CHAPTER 3
UNCONSTRAINED OPTIMIZATION

3.1 Preliminaries

3.1.1 Introduction

In this chapter we will examine some theory for the optimization of unconstrained functions. We will assume all functions are continuous and differentiable. Although most engineering problems are constrained, much of constrained optimization theory is built upon the concepts and theory presented in this chapter.

3.1.2 Notation

We will use lower case italics, e.g., *x*, to represent a scalar quantity. Vectors will be represented by lower case bold, e.g., **x**, and matrices by upper case bold, e.g., **H**. We consider vectors to be column vectors; their transpose, e.g. $\mathbf{x}^T$, is a row vector.

The set of *n* design variables will be represented by the *n*-dimensional vector **x**. For example, previously we considered the design variables for the two-bar truss to be represented by scalars such as diameter, d, thickness, t, height, h; now we consider diameter to be the first element, x_1, of the vector **x**, thickness to be the second element, x_2, and so forth. Thus for any problem the set of design variables is given by **x**.

Elements of a vector are denoted by subscripts. Values of a vector at specific points are denoted by superscripts. Typically $\mathbf{x}^0$ will be the starting vector of values for the design variables. We will then move to $\mathbf{x}^1, \mathbf{x}^2$, until we reach the optimum, which will be $\mathbf{x}^*$. A summary of notation used in this chapter is given in Table 3.1.

Table 3.1 Notation

$\mathbf{A}$	Matrix **A**	$\mathbf{x}, \mathbf{x}^k, \mathbf{x}^*$	Vector of design variables, vector at iteration k, vector at the optimum
$\mathbf{I}$	Identity matrix	$x_1, x_2, \ldots, x_n$	Elements of vector **x**
$\mathbf{a}$	Column vector	$\mathbf{s}, \mathbf{s}^k$	Search direction, search direction at iteration k
$\mathbf{a}_i \quad i = 1, 2, \ldots$	Columns of **A**	$\alpha, \alpha^k, \alpha^*$	Step length, step length at iteration k, step length at minimum along search direction
$\mathbf{A}^T, \mathbf{a}^T$	transpose	$f(\mathbf{x}), f(\mathbf{x}^k), f^k$	Objective function; objective evaluated at $\mathbf{x}^k$
$\nabla f(\mathbf{x}), \nabla f(\mathbf{x}^k), \nabla f^k$	Gradient of $f(\mathbf{x})$, gradient evaluated at $\mathbf{x}^k$	$\nabla^2 f(\mathbf{x}^k), \nabla^2 f^k$ $\mathbf{H}(\mathbf{x}^k), \mathbf{H}^k$	Hessian matrix at $\mathbf{x}^k$

$\|\mathbf{x}\|$	Magnitude of vector $\mathbf{x}$, (L_2 norm)	$\lvert x\rvert$	Absolute value of x
$\Delta\mathbf{x}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$	Difference in $\mathbf{x}$ vectors	$\mathbf{x} \in R^n$	The vector $\mathbf{x}$ is an element of n-dimensional Euclidean space
$\boldsymbol{\gamma}^k = \nabla f^{k+1} - \nabla f^k$	Difference in gradients at $\mathbf{x}^{k+1}$, $\mathbf{x}^k$	$\mathbf{N}^k$	Direction matrix at $\mathbf{x}^k$

3.1.3 Statement of Problem

The problem we are trying to solve in this chapter can be stated as,

Find $\mathbf{x}$, $\quad \mathbf{x} \in R^n$
To Minimize $f(\mathbf{x})$

3.1.4 Gradient Vector

3.1.4.1 Definition

The gradient of $f(\mathbf{x})$ is denoted $\nabla f(\mathbf{x})$. The gradient is defined as a column vector of the first partial derivatives of $f(\mathbf{x})$:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix} \tag{3.1}$$

3.1.4.2 Example: Gradient of a Function

Evaluate the gradient of the function $f(\mathbf{x}) = 6 - 2x_1 + x_2 + 2x_1^2 + 3x_1x_2 + x_2^2$

$$\nabla f = \begin{bmatrix} -2 + 4x_1 + 3x_2 \\ 1 + 3x_1 + 2x_2 \end{bmatrix} \text{ If evaluated at } \mathbf{x}^0 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}, \ \nabla f^0 = \begin{bmatrix} -4 \\ -1 \end{bmatrix}$$

3.1.4.3 Properties of the Gradient Vector

A very important property of the gradient vector is that it *is orthogonal to the function contours and points in the direction of greatest increase of a function*. The negative gradient points in the direction of greatest decrease. The dot product of a vector $\mathbf{v}$ which is orthogonal to $\nabla f(\mathbf{x})$ will be zero, i.e. $\mathbf{v}^{\mathrm{T}}\nabla f(\mathbf{x}) = 0$.

3.1.5 Vectors That Point "Downhill" or "Uphill"

If we have some search direction **s**, then $\mathbf{s}^{\mathrm{T}}\nabla f$ is proportional to the projection of **s** onto the gradient vector, which is given by $\dfrac{\mathbf{s}^{T}\nabla f}{\|\nabla f\|}$. This can be developed from the definition for a dot product,

$$\mathbf{s}^{\mathrm{T}}\nabla f = \|\mathbf{s}\| \cdot \|\nabla f\| \cos\theta \tag{3.2}$$

where $\|\mathbf{s}\|, \|\nabla f\|$ represent the magnitude of these vectors, and θ is the angle in between. If $\mathbf{s}^{\mathrm{T}}\nabla f > 0$, then the projection onto the gradient is positive and θ is < 90 degrees. If $\mathbf{s}^{\mathrm{T}}\nabla f < 0$, then the projection is negative and θ is > 90 degrees. We can see this geometrically in Fig. 3.1:

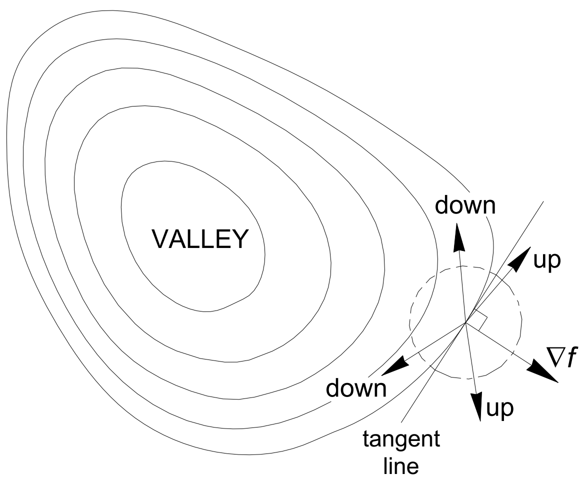

Fig. 3.1. Vectors that point uphill or downhill.

As long as $\mathbf{s}^{\mathrm{T}}\nabla f > 0$, then **s** points, at least for some small distance, in a direction that increases the function (points uphill). In like manner, if $\mathbf{s}^{\mathrm{T}}\nabla f < 0$, then **s** points downhill. As an example, suppose at the current point in space the gradient vector is $\nabla f(\mathbf{x}^{\mathbf{k}})^{T} = [\ 6 \quad 1 \quad -2\ ]$. We propose to move from this point in a search direction $\mathbf{s}^{T} = [\ -1 \quad -1 \quad 0\ ]$. Does this direction go downhill? We evaluate

$$\mathbf{s}^{\mathrm{T}}\nabla f = \begin{bmatrix} -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} 6 \\ 1 \\ -2 \end{bmatrix} = -7$$

So this direction would take us downhill, at least for a short step. A little later in this chapter we will see that this product represents the *directional derivative,* i.e., the derivative of the function in this search direction.

