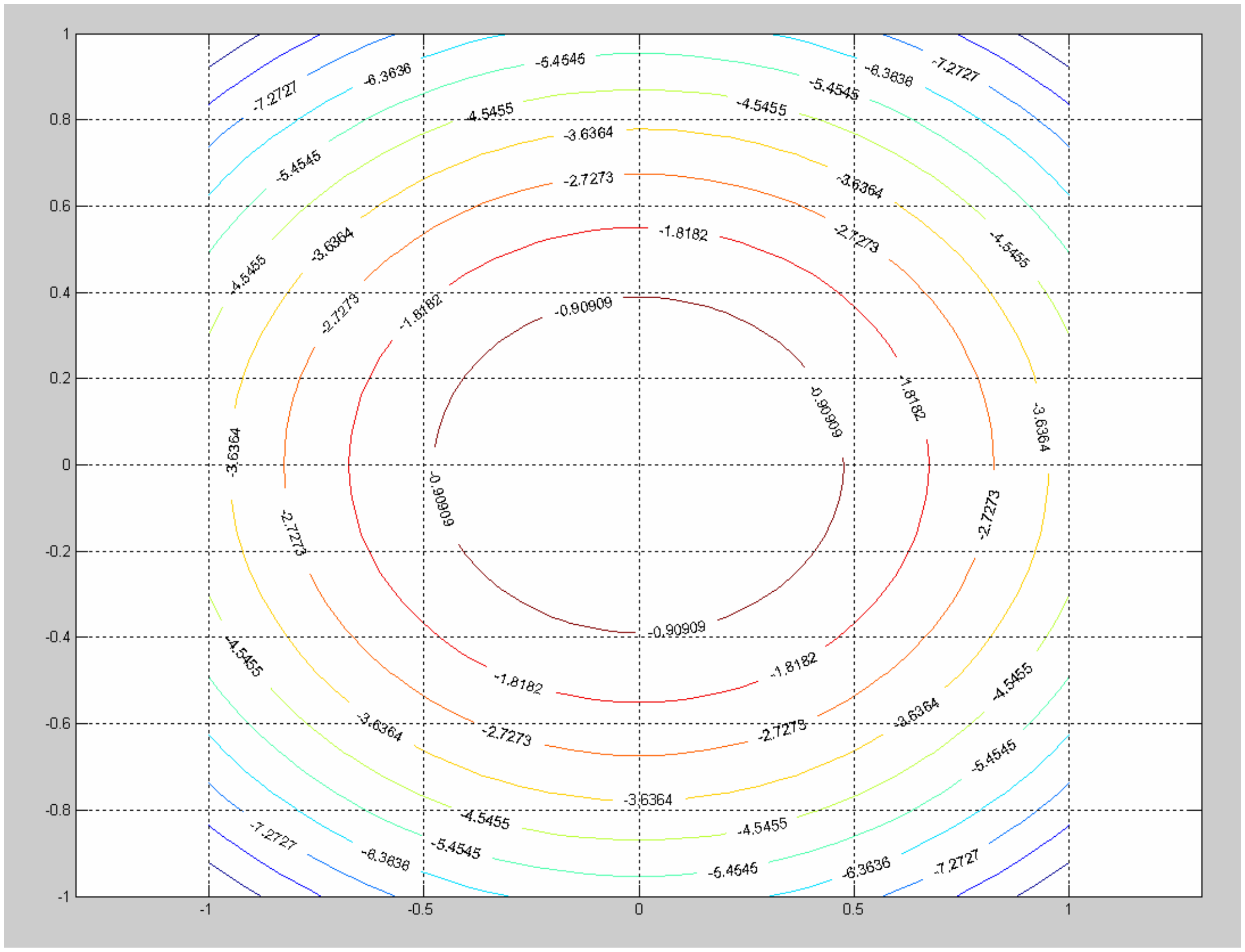


Figure 3-4: Contour plot of $f(x,y) = -4x^2 - 6y^2$.

♦

Example: Find the stationary point and determine its type for the function
$f(x,y) = -x^2 - 3y^2 + 9xy + 5x + 10y$

Solution: The stationary point is $(-1.7391, -0.9420)$ and it is a **saddle** point since the eigenvalues are 5.21 & -13.21, i.e., of opposite signs. It is a minimum along x and a maximum along y, therefore a saddle-point, as illustrated below.

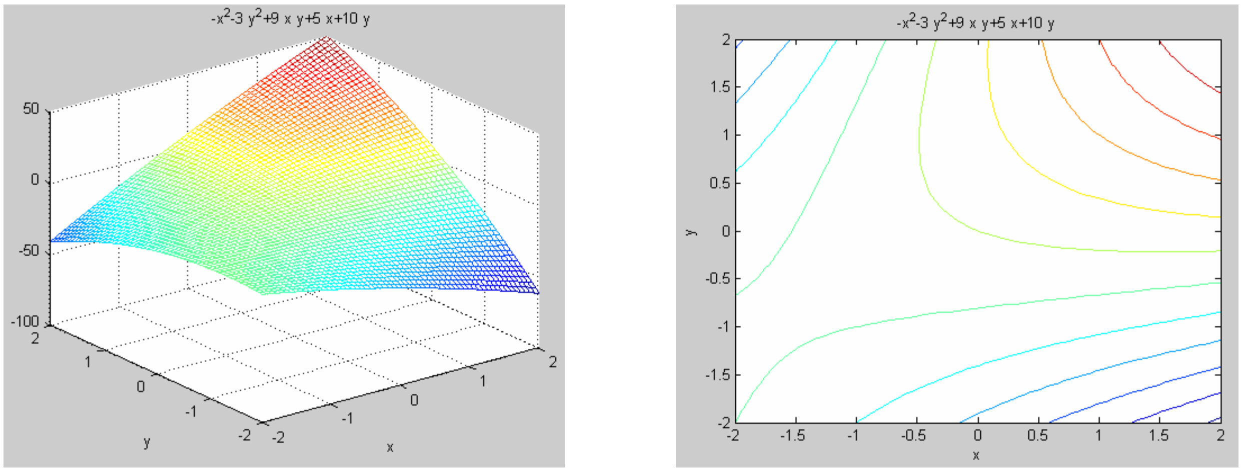


Figure 3-5: Plot of $f(x,y) = -1x^2 - 3y^2 + 9xy + 5x + 10y$.

♦

Example: Consider the function:

$$f = 4x_1^2 + 4x_2^2 + 4x_3^2 + 4x_3^2 - x_1x_2 - x_2x_3 - x_3x_4 + 10x_3 + 8x_4 .$$

Find the stationary values and the nature of these stationary values.

Solution: Find the gradient of the function and set it equal to zero to find the stationary points:

$$\nabla f = \begin{Bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \\ \dfrac{\partial f}{\partial x_3} \\ \dfrac{\partial f}{\partial x_4} \end{Bmatrix} = \begin{Bmatrix} 8x_1 - x_2 \\ 8x_2 - x_1 - x_3 \\ 8x_3 - x_2 - x_4 + 10 \\ 8x_4 - x_3 + 8 \end{Bmatrix} = 0$$

The four equations can be written in a matrix form:

$$\begin{bmatrix} 8 & -1 & 0 & 0 \\ -1 & 8 & -1 & 0 \\ 0 & -1 & 8 & -1 \\ 0 & 0 & -1 & 8 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ -10 \\ -8 \end{Bmatrix}$$

Solving the linear system, the stationary point is $(0.0225, -0.180, \text{-}1.42, 1.18)$, and the Hessian is(at this point).

$$H=\begin{bmatrix}\frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\partial x_4}\\ \frac{\partial^2 f}{\partial x_1\partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\partial x_4}\\ \cdots & \cdots & \cdots & \cdots\\ \frac{\partial^2 f}{\partial x_4\partial x_1} & \frac{\partial^2 f}{\partial x_4\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_4^2}\end{bmatrix}=\begin{bmatrix}8 & -1 & 0 & 0\\ -1 & 8 & -1 & 0\\ 0 & -1 & 8 & -1\\ 0 & 0 & -1 & 8\end{bmatrix}$$

Note that the Hessian is a constant since the function is a quadratic, i.e., the highest order of any variable is two. In fact, for quadratic functions, the Hessian is exactly the same as the matrix A constructed above to find the stationary points. The eigen values of this matrix are $(6.38, 7.38, 8.62, 9.62)$. Since all the eigen-values are positive, the stationary point is a minimum. ♦

Of course, it is not always possible to find the stationary points analytically. For example, consider the function (that we will come across again in the next Section):

$$f(u,v)=\frac{1}{2}100\left(\sqrt{u^2+(v+1)^2}-1\right)^2+\frac{1}{2}500\left(\sqrt{u^2+(v-1)^2}-1\right)^2-(10u+8v) \tag{3.16}$$

The figure plots the above function as a function of (u,v). Observe that the function exhibits both a minimum close to origin and a maximum further away from it.

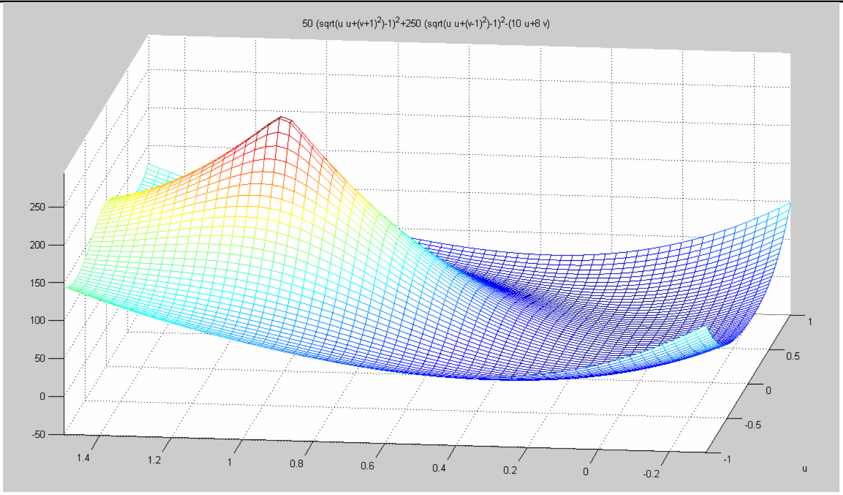


Figure 3-6: The surface plot of Equation (3.16).

To construct a contour plot, we first create a function as shown below.

```
function [f] = fNonQuadratic(xBar)
u = xBar(1); v = xBar(2)
f = 0.5*100*(sqrt(u*u+(v+1)^2)-1)^2+0.5*500*(sqrt(u*u+(v-1)^2)-1)^2-(10*u+8*v);
```

Script 3-7: Matlab function fNonQuadratic.

Then, we call plotContour as shown below.

```
plotContour(@fNonQuadratic,[-1 1],[-1 2])
```

Script 3-8: Contour plotting via plotContour.

Indeed, from the contour plot, observe that the minimum occurs approximately at (0.25,0.1), and a maximum approximately at (0, 1). We must rely on numerical methods to find both these stationary points.

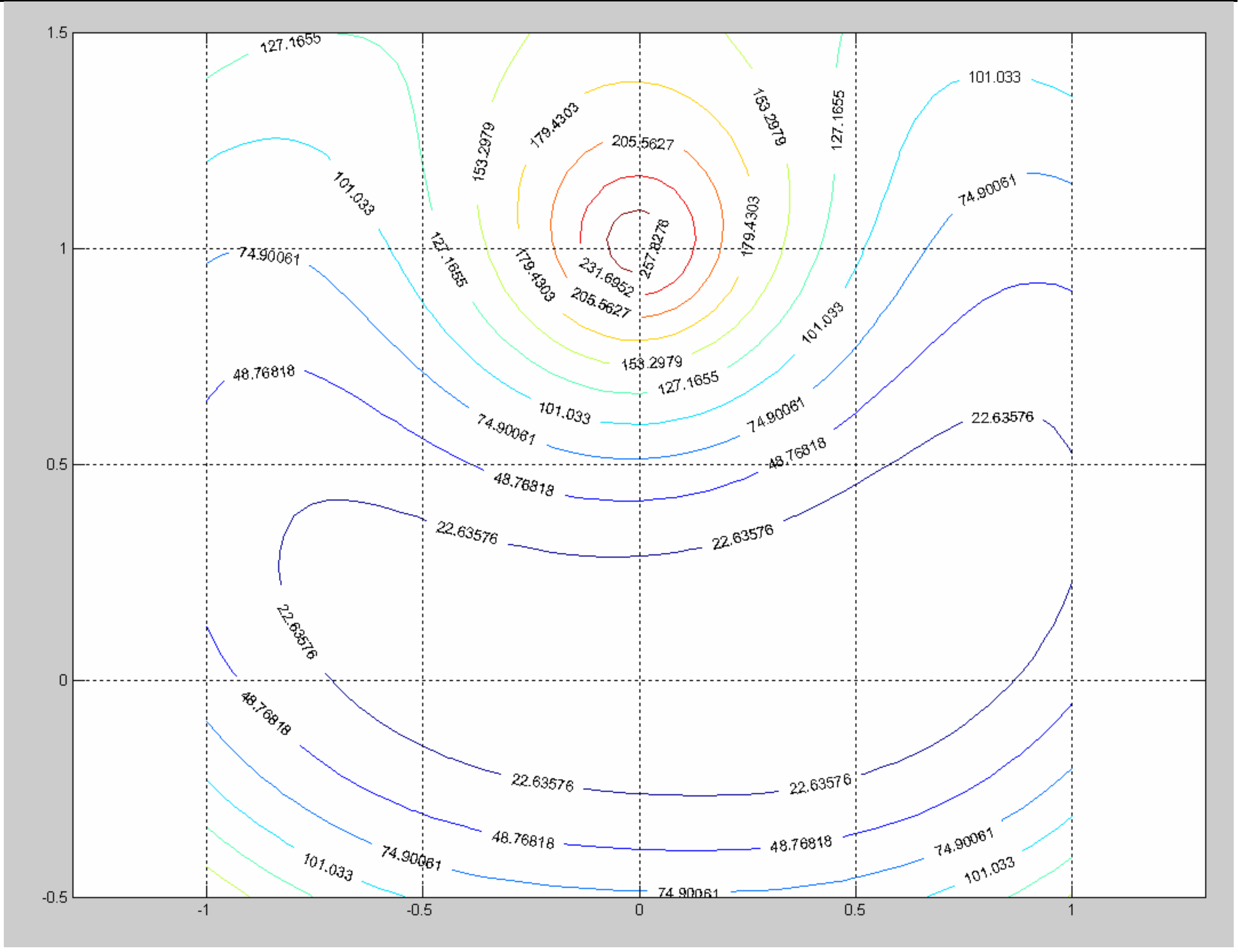


Figure 3-7: Contour plot of Equation (3.16)

Recall that a function is quadratic if it is a polynomial with the highest power with respect to any variable being no more than 2. If a function is quadratic, it is easy to see that there is only **one stationary point**, and the **Hessian is a constant**. Note that:

1. The matrix Q is defined only for quadratic functions
2. On the other hand, the Hessian matrix H is defined for all types of functions, and it happens to be equal to Q for quadratic functions

3.5 Zeroth Order Minimization Algorithms

As in the case of 1-dimensional minimization, we must, in general, rely on numerical algorithms to find minima. First, we shall consider zeroth order numerical algorithms (that only require the function value and not its derivatives) to solve:

$$\underset{\bar{x}}{Min}\ f(\bar{x}) \tag{3.17}$$

3.5.1 Random/Grid Search

The simplest 0th order algorithm is a random search algorithm where we sample $f(\overline{x})$ at random points within a given-space, and pick the point $\overline{x}^*$ with the lowest value. Engineers often try this approach when the nature of the underlying problem is unfamiliar. However, random search techniques are slow, and converging to a true local minimum is unlikely to happen. Many global search techniques are a variation of random searcg, for example: genetic algorithms

An improvement over random search is a grid search, where we sample $f(\overline{x})$ over a coarse grid, and then zoom-in with finer-and-finer grids to converge on a local minimum. Grid-search is easy to implement, and can be particularly useful if there are numerous minima within a given grid. It is however computationally prohibitive when the number of design variables is large.

3.5.2 Line Search Concept

A variety of 0th order algorithms, including the popular Powell's algorithm discussed next, rely on the 'line-search' concept which essentially reduces an N-dimensional minimization problem to a series of 1-dimensional minimization problem. Line-search is easiest to explain in 2-dimensions, but can be easily generalized to N-dimensions.

Consider a 2-D function $f(x,y)$ that we wish to minimize. Let us sample the function at some initial point (x_0,y_0). Our objective is to find a second suitable point (x_1,y_1). Towards this end, imagine that a direction (d_x,d_y) of search has been chosen at (x_0,y_0) (we shall study how these directions may be chosen later on). Now imagine a straight line passing through (x_0,y_0), and along the direction (d_x,d_y). Observe that all points along this straight-line may be expressed as:

$$(x,y) = (x_0,y_0) + \alpha(d_x,d_y)$$
$$-\infty < \alpha < \infty$$

Line-search essentially amounts to searching for a minimum along this straight-line, i.e., finding the value of the scalar α, a 1-D problem.

For example, consider the function $f(x,y) = 3x^2 + 4y^2 - xy + 6x - 9y$, whose minimum we wish to find. Let the starting point be the origin (0, 0). Let the direction of search be (1,1). The straight line is given by:

$$(x,y) = (0,0) + \alpha(1,1), i.e.,$$
$$(x,y) = (\alpha,\alpha)$$

Now substitute this back into the function, i.e.,

$$f_{1D}(\alpha) = 3\alpha^2 + 4\alpha^2 - \alpha\alpha + 6\alpha - 9\alpha = 6\alpha^2 - 3\alpha$$

Observe that the problem has been reduced to a 1-D problem. Differentiating with respect to α, we have:

$$\frac{df_{1D}}{d\alpha} = 12\alpha - 3 = 0$$
$$\Rightarrow \alpha = 1/4$$

Thus, the optimal point along this line is (0.25, 0.25). Is this a true minimum? Is gradient of $f(x,y) = 0$?

We can now start at this point, and search along a different line, for example, along (1, 0). Thus the straight line is:

$$(x,y) = (0.25, 0.25) + \alpha(1,0), i.e.,$$

$$(x,y) = (0.25 + \alpha, 0.25)$$

Substitute into $f(x,y)$ and repeat the process to find that the $\alpha = -29/24$.

In practice, we can implement the above line-search process via a MATLAB code. Before we discuss an implementation of line-search, we discuss how the concept may be extended to N-dimensions. Suppose the initial sample is $\overline{x}^0 = (x_1^0, x_2^0, ..., x_N^0)$, and the direction of search be $\overline{d}^0 = (d_1^0, d_2^0, ..., d_N^0)$. The point that lie on the straight line passing through $\overline{x}$ and in the direction $\overline{d}$ satisfy:

$$(x_1, x_2, ..., x_N) = (x_1^0, x_2^0, ..., x_N^0) + \alpha(d_1^0, d_2^0, ..., d_N^0)$$

$$-\infty < \alpha < \infty$$

For general quadratic functions of the form:

$$f(\overline{x}) = \frac{1}{2}\overline{x}^T Q \overline{x} + \overline{x}^T b + c$$

let $\overline{x} = \overline{x}^0 + \alpha d^0$ be the straight line. Thus:

$$f_{1D}(\alpha) = \frac{1}{2}\left(\overline{x}^0 + \alpha d^0\right)^T Q \left(\overline{x}^0 + \alpha d^0\right) + \left(\overline{x}^0 + \alpha d^0\right)^T b + c$$

i.e., the minimum occurs when

$$\alpha = -\frac{d^T(Q\overline{x}^0 + b)}{d^T Q d}$$

The script below shows an implementation of quadratic line search.

```
function [xmin,fmin] = lineSearchQuadratic(x0,dir,Q,b,c)
% f(alpha) = A*(alpha^2) + B*(alpha) + C
x0 = x0(:);
b = b(:);
dir = dir(:);
alpha = -dir'*(Q*x0+b)/(dir'*Q*dir);
xmin = x0+dir*alpha;
fmin = 0.5*xmin'*Q*xmin + xmin'*b + c;
```

Script 3-9: Quadratic line search.

For non-quadratic functions, line-search is usually implemented in two stages: in the first stage, an interval $[0, \alpha_{\max}]$ is determined, within which a local minimum is expected to lie. Then, standard techniques such as Golden-Section may be used to find the local minimum within this interval. The script below shows an implementation of line-search.

```
function [xmin,fmin] = lineSearchNonQuadratic(f,x0,d,xTol,fTol)
% An implementation of line-search in two-stages
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Stage 1 of Line-search: Find an interval [0,alphaMax]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Normalize the direction
d = d(:); % column format
d = d/norm(d);
x0 = x0(:);
deltaAlpha = 100*xTol; % arbitrary
f0 = f(x0);
```

```
fPlus = f(x0+deltaAlpha*d);
fMinus = f(x0-deltaAlpha*d);
alphaMax = deltaAlpha; % current interval
% Check if d is a descent direction
fCurrent = f0;
if (fPlus < f0)
   d = d; %direction d is descent
   fNext = fPlus;
else
   d = -d; % flip direction
   fNext = fMinus;
end
% Keep increasing alphaMax until we see an increase in f
% If the function continues to decrease indefinitely, stop
while (fNext < fCurrent)
      alphaMax = 2*alphaMax; % geometric progression
      fCurrent = fNext; % replace
      fNext = f(x0+alphaMax*d);
       if (fNext < -1e20) % some large number
            xmin = x0+alphaMax*d;
            fmin = fNext;
            return
       end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Stage 2 of Line-search: Find minimum in [0,alphaMax]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Can use any 1-D line search for Stage-2
op = optimset('TolFun',fTol,'TolX',xTol);
[alphaS,fmin]=fminbnd(@(alpha)f1D(alpha,x0,d,f),0,alphaMax,op);
xmin = x0 + alphaS*d;
return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function fValue = f1D(alpha,x0,d,f)
% 1-D reduction of the N-dimension function
fValue = f(x0+alpha*d);
return;
```

Script 3-10: Matlab non-quadratic line-search.

Before demonstrating line-search, let us create a 'm' file that captures the function in Equation (3.16), i.e.,

$$f(u,v) = \frac{1}{2}100\left(\sqrt{u^2+(v+1)^2}-1\right)^2 + \frac{1}{2}500\left(\sqrt{u^2+(v-1)^2}-1\right)^2 - (10u+8v)$$

as below.

As an example, let us carry out a line-search on the above function from the origin (0,0) in the direction (0.4 0.1). The code is shown below, along with some plotting commands. With a single line search, the xmin is (0.45581, 0.11395). (The 'exact' answer returned by fminunc is 0.439446922780612 0.082927406262034). Try a different direction!

```
ezcontour(@(x,y)fNonQuadratic(x,y), [-1 1 -0.3 1.5]);hold on;
x0 = [0 0];
d = [0.4 0.1]-x0;
[xmin,fmin]= lineSearchNonQuadratic(@(xBar)fNonQuadratic(xBar),x0,d,1e-5,1e-5)
axis('equal');
plot(x0(1),x0(2), 'o')
plot(xmin(1),xmin(2), '*')
plot([x0(1) xmin(1)],[x0(2) xmin(2)]);
```

Script 3-11: Matlab script for demonstrating line-search.

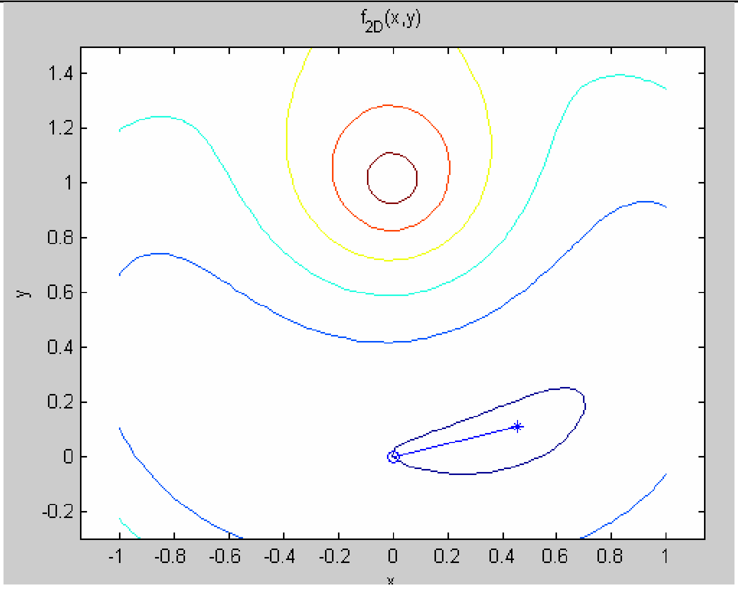


Figure 3-8: Contour plot with single line-search.

3.5.3 Coordinate Cycling

In coordinate cycling, the direction of search is chosen along each coordinate direction. Suppose the initial sample is $\bar{x}^0 = (x_1^0, x_2^0, \ldots, x_N^0)$, then the direction of search during the first iteration is $\bar{d}^0 = (1,0,\ldots,0)$. Let the minimum along this straight line be $\bar{x}^1 = (x_1^1, x_2^1, \ldots, x_N^1)$, then the next direction is $\bar{d}^1 = (0,1,\ldots,0)$. After the last direction search $\bar{d}^{N-1} = (0,0,\ldots,1)$, we start all over again with $\bar{d}^N = (1,0,\ldots,0)$.

```
function [xMin,fMin,iter] = coordinateCycling(f,xBar0,xTol,fTol,maxIterations)
% Coordinate cycling algorithm in N dimensions
N = length(xBar0); % dimension of the problem
D = eye(N,N); % direction matrix for coordinate cycling
xCurrent = xBar0;
fCurrent = feval(f,xCurrent);
iter = 1;
while (iter < maxIterations)
    m = mod(iter-1,N)+1;
    d = (D(:,m));
    [xNew,fNew] = lineSearchNonQuadratic(f,xCurrent,d,xTol,fTol);
    if (N == 2)
      plot([xCurrent(1) xNew(1)],[xCurrent(2) xNew(2)],':r');
    end
    if((norm(xNew-xCurrent)<xTol)||(abs(fNew-fCurrent)< fTol))
        xMin = xNew;
        fMin = fNew;
        break;
    end
    xCurrent = xNew;
    iter = iter+1;
end
xMin = xNew;
fMin = fNew;
```

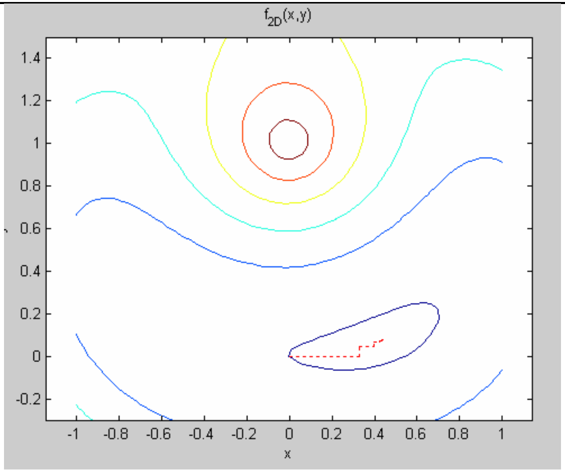
Script 3-12: Matlab script for coordinate cycling.

