CHAPTER 7
CONSTRAINED OPTIMIZATION 2: SQP AND GRG

1 Introduction

In the previous chapter we examined the necessary and sufficient conditions for a constrained optimum. We did not, however, discuss any algorithms for constrained optimization. That is the purpose of this chapter.

In an oft referenced study done in 1980[1], dozens of nonlinear algorithms were tested on roughly 100 different nonlinear problems. The top-ranked algorithm was SQP. Of the five top algorithms, two were SQP and three were GRG. Although many different algorithms have been proposed, based on these results, we will study only these two.

SQP works by solving for where the KT equations are satisfied. SQP is a very efficient algorithm in terms of the number of function calls needed to get to the optimum. It converges to the optimum by simultaneously improving the objective and tightening feasibility of the constraints. Only the optimal design is guaranteed to be feasible; intermediate designs may be infeasible.

The GRG algorithm works by computing search directions which improve the objective and satisfy the constraints, and then conducting line searches in a very similar fashion to the algorithms we studied in Chapter 3. GRG requires more function evaluations than SQP, but it has the desirable property that it stays feasible once a feasible point is found. If the optimization process is halted before the optimum is reached, the designer is guaranteed to have in hand a better design than the starting design. GRG also appears to be more robust (able to solve a wider variety of problems) than SQP, so for engineering problems it is often the algorithm tried first.

2 The Sequential Quadratic Programming (SQP) Algorithm

The SQP algorithm was developed in the early 1980's by M. J. D. Powell, a mathematician at Cambridge University. Before we begin describing this algorithm, we need to present some background information.

2.1 The Newton Raphson Method for Solving Nonlinear Equations

If we were to sum up how the SQP methods works in one sentence it would be: the SQP algorithm applies the Newton-Raphson method to solve the Kuhn-Tucker equations. Or, in short, SQP does N-R on the K-T! Thus we will begin by reviewing how the Newton Raphson method works.

[1] Schittkowski, K., "Nonlinear Programming codes: Information, Tests, Performance," *Lecture Notes in Economics and Mathematical Systems,* vol. 183, Springer-Verlag, New York, 1980.

2.1.1 One equation with One Unknown

The N-R method is used to find the solution to sets of nonlinear equations. For example, suppose we wish to find the solution to the equation:

$$x + 2 = e^x$$

We cannot solve for x directly. The N-R method solves the equation in an iterative fashion based on results derived from the Taylor expansion.

First, the equation is rewritten in the form,

$$x + 2 - e^x = 0 \tag{7.1}$$

We then provide a starting estimate of the value of x that solves the equation. This point becomes the point of expansion for a Taylor series:

$$F \approx F^0 + \frac{dF^0}{dx}\left(x - x^0\right) \tag{7.2}$$

(For reasons that will become apparent later, we will use F instead of f for our functions here.) We would like to drive the value of the function to zero:

$$0 = F^0 + \frac{dF^0}{dx}\left(x - x^0\right) \tag{7.3}$$

If we denote $\Delta x = x - x^0$, and solve for Δx in (7.3):

$$\Delta x = \frac{-F^0}{dF/dx} \tag{7.4}$$

We then add Δx to x^0 to obtain a new guess for the value of x that satisfies the equation, obtain the derivative there, get the new function value, and iterate until the function value or *residual*, goes to zero. The process is illustrated in Fig. 7.1 for the example given in (7.1), with a starting guess $x = 2.0$.

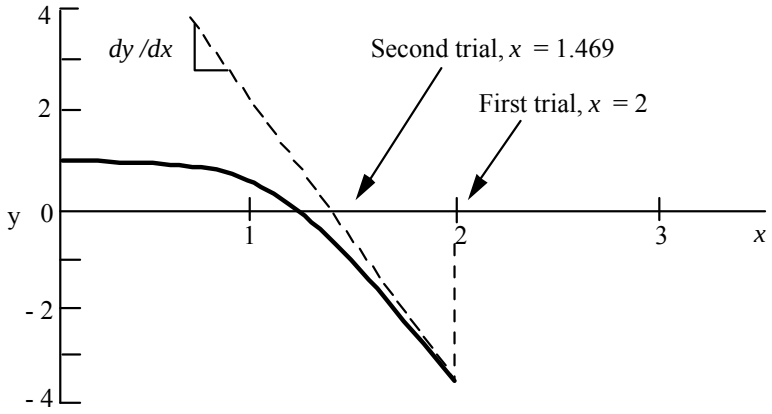

Fig. 7.1 Newton Raphson method on (7.1)

Numerical results are:

K	x	f(x)	df/dx
1	2	–3.389056	–6.389056
2	1.469553	–0.877738	–3.347291
3	1.20732948	–0.13721157	–2.34454106
4	1.14880563	–0.00561748	–2.15442311
5	1.146198212	–0.000010714	–2.146208926
6	1.1461932206	–0.00000000004	–2.1461932254

For simple roots, N-R has *second order convergence*. This means that the number of significant figures in the solution roughly doubles at each iteration. We can see this in the above table, where the value of *x* at iteration 2 has one significant figure (1); at iteration 3 it has one (1); at iteration 4 it has three (1.14); at iteration 5 it has six (1.14619), and so on. We also see that the error in the residual, as indicated by the number of zeros after the decimal point, also decreases in this fashion, i.e., the number of zeros roughly doubles at each iteration.

2.1.2 Multiple Equations with Multiple Unknowns

The N-R method is easily extended to solve *n* equation in *n* unknowns. Writing the Taylor series for the equations in vector form:

$$0 = F_1^0 + \left(\nabla F_1^0\right)^{\mathrm{T}} \Delta\mathbf{x}$$

$$0 = F_2^0 + \left(\nabla F_2^0\right)^{\mathrm{T}} \Delta\mathbf{x}$$

$$\cdots \quad \cdots$$

$$0 = F_n^0 + \left(\nabla F_n^0\right)^{\mathrm{T}} \Delta\mathbf{x}$$

We can rewrite these relationships in matrix form:

$$\begin{bmatrix} \left(\nabla F_1^0\right)^{\mathrm{T}} \\ \left(\nabla F_2^0\right)^{\mathrm{T}} \\ \cdots \\ \left(\nabla F_n^0\right)^{\mathrm{T}} \end{bmatrix} \Delta\mathbf{x} = \begin{bmatrix} -F_1^0 \\ -F_2^0 \\ \cdots \\ -F_n^0 \end{bmatrix} \tag{7.5}$$

For 2 X 2 System,

$$\begin{bmatrix} \dfrac{\partial F_1}{\partial x_1} & \dfrac{\partial F_1}{\partial x_2} \\ \dfrac{\partial F_2}{\partial x_1} & \dfrac{\partial F_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} -F_1 \\ -F_2 \end{bmatrix} \tag{7.6}$$

In (7.5) we will denote the vector of residuals as $\mathbf{F}^0$. (This violates our notation convention that bold caps represent matrices, not vectors. Just remember that **F** is a vector, not a matrix.) We will denote the matrix of coefficients as **G.** Equation (7.5) can then be written,

$$\mathbf{G}\Delta\mathbf{x} = -\mathbf{F} \tag{7.7}$$

The solution is obviously

$$\Delta\mathbf{x} = -\left(\mathbf{G}^{-1}\right)\mathbf{F} \tag{7.8}$$

2.1.3 Using the N-R method to Solve the Necessary Conditions

In this section we will make a very important connection—we will apply N-R to solve the necessary conditions. Consider for example, a very simple case—an unconstrained problem in two variables. We know the necessary conditions are,

$$\begin{aligned} \frac{\partial f}{\partial x_1} &= 0 \\ \frac{\partial f}{\partial x_2} &= 0 \end{aligned} \tag{7.9}$$

Now suppose we wish to solve these equations using N-R, that is we wish to find **x*** to drive the partial derivatives of *f* to zero. In terms of notation and discussion this gets a little tricky because the N-R method involves taking derivatives of the equations to be solved, and the equations we wish to solve are composed of derivatives. So when we substitute (7.9) into the N-R method, we end up with *second* derivatives.

For example, if we set $F_1 = \dfrac{\partial f}{\partial x_1}$ and $F_2 = \dfrac{\partial f}{\partial x_2}$. Then we can write (7.6) as,

$$\begin{bmatrix} \frac{\partial}{\partial x_1}\left(\frac{\partial f}{\partial x_1}\right) & \frac{\partial}{\partial x_2}\left(\frac{\partial f}{\partial x_1}\right) \\ \frac{\partial}{\partial x_1}\left(\frac{\partial f}{\partial x_2}\right) & \frac{\partial}{\partial x_2}\left(\frac{\partial f}{\partial x_2}\right) \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} -\left(\frac{\partial f}{\partial x_1}\right) \\ -\left(\frac{\partial f}{\partial x_2}\right) \end{bmatrix} \tag{7.10}$$

or,

$$\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_2 \partial x_1} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} -\frac{\partial f}{\partial x_1} \\ -\frac{\partial f}{\partial x_2} \end{bmatrix} \tag{7.11}$$

which should be familiar from Chapter 3, because (7.11) can be written in vector form as,

$$\mathbf{H}\Delta\mathbf{x} = -\nabla f \tag{7.12}$$

and the solution is,

$$\Delta\mathbf{x} = -\left(\mathbf{H}^{-1}\right)\nabla f \tag{7.13}$$

We recognize (7.12-7.13) as Newton's method for solving for an unconstrained optimum. Thus we have the important result that *Newton's method is the same as applying N-R on the necessary conditions for an unconstrained problem*. From the properties of the N-R method, we know that if Newton's method converges (and recall that it doesn't always converge), it will do so with second order convergence—which is very fast.

Now just to indicate where we are heading, after we introduce the SQP approximation, the next step is to show that the SQP method is the same as doing N-R on the necessary conditions *for a constrained problem* (with some tweaks to make it efficient and to ensure it converges).

2.2 Constrained Optimization: Equality Constraints

2.2.1 Problem Definition

We will start with a problem which only has equality constraints. We recall that when we only have equality constraints, we do not have to worry about complementary slackness which makes things simpler. So the problem we will focus on is,

$$\text{Min} \quad f(\mathbf{x}) \tag{7.14}$$

$$\text{st.} \quad g_i(\mathbf{x}) - b_i = 0 \quad i = 1, 2, \ldots, m \tag{7.15}$$

The necessary conditions for a constrained optimal solution are:

$$\nabla f - \sum_{i=1}^{m} \lambda_i \nabla g_i = \mathbf{0} \tag{7.16}$$

$$g_i - b_i = 0 \quad i = 1, \ldots, m \tag{7.17}$$

2.2.2 The SQP Approximation

As we have previously mentioned in Chapter 6, a problem with a quadratic objective and linear constraints is known as a *quadratic programming problem*. These problems have a special name because the K-T equations are linear and are easily solved. We will make a quadratic programming approximation at the point $\mathbf{x^0}$ to the problem given by (7.14-7.15)

$$f_a = f^0 + \left(\nabla f^0\right)^{\mathrm{T}} \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^{\mathrm{T}} \nabla_x^2 \mathrm{L}^0 \Delta\mathbf{x} \tag{7.18}$$

$$g_{i,a} = g_i^0 + \left(\nabla g_i^0\right)^{\mathrm{T}} \Delta\mathbf{x} = b_i \quad i = 1, \ldots, m \tag{7.19}$$

where the subscript *a* is used in f_a to indicate the approximation. Close examination of (7.18) shows something unexpected. Instead of $\nabla^2 f$ as we would normally have if we were doing a Taylor approximation of the objective, we have $\nabla_x^2 L$, the Hessian of the Lagrangian function with respect to x. Why is this case? It is directly tied to applying N-R on the K-T, as we will presently show. For now we will just accept that the objective uses the Hessian of the Lagrangian instead of the Hessian of the objective.