3.1.6 Hessian Matrix

3.1.6.1 Definition

The Hessian Matrix, $\mathbf{H}(\mathbf{x})$ or $\nabla^2 f(\mathbf{x})$, is defined to be the square matrix of second partial derivatives:

$$\mathbf{H}(\mathbf{x}) = \nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \cdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{3.3}$$

We can also obtain the Hessian by applying the gradient operator on the gradient transpose,

$$\mathbf{H}(\mathbf{x}) = \nabla^2 f(\mathbf{x}) = \nabla(\nabla f(\mathbf{x})^T) = \begin{bmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ \vdots \\ \frac{\partial}{\partial x_n} \end{bmatrix} \begin{bmatrix} \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \cdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

The Hessian is a symmetric matrix. The Hessian matrix gives us information about the curvature of a function, and tells us how the gradient is changing.

For simplicity, we will sometimes write $\mathbf{H}^k$ instead of $\mathbf{H}(\mathbf{x}^k)$.

3.1.6.2 Example: Hessian Matrix

Find the Hessian matrix for the function, $f(x) = 6 - 2x_1 + x_2 + 2x_1^2 + 3x_1x_2 + x_2^2$

$$\nabla f = \begin{bmatrix} -2 + 4x_1 + 3x_2 \\ 1 + 3x_1 + 2x_2 \end{bmatrix}, \qquad \frac{\partial^2 f}{\partial x_1^2} = 4 \quad \frac{\partial^2 f}{\partial x_1 \partial x_2} = 3$$

$$\frac{\partial^2 f}{\partial x_2 \partial x_1} = 3 \quad \frac{\partial^2 f}{\partial x_2^2} = 2$$

so the Hessian is:

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} 4 & 3 \\ 3 & 2 \end{bmatrix}$$

3.1.7 Positive and Negative Definiteness

3.1.7.1 Definitions

If for any vector, **x**, the following is true for a symmetric matrix **B**,

$$\begin{aligned} &\mathbf{x}^{\mathrm{T}}\mathbf{B}\mathbf{x} > 0, \text{ then } B \textit{ is positive definite} \\ &\mathbf{x}^{\mathrm{T}}\mathbf{B}\mathbf{x} < 0, \text{ then } B \textit{ is negative definite} \end{aligned} \qquad (3.4)$$

3.1.7.2 Checking Positive Definiteness

The above definition is not very useful in terms of checking if a matrix is positive definite, because it would require that we examine every possible vector **x** to see if the condition given in (3.4) is true. So, how can we tell if a matrix is positive definite? There are three ways we will mention,

1. A symmetric matrix **B** is positive definite if all eigenvalues of **B** are positive.

2. A symmetric matrix is positive definite if the determinant of each of its principal minor matrices is positive.

3. A $n \times n$ symmetric matrix $\mathbf{B}$ is positive definite if it can be written as $\mathbf{B} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$ where $\mathbf{L}$ is a lower triangular matrix with positive diagonal elements. The $\mathbf{L}$ matrix can be developed through Choleski decomposition.

The matrix we will be most interested in checking is the Hessian matrix, $\mathbf{H}(\mathbf{x})$.

What does it mean for the Hessian to be positive or negative definite? If positive definite, it means curvature of the function is everywhere positive. This will be an important condition for checking if we have a minimum. If negative definite, curvature is everywhere negative. This will be a condition for verifying we have a maximum.

3.1.7.3 Example: Checking if a Matrix is Positive Definite Using Principal Minor Matrices

Is the matrix given below positive definite? We need to check the determinants of the principal minor matrices, found by taking the determinant of a 1x1 matrix along the diagonal, the determinant of a 2x2 matrix along the diagonal, and finally the determinant of the entire matrix. If any one of these determinants is not positive, the matrix is not positive definite.

$$\begin{bmatrix} 2 & 3 & -2 \\ 3 & 5 & -1 \\ -2 & -1 & 5 \end{bmatrix}$$

$$\left\|[2]\right\| = 2 > 0 \qquad \left\|\begin{bmatrix} 2 & 3 \\ 3 & 5 \end{bmatrix}\right\| = 1 > 0 \qquad \left\|\begin{bmatrix} 2 & 3 & -2 \\ 3 & 5 & -1 \\ -2 & -1 & 5 \end{bmatrix}\right\| = -5 < 0$$

The determinants of the first two principal minors are positive. However, because the determinant of the matrix as a whole is negative, this matrix is not positive definite.

We also note that the eigenvalues are –0.15, 4.06, 8.09. That these are not all positive also indicates the matrix is not positive definite.

3.1.7.4 Checking Negative Definiteness

How can we check to see if a matrix is negative definite? There are two ways we will mention,

1. A symmetric matrix **B** is negative definite if all eigenvalues of **B** are negative.

2. A symmetric matrix is negative definite if we reverse the sign of each element and the resulting matrix is positive definite.

Note: A symmetric matrix is not negative definite if the determinant of each of its principal minor matrices is negative. Rather, in the negative definite case, the signs of the determinants alternate minus and plus, so the easiest way to check for negative definiteness using principal minor matrices is to reverse all signs and see if the resulting matrix is positive definite.

It is also possible for a matrix to be positive *semi*-definite, or negative *semi*-definite. This occurs when one or more of the determinants or eigenvalues are equal to zero, and the others are all positive (or negative, as the case may be). These are special cases we won't worry about here.

If a matrix is neither positive definite nor negative definite (nor semi-definite), then it is *indefinite*. If using principal minor matrices, note that we need to check both cases before we reach a conclusion that a matrix is indefinite.

3.1.7.5 Example: Checking if a Matrix is Negative Definite Using Principal Minor Matrices

Is the matrix given above negative definite? We reverse the signs and see if the resulting matrix is positive definite:

$$\begin{bmatrix} -2 & -3 & 2 \\ -3 & -5 & 1 \\ 2 & 1 & -5 \end{bmatrix}$$

$$\left\| [-2] \right\| = -2 < 0$$

Because the first determinant is negative there is no reason to go further. We also note that the eigenvalues of the "reversed sign" matrix are not all positive.

Because this matrix is neither positive nor negative definite, it is indefinite.

3.1.8 Taylor Expansion

3.1.8.1 Definition

The Taylor expansion is an approximation to a function at a point $\mathbf{x}^k$ and can be written in vector notation as:

$$f\left(\mathbf{x}^{k+1}\right) = f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^{\mathrm{T}}\left(\mathbf{x}^{k+1} - \mathbf{x}^k\right) + \frac{1}{2}\left(\mathbf{x}^{k+1} - \mathbf{x}^k\right)^{\mathrm{T}} \nabla^2 f(\mathbf{x}^k)\left(\mathbf{x}^{k+1} - \mathbf{x}^k\right) + \ldots \quad (3.5)$$

If we note that $(\mathbf{x}^{k+1} - \mathbf{x}^k)$ can be written as $\Delta\mathbf{x}^k$, and using notation $f\left(\mathbf{x}^k\right) = f^k$, we can write (3.5) more compactly as,

$$f^{k+1} = f^k + \left(\nabla f^k\right)^{\mathrm{T}} \Delta\mathbf{x}^k + \frac{1}{2}\left(\Delta\mathbf{x}^k\right)^{\mathrm{T}} \nabla^2 f^k \Delta\mathbf{x}^k + \ldots \quad (3.6)$$

The Taylor expansion allows us to approximate any continuous function as a polynomial in terms of its derivatives at the point $\mathbf{x}^k$. We can make a linear approximation by taking the first two terms of the expansion. We can make a quadratic approximation by taking the first three terms of the expansion.