Below is a script that exploits coordinate cycling

```
x0 = [0 0];
ezcontour(@(x,y)fNonQuadratic(x,y),[-1 1 -0.3 1.5]);
hold on; axis('equal');
[xMin,fMin,iter]=coordinateCycling(@(xBar)fNonQuadratic(xBar),x0,1e-6,1e-6,20)
```

Script 3-13: Matlab script for demonstrating coordinate cycling.

This would result in the plot shown below, with minimum of 0.439365602138160 0.082863271228146 (after 18 iterations). The 'exact' answer returned by fminunc is 0.439446922780612 0.082927406262034).



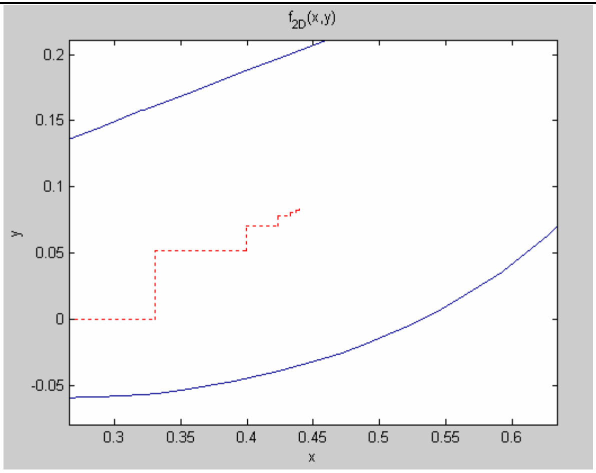


Figure 3-9: Coordinate cycling over contour plot.

The strength and weakness of coordinate cycling can be illustrated via two simple quadratic functions. Consider the quadratic function $f(x,y) = x^2 + 4y^2 - x - y$. If we start at the origin and execute coordinate cycling, we will find that the algorithm converges, within the tolerance specified, to the minimum (0.5, 0.125), in a 3 iterations starting from the origin!

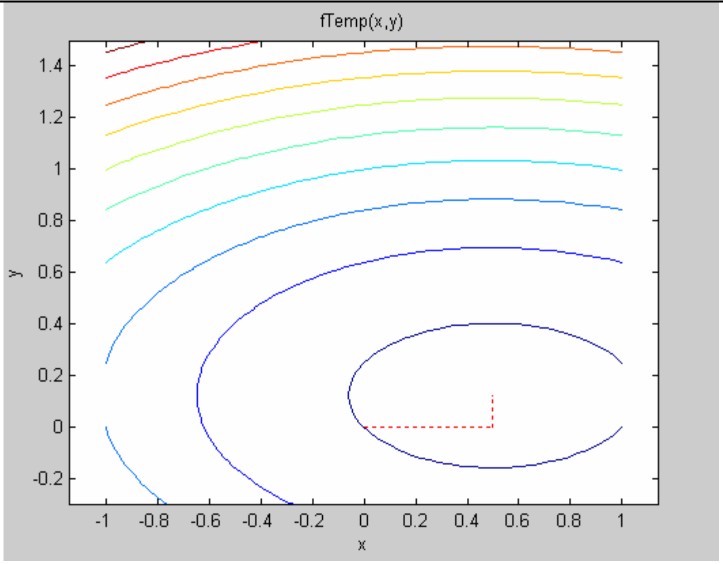


Figure 3-10: Coordinate cycling over contour plot of $f(x,y) = x^2 + 4y^2 - x - y$.

Now consider the function $f(x,y) = x^2 - 2xy + 4y^2 - x - y$ that has an additional 'xy' term that essentially rotates the contour. Now, observe that the coordinate cycling algorithm takes a much larger number of iterations, since the coordinate directions are not optimal.

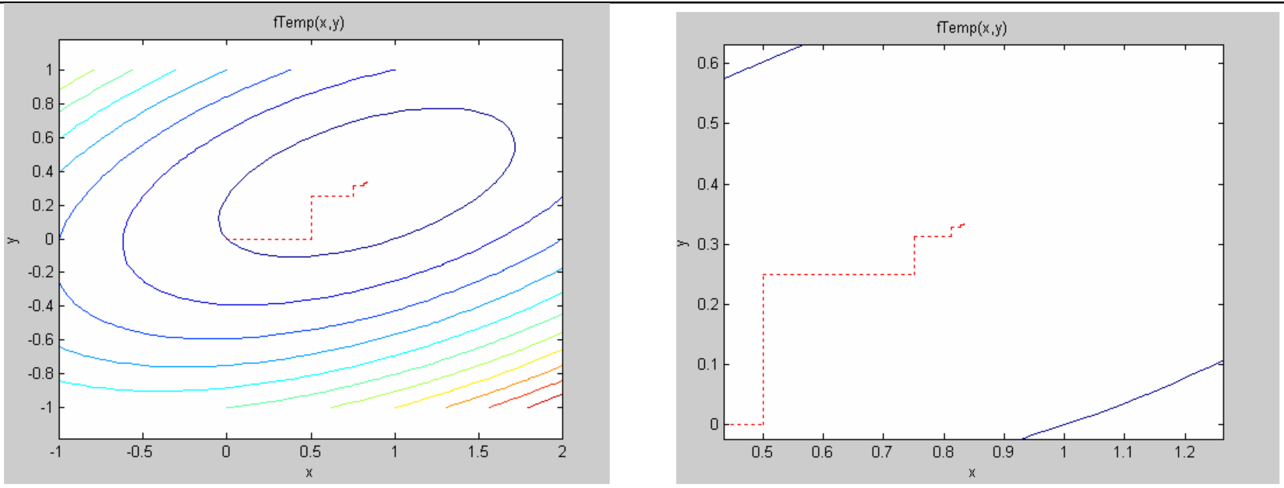


Figure 3-11: Coordinate cycling over contour plot of $f(x,y) = x^2 - 2xy + 4y^2 - x - y$.

Indeed, the ideal directions for any quadratic function are 'obviously' along its principal directions, i.e., along the major and minor axis of the ellipse. But, how does one find the major/minor axis?

3.5.4 Eigen-Vector Search

Observe that the quadratic function $f(x,y) = x^2 - 2xy + 4y^2 - x - y$ can be expressed as:

$$f(x,y) = \frac{1}{2}\begin{Bmatrix} x & y \end{Bmatrix}\begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix}\begin{Bmatrix} x \\ y \end{Bmatrix} + \begin{Bmatrix} x & y \end{Bmatrix}\begin{Bmatrix} -1 \\ -1 \end{Bmatrix}$$

First, we will make the following claim, and then 'prove' the claim informally via an example.

Claim: For any quadratic function:

$$f(x,y) = \frac{1}{2}\begin{Bmatrix} x & y \end{Bmatrix}\mathbf{Q}\begin{Bmatrix} x \\ y \end{Bmatrix} + \begin{Bmatrix} x & y \end{Bmatrix}\mathbf{b} + c$$

the major and minor axis are oriented along the eigen-vectors of Q.

For the example above, let us define a coordinate system (X,Y) oriented along major and minor axis, but with the same origin at the (x,y) system. Let the X axis make an unknown angle θ with the x-axis, i.e.,

$$\begin{Bmatrix} X \\ Y \end{Bmatrix} = R\begin{Bmatrix} x \\ y \end{Bmatrix}$$

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

In the new coordinate system, we have:

$$f(X,Y) = \frac{1}{2}\begin{Bmatrix} X & Y \end{Bmatrix}R^T\begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix}R\begin{Bmatrix} X \\ Y \end{Bmatrix} + \begin{Bmatrix} X & Y \end{Bmatrix}R^T\begin{Bmatrix} -1 \\ -1 \end{Bmatrix}$$

The objective is to find the matrix R such that the cross-coupling XY vanishes, i.e.,

$$R^T \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix} R = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$$

for some constants A and B. From basic linear algebra, the matrix R that accomplishes this task is nothing but the eigen-matrix:

$$H = Q = \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix}$$

In this example, the eigen-vectors of the above matrix are $v_1 = (-0.957, -0.2898)$ and $v_2 = (-0.2898, 0.957)$, and the matrix R is therefore given by:

$$R = \begin{bmatrix} -0.957 & -0.2898 \\ -0.2898 & 0.957 \end{bmatrix}$$

One can verify that:

$$R^T \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix} R = \begin{bmatrix} 1.3944 & 0 \\ 0 & 8.6056 \end{bmatrix}$$

In other words, the exact minimum can be found in two steps as the script below demonstrates where fTemp is a Matlab script file for $f(x,y) = x^2 - 2xy + 4y^2 - x - y$.

```
H = [2 -2; -2 8];
[R,rho] = eig(H);
x0 = [0 0];
[xmin,fmin]=lineSearchNonQuadratic(@(x)fTemp(x),x0,R(:,1),1e-6,1e-6)
[xmin,fmin]=lineSearchNonQuadratic(@(x)fTemp(x),xMin,R(:,2),1e-6,1e-6)
```

Script 3-14: Matlab script for line-search along eigen-vectors.

Indeed, the above conclusion can be generalized to any quadratic function of order N via the following theorem.

Theorem 3-2: Consider the quadratic function of order N:

$$f(\overline{x}) = \frac{1}{2}\overline{x}^T Q \overline{x} + \overline{x}^T b + c$$

Let the eigen vectors of the symmetric real matrix Q be $\{v_i\}_{i=1,N}$, then a series of line-searches along the directions $d_i = v_i$ will yield the exact minimum, starting from any point $\overline{x}_0$.

The theorem is 'interesting' but appears to have very little practical value since: (1) for quadratic functions where N is large, computing the eigen-vectors is expensive. Indeed, it is cheaper to ‘invert’ the Hessian matrix to find the stationary point, and (2) for non-quadratic functions the above theorem does not apply. However, eigen-vector line-search motivates the concept of conjugate directions, discussed next.

3.5.5 Conjugate Directions

Suppose $\{v_i\}_{i=1,N}$ are the eigen-vectors of a symmetric real matrix Q. Observe that since $v_j^T v_i = 0, i \neq j$, we have:

$$v_j^T Q v_i = \rho_i v_j^T v_i = 0$$

i.e., $v_j^T Q v_i = 0$ all pairs of eigen-vectors v_i and v_j. Such pairs are said to be *Q-conjugate*. Further, there are other pairs of vectors s_1 and s_2 (besides eigen-vectors) that are Q-conjugate, i.e., that satisfy: $s_2^T Q s_1 = 0$. For example, consider the matrix

$$Q = \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix}$$

Let $s_1 = (1,0)^T$ be some vector (not an eigen-vector of Q). Let us try and find a vector $s_2 = \{a,b\}^T$ that is Q-conjugate to s_1. By definition:

$$s_2^T Q s_1 = 0, \text{ i.e.,}$$

$$\begin{Bmatrix} a & b \end{Bmatrix} \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix} \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} = 0 \Rightarrow \begin{Bmatrix} a & b \end{Bmatrix} \begin{Bmatrix} 2 \\ -2 \end{Bmatrix} = 0$$

Thus, one may choose $s_2 = \{1,1\}^T$. Let us now try a line-search along s_1 followed by a line search along s_2, as captured by the following script, where fTemp is a Matlab script file for $f(x,y) = x^2 - 2xy + 4y^2 - x - y$.

```
ezcontour(@(x,y)fTemp(x,y),[-1 2 -1 1]);
axis('equal'); hold on;
x0 = [0 0];
s1 = [1 0];
s2 = [1 1];
[xmin,fmin]=lineSearchNonQuadratic(@(x)fTemp(x),x0,s1,1e-6,1e-6)
[xmin,fmin]=lineSearchNonQuadratic(@(x)fTemp(x),xmin,s2,1e-6,1e-6)
```

Script 3-15: Matlab script for line-search along conjugate.

The figure below shows that the above two line searches yields the exact minimum, despite the fact that we did not compute eigenvectors of Q. The 'trick' is that the 2nd vector s_2 complements the search direction s_1 perfectly!

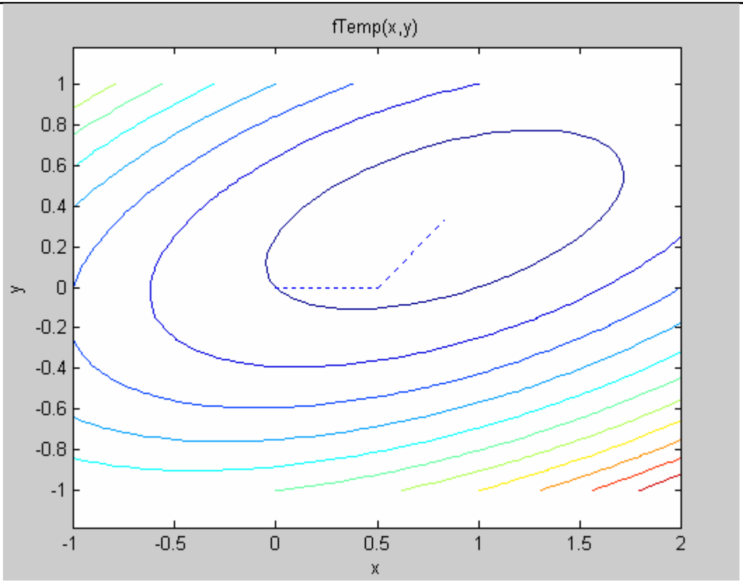


Figure 3-12: Conjugate direction search for $f(x,y) = x^2 - 2xy + 4y^2 - x - y$.

Indeed, the above conclusion can be generalized to any quadratic function of order N as follows.

Theorem 3-3: Consider the quadratic function of order N:

$$f(\overline{x}) = \frac{1}{2}\overline{x}^T Q \overline{x} + \overline{x}^T b + c$$

Let $\{s_i\}_{i=1,N}$ be a set of Q-conjugate vectors, then a line-search along the directions $d_i = s_i$ will yield the exact minimum, starting from any point $\overline{x}_0$.

Now two questions remain: (1) for quadratic functions, how does one find conjugate directions $\{s_i\}_{i=1,N}$ in a computationally efficient manner? (2) how does one apply this theorem to non-quadratic functions?

3.5.6 Powell's Method

Powell's method is one of the best known 0th order algorithm that generates Q-conjugate vectors 'on-the-fly'. The method is best understood in terms of iterations, where each iteration consists of N line-searches. In the first iteration, the N-directions are given by:

$$\left\{d^1, d^2, \ldots, d^N\right\} = \left\{e^1, e^2, \ldots, e^N\right\}$$

i.e., in the first iteration, the directions are essentially the N coordinate directions (just as in coordinate cycling). Let the converged point after 1st iteration be $\overline{x}^N$.

Then Powell suggested that the first direction be replaced by the average search direction thus far. Since the search moved from $\overline{x}^0 \rightarrow \overline{x}^N$, the average search direction during the 1st iteration is:

$$n_1 = \frac{\overline{x}^N - x^0}{\left\|\overline{x}^N - x^0\right\|}$$

Thus, Powell recommended the following N directions for the next direction:

$$\left\{d^1, d^2, \ldots, d^N\right\} = \left\{e^2, e^3, \ldots, e^N, n_1\right\}$$

where as one can observe the first direction has been discarded, and all other directions have been shifted forwarded to make room for a new direction. Now, a second iteration is carried out along the above N directions to result in a new point $\overline{x}^{2N}$. Since the search moved from $\overline{x}^N \rightarrow \overline{x}^{2N}$, the average search direction during the 2nd iteration is:

$$n_2 = \frac{\overline{x}^{2N} - x^N}{\left\|\overline{x}^{2N} - x^N\right\|}$$

In the next iteration, the search directions are:

$$\left\{d^1, d^2, \ldots, d^N\right\} = \left\{e^3, \ldots, e^N, n_1, n_2\right\}$$

and so on. It is shown in [Brent] that, in the limit the N vectors, $\left\{n_1, n_2, \ldots, n_N\right\}$ form a set of Q-conjugate vectors, for a quadratic function. Further, since non-quadratic functions are well approximated by quadratic functions as one approached the stationary point, Powell method converges rapidly to a minimum (however, see caveats of linear dependence of $\left\{n_1, n_2, \ldots, n_N\right\}$ in [Reference], and how to overcome this issue).

3.6 Gradient Computation

In 0th order algorithms, the derivative of the objective function $f(\overline{x})$ was not used. Here we see how the gradient of a function can be used, to speed-up the minimization.

3.6.1 Analytic Gradient

Recall that the gradient of a function is a N-dimensional vector. For quadratic functions of the form:

$$f(\overline{x}) = \frac{1}{2}\overline{x}^T Q\overline{x} + \overline{x}^T b + c$$

the gradient can be determined in a closed form:

$$\nabla f(\overline{x}) = Q\overline{x} + b$$

A Matlab script below returns both the function and the gradient for a generic quadratic function.

```
function [f,grad] = fQuadraticWithGrad(x,Q,b,c)
x = x(:); % column format
b = b(:); % column format
f = 0.5*x'*Q*x + x'*b + c;
grad = Q*x+b;
```

Script 3-16: Matlab function fQuadraticWithGrad.

For non-quadratic functions, analytic gradients must be determined case-by-case. For example, consider again the function:

$$f(u,v) = \frac{1}{2}100\left(\sqrt{u^2+(v+1)^2}-1\right)^2 + \frac{1}{2}500\left(\sqrt{u^2+(v-1)^2}-1\right)^2 - (10u+8v) \qquad (3.18)$$

Its gradient is given by (as one can verify):

$$\nabla f = \begin{Bmatrix} \dfrac{100\left(\sqrt{u^2+(v+1)^2}-1\right)}{\sqrt{u^2+(v+1)^2}}u + \dfrac{500\left(\sqrt{u^2+(v-1)^2}-1\right)}{\sqrt{u^2+(v-1)^2}}u - (10) \\ \dfrac{100\left(\sqrt{u^2+(v+1)^2}-1\right)}{\sqrt{u^2+(v+1)^2}}(v+1) + \dfrac{500\left(\sqrt{u^2+(v-1)^2}-1\right)}{\sqrt{u^2+(v-1)^2}}(v-1) - (8) \end{Bmatrix} \qquad (3.19)$$

For more complex functions, it is not possible to find the gradient in an analytic form. In such circumstances, one can exploit a number of different techniques such as finite difference, complex variables and forward/backward differentiation [References].

A script file capturing the above equation, that we shall use later on is given below.

```
function [f, grad] = fNonQuadraticWithGrad(xBar)
u = xBar(1);
v = xBar(2);
f = 0.5*100*(sqrt(u*u+(v+1)^2)-1)^2+ 0.5*500*(sqrt(u*u+(v-1)^2)-1)^2-(10*u+8*v);
grad(1,1) = 100*(sqrt(u*u+(v+1)^2)-1)*u/sqrt(u*u+(v+1)^2)+ ...
    500*(sqrt(u*u+(v-1)^2)-1)*u/sqrt(u*u+(v-1)^2)-(10);
grad(2,1) = 100*(sqrt(u*u+(v+1)^2)-1)*(v+1)/sqrt(u*u+(v+1)^2)+ ...
    500*(sqrt(u*u+(v-1)^2)-1)*(v-1)/sqrt(u*u+(v-1)^2)-(8);
```

Script 3-17: Matlab script for a non-quadratic function and its gradient.

3.6.2 Finite Difference Gradient

Typically, when an analytic gradient is not available, engineers rely on finite difference to compute an approximate gradient. There are various flavors of finite differences; for example, the forward difference for 2-D problems is:

$$\nabla f\Big|_{(u_0,v_0)} \approx \begin{Bmatrix} \dfrac{f(u_0+\Delta u, v_0) - f(u_0, v_0)}{\Delta u} \\ \dfrac{f(u_0, v_0+\Delta v) - f(u_0, v_0)}{\Delta v} \end{Bmatrix} \tag{3.20}$$

the backward difference:

$$\nabla f\Big|_{(u_0,v_0)} \approx \begin{Bmatrix} \dfrac{f(u_0, v_0) - f(u_0-\Delta u, v_0)}{\Delta u} \\ \dfrac{f(u_0, v_0) - f(u_0, v_0-\Delta v)}{\Delta v} \end{Bmatrix} \tag{3.21}$$

and the most popular central difference:

$$\nabla f\Big|_{(u_0,v_0)} \approx \begin{Bmatrix} \dfrac{f(u_0+\Delta u, v_0) - f(u_0-\Delta u, v_0)}{2\Delta u} \\ \dfrac{f(u_0, v_0+\Delta v) - f(u_0, v_0-\Delta v)}{2\Delta v} \end{Bmatrix} \tag{3.22}$$

Observe that, for the central difference, you need a total of 1+2N function calls (where N is the number of independent variables) at each step of the optimization process.

The problem with finite differences is that finding the optimal step-lengths ($\Delta u, \Delta v$) is not straight forward. Too large a step length will result in a poor approximation of the gradient. On the other too small a step length will also lead to inaccurate gradients due to round-off errors.

For example, consider the function in Equation (3.18). Define the errors:

$$e_u = \frac{\left| \left.\dfrac{\partial f}{\partial u}\right|_{exact} - \left.\dfrac{\partial f}{\partial u}\right|_{finitedifference} \right|}{\left| \left.\dfrac{\partial f}{\partial u}\right|_{exact} \right|} \tag{3.23}$$

We plot below the error for various forward and central finite differences, as a function of Δu, at the point $(1,10)$, where the exact derivative is computed via Equation (3.19).

```
%Script for comparing gradient error
clear all; close all
u = 1;v = 10; % point of interest
[f, grad] = fNonQuadraticWithGrad([u v]);
dfdu_Exact = grad(1);
dfdv_Exact = grad(2);

deltauRange = 10.^[-3:-1:-12];
%%%%%%%%%%%%%%%%%%%  df_du via finite difference*************************
err_Forward = [];
err_Central = [];
for deltau = deltauRange
    dfdu_Forward = (fNonQuadratic(u+deltau,v)-fNonQuadratic(u,v))/deltau;
    err_Forward(end+1)=abs(dfdu_Exact-dfdu_Forward)/ abs(dfdu_Exact);
    dfdu_Central=(fNonQuadratic(u+deltau,v)-fNonQuadratic(u-
deltau,v))/(2*deltau);
    err_Central(end+1) = abs(dfdu_Exact-dfdu_Central)/ abs(dfdu_Exact);
```

```
end
loglog(deltauRange,err_Forward,'b'); hold on;
loglog(deltauRange,err_Central,':r');
h = legend('Forward Difference Error','Central Difference Error');
set(h,'fontsize',14);
h = xlabel('delta u');set(h,'fontsize',14);
h = ylabel('err-u');set(h,'fontsize',14);
h = title('Finite difference error in df/du');set(h,'fontsize',14);
set(gca,'fontsize',14);
```

Script 3-18: Matlab script for generating finite difference gradient error plots.

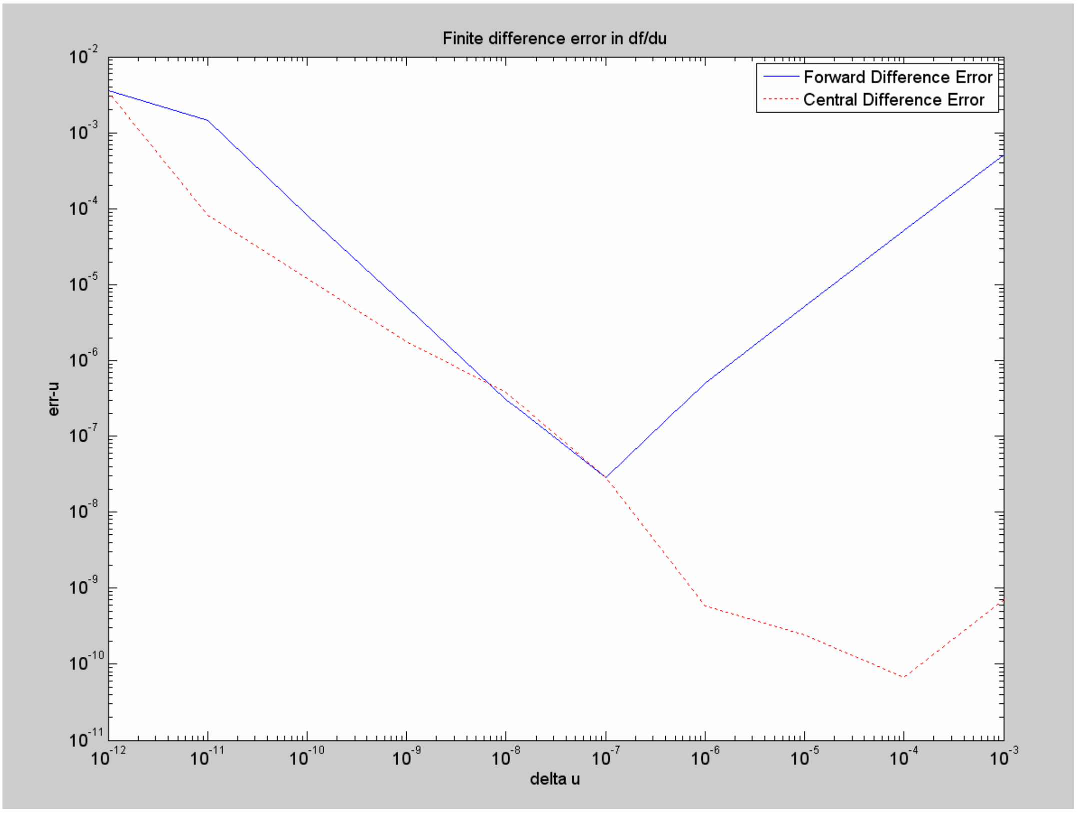


Figure 3-13: Finite difference error at (1,10).

As one can observe:

1. The central difference, in general, results in lower accuracy for a given step-size Δu (but not necessarily).
2. The optimal choice for the step-size Δu is around 10^{-4} for central difference, and 10^{-7} for forward difference

Let us now repeat the same experiment at a different point, say (-1, 1). The resulting plot is shown below.