We will solve the QP approximation, (7.18-7.19), by solving the K-T equations for this problem, which, as mentioned, are linear. These equations are given by,

$$\nabla f_a - \sum_{i=1}^{m} \lambda_i \nabla g_{i,a} = \mathbf{0} \tag{7.20}$$

$$g_{i,a} - b_i = 0 \quad \textit{for } i = 1, \ldots, m \tag{7.21}$$

Since, for example, from (7.18),

$$\nabla f_a = \nabla f^0 + \nabla_x^2 \mathrm{L}^0 \Delta\mathbf{x}$$

we can also write these equations in terms of the original problem,

$$\nabla f^0 + \nabla_x^2 \mathrm{L}^0 \Delta\mathbf{x} - \sum_{i=1}^{m} \lambda_i \nabla g_i^0 = \mathbf{0} \tag{7.22}$$

$$g_i^0 + \left(\nabla g_i^0\right)^{\mathrm{T}} \Delta\mathbf{x} - b_i = 0 \quad \textit{for } i = 1, \ldots, m \tag{7.23}$$

These are a linear set of equations we can readily solve, as shown in the example in the next section. For Section 2.2.4, we will want to write these equations even more concisely. If we define the following matrices and vectors,

$$\mathbf{J}^0 = \begin{bmatrix} \left(\nabla g_1^0\right)^{\mathrm{T}} \\ \left(\nabla g_2^0\right)^{\mathrm{T}} \\ \vdots \\ \left(\nabla g_m^0\right)^{\mathrm{T}} \end{bmatrix} \qquad \left(\mathbf{J}^0\right)^T = \begin{bmatrix} \nabla g_1^0, & \nabla g_2^0, & \cdots & \nabla g_m^0 \end{bmatrix}$$

$$\mathbf{g}^0 = \begin{bmatrix} g_1^0 \\ g_2^0 \\ \vdots \\ g_m^0 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \qquad \nabla_x^2 \mathrm{L}^0 = \nabla^2 f^0 - \sum_{i=1}^{m} \lambda_i \nabla^2 g_i^0$$

We can write (7.22-7.23) as,

$$\nabla_x^2 \mathrm{L}^0 \Delta\mathbf{x} + \left(-\mathbf{J}^0\right)^{\mathrm{T}} \boldsymbol{\lambda} = -\nabla f^0 \tag{7.24}$$

$$\mathbf{J}^0 \Delta\mathbf{x} = -\left(\mathbf{g}^0 - \mathbf{b}\right) \tag{7.25}$$

Again, to emphasize, this set of equations represents the solution to (7.18-7.19).

2.2.3 Example 1: Solving the SQP Approximation

Suppose we have as our approximation the following,

$$\begin{aligned} f_a &= 3 + \begin{bmatrix} 3 & 2 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \Delta x_1 & \Delta x_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} \\ g_a &= 5 + \begin{bmatrix} 1 & 3 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = 0 \end{aligned} \tag{7.26}$$

We can write out the K-T equations for this approximation as,

$$\begin{aligned} \nabla f_a &= \begin{bmatrix} 3 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} - \lambda \begin{bmatrix} 1 \\ 3 \end{bmatrix} = 0 \\ g_a &= 5 + \begin{bmatrix} 1 & 3 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = 0 \end{aligned} \tag{7.27}$$

We can rewrite these equations in matrix form as,

$$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -3 \\ 1 & 3 & 0 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} -3 \\ -2 \\ -5 \end{bmatrix} \tag{7.28}$$

The solution is,

$$\begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} -2.6 \\ -0.8 \\ 0.4 \end{bmatrix} \tag{7.29}$$

Observations: This calculation represents the main step in an iteration of the SQP algorithm which solves a *sequence* of *quadratic programs*. If we wanted to continue, we would add $\Delta\mathbf{x}$ to our current $\mathbf{x}$, update the Lagrangian Hessian, make a new approximation, solve for that solution, and continue iterating in this fashion.

If we ever reach a point where $\Delta\mathbf{x}$ goes to zero as we solve for the optimum of the approximation, *the original K-T equations are satisfied.* We can see this by examining (7.22-7.23). If $\Delta\mathbf{x}$ is zero, we have,

$$\nabla f + \underbrace{\nabla_x^2 \mathrm{L} \Delta\mathbf{x}}_{=0} - \sum_{i=1}^{m} \lambda_i^* \nabla g_i = \mathbf{0} \tag{7.30}$$

$$g_i + \underbrace{\left(\nabla g_i\right)^{\mathrm{T}} \Delta\mathbf{x}}_{=0} - b_i = 0 \quad \textit{for } i = 1, \ldots, m \tag{7.31}$$

which then match (7.16-7.17).

2.2.4 N-R on the K-T Equations for Problems with Equality Constraints

In this section we wish to look at applying the N-R method to the original K-T equations. The K-T equations for a problem with equality constraints only, as given by (7.16-7.17), are,

$$\nabla f - \sum_{i=1}^{m} \lambda_i \nabla g_i = \mathbf{0} \tag{7.32}$$

$$g_i - b_i = 0 \quad i = 1, \ldots, m \tag{7.33}$$

Now suppose we wish to solve these equations using the N-R method. To implement N-R we would have,

$$\begin{bmatrix} \left(\nabla_x F_1\right)^T & \left(\nabla_\lambda F_1\right)^T \\ \vdots & \vdots \\ \left(\nabla_x F_n\right)^T & \left(\nabla_\lambda F_n\right)^T \\ \left(\nabla_x g_1\right)^T & \left(\nabla_\lambda g_1\right)^T \\ \vdots & \vdots \\ \left(\nabla_x g_m\right)^T & \left(\nabla_\lambda g_m\right)^T \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -F_1 \\ \vdots \\ -F_n \\ -\left(g_1 - b_1\right) \\ \vdots \\ -\left(g_m - b_m\right) \end{bmatrix} \tag{7.34}$$

where F_1 for example, is given by,

$$F_1 = \frac{\partial f}{\partial x_1} - \sum_{i=1}^{m} \lambda_i \frac{\partial g_i}{\partial x_1} \tag{7.35}$$

If we substitute (7.35) into matrix (7.32), the first row becomes,

$$\left[\frac{\partial^2 f}{\partial x_1^2} - \sum_{i=1}^{m} \lambda_i \frac{\partial^2 g_i}{\partial x_1^2}\right], \left[\frac{\partial^2 f}{\partial x_2 \partial x_1} - \sum_{i=1}^{m} \lambda_i \frac{\partial^2 g_i}{\partial x_2 \partial x_1}\right], \ldots, \left[\frac{\partial^2 f}{\partial x_n \partial x_1} - \sum_{i=1}^{m} \lambda_i \frac{\partial^2 g_i}{\partial x_n \partial x_1}\right], \left[-\frac{\partial g_1}{\partial x_1}\right], \left[-\frac{\partial g_2}{\partial x_1}\right], \ldots, \left[-\frac{\partial g_m}{\partial x_1}\right]$$

Recalling,

$$\nabla_x^2 L = \nabla^2 f - \sum_{i=1}^{m} \lambda_i \nabla^2 g_i$$

And using the matrices we defined above,

$$\mathbf{J}^0 = \begin{bmatrix} \left(\nabla g_1^0\right)^{\mathrm{T}} \\ \left(\nabla g_2^0\right)^{\mathrm{T}} \\ \vdots \\ \left(\nabla g_m^0\right)^{\mathrm{T}} \end{bmatrix} \qquad \left(\mathbf{J}^0\right)^T = \begin{bmatrix} \nabla g_1^0, & \nabla g_2^0, & \cdots & \nabla g_m^0 \end{bmatrix}$$

$$\mathbf{g}^0 = \begin{bmatrix} g_1^0 \\ g_2^0 \\ \vdots \\ g_m^0 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \qquad \nabla_x^2 \mathrm{L}^0 = \nabla^2 f^0 - \sum_{i=1}^{m} \lambda_i \nabla^2 g_i^0$$

we can rewrite (7.34) as,

$$\begin{bmatrix} \nabla_x^2 \mathrm{L}^0 & \left(-\mathbf{J}^0\right)^T \\ \mathbf{J}^0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} - \mathbf{x}^0 \\ \boldsymbol{\lambda} - \boldsymbol{\lambda}^0 \end{bmatrix} = \begin{bmatrix} -\nabla f^0 + \left(\mathbf{J}^0\right)^T \boldsymbol{\lambda}^0 \\ -\left(\mathbf{g}^0 - \mathbf{b}\right) \end{bmatrix} \tag{7.36}$$

If we do the matrix multiplications we have

$$\begin{aligned} &\nabla_x^2 \mathrm{L}^0 \Delta\mathbf{x} + \left(-\mathbf{J}^0\right)^{\mathrm{T}} (\boldsymbol{\lambda} - \boldsymbol{\lambda}^0) = -\nabla f^0 + \left(\mathbf{J}^0\right)^T \boldsymbol{\lambda}^0 \\ &\mathbf{J}^0 \Delta\mathbf{x} = -\left(\mathbf{g}^0 - \mathbf{b}\right) \end{aligned} \tag{7.37}$$

and collecting terms,

$$\begin{aligned} &\nabla_x^2 \mathrm{L}^0 \Delta\mathbf{x} + \left(-\mathbf{J}^0\right)^{\mathrm{T}} \boldsymbol{\lambda} = -\nabla f^0 \\ &\mathbf{J}^0 \Delta\mathbf{x} = -\left(\mathbf{g}^0 - \mathbf{b}\right) \end{aligned} \tag{7.38}$$

which equations are the same as (7.24-7.25). Thus we see that *doing a N-R iteration on the K-T equations is the same as solving for the optimum of the QP approximation.* This is the reason we use the Hessian of the Lagrangian function rather than the Hessian of the objective in the approximation.

2.3 Constrained Optimization: Inequality and Equality Constraints

In the previous section we considered equality constraints only. We need to extend these results to the general case. We will state this problem as

$$\begin{aligned} \text{Min} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) - b_i \geq 0 \quad i = 1, \ldots, k \\ & g_i(\mathbf{x}) - b_i = 0 \quad i = k+1, \ldots, m \end{aligned} \tag{7.39}$$

The quadratic approximation at point $\mathbf{x}^{\mathrm{o}}$ is:

$$\begin{aligned} \text{Min} \quad & f_a = f^0 + \left(\nabla f^0\right)^{\mathrm{T}} \Delta\mathbf{x} + \frac{1}{2}\left(\Delta\mathbf{x}\right)^{\mathrm{T}} \nabla_x^2 \mathrm{L}^0 \Delta\mathbf{x} \\ \text{s.t.} \quad & g_{i,a}: \quad g_i^0 + \left(\nabla g_i^0\right)^T \Delta\mathbf{x} \geq b_i \quad i = 1, 2, \ldots, k \\ & \qquad\qquad g_i^0 + \left(\nabla g_i^0\right)^T \Delta\mathbf{x} = b_i \quad i = k+1, \ldots, m \end{aligned} \tag{7.40}$$

Notice that the approximations are a function only of $\Delta\mathbf{x}$**.** All gradients and the Lagrangian hessian in (7.40) are evaluated at the point of expansion and so represent known quantities.