3.1.8.2 Example: Quadratic Approximation of a Transcendental Function

Suppose $f(\mathbf{x}) = 2\left(x_1\right)^{1/2} + 3\ln\left(x_2\right)$

$$\text{at } \left(\mathbf{x}^k\right)^{\mathrm{T}} = [5,4] \qquad \nabla f^T = \left[x_1^{(-1/2)}, \frac{3}{x_2}\right] \qquad \left(\nabla f^k\right)^T = \left[0.447, 0.750\right]$$

$$\frac{\partial^2 f}{\partial x_1^2} = -\frac{1}{2}x_1^{(-3/2)} \qquad \frac{\partial^2 f}{\partial x_1 \partial x_2} = 0 \qquad \frac{\partial^2 f}{\partial x_2 \partial x_1} = 0 \qquad \frac{\partial^2 f}{\partial x_2^2} = \frac{-3}{x_2^2}$$

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} -\frac{1}{2}x_1^{(-3/2)} & 0 \\ 0 & \frac{-3}{x_2^2} \end{bmatrix} \text{ at } \begin{bmatrix} 5 \\ 4 \end{bmatrix} = \begin{bmatrix} -0.045 & 0.0 \\ 0.0 & -0.188 \end{bmatrix}$$

$$f(\mathbf{x}) \approx 8.631 + [0.447, 0.750] \begin{bmatrix} x_1 - 5 \\ x_2 - 4 \end{bmatrix} + \frac{1}{2}[x_1 - 5, x_2 - 4] \begin{bmatrix} -0.045 & 0.0 \\ 0.0 & -0.188 \end{bmatrix} \begin{bmatrix} x_1 - 5 \\ x_2 - 4 \end{bmatrix}$$

If we wish, we can stop here with the equation in vector form. To see the equation in scalar form we can carry out the vector multiplications and combine similar terms:

$$f(\mathbf{x}) \approx 8.631 + 0.447x_1 - 2.235 + 0.750x_2 - 3.000 + \frac{1}{2}\left[(-0.045x_1 + 0.225), (-0.188x_2 + 0.752)\right] \begin{bmatrix} x_1 - 5 \\ x_2 - 4 \end{bmatrix}$$

$$f(\mathbf{x}) \approx 3.396 + 0.447x_1 + 0.750x_2 + \frac{1}{2}\left(-0.045x_1^2 + 0.450x_1 - 1.125 - 0.188x_2^2 + 1.504x_2 - 3.008\right)$$

$$f(\mathbf{x}) \approx 1.300 + 0.672x_1 + 1.502x_2 - 0.023x_1^2 - 0.094x_2^2$$

Evaluating and comparing this approximation to the original:

$[\mathbf{x}]^T$	**Quadratic**	**Actual**	**Error**
[5,4]	8.63	8.63	0.00
[5,5]	9.28	9.3	0.02
[6,4]	9.05	9.06	0.01
[7,6]	10.55	10.67	0.12
[2,1]	3.98	2.83	-1.15
[9,2]	8.19	8.08	-0.11

We notice that the further the point gets from the expansion point, the greater the error that is introduced. We also see that at the point of expansion the approximation is exact.

3.2 Properties and Characteristics of Quadratic Functions

A lot of optimization theory is based on optimizing quadratic functions. It is therefore helpful to investigate some of the properties of these functions.

3.2.1 Representation

We can represent a quadratic function three ways—as a scalar equation, a general vector equation, and as a Taylor expansion. Although these representations look different, they give exactly the same results. For example, consider the equation,

$$f(x) = 6 - 2x_1 + x_2 + 2x_1^2 + 3x_1x_2 + x_2^2 \tag{3.7}$$

This is a scalar representation of a quadratic.

As another representation, we can write a quadratic in general vector form,

$$f(\mathbf{x}) = a + \mathbf{b}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{C}\mathbf{x} \tag{3.8}$$

By inspection, the example given of (3.7), in the form of (3.8), is:

$$f(\mathbf{x}) = 6 + [-2,\ 1]\mathbf{x} + \frac{1}{2}\mathbf{x}^T\begin{bmatrix} 4 & 3 \\ 3 & 2 \end{bmatrix}\mathbf{x} \tag{3.9}$$

where,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

We also observe that **C** in (3.8) ends up being **H**.

A third form is a Taylor representation,

$$f(\mathbf{x}) = f^k + \left(\nabla f^k\right)^{\mathrm{T}} \Delta\mathbf{x}^k + \frac{1}{2}\left(\Delta\mathbf{x}^k\right)^{\mathrm{T}} \mathbf{H}\Delta\mathbf{x}^k \tag{3.10}$$

We note for (3.7), $\nabla f = \begin{bmatrix} -2 + 4x_1 + 3x_2 \\ 1 + 3x_1 + 2x_2 \end{bmatrix}$ and $\mathbf{H} = \begin{bmatrix} 4 & 3 \\ 3 & 2 \end{bmatrix}$

We will assume a point of expansion, $\mathbf{x}^k = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$, at which $\nabla f^k = \begin{bmatrix} -4 \\ -1 \end{bmatrix}$. (It may not be apparent, but if we are approximating a quadratic, it doesn't matter what point of expansion we assume. The Taylor expansion will be exact regardless of the point we pick.)

The example of (3.7), as a Taylor representation, becomes,

$$f(\mathbf{x}) = 12 + [-4,\ -1]\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^T\begin{bmatrix} 4 & 3 \\ 3 & 2 \end{bmatrix}\Delta\mathbf{x} \tag{3.11}$$

where,

$$\Delta \mathbf{x} = \begin{bmatrix} x_1 + 2 \\ x_2 - 2 \end{bmatrix}$$

These three representations are equivalent. If we pick the point $\mathbf{x}^T = \begin{bmatrix} 1.0 & 2.0 \end{bmatrix}$, all three representations give $f = 18$, as you can verify by substitution.

3.2.2 Characteristics of Quadratic Functions

It is useful to note the following characteristics of quadratic equations:

- The equations for the gradient vector of a quadratic function are linear. This makes it easy to solve for where the gradient is equal to zero.
- The Hessian for a quadratic function is a matrix of constants (so we will write as **H** or $\nabla^2 f$ instead of $\mathbf{H}(\mathbf{x})$ or $\nabla^2 f(\mathbf{x})$). Thus the curvature of a quadratic is everywhere the same.
- Excluding the cases where we have a semi-definite Hessian, quadratic functions have only one *stationary point*, i.e. only one point where the gradient is zero.
- Given the gradient and Hessian at some point $\mathbf{x}^k$, the gradient at some other point, $\mathbf{x}^{k+1}$, is given by,

$$\nabla f^{k+1} = \nabla f^k + \mathbf{H}\left(\mathbf{x}^{k+1} - \mathbf{x}^k\right) \tag{3.12}$$

This expression is developed in the Appendix, Section 3.11.1, by differentiating a Taylor expansion in vector form.

- Given the gradient at some point $\mathbf{x}^k$, Hessian, **H**, and a search direction, **s**, the optimal step length, α^*, in the direction **s** is given by,

$$\alpha^* = -\frac{\left(\nabla f^k\right)^T \mathbf{s}}{\mathbf{s}^T \mathbf{H} \mathbf{s}} \tag{3.13}$$

This expression is derived in the Appendix in Section 3.11.2.

- Some of the best methods of optimization are methods of *conjugate directions*. A method of conjugate directions will solve for the optimum of a quadratic function of *n* variables in *n* steps, providing minimizing steps are taken in each search direction. We will learn more about these methods in sections which follow.

3.2.3 Examples

We start with the example,

$$f(\mathbf{x}) = 4x_1 + 2x_2 + x_1^2 - 4x_1x_2 + x_2^2 \tag{3.14}$$

Since this is a quadratic, we know it only has one stationary point. We note that the Hessian,

$$\mathbf{H}=\begin{bmatrix} 2 & -4 \\ -4 & 2 \end{bmatrix}$$

is indefinite (eigenvalues are –0.16 and 6.1). This means we should have a saddle point. The contour plots in Fig 3.2 and Fig. 3.3 confirm this.