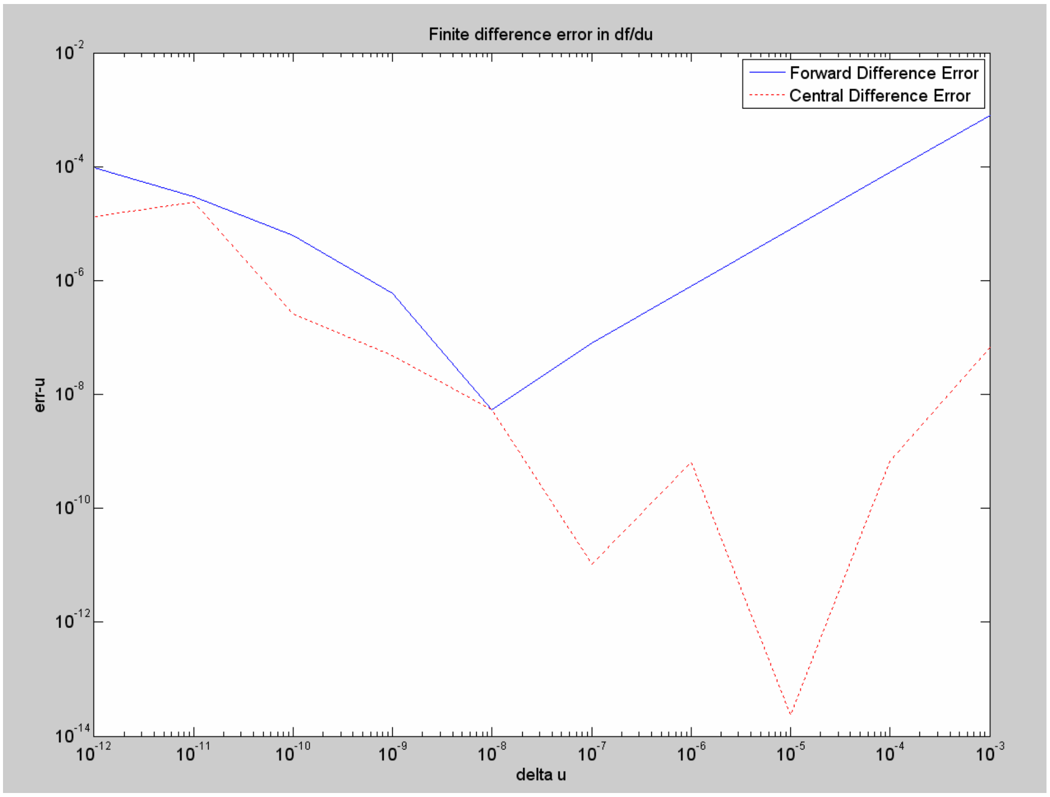


Figure 3-14: Finite difference errors at (1,-1).

Observe that:

3. The optimal choice for Δu is now around 10^{-5} for central difference, and 10^{-8} for forward difference.

Thus, it is impossible to pick *a priori* the optimal step-size. This is one of the serious draw-backs of finite difference.

3.6.3 Complex Variables

The use of complex variables to find derivatives was proposed in [Reference]. The fundamental idea can be captured as follows. Let $f(x_1, x_2, \ldots, x_N)$ be a real function of N variables, then the partial derivative of f with respect to any of its variables, say x_1, may be obtained via the following approximation

$$\frac{\partial f}{\partial x_1} \approx \frac{\operatorname{Im} f(x_1 + ih, x_2, \ldots, x_N)}{h} \tag{3.24}$$

where the function is first evaluated over the complex variable $x_1 + ih$ (h being a small real number, and i being the complex number $\sqrt{-1}$), and then the imaginary component is extracted, and divided by h. Since there is no subtraction involved, round-off errors are minimized.

Consider the function $f(x) = x^2$ of one variable. Let us carry out the above process symbolically to get an insight into Equation (3.24). Observe that

$$f(x + ih) = (x + ih)^2 = x^2 + 2ihx + (ih)^2$$

i.e.,

$$f(x+ih) = x^2 - h^2 + 2ihx$$

The real part being $x^2 - h^2$ and the imaginary part being $2ihx$. Thus, from Equation (3.24):

$$\frac{df}{dx} \approx \frac{2hx}{h} = 2x \tag{3.25}$$

Indeed, for the above quadratic function, we got the exact derivative. Next consider $f(x) = \sin(2\pi x)$. From Equation (3.24):

$$\frac{df}{dx} \approx \frac{\operatorname{Im}[\sin 2\pi(x+ih)]}{h} = \frac{\operatorname{Im}[\sin(2\pi x + 2\pi ih)]}{h} \tag{3.26}$$

Exploiting standard trigonometric expressions:

$$\frac{df}{dx} \approx \frac{\operatorname{Im}[\sin 2\pi x \cos 2\pi ih + \cos 2\pi x \sin 2\pi ih]}{h} \tag{3.27}$$

For sufficiently small h, $\cos 2\pi ih \approx 1$ and $\sin 2\pi ih \approx 2\pi ih$, thus:

$$\frac{df}{dx} \approx \frac{\operatorname{Im}[\sin 2\pi x + 2\pi ih \cos 2\pi x]}{h} \tag{3.28}$$

Extracting the imaginary component, we have the desired result:

$$\frac{df}{dx} \approx \frac{2\pi h \cos 2\pi x}{h} = 2\pi \cos 2\pi x \tag{3.29}$$

In practice, we will be evaluating functions numerically. Thus, for the above method to work, one should be able to numerical compute on complex numbers … Matlab fortunately deals with complex numbers with ease. Indeed, the script below shows how to numerically compute the derivative of $f(x) = \sin(2\pi x)$ at $x = 0.5$.

```
clear i; % just in case i has been defined previously
x = 0.5;
h = 1e-15;
f = sin(2*pi*(x+i*h))
dfdx = imag(f)/h
2*pi*cos(2*pi*x) % as a verification
```

Script 3-19: Matlab script for finding the (complex) derivative of $\sin(2\pi x)$

For more complex problems, the process is fairly straightforward. For Equation(3.18), we have the following script.

```
clear i; % just in case i has been defined previously
u = 1; v = 10;
h = 1e-15;
f = fNonQuadratic(u+i*h,v);
dfdu = imag(f)/h
f = fNonQuadratic(u,v+i*h);
dfdv = imag(f)/h
```

Script 3-20: Matlab script for finding the (complex) derivative via an m-file.

The script used for generating the error plots is given below

```
%Script for comparing gradient error
clear all; close all
u = 1;v = 10; % point of interest
[f, grad] = fNonQuadraticWithGrad([u v]);
dfdu_Exact = grad(1);
dfdv_Exact = grad(2);

deltauRange = 10.^[-3:-1:-12];

%%%%%%%%%%%%%%%%%%%%  df_du via complex*************************
err_Complex = [];
for deltau = deltauRange
    dfdu_Complex = imag(fNonQuadratic(u+i*deltau,v))/deltau;
    err_Complex(end+1) = abs(dfdu_Exact-dfdu_Complex)/abs(dfdu_Exact);
end
figure;
% Added eps below, else plotting does not work
loglog(deltauRange,err_Complex+eps,'b'); hold on;
h = legend('Complex Derivative Error');
set(h,'fontsize',14);
h = xlabel('delta u');set(h,'fontsize',14);
h = ylabel('err-u');set(h,'fontsize',14);
h = title('Complex derivative error in df/du');set(h,'fontsize',14);
set(gca,'fontsize',14);
[xmin,fmin]=lineSearch(@(x)fTemp(x),x0,grad,1e-6,1e-6)
```

Script 3-21: Matlab script for generating complex gradient error plots.

The plot shows the error in the defined in Equation (3.23) at (1,10).

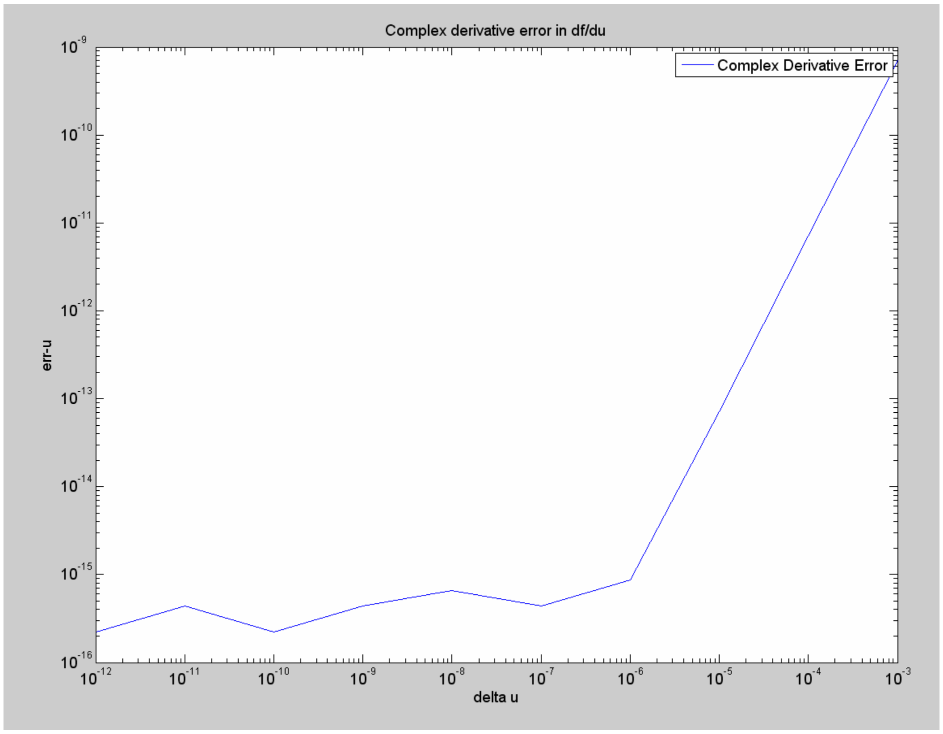


Figure 3-15: Complex derivative error at (1,10).

And the plot below illustrates the same error at (1, -1).

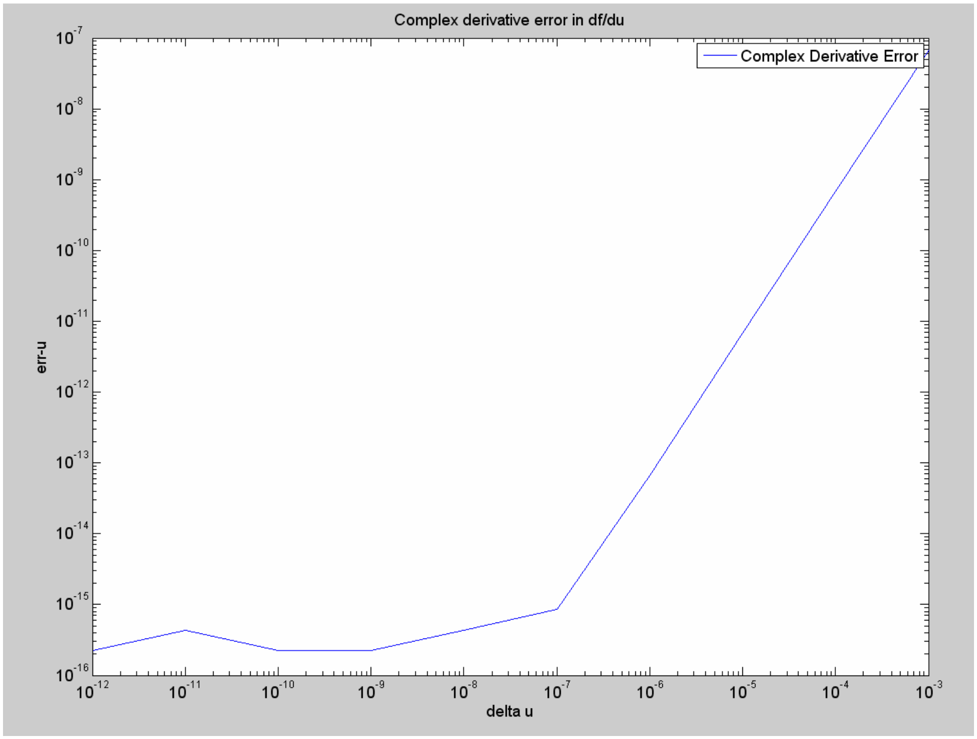


Figure 3-16: Complex derivative error at (1,-1).

As one can observe, the error almost monotonically goes to zero at the step-size drops to zero, i.e., the smaller the h in Equation (3.24), the better the approximation. Further, observe that only one function is needed at every point to obtain both the function value (the real component) and the derivative (the imaginary component). The downside is that the underlying code/kernel must be capable of handling complex numbers.

3.7 First Order Minimization Algorithms

We shall assume that one of the above methods is available to compute the gradient.

3.7.1 *Steepest Descent*

In steepest descent, the direction of search at every point is simply along the gradient, i.e.,
$d^i = \nabla f(\overline{x}^i) \equiv \nabla f^i$

In other words, at $\overline{x}^0$, we search along $d^0 = \nabla f(\overline{x}^0)$. Once the line-search converges to $\overline{x}^1$, we search along $d^1 = \nabla f(\overline{x}^1)$, and so on. The script below illustrates five steps of the steepest gradient line search for $f(x,y) = x^2 - 2xy + 4y^2 - x - y$.

```
xBar0 = [0 0];
Q = [2 -2;-2 8];
b = [-1 -1]';
c = 0;
xPath(1) = xBar0(1); % store all points
yPath(1) = xBar0(2);
for i = 1:5
    [f,grad] = fQuadraticWithGrad(xBar0,Q,b,c);
    [xBarMin,fmin] = lineSearchQuadratic(xBar0,grad,Q,b,c);
    xBar0 = xBarMin; % update for next iteration
    xPath(end+1) = xBarMin(1); %store for plotting
    yPath(end+1) = xBarMin(2);
```

```
end
plotQuadraticContour(@fQuadratic,[-0.1 1.5],[-0.1 0.8],Q,b,c);
hold on;
plotPath(xPath,yPath);
```

Script 3-22: Matlab script for illustrating steepest descent.

Unfortunately, steepest descent is not very efficient since (even for quadratic functions) the gradients at consecutive steps are not conjugate, as can be observed in the figure below. If the directions were conjugate, the search should have terminated in 2 steps. Indeed, steepest descent is similar to coordinate cycling in that every direction is perpendicular to the previous, except that the first direction may not be along the coordinate axis.

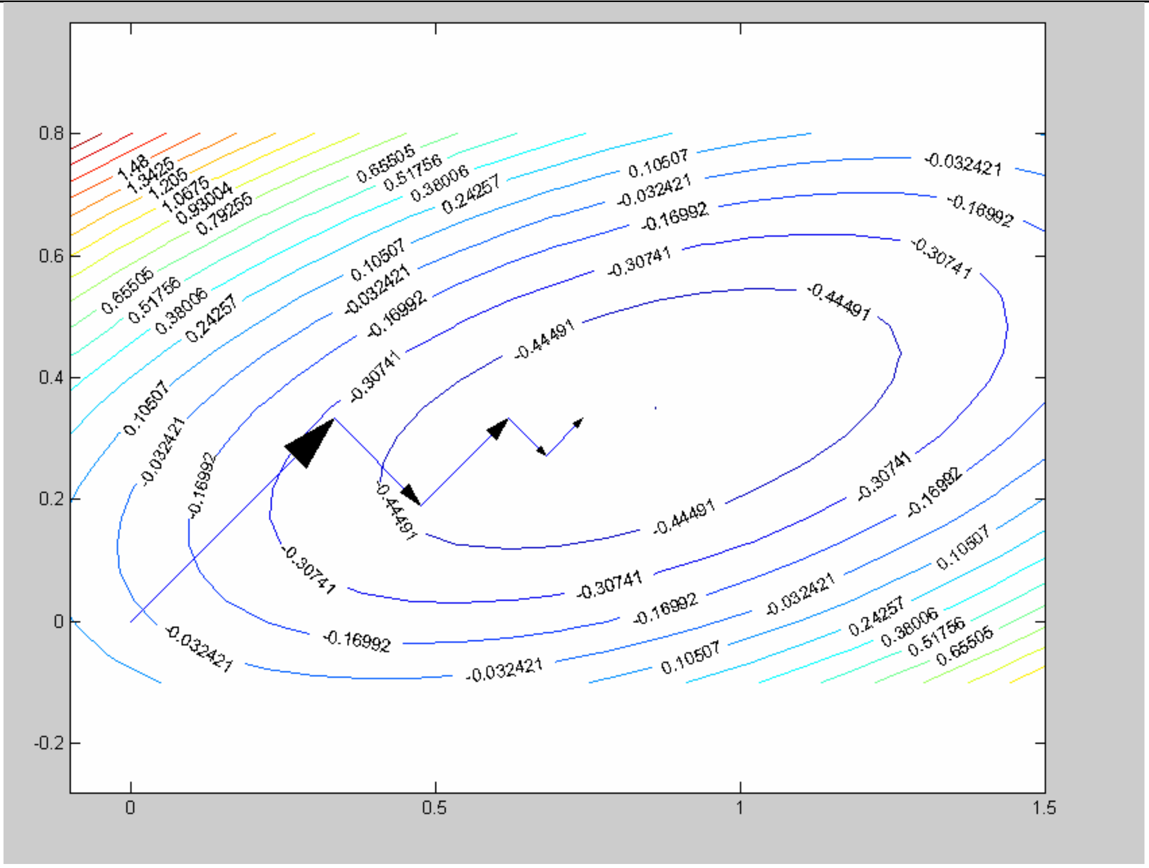


Figure 3-17: A single steepest descent for $f(x,y) = x^2 - 2xy + 4y^2 - x - y$.

3.7.2 Conjugate Gradient for Quadratic Functions

In conjugate gradient, we exploit the gradient, except that the direction of search is modified to be conjugate with respect to all previous directions of search as follows. First, we focus on quadratic functions of the form:

$$f(\overline{x}) = \frac{1}{2}\overline{x}^T Q \overline{x} + \overline{x}^T b + c$$

At the first step, we start at $\overline{x}^0$ and search along the gradient at $\overline{x}^0$

$$d^0 = \nabla f^0$$

After a line search along d^1, let the point be $\overline{x}^1$. In the next step, we let the direction be a linear combination of the current gradient and the previous descent direction.

$$d^1 = \nabla f^1 + \beta d^0$$

The constant β is determined by enforcing the conjugacy condition, i.e.,

$$\left(d^0\right)^T Q d^1 = 0$$

i.e.,

$$\left(d^0\right)^T Q\left(\nabla f^1 + \beta d^0\right) = 0$$

$$\Rightarrow$$

$$\beta = -\frac{\left(d^0\right)^T Q \nabla f^1}{\left(d^0\right)^T Q d^0}$$

i.e.,

$$d^1 = \nabla f^1 - \left(\frac{\left(d^0\right)^T Q \nabla f^1}{\left(d^0\right)^T Q d^0}\right) d^0$$

In the next step, we let:

$$d^2 = \nabla f^2 + \beta d^1$$

The constant is now determined by enforcing conjugacy with respect to the previous direction (this, one can show implies conjugacy with all other directions [Reference]):

$$\left(d^1\right)^T Q d^2 = 0$$

As before, one can show that:

$$\beta = -\frac{\left(d^1\right)^T Q \nabla f^2}{\left(d^1\right)^T Q d^1}$$

resulting in:

$$d^2 = \nabla f^2 - \left(\frac{\left(d^1\right)^T Q \nabla f^2}{\left(d^1\right)^T Q d^1}\right) d^1$$

Thus, in general, the conjugate directions are generated via a simple iterative process:

$$\begin{aligned} & d^0 = \nabla f^0 \\ & For \quad 1 \le i \le N-1 \\ & d^i = \nabla f^i - \left(\frac{\left(d^{i-1}\right)^T Q \nabla f^i}{\left(d^{i-1}\right)^T Q d^{i-1}}\right) d^{i-1} \end{aligned} \tag{3.30}$$

It can be shown (see [2]) that the above expression can be simplified as follows (note that there is no explicit reference to matrix Q):

$$\begin{aligned} & d^0 = \nabla f^0 \\ & For \quad 1 \le i \le N-1 \\ & d^i = \nabla f^i + \left(\frac{\left(\nabla f^i\right)^T \nabla f^i}{\left(\nabla f^{i-1}\right)^T \nabla f^{i-1}}\right) d^{i-1} \end{aligned} \tag{3.31}$$

Then, the following script illustrates an implementation of conjugate gradient for quadratic functions (also referred to as linear conjugate gradient).

```
function [xmin,fmin] = linearConjugateGradient(Q,b,c,x0)
xmin = x0(:); % column format,current minimum
b = b(:); % column format
[f0,G] = fQuadraticWithGrad(xmin,Q,b,c);
dir = G;
GG = G'*G;
for i = 1:length(x0)
    % do a line search
    [xmin] = lineSearchQuadratic(xmin,dir,Q,b,c);
    [fmin,G] = fQuadraticWithGrad(xmin,Q,b,c); % find new gradient
    GGNew = G'*G;
    dir = G +(GGNew)/(GG)*dir; % new direction
    GG = GGNew;
end
```

Script 3-23: Matlab function linearConjugateGradient..

3.7.3 Conjugate Gradient for Non-Quadratic Functions

For non-quadratic functions the above method does not apply since the matrix Q is not defined. The adaptation of the above idea to non-quadratic functions leads to non-linear conjugate gradient methods; two popular ones include Fletcher-Reeves and Polak-Ribere [Reference].

The important task is to generate a set of directions; in FR-CG (Fletcher-Reeves) conjugate gradient the directions are generated exactly as in Equation (3.31) (note that there is no reference to matrix Q):

$$\begin{aligned}&d^0 = \nabla f^0\\ &\text{while not converged}\\ &\qquad d^i = \nabla f^i + \left(\frac{\left(\nabla f^i\right)^T \nabla f^i}{\left(\nabla f^{i-1}\right)^T \nabla f^{i-1}}\right) d^{i-1}\end{aligned} \tag{3.32}$$

Unlike Quadratic problems, explicit termination criteria must be provided. Two types of termination criteria may be desirable:

1. Termination on the line search.
2. Termination of conjugate gradient.

The Fletcher-Reeves non-linear Conjugate Gradient script is given below.

```
function[xmin,fmin]=FRnonlinearConjugateGradient(f,x0,xtol,ftol,gradtol,maxIter)
xmin = x0(:); % column format,current minimum
[f0,G] = feval(f,x0);
dir = G;
GG = G'*G;
iter = 0;
while (sqrt(GG) > gradtol) && (iter < maxIter)
    % do a line search
    [xmin] = lineSearchNonQuadratic(f,xmin,dir,xtol,ftol);
    [fmin,G] = feval(f,xmin); % find new gradient
    GGNew = G'*G;
    dir = G +(GGNew)/(GG)*dir; % new direction
    GG = GGNew;
    iter = iter+1;
end
```

Script 3-24: Matlab function FRNonlinearConjugateGradient.

The script file 'fNonQuadraticWithGrad' to compute a non-linear function and its gradient appears below Equation (3.19). Then, the following script computes the minimum via conjugate gradient, starting from the origin.

```
[xmin,fmin]=  FRnonlinearConjugateGradient(@(x)fNonQuadraticWithGrad(x),[0;0],1e-
6,1e-6,1e-3,20);
```

Script 3-25: Matlab function FRNonlinearConjugateGradient.

An alternate Polak-Ribere conjugate gradient is given by follows:

$$\begin{aligned} & d^0 = \nabla f^0 \\ & For \quad 1 \le i \le N-1 \\ & d^i = \nabla f^i + \left(\frac{\left(\nabla f^i\right)^T \left(\nabla f^i - \nabla f^{i-1}\right)}{\left(\nabla f^{i-1}\right)^T \nabla f^{i-1}} \right) d^{i-1} \end{aligned} \tag{3.33}$$

3.7.4 *Quasi-Newton Method*

Quasi-Newton methods rely on predicting the Hessian of functions and using this prediction to converge rapidly; see [2] for further details.

3.8 Non-Linear Solvers

Recall that the non-linear solvers find the stationary point (rather than the minimum) by numerically solving:

$$\nabla f = 0 \tag{3.34}$$

The same method now applies to higher dimensions. Let:

$$G \equiv \nabla f \tag{3.35}$$

Observe that G is a vector. The objective is to find stationary points $\overline{x}^*$ such that

$$G(\overline{x}^*) = \{0\} \tag{3.36}$$

Suppose, we are at a point $\overline{x}^i$ such that

$$G(\overline{x}^i) \ne \{0\} \tag{3.37}$$

The question is what should be the step size Δx such that:

$$G(\overline{x}^i + \Delta x) \approx \{0\} \tag{3.38}$$

Exploiting Taylor Series:

$$\begin{aligned} & G(\overline{x}^i + \Delta x) \approx G(\overline{x}^i) + \nabla G(\overline{x}^i)\Delta x \approx 0 \\ & \Rightarrow \\ & \Delta x = -\left[\nabla G(\overline{x}^i)\right]^{-1} G(\overline{x}^i) \end{aligned} \tag{3.39}$$

The matrix $\left[\nabla G(\overline{x}^i)\right]$ is called the Jacobian, and is nothing but the Hessian, ie.,

$$\Delta x = -\left[H(\overline{x}^i)\right]^{-1} G(\overline{x}^i) \tag{3.40}$$

Thus, the next point in the iteration is:

$$\overline{x}^{i+1} = \overline{x}^{i} - \left[H(\overline{x}^{i})\right]^{-1} G(\overline{x}^{i}) \tag{3.41}$$

4 Optimization and Structural Mechanics

Highlights

- *Structural systems when subject to external forces come to rest in a state where their potential energy is a minimum.*
- Thus finding the stable equilibrium of structural systems can be posed as an optimization problem
- When the displacements of the structural system can be assumed to be small, the potential energy may be approximated by a quadratic function.

Equation Section (Next) Thus far, we have studied optimization from mathematical and numerical perspectives. However, one can also study optimization from a physical perspective since nature optimizes! In particular, real world structural systems (naturally occurring and man-built) usually come to 'rest' such that their potential energy is at a minimum. One can get a better insight into optimization by studying the physical laws governing such systems. In particular, we shall use a system of connected springs, subject to external forces, as a vehicle for illustration.

4.1 Equilibrium of Spring Systems

Consider the spring system illustrated below consisting of 2 springs that are connected at the center node (marked with an 'o'). The top and bottom nodes are assumed to be fixed (marked with a '*'). Now suppose a force is applied at the center node. The spring system may deflect to an equilibrium position as shown on the right. The exact location will of course depend on the stiffness of the two springs, and the force applied. Traditionally, engineers find the equilibrium via balances of forces, and we shall do the same in a later Section.

However, one can also find the equilibrium position by solving an optimization problem, specifically by minimizing the potentially energy of the spring system [Reference]. In fact, we have the following principle of structural systems: *structural systems when subject to external forces come to rest in a state where their potential energy is a minimum*.

Recall that potential energy Π is the difference between the elastic energy of the system and the external energy [Reference], i.e.,

$$\Pi = U - W \tag{4.1}$$

In this section we shall derive an explicit expression for the potential energy of the spring system, and use optimization routines such as fminunc to find its equilibrium state.