In the article where Powell describes this algorithm,[2] he makes a significant statement at this point. Quoting, "The extension of the Newton iteration to take account of inequality constraints on the variables arises from the fact that the value of $\Delta\mathbf{x}$ that solves (7.39) can

[2] Powell, M.J.D., "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," *Numerical Analysis, Dundee 1977,* Lecture Notes in Mathematics no. 630, Springer-Verlag, New York, 1978

also be found by solving a quadratic programming problem. Specifically, $\Delta\mathbf{x}$ is the value that makes the quadratic function in (7.40) stationary."

Further, the value of $\boldsymbol{\lambda}$ for the K-T conditions is equal to the vector of Lagrange multipliers of the quadratic programming problem. Thus solving the quadratic objective and linear constraints in (7.40) is the *same as solving the N-R iteration on the original K-T equations.*

The main difficulty in extending SQP to the general problem has to do the with the complementary slackness condition. This equation is non-linear, and so makes the QP problem nonlinear. We recall that complementary slackness basically enforces that either a constraint is binding or the associated Lagrange multiplier is zero. Thus we can incorporate this condition if we can develop a method to *determine which inequality constraints are binding at the optimum*. An example of a modern solution technique is given by Goldfarb and Idnani.[3] This algorithm starts out by solving for the unconstrained optimum to the problem and evaluating which constraints are violated. It then moves to add in these constraints until it is at the optimum. Thus it tends to drive to the optimum from infeasible space.

There are other important details to develop a realistic, efficient SQP algorithm. For example, the QP approximation involves the Lagrangian hessian matrix, which involves second derivatives. As you might expect, we don't evaluate the Hessian directly but approximate it using a quasi-Newton update, such as the BFGS update.

Recall that updates use differences in **x** and differences in gradients to estimate second derivatives. To estimate $\nabla_x^2 \mathrm{L}$ we will need to use differences in the gradient of the Lagrangian function,

$$\nabla_x \mathrm{L} = \nabla f - \sum_{i=1}^{m} \lambda_i \nabla g_i$$

Note that to evaluate this gradient we need values for λ_i. We will get these from our solution to the QP problem. Since our update stays positive definite, we don't have to worry about the method diverging, like Newton's method does for unconstrained problems.

2.4 Comments on the SQP Algorithm

The SQP algorithm has the following characteristics,

- The algorithm is very fast. It is the most efficient optimization algorithm available today.
- Because it does not rely on a traditional line search, it is often more accurate in identifying an optimum.

[3] Goldfarb, D., and A. Idnani, "A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programs," *Math. Programming,* v. 27, 1983, p.1-33.

- The efficiency of the algorithm is partly because it does not enforce feasibility of the constraints at each step. Rather it gradually enforces feasibility as part of the K-T conditions. It is only guaranteed to be feasible at the optimum.

Relative to engineering problems, there are some drawbacks:

- Because it can go infeasible during optimization—sometimes by relatively large amounts—it can crash engineering models.
- It is more sensitive to numerical noise and/or error in derivatives than GRG.
- If we terminate the optimization process before the optimum is reached, SQP does not guarantee that we will have in-hand a better design than we started with. GRG does guarantee this.

2.5 Summary of Steps for SQP Algorithm

1. Make a QP approximation to the original problem. For the first iteration, use a Lagrangian Hessian equal to the identity matrix.

2. Solve for the optimum to the QP problem. As part of this solution, values for the Lagrange multipliers are obtained.

3. Execute a simple line search by first stepping to the optimum of the QP problem. So the initial step is $\Delta\mathbf{x}$, and $\mathbf{x}^{new} = \mathbf{x}^{old} + \Delta\mathbf{x}$. See if at this point a penalty function, composed of the values of the objective and violated constraints, is reduced. If not, cut back the step size until the penalty function is reduced. The penalty function is given by $P = f + \sum_{i=1}^{vio} \lambda_i |g_i|$ where the summation is done over the set of violated constraints, and the absolute values of the constraints are taken. The Lagrange multipliers act as scaling or weighting factors between the objective and violated constraints.

4. Evaluate the Lagrangian gradient at the new point. Calculate the difference in $\mathbf{x}$ and in the Lagrangrian gradient, $\boldsymbol{\gamma}$. Update the Lagrangian Hessian using the BFGS update.

5. Return to Step 1 until $\Delta\mathbf{x}$ is sufficiently small. When $\Delta\mathbf{x}$ approaches zero, the K-T conditions for the original problem are satisfied.

2.6 Example of SQP Algorithm

Find the optimum to the problem,

$$\begin{aligned} \text{Min} \quad & f(\mathbf{x}) = x_1^4 - 2x_2x_1^2 + x_2^2 + x_1^2 - 2x_1 + 5 \\ \text{s.t.} \quad & g(\mathbf{x}) = -(x_1 + 0.25)^2 + 0.75x_2 \geq 0 \end{aligned}$$

starting from the point $[-1,4]$. A contour plot of the problem is shown in Fig. 7.2. This problem is interesting for several reasons: the objective is quite eccentric at the optimum, the algorithm starts at point where the search direction is pointing away from the optimum, and the constraint boundary at the starting point has a slope opposite to that at the optimum.

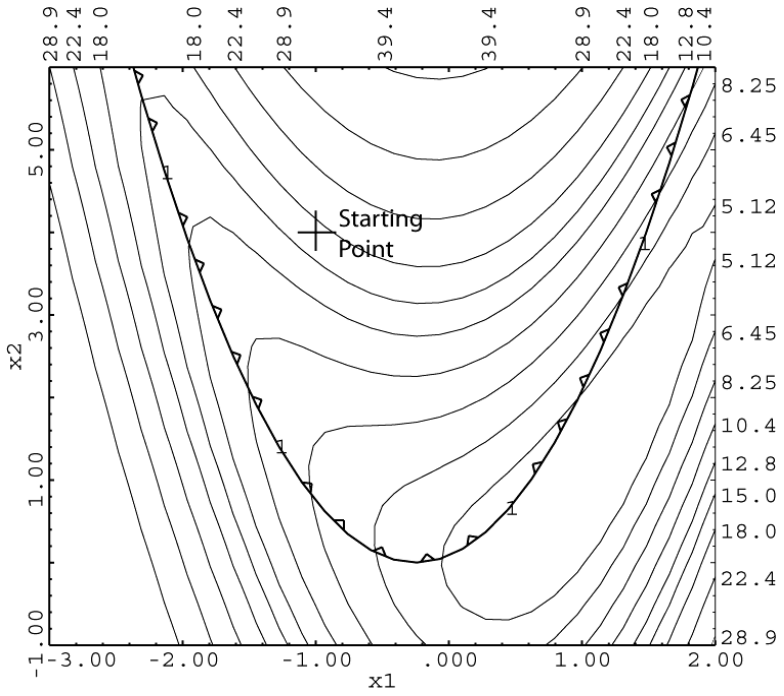

Fig. 7.2. Contour plot of example problem for SQP algorithm.

Iteration 1

We calculate the gradients, etc. at the beginning point. The Lagrangian Hessian is initialized to the identity matrix.

$$\text{At } \left(\mathbf{x}^0\right)^{\mathrm{T}} = [-1,4],\ f^0 = 17,\ \left(\nabla f^0\right)^T = [8,6],\ \nabla^2 \mathrm{L}^0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$g^0 = 2.4375,\ \left(\nabla g^0\right)^T = [1.5,\ 0.75]$$

Based on these values, we create the first approximation,

$$f_a = 17.0 + \begin{bmatrix} 8 & 6 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \Delta x_1 & \Delta x_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix}$$

$$g_a = 2.4375 + \begin{bmatrix} 1.5 & 0.75 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} \geq 0$$

We will assume the constraint is binding. Then the K-T conditions for the optimum of the approximation are given by the following equations:

$$\nabla f_a - \lambda \nabla g_a = 0$$
$$g_a = 0$$

These equations can be written as,

$$8+\Delta x_1-\lambda(1.5)=0$$
$$6+\Delta x_2-\lambda(0.75)=0$$
$$2.4375+1.5\Delta x_1+0.75\Delta x_2=0$$

The solution to this set of equations is $\Delta x_1=-0.5,\ \Delta x_2=-2.25,\ \lambda=5.00$

The proposed step is, $\mathbf{x}^1=\mathbf{x}^0+\Delta\mathbf{x}=\begin{bmatrix}-1\\4\end{bmatrix}+\begin{bmatrix}-0.5\\-2.25\end{bmatrix}=\begin{bmatrix}-1.5\\1.75\end{bmatrix}$

Before we accept this step, however, we need to check the penalty function,

$$P=f+\sum_{i=1}^{vio}\lambda_i|g_i|$$

to make sure it decreased with the step. At the starting point, the constraint is satisfied, so the penalty function is just the value of the objective, $P=17$. At the proposed point the objective value is $f=10.5$ and the constraint is slightly violated with $g=-0.25$. The penalty function is therefore, $P=10.5+5.0*|-0.25|=11.75$. Since this is less than 17, we accept the full step. Contours of the first approximation and the path of the first step are shown in Fig. 7.3.

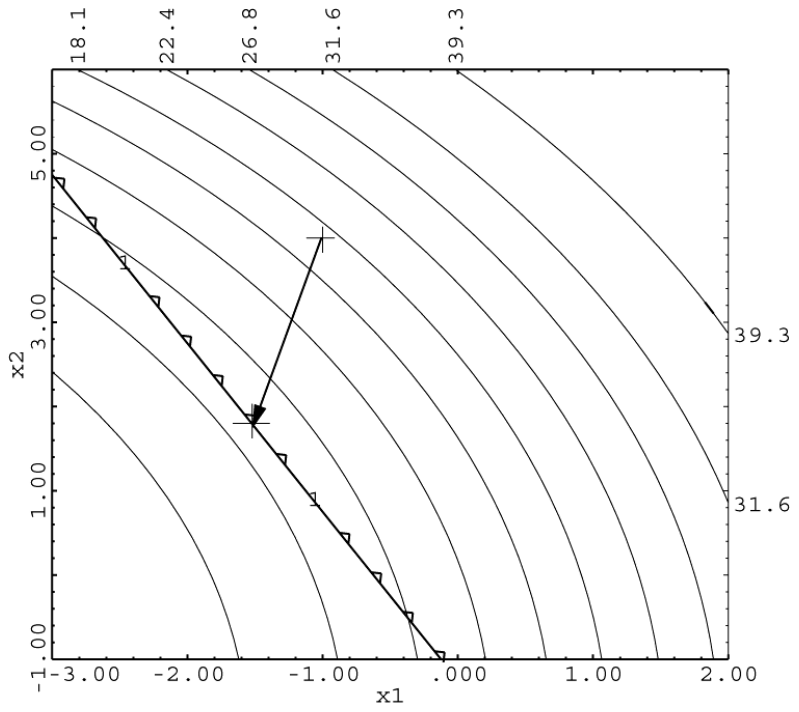

Fig. 7.3 The first SQP approximation and step.