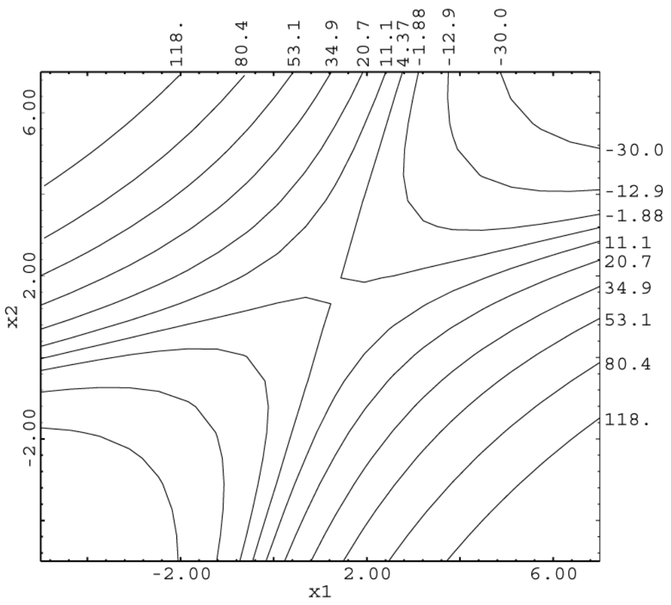

Fig. 3.2 Contour plot of Eq (3.14).

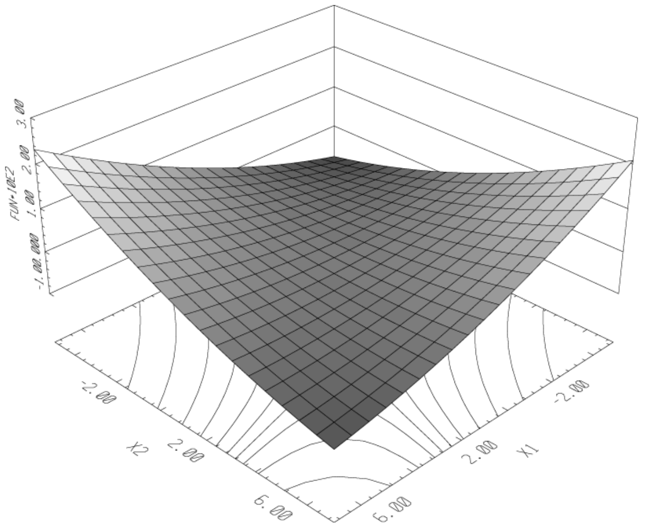

Fig. 3.3. 3D contour plot of (3.14).

We will do a second example. Suppose we have the function,

$$f(\mathbf{x})=x_1+2x_2+4x_1^2-x_1x_2+2x_2^2 \tag{3.15}$$

$$\nabla f = \begin{bmatrix} 1+8x_1 - x_2 \\ 2 - x_1 + 4x_2 \end{bmatrix} \text{ and } \mathbf{H} = \begin{bmatrix} 8 & -1 \\ -1 & 4 \end{bmatrix}$$

By inspection, we see that the determinants of the principal minor matrices are all positive. Thus this function should have a min and look like a bowl. The contour plots follow.

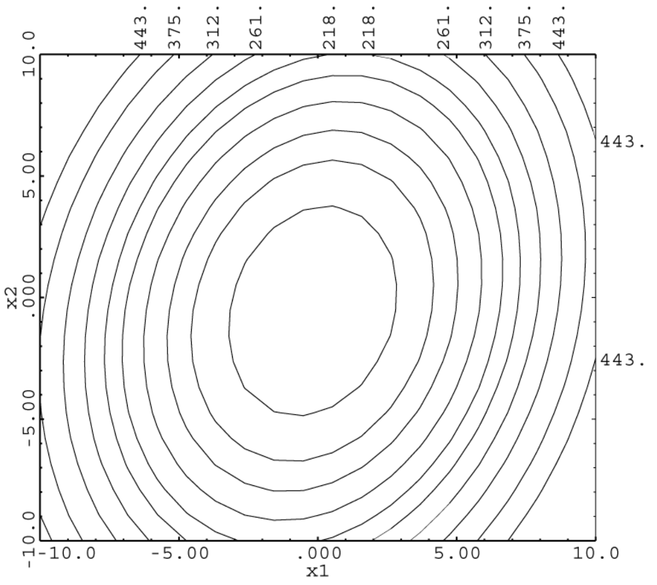

Fig. 3.4. Contour plot for (3.15)

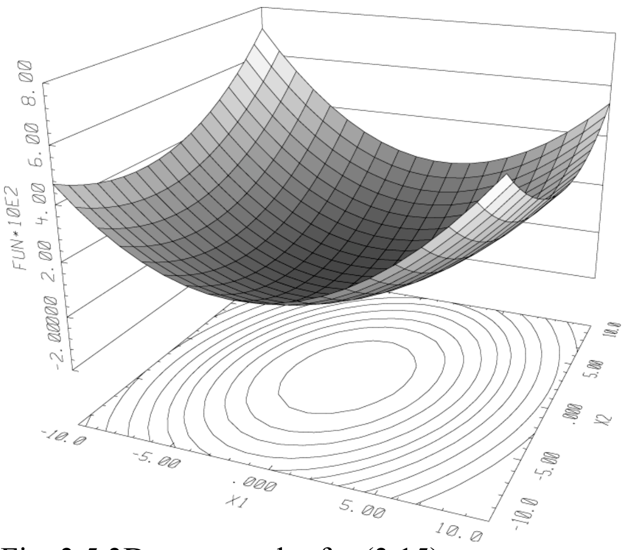

Fig. 3.5 3D contour plot for (3.15)

3.3 Necessary and Sufficient Conditions for an Unconstrained Optimum

With some preliminaries out of the way, we are now ready to begin discussing the theory of unconstrained optimization of differentiable functions. We start with the mathematical conditions which must hold at an unconstrained, local optimum.

3.3.1 Definitions

3.3.1.1 Necessary Conditions for an Unconstrained Optimum

The necessary conditions for an unconstrained optimum at $\mathbf{x}^*$ are,

$$\nabla f(\mathbf{x}^*) = 0 \text{ and } f(\mathbf{x}) \text{ be differentiable at } \mathbf{x}^* \tag{3.16}$$

These conditions are *necessary* but not *sufficient*, inasmuch as $\nabla f(\mathbf{x}) = 0$ can apply at a max, min or a saddle point. However, if at a point $\nabla f(\mathbf{x}) \neq 0$, then that point *cannot* be an optimum.

3.3.1.2 Sufficient Conditions for a Minimum

The sufficient conditions include the necessary conditions but add other conditions such that we *know* we have an optimum. For a minimum,

$$\nabla f(\mathbf{x}^*) = 0,\ f(\mathbf{x}) \text{ twice differentiable at } \mathbf{x}^*,$$
$$\text{plus } \nabla^2 f(\mathbf{x}^*) \text{ is positive definite.} \tag{3.17}$$

3.3.1.3 Sufficient Conditions for a Maximum

For a maximum,

$$\nabla f(\mathbf{x}^*) = 0,\ f(\mathbf{x}) \text{ twice differentiable at } \mathbf{x}^*,$$
$$\text{plus } \nabla^2 f(\mathbf{x}^*) \text{ is negative definite.} \tag{3.18}$$

3.3.2 Examples: Applying the Necessary, Sufficient Conditions

Apply the necessary and sufficient conditions to find the optimum for the quadratic function,

$$f(\mathbf{x}) = x_1^2 - 2x_1x_2 + 4x_2^2$$

Since this is a quadratic function, the partial derivatives will be linear equations. We can solve these equations directly for a point that satisfies the necessary conditions. The gradient vector is,

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 - 2x_2 \\ -2x_1 + 8x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

When we solve these two equations, we have a solution, $x_1 = 0$, $x_2 = 0$--this is a point where the gradient is equal to zero. This represents a minimum, a maximum, or a saddle point. At this point, the Hessian is,

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

Since this Hessian is positive definite (eigenvalues are 1.4, 8.6), this must be a minimum.

As a second example, apply the necessary and sufficient conditions to find the optimum for the quadratic function,

$$f(\mathbf{x}) = 4x_1 + 2x_2 + x_1^2 - 4x_1x_2 + x_2^2$$

As in example 1, we will solve the gradient equations directly for a point that satisfies the necessary conditions. The gradient vector is,

$$\begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 - 4x_2 + 4 \\ -4x_1 + 2x_2 + 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

When we solve these two equations, we have a solution, $x_1 = 1.333$, $x_2 = 1.667$. The Hessian is,

$$\mathbf{H} = \begin{bmatrix} 2 & -4 \\ -4 & 2 \end{bmatrix}$$

The eigenvalues are -2, 6. The Hessian is indefinite. This means this is neither a max nor a min—it is a saddle point.