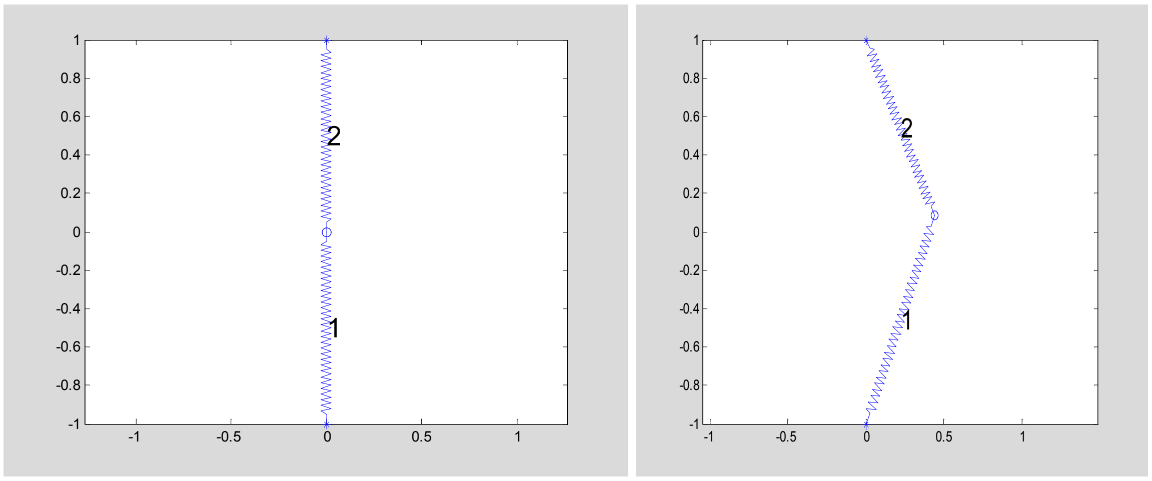


Figure 4-1: A 2-spring system.

Let the spring stiffness of the two springs be k_1 and k_2. When a force (F_x, F_y) is applied on the centre node, let the center node move by (u,v). Since the two springs are initially of unit-length, the change in length of the two springs is given by:

$$\Delta l_1 = \sqrt{(u-0)^2 + (v-(-1))^2} - 1 = \sqrt{u^2 + (v+1)^2} - 1$$
$$\Delta l_2 = \sqrt{(u-0)^2 + (v-1)^2} - 1 = \sqrt{u^2 + (v-1)^2} - 1$$

i.e.,

$$\begin{aligned}\Delta l_1 &= \sqrt{u^2 + (v+1)^2} - 1 \\ \Delta l_2 &= \sqrt{u^2 + (v-1)^2} - 1\end{aligned} \tag{4.2}$$

Given the increase in lengths, recall that the elastic energy of the two-spring system is:

$$U = \frac{1}{2}k_1(\Delta l_1)^2 + \frac{1}{2}k_2(\Delta l_2)^2 \tag{4.3}$$

i.e.,

$$U = \frac{1}{2}k_1\left(\sqrt{u^2+(v+1)^2} - 1\right)^2 + \frac{1}{2}k_2\left(\sqrt{u^2+(v-1)^2} - 1\right)^2 \tag{4.4}$$

Further, the work done by the force as the node moves from origin to (u,v) is:

$$W = F_x u + F_y v \tag{4.5}$$

The potential energy is thus:

$$\Pi(u,v) = U - W = \frac{1}{2}k_1\left(\sqrt{u^2+(v+1)^2} - 1\right)^2 + \frac{1}{2}k_2\left(\sqrt{u^2+(v-1)^2} - 1\right)^2 - \left(F_x u + F_y v\right) \tag{4.6}$$

Note that the potential energy is a function of (u,v). Since the structural system comes to a stable equilibrium where the potential energy is a minimum, we have the following problem:

$$\underset{(u,v)}{Min}\ \Pi(u,v) \tag{4.7}$$

As a specific example, let $k_1 = 100N/m$ and $k_2 = 500N/m$, and let $(F_x, F_y) = (10, 8)$. Thus:

$$\Pi(u,v) = \frac{1}{2}100\left(\sqrt{u^2+(v+1)^2} - 1\right)^2 + \frac{1}{2}500\left(\sqrt{u^2+(v-1)^2} - 1\right)^2 - (10u + 8v) \tag{4.8}$$

Recall that this is exactly the function that we considered in Equation (3.16). Indeed, we found that the minimum for the function occurs at (0.4394, 0.0829), i.e., given the force $(F_x, F_y) = (10, 8)$, the spring system comes to rest when the center is at (0.4394, 0.0829).

4.2 Force Balance

In this Section, we shall exploit force balance techniques to determine the equilibrium of the two-spring system. When the center node is displaced by (u, v), the magnitude of the internal forces in the two springs are $k_1 \Delta l_1$ and $k_2 \Delta l_2$ respectively, where Δl_1 and Δl_2 are given by Equation (4.2). The directions of the two forces are:

$$\begin{aligned} \hat{d}_1 &= \frac{(0-u)\hat{i} + (-1-v)\hat{j}}{\sqrt{(0-u)^2 + (1+v)^2}} = -\frac{u\hat{i} + (1+v)\hat{j}}{L_1} \\ \hat{d}_2 &= \frac{(0-u)\hat{i} + (1-v)\hat{j}}{\sqrt{(0-u)^2 + (1-v)^2}} = -\frac{u\hat{i} + (v-1)\hat{j}}{L_2} \end{aligned} \tag{4.9}$$

Thus the internal forces in the two springs are:

$$\begin{aligned} \vec{F}_1 &= (k_1 \Delta l_1)\hat{d}_1 = -\frac{(k_1 \Delta l_1)}{L_1}\left[u\hat{i} + (1+v)\hat{j}\right] \\ \vec{F}_2 &= (k_2 \Delta l_2)\hat{d}_2 = -\frac{(k_2 \Delta l_2)}{L_2}\left[u\hat{i} + (v-1)\hat{j}\right] \end{aligned} \tag{4.10}$$

The external force is given by:

$$\vec{F}^{ext} = \left[F_x \hat{i} + F_y \hat{j}\right] \tag{4.11}$$

Force balance requires that:

$$\vec{F}_1 + \vec{F}_2 + \vec{F}^{ext} = 0 \tag{4.12}$$

resulting in two equations (corresponding to the x and y component):

$$\begin{aligned} &-\frac{(k_1 \Delta l_1)}{L_1} u - \frac{(k_2 \Delta l_2)}{L_2} u + F_x = 0 \\ &-\frac{(k_1 \Delta l_1)}{L_1}(v+1) - \frac{(k_2 \Delta l_2)}{L_2}(v-1) + F_y = 0 \end{aligned} \tag{4.13}$$

Substituting for Δl_1, Δl_2, etc, and with a change in sign, we have:

$$\begin{aligned} &\frac{k_1\left(\sqrt{u^2 + (v+1)^2} - 1\right)}{\sqrt{u^2 + (v+1)^2}} u + \frac{k_2\left(\sqrt{u^2 + (v-1)^2} - 1\right)}{\sqrt{u^2 + (v-1)^2}} u - F_x = 0 \\ &\frac{k_1\left(\sqrt{u^2 + (v+1)^2} - 1\right)}{\sqrt{u^2 + (v+1)^2}}(v+1) + \frac{k_2\left(\sqrt{u^2 + (v-1)^2} - 1\right)}{\sqrt{u^2 + (v-1)^2}}(v-1) - F_y = 0 \end{aligned} \tag{4.14}$$

Thus we have two non-linear equations governing two unknowns $u \,\&\, v$. These are exactly the same non-linear equations as in Equation (3.19). In other words, the *force-balance equations for any conservative structural system are precisely the gradient of the potential energy function!* We can also conclude that solving the force balance is equivalent to finding the stationary points of a system, and these are not necessarily the minima!

4.3 Potential Energy and Force Balance for Multi-DOF Springs

The conclusions drawn from the previous two sections can be generalized to more complex spring systems. For example, consider the 5-spring system shown below, where the 4 outer nodes are considered fixed, and the two middle nodes are free to move. Suppose an arbitrary force is applied at each of the free nodes (1 refers to the node on the left), the task lies in the finding the equilibrium of the spring system. Observe that there are 4 degrees of freedom: $\{u_1, v_1, u_2, v_2\}$. As before, we can find the equilibrium either via minimization of potential energy or via force balance. The former yield minima, while the latter will yield stationary points. For example, the elastic energy of the system is given by:

$$U = \frac{1}{2}\sum_{e=1}^{4} k_e \left(\Delta l_e\right)^2 \tag{4.15}$$

where the change in length Δl_e for each spring is a function of the unknowns u_n, v_n (and known end points). The work done is:

$$W = \sum_{n=1}^{2} \vec{F}_n \cdot (u_n\hat{i} + v_n\hat{j}) \tag{4.16}$$

Thus the potential energy is given by:

$$\Pi = \frac{1}{2}\sum_{s=1}^{S=5} k_s \left(\Delta l_s\right)^2 - \sum_{k=1}^{K=2} (F_{kx}^{ext} u_k + F_{ky}^{ext} v_k) \tag{4.17}$$

The minimization problem can be posed as:

$$\underset{\{u_1,v_1,u_2,v_2\}}{Min} \quad \Pi(u_1, v_1, u_2, v_2) \tag{4.18}$$

The force balance equation can be derived in two ways: (1) explicitly write out the force balance at each node, or (2) set the derivative Equation (4.17) with respect to the four degrees of freedom to zero. Either way, we end up in four nonlinear equations:

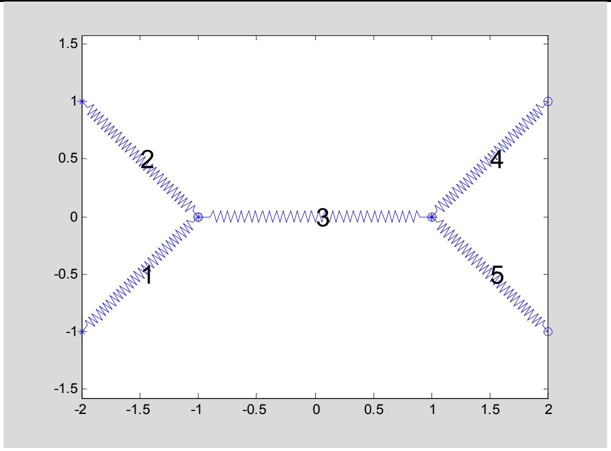


Figure 4-2: A five-spring system.

Clearly, the above process may be generalized to an arbitrary collection of connected springs. We shall describe a typical spring system via the following:

1. A $(2, M)$ 'NodeXY' matrix that specifies the (x, y) location of all M end-points in its un-deformed state. *Henceforth, for convenience, we shall assume that all the free nodes are numbered first, followed by all the fixed nodes.* For the 2-spring system, M = 3, wit 1

free node, and for the 5-spring system, M = 6, with 2 free nodes. Further, for the 2-spring system, the node matrix is given by:

$$NodeXY = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \end{bmatrix}$$

2. A $(2, S)$ 'Connectivity' matrix, where S is the number of springs, specifies how the nodes are connected. *Henceforth, for convenience, we shall also assume that for any spring (connecting two nodes) the first node is always smaller of the two.* For the 2-spring system, S = 2, and for the 5-spring system, S = 5. Further, for the 2-spring system, the connectivity-matrix is given by:

$$Connectivity = \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix}$$

3. A $(1, S)$ 'Stiffness' vector specifies the stiffness of each spring. For the 2-spring system, the stiffness-matrix is given by:
$Stiffness = [100 \quad 500]$
4. A (1, K) 'FreeNodes' vector that specifies which of the nodes are free. Thus all the free nodes are numbered first from 1 through K, followed by the fixed nodes. For the 2-spring system, K = 1, for the 5-spring system K =2. Further, for the 2-spring system, since node-1 is free, we have:
$FreeNodes = [1]$
5. A $(2, K)$ 'Force' matrix specifies the forces at each of the K free nodes. For the 2-spring system, the force-matrix is given by:

$$Force = \begin{bmatrix} 10 \\ 8 \end{bmatrix}$$

Observe that given the above information, the spring system is complete and unambiguous. In Matlab the 2-spring system may therefore be captured via a Spring structure as follows:

```
Spring.NodeXY = [0 0 0; 0 -1 1];
Spring.Connectivity = [1 2; 1 3]';
Spring.Stiffness = [100 500];
Spring.FreeNodes = [1];
Spring.Force = [10 8]';
```

Script 4-1: A 2-spring system.

The five-spring system is given by the Spring structure below (the first two nodes are free):

```
Spring.NodeXY = [-1 1 -2 2 2 -2; 0 0 -1 -1 1 1];
Spring.Connectivity = [1 3; 2 4; 2 5; 1 6; 1 2]';
Spring.Stiffness = [100 100 100 100 200];
Spring.FreeNodes = [1 2];
Spring.Force = [10 8; -5 12]';
```

Script 4-2: A 5-spring system.

Given the structure above, one can now find the potential energy of any spring system. Observe that the total degree of freedom is 2*K of the form:

$$\overline{x} = \left\{ u_1 \quad v_1 \quad \dots \quad \dots \quad u_K \quad v_K \right\}$$

The above notation is consistent with the earlier assumption that all the free nodes are numbered from 1 through K. A potential energy function will take two inputs: a 2*K vector $\bar{x}$ and a Spring structure, and it returns a scalar.

```
function [PE] = PEFunction(xBar,Spring)
% the code for computing potential energy
% given an xBar and the Spring system
```

Script 4-3: Potential energy template without gradient.

Once the above function has been created, then the minimum can be easily determined as follows:

```
[xBar] = fminunc(@(xBar)PEFunction(xBar,Spring))
```

Script 4-4: Minimizing the potential energy

For visualization purposes, one can use the following script to draw springs given an $\bar{x}$ and a Spring structure. You can use this function to check if the data has been specified correctly, and also to check if the displacements are as expected.

```
function drawSpringSystem(xBar,Spring)
% Example
% xBar = [0 0];
% Spring.NodeXY = [0 0 0; 0 -1 1];
% Spring.Connectivity = [1 2; 1 3]';
% Spring.Stiffness = [100 500];
% Spring.FreeNodes = [1];
% Spring.Force = [10 8]';
NodeXY = Spring.NodeXY; % extract undeformed location
FreeNodes = Spring.FreeNodes;
% Add the deformation
u = xBar(1:2:end);v = xBar(2:2:end);
NodeXY(1,FreeNodes) = NodeXY(1,FreeNodes) + u;
NodeXY(2,FreeNodes) = NodeXY(2,FreeNodes) + v;
nSprings  = length(Spring.Stiffness);
for i = 1:nSprings
    startNode = Spring.Connectivity(1,i);
    endNode = Spring.Connectivity(2,i);
    p = NodeXY(:,[startNode endNode]);
    plot(p(1,:),p(2,:),'b'); hold on;
    text(mean(p(1,:)),mean(p(2,:)),num2str(Spring.Stiffness(i)));
end
nNodes = size(NodeXY,2);
nFreeNodes = 0;
for j = 1:nNodes
    if (find(FreeNodes(:) == j))
        nFreeNodes =nFreeNodes+1;
        plot(NodeXY(1,j),NodeXY(2,j),'o');
        F = Spring.Force(:,nFreeNodes);
        str = ['(' num2str(F(1))  ', ' num2str(F(2)) ')' ];
        text(NodeXY(1,j),NodeXY(2,j),str);
    else
        plot(NodeXY(1,j),NodeXY(2,j),'*');
    end
end
```

Script 4-5: A spring drawing routine.

One can also construct a potential energy function that returns a scalar and a gradient as shown below. The gradient is determined by systematically writing out the force balance at each free node. Note that PEGrad is (2*K, 1) vector.

```
function [PE, PEGrad] = PEFunctionWithGrad(xBar,Spring)
% the code for computing potential energy
% given an xBar and the Spring system
```

Script 4-6: Potential energy template with gradient.

4.4 Small Displacement Spring Systems: Trusses

Considerable simplification of the potential energy is possible if the displacements are assumed to be small compared to the lengths of the springs. Such assumptions are typically valid in most engineering systems such as trusses, beam-structures (frames), etc.

Recall that the potential energy of a 2-D spring system is given by:

$$\Pi = \frac{1}{2}\sum_{s=1}^{S} k_s \left(\Delta l_s\right)^2 - \sum_{k=1}^{K}(F_{kx}^{ext}u_k + F_{ky}^{ext}v_k) \tag{4.19}$$

The objective is to replace the above expression with a quadratic potential energy assuming $|u| \ll l$ and $|v| \ll l$. Our strategy is to make a linear approximation of the 'change in lengths' Δl_s with respect to displacements u & v, that rendering Equation (4.19) quadratic.

Consider a spring whose end nodes are 1 and 2, for simplicity. Let the initial location of the two ends of a spring be (x_1, y_1) and (x_2, y_2). The initial length is:

$$l = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{4.20}$$

Suppose the displacement of the two ends are (u_1, v_1) and (u_2, v_2), the final length is:

$$L = \sqrt{\left((x_2 + u_2) - (x_1 + u_1)\right)^2 + \left((y_2 + v_2) - (y_1 + v_1)\right)^2} \tag{4.21}$$

i.e.,

$$L = \sqrt{\left((x_2 - x_1) + (u_2 - u_1)\right)^2 + \left((y_2 - y_1) + (v_2 - v_1)\right)^2} \tag{4.22}$$

We shall assume that the terms $(u_2 - u_1)^2$ and $(v_2 - v_1)^2$ are small compared to $(x_2 - x_1)^2 + (y_2 - y_1)^2$. Thus:

$$L \approx \sqrt{(x_2 - x_1)^2 + 2(u_2 - u_1)(x_2 - x_1) + (y_2 - y_1)^2 + 2(v_2 - v_1)(y_2 - y_1)} \tag{4.23}$$

Exploiting Equation (4.20), we have:

$$L \approx \sqrt{l^2 + 2(u_2 - u_1)(x_2 - x_1) + 2(v_2 - v_1)(y_2 - y_1)} \tag{4.24}$$

i.e.,

$$L \approx l\sqrt{1 + \frac{2(u_2 - u_1)(x_2 - x_1) + 2(v_2 - v_1)(y_2 - y_1)}{l^2}} \tag{4.25}$$

Recall that, if $x \ll 1$ then:

$$\sqrt{1 + x} \approx 1 + x/2 \tag{4.26}$$

Thus:

$$L \approx l\left[1 + \frac{(u_2 - u_1)(x_2 - x_1) + (v_2 - v_1)(y_2 - y_1)}{l^2}\right] \tag{4.27}$$

Therefore the increase in length is:

$$\Delta l = L - l \approx l\left[1 + \frac{(u_2 - u_1)(x_2 - x_1) + (v_2 - v_1)(y_2 - y_1)}{l^2}\right] - l \tag{4.28}$$

i.e.,

$$\Delta l \approx \frac{(u_2 - u_1)(x_2 - x_1) + (v_2 - v_1)(y_2 - y_1)}{l} \tag{4.29}$$

Let:

$$\begin{aligned} \frac{(x_2 - x_1)}{l} &= c \\ \frac{(y_2 - y_1)}{l} &= s \end{aligned} \tag{4.30}$$

Thus:

$$\Delta l \approx (u_2 - u_1)c + (v_2 - v_1)s \tag{4.31}$$

that is indeed linear with respect to the nodal displacements. Thus the elastic energy of the spring is given by:

$$U = \frac{1}{2}k(\Delta l)^2 \approx \frac{1}{2}k\left[(u_2 - u_1)c + (v_2 - v_1)s\right]^2 \tag{4.32}$$

Expanding:

$$U = \frac{1}{2}k\begin{bmatrix} (u_2 - u_1)^2 c^2 \\ +(v_2 - v_1)^2 s^2 \\ +2(u_2 - u_1)(v_2 - v_1)cs \end{bmatrix} \tag{4.33}$$

i.e.,

$$U = \frac{1}{2}k\begin{bmatrix} (u_2^2 - 2u_1u_2 + u_1^2)c^2 \\ +(v_2^2 - 2v_1v_2 + v_1^2)s^2 \\ +2(u_2v_2 + u_1v_1 - u_1v_2 - u_2v_1)cs \end{bmatrix} \tag{4.34}$$

We shall write the above in a matrix form:

$$\begin{gathered} U = \frac{1}{2}\begin{Bmatrix} u_1 & v_1 & u_2 & v_2 \end{Bmatrix} q \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \end{Bmatrix} \text{ where } q = k\begin{bmatrix} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ -c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{bmatrix} \\ \text{and} \\ c = \frac{(x_2 - x_1)}{l}; s = \frac{(y_2 - y_1)}{l}; l_s = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \end{gathered} \tag{4.35}$$

Let us now generalize the notation as follows. Let the spring be s, joining two free nodes, node-m to node-n. Thus Equation (4.35) can be generalized as follows:

$$U_s = \frac{1}{2}\begin{Bmatrix} u_m & v_m & u_n & v_n \end{Bmatrix} q_s \begin{Bmatrix} u_m \\ v_m \\ u_n \\ v_n \end{Bmatrix}; q_s = k_s \begin{bmatrix} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ -c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{bmatrix};$$
$$c = \frac{(x_n - x_m)}{l_s}; s = \frac{(y_n - y_m)}{l_s}; l_s = \sqrt{(x_n - x_m)^2 + (y_n - y_m)^2} \tag{4.36}$$

Equation (4.36) captures the quadratic potential energy of a small-displacement spring with two free ends. If the spring has only one free end, say m, joining a fixed node-n then it is easy to show that the potential energy can be approximated via:

$$U_s = \frac{1}{2}\begin{Bmatrix} u_m & v_m \end{Bmatrix} q_s \begin{Bmatrix} u_m \\ v_m \end{Bmatrix}; q_s = k_s \begin{bmatrix} c^2 & cs \\ cs & s^2 \end{bmatrix}$$
$$c = \frac{(x_n - x_m)}{l_s}; s = \frac{(y_n - y_m)}{l_s}; l_s = \sqrt{(x_n - x_m)^2 + (y_n - y_m)^2} \tag{4.37}$$

As an illustrative example, consider the 5-spring system below, whose Spring structure was described earlier.

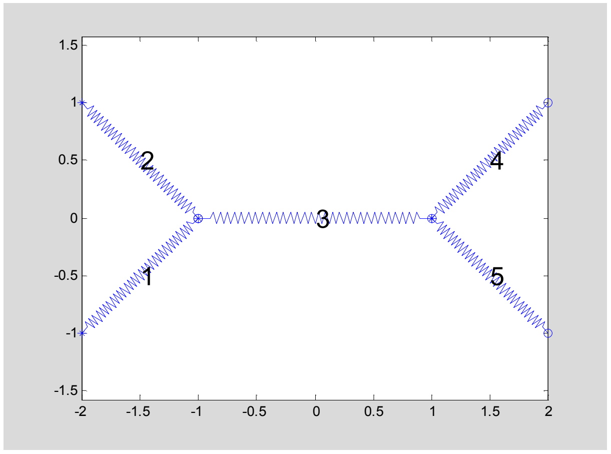


Figure 4-3: A five-spring system.

For Spring-1 that joins node-1 and node-3:

$$l_1 = \sqrt{2}$$
$$c = \frac{(x_3 - x_1)}{l_1} = -1/\sqrt{2} \tag{4.38}$$
$$s = \frac{(y_3 - y_1)}{l_1} = -1/\sqrt{2}$$

Thus the elastic energy of spring-1 is given by:

$$U_1 = \frac{1}{2}\begin{Bmatrix} u_1 & v_1 \end{Bmatrix} q_1 \begin{Bmatrix} u_1 \\ v_1 \end{Bmatrix} \text{ where } q_1 = k_1 \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix} \tag{4.39}$$

Similarly, for spring-2

$$U_2 = \frac{1}{2}\begin{Bmatrix} u_2 & v_2 \end{Bmatrix} q_2 \begin{Bmatrix} u_2 \\ v_2 \end{Bmatrix} \text{ where } q_2 = k_2 \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{bmatrix} \tag{4.40}$$

For spring-3:

$$U_3 = \frac{1}{2}\begin{Bmatrix} u_2 & v_2 \end{Bmatrix} q_3 \begin{Bmatrix} u_2 \\ v_2 \end{Bmatrix} \text{ where } q_3 = k_3 \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix} \tag{4.41}$$

For spring-4:

$$U_4 = \frac{1}{2}\begin{Bmatrix} u_1 & v_1 \end{Bmatrix} q_4 \begin{Bmatrix} u_1 \\ v_1 \end{Bmatrix} \text{ where } q_4 = k_4 \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{bmatrix} \tag{4.42}$$

Finally, for spring-5:

$$U_5 = \frac{1}{2}\begin{Bmatrix} u_1 & v_1 & u_2 & v_2 \end{Bmatrix} q_5 \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \end{Bmatrix} \text{where } q_5 = k_5 \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.43}$$

To find the potential energy of the entire spring system, we simply add all the terms from Equation (4.39) through Equation (4.43):

$$U = \sum_{s=1}^{5} U_s \tag{4.44}$$

i.e.,

$$U = \begin{bmatrix} \frac{1}{2}\begin{Bmatrix} u_1 & v_1 \end{Bmatrix} q_1 \begin{Bmatrix} u_1 \\ v_1 \end{Bmatrix} + \frac{1}{2}\begin{Bmatrix} u_2 & v_2 \end{Bmatrix} q_2 \begin{Bmatrix} u_2 \\ v_2 \end{Bmatrix} + \frac{1}{2}\begin{Bmatrix} u_2 & v_2 \end{Bmatrix} q_3 \begin{Bmatrix} u_2 \\ v_2 \end{Bmatrix} \\ + \frac{1}{2}\begin{Bmatrix} u_1 & v_1 \end{Bmatrix} q_4 \begin{Bmatrix} u_1 \\ v_1 \end{Bmatrix} + \frac{1}{2}\begin{Bmatrix} u_1 & v_1 & u_2 & v_2 \end{Bmatrix} q_5 \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \end{Bmatrix} \end{bmatrix} \tag{4.45}$$

Note that Equation (4.45) is quadratic with respect to each of the degrees of freedom in $\begin{Bmatrix} u_1 & v_1 & u_2 & v_2 \end{Bmatrix}$. Thus the strategy is to express the above equation as:

$$U = \frac{1}{2}\begin{Bmatrix} u_1 & v_1 & u_2 & v_2 \end{Bmatrix} Q \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \end{Bmatrix} \tag{4.46}$$

It can be easily observed that the matrix Q can be assembled as follows:

$$Q = \begin{bmatrix} [q_1 + q_4 + q_5(1:2,1:2)] & [q_5(1:2,3:4)] \\ [q_5(3:4,1:2)] & [q_2 + q_3 + q_5(3:4,3:4)] \end{bmatrix} \tag{4.47}$$

For a more complex spring consisting of S springs, the total elastic energy can be expressed as:

$$U = \sum_{s=1}^{S} U_s \tag{4.48}$$

When expressed as a single quadratic expression, we have:

$$U = \frac{1}{2}\left\{u_1 \quad v_1 \quad \dots \quad \dots \quad u_K \quad v_K\right\} Q \begin{Bmatrix} u_1 \\ v_1 \\ \dots \\ \dots \\ u_K \\ v_K \end{Bmatrix} \tag{4.49}$$

The algorithm to construct the Q matrix for an arbitrary spring system is as follows:

1. Initialize the Q matrix to a zero matrix of size (2*K, 2*K)
2. For each spring
   a. Suppose the spring connects a *free* node-m and a fixed node-n, then, construct a 2x2 q matrix via Equation (4.37), and update the Q matrix as follows:

$$\begin{aligned} rows &= \begin{bmatrix} 2m-1 & 2m \end{bmatrix} \\ cols &= \begin{bmatrix} 2m-1 & 2m \end{bmatrix} \\ Q(rows, cols) &= Q(rows, cols) + q \end{aligned} \tag{4.50}$$

   b. Suppose the spring connects a *free* node-m and a *free* node-n, then, construct the 4x4 q matrix via Equation(4.36), and update the Q matrix as follows:

$$\begin{aligned} rows &= \begin{bmatrix} 2m-1 & 2m & 2n-1 & 2n \end{bmatrix} \\ cols &= \begin{bmatrix} 2m-1 & 2m & 2n-1 & 2n \end{bmatrix} \\ Q(rows, cols) &= Q(rows, cols) + q \end{aligned} \tag{4.51}$$

We now turn our attention to the external force vector in Equation (4.19). Observe that the potential energy can now be approximated as:

$$\Pi = \frac{1}{2}\overline{x}^T Q \overline{x} + \overline{x}^T b \tag{4.52}$$

where the b vector can be assembled as follows:

1. Initialize the b vector to a zero vector of size (2*K, 1)
2. For each free-node k (1 through K) update the b vector as follows:

$$rows = \begin{bmatrix} 2k-1 & 2k \end{bmatrix}$$
$$b(rows,1) = b(rows,1) + \begin{Bmatrix} -F_{kx}^{ext} \\ -F_{ky}^{ext} \end{Bmatrix} \tag{4.53}$$

Then, the minimum can be determined via:

$$\overline{x}^* = -Q \setminus b \tag{4.54}$$

4.5 Design Optimization of Spring Systems

We now consider the design optimization of large and small displacement spring systems. Here the task is find one or more parameters of a spring system such that certain design objectives are met. For example, consider again the 2-spring system discussed earlier. We shall now allow the top node to move on a circle of radius 1 centered at the origin. In other words, the nodes of the spring system are given by:

$$NodeXY = \begin{bmatrix} 0 & 0 & \sin\theta \\ 0 & -1 & \cos\theta \end{bmatrix}$$

where θ the angle between spring-2 and the y-axis; it is a free variable that we need to optimize. The spring constants are $k_1 = 100N/m$ and $k_2 = 500N/m$, as before and a force $(F_x, F_y) = (10,8)$ is applied at the center node. The problem is to find θ such that the equilibrium position is as close to the point $(0.5, 0.1)$ as possible.