Iteration 2

At $\left(\mathbf{x}^1\right)^{\mathrm{T}} = \begin{bmatrix} -1.5 & 1.75 \end{bmatrix}$, $f^1 = 10.5$, $\left(\nabla f^1\right)^T = \begin{bmatrix} -8.0 & -1.0 \end{bmatrix}$,

$g^1 = -0.25$, $\left(\nabla g^1\right)^T = \begin{bmatrix} 2.5 & 0.75 \end{bmatrix}$

We now need to update the Hessian of the Lagrangian. To do this we need the Lagrangian gradient at $\mathbf{x}^0$ and $\mathbf{x}^1$. (Note that we use the same Lagrange multiplier, λ^1, for both gradients.)

$$\nabla \mathrm{L}\left(\mathbf{x}^0, \boldsymbol{\lambda}^1\right) = \begin{bmatrix} 8.0 \\ 6.0 \end{bmatrix} - (5.0)\begin{bmatrix} 1.5 \\ 0.75 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 2.25 \end{bmatrix}$$

$$\nabla \mathrm{L}\left(\mathbf{x}^1, \boldsymbol{\lambda}^1\right) = \begin{bmatrix} -8.0 \\ -1.0 \end{bmatrix} - (5.0)\begin{bmatrix} 2.5 \\ 0.75 \end{bmatrix} = \begin{bmatrix} -20.5 \\ -4.75 \end{bmatrix}$$

$$\boldsymbol{\gamma}^0 = \nabla \mathrm{L}\left(\mathbf{x}^1, \boldsymbol{\lambda}^1\right) - \nabla \mathrm{L}\left(\mathbf{x}^0, \boldsymbol{\lambda}^1\right) = \begin{bmatrix} -21.0 \\ -7.0 \end{bmatrix}$$

$$\Delta \mathbf{x}^0 = \begin{bmatrix} -1.5 \\ 1.75 \end{bmatrix} - \begin{bmatrix} -1.0 \\ 4.0 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -2.25 \end{bmatrix}$$

From Chapter 3, we will use the BFGS Hessian update,

$$\mathbf{H}^{k+1} = \mathbf{H}^k + \frac{\boldsymbol{\gamma}^k \left(\boldsymbol{\gamma}^k\right)^{\mathrm{T}}}{\left(\boldsymbol{\gamma}^k\right)^{\mathrm{T}} \Delta \mathbf{x}^k} - \frac{\mathbf{H}^k \Delta \mathbf{x}^k \left(\Delta \mathbf{x}^k\right)^{\mathrm{T}} \mathbf{H}^k}{\left(\Delta \mathbf{x}^k\right)^{\mathrm{T}} \mathbf{H}^k \Delta \mathbf{x}^k}$$

Substituting:

$$\nabla^2 \mathrm{L}^1 = \begin{bmatrix} 1. & 0. \\ 0. & 1. \end{bmatrix} + \frac{\begin{bmatrix} -21.0 \\ -7.0 \end{bmatrix}\begin{bmatrix} -21.0 & -7.0 \end{bmatrix}}{\begin{bmatrix} -21.0 & -7.0 \end{bmatrix}\begin{bmatrix} -0.5 \\ -2.25 \end{bmatrix}} - \frac{\begin{bmatrix} 1. & 0. \\ 0. & 1. \end{bmatrix}\begin{bmatrix} -0.5 \\ -2.25 \end{bmatrix}\begin{bmatrix} -0.5 & -2.25 \end{bmatrix}\begin{bmatrix} 1. & 0. \\ 0. & 1. \end{bmatrix}}{\begin{bmatrix} -0.5 & -2.25 \end{bmatrix}\begin{bmatrix} 1. & 0. \\ 0. & 1. \end{bmatrix}\begin{bmatrix} -0.5 \\ -2.25 \end{bmatrix}}$$

$$\nabla^2 \mathrm{L}^1 = \begin{bmatrix} 1. & 0. \\ 0. & 1. \end{bmatrix} + \begin{bmatrix} 16.8000 & 5.6000 \\ 5.6000 & 1.8667 \end{bmatrix} - \begin{bmatrix} 0.0471 & 0.2118 \\ 0.2118 & 0.9529 \end{bmatrix}$$

$$\nabla^2 \mathrm{L}^1 = \begin{bmatrix} 17.7529 & 5.3882 \\ 5.3882 & 1.9137 \end{bmatrix}$$

The second approximation is therefore,

$$f_a = 10.5 + \begin{bmatrix} -8.0 & -1.0 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \Delta x_1 & \Delta x_2 \end{bmatrix} \begin{bmatrix} 17.753 & 5.3882 \\ 5.3882 & 1.9137 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix}$$

$$g_a = -0.25 + \begin{bmatrix} 2.5 & 0.75 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} \geq 0$$

As we did before, we will assume the constraint is binding. The K-T equations are,

$$\begin{aligned} &-8 + 17.753\Delta x_1 + 5.3882\Delta x_2 - \lambda(2.5) = 0 \\ &-1 + 5.3882\Delta x_1 + 1.9137\Delta x_2 - \lambda(0.75) = 0 \\ &-0.25 + 2.5\Delta x_1 + 0.75\Delta x_2 = 0 \end{aligned}$$

The solution to this set of equations is $\Delta x_1 = 1.6145,\ \Delta x_2 = -5.048,\ \lambda = -2.615$. Because λ is negative, we need to drop the constraint from the picture. (We can see in Fig. 7.4 below that the constraint is not binding at the optimum.) With the constraint dropped, the solution is, $\Delta x_1 = 2.007,\ \Delta x_2 = -5.131,\ \lambda = 0.$ This gives a new **x** of,

$$\mathbf{x}^2 = \mathbf{x}^1 + \Delta\mathbf{x} = \begin{bmatrix} -1.5 \\ 1.75 \end{bmatrix} + \begin{bmatrix} 2.007 \\ -5.131 \end{bmatrix} = \begin{bmatrix} 0.507 \\ -3.381 \end{bmatrix}$$

However, when we try to step this far, we find the penalty function has increased from 11.75 to 17.48 (this is the value of the objective only—the violated constraint does not enter in to the penalty function because the Lagrange multiplier is zero). We cut the step back. How much to cut back is somewhat arbitrary. We will make the step 0.5 times the original. The new value of **x** becomes,

$$\mathbf{x}^2 = \mathbf{x}^1 + \Delta\mathbf{x} = \begin{bmatrix} -1.5 \\ 1.75 \end{bmatrix} + 0.5 \begin{bmatrix} 2.007 \\ -5.131 \end{bmatrix} = \begin{bmatrix} -0.4965 \\ -0.8155 \end{bmatrix}$$

At which point the penalty function is 7.37. So we accept this step. Contours of the second approximation are shown in Fig. 7.4, along with the step taken.

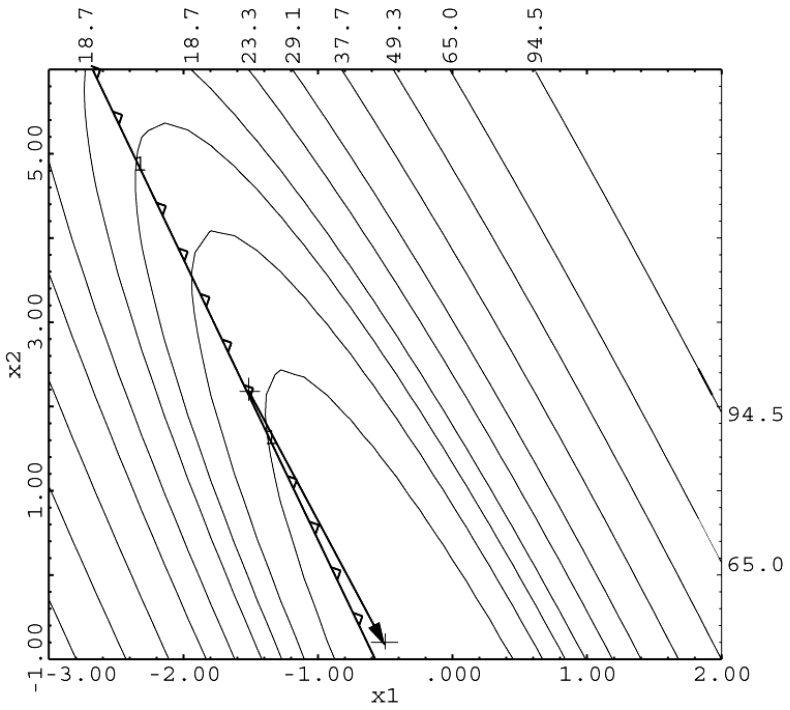

Fig. 7.4 The second approximation and step.

Iteration 3

At $\left(\mathbf{x}^2\right)^{\mathrm{T}} = \begin{bmatrix} -0.4965 & -0.8155 \end{bmatrix}$, $f^2 = 7.367$, $\left(\nabla f^2\right)^T = \begin{bmatrix} -5.102 & -2.124 \end{bmatrix}$,

$g^2 = -0.6724$, $\left(\nabla g^2\right)^T = \begin{bmatrix} 0.493 & 0.75 \end{bmatrix}$

$$\nabla \mathrm{L}\left(\mathbf{x}^1, \boldsymbol{\lambda}^2\right) = \begin{bmatrix} -8.0 \\ -1.0 \end{bmatrix} - (0) \begin{bmatrix} 2.5 \\ 0.75 \end{bmatrix} = \begin{bmatrix} -8.0 \\ -1.0 \end{bmatrix}$$

$$\nabla \mathrm{L}\left(\mathbf{x}^2, \boldsymbol{\lambda}^2\right) = \begin{bmatrix} -5.102 \\ -2.124 \end{bmatrix} - (0) \begin{bmatrix} 0.493 \\ 0.75 \end{bmatrix} = \begin{bmatrix} -5.102 \\ -2.124 \end{bmatrix}$$

$$\boldsymbol{\gamma}^1 = \nabla \mathrm{L}\left(\mathbf{x}^2, \boldsymbol{\lambda}^2\right) - \nabla \mathrm{L}\left(\mathbf{x}^1, \boldsymbol{\lambda}^2\right) = \begin{bmatrix} 2.898 \\ -1.124 \end{bmatrix}$$

$$\Delta \mathbf{x}^1 = \begin{bmatrix} -0.4965 \\ -0.8155 \end{bmatrix} - \begin{bmatrix} -1.5 \\ 1.75 \end{bmatrix} = \begin{bmatrix} 1.004 \\ -2.5655 \end{bmatrix}$$

Based on these vectors, the new Lagrangian Hessian is,

$$\nabla^2 \mathrm{L}^2 = \begin{bmatrix} 17.7529 & 5.3882 \\ 5.3882 & 1.9137 \end{bmatrix} + \begin{bmatrix} 1.4497 & -0.5623 \\ -0.5623 & 0.2181 \end{bmatrix} - \begin{bmatrix} 5.8551 & 0.7320 \\ 0.7320 & 0.0915 \end{bmatrix}$$

$$\nabla^2 L^2 = \begin{bmatrix} 13.3475 & 4.0939 \\ 4.0939 & 2.0403 \end{bmatrix}$$

So our next approximation is,

$$f_a = 7.367 + \begin{bmatrix} -5.102 & -2.124 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \Delta x_1 & \Delta x_2 \end{bmatrix} \begin{bmatrix} 13.3475 & 4.0939 \\ 4.0939 & 2.0403 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix}$$

$$g_a = -0.6724 + \begin{bmatrix} 0.493 & 0.75 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} \geq 0$$

The K-T equations, assuming the constraint is binding, are,

$$\begin{aligned} -5.102 + 13.3475\Delta x_1 + 4.0939\Delta x_2 - \lambda(0.493) &= 0 \\ -2.124 + 4.0939\Delta x_1 + 2.0403\Delta x_2 - \lambda(0.75) &= 0 \\ -0.6724 + 0.493\Delta x_1 + 0.75\Delta x_2 &= 0 \end{aligned}$$

The solution to this set of equations is $\Delta x_1 = 0.1399,\ \Delta x_2 = 0.8046,\ \lambda = 0.1205$.