<u>Comments</u>: As mentioned, the equations for the gradient of a quadratic function are linear, so they are easy to solve. Obviously we don't usually have a quadratic objective, so the equations are usually not linear. Often we will use the necessary conditions to *check* a point to see if we are at an optimum. Some algorithms, however, solve for an optimum by solving directly where the necessary conditions are satisfied.

Other algorithms search for the optimum by taking downhill steps and continuing until they can go no further. In the next section we will study one of the simplest unconstrained algorithms that steps downhill: steepest descent.

3.4 Steepest Descent with a Quadratic Line Search

3.4.1 Description

One of the simplest unconstrained optimization methods is steepest descent. Given an initial starting point, the algorithm moves downhill until it can go no further.

The search can be broken down into stages. For any algorithm, at each stage (or iteration) we must determine two things:

1. What should the search direction be?
2. How far should we go in that direction?

Answer to question 1: For the method of steepest descent, the search direction is $-\nabla f(\mathbf{x})$

Answer to question 2: A *line search* is performed. "Line" in this case means we search along a direction vector. The line search strategy presented here, bracketing the function with quadratic fit, is one of many that have been proposed, and is one of the most common. Later in the chapter we will look at other line search methods.

General Approach for each step:
Given some starting point, $\mathbf{x}^k$, we wish to determine,

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha\mathbf{s} \tag{3.19}$$

where **s** is the search direction vector, usually normalized, and α is the step length, a scalar.

We will step in direction **s** with increasing values of α until the function begins to increase. Then we will curve fit the data with a parabola, and step to the minimum of the parabola.

3.4.2 Example: Steepest Descent with Quadratic Line Search

Min $f(\mathbf{x}) = x_1^2 - 2x_1x_2 + 4x_2^2 \qquad f^0 = 19$

starting at $\mathbf{x}^0 = \begin{bmatrix} -3 \\ 1 \end{bmatrix} \qquad -\nabla f^0 = \begin{bmatrix} 8 \\ -14 \end{bmatrix} \quad \mathbf{s}^0 = \begin{bmatrix} 8 \\ -14 \end{bmatrix}$

normalized $\mathbf{s}^0 = \begin{bmatrix} 0.50 \\ -0.86 \end{bmatrix} \qquad \mathbf{x}^1 = \begin{bmatrix} -3 \\ 1 \end{bmatrix} + \alpha \begin{bmatrix} 0.50 \\ -0.86 \end{bmatrix}$

For generality, we will find α^*, the optimal step length, by trial and error, although, since this is a quadratic, we could compute it directly using (3.13).

Guess $\alpha^* = .4$ for step number 1:

Line Search Step	α	$\mathbf{x}^1 = \mathbf{x}^0 + \alpha \mathbf{s}^0$	$f(\mathbf{x})$
1	0.4	$\mathbf{x}^1 = \begin{bmatrix} -3.0 \\ 1.0 \end{bmatrix} + .4 \begin{bmatrix} 0.50 \\ -0.86 \end{bmatrix} = \begin{bmatrix} -2.80 \\ 0.66 \end{bmatrix}$	13.3

We see that the function has decreased; we decide to double the step length and continue doubling until the function begins to increase:

Line Search Step	α	$\mathbf{x}^1 = \mathbf{x}^0 + \alpha \mathbf{s}^0$	$f(\mathbf{x})$
2	0.8	$\mathbf{x}^1 = \begin{bmatrix} -3.0 \\ 1.0 \end{bmatrix} + .8 \begin{bmatrix} 0.50 \\ -0.86 \end{bmatrix} = \begin{bmatrix} -2.60 \\ 0.31 \end{bmatrix}$	8.75
3	1.6	$\mathbf{x}^1 = \begin{bmatrix} -3.0 \\ 1.0 \end{bmatrix} + 1.6 \begin{bmatrix} 0.50 \\ -0.86 \end{bmatrix} = \begin{bmatrix} -2.20 \\ -0.38 \end{bmatrix}$	3.74
4	3.2	$\mathbf{x}^1 = \begin{bmatrix} -3.0 \\ 1.0 \end{bmatrix} + 3.2 \begin{bmatrix} 0.50 \\ -0.86 \end{bmatrix} = \begin{bmatrix} -1.40 \\ -1.75 \end{bmatrix}$	9.31

The objective function has started to increase—we have gone too far.

We will cut the change in the last step by half:

5	2.4	$\mathbf{x}^1 = \begin{bmatrix} -3.0 \\ 1.0 \end{bmatrix} + 2.4 \begin{bmatrix} 0.50 \\ -0.86 \end{bmatrix} = \begin{bmatrix} -1.80 \\ -1.06 \end{bmatrix}$	3.91

A graph of our progress is shown in Fig. 3.6:

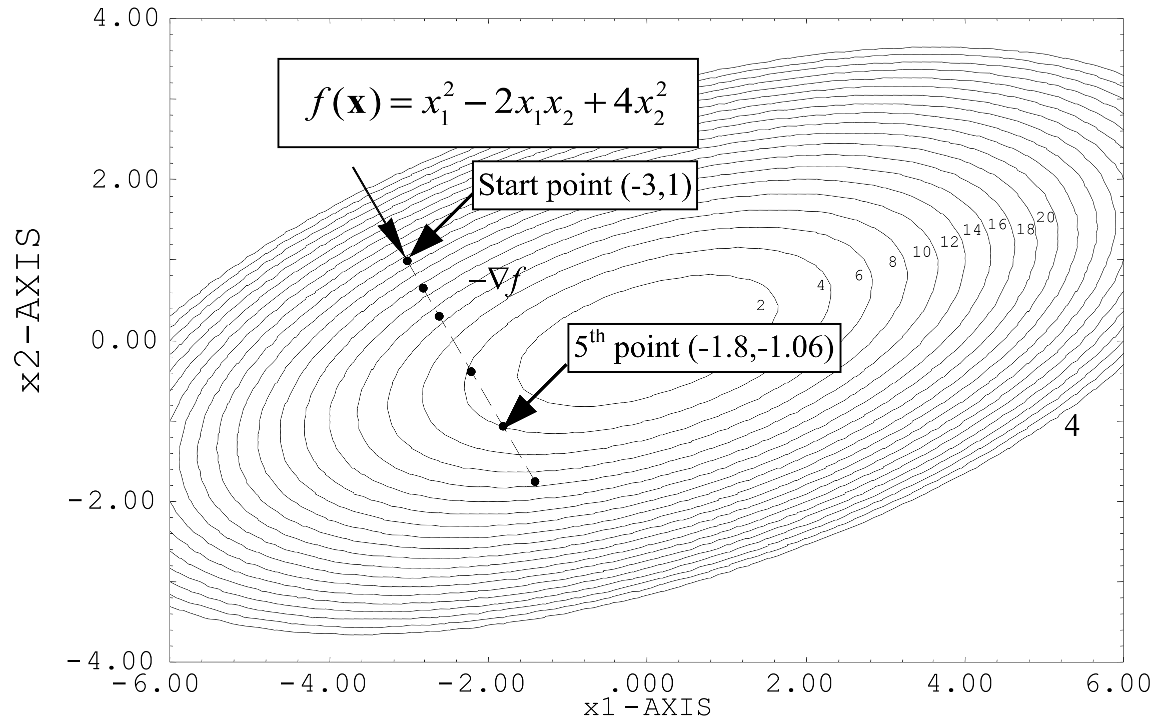

Fig. 3.6 Progress in the line search shown on a contour plot.