For a given value of θ, observe that the potential energy function in Equation (4.8) may be generalized to:

$$\Pi(u,v;\theta) = \begin{bmatrix} \frac{1}{2}100\left(\sqrt{u^2+(v+1)^2}-1\right)^2 + \\ \frac{1}{2}500\left(\sqrt{(u-\sin\theta)^2+(v-\cos\theta)^2}-1\right)^2 \\ -(10u+8v) \\ \end{bmatrix} \tag{4.55}$$

The potential energy function can be captured below; observe that it takes (u, v) and θ as input.

```
function [f] = fEquilibrium (xBar, theta)
u = xBar(1); v = xBar(2);
f=0.5*100*(sqrt(u*u+(v+1)^2)-1)^2+0.5*500*(sqrt((u-sin(theta))^2+(v-
cos(theta))^2)-1)^2-(10*u+8*v);
```

Script 4-7: Matlab function fEquilibrium.

The script below shows how one may obtain the equilibrium position for $\theta = 0$ and $\theta = 10^0$. i.e., 0.1745 radians.
input.

```
[xBar] = fminunc(@(xBar)fEquilibrium(xBar,0),[0;0])
[xBar] = fminunc(@(xBar)fEquilibrium(xBar,0.1745),[0;0])
```

Script 4-8: Matlab script minimizing fEquilibrium.

If $\theta = 0$ we recover the original potential energy whose equilibrium position is (0.4394,0.0829), and for $\theta = 10^0$. i.e., 0.1745 radians, the equilibrium occurs at (0.5316,0.0359).

The challenge is find θ such that the equilibrium is as close to $(0.5, 0.1)$ as possible. We shall pose this as a standard optimization problem where the objective is to find θ such that the distance (or equivalently the distance-squared) from (u, v) to $(0.5, 0.1)$ is minimized. Thus, we have dual-optimization problem:

$$\begin{aligned} &\underset{\theta}{Min}\,(u-0.5)^2 + (v-0.1)^2 \\ &\quad \underset{(u,v)}{Min}\,\Pi(u,v;\theta) \end{aligned} \tag{4.56}$$

where in the outer loop, the design parameter θ is optimized, while in the inner loop, given a θ, the potential energy is optimized. This dual optimization is the central theme in engineering. To solve Equation (4.56), i.e., to find θ and to find the resulting equilibrium position, via Matlab, we create a design objective function below.

```
function [dSqr] = fDesignObjective (theta)
% For a given theta, solve the potential energy minimization
xBar0 = [0;0];
[xBar] = fminunc(@(xBar)fEquilibrium(xBar,theta),xBar0);
% Now find the distance squared from desired position
u = xBar(1);
v = xBar(2);
dSqr = (u-0.5)^2 + (v-0.1)^2;
```

Script 4-9: Matlab function fDesignObjective.

Finally, we minimize the design objective as follows.

```
theta0 = 0; % guess
[theta] = fminunc(@(theta)fDesignObjective(theta),theta0);
```

Script 4-10: Matlab script for minimizing design objective.

The optimal θ turns out to be 0.0778 radians, i.e., 4.46 degrees, and the equilibrium occurs at (0.4798, 0.0641). From the above example, it is easy to see why design optimization is computationally challenging.

5 Equality Constraints

Equation Section (Next) Highlights

- Equality constrained problems naturally arise in engineering when two or more degrees of freedom are constrained via a set of equations. Such problems can be posed as:
$$\underset{\bar{x}}{Min}\ f(\bar{x})$$
$$h_i(\bar{x}) = 0; i = 1,2,\ldots,M$$
- The first order condition for the stationary points $\bar{x}^*$ involves as many *Lagrangian multipliers* μ_i as the number of equality constraints, and is given by the set of (M+N) Karush-Kuhn-Tucker (KKT) equations:
$$\nabla f(\bar{x}^*) + \mu_1^* \nabla h_1(\bar{x}^*) + \mu_2^* \nabla h_2(\bar{x}^*) + \ldots + \mu_M^* \nabla h_M(\bar{x}^*) = 0$$
$$h_i(\bar{x}^*) = 0$$
- The 2nd order sufficient but not necessary condition for a stationary point to be a minimum:

  the matrix $\nabla^2 f(\bar{x}^*) + \sum_{i=1}^{M} \mu_i^* \nabla^2 h_i(\bar{x}^*)$ be positive definite.
- A Lagrange multiplier may be interpreted in two ways: (1) It captures the sensitivity of the objective function to the associated constraint, and (2) In a structural problem, it represents the reaction force at a kinematic constraint.
- The first and second order conditions may be simplified if one defines a Lagrangian as:
$$L(\bar{x},\bar{\mu}) = f(\bar{x}) + \mu_1 h_1(\bar{x}) + \mu_2 h_2(\bar{x}) + \ldots + \mu_M h_M(\bar{x})$$
  then the first order condition is: $\nabla L(\bar{x},\bar{\mu}) = 0$

  the second order (sufficient) condition is: $\nabla_{\bar{x}}^2 L\big|_{(\bar{x}^*,\bar{\mu}^* 0}$ be positive definite.
- A Quadratic problem is of the form:
$$\underset{\bar{x}}{Min}\ \frac{1}{2}\bar{x}^T Q \bar{x} + \bar{x}^T b + c$$
$$A\bar{x} + d = 0$$
- The stationary point of a Quadratic problem satisfies the KKT condition:
$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{Bmatrix} \bar{x} \\ \bar{\mu} \end{Bmatrix} = \begin{Bmatrix} -b \\ -d \end{Bmatrix}$$
  The stationary point is a minimum if Q is positive definite (a sufficient condition).
- Typical algorithms for solving constraint minimization problems include: (a) Sequential Quadratic Programming (akin to Newton-Raphson) and (b) Penalty based methods where the constrained problems are transformed into unconstrained problems.

In this chapter, we consider *equality constrained* multivariable minimization problems of the form:

$$\begin{aligned}&\underset{\overline{x}}{Min}\ \ f(\overline{x})\\&h_i(\overline{x})=0; i=1,2,...,M\end{aligned}\tag{5.1}$$

5.1 Constraint Elimination

The most 'obvious' method for solving equality constrained problem is to eliminate the constraints, i.e., reduce the problem to an unconstrained minimization problem. For example, consider the problem:

$$\begin{aligned}&\underset{x,y}{Min}\ \ x^2+y^2\\&x+y-1=0\end{aligned}\tag{5.2}$$

The above problem can be interpreted geometrically as "Find the point on the straight line $x+y-1=0$ that is closest to the origin." Thus, geometrically, it is easy to see that the solution is $x=y=1/2$.

Keeping this geometric solution aside, let us reduce the problem to an unconstrained problem by substituting $y=1-x$ in the objective, leading to:

$$\underset{x}{Min}\ \ x^2+(1-x)^2\tag{5.3}$$

i.e.,

$$\underset{x}{Min}\ \ 2x^2-2x+1\tag{5.4}$$

whose minimum occurs at $x=1/2$ (and $y=1/2$).

For nonlinear constraints, elimination can unfortunately lead to erroneous results. For example, consider:

$$\begin{aligned}&\underset{x,y}{Min}\ \ x^2+y^2\\&(x-2)^2+y^2-1=0\end{aligned}\tag{5.5}$$

The above problem can be interpreted geometrically as "Find the point on the circle of radius 1, centered at (2, 0), that is closest to the origin." Clearly, the closest point is (1, 0).

Once again, the problem can be reduced to an unconstrained problem by substituting $y^2=1-(x-2)^2$ in the objective, leading to:

$$\underset{x}{Min}\ \ x^2+1-(x-2)^2\tag{5.6}$$

i.e.,

$$\underset{x}{Min}\ \ 4x-3\tag{5.7}$$

Note that the problem is unbounded as $x\to-\infty$, implying that the problem in Equation (5.5) has no minimum. So one can ponder on what went wrong in the elimination method!

Further, for nonlinear constraints, elimination may not always possible; for example, consider:

$$\begin{aligned}&\underset{x,y}{Min}\ \ x^2+y^2\\&x+e^x\sin y-x^2\cos xy=0\end{aligned}\tag{5.8}$$

Finally, retaining constraints often reveals certain important characteristics of the problem, as discussed in the next few Sections.

5.2 Constrained Spring System

We shall now develop methods for finding the minimum of equality constrained system, *without* eliminating the constraints. We shall illustrate the underlying theory via the now familiar two spring system except that the free-node will now be constrained to follow a 'frictionless' track $v=u^3$ illustrated below via a dotted line, i.e., the displacements are constrained via $v=u^3$.

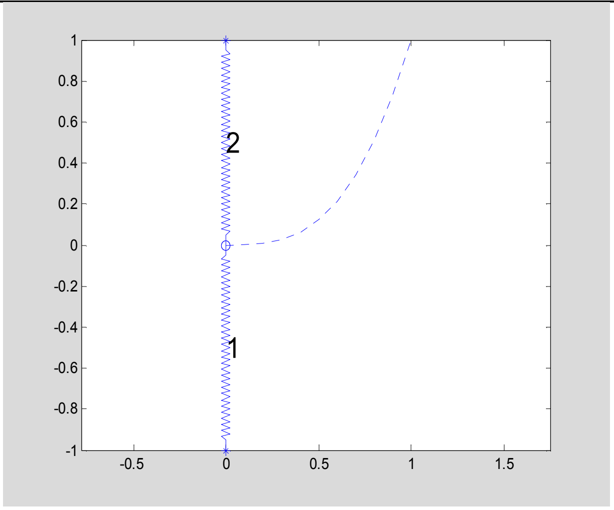


Figure 5-1: An equality-constrained spring system.

Mathematically, the problem can be posed as follows:

$$\begin{aligned}&\underset{u,v}{Min}\ \ \Pi=\frac{1}{2}100\left(\sqrt{u^2+(v+1)^2}-1\right)^2+\frac{1}{2}500\left(\sqrt{u^2+(v-1)^2}-1\right)^2-(10u+8v)\\&h=v-u^3=0\end{aligned}\tag{5.9}$$

First observe that at the origin $u=0,v=0$, i.e., the constraint is satisfied, and the potential energy is $\Pi=0$. Further, it is easy to verify that at a nearby point on the curve, say $u=0.1,v=0.001$, the potential energy is $\Pi=-1.06$. The question is: How far along the track can we move, while continuing to reduce the potential energy?

Let us say we are at an arbitrary point u,v on the curve such that $h(u,v)=0$. Consider the gradient $\nabla\Pi$ of the potential energy. Clearly, if the gradient decreases along the curve at that point, then we can continue further in the direct of descent. In other words, suppose t is the tangent to the curve at u,v, we can continue further if $\nabla\Pi\cdot t<0$. We stop when there is no further increase along the curve, i.e., when:

$$\nabla\Pi\cdot t=0\tag{5.10}$$

By definition, points that satisfy Equation (5.10) are stationary points of Equation (5.9)! Observe that Equation (5.10) is a generalization of the usual statement $\nabla\Pi=0$ for unconstrained problems. Equation (5.10) can also be interpreted as follows: at a stationary point, the gradient of the objective ($\nabla\Pi$) is perpendicular to the tangent of the constraint curve. But the tangent is always perpendicular to the gradient of the curve ∇h. Thus, we conclude that at a stationary point, the gradient of the objective ($\nabla\Pi$) must have the same orientation as the gradient of the constraint (∇h). Mathematically,

$$\frac{\nabla\Pi}{\|\nabla\Pi\|} = \pm\frac{\nabla h}{\|\nabla h\|} \tag{5.11}$$

i.e.,

$$\nabla\Pi \pm \frac{\|\nabla\Pi\|}{\|\nabla h\|}\nabla h = 0 \tag{5.12}$$

In other words, the gradient of the objective and the gradient of the constraint are scalar multiples at a stationary point, i.e., there exists a scalar μ (positive or negative) such that:

$$\nabla\Pi + \mu\nabla h = 0 \tag{5.13}$$

Thus, the objective is to find the stationary point (u,v,μ) such that:

$$\begin{aligned} &\nabla\Pi(u,v) + \mu\nabla h(u,v) = 0 \\ &h(u,v) = 0 \end{aligned} \tag{5.14}$$

5.3 First and Second Order Conditions

Having motivated the first order constraint equation via the two-spring problem, we now generalize the above result via a formal theorem involving N variables and a single equality constraint, and state a sufficient condition for a minimum.

Theorem 5-1: Consider the equality constrained optimization problem:

$$\begin{aligned} &\underset{\overline{x}}{Min}\;\; f(\overline{x}) \\ &h(\overline{x}) = 0 \end{aligned} \tag{5.15}$$

A stationary point $\overline{x}^*$ satisfies the set of equations:

$$\begin{aligned} &\nabla f(\overline{x}^*) + \mu^*\nabla h(\overline{x}^*) = 0 \\ &h(\overline{x}^*) = 0 \end{aligned} \tag{5.16}$$

for some scalar μ^*. Further, the stationary point is a minimum if the matrix:

$$\nabla^2 f(\overline{x}^*) + \mu^*\nabla^2 h(\overline{x}^*) \tag{5.17}$$

is positive definite (a sufficient, but not a necessary condition).

Proof (Sketch): We provide here a sketch of the proof; see [2] for further details.

First Order Condition

Consider the first two terms of the Taylor series approximation of $f(\overline{x})$ about a point $\overline{x}^*$:

$$f(\overline{x}^* + \Delta\overline{x}) \approx f(\overline{x}^*) + \Delta\overline{x}^T\nabla f(\overline{x}^*) \tag{5.18}$$

However, observe that the increment $\Delta\overline{x}$ cannot be arbitrary; it must satisfy the constraint. In other words, the increment must be such that:

$$\begin{aligned}&h(\overline{x}^* + \Delta\overline{x}) \approx 0\\ &i.e.\\ &h(\overline{x}^*) + \Delta\overline{x}^T \nabla h(\overline{x}^*) \approx 0\end{aligned} \tag{5.19}$$

But, $h(\overline{x}^*) = 0$, thus the increment $\Delta\overline{x}$ must be such that:

$$\Delta\overline{x}^T \nabla h(\overline{x}^*) = 0 \tag{5.20}$$

Now, we define a stationary point as the point where the linear term of Equation (5.18) vanishes, i.e.,

$$\begin{aligned}&\Delta\overline{x}^T \nabla f(\overline{x}^*) = 0\\ &s.t.\\ &\Delta\overline{x}^T \nabla h(\overline{x}^*) = 0\end{aligned} \tag{5.21}$$

This is equivalent to:

$$\nabla f(\overline{x}^*) + \mu^* \nabla h(\overline{x}^*) = 0 \tag{5.22}$$

for some real value (positive or negative) μ^*. In addition the stationary point must satisfy the constraint:

$$h(\overline{x}^*) = 0 \tag{5.23}$$

<u>Second Order Condition</u>

Consider the first three terms of the Taylor series approximation of $f(\overline{x})$ and $h(\overline{x})$ about a stationary point $\overline{x}^*$

$$f(\overline{x}^* + \Delta\overline{x}) \approx f(\overline{x}^*) + \Delta\overline{x}^T \nabla f(\overline{x}^*) + \frac{1}{2}\Delta\overline{x}^T \nabla^2 f(\overline{x}^*)\Delta\overline{x} \tag{5.24}$$

where $\Delta\overline{x}$ is such that

$$h(\overline{x}^* + \Delta\overline{x}) = h(\overline{x}^*) + \Delta\overline{x}^T \nabla h(\overline{x}^*) + \frac{1}{2}\Delta\overline{x}^T \nabla^2 h(\overline{x}^*)\Delta\overline{x} = 0 \tag{5.25}$$

Since $h(\overline{x}^*) = 0$

$$\Delta\overline{x}^T \nabla h(\overline{x}^*) + \frac{1}{2}\Delta\overline{x}^T \nabla^2 h(\overline{x}^*)\Delta\overline{x} = 0 \tag{5.26}$$

Now, multiply Equation (5.26) by μ^* and add to Equation (5.24):

$$f(\overline{x}^* + \Delta\overline{x}) \approx \left\{\begin{matrix} f(\overline{x}^*) + \\ \Delta\overline{x}^T\left[\nabla f(\overline{x}^*) + \mu^*\nabla h(\overline{x}^*)\right] + \\ \frac{1}{2}\Delta\overline{x}^T\left[\nabla^2 f(\overline{x}^*) + \mu^*\nabla^2 h(\overline{x}^*)\right]\Delta\overline{x}\end{matrix}\right\} \tag{5.27}$$

From the definition of the stationary point, the 2nd term vanishes:

$$f(\overline{x}^* + \Delta\overline{x}) = f(\overline{x}^*) + \frac{1}{2}\Delta\overline{x}^T\left[\nabla^2 f(\overline{x}^*) + \mu^*\nabla^2 h(\overline{x}^*)\right]\Delta\overline{x} \tag{5.28}$$

For $\overline{x}^*$ to be a minimum, the quadratic term above must be positive for all $\Delta\overline{x}$ that are tangent to $h(\overline{x}^*)$. A sufficient condition is that the matrix $\nabla^2 f(\overline{x}^*) + \mu^* \nabla^2 h(\overline{x}^*)$ be positive definite (but not necessary).

•

Equation (5.16) is referred to as the *Karush-Kuhn-Tucker* (KKT) condition, and the scalar μ is referred to as the *Lagrange multiplier*.

Example: Consider:

$$\begin{aligned} &\underset{x,y}{Min}\ x^2 + y^2 \\ &x + y - 1 = 0 \end{aligned} \tag{5.29}$$

We have:

$$\nabla f = \begin{Bmatrix} 2x \\ 2y \end{Bmatrix}; \nabla h = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} \tag{5.30}$$

Thus, we must find x, y & μ such that:

$$\begin{aligned} &\begin{Bmatrix} 2x \\ 2y \end{Bmatrix} + \mu \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} = 0 \\ &x + y - 1 = 0 \end{aligned} \tag{5.31}$$

i.e.,

$$\begin{bmatrix} 2 & 0 & 1 \\ 0 & 2 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{Bmatrix} x \\ y \\ \mu \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix} \tag{5.32}$$

whose solution is given by $x^* = 1/2, y^* = 1/2$ & $\mu^* = -1$. Observe that this is precisely the solution we obtained via the elimination method. Now consider the matrix:

$$\nabla^2 f(\overline{x}^*) + \mu^* \nabla^2 h(\overline{x}^*) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} - 1 \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \tag{5.33}$$

Since the matrix is positive definite, the stationary point is a minimum. The objective at the minimum is ½.

♦

Example: As a second example, consider:

$$\begin{aligned} &\underset{x,y}{Min}\ x^2 + y^2 \\ &(x-2)^2 + y^2 - 1 = 0 \end{aligned} \tag{5.34}$$

We have:

$$\nabla f = \begin{Bmatrix} 2x \\ 2y \end{Bmatrix}$$
$$\nabla h = \begin{Bmatrix} 2x-4 \\ 2y \end{Bmatrix} \tag{5.35}$$

Thus, we must find x, y & μ such that:

$$\begin{Bmatrix} 2x \\ 2y \end{Bmatrix} + \mu \begin{Bmatrix} 2x-4 \\ 2y \end{Bmatrix} = 0$$
$$x^2 - 2x + y^2 = 0 \tag{5.36}$$

i.e.,

$$\begin{aligned} &2x + 2\mu x - 4\mu = 0 \\ &2y + 2\mu y = 0 \\ &x^2 - 2x + y^2 = 0 \end{aligned} \tag{5.37}$$

Unlike in the previous example, we arrive at a set of non-linear equations. We can use Matlab's fsolve to arrive at two sets of solutions: (a) $x^* = 1, y^* = 0, \mu^* = 1$ and (b) $x^* = 3, y^* = 0, \mu^* = -1.5$. Geometrically we can interpret the first solution as a local minimum, and the second solution as a local maximum.

Now consider the matrix:

$$\nabla^2 f(\overline{x}^*) + \mu^* \nabla^2 h(\overline{x}^*) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} + \mu^* \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \tag{5.38}$$

At $x^* = 1, y^* = 0, \mu^* = 1$, we have:

$$\nabla^2 f(\overline{x}^*) + \mu^* \nabla^2 h(\overline{x}^*) = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$$

which is positive definite, i.e., the point is a minimum. On the other hand, at $x^* = 3, y^* = 0, \mu^* = -1.5$,

$$\nabla^2 f(\overline{x}^*) + \mu^* \nabla^2 h(\overline{x}^*) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

which is negative definite, i.e., the point is a maximum.

♦

Example: Returning back to the two-spring problem, in Equation (5.9), recall that:

$$\nabla \Pi = \begin{Bmatrix} \dfrac{100\left(\sqrt{u^2 + (v+1)^2} - 1\right)}{\sqrt{u^2 + (v+1)^2}} u + \dfrac{500\left(\sqrt{u^2 + (v-1)^2} - 1\right)}{\sqrt{u^2 + (v-1)^2}} u - (10) \\ \dfrac{100\left(\sqrt{u^2 + (v+1)^2} - 1\right)}{\sqrt{u^2 + (v+1)^2}} (v+1) + \dfrac{500\left(\sqrt{u^2 + (v-1)^2} - 1\right)}{\sqrt{u^2 + (v-1)^2}} (v-1) - (8) \end{Bmatrix} \tag{5.39}$$

Further since $h = v - u^3$

$$\nabla h(u,v) = \begin{Bmatrix} -3u^2 \\ 1 \end{Bmatrix} \tag{5.40}$$

Thus, the three non-linear equations to solve are:

$$\begin{Bmatrix} \dfrac{100\left(\sqrt{u^2+(v+1)^2}-1\right)}{\sqrt{u^2+(v+1)^2}}u + \dfrac{500\left(\sqrt{u^2+(v-1)^2}-1\right)}{\sqrt{u^2+(v-1)^2}}u - 10 \\ \dfrac{100\left(\sqrt{u^2+(v+1)^2}-1\right)}{\sqrt{u^2+(v+1)^2}}(v+1) + \dfrac{500\left(\sqrt{u^2+(v-1)^2}-1\right)}{\sqrt{u^2+(v-1)^2}}(v-1) - 8 \end{Bmatrix} + \mu \begin{Bmatrix} -3u^2 \\ 1 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{5.41}$$

$$v - u^3 = 0$$

Exploiting Matlab's fsolve, we find that the stationary point occurs at $u^* = 0.4372, v^* = 0.0835$ with $\mu^* = -0.6766$. (Recall that the unconstrained minimum occurs at $u^* = 0.4394, v^* = 0.0829$). In practice, one would rely on numerical methods such as sequential quadratic programming (SQP) to find the constrained minimum. We shall study such algorithms later on. Equation **Error! Reference source not found.** must be viewed as the condition that must be satisfied at a constrained minimum, and not necessarily as an algorithm since generating and solving non-linear equations is a non-trivial task. We will leave it as an exercise to prove that the point $u^* = 0.4372, v^* = 0.0835$ is indeed a local minimum.

♦

We generalize Theorem 5-1 to multiple constraints as follows.

Theorem 5-2: Consider the equality constrained optimization problem (where the constraints are assumed to be *linearly independent*):

$$\begin{aligned} &\underset{\bar{x}}{Min}\ f(\bar{x}) \\ &h_1(\bar{x}^*) = 0 \\ &h_2(\bar{x}^*) = 0 \\ &\ldots \\ &h_M(\bar{x}^*) = 0 \end{aligned} \tag{5.42}$$

A stationary point $\bar{x}^*$ of above satisfies the set of equations:

$$\begin{aligned} &\nabla f(\bar{x}^*) + \mu_1^* \nabla h_1(\bar{x}^*) + \mu_2^* \nabla h_2(\bar{x}^*) + \ldots + \mu_M^* \nabla h_M(\bar{x}^*) = 0 \\ &h_1(\bar{x}^*) = 0 \\ &h_2(\bar{x}^*) = 0 \\ &\ldots \\ &h_M(\bar{x}^*) = 0 \end{aligned} \tag{5.43}$$

for some set of scalars $\left\{\mu_1^* \quad \mu_2^* \quad \ldots \quad \mu_M^*\right\}$. Further, the stationary point is a minimum if the matrix

$$\nabla^2 f(\overline{x}^*) + \sum_{i=1}^{M} \mu_i^* \nabla^2 h_i(\overline{x}^*) \tag{5.44}$$

is positive definite (a sufficient, but not a necessary condition).