Our new proposed point is, $\mathbf{x}^3 = \mathbf{x}^2 + \Delta\mathbf{x} = \begin{bmatrix} -0.4965 \\ -0.8155 \end{bmatrix} + \begin{bmatrix} 0.1399 \\ 0.8046 \end{bmatrix} = \begin{bmatrix} -0.3566 \\ -0.0109 \end{bmatrix}$

At this point the penalty function has decreased from 7.37 to 5.85. We accept the full step. A contour plot of the third approximation is shown in Fig. 7.5.

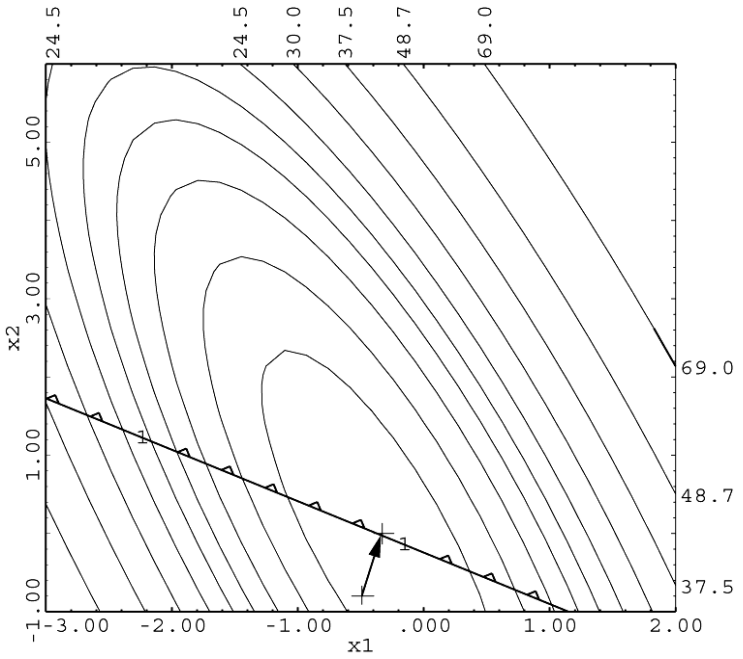

Fig. 7.5 The third approximation and step.

Iteration 4

At $\left(\mathbf{x}^3\right)^T = \begin{bmatrix} -0.3566 & -0.0109 \end{bmatrix}$, $f^3 = 5.859$, $\left(\nabla f^3\right)^T = \begin{bmatrix} -2.9101 & -0.2761 \end{bmatrix}$,

$g^3 = -0.01954$, $\left(\nabla g^3\right)^T = \begin{bmatrix} 0.2132 & 0.75 \end{bmatrix}$

$$\nabla \mathrm{L}\left(\mathbf{x}^2, \boldsymbol{\lambda}^3\right) = \begin{bmatrix} -5.102 \\ -2.124 \end{bmatrix} - \left(0.1205\right)\begin{bmatrix} 0.493 \\ 0.75 \end{bmatrix} = \begin{bmatrix} -5.161 \\ -2.214 \end{bmatrix}$$

$$\nabla \mathrm{L}\left(\mathbf{x}^3, \boldsymbol{\lambda}^3\right) = \begin{bmatrix} -2.910 \\ -0.2761 \end{bmatrix} - \left(0.1205\right)\begin{bmatrix} 0.2132 \\ 0.75 \end{bmatrix} = \begin{bmatrix} -2.936 \\ -0.3665 \end{bmatrix}$$

$$\boldsymbol{\gamma}^2 = \nabla \mathrm{L}\left(\mathbf{x}^3, \boldsymbol{\lambda}^3\right) - \nabla \mathrm{L}\left(\mathbf{x}^2, \boldsymbol{\lambda}^3\right) = \begin{bmatrix} 2.225 \\ 1.8475 \end{bmatrix}$$

$$\Delta \mathbf{x}^2 = \begin{bmatrix} -0.3566 \\ -0.0109 \end{bmatrix} - \begin{bmatrix} -0.4965 \\ -0.8155 \end{bmatrix} = \begin{bmatrix} 0.1399 \\ 0.8046 \end{bmatrix}$$

Based on these vectors, the new Lagrangian Hessian is,

$$\nabla^2 \mathrm{L}^3 = \begin{bmatrix} 13.3475 & 4.0939 \\ 4.0939 & 2.0403 \end{bmatrix} + \begin{bmatrix} 2.7537 & 2.2865 \\ 2.2865 & 1.8986 \end{bmatrix} - \begin{bmatrix} 10.6397 & 4.5647 \\ 4.5647 & 1.9584 \end{bmatrix}$$

$$\nabla^2 \mathrm{L}^3 = \begin{bmatrix} 5.4616 & 1.8157 \\ 1.8157 & 1.9805 \end{bmatrix}$$

Our new approximation is,

$$f_a = 5.859 + \begin{bmatrix} -2.910 & -0.2761 \end{bmatrix}\begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \Delta x_1 & \Delta x_2 \end{bmatrix}\begin{bmatrix} 5.4616 & 1.8157 \\ 1.8157 & 1.9805 \end{bmatrix}\begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix}$$

$$g_a = -0.0195 + \begin{bmatrix} 0.2132 & 0.75 \end{bmatrix}\begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} \geq 0$$

The K-T equations, assuming the constraint is binding, are,

$$\begin{aligned} &-2.910 + 5.4616\Delta x_1 + 1.8157\Delta x_2 - \lambda(0.2132) = 0 \\ &-0.2761 + 1.8157\Delta x_1 + 1.9805\Delta x_2 - \lambda\left(0.75\right) = 0 \\ &-0.0195 + 0.2132\Delta x_1 + 0.75\Delta x_2 = 0 \end{aligned}$$

The solution to this problem is, $\Delta x_1 = 0.6099,\ \Delta x_2 = -0.1474,\ \lambda = 0.7192$. Since λ is positive, our assumption about the constraint was correct. Our new proposed point is,

$$\mathbf{x}^4 = \mathbf{x}^3 + \Delta\mathbf{x} = \begin{bmatrix} -0.3566 \\ -0.0109 \end{bmatrix} + \begin{bmatrix} 0.6099 \\ -0.1474 \end{bmatrix} = \begin{bmatrix} 0.2533 \\ -0.1583 \end{bmatrix}$$

At this point the penalty function is 4.87, a decrease from 5.85, so we take the full step. The contour plot is given in Fig. 7.6

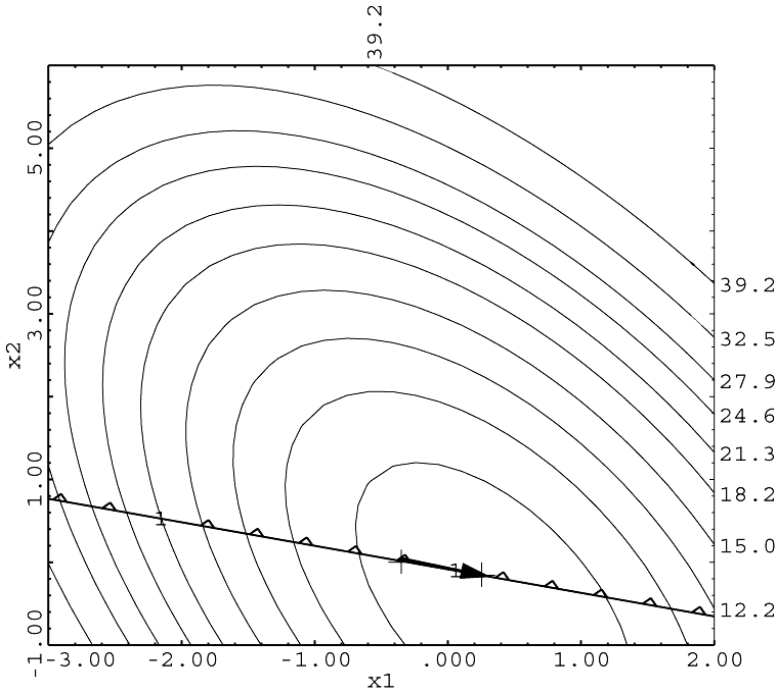

Fig. 7.6 The fourth approximation and step.

Iteration 5

At $\left(\mathbf{x}^4\right)^T = \begin{bmatrix} 0.2533 & -0.1583 \end{bmatrix}$, $f^4 = 4.6071$, $\left(\nabla f^4\right)^T = \begin{bmatrix} -1.268 & -0.4449 \end{bmatrix}$,

$g^4 = -0.3724$, $\left(\nabla g^4\right)^T = \begin{bmatrix} -1.007 & 0.75 \end{bmatrix}$

$$\nabla \mathrm{L}\left(\mathbf{x}^3, \boldsymbol{\lambda}^4\right) = \begin{bmatrix} -2.9101 \\ -0.2761 \end{bmatrix} - (0.7192) \begin{bmatrix} 0.2132 \\ 0.75 \end{bmatrix} = \begin{bmatrix} -3.063 \\ -0.8155 \end{bmatrix}$$

$$\nabla \mathrm{L}\left(\mathbf{x}^4, \boldsymbol{\lambda}^4\right) = \begin{bmatrix} -1.268 \\ -0.4449 \end{bmatrix} - (0.7192) \begin{bmatrix} -1.007 \\ 0.75 \end{bmatrix} = \begin{bmatrix} -0.5438 \\ -0.9843 \end{bmatrix}$$

$$\boldsymbol{\gamma}^3 = \nabla \mathrm{L}\left(\mathbf{x}^4, \boldsymbol{\lambda}^4\right) - \nabla \mathrm{L}\left(\mathbf{x}^3, \boldsymbol{\lambda}^4\right) = \begin{bmatrix} 2.519 \\ -0.1688 \end{bmatrix}$$

$$\Delta\mathbf{x}^3 = \begin{bmatrix} 0.2533 \\ -0.1583 \end{bmatrix} - \begin{bmatrix} -0.3566 \\ -0.0109 \end{bmatrix} = \begin{bmatrix} 0.6099 \\ -0.1474 \end{bmatrix}$$