If we plot the objective value as a function of step length, as shown in Fig 3.7, we have,

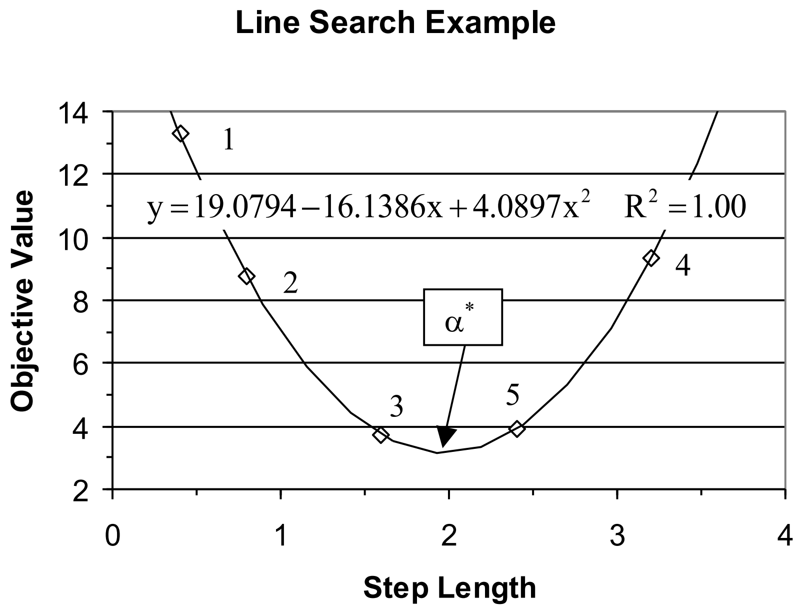

Fig. 3.7 The objective value vs. step length for the line search.

We see that the data plot up to be a parabola. We would like to estimate the minimum of this curve. We will curve fit points 2, 5, 3. These points are equally spaced and bracket the minimum.

||2 ||3 ||5

Renumbering these points as α_1, α_2, α_3 the minimum of the parabola is given by

$$\alpha^* = \alpha_2 + \frac{\Delta\alpha\left[f(\alpha_1) - f(\alpha_3)\right]}{2\left[f(\alpha_1) - 2f(\alpha_2) + f(\alpha_3)\right]}$$

$$\alpha^* = 1.60 + \frac{(0.8)\left[8.75 - 3.91\right]}{2\left[8.75 - 2(3.74) + 3.91\right]} \tag{3.20}$$

$$\alpha^* = 1.97$$

where $f(\mathbf{x}) = 3.2$

When we step back, after the function has become worse, we have four points to choose from (points 2, 3, 5, 4). How do we know which three to pick to make sure we don't lose the bracket on the minimum? The rule is this: take the point with the lowest function value (point 3) and the two points to either side (points 2 and 5).

In summary, the line search consists of stepping along the search direction until the minimum of the function in this direction is bracketed, fitting three points which bracket the minimum with a parabola, and calculating the minimum of the parabola. If necessary the parabolic fit can be carried out several times until the change in the minimum is very small (although the α are then no longer equally spaced, so the following formula must be used):

$$\alpha^* = \frac{f(\alpha_1)(\alpha_2^2 - \alpha_3^2) + f(\alpha_2)(\alpha_3^2 - \alpha_1^2) + f(\alpha_3)(\alpha_1^2 - \alpha_2^2)}{2\left[f(\alpha_1)(\alpha_2 - \alpha_3) + f(\alpha_2)(\alpha_3 - \alpha_1) + f(\alpha_3)(\alpha_1 - \alpha_2)\right]} \tag{3.21}$$

Each sequence of obtaining the gradient and moving along the negative gradient direction until a minimum is found (i.e. executing a line search) is called an *iteration*. The algorithm consists of executing iterations until the norm of the gradient drops below a specified tolerance, indicating the necessary conditions have been met.

As shown in Fig. 3.7, at α^*, $\frac{df}{d\alpha} = 0$. The process of determining α^* will be referred to as *taking a minimizing step*, or, *executing an exact line search.*

3.4.3 Pros and Cons of Steepest Descent

Steepest descent has several advantages. It usually makes good progress when far from the optimum (in the above example the objective decreased from 19 to 3 in the first iteration), and it is very simple to implement. It always goes downhill. It is also guaranteed to converge to a local optimum if enough steps are taken.

However, if the function to be minimized is *eccentric*, convergence of steepest descent can be very slow, as indicated by the following theorem from Luenberger and Ye [4].

THEOREM. Convergence of Steepest Descent. For a quadratic function, if we take enough steps, the method of steepest descent converges to the unique minimum point $\mathbf{x}^*$ of *f*. If we define the error in the objective function at the current value of **x** as,

$$E(\mathbf{x}) = \frac{1}{2}\left(\mathbf{x}-\mathbf{x}^*\right)^{\mathrm{T}} \mathbf{H}\left(\mathbf{x}-\mathbf{x}^*\right) \tag{3.22}$$

there holds at every step *k*,

$$E\left(\mathbf{x}^{k+1}\right) \le \left(\frac{A-a}{A+a}\right)^2 E\left(\mathbf{x}^k\right)$$

$$\text{where} \tag{3.23}$$

$$A = \text{Largest eigenvalue of } \mathbf{H}$$

$$a = \text{Smallest eigenvalue of } \mathbf{H}$$

Thus if A=50 and a=1, we have that the error at the *k*+1 step is only guaranteed to be less than the error at the *k* step by,

$$E^{k+1} \le \left(\frac{49}{51}\right)^2 E^k$$

and thus the error may be reduced very slowly.

"Roughly speaking, the above theorem says that the convergence rate of steepest descent is slowed as the contours of *f* become more eccentric. If $a = A$, corresponding to circular contours, convergence occurs in a single step. Note, however, that even if $n-1$ of the *n* eigenvalues are equal and the remaining one is a great distance from these, convergence will be slow, and hence a single abnormal eigenvalue can destroy the effectiveness of steepest descent."

As an example, the function,

$$f = x_1^2 + x_2^2$$

has equal eigenvalues of (2, 2) and circular contours. Steepest descent converges in one step, as shown in Fig. 3.8. However, consider the function,

$$f = 2x_1^2 + 8x_2^2$$

which has eigenvalues of (4, 16). Even though the contours are only mildly eccentric, the zigzag pattern of steepest descent is clearly evident in Fig 3.9.

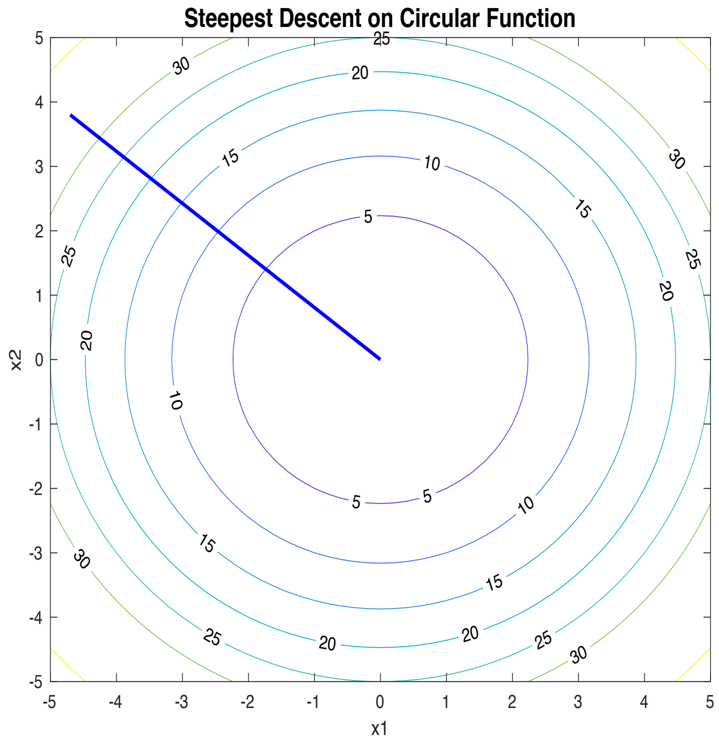

Fig. 3.8. Steepest Descent on function with circular contours.