Proof: See [2].

•

Example: Consider the problem:

$$\begin{aligned} &\underset{x,y}{Min} \ \ x^2 + y^2 + z^2 \\ &x + y + z - 1 = 0 \\ &x + 2y + 3z - 4 = 0 \end{aligned} \tag{5.45}$$

We have:

$$\nabla f = \begin{Bmatrix} 2x \\ 2y \\ 2z \end{Bmatrix}; \nabla h_1 = \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix}; \nabla h_2 = \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix} \tag{5.46}$$

Thus, the first order conditions are:

$$\begin{aligned} &\begin{Bmatrix} 2x \\ 2y \\ 2z \end{Bmatrix} + \mu_1 \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} + \mu_2 \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \\ &x + y + z - 1 = 0 \\ &x + 2y + 3z - 4 = 0 \end{aligned} \tag{5.47}$$

We can write this in a matrix form as:

$$\begin{bmatrix} 2 & 0 & 0 & 1 & 1 \\ 0 & 2 & 0 & 1 & 2 \\ 0 & 0 & 2 & 1 & 3 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \\ \mu_1 \\ \mu_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 4 \end{Bmatrix} \tag{5.48}$$

The solution is $x = 1/6; y = 1/3; z = 4/3$, and $\mu_1 = 10/3; \mu_2 = -2$. It is easy to show that the stationary point is a minimum.

♦

5.4 Two Interpretations of Lagrangian Multipliers

Here we consider two interpretations of the Lagrangian multipliers μ_i.

5.4.1 Sensitivity of Constraints

The first interpretation may be stated as follows. Consider the problem:

$$\begin{aligned} &\underset{\overline{x}}{Min}\ \ f(\overline{x}) \\ &h(\overline{x}) = 0 \end{aligned} \tag{5.49}$$

Let the stationary point be $\overline{x}^*$, i.e.,

$$\begin{aligned} &\nabla f(\overline{x}^*) + \mu^* \nabla h(\overline{x}^*) = 0 \\ &h(\overline{x}^*) = 0 \end{aligned} \tag{5.50}$$

for some scalar μ^*. Suppose the constraint is perturbed by a small amount ε as follows:

$$\begin{aligned} &\underset{\overline{x}}{Min}\ \ f(\overline{x}) \\ &h(\overline{x}) + \varepsilon = 0 \end{aligned} \tag{5.51}$$

then one can show that the change in objective is $\mu^*\varepsilon$, i.e.,

$$f_\varepsilon^{\min} = f^{\min} + \mu^*\varepsilon \tag{5.52}$$

Thus, if a Lagrange multiplier associated with a particular constraint is very large, then perturbing that constraint will lead to a large change in the objective.

Example: Recall the problem:

$$\begin{aligned} &\underset{x,y}{Min}\ \ x^2 + y^2 \\ &x + y - 1 = 0 \end{aligned} \tag{5.53}$$

whose solution is given by $x^* = 1/2, y^* = 1/2$ & $\mu^* = -1$, and minimum was ½. Now consider a perturbed problem:

$$\begin{aligned} &\underset{x,y}{Min}\ \ x^2 + y^2 \\ &(x + y - 1) + 0.01 = 0 \end{aligned} \tag{5.54}$$

As before the stationary point satisfies:

$$\begin{bmatrix} 2 & 0 & 1 \\ 0 & 2 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{Bmatrix} x \\ y \\ \mu \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0.99 \end{Bmatrix} \tag{5.55}$$

whose solution is given by $x^* = y^* = 0.495$, and the new objective is 0.49, which is exactly equal to 0.5 + (-1)*0.01.

♦

Equation (5.52) is particular useful in many engineering applications. For example, suppose we are solving an optimization problem, where the objective is to minimize the mass of a structure subject to stress constraints:

$$\begin{aligned} &\underset{\overline{x}}{Min}\ \ m \\ &\sigma - \sigma_{\max} = 0 \\ &\ldots \end{aligned} \tag{5.56}$$

Suppose the optimal mass has been determined. Now, the designer may be interested in finding out how the optimal mass will change if a different material (with a different $\sigma_{\max}$) is chosen. Equation (5.52) can be used to answer this question without resolving the problem.

5.4.2 Reaction Forces in a Structural System

A second interpretation of the Lagrangian multiplier is as follows. Consider again the constrained two-spring problem in Figure 5-1, but now we shall solve the problem via force balance. Let the constrained equilibrium occur at some point (u,v) such that $v = u^3$. As before, there are the external forces and internal spring forces that act along the spring directions. The sum of these two forces is given as usual by (see previous Chapter) $\nabla\Pi$.

In addition, we now have a reaction force from the walls of the 'frictionless' track (i.e., the constraint). Although, we don't know the magnitude of this reaction force, the force will act normal to the curve since the track is frictionless, i.e., along the gradient of the curve, i.e.,let:

$$\vec{F}^{\text{Constraint}} = R\nabla h \tag{5.57}$$

Force balance requires that:

$$\nabla\Pi + R\nabla h = 0 \tag{5.58}$$

Comparing Equation (5.58) and Equation (5.13), we can identify the scalar $\|\mu\nabla h\|$ as the magnitude the reaction force due to a constraint on a structural system!

5.5 Lagrangian

The first order condition in Equation (5.43) can be made more compact, if one defines a Lagrangian

$$L(\overline{x},\overline{\mu}) = f(\overline{x}) + \mu_1 h_1(\overline{x}) + \mu_2 h_2(\overline{x}) + \ldots + \mu_M h_M(\overline{x}) \tag{5.59}$$

Observe that the gradient of the Lagrangian with respect to both $\overline{x}$ and each of the Lagrangian multiplier is given by:

$$\nabla L(\overline{x},\overline{\mu}) = \begin{Bmatrix} \nabla_{\overline{x}} L(\overline{x},\overline{\mu}) \\ \nabla_{\mu_1} L(\overline{x},\overline{\mu}) \\ \nabla_{\mu_2} L(\overline{x},\overline{\mu}) \\ \ldots \\ \nabla_{\mu_M} L(\overline{x},\overline{\mu}) \end{Bmatrix} = \begin{Bmatrix} \nabla f + \sum_{i=1}^{M} \mu_i \nabla h_i \\ h_1 \\ h_2 \\ \ldots \\ h_M \end{Bmatrix} \tag{5.60}$$

Thus, Equation (5.16) can replaced by a simple statement, that at a stationary point, the gradient of the Lagrangian vanishes:

$$\nabla L(\overline{x},\overline{\mu}) = 0 \tag{5.61}$$

Further, the second order condition may be stated as follows: a stationary point is minimum if the matrix:

$$\nabla^2_{\overline{x}} L(\overline{x},\overline{\mu}) = 0 \tag{5.62}$$

is positive definite. Observe that the differentiation in Equation (5.61) is with respect to both $\overline{x}$ and $\overline{\mu}$, whereas the differentiation in Equation(5.62) is only with respect to both $\overline{x}$. Although, no new concept has been introduced via the Lagrangian function, it plays an important role in constrained optimization for various reasons.

Example: For the problem:

$$\begin{aligned}&\underset{x,y}{Min}\ \ x^2+y^2\\&2x+y-1=0\end{aligned}\tag{5.63}$$

the Lagrangian is:

$$L(x,y,\mu)=x^2+y^2+\mu(2x+y-1)\tag{5.64}$$

♦

Example: For the problem:

$$\begin{aligned}&\underset{x,y}{Min}\ \ x^2+y^2+z^2\\&x+y+z-1=0\\&x+2y+3z-4=0\end{aligned}\tag{5.65}$$

the Lagrangian is:

$$L(x,y,z,\mu_1,\mu_2)=x^2+y^2+z^2+\mu_1(x+y+z-1)+\mu_2(x+2y+3z-4)\tag{5.66}$$

♦

Example: For the two-spring problem:

$$\begin{aligned}&\underset{u,v}{Min}\ \ \Pi=\frac{1}{2}100\left(\sqrt{u^2+(v+1)^2}-1\right)^2+\frac{1}{2}500\left(\sqrt{u^2+(v-1)^2}-1\right)^2-(10u+8v)\\&h=v-u^3=0\end{aligned}\tag{5.67}$$

the Lagrangian is:

$$L(u,v,\mu)=\begin{bmatrix}\frac{1}{2}100\left(\sqrt{u^2+(v+1)^2}-1\right)^2+\\\frac{1}{2}500\left(\sqrt{u^2+(v-1)^2}-1\right)^2\\-(10u+8v)\\+\mu(v-u^3)\end{bmatrix}\tag{5.68}$$

♦

5.6 Quadratic Problems

Recall that for unconstrained problems, if the objective is quadratic:

$$f(\overline{x})=\frac{1}{2}\overline{x}^TQ\overline{x}+\overline{x}^Tb+c\tag{5.69}$$

then the stationary point could be determined in a 'single' step via:

$$\overline{x}^*=-Q^{-1}b\tag{5.70}$$

Further, the stationary point is a minimum if the matrix Q is positive definite. Here, we seek similar results for constrained problems.

Consider the first order condition in Equation (5.43). The objective is to solve for the N+M unknowns $\overline{x}^*$ & $\overline{\mu}^*$. Equation (5.43) will be linear in $\overline{x}^*$ & $\overline{\mu}^*$ only if both the following two conditions are satisfied:

1. The gradient ∇f is linear with respect to $\overline{x}$, i.e., if the objective f is utmost quadratic with respect to $\overline{x}$, and

2. The gradients ∇h_i are independent of $\overline{x}$ (since we are multiplying ∇h_i by μ_i), i.e., the constraints h_i must be linear with respect to $\overline{x}$.

We therefore define a quadratic problem as one where the objective f is quadratic, and each of the constraints is linear. Further, linear constraints can always be expressed in the form:

$$A\overline{x} + d = 0 \tag{5.71}$$

where A is a (M,N) matrix and d is a (N,1) vector. Thus, a quadratic problem is given by:

$$\begin{aligned} &\underset{\overline{x}}{Min}\ \frac{1}{2}\overline{x}^T Q\overline{x} + \overline{x}^T b + c \\ &A\overline{x} + d = 0 \end{aligned} \tag{5.72}$$

Given a generic quadratic problem in Equation (5.72), one can now exploit Equation (5.43) to state the first order conditions as:

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{Bmatrix} \overline{x} \\ \overline{\mu} \end{Bmatrix} = \begin{Bmatrix} -b \\ -d \end{Bmatrix} \tag{5.73}$$

The matrix in the above equation is referred to as the KKT matrix for a quadratic problem. Further, since the constraints are linear, we conclude from Equation (5.17) that the point is a minimum if the matrix Q is positive definite.

Example: The problem in Equation (5.63) can be posed as:

$$\begin{aligned} &\underset{\overline{x}}{Min}\ \frac{1}{2}\overline{x}^T \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \overline{x} + \overline{x}^T \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} + 0 \\ &\begin{bmatrix} 2 & 1 \end{bmatrix} \overline{x} - 1 = 0 \end{aligned} \tag{5.74}$$

The KKT condition is:

$$\begin{bmatrix} 2 & 0 & 2 \\ 0 & 2 & 1 \\ 2 & 1 & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ \mu \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix} \tag{5.75}$$

♦

Example: The problem in Equation (5.65) can be posed as:

$$\begin{aligned} &\underset{\overline{x}}{Min}\ \frac{1}{2}\overline{x}^T \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \overline{x} + \overline{x}^T \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} + 0 \\ &\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \overline{x} + \begin{Bmatrix} -1 \\ -4 \end{Bmatrix} = 0 \end{aligned} \tag{5.76}$$

The KKT condition is:

$$\begin{bmatrix} 2 & 0 & 0 & 1 & 1 \\ 0 & 2 & 0 & 1 & 2 \\ 0 & 0 & 2 & 1 & 3 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \mu_1 \\ \mu_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 4 \end{Bmatrix} \tag{5.77}$$

♦

5.7 Overview of Algorithms for Constrained Optimization

We shall briefly discuss three classes of methods for solving equality constrained problems: (1) Nonlinear solvers, (2) Quadratic Penalty methods, and (3) Augmented Lagrangian Penalty methods.

5.7.1 Quadratic Penalty Methods

In penalty methods, given a constrained problem:

$$\begin{aligned} &\underset{\overline{x}}{Min}\ f(\overline{x}) \\ &h(\overline{x}) = 0 \end{aligned} \tag{5.78}$$

we replace it with an unconstrained problem of the form:

$$\underset{\overline{x}}{Min}\ f(\overline{x}) + P(h(\overline{x})) \tag{5.79}$$

where $P(h(\overline{x}))$ is a penalty function that penalizes if the constraint is not satisfied. Various penalty functions are used in practice; the simplest one is the quadratic penalty:

$$\underset{\overline{x}}{Min}\ f(\overline{x}) + \alpha h^2(\overline{x}) \tag{5.80}$$

where α is the penalty.

By choosing a sequence of increasing α, one can show that, under certain conditions [Reference], the solution of Equation (5.80) will converge to the solution of Equation (5.78). Thus, algorithms such as nonlinear conjugate gradient, discussed previously, can be used in solving constrained problems.

Example: Consider the problem:

$$\begin{aligned} &\underset{x,y}{Min}\ x^2 + y^2 \\ &(x-2)^2 + y^2 - 1 = 0 \end{aligned} \tag{5.81}$$

In quadratic penalty methods, the above is replaced by:

$$\underset{x,y}{Min}\ \left(x^2 + y^2\right) + \alpha((x-2)^2 + y^2 - 1)^2 \tag{5.82}$$

First we create a quadratic penalty function as follows.

```
function val = QuadraticPenalty(xBar,alpha)
% The constrained problem is
% Min x^2 + y^2
% (x-2)^2 + y^2 -1 = 0;
% The quadratic penalty problem is
% Min (x^2 + y^2) + alpha*((x-2)^2 + y^2 -1)^2
x = xBar(1);
```

```
y = xBar(2);
val = (x^2 + y^2) + alpha*((x-2)^2 + y^2 -1)^2;
```

Script 5-1: Quadratic Penalty function.

We minimize the quadratic penalty function via fminunc, where the solution from one iteration is used as an initial guess for the next iteration (as α is increased).

```
xBar0 = [0;0];
for alpha = [1 10 10^2 10^3 10^4]
      alpha
      xBar0 = fminunc(@(xBar)QuadraticPenalty(xBar,alpha),xBar0)
end
```

Script 5-2: Matlab script for minimizing quadratic penalty function.

For $\alpha = 1$, an approximate solution $(0.834, 0)$ is obtained, and for $\alpha = 10$, a solution is $(0.976, 0)$ is obtained, eventually converging to the exact solution is $(1, 0)$.

♦

5.7.2 Augmented Lagrangian Penalty Methods

Unfortunately, quadratic penalty functions do not converge to the correct solution for small values of α, and may exhibit ill conditioning for large values of α.

Therefore, in practice, augmented Lagrangian methods have more favorable convergence properties. Recall that, for the problem in Equation(5.78), the Lagrangian is defined by:

$$L = f(\bar{x}) + \mu h(\bar{x}) \tag{5.83}$$

An augmented Lagrangian is defined by adding a quadratic term to the Lagrangian:

$$L_A = f(\bar{x}) + \mu h(\bar{x}) + \alpha h^2(\bar{x}) \tag{5.84}$$

Now we solve the unconstrained problem (where α_k and μ_k are given):

$$\underset{\bar{x}}{Min} \;\; f(\bar{x}) + \mu_k h(\bar{x}) + \alpha_k h^2(\bar{x}) \tag{5.85}$$

for $\bar{x}_k$. Further, we can get a better estimate of the Lagrangian multiplier for the next iteration as follows. From the first order condition, we have:

$$\nabla f + \mu_k \nabla h + \alpha_k 2h \nabla h = 0 \tag{5.86}$$

i.e.,

$$\nabla f + \left(\mu_k + \alpha_k 2h\right) \nabla h = 0 \tag{5.87}$$

As we can see the above equation is 'identical' to the first part of Equation (5.16). Thus, we estimate the new Lagrangian multiplier as:

$$\mu_{k+1} = \left(\mu_k + \alpha_k 2h_k\right) \tag{5.88}$$

The convergence properties of augmented Lagrangian are discussed, for example, in [Reference].

Example: Consider the problem:

$$\begin{aligned} &\underset{x,y}{Min} \;\; x^2 + y^2 \\ &(x-2)^2 + y^2 - 1 = 0 \end{aligned} \tag{5.89}$$

In augmented Lagrangian, the above is replaced by:

$$\underset{x,y}{Min}\ \left(x^2+y^2\right)+\mu((x-2)^2+y^2-1)+\alpha((x-2)^2+y^2-1)^2 \tag{5.90}$$

The augmented Lagrangian script is as follows.

```
function val = AugmentedLagrangian(xBar,mu,alpha)
% The constrained problem is
% Min x^2 + y^2
% (x-2)^2 + y^2 -1 = 0;
% The augmented Lagrangian is
% Min (x^2 + y^2) + mu*((x-2)^2 + y^2 -1) + alpha*((x-2)^2 + y^2 -1)^2
x = xBar(1);
y = xBar(2);
val = (x^2 + y^2) + mu*((x-2)^2 + y^2 -1) + alpha*((x-2)^2 + y^2 -1)^2;
```

Script 5-3: Augmented Lagrangian example.

We minimize the augmented Lagrangian as follows, where the solution from one iteration is used as an initial guess for the next iteration (as α is increased).

```
xBar0 = [0;0];
mu = 0;
for alpha = [1 10 10^2 10^3 10^4]
      alpha
      xBar0 = fminunc(@(xBar)AugmentedLagrangian(xBar,mu,alpha),xBar0)
      mu = mu + 2*alpha*((xBar0(1)-2)^2 + xBar0(2)^2 -1)
end
```

Script 5-4: Matlab script for minimizing augmented Lagrangian.

For $\alpha=1$, $\overline{x}=(0.834,0),\mu=0.716$, for $\alpha=10$, $\overline{x}=(0.993,0),\mu=0.987$; for $\alpha=100$, $\overline{x}=(1,0),\mu=1$; recall that the exact solution is $\overline{x}=(1,0),\mu=1$.

♦

5.7.3 Nonlinear Solvers

Consider the constrained nonlinear problem:

$$\begin{aligned}&\underset{\overline{x}}{Min}\ f(\overline{x})\\&h(\overline{x})=0\end{aligned} \tag{5.91}$$

Recall that the first order condition results in a set of nonlinear equations:

$$G(\overline{x},\mu)=\begin{Bmatrix}\nabla f+\mu\nabla h\\ h\end{Bmatrix}=\begin{Bmatrix}0\\0\end{Bmatrix} \tag{5.92}$$

Let us assume that we have a guess $\overline{x}_k,\mu_k$ for the above set of equations. The Newton-Raphson method for finding the next guess is (see Equation (3.41)):

$$\begin{Bmatrix}\overline{x}_{k+1}\\ \mu_{k+1}\end{Bmatrix}=\begin{Bmatrix}\overline{x}_k\\ \mu_k\end{Bmatrix}-\left(\nabla G_k\right)^{-1}G_k \tag{5.93}$$

where the Jacobian ∇G_k is given by:

$$\nabla G_k=\begin{bmatrix}\nabla^2 f+\mu\nabla^2 h & \nabla h\\ (\nabla h)^T & 0\end{bmatrix}_{(\overline{x}_k,\mu_k)} \tag{5.94}$$

and

$$G_k = \begin{Bmatrix} \nabla f + \mu \nabla h \\ h \end{Bmatrix}_{(\bar{x}_k, \mu_k)} \tag{5.95}$$

Direct implementation of Newton's method is non-trivial since the Jacobian computation involves second derivatives. In practice, ∇G_k is computed approximately via Quasi-Newton methods such as BFGS [Reference]. This is referred to as Sequential Quadratic Programming (SQP). One drawback of the above method is that solving nonlinear equations yields stationary points, and not necessarily minima.

5.8 Solving Constrained Optimization via MATLAB's fmincon

Matlab's optimization toolbox provides a routine 'fmincon' for constrained optimization that is based on solving the nonlinear problem discussed earlier. The syntax of fmincon varies depending on whether the constraints are linear or nonlinear (the reader is encouraged to use the help files in Matlab to understand the syntax).

Example: Consider the problem:

$$\begin{aligned} &\underset{x,y}{Min}\ \ x^2 + y^2 \\ &x + y - 1 = 0 \end{aligned} \tag{5.96}$$

The Matlab script for solving the above problem is as follows (where fObj computes $x^2 + y^2$).

```
[xBar,f,flag,out,lambda] = fmincon(@(xBar)fObj(xBar),[0;0],[],[],[1 1],[1])
```

Script 5-5: Solving a 2-variable linear constraint problem.

♦

Example: Consider the problem:

$$\begin{aligned} &\underset{x,y}{Min}\ \ x^2 + y^2 + z^2 \\ &x + y + z - 1 = 0 \\ &x + 2y + 3z - 4 = 0 \end{aligned} \tag{5.97}$$

The Matlab script for solving the above problem is as follows (where fObj3D now computes $x^2 + y^2 + z^2$).

```
[xBar,f,flag,out,lam]=fmincon(@(xBar)fObj3D(xBar),[0;0;0],[],[],[1   1   1;   1   2
3],[1;4])
```

Script 5-6: Solving a 3 variable linear constraint problem.

♦

Example: Consider the problem:

$$\begin{aligned} &\underset{x,y}{Min}\ \ x^2 + y^2 \\ &(x-2)^2 + y^2 - 1 = 0 \end{aligned} \tag{5.98}$$

Since we have a nonlinear constraint, we must first create a Matlab script to capture the constraint; see below. Matlab syntax requires that both equality constraints and inequality constraints (to be discussed in the next chapter) be handled within the same function. Since we do not have any inequality constraints we set the first output g = []. The second output argument is the equality constraint above.

```
function [g,h] = NonLinearConstraints(xBar)
g = [];
```

```
h = (xBar(1)-2)^2 + xBar(2)^2 -1;
```

Script 5-7: The nonlinear constraint function.

Then the Matlab script for solving the above problem is as follows (where fObj computes $x^2 + y^2$).

```
[xBar,f,flag,out,lambda]   =   fmincon(@(xBar)fObj(xBar),[0;0],[],[],[],[],[],[],
@(xBar) NonLinearConstraints(xBar))
[xBar,f,flag,out,lambda]   =   fmincon(@(xBar)fObj(xBar),[4;0],[],[],[],[],[],[],
@(xBar) NonLinearConstraints(xBar))
```

Script 5-8: Solving a nonlinear constraint problem

Observe that, depending on the starting point, different solutions are returned (can you figure out why?).

♦

6 Equality and Inequality Constraints

Equation Section (Next) Highlights

- In this chapter, we consider problems of the form:

$$\underset{\overline{x}}{Min}\ f(\overline{x})$$
$$h_i(\overline{x}) = 0; i = 1,2,\ldots,M$$
$$g_j(\overline{x}) \le 0; j = 1,2,\ldots,L$$

- The first order condition for the stationary points $\overline{x}^*$ involves as many *Lagrangian multipliers* λ_i as the number of inequality constraints, and is given by the set of (L+N) Karush-Kuhn-Tucker (KKT) equations:

$$\nabla f(\overline{x}^*) + \sum_{j=1}^{L} \lambda_j^* \nabla g_j(\overline{x}^*) = 0$$

$$\left\{\begin{matrix} g_j(\overline{x}^*) \le 0 \\ \lambda_j^* g_j(\overline{x}^*) = 0 \\ \lambda_j^* \ge 0 \end{matrix}\right\} j = 1,2,\ldots,L$$

- As before, a Lagrange multiplier may be interpreted in two ways: (1) It captures the sensitivity of the objective function to the associated constraint, and (2) In a structural problem, it represents the reaction force at a kinematic constraint.
- Typical algorithms for solving inequality constraint minimization problems include active set framework that convert such problems into unconstrained or equality constrained problems.

In this chapter, we consider multivariable minimization problems with equality and *inequality constraints* of the form:

$$\begin{aligned} &\underset{\overline{x}}{Min}\ f(\overline{x}) \\ &h_i(\overline{x}) = 0; i = 1,2,\ldots,M \\ &g_j(\overline{x}) \le 0; j = 1,2,\ldots,L \end{aligned} \tag{6.1}$$

6.1 Feasible Regions

A feasible region is defined as the set of points that satisfy:

$$\begin{aligned} &h_i(\overline{x}) = 0; i = 1,2,\ldots,M \\ &g_j(\overline{x}) \le 0; j = 1,2,\ldots,L \end{aligned} \tag{6.2}$$

For example, for the problem:

$$\begin{aligned} &\underset{x,y}{Min}\ x^2 + y^2 \\ &x + y - 1 \le 0 \end{aligned} \tag{6.3}$$

the feasible region is the region *below* and *including* the straight line $x + y - 1 = 0$.

Similarly, for the problem:

$$\begin{aligned} &\underset{x,y}{Min}\ \ x^2 + y^2 \\ &x - 2 = 0 \\ &(x-2)^2 + y^2 - 1 \leq 0 \end{aligned} \tag{6.4}$$

the feasible region consists of the points on the line x =2, that lie within the disc of radius 1, centered at (2,0).

Sometimes, a problem has no feasible region. For example:

$$\begin{aligned} &\underset{x,y}{Min}\ \ x^2 + y^2 \\ &x + 6 \leq 0 \\ &(x-2)^2 + y^2 - 1 \leq 0 \end{aligned} \tag{6.5}$$

Finally, for the problem:

$$\begin{aligned} &\underset{x,y}{Min}\ \ (x - 10)^2 + (y-8)^2 \\ &x + y - 12 \leq 0 \\ &x - 8 \leq 0 \end{aligned} \tag{6.6}$$

the feasible region is intersection of (1) the region *below* and *including* the straight line $x + y - 12 = 0$, and (2) region *below* and *including* the straight line $x - 8 = 0$. All solutions to an optimization problem must lie within the feasible region.

6.2 First Order Conditions for Inequality Problems

For simplicity, consider a problem with a single inequality constraint (and no equality constraints):

$$\begin{aligned} &\underset{\overline{x}}{Min}\ \ f(\overline{x}) \\ &g(\overline{x}) \leq 0 \end{aligned} \tag{6.7}$$

Let $\overline{x}^*$ be a minimum of the above problem. There are two possible scenarios either (a) $g(\overline{x}^*) < 0$ or (b) $g(\overline{x}^*) = 0$. In other words, either the point $\overline{x}^*$ lies in the interior of the feasible region, or on the boundary of the constraint. Let us consider the two cases.

Case (A): If $\overline{x}^*$ lies in the interior, i.e., if $g(\overline{x}^*) < 0$, the constraint is said to be *inactive*.