Based on these vectors, the new Lagrangian Hessian is,

$$\nabla^2 \mathrm{L}^4 = \begin{bmatrix} 5.4616 & 1.8157 \\ 1.8157 & 1.9805 \end{bmatrix} + \begin{bmatrix} 4.0644 & -0.2724 \\ -0.2724 & 0.0183 \end{bmatrix} - \begin{bmatrix} 5.3681 & 1.4290 \\ 1.4290 & 0.3804 \end{bmatrix}$$

$$\nabla^2 \mathrm{L}^4 = \begin{bmatrix} 4.1578 & 0.1144 \\ 0.1144 & 1.6184 \end{bmatrix}$$

Our new approximation is,

$$f_a = 4.6071 + \begin{bmatrix} -1.268 & -0.4449 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \Delta x_1 & \Delta x_2 \end{bmatrix} \begin{bmatrix} 4.1578 & 0.1144 \\ 0.1144 & 1.6184 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix}$$

$$g_a = -0.3724 + \begin{bmatrix} -1.007 & 0.75 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} \geq 0$$

The K-T equations, assuming the constraint is binding, are,

$$\begin{aligned} &-1.268 + 4.1578\Delta x_1 + 0.1144\Delta x_2 - \lambda(-1.007) = 0 \\ &-0.4449 + 0.1144\Delta x_1 + 1.6184\Delta x_2 - \lambda(0.75) = 0 \\ &-0.3724 - 1.007\Delta x_1 + 0.75\Delta x_2 = 0 \end{aligned}$$

The solution to this problem is, $\Delta x_1 = 0.0988,\ \Delta x_2 = 0.6292,\ \lambda = 0.7797$. Since λ is positive, our assumption about the constraint was correct. Our new proposed point is,

$$\mathbf{x}^5 = \mathbf{x}^4 + \Delta\mathbf{x} = \begin{bmatrix} 0.2533 \\ -0.1583 \end{bmatrix} + \begin{bmatrix} 0.0988 \\ 0.6292 \end{bmatrix} = \begin{bmatrix} 0.3521 \\ 0.4709 \end{bmatrix}$$

At this point the penalty function is 4.55, a decrease from 4.87, so we take the full step. The contour plot is given in Fig. 7.7

We would continue in this fashion until $\Delta\mathbf{x}$ goes to zero. We would then know the original K-T equations were satisfied. The solution to this problem occurs at,

$$\left(\mathbf{x}^*\right)^{\mathrm{T}} = \begin{bmatrix} 0.495 & 0.739 \end{bmatrix}, \quad f^* = 4.50$$

Using OptdesX, SQP requires 25 calls to the analysis program. GRG takes 50 calls.

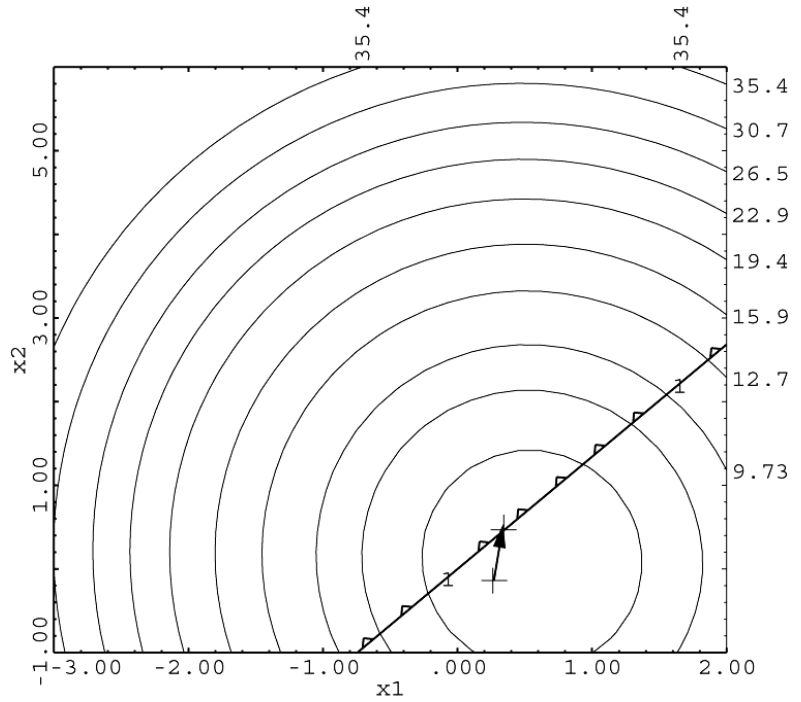

Fig. 7.7 The fifth approximation and step.

A summary of all the steps is overlaid on the original problem in Fig. 7.8.

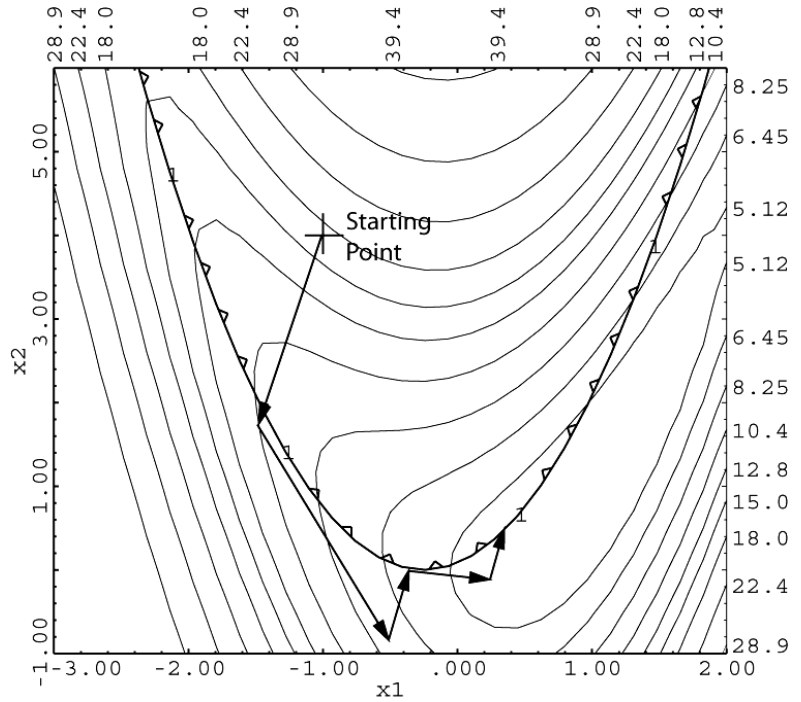

Fig. 7.8 The path of the SQP algorithm.

3 The Generalized Reduced Gradient (GRG) Algorithm

3.1 Introduction

In the previous section we learned that SQP works by solving for the point where the K-T equations are satisfied. SQP gradually enforces feasibility of the constraints as part of the K-T equations. In this section we will learn how GRG works. We will find it is very different from SQP. If started inside feasible space, GRG goes downhill until it runs into fences—constraints--and then corrects the search direction such that it follows the fences downhill. At every step it enforces feasibility. The strategy of GRG in following fences works well for engineering problems because most engineering optimums are constrained.

3.2 Explicit vs. Implicit Elimination

Suppose we have the following optimization problem,

$$\text{Min} \quad f(\mathbf{x}) = x_1^2 + 3x_2^2 \tag{7.41}$$

$$\text{s.t.} \quad g(\mathbf{x}) = 2x_1 + x_2 - 6 = 0 \tag{7.42}$$

A contour plot is given in Fig. 7.9a.

From previous discussions about modeling in Chapter 2, we know there are two approaches to this problem—we can solve it as a problem in two variables with one equality constraint, or we can use the equality constraint to eliminate a variable and the constraint. We will use the second approach. Using (7.42) to solve for x_2,

$$x_2 = 6 - 2x_1$$

Substituting into the objective function, (7.41), we have,

$$\text{Min} \quad f(\mathbf{x}) = x_1^2 + 3(6 - 2x_1)^2 \tag{7.43}$$

Mathematically, solving the problem given by (7.41-7.42) is the same as solving the problem in (7.43). We have used the constraint to *explicitly* eliminate a variable and a constraint. Once we solve for the optimal value of x_1, we will obviously have to back substitute to get the value of x_2 using 7.42. The solution in x_1 is illustrated in Fig. 7.9b, where the sensitivity plot for 7.43 is given (because we only have one variable, we can't show a contour plot). The derivative $\frac{df}{dx_1}$ of (7.43) would be considered to be the *reduced gradient* relative to the original problem.

Usually we cannot make an explicit substitution as we did in this example. So we eliminate variables *implicitly*. We show how this can be done in the next section.

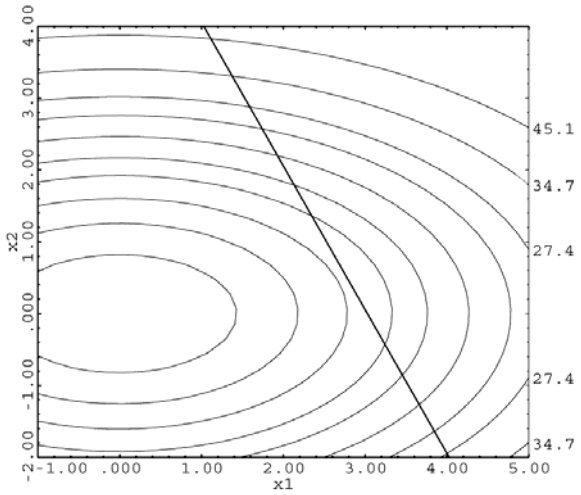

Fig. 7.9 a) Contour plot in x_1 , x_2 with equality constraint. The optimum is at $\mathbf{x}^T = \begin{bmatrix} 2.7693 & 0.4613 \end{bmatrix}$.

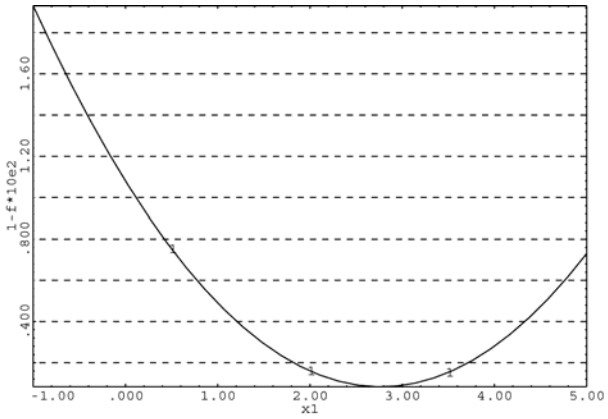

Fig. 7.9 b) Sensitivity plot for 7.43. The optimum is at $x_1 = 2.7693$

3.3 Implicit Elimination

In this section we will look at how we can eliminate variables implicitly. We do this by considering *differential* changes in the objective and constraints. We will start by considering a simple problem of two variables with one equality constraint,

$$\text{Min} \quad f(\mathbf{x}) \quad \mathbf{x}^T = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \tag{7.44}$$

$$\text{s.t.} \quad g(\mathbf{x}) - b = 0 \tag{7.45}$$

Suppose we are at a feasible point. Thus the equality constraint is satisfied. We wish to move to improve the objective function. The differential change is given by,

$$df = \frac{\partial f}{\partial x_1} dx_1 + \frac{\partial f}{\partial x_2} dx_2 \tag{7.46}$$

to keep the constraint satisfied the differential change must be zero:

$$dg = \frac{\partial g}{\partial x_1} dx_1 + \frac{\partial g}{\partial x_2} dx_2 = 0 \tag{7.47}$$