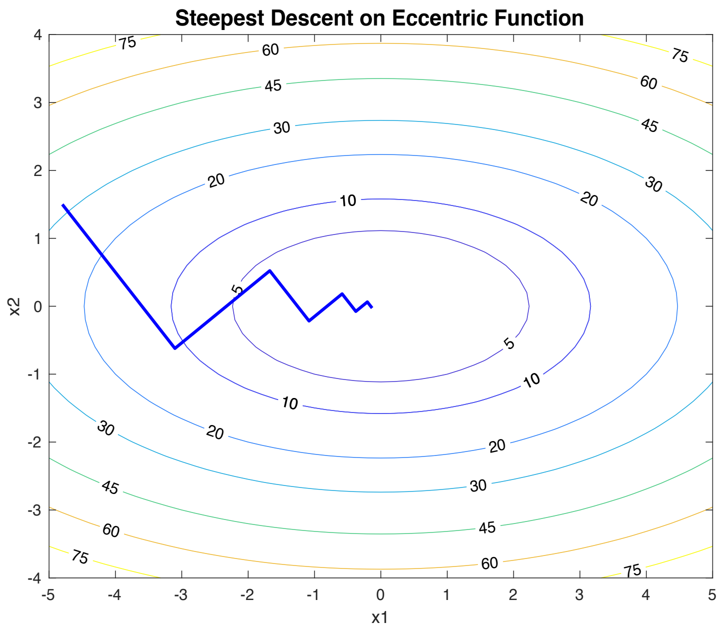

Fig. 3.9. Steepest Descent on a mildly eccentric function.

3.5 The Directional Derivative

It is sometimes useful to calculate $\frac{df}{d\alpha}$ along some search direction **s.** From the chain rule for differentiation,

$$\frac{df}{d\alpha} = \sum \left(\frac{\partial f}{\partial x_i}\right)\left(\frac{dx_i}{d\alpha}\right)$$

Noting that $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha \mathbf{s}$, or, for a single element of vector **x**, $x_i^{k+1} = x_i^k + \alpha s_i^k$, we have $\frac{dx_i}{d\alpha} = s_i$, so

$$\frac{df}{d\alpha} = \sum \left(\frac{\partial f}{\partial x_i}\right)\left(\frac{dx_i}{d\alpha}\right) = \sum \left(\frac{\partial f}{\partial x_i}\right) s_i = \nabla f^{\mathrm{T}} \mathbf{s} \tag{3.24}$$

As an example, we will find the directional derivative, $\frac{df}{d\alpha}$, for the problem given in Section 3.4.2 above, at α =0. From (3.24): $\frac{df}{d\alpha} = \nabla f^{\mathrm{T}} \mathbf{s} = \begin{bmatrix} -8 & 14 \end{bmatrix} \begin{bmatrix} 0.5 \\ -0.86 \end{bmatrix} = -16.04$

This gives us the change in the function for a small step in the search direction, i.e.,

$$\Delta f \approx \frac{df}{d\alpha} \Delta \alpha \tag{3.25}$$

If $\Delta\alpha = 0.01$, the predicted change is 0.161. The actual change in the function is 0.160.

Equation (3.24) is the same equation for checking if a direction goes downhill, given in Section 3.1.4. Before we just looked at the sign; if negative we knew we were going downhill. Now we see that the value represents the expected change in the function for a small step in the search direction **s**. If, for example, the value of $\left.\frac{df}{d\alpha}\right|_{\alpha=0}$ is less than some epsilon, we could terminate the line search, because the predicted change in the objective function is below a minimum threshold.

Another important value of $\frac{df}{d\alpha}$ occurs at α^*. If we locate the minimum exactly, then

$$\left.\frac{df}{d\alpha}\right|_{\alpha=\alpha^*} = \left(\nabla f^{k+1}\right)^{\mathrm{T}} \mathbf{s}^k = 0 \tag{3.26}$$

As we have seen in examples, when we take a minimizing step we stop where the search direction is tangent to the contours of the function. Thus the gradient at this new point is orthogonal to the previous search direction.

3.6 Newton's Method

3.6.1 Derivation

Another classical method we will study is called Newton's method. It simply makes a quadratic approximation to a function at the current point and solves for where the necessary conditions (to the approximation) are satisfied. Starting with a Taylor series:

$$f^{k+1} = f^k + \left(\nabla f^k\right)^{\mathrm{T}} \Delta\mathbf{x}^k + \frac{1}{2}\left(\Delta\mathbf{x}^k\right)^{\mathrm{T}} \mathbf{H}^k \Delta\mathbf{x}^k \tag{3.27}$$

Since the gradient and Hessian are evaluated at *k*, they are just a vector and matrix of constants. Taking the gradient (See the Appendix),

$$\nabla f^{k+1} = \nabla f^k + \mathbf{H}^k \Delta\mathbf{x}^k$$

and setting $\nabla f^{k+1} = 0$, we have,

$$\mathbf{H}^k \Delta\mathbf{x}^k = -\nabla f^k$$

Solving for $\Delta\mathbf{x}$:

$$\Delta\mathbf{x}^k = -\left(\mathbf{H}^k\right)^{-1} \nabla f^k \tag{3.28}$$

Note that we have solved for a vector, i.e. $\Delta\mathbf{x}$, which has both a step length and direction.

3.6.2 Example: Newton's Method

We wish to optimize the function, $f(\mathbf{x}) = x_1^2 - 2x_1x_2 + 4x_2^2$ from the point $\mathbf{x}^0 = \begin{bmatrix} -3 \\ 1 \end{bmatrix}$.

At this point $\nabla f^0 = \begin{bmatrix} -8 \\ 14 \end{bmatrix}$ and the Hessian is, $\mathbf{H} = \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix}$. The Hessian inverse is given by: $\begin{bmatrix} 0.6667 & 0.16667 \\ 0.16667 & 0.16667 \end{bmatrix}$. Thus $\Delta\mathbf{x} = -\begin{bmatrix} 0.6667 & 0.16667 \\ 0.16667 & 0.16667 \end{bmatrix}\begin{bmatrix} -8 \\ 14 \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$

So, $\mathbf{x}^1 = \mathbf{x}^0 + \Delta\mathbf{x} = \begin{bmatrix} -3 \\ 1 \end{bmatrix} + \begin{bmatrix} 3 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

This step is shown in Fig. 3.10.

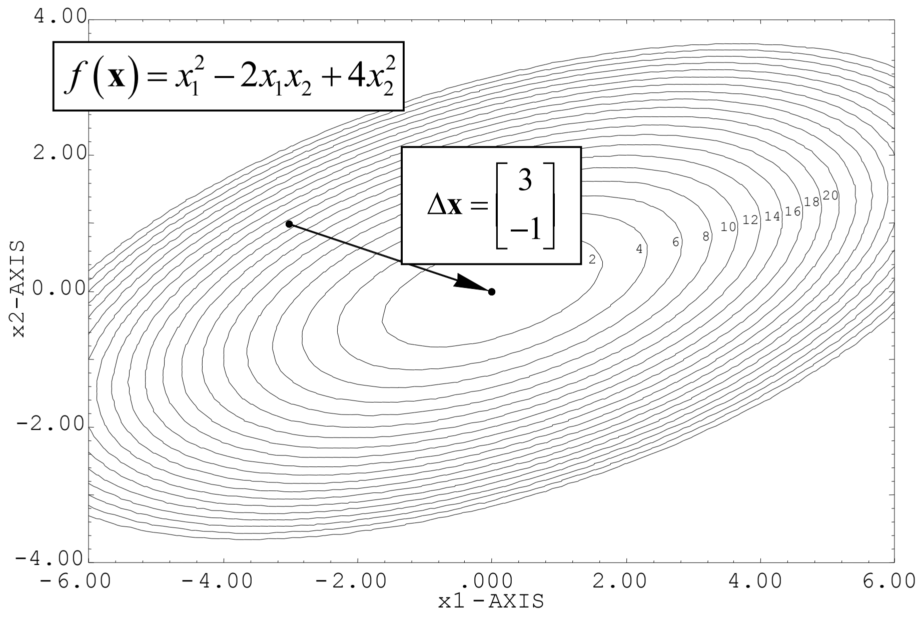

Fig. 3.10. The operation of Newton's method.