Since the point lies inside the feasible region, deleting the constraint does not affect the solution (prove this statement!), i.e., $\overline{x}^*$ is also the minimum of the unconstrained problem:

$$\underset{\overline{x}}{Min}\ \ f(\overline{x}) \tag{6.8}$$

Therefore, $\overline{x}^*$ must satisfy the following first order condition:

$$\begin{aligned} &\nabla f(\overline{x}^*) = 0 \\ &g(\overline{x}^*) < 0 \end{aligned} \tag{6.9}$$

Case (B): If $\overline{x}^*$ lies on the boundary, i.e., if $g(\overline{x}^*) = 0$, the constraint is said to be *active*

By definition, Equation (6.7) reduces to the equality constrained problem:

$$\begin{aligned} &\underset{\overline{x}}{Min}\ f(\overline{x}) \\ &g(\overline{x}) = 0 \end{aligned} \tag{6.10}$$

Therefore, $\overline{x}^*$ must satisfy the following first order condition:

$$\begin{aligned} &\nabla f(\overline{x}^*) + \lambda^* \nabla g(\overline{x}^*) = 0 \\ &g(\overline{x}^*) = 0 \end{aligned} \tag{6.11}$$

for some λ^*.

Thus, in conclusion, for the problem in Equation (6.7), either Equation (6.9) or Equation (6.11) must be satisfied at a minimum $\overline{x}^*$. Further Equations (6.9) and (6.11)can be combined into a single equation:

$$\begin{aligned} &\nabla f(\overline{x}^*) + \lambda^* \nabla g(\overline{x}^*) = 0 \\ &\lambda^* g(\overline{x}^*) = 0 \\ &g(\overline{x}^*) \leq 0 \end{aligned} \tag{6.12}$$

where $\lambda^* g(\overline{x}^*) = 0$, referred to as the complementarity condition, that states that either $\lambda^* = 0$ or $g(\overline{x}^*) = 0$. In other words, $\overline{x}^*$ must satisfy Equation (6.12).

However, an additional constraint must be posed on λ^* as the following argument reveals. Consider again the case when $\overline{x}^*$ lies on the boundary, i.e., $g(\overline{x}^*) = 0$. Note that the gradient $\nabla g(\overline{x}^*)$, by definition, is directed from the feasible region (where $g(\overline{x}) < 0$) to the infeasible region (where $g(\overline{x}) > 0$). Further, from Equation (6.12):

$$\nabla f(\overline{x}^*) = -\lambda^* \nabla g(\overline{x}^*) \tag{6.13}$$

Now, suppose $\lambda^* < 0$, since $\nabla g(\overline{x}^*)$ points from feasible to infeasible region, $\nabla f(\overline{x}^*)$ must also point in the same direction, i.e., from feasible to infeasible region. In other words, the function f is increasing towards the infeasible region, and decreasing towards the feasible region. Thus, we can always take an infinitesimal step into the feasible region and reduce f, i.e., $\overline{x}^*$ cannot possibly be the minimum. Thus, we conclude $\lambda^* \geq 0$. Adding this additional constraint to Equation (6.12), we arrive at the following theorem.

Theorem 6-1: For the problem:

$$\begin{aligned} &\underset{\overline{x}}{Min}\ f(\overline{x}) \\ &g(\overline{x}) \leq 0 \end{aligned} \tag{6.14}$$

the minimum $\overline{x}^*$ must satisfy the first order condition (a necessary but not sufficient condition):

$$\begin{aligned}
&\nabla f(\bar{x}^*) + \lambda^* \nabla g(\bar{x}^*) = 0 \\
&g(\bar{x}^*) \le 0 \\
&\lambda^* g(\bar{x}^*) = 0 \\
&\lambda^* \ge 0
\end{aligned} \tag{6.15}$$

Further, a point $\bar{x}^*$ that satisfies the above equation is a minimum if the matrix:

$$\nabla^2 f(\bar{x}^*) + \lambda^* \nabla^2 g(\bar{x}^*) \tag{6.16}$$

is positive definite (a sufficient but not necessary condition).

Proof: See [Reference].

♦

In the examples below, we will guess the minimum via a geometric analysis, and verify that Equations (6.15) and (6.16) are indeed satisfied. In the next Section, we shall discuss an algorithm for finding the minimum of inequality problems.

Example: Consider the problem:

$$\begin{aligned}
&\underset{x,y}{Min}\ \ x^2 + y^2 \\
&x + y - 1 \le 0
\end{aligned} \tag{6.17}$$

By plotting the constraint, we can easily see that the minimum occurs at $\bar{x}^* = (0,0)$, i.e., the constraint is inactive; therefore $\lambda^* = 0$. Given $\bar{x}^* = (0,0)$ and $\lambda^* = 0$, one can verify that Equations (6.15) and (6.16) are satisfied.

♦

Example: Consider the problem:

$$\begin{aligned}
&\underset{x,y}{Min}\ \ x^2 + y^2 \\
&x + y + 1 \le 0
\end{aligned} \tag{6.18}$$

Once again, by plotting the constraint, we can see that the minimum occurs at $\bar{x}^* = (-1/2, -1/2)$, i.e., the constraint is active. Further, by solving the equality constrained problem:

$$\begin{aligned}
&\underset{x,y}{Min}\ \ x^2 + y^2 \\
&x + y + 1 = 0
\end{aligned} \tag{6.19}$$

we obtain $\lambda^* = 1$. One can verify that Equations (6.15) and (6.16) are satisfied.

♦

6.3 Active Set Algorithm

Consider the problem:

$$\begin{aligned}
&\underset{\bar{x}}{Min}\ \ f(\bar{x}) \\
&g(\bar{x}) \le 0
\end{aligned} \tag{6.20}$$

In general, we don't know a priori whether the inequality is active or inactive. Therefore:

1. First, we assume that the constraint is inactive, i.e., we solve the unconstrained problem to obtain all stationary points $\overline{x}^*$ (with $\lambda^* = 0$). Now, for each stationary point, we verify if Equations (6.15) and (6.16) are satisfied, otherwise, such points are rejected.
2. Next, we assume that the constraint is active, and solve the equality problem to obtain all stationary points $\overline{x}^*$ and corresponding Lagrange multipliers λ^*. For each stationary point, we verify if Equations (6.15) and (6.16) are satisfied, otherwise, such points are rejected.

Observe that, both cases must be considered.

Example: Consider the problem:

$$\begin{aligned} &\underset{x,y}{Min}\ \ x^2 + y^2 \\ &x + y - 1 \leq 0 \end{aligned} \tag{6.21}$$

First, we assume that the constraint is inactive, i.e.,

$$\underset{x,y}{Min}\ \ x^2 + y^2 \tag{6.22}$$

leading to the stationary point $\overline{x}^* = (0,0)$ with $\lambda^* = 0$. Since Equations (6.15) and (6.16) are indeed satisfied, $\overline{x}^* = (0,0)$ is a valid minimum.

Next, we solve:

$$\begin{aligned} &\underset{x,y}{Min}\ \ x^2 + y^2 \\ &x + y - 1 = 0 \end{aligned} \tag{6.23}$$

resulting in the stationary point $\overline{x}^* = (1/2, 1/2)$ with $\lambda^* = -1$. Equation (6.15) is not satisfied, and the point is rejected.

♦

Example: Consider the problem:

$$\begin{aligned} &\underset{x,y}{Min}\ \ x^2 + y^2 \\ &x + y + 1 \leq 0 \end{aligned} \tag{6.24}$$

First, we assume that the constraint is inactive:

$$\underset{x,y}{Min}\ \ x^2 + y^2 \tag{6.25}$$

leading to the minimum $\overline{x}^* = (0,0)$ with $\lambda^* = 0$. Since $g(\overline{x}^*) > 0$, Equation (6.15) is not satisfied … the point is rejected.

Next, we assume that the constraint is active:

$$\begin{aligned} &\underset{x,y}{Min}\ \ x^2 + y^2 \\ &x + y + 1 = 0 \end{aligned} \tag{6.26}$$

leading to the minimum $\overline{x}^* = (-1/2, -1/2)$, and $\lambda^* = 1$. Since Equations (6.15) and (6.16) are satisfied, and $\overline{x}^* = (-1/2, -1/2)$ is a valid minimum.

♦

Example: Consider the problem:

$$\begin{aligned} &\underset{x,y}{Min}\ x^2 + y^2 \\ &(x-2)^2 + y^2 - 1 \le 0 \end{aligned} \tag{6.27}$$

Suppose we assume that the constraint is inactive:

$$\underset{x,y}{Min}\ x^2 + y^2 \tag{6.28}$$

leading to the stationary point $\overline{x}^* = (0,0)$. Since $g(\overline{x}^*) = 3 > 0$, $\overline{x}^* = (0,0)$ is not a minimum. Next, solve:

$$\begin{aligned} &\underset{x,y}{Min}\ x^2 + y^2 \\ &(x-2)^2 + y^2 - 1 = 0 \end{aligned} \tag{6.29}$$

leading to the two stationary points: (a) $\overline{x}^* = (1,0)$, and $\lambda^* = 1$, and (b) $\overline{x}^* = (3,0)$, and $\lambda^* = -1.5$. For the first stationary point, since Equations (6.15) and (6.16) are satisfied, $\overline{x}^* = (1,0)$ is a valid stationary point. For the second stationary point, since $\lambda^* < 0$, $\overline{x}^* = (3,0)$ is not a valid minimum. There is only one minimum.

♦

Example: We now modify the previously problem:

$$\begin{aligned} &\underset{x,y}{Min}\ x^2 + y^2 \\ &1 - (x-2)^2 - y^2 \le 0 \end{aligned} \tag{6.30}$$

First:

$$\underset{x,y}{Min}\ x^2 + y^2 \tag{6.31}$$

Once again, we get the minimum $\overline{x}^* = (0,0)$. Since Equations (6.15) and (6.16) are satisfied, $\overline{x}^* = (0,0)$ is a valid minimum.

Next:

$$\begin{aligned} &\underset{x,y}{Min}\ x^2 + y^2 \\ &1 - (x-2)^2 - y^2 = 0 \end{aligned} \tag{6.32}$$

leading to two stationary points: (a) $\overline{x}^* = (1,0)$, and $\lambda^* = -1$, and (b) $\overline{x}^* = (3,0)$, and $\lambda^* = 1.5$. For the first stationary point, since $\lambda^* < 0$, $\overline{x}^* = (1,0)$ is rejected. For the second stationary point, we can verify that Equations (6.15) and (6.16) are satisfied. Thus, there two minima: $\overline{x}^* = (0,0)$ and $\overline{x}^* = (3,0)$.

♦

6.4 Multiple Inequality Constraints

Equations (6.15) and (6.16) may now be generalized to multiple inequality constraints. **Theorem 6-2**: For the problem:

$$\begin{aligned}&\underset{\bar{x}}{Min}\ \ f(\bar{x})\\&g_j(\bar{x}) \le 0; j = 1,2,\ldots,L\end{aligned} \tag{6.33}$$

the minimum $\bar{x}^*$ must satisfy the first order condition (a necessary but not sufficient condition):

$$\begin{aligned}&\nabla f(\bar{x}^*) + \sum_{j=1}^{L}\lambda_j^*\nabla g_j(\bar{x}^*) = 0\\&\left.\begin{cases}g_j(\bar{x}^*) \le 0\\ \lambda_j^* g_j(\bar{x}^*) = 0\\ \lambda_j^* \ge 0\end{cases}\right\} j = 1,2,\ldots,L\end{aligned} \tag{6.34}$$

for some set of scalars λ_j^*. Further, a point $\bar{x}^*$ that satisfies the above equation is a minimum if the matrix:

$$\nabla^2 f(\bar{x}^*) + \sum_{j=1}^{L}\lambda_j^*\nabla^2 g_j(\bar{x}^*) \tag{6.35}$$

is positive definite.

Proof: See [Reference].

♦

Similarly, the notion of active set framework can be generalized to multiple inequality constraints as follows.

Consider the generic problem:

$$\begin{aligned}&\underset{\bar{x}}{Min}\ \ f(\bar{x})\\&g_j(\bar{x}) \le 0; j = 1,2,\ldots,L\end{aligned} \tag{6.36}$$

One must now consider all possible combinations of active constraints. Given L inequality constraints, one must consider 2^L combinations, namely:

$$\begin{aligned}&\text{No constraint: } \{\phi\}\\&\text{One constraint: } \{1\},\{2\}\ldots\{L\}\\&\text{Two constraints: } \{1,2\},\{1,3\},\ldots\{L-1,L\}\\&\ldots\\&\text{All constraints: } \{1,2,3,\ldots,L\}\end{aligned} \tag{6.37}$$

For each of the above choice, we shall use the symbol A to refer to the set of active constraints. For example, $A = \{1,3,7\}$ implies constraints 1,3 and 7 are active, while the remaining are assumed to be inactive.

1. For each set A, we solve the equality problem:

$$\begin{aligned}&\underset{\bar{x}}{Min}\ \ f(\bar{x})\\&g_k(\bar{x}) = 0; k \in A\end{aligned} \tag{6.38}$$

Let a stationary point be $\overline{x}^*$, and let the Lagrangian corresponding to the active constraints be $\lambda_k^*, k \in A$. Only the points that satisfy Equations (6.34) and (6.35) are retained, and the rest are rejected.

Example: Consider the problem:

$$\begin{aligned} &\underset{x,y}{Min}\ \ (x - 10)^2 + (y-8)^2 \\ &x + y - 12 \le 0 \\ &x - 8 \le 0 \end{aligned} \qquad (6.39)$$

The combinations of active constraints are:

$$\{\phi\}, \{1\}, \{2\}, \{1,2\} \qquad (6.40)$$

1. $A = \{\phi\}$

$$\underset{x,y}{Min}\ \ (x - 10)^2 + (y-8)^2 \qquad (6.41)$$

The stationary point is $\overline{x}^* = (10, 8)$. However, Equation (6.34) is not satisfied.

2. $A = \{1\}$

$$\begin{aligned} &\underset{x,y}{Min}\ \ (x - 10)^2 + (y-8)^2 \\ &x + y - 12 = 0 \end{aligned} \qquad (6.42)$$

The stationary point is $\overline{x}^* = (7, 5)$ with Lagrange multiplier 6. Equations (6.34) and (6.35) are satisfied.

3. $A = \{2\}$

$$\begin{aligned} &\underset{x,y}{Min}\ \ (x - 10)^2 + (y-8)^2 \\ &x - 8 = 0 \end{aligned} \qquad (6.43)$$

The stationary point is $\overline{x}^* = (8, 8)$ with Lagrange multiplier 4. However, Equation (6.34) is not satisfied.

4. $A = \{1,2\}$

$$\begin{aligned} &\underset{x,y}{Min}\ \ (x - 10)^2 + (y-8)^2 \\ &x + y - 12 = 0 \\ &x - 8 = 0 \end{aligned} \qquad (6.44)$$

The stationary point is $\overline{x}^* = (8, 4)$ with Lagrange multipliers 8 and -4 (for the two constraints). However, Equation (6.34) is not satisfied.

Conclusion: In conclusion, the minimum for the problem in Equation (6.39) is $\overline{x}^* = (7, 5)$ with Lagrange multipliers $\lambda_1^* = 6$ (active) and $\lambda_2^* = 0$ (inactive).

♦

The active set methodology is one popular framework for solving inequality constrained problems. Other algorithms include log-barrier methods [Reference].

6.5 Summary for Equality and Inequality Problems

We now combine Theorem 5-2 on equality constraints and Theorem 6-2 on inequality constraints into a single theorem.

Theorem 6-3: For the problem:

$$\begin{aligned}&\underset{\overline{x}}{Min}\ \ f(\overline{x})\\ &h_i(\overline{x})=0; i=1,2,\ldots,M\\ &g_j(\overline{x})\le 0; j=1,2,\ldots,L\end{aligned} \tag{6.45}$$

the minimum $\overline{x}^*$ must satisfy the first order (a necessary but not sufficient) condition:

$$\begin{aligned}&\nabla f(\overline{x}^*)+\sum_{i=1}^{M}\mu_i^*\nabla h_i(\overline{x}^*)+\sum_{j=1}^{L}\lambda_j^*\nabla g_j(\overline{x}^*)=0\\ &h_i(\overline{x}^*)=0, i=1,2,\ldots,M\\ &\left.\begin{cases} g_j(\overline{x}^*)\le 0\\ \lambda_j^* g_j(\overline{x}^*)=0\\ \lambda_j^*\ge 0\end{cases}\right\} j=1,2,\ldots,L\end{aligned} \tag{6.46}$$

for some set of scalars μ_i^* and λ_j^*. Further, a point $\overline{x}^*$ that satisfies the above equation is a minimum if the matrix:

$$\nabla^2 f(\overline{x}^*)+\sum_{i=1}^{M}\mu_i^*\nabla^2 h_i(\overline{x}^*)+\sum_{j=1}^{L}\lambda_j^*\nabla^2 g_j(\overline{x}^*) \tag{6.47}$$

is positive definite (a sufficient but not necessary condition).

Proof: See [Reference].

♦

The active set algorithm applies as before on the inequality constraints.

Example: Consider the problem:

$$\begin{aligned}&\underset{x,y}{Min}\ \ x^2+y^2\\ &x-2=0\\ &(x-2)^2+y^2-1\le 0\end{aligned} \tag{6.48}$$

1. $A=\{\phi\}$, leads to

$$\begin{aligned}&\underset{x,y}{Min}\ \ x^2+y^2\\ &x-2=0\end{aligned} \tag{6.49}$$

The stationary point is $\overline{x}^*=(2,0)$, with $\mu_1^*=1$ and $\lambda_1^*=0$. One can verify that Equations (6.46) and (6.47) are satisfied.

2. $A=\{1\}$, leads to

$$\underset{x,y}{Min}\ x^2 + y^2$$
$$x - 2 = 0 \tag{6.50}$$
$$(x-2)^2 + y^2 - 1 = 0$$

The stationary points are: (1) $\bar{x}^* = (2,1)$, and with $\mu_1^* = 1$ and $\lambda_1^* = 0$, and (2) $\bar{x}^* = (2,-1)$, and with $\mu_1^* = 1$ and $\lambda_1^* = 0$. One can verify that Equation (6.46) is not satisfied at either point.

Conclusion: The only minimum to the problem is $\bar{x}^* = (2,0)$, with $\mu_1^* = 1$ and $\lambda_1^* = 0$.

♦

6.6 Solving Constrained Optimization via MATLAB's fmincon

Matlab's optimization toolbox provides a routine 'fmincon' for constrained optimization that is based on the active set methodology. The reader is encouraged to use the help files in Matlab to understand the syntax of fmincon.

Example: Consider the problem:

$$\underset{x,y}{Min}\ x^2 + y^2$$
$$x + y - 1 \le 0 \tag{6.51}$$

The Matlab script for solving the above problem is as follows (where fObj computes $x^2 + y^2$).

```
[xBar,f,flag,out,lambda] = fmincon(@(xBar)fObj(xBar),[0;0],[1 1],[1])
```

Script 6-1: Solving a 2-variable linear constraint problem.

Matlab reports that the minimum is (0, 0) and the constraint is inactive.

♦

Example: Consider the problem:

$$\underset{x,y}{Min}\ (x - 10)^2 + (y-8)^2$$
$$x + y - 12 \le 0 \tag{6.52}$$
$$x - 8 \le 0$$

The Matlab script for solving the above problem is as follows (where fObj2 computes $\underset{x,y}{Min}\ (x - 10)^2 + (y-8)^2$).

```
[xBar,f,flag,out,lam]=fmincon(@(xBar)fObj2(xBar),[0;0],[1 1; 1 0],[12;8])
```

Script 6-2: Solving a 3 variable linear constraint problem.

Matlab reports that the minimum is (7, 5); further constraint-1 is active, but not constraint-2.

♦

Example: Consider the problem:

$$\underset{x,y}{Min}\ x^2 + y^2$$
$$(x-2)^2 + y^2 - 1 \le 0 \tag{6.53}$$

Since we have a nonlinear constraint, we must first create a Matlab script to capture the constraint; see below. Since we do not have any equality constraints we set the second output h = []. The first output argument is the inequality constraint above.

```
function [g,h] = NonLinearConstraints(xBar)
g = (xBar(1)-2)^2 + xBar(2)^2 -1;
h = [];
```

Script 6-3: The nonlinear constraint function.

Then the Matlab script for solving the above problem is as follows (where fObj computes $x^2 + y^2$).

```
[xBar,f,flag,out,lambda]   =   fmincon(@(xBar)fObj(xBar),[0;0],[],[],[],[],[],[],
@(xBar) NonLinearConstraints(xBar))
[xBar,f,flag,out,lambda]   =   fmincon(@(xBar)fObj(xBar),[4;0],[],[],[],[],[],[],
@(xBar) NonLinearConstraints(xBar))
```

Script 6-4: Solving a nonlinear constraint problem

♦

Example: Consider the problem:

$$\begin{aligned} &\underset{x,y}{Min}\ x^2 + y^2 \\ &x - 2 = 0 \\ &(x-2)^2 + y^2 - 1 \le 0 \end{aligned} \tag{6.54}$$

Then the Matlab script for solving the above problem is as follows (where fObj computes $x^2 + y^2$ and NonLinearConstraints is as before since the linear constraints have been handled separately below).

```
[xBar,f,flag,out,lambda]       =       fmincon(@(xBar)fObj(xBar),[0;0],[],[],[1
0],[2],[],[], @(xBar) NonLinearConstraints(xBar))
```

Script 6-5: Solving a nonlinear constraint problem

Matlab finds a single minimum (2, 0) with $\mu^* = -4$ and $\lambda^* = 0$ (the inequality constraint is inactive).

♦

6.7 Linearly Independent Gradients

Consider Equation (6.46) of Theorem 6-3 that consists of (N+M+L) equations and equal number of unknowns. In order to solve Equation (6.46), these equations must be linearly independent. There are many instances where this condition is not met and the underlying algorithm to solve Equation (6.46) may fail. We consider below a few typical instances.

Linearly dependent constraints:

One of the most common errors that a beginner might commit is to account for a constraint twice. For example, the nonlinear script file above is modified to include the constraint $x - 2 = 0$:

```
function [g,h] = NonLinearConstraints(xBar)
g = (xBar(1)-2)^2 + xBar(2)^2 -1;
h = [xBar(1)-2];
```

Script 6-6: The nonlinear constraint function.

Now, if one runs the script in Script 6-5, Matlab returns an error (flag = -2). The underlying reason, of course, is that one is attempting to solve the following (poorly posed) problem:

$$\begin{aligned}&\underset{x,y}{Min}\ x^2+y^2\\&x-2=0\\&x-2=0\\&(x-2)^2+y^2-1\le 0\end{aligned}\tag{6.55}$$

Linearly dependent gradients:

Similar problems arise when the gradients of the constraints are linearly dependent at a stationary point. For example, consider

$$\begin{aligned}&\underset{x,y}{Min}\ x^2+y^2\\&x-3=0\\&(x-2)^2+y^2-1=0\end{aligned}\tag{6.56}$$

The constraints are linearly independent. Further, geometrically, one can argue that the minimum occurs at (3, 0). Unfortunately, Matlab will fail since at the stationary point (3,0), the gradients of the constraints are linearly dependent since $\nabla h_1=(1,0)$ and $\nabla h_2=(2,0)$. Thus, not only must one be concerned about linearly dependent constraints, one must also be aware of constraints whose gradients are linearly dependent on each other at a stationary point.

Vanishing gradients:

A similar error occurs when the gradient of a constraint vanishes at a stationary point. For example, consider solving the following problem:

$$\begin{aligned}&\underset{x,y}{Min}\ x^2+y^2\\&(x-2)^2=0\end{aligned}\tag{6.57}$$

Although Matlab does find the correct solution (2,0), an error flag of 5 is returned (the Lagrange multiplier is a large value signaling a possible failure). The problem is that the gradient $\nabla h=2(x-2)$ vanishes at (2,0). Thus, numerically, one cannot solve for the associated Lagrange multiplier.

7 Discrete-Variable and Multi-Objective Optimization

Highlights

-

7.1 Discrete-Variable Problems

Equation Section (Next) In discrete-variable problems, some or all of the design variables are permitted to take only discrete values. There are two broad categories of discrete-variable problems: (1) problems where the objective and constraints can be evaluated at all continuous values (example: one may be interested in discrete values of bolt diameters, but their structural properties, etc may be evaluated at all positive continuous values), (2) problems where either the objective and/or constraints can only be evaluated at discrete values (example: where the design variables are the number of bolt holes … fractional bolt holes are not meaningful).

Here we consider problems of the former type, where the objective and constraints can be evaluated at all continuous values:

$$\begin{aligned}&\underset{\bar{x}}{Min}\ f(\bar{x}_D,\bar{x}_C),\\&h_i(\bar{x}_D,\bar{x}_C)=0; i=1,2,...,M\\&g_j(\bar{x}_D,\bar{x}_C)\le 0; j=1,2,...,L\\&x_D(i)\in D\end{aligned}\tag{7.1}$$

where $\bar{x}_D$ are the discrete design variables. The branch and bound algorithm is a popular method for solving Equation (7.1). It relies on continuous optimization algorithms, and is illustrated below via an example.

Example: Consider the discrete variable problem

$$\begin{aligned}&\underset{x,y}{Min}\ (x-3\pi)^2+(y-4\pi)^2\\&2x+y-10\le 0\\&x\in\left\{\frac{1}{\pi},\frac{2}{\pi},\frac{3}{\pi},...\right\}\\&y:\text{Positive integer}\end{aligned}\tag{7.2}$$

where both variables are discrete.

The branch and bound algorithm proceeds as follows. First we solve Equation (7.2) assuming all variables are continuous, i.e. we solve

Problem 1:

$$\begin{aligned}&\underset{x,y}{Min}\ (x-3\pi)^2+(y-4\pi)^2\\&2x+y-10\le 0\end{aligned}\tag{7.3}$$

The solution is:

Problem 1: $(0.85,\ 8.28); f_{\min}=91.72$

This is, of course, infeasible since the discrete requirements on x and y are not met. (But observe that with additional constraints on the variables, the objective cannot be any smaller than 91.72.) Now, since:

$$\frac{2}{\pi} \le 0.8545 \le \frac{3}{\pi} \tag{7.4}$$

we first branch into two continuous problems.