Solving for dx_2 in (7.47) gives:

$$dx_2 = \frac{-\partial g/\partial x_1}{\partial g/\partial x_2} dx_1$$

substituting into (7.46) gives,

$$df = \left[\frac{\partial f}{\partial x_1} - \frac{\partial f}{\partial x_2} \left(\frac{\partial g/\partial x_1}{\partial g/\partial x_2} \right) \right] dx_1 \tag{7.48}$$

where the term in brackets is the reduced gradient.

i.e., $$\frac{df_R}{dx_1} = \left[\frac{\partial f}{\partial x_1} - \frac{\partial f}{\partial x_2} \left(\frac{\partial g/\partial x_1}{\partial g/\partial x_2} \right) \right] \tag{7.49}$$

If we substitute Δx for dx, then the equations are only approximate. *We are stepping tangent to the constraint in a direction that improves the objective function.*

3.4 GRG Algorithm with Equality Constraints Only

We can extend the concepts of the previous section to the general problem which we represent in vector notation. Suppose now we consider the general problem with equality constraints,

$$\begin{aligned} &\text{Min} && f(\mathbf{x}) \\ &\text{s.t.} && g_i(\mathbf{x}) - b_i = 0 \quad i = 1, \ldots, m \end{aligned}$$

We have *n* design variables and *m* equality constraints. We begin by partitioning the design variables into (*n*-*m*) independent variables, **z**, and *m* dependent variables **y**. The independent variables will be used to improve the objective function, and the dependent variables will be used to satisfy the binding constraints. If we partition the gradient vectors as well we have,

$$\nabla f(\mathbf{z})^{\mathrm{T}} = \left[\frac{\partial f(\mathbf{x})}{\partial z_1} \quad \frac{\partial f(\mathbf{x})}{\partial z_2} \quad \cdots \quad \frac{\partial f(\mathbf{x})}{\partial z_{n-m}} \right]$$

$$\nabla f(\mathbf{y})^{\mathrm{T}} = \left[\frac{\partial f(\mathbf{x})}{\partial y_1} \quad \frac{\partial f(\mathbf{x})}{\partial y_1} \quad \cdots \quad \frac{\partial f(\mathbf{x})}{\partial y_m} \right]$$

We will also define independent and dependent matrices of the partial derivatives of the constraints:

$$\frac{\partial \psi}{\partial \mathbf{z}} = \begin{bmatrix} \frac{\partial g_1}{\partial z_1} & \frac{\partial g_1}{\partial z_2} & \cdots & \frac{\partial g_1}{\partial z_{n-m}} \\ \frac{\partial g_m}{\partial z_1} & \frac{\partial g_m}{\partial z_2} & \cdots & \frac{\partial g_m}{\partial z_{n-m}} \end{bmatrix} \qquad \frac{\partial \psi}{\partial \mathbf{y}} = \begin{bmatrix} \frac{\partial g_1}{\partial y_1} & \frac{\partial g_1}{\partial y_2} & \cdots & \frac{\partial g_m}{\partial y_m} \\ \frac{\partial g_m}{\partial y_1} & \frac{\partial g_m}{\partial y_2} & \cdots & \frac{\partial g_m}{\partial y_m} \end{bmatrix}$$

We can write the differential changes in the objective and constraints in vector form as:

$$df = f(\mathbf{z})^{\mathrm{T}}\, d\mathbf{z} + f(\mathbf{y})^{\mathrm{T}}\, d\mathbf{y} \tag{7.50}$$

$$d\psi = \frac{\partial \psi}{\partial \mathbf{z}} d\mathbf{z} + \frac{\partial \psi}{\partial \mathbf{y}} d\mathbf{y} = \mathbf{0} \tag{7.51}$$

Noting that $\frac{\partial \psi}{\partial \mathbf{y}}$ is a square matrix, and solving (7.51) for d$\mathbf{y}$,

$$d\mathbf{y} = -\frac{\partial \psi^{-1}}{\partial \mathbf{y}} \frac{\partial \psi}{\partial \mathbf{z}} d\mathbf{z} \tag{7.52}$$

substituting (7.52) into (7.50),

$$df = \nabla f(\mathbf{z})^{\mathrm{T}}\, d\mathbf{z} - \nabla f(\mathbf{y})^{\mathrm{T}} \frac{\partial \psi^{-1}}{\partial \mathbf{y}} \frac{\partial \psi}{\partial \mathbf{z}} d\mathbf{z}$$

or

$$\nabla f_R^{\mathrm{T}} = \nabla f(\mathbf{z})^{\mathrm{T}} - \nabla f(\mathbf{y})^{\mathrm{T}} \frac{\partial \psi^{-1}}{\partial \mathbf{y}} \frac{\partial \psi}{\partial \mathbf{z}} \tag{7.53}$$

where ∇f_R^{T} is the reduced gradient. *The reduced gradient is the direction of steepest ascent that stays tangent to the binding constraints*.

3.5 GRG Example 1: One Equality Constraint

We will illustrate the theory of the previous section with the following example. For this example we will have three variables and one equality constraint. We state the problem as,

Min $\quad f = 4x_1^2 + x_2^2 + 3x_3^2$

s.t. $\quad g = 2x_1 + 4x_2 - x_3 = 10$

Step 1: Evaluate the objective and constraints at the starting point.
The starting point will be $\mathbf{x}^T = [2 \quad 2 \quad 2]$, at which point $f = 32$ and $g = 10$. So the constraint is satisfied.

Step 2: Partition the variables.
We have one binding constraint so we will need one dependent variable. We will arbitrarily choose x_1 as the dependent variable, so $\mathbf{y} = [x_1]$. The independent variables will therefore be $\mathbf{z}^T = [x_2 \quad x_3]$. Thus,

$$\mathbf{z} = \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} x_1 \end{bmatrix} \quad \nabla f(\mathbf{z}) = \begin{bmatrix} \dfrac{\partial f}{\partial x_2} \\ \dfrac{\partial f}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 2x_2 \\ 6x_3 \end{bmatrix}_{@2,2} = \begin{bmatrix} 4 \\ 12 \end{bmatrix} \quad \nabla f(\mathbf{y}) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \end{bmatrix} = \begin{bmatrix} 8x_1 \end{bmatrix}_{@2} = \begin{bmatrix} 16 \end{bmatrix}$$

$$\frac{\partial \psi}{\partial z} = \begin{bmatrix} \dfrac{\partial g}{\partial x_2} & \dfrac{\partial g}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 4 & -1 \end{bmatrix} \qquad \frac{\partial \psi}{\partial y} = \begin{bmatrix} \dfrac{\partial g}{\partial x_1} \end{bmatrix} = \begin{bmatrix} 2 \end{bmatrix}$$

Step 3: Compute the reduced gradient.
We now have the information we need to compute the reduced gradient:

$$\nabla f_R^{\mathrm{T}} = \nabla f(\mathbf{z})^{\mathrm{T}} - \nabla f(\mathbf{y})^{\mathrm{T}} \frac{\partial \psi^{-1}}{\partial \mathbf{y}} \frac{\partial \psi}{\partial \mathbf{z}}$$

$$\nabla f_R^{\mathrm{T}} = \begin{bmatrix} 4 & 12 \end{bmatrix} - \begin{bmatrix} 16 \end{bmatrix} \begin{bmatrix} \dfrac{1}{2} \end{bmatrix} \begin{bmatrix} 4 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} -28 & 20 \end{bmatrix}$$

Step 4: Compute the direction of search.
We will step in the direction of steepest descent, i.e., the negative reduced gradient direction, which is the direction of steepest descent which stays tangent to the constraint.

$$\mathbf{s} = \begin{bmatrix} 28 \\ -20 \end{bmatrix} \qquad \text{or, normalized, } \mathbf{s} = \begin{bmatrix} 0.8137 \\ -0.5812 \end{bmatrix}$$

Step 5: Do a line search in the independent variables
We will use our regular formula,

$$\mathbf{z}^{new} = \mathbf{z}^{old} + \alpha \mathbf{s}$$

We will arbitrarily pick a starting step length $\alpha = 0.5$

$$\begin{bmatrix} x_2^{new} \\ x_3^{new} \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} + 0.5 \begin{bmatrix} 0.8137 \\ -0.5812 \end{bmatrix} = \begin{bmatrix} 2.4068 \\ 1.7094 \end{bmatrix}$$

Step 6: Solve for the value of the dependent variable.
We do this using (7.52) above, only we will substitute Δy for dy:

$$\Delta \mathbf{y} = -\frac{\partial \psi^{-1}}{\partial \mathbf{y}} \frac{\partial \psi}{\partial \mathbf{z}} \Delta \mathbf{z}$$

$$
\begin{aligned}
\left[\Delta x_1\right] &= -\frac{\partial \psi^{-1}}{\partial \mathbf{y}} \frac{\partial \psi}{\partial \mathbf{z}} \begin{bmatrix} \Delta x_2 \\ \Delta x_3 \end{bmatrix} \\
&= -\begin{bmatrix} \frac{1}{2} \end{bmatrix} \begin{bmatrix} 4 & -1 \end{bmatrix} \begin{bmatrix} 0.4069 \\ -0.2906 \end{bmatrix} \\
&= -0.9590
\end{aligned}
$$

So the new value of x_1 is,

$$
\begin{aligned}
x_1^{new} &= x_1^{old} + \Delta x \\
&= 2 - 0.9590 \\
&= 1.041
\end{aligned}
$$

Our new point is $\mathbf{x}^T = \begin{bmatrix} 1.041 & 2.4069 & 1.7094 \end{bmatrix}$ at which point $f = 18.9$ and $g = 10$. We observe that the objective has decreased from 32 to 18.9 and the constraint is still satisfied. This only represents one step in the line search. We would continue the line search until we reach a minimum.

3.6 GRG Algorithm with Equality and Inequality Constraints

In this section we will consider the general problem with both inequality and equality constraints,

$$
\begin{aligned}
\text{Min} \quad & f(\mathbf{x}) \\
\text{s.t.} \quad & g_i(\mathbf{x}) - b_i \le 0 \quad i = 1, \ldots, k \\
& g_i(\mathbf{x}) - b_i = 0 \quad i = k+1, \ldots, m
\end{aligned}
$$

The extension of the GRG algorithm to include inequality constraints involves some additional complexity, because the derivation of GRG is based on equality constraints. We therefore convert inequalities into equalities by adding *slack variables*. So for example,

$$x_1 + 5x_2 \le 6 \text{ is changed to } x_1 + 5x_2 + s_1 = 6$$

where s_1 is the slack variable and must be positive for the constraint to be feasible. The slack variable is zero when the constraint is binding. The word "slack" comes from the idea the variable "takes up the slack" between the function value and the right hand side.

The GRG algorithm described here is an *active constraint* algorithm—only the binding inequality constraints are used to determine the search direction. The non-binding constraints enter into the problem only if they become binding or violated.

With these changes the equations of Section 3.4 can be used. In particular, (7.53) is used to compute the reduced gradient.

3.7 Steps of the GRG Algorithm for the General Problem

1. Evaluate the objective function and all constraints at the current point.

2. For any binding inequality constraints, add a slack variable, s_i

3. Partition the variables into independent variables and dependent variables. We will need one dependent variable for each binding constraint. Any variable at either its upper or lower limit should become an independent variable.

4. Compute the reduced gradient using (7.53).

5. Calculate a direction of search. We can use any method to calculate the search direction that relies on gradients since the reduced gradient is a gradient. For example, we can use a quasi-Newton update.

6. Do a line search in the independent variables. For each step, find the corresponding values in the dependent variables using (7.52) with $\Delta\mathbf{z}$ and $\Delta\mathbf{y}$ substituted for $d\mathbf{z}$ and $d\mathbf{y}$.

7. At each step in the line search, drive back to the constraint boundaries for any violated constraints using Newton-Raphson to adjust the dependent variables. If an independent variable hits its bound, set it equal to its bound.

   The N-R iteration is given by $\Delta\mathbf{y} = -\frac{\partial \psi^{-1}}{\partial \mathbf{y}}(\mathbf{g} - \mathbf{b})$ We note we already have the matrix $\frac{\partial \psi^{-1}}{\partial \mathbf{y}}$ from the calculation of the reduced gradient.

8. The line search may terminate either of 4 ways

   1) The minimum in the direction of search is found (using, for example, quadratic interpolation).
   2) A dependent variable hits its upper or lower limit.
   3) A formerly non-binding constraint becomes binding.
   4) N-R fails to converge. In this case we must cut back the step size until N-R does converge.

9. If at any point the reduced gradient in step 4 is equal to **0**, the K-T conditions are satisfied.

3.8 GRG Example 2: Two Inequality Constraints

In this problem we have two inequality constraints and will therefore need to add in slack variables.

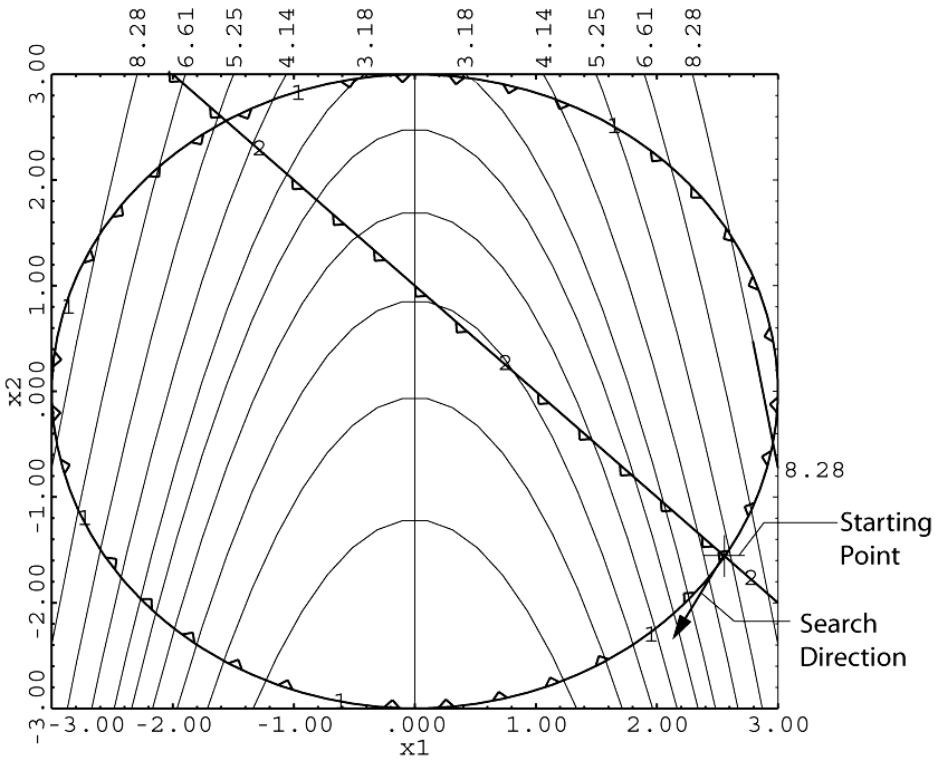

Fig. 7.10 Example problem for GRG algorithm

$$\text{Min.} \quad f(\mathbf{x}) = x_1^2 + x_2$$

$$\text{s.t.:} \quad g_1(\mathbf{x}) = x_1^2 + x_2^2 - 9 \le 0$$

$$g_2(\mathbf{x}) = x_1 + x_2 - 1 \le 0$$

Suppose, to make things interesting, we are starting at $\mathbf{x}^T = [2.56155, -1.56155]$ where both constraints are binding.

Step 1: Evaluate functions.

$$f(\mathbf{x}) = 5.0 \qquad g_1(\mathbf{x}) = 0.0 \qquad g_2(\mathbf{x}) = 0.0$$

Step 2: Add in slack variables.

We note that both constraints are binding so we will add in two slack variables. $s_1,\ s_2$.

Step 3: Partition the variables

Since the slack variables are at their lower limits (=0) they will become the independent variables; x_1, x_2 will be the dependent variables.

$$\mathbf{z}^T = [s_1 \quad s_2] \qquad\qquad \mathbf{y}^T = [x_1 \quad x_2]$$

Step 4: Compute the reduced gradient

$$\nabla f(\mathbf{z})^{\mathrm{T}} = \begin{bmatrix} 0.0 & 0.0 \end{bmatrix} \qquad \nabla f(\mathbf{y})^{\mathrm{T}} = \begin{bmatrix} 5.123 & 1.0 \end{bmatrix}$$

$$\frac{\partial \psi}{\partial \mathbf{z}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \frac{\partial \psi}{\partial \mathbf{y}} = \begin{bmatrix} 5.123 & -3.123 \\ 1.0 & 1.0 \end{bmatrix}$$

$$\frac{\partial \psi^{-1}}{\partial \mathbf{y}} = \begin{bmatrix} 0.1213 & 0.3787 \\ -0.1213 & 0.6213 \end{bmatrix}$$

thus
$$\frac{\partial \psi^{-1}}{\partial \mathbf{y}} \frac{\partial \psi}{\partial \mathbf{z}} = \begin{bmatrix} 0.1213 & 0.3787 \\ -0.1213 & 0.6213 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.1213 & 0.3787 \\ -0.1213 & 0.6213 \end{bmatrix}$$

$$\begin{aligned} \nabla f_r^{\mathrm{T}} &= \begin{bmatrix} 0.0 & 0.0 \end{bmatrix} - \begin{bmatrix} 5.123 & 1 \end{bmatrix} \begin{bmatrix} 0.1213 & 0.3787 \\ -0.1213 & 0.6213 \end{bmatrix} \\ &= \begin{bmatrix} 0.0 & 0.0 \end{bmatrix} - \begin{bmatrix} 0.50 & 2.56 \end{bmatrix} \\ &= \begin{bmatrix} -0.50 & -2.56 \end{bmatrix} \end{aligned}$$

Step 5: Calculate a search direction.
We want to move in the negative gradient direction, so our search direction will be $\mathbf{s}^{\mathrm{T}} = \begin{bmatrix} 0.50 & 2.56 \end{bmatrix}$. This is the direction for the independent variables (the slacks). When normalized this direction is $\mathbf{s}^{\mathrm{T}} = \begin{bmatrix} 0.19 & 0.98 \end{bmatrix}$.

Step 6: Conduct the line search in the independent variables
We will start our line search, denoting the current point as $\mathbf{z}^0$,

$$\mathbf{z}^1 = \mathbf{z}^0 + \alpha \mathbf{s}^0$$

Suppose we pick $\alpha = 1.0$. Then

$$\mathbf{z}^1 = \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix} + (1.0) \begin{bmatrix} 0.19 \\ 0.98 \end{bmatrix}$$

$$\mathbf{z}^1 = \begin{bmatrix} 0.19 \\ 0.98 \end{bmatrix}$$

Step 7: Adjust the dependent variables
To find the change in the dependent variables, we use (7.52)

$$\Delta \mathbf{y} = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = -\left[\frac{\partial \psi^{-1}}{\partial \mathbf{y}} \right] \left[\frac{\partial \psi}{\partial \mathbf{z}} \right] \Delta \mathbf{z}$$

$$= \begin{bmatrix} 0.1213 & 0.3787 \\ -0.1213 & 0.6213 \end{bmatrix} \begin{bmatrix} 0.19 \\ 0.98 \end{bmatrix}$$
$$= \begin{bmatrix} -0.394 \\ -0.586 \end{bmatrix}$$

$$x_1^1 = 2.56155 - 0.394 = 2.168$$
$$x_2^1 = -1.56155 - 0.586 = -2.148$$

at which point $f(\mathbf{x}) = 2.522$

Have we violated any constraints?

$$g_1(\mathbf{x}) = x_1^2 + x_2^2 - 9 = (2.168)^2 + (-2.148)^2 - 9 = 0.31 \text{ (violated)}$$
$$g_2(\mathbf{x}) = x_1 + x_2 - 1 = 2.168 - 2.148 - 1 = -0.98 \text{ (satisfied)}$$

We need to drive back to where the violated constraint is satisfied. We will use N-R to do this. Since we don't want to drive back to where both constraints are binding, we will set the residual for constraint 2 to zero.

N-R Iteration 1:

$$\mathbf{y}^{(n)} = \mathbf{y}^{(0)} - \frac{\partial \psi^{-1}}{\partial \mathbf{y}}(\mathbf{g} - \mathbf{b})$$
$$= \begin{bmatrix} 2.168 \\ -2.148 \end{bmatrix} - \begin{bmatrix} 0.1213 & 0.3787 \\ -0.1213 & 0.6213 \end{bmatrix} \begin{bmatrix} 0.31 \\ 0.0 \end{bmatrix}$$
$$= \begin{bmatrix} 2.130 \\ -2.110 \end{bmatrix}$$

at this point

$$g_1 = (2.130)^2 + (2.110)^2 - 9 = -0.011$$
$$g_2 = -0.98$$

N-R Iteration 2:

$$= \begin{bmatrix} 2.130 \\ -2.110 \end{bmatrix} - \begin{bmatrix} 0.1213 & 0.3787 \\ -0.1213 & 0.6213 \end{bmatrix} \begin{bmatrix} -0.011 \\ 0.0 \end{bmatrix}$$
$$= \begin{bmatrix} 2.1313 \\ -2.113 \end{bmatrix}$$

evaluating constraints:

$$g_1 = (2.1313)^2 + (-2.113)^2 - 9 = 0$$
$$g_2 = -0.98$$

We are now feasible again. We have taken one step in the line search!

Our new point is $\mathbf{x} = \begin{bmatrix} 2.1313 \\ -2.113 \end{bmatrix}$ at which point the objective is 2.43, and all constraints are satisfied.

We would continue the line search until we run into a new constraint boundary, a dependent variable hits a bound, or we can no longer get back on the constraint boundaries (which is not an issue in this example, since the constraint is linear).

4 References

M. J. D. Powell, "Algorithms for Nonlinear Constraints That Use Lagrangian Functions," *Mathematical Programming,* 14, (1978) 224-248.

Powell, M.J.D., "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," *Numerical Analysis, Dundee 1977,* Lecture Notes in Mathematics no. 630, Springer-Verlag, New York, 1978.

L.S. Lasdon et. al, "Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Programming," *ACM Transactions on Mathematical Software*, Vol. 4 No. 1 March 1978

Gabriele G.A., and K.M. Ragsdell, "The Generalized Reduced Gradient Method: A Reliable Tool for Optimal Design," *Trans ASME, J. Eng. Industry,* May 1977.