Problem 2:

$$\begin{aligned} &\underset{x,y}{Min}\;\; (x-3\pi)^2 + (y-4\pi)^2 \\ &2x + y - 10 \le 0 \\ &x \le \frac{2}{\pi} \end{aligned} \tag{7.5}$$

Problem 3:

$$\begin{aligned} &\underset{x,y}{Min}\;\; (x-3\pi)^2 + (y-4\pi)^2 \\ &2x + y - 10 \le 0 \\ &x \ge \frac{3}{\pi} \end{aligned} \tag{7.6}$$

The two problems are solved resulting in the following solutions:

Problem 2: $(2/\pi, 8.72); f_{\min} = 91.97$

Problem 3: $(3/\pi, 8.09); f_{\min} = 91.77$

We branch from Problem-2 to solve two additional problems with a single continuous variable:

Problem 4:

$$\begin{aligned} &\underset{y}{Min}\;\; (2/\pi-3\pi)^2 + (y-4\pi)^2 \\ &2(2/\pi) + y - 10 \le 0 \\ &y \le 8 \end{aligned} \tag{7.7}$$

Problem 5:

$$\begin{aligned} &\underset{y}{Min}\;\; (2/\pi-3\pi)^2 + (y-4\pi)^2 \\ &2(2/\pi) + y - 10 \le 0 \\ &y \ge 9 \end{aligned} \tag{7.8}$$

The two problems are solved resulting in:

Problem 4: $(2/\pi, 8); f_{\min} = 98.08$

Problem 5: No feasible solution

Next, we branch from Problem-3 to solve:

Problem 6:

$$\begin{aligned}
&\underset{y}{Min}\;\; (3/\pi - 3\pi)^2 + (y-4\pi)^2 \\
&2(3/\pi) + y - 10 \le 0 \\
&y \le 8
\end{aligned} \tag{7.9}$$

Problem 7:

$$\begin{aligned}
&\underset{y}{Min}\;\; (2/\pi - 3\pi)^2 + (y-4\pi)^2 \\
&2(2/\pi) + y - 10 \le 0 \\
&y \ge 9
\end{aligned} \tag{7.10}$$

The two problems are solved resulting in:

Problem 6: $(3/\pi, 8); f_{\min} = 92.59$

Problem 7: $\text{No feasible solution}$

Thus the solution is $(3/\pi, 8); f_{\min} = 92.59$.

♦

8 Truss Optimization

8.1 Tensile Bar

Equation Section (Next)Consider a tensile bar of length l, area A, Young's modulus E, fixed at ine end and subject to tip load 'P'. Since the bar is of uniform material and cross-section, recall that:

1. The stress within the bar at any cross-section is $\sigma = P / A$
2. The strain within the bar at any cross-section is $\varepsilon = \sigma / E = P / (EA)$
3. The net deflection is therefore $\Delta l = l\varepsilon = Pl / (EA)$

Observe that once the tip displacement has been determined, displacement at any cross-section can be determined via linear interpolation.

Truss systems consist of such axially loaded bars joined together at pinned joints (pinned joints can transfer forces, but not moments). For example, consider the truss system shown in the figure below, where node-1 is subject to a downward vertical force of 'P', while nodes 3 and 4 are pinned. In Matlab, one can describe the truss as follows:

```
Truss.NodeXY = [2 1 0 1; 0 1 0 2];
Truss.Connectivity = [1 3; 1 2; 2 3; 2 4]';
Truss.E = 2e11*[1 1 1 1];
Truss.Area = 0.0001*[1 1 1 1];
Truss.FreeNodes = [1 2];
Truss.Force = [0 -100;0 0]';
```

Script 8-1: Truss structure.

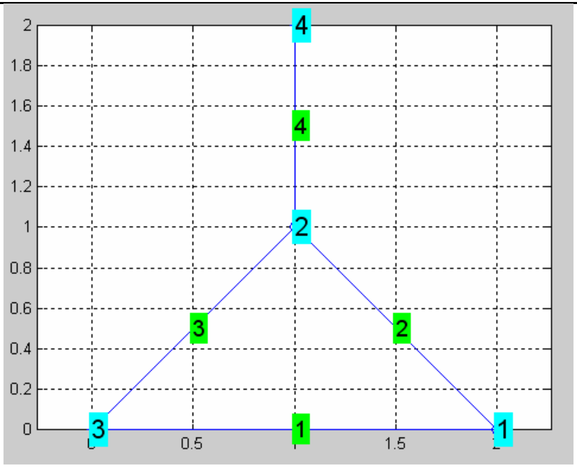


Figure 8-1: A four-bar truss system.

Various optimization problems may now be posed on such truss systems. For simplicity, we shall assume that the design variables are the cross-sectional areas of the truss members. For example:

Problem 1: Find the optimal areas such that the total volume of the truss system is minimized, with the following constraints: (1) the tip displacement of node-1 does not exceed a specified value, and (2) the stresses within the members do not exceed a specified tension and compression values.

Problem 2: Find the optimal areas such that the tip displacement of node-1 is minimized, with the following constraints: (1) the total volume displacement of node-1 is a specified value, and (2) the stresses within the members do not exceed specified tension and compression values.

One strategy, but an inefficient one, is to write specialized code for each optimization problem. A better strategy is to first develop a collection of common (truss) analysis modules, and test these modules prior to posing optimization problems. Typical analysis modules include: (1) Given the geometry, material and loading of any truss system, find the displacement at all nodes (assuming small displacement), (2) given the nodal displacements find the stresses, etc.

8.2 Truss Analysis

Consider the problem of finding the nodal displacements $u \equiv \{u_1, v_1, u_2, v_2, ...\}^T$ of a truss structure. Recall that trusses are a linear approximation of spring systems, and the nodal displacements satisfy:

$$Ku = f \tag{8.1}$$

where 'K' is the stiffness matrix (K is the Q matrix for small displacement spring systems), and f is the forcing vector (f is –b of spring systems). Recall that the stiffness matrix K is obtained via assembly (see algorithm for assembling Q matrix for small displacement spring systems):

$$K = \sum_{i=1,2...} k_i \tag{8.2}$$

where the element stiffness matrices are of the form:

$$k_i = \frac{EA_i}{l_i}\left[\quad\right] \tag{8.3}$$

For the truss in Figure 8-1 the displacements are found to be:

$$\{u_1, v_1, u_2, v_2\} = 10^{-4}\left[\text{-0.1} \quad \text{-0.583} \quad 0.24 \quad \text{-0.1}\right] \tag{8.4}$$

The deformed truss is illustrated below.

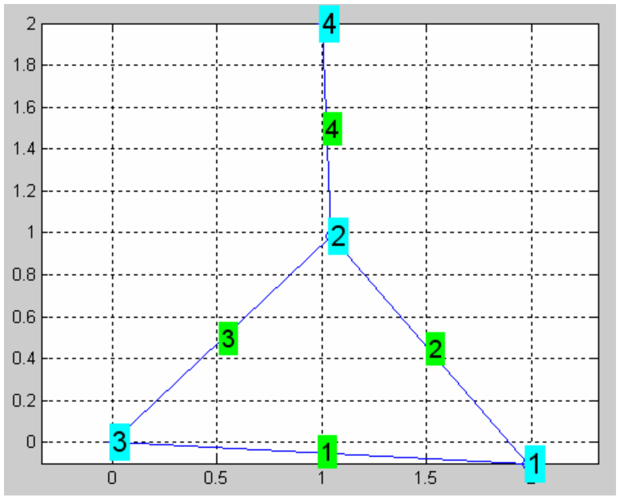


Figure 8-2: A four-bar truss system.

Once the nodal displacements the stresses in each element is given by:

$$\sigma = E\frac{\Delta l}{l} \tag{8.5}$$

and are found to be:

$$\{\sigma_1, \sigma_2, \sigma_3, \sigma_4\} = 10^6 \begin{bmatrix} -1 & 1.41 & 1.41 & 2 \end{bmatrix} \tag{8.6}$$

For verification, observe that the stress in the four bars is given by:

$$\sigma_1 = -\frac{P}{A_1}; \sigma_2 = \frac{P\sqrt{2}}{A_2}; \sigma_3 = \frac{P\sqrt{2}}{A_3}; \sigma_4 = \frac{2P}{A_4} \tag{8.7}$$

8.3 Truss Optimization

Having developed the basic modules, consider the optimization problem:

Problem 1: Find the optimal areas such that the total volume of the truss system is minimized, with the following constraints: (1) the tip displacement of node-1 does not exceed 0.01, and (2) the stresses within the members lies between 160MPa and -80 MPa.

Mathematically, the problem may be posed as:

$$\begin{aligned} &\underset{\{A_i\}}{Min} \sum_{i=1}^{4} A_i l_i \\ &v_1 - 0.01 \le 0 \\ &\sigma_i - 160e^6 \le 0; i = 1...4 \\ &-\sigma_i - 80e^6 \le 0; i = 1...4 \end{aligned} \tag{8.8}$$

Unfortunately, the above formulation is ill-posed in that the formulation permits zero and negative cross-sectional areas. To avoid degeneracy, we need to ensure that the areas take positive values (say greater than 10^{-8}) to result in:

$$\begin{aligned} &\underset{\{A_i\}}{Min} \sum_{i=1}^{4} A_i l_i \\ &v_1 - 0.001 \le 0 \\ &\sigma_i - 160e^6 \le 0; i = 1...4 \\ &-\sigma_i - 80e^6 \le 0; i = 1...4 \\ &-A_i + 10^{-8} \le 0; i = 1...4 \end{aligned} \tag{8.9}$$

It is recommended that the constraints and objective be scaled as follows:

$$\begin{aligned} &\underset{\{A_i\}}{Min} \left(\sum_{i=1}^{4} A_i l_i \right) * 10^6 \\ &v_1 / 0.001 - 1 \le 0 \\ &\sigma_i / 160e^6 - 1 \le 0; i = 1...4 \\ &-\sigma_i / 80e^6 - 1 \le 0; i = 1...4 \\ &-A_i / 10^{-8} - 1 \le 0; i = 1...4 \end{aligned} \tag{8.10}$$

The problem may now be solved in Matlab via the following script (exploiting the truss analysis code). The linear constraints on area may be imposed in multiple ways (explore these).

```
Truss = PreprocessTruss(Truss);
Truss.SigmaTensionMax = 160e6;
Truss.SigmaCompressionMax = 80e6;
Truss.DeflectionMax = 0.01;
Truss.AMin = 1e-8;
Truss.VolumeScale = 1e6;
options = optimset('LargeScale','off','Diagnostics','Off');
N =length(Truss.Area);
xBar0 = Truss.Area;
LB = Truss.AMin*ones(1,N);
[FinalArea,Vol,Flag,output,lambda] = fmincon(@(x)fObj1(x,Truss),xBar0,...
    [],[],[],[],LB,[],@(x)fCon1(x,Truss),options);
```

Script 8-2: Optimizing a truss structure.

8.4 Truss Sensitivity/Gradients

We now focus computing the gradients of the displacements, stresses, etc with respect to the design variables. Consider the displacement equation:

$$Ku = f \tag{8.11}$$

Consider now the derivative of the above equation with respect to a cross-sectional area A_j :

$$\frac{\partial(Ku)}{\partial A_j} = \frac{\partial(f)}{\partial A_j} \tag{8.12}$$

Since the right hand side is independent of the cross-sectional area, we have:

$$\frac{\partial K}{\partial A_j}u + K\frac{\partial u}{\partial A_j} = 0 \tag{8.13}$$

i.e.,

$$K\frac{\partial u}{\partial A_j} = -\frac{\partial K}{\partial A_j}u \tag{8.14}$$

However, from Equation (8.2), we have:

$$\frac{\partial K}{\partial A_j} = \sum_{i=1,2\ldots}\frac{\partial k_i}{\partial A_j} \tag{8.15}$$

From Equation (8.3), we have:

$$\begin{aligned} &\frac{\partial k_i}{\partial A_j} = \frac{E}{l_i}\begin{bmatrix} \quad \end{bmatrix}; i = j \\ &\frac{\partial k_i}{\partial A_j} = 0; i \neq j \end{aligned} \tag{8.16}$$

One can now assemble the sensitivity matrix, and solve for the displacement sensitivity in Equation (8.14). The gradients can be used in the optimization process.

```
Truss = PreprocessTruss(Truss);
Truss.SigmaTensionMax = 160e6;
Truss.SigmaCompressionMax = 80e6;
Truss.DeflectionMax = 0.01;
Truss.AMin = 1e-8;
Truss.VolumeScale = 1e6;
options = optimset('LargeScale','off','Diagnostics','Off',...
                   'GradObj','On','GradConstr','On');
N =length(Truss.Area);
xBar0 = Truss.Area;
LB = Truss.AMin*ones(1,N);
AInEq = -eye(N,N)/Truss.AMin; % Can use these instead of LB
BInEq = -ones(N,1); % Can use these instead of LB
[FinalArea,Vol,Flag,output,lambda] = ...
fmincon(@(x)fObjWithGradients1(x,Truss),xBar0,...
    [],[],[],[],LB,[],@(x)fConWithGradients1(x,Truss),options);
```

Script 8-3: Optimizing a truss structure with gradients.

8.5 Continuum Tensile Bars

Thus far, we have considered tensile bars that are of uniform cross-section, uniform material, and subject to a single tip load. Here we consider continuum tensile bars that are of non-uniform cross-section $A(x)$, non-uniform material $E(x)$, subject to a body force $b(x)$ (units are force per unit volume). Our objective is to find the displacement of the bar $u(x)$, and relate its computation to a minimization problem.

8.5.1 Force Balance Strategy

First, we use a classic force balance strategy to 'solve' for $u(x)$. Observe that, by considering a thin slice Δx of the bar, we arrive at the force balance equation:

$$(\sigma + \Delta\sigma)\big(A(x) + \Delta A\big) - \sigma A(x) + b(x)A(x)\Delta x = 0 \tag{8.17}$$

i.e.,

$$\sigma\Delta A + \Delta\sigma A(x) + b(x)A(x)\Delta x = 0 \tag{8.18}$$

in the limit

$$\sigma\frac{dA}{dx} + \frac{d\sigma}{dx}A(x) + b(x)A(x) = 0 \tag{8.19}$$

i.e.,

$$\frac{d\left(A\sigma\right)}{dx} + b(x)A(x) = 0 \tag{8.20}$$

Exploiting the constitutive relationship $\sigma = E\varepsilon$, we have:

$$\frac{d\left(EA\varepsilon\right)}{dx} + b(x)A(x) = 0 \tag{8.21}$$

Finally, since $\varepsilon = du/dx$, we have:

$$\frac{d}{dx}\left(EA\frac{du}{dx}\right) + b(x)A(x) = 0 \tag{8.22}$$

Equation (8.22) is 2nd order differential equation governing $u(x)$. To solve for $u(x)$, one must specify 2 boundary conditions; these are:

$$u(0) = 0$$
$$\sigma|_{x=l} = 0 \tag{8.23}$$

i.e., $u(x)$ must satisfy:

$$\frac{d}{dx}\left(EA\frac{du}{dx}\right) + b(x)A(x) = 0$$
$$u(0) = 0; \left.\frac{du}{dx}\right|_{x=l} = 0 \tag{8.24}$$

Example: Suppose $A(x) = A_0$, $E(x) = E_0$, $b(x) = b_0$, we have:

$$E_0 A_0 \frac{d}{dx}\left(\frac{du}{dx}\right) + b_0 A_0 = 0$$
$$u(0) = 0; \left.\frac{du}{dx}\right|_{x=l} = 0 \tag{8.25}$$

i.e.,

$$\frac{d^2u}{dx^2} = -b_0 / E_0$$
$$u(0) = 0; \left.\frac{du}{dx}\right|_{x=l} = 0 \tag{8.26}$$

whose solution, as one can verify, is given by:

$$u(x) = -\frac{b_0 x^2}{2E_0} + \frac{b_0 l x}{E_0} \tag{8.27}$$

♦

Although the differential formulation in Equation (8.24) is insightful and provides 'exact' solutions to simple problems, it is less useful when approximate solutions are needed (for non-trivial $b(x), A(x)$ & $E(x)$). We therefore turn to an alternate minimization of energy formulation.

8.5.2 Minimization of Potential Energy Strategy

Recall that "*structural systems when subject to external forces come to rest in a state where their potential energy is a minimum*". We exploit this principle once again to arrive at a formulation for $u(x)$. Recall that the elastic energy within a continuum is given by:

$$U = \frac{1}{2}\int_V \varepsilon\sigma dV \tag{8.28}$$

For the continuum bar, this reduces to:

$$U = \frac{1}{2}\int_0^l \varepsilon\sigma A\,dx = \frac{1}{2}\int_0^l \frac{du}{dx} E \frac{du}{dx} A\,dx = \frac{1}{2}\int_0^l EA\left(\frac{du}{dx}\right)^2 dx \tag{8.29}$$

The external work done is:

$$W = \int_0^l buA\,dx \tag{8.30}$$

Thus, the potential energy is given by:

$$\Pi = \frac{1}{2}\int_0^l EA\left(\frac{du}{dx}\right)^2 dx - \int_0^l buA\,dx \tag{8.31}$$

The problem of finding $u(x)$ may now be posed as a minimization problem:

$$\begin{gathered}\underset{u(x)}{Min}\left[\frac{1}{2}\int_0^l EA\left(\frac{du}{dx}\right)^2 dx - \int_0^l buA\,dx\right] \\ u(0) = 0\end{gathered} \tag{8.32}$$

Observe that unlike other minimization problems considered so far, the design variable $u(x)$ is a continuous function! However, one can reduce the problem to a continuous variable problem as illustrated below.

Example: Suppose $A(x) = A_0$, $E(x) = E_0$, $b(x) = b_0$, we have:

$$\begin{gathered}\underset{u(x)}{Min}\left[\frac{E_0A_0}{2}\int_0^l \left(\frac{du}{dx}\right)^2 dx - b_0A_0\int_0^l u\,dx\right] \\ u(0) = 0\end{gathered} \tag{8.33}$$

Now, to solve for $u(x)$, let us assume that it is quadratic, i.e.,

$$u(x) \approx a_0 + a_1x + a_2x^2 \tag{8.34}$$

But since we require $u(0) = 0$, we have:

$$u(x) \approx a_1x + a_2x^2 \tag{8.35}$$

Thus, the minimization problem reduces to:

$$\underset{a_1,a_2}{Min}\left[\frac{E_0A_0}{2}\int_0^l (a_1 + 2a_2x)^2 dx - b_0A_0\int_0^l \left(a_1x + a_2x^2\right)dx\right] \tag{8.36}$$

i.e.,

$$\underset{a_1,a_2}{Min}\left[\frac{E_0A_0}{2}\int_0^l \left(a_1^2 + 4a_2^2x^2 + 4a_1a_2x\right)dx - b_0A_0\int_0^l \left(a_1x + a_2x^2\right)dx\right] \tag{8.37}$$

Carrying out the symbolic integration:

$$\underset{a_1,a_2}{Min}\left[\frac{E_0A_0}{2}\left(a_1^2l + 4a_2^2l^3/3 + 2a_1a_2l^2\right) - b_0A_0\left(a_1l^2/2 + a_2l^3/3\right)\right] \tag{8.38}$$

Observe that the objective is now a quadratic function of a_1 & a_2. Indeed, we have:

$$\underset{a_1,a_2}{Min}\frac{E_0A_0}{4}\begin{Bmatrix}a_1 & a_2\end{Bmatrix}\begin{bmatrix}l & 2l^2 \\ 2l^2 & 4l^3/3\end{bmatrix}\begin{Bmatrix}a_1 \\ a_2\end{Bmatrix} - b_0A\begin{Bmatrix}a_1 & a_2\end{Bmatrix}\begin{Bmatrix}l^2/2 \\ l^3/3\end{Bmatrix} \tag{8.39}$$

whose solution is given by:

$$\begin{Bmatrix}a_1 \\ a_2\end{Bmatrix} = \frac{4b_0}{E_0}\begin{bmatrix}l & 2l^2 \\ 2l^2 & 4l^3/3\end{bmatrix}^{-1}\begin{Bmatrix}l^2/2 \\ l^3/3\end{Bmatrix} \tag{8.40}$$

i.e.,

$$\begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} = \frac{b_0}{E_0} \begin{Bmatrix} l \\ -\frac{1}{2} \end{Bmatrix} \tag{8.41}$$

Substituting in Equation (8.35):

$$u(x) = \frac{b_0 l x}{E_0} - \frac{b_0 x^2}{2E_0} \tag{8.42}$$

This is precisely the solution in Equation (8.27).

♦

From a computational perspective, the potential energy formulation is much more powerful than the force–balance (differential equation) formulation in that one can always obtain approximate numerical solutions to fairly complex problems.

Example: Suppose $A(x) = A_0(1+x)$, $E(x) = E_0$, $b(x) = b_0/(1+x^2)$, and let the length of the bar $l = 1$. Thus, we have:

$$\underset{u(x)}{Min}\left[\frac{1}{2}\int_0^1 E_0 A_0 e^{-x}\left(\frac{du}{dx}\right)^2 dx - \int_0^1 \frac{b_0 u A}{1+x^2}dx\right] \tag{8.43}$$

$$u(0) = 0$$

As before, we shall assume that:

$$u(x) \approx a_1 x + a_2 x^2 \tag{8.44}$$

Thus, the minimization problem reduces to:

$$\underset{a_1,a_2}{Min}\left[\frac{E_0A_0}{2}\int_0^1 e^{-x}\left(a_1 + 2a_2 x\right)^2 dx - b_0A_0\int_0^1 \frac{\left(a_1x + a_2x^2\right)}{1+x^2}dx\right] \tag{8.45}$$

i.e.,

$$\underset{a_1,a_2}{Min}\left[\frac{E_0A_0}{2}\int_0^1 e^{-x}\left(a_1^2 + 4a_1a_2x + 4a_2^2x^2\right)dx - b_0A_0\int_0^1 \frac{\left(a_1x + a_2x^2\right)}{1+x^2}dx\right] \tag{8.46}$$

Instead of symbolic integration, we shall use Matlab 'quad' function to carry out numerical integration. For example:

$$\int_0^1 e^{-x} x dx \tag{8.47}$$

can be carried out in Maltlab via:

```
f = inline('exp(-x).*x');
quad(f,0,1)
```

Script 8-4: Optimizing a truss structure with gradients.

Leading to:

$$\int_0^1 e^{-x}dx \approx 0.6321; \int_0^1 e^{-x}xdx \approx 0.2642; \int_0^1 e^{-x}x^2dx \approx 0.1606 \tag{8.48}$$

and

$$\int_0^1 \frac{x}{1+x^2}\,dx \approx 0.3466; \int_0^1 \frac{x^2}{1+x^2}\,dx \approx 0.2146 \tag{8.49}$$

Thus

$$\underset{a_1,a_2}{Min}\left[\frac{E_0A_0}{2}\left(0.6321a_1^2 + 1.0568a_1a_2 + 0.6424a_2^2\right) - b_0A_0\left(a_1 0.3466 + a_2 0.2146\right)\right] \tag{8.50}$$

Once again the objective function is a quadratic function of a_1 & a_2. Indeed, we have:

$$\underset{a_1,a_2}{Min}\frac{E_0A_0}{4}\begin{Bmatrix} a_1 & a_2 \end{Bmatrix}\begin{bmatrix} 0.6321 & 1.0568 \\ 1.0568 & 0.6424 \end{bmatrix}\begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} - b_0A\begin{Bmatrix} a_1 & a_2 \end{Bmatrix}\begin{Bmatrix} 0.3466 \\ 0.2146 \end{Bmatrix} \tag{8.51}$$

whose solution is given by:

$$\begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} = \frac{4b_0}{E_0}\begin{bmatrix} 0.6321 & 1.0568 \\ 1.0568 & 0.6424 \end{bmatrix}^{-1}\begin{Bmatrix} 0.3466 \\ 0.2146 \end{Bmatrix} \tag{8.52}$$

i.e.,

$$\begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} = \frac{b_0}{E_0}\begin{Bmatrix} 0.0015 \\ 0.0811 \end{Bmatrix} \tag{8.53}$$

Substituting in Equation (8.35):

$$u(x) = 0.0015\frac{b_0x}{E_0} + 0.0811\frac{b_0x^2}{E_0} \tag{8.54}$$

♦

8.5.3 From Minimization Formulation to Differential Equation

Thus, for the same physical system, we have two different problem formulations: (a) the differential equation (or the force balance) formulation:

$$\begin{gathered}\frac{d}{dx}\left(EA\frac{du}{dx}\right) + b(x)A(x) = 0 \\ u(0) = 0; \left.\frac{du}{dx}\right|_{x=l} = 0\end{gathered} \tag{8.55}$$

and the minimization formulation:

$$\begin{gathered}\underset{u(x)}{Min}\left[\frac{1}{2}\int_0^l EA\left(\frac{du}{dx}\right)^2 dx - \int_0^l buA\,dx\right] \\ u(0) = 0\end{gathered} \tag{8.56}$$

Indeed, one can derive the differential equation from the minimization formulation by appealing to Euler-Lagrange theorem of calculus, which states that: If $u(x)$ is the solution to the problem:

$$\underset{u(x)}{Min}\,F\left(x, u, \frac{du}{dx}\right) \tag{8.57}$$

then $u(x)$ satisfies the differential equation given by:

$$\frac{\partial F}{\partial u} - \frac{d}{dx}\left(\frac{\partial F}{\partial u'}\right) = 0 \tag{8.58}$$

Equation (8.58) is called the Euler-Lagrange equation. Applying this theorem to Equation (8.56), we have:

$$F = \frac{1}{2} EA\left(\frac{du}{dx}\right)^2 - buA \tag{8.59}$$

Thus:

$$\frac{\partial F}{\partial u} = -bu \tag{8.60}$$

and

$$\begin{aligned} \frac{\partial F}{\partial u'} &= EA\left(\frac{du}{dx}\right) \\ \frac{d}{dx}\left(\frac{\partial F}{\partial u'}\right) &= \frac{d}{dx}\left(EA\frac{du}{dx}\right) \end{aligned} \tag{8.61}$$

i.e., the differential equation is given by:

$$-bu - \frac{d}{dx}\left(EA\frac{du}{dx}\right) = 0 \tag{8.62}$$

i.e.,

$$\frac{d}{dx}\left(EA\frac{du}{dx}\right) + bu = 0 \tag{8.63}$$

that is identical to Equation (8.55).

8.6 Optimization of Continuum Tensile Bars

Analogous to the optimization of constant cross-section tensile bars, we can now pose optimization of continuum design variables. For example:

Problem 1: Consider a tensile bar subject to a gravity load. The problem then is to find the optimal area $A(x); 0 \le x \le 1$ such that the tip displacement is minimized, with the following constraints: (1) the total volume is a prescribed value, and (2) the stress within the bar does not exceed a specified value.

Mathematically, the problem may be posed as:

$$\begin{aligned} &\underset{A(x)}{Min}\, u(1) \\ &\int_0^1 A\,dx = V_0 \\ &\sigma(x) \le \sigma_{\max} \\ &A(x) > A_{\min} \end{aligned} \tag{8.64}$$

9 References

1. Bronshtein, I.N., Semendyayev, K. A., *Handbook of Mathematics*. 1985, New York: Van Nostrand Reinhold.
2. Nocedal, J., Wright, S. J., *Numerical Optimization*. Springer Series in Operations Research. 1999: Springer.