3.6.3 Pros and Cons of Newton's Method

We can see in the above example Newton's method solved the problem in one step. This is true in general: Newton's method will drive to the stationary point of a quadratic in one step. On a non-quadratic, if we are near an optimum, Newton's method will drive very close to the optimum in one step.

However we should note some drawbacks. First, it requires second derivatives. Computing second derivatives can be computationally expensive and we typically avoid this if we can.

Second, the derivation of Newton's method solved for where the gradient is equal to zero. The gradient is equal to zero at a min, a max or a saddle, and nothing in the method differentiates between these. Thus Newton's method can diverge, or fail to go downhill (indeed, not only not go downhill, but go to a maximum). This is obviously a serious drawback.

3.7 Quasi-Newton Methods

3.7.1 Introduction

Let's summarize the pros and cons of Newton's method and Steepest Descent:

	Pros	**Cons**
Steepest Descent	Always goes downhill Always converges Simple to implement	Slow on eccentric functions
Newton's Method	Solves quadratic in one step. Very fast when close to optimum on non-quadratic.	Requires second derivatives, Can diverge

We want to develop a method that starts out like steepest descent and gradually becomes Newton's method, doesn't need second derivatives, doesn't have trouble with eccentric functions and doesn't diverge. Fortunately such methods exist. They combine the good aspects of steepest descent and Newton's method without the drawbacks. These methods are called *quasi-Newton* methods.

In general we will define our search direction by the expression

$$\mathbf{s} = -\mathbf{N}\nabla f(\mathbf{x}) \tag{3.29}$$

where **N** will be called the "direction matrix."

If $\mathbf{N} = \mathbf{I}$, then $\mathbf{s} = -\nabla f(\mathbf{x}) \quad \rightarrow \quad \text{Steepest Descent}$

If $\mathbf{N} = \mathbf{H}^{-1}$, then $\mathbf{s} = -\mathbf{H}^{-1}\nabla f(\mathbf{x}) \quad \rightarrow \quad \text{Newton's Method}$

If **N** is always positive definite, then **s** always points downhill. To show this, our criterion for moving downhill is:

$$\mathbf{s}^{\mathrm{T}}\nabla f < 0$$

Or,

$$\nabla f^{\mathrm{T}}\mathbf{s} < 0 \tag{3.30}$$

Substituting (3.29) into (3.30):

$$-\left(\nabla f^{\mathrm{T}}\mathbf{N}\nabla f\right) < 0 \tag{3.31}$$

Since **N** is positive definite, we know that any vector which pre-multiplies **N** and post-multiplies **N** will result in a positive scalar. Thus the quantity within the parentheses is always positive; with the negative sign it becomes always negative, and therefore always goes downhill.

3.7.2 A Rank One Hessian Inverse Update

3.7.2.1 Development

In this section we will develop one of the simplest quasi-Newton methods. This method updates the direction matrix **N** at every iteration with new information. It is called a "rank one" update because the update to the direction matrix is a rank one matrix (i.e., it only has one independent row or column). We first give some preliminaries.

Starting with a Taylor series:

$$f^{k+1} = f^{k} + \left(\nabla f^{k}\right)^{\mathrm{T}} \Delta\mathbf{x}^{k} + \frac{1}{2}\left(\Delta\mathbf{x}^{k}\right)^{\mathrm{T}} \mathbf{H}\, \Delta\mathbf{x}^{k} \tag{3.32}$$

where $$\Delta\mathbf{x}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$$

The gradient is given by,

$$\nabla f^{k+1} = \nabla f^k + \mathbf{H}\Delta\mathbf{x}^k \tag{3.33}$$

Defining:

$$\boldsymbol{\gamma}^{\mathrm{k}} = \nabla f^{k+1} - \nabla f^k \tag{3.34}$$

then we have,

$$\boldsymbol{\gamma}^k = \mathbf{H}\Delta\mathbf{x}^k \quad \mathit{or} \quad \mathbf{H}^{-1}\boldsymbol{\gamma}^k = \Delta\mathbf{x}^k \tag{3.35}$$

Equation (3.35) is very important: it shows that for a quadratic function, the inverse of the Hessian matrix ($\mathbf{H}^{-1}$) maps differences in the gradients to differences in **x**. The relationship expressed by (3.35) is called the *Newton condition*.

We will make the direction matrix satisfy this relationship. However, since we can only calculate $\boldsymbol{\gamma}^k$ and $\Delta\mathbf{x}^k$ after the line search, we will make

$$\mathbf{N}^{k+1}\boldsymbol{\gamma}^k = \Delta\mathbf{x}^k \tag{3.36}$$

This expression is called the *quasi-Newton condition*. It is "quasi" in that it involves *k*+1 for **N** instead of *k*. Expression (3.36) involves more unknowns (the elements of $\mathbf{N}^{k+1}$) than equations, so how do we solve for $\mathbf{N}^{k+1}$?

One of the simplest possibilities is:

$$\mathbf{N}^{k+1} = \mathbf{N}^k + a\mathbf{u}\mathbf{u}^{\mathrm{T}} \tag{3.37}$$

where we update the direction matrix with a correction which is of the form $a\mathbf{u}\mathbf{u}^{\mathrm{T}}$, which is a rank one symmetric matrix.

If we substitute (3.37) into (3.36), we have,

$$\mathbf{N}^k\boldsymbol{\gamma}^k + a\mathbf{u}\mathbf{u}^{\mathrm{T}}\boldsymbol{\gamma}^k = \Delta\mathbf{x}^k \tag{3.38}$$

or

$$a\mathbf{u}\underbrace{\mathbf{u}^{\mathrm{T}}\gamma^k}_{scalar} = \left(\Delta\mathbf{x}^k - \mathbf{N}^k\gamma^k\right) \tag{3.39}$$

Noting that $\mathbf{u}^{\mathrm{T}}\boldsymbol{\gamma}^k$ is a scalar, then **u** must be proportional to $\left(\Delta\mathbf{x}^k - \mathbf{N}^k\boldsymbol{\gamma}^k\right)$. Since any change in length can be absorbed by *a*, we will set

$$\mathbf{u} = \left(\Delta\mathbf{x}^k - \mathbf{N}^k\boldsymbol{\gamma}^k\right) \tag{3.40}$$

Substituting (3.40) into (3.39):

$$\underbrace{a\left(\Delta\mathbf{x}^k - \mathbf{N}^k\gamma^k\right)}_{vector}\underbrace{\left(\Delta\mathbf{x}^k - \mathbf{N}^k\gamma^k\right)^{\mathrm{T}}\gamma^k}_{scalar} = \underbrace{\left(\Delta\mathbf{x}^k - \mathbf{N}^k\gamma^k\right)}_{vector} \tag{3.41}$$

For this to be true,

$$\underbrace{a\left(\Delta\mathbf{x}^k - \mathbf{N}^k\gamma^k\right)^{\mathrm{T}}\gamma^k}_{scalar} = 1$$

so

$$a = \frac{1}{\left(\Delta\mathbf{x}^k - \mathbf{N}^k\boldsymbol{\gamma}^k\right)^{\mathrm{T}}\boldsymbol{\gamma}^k} \tag{3.42}$$

Substituting (3.42) and (3.40) into (3.37) gives the expression we need:

$$\mathbf{N}^{k+1} = \mathbf{N}^k + \frac{\left(\Delta\mathbf{x}^k - \mathbf{N}^k\boldsymbol{\gamma}^k\right)\left(\Delta\mathbf{x}^k - \mathbf{N}^k\boldsymbol{\gamma}^k\right)^{\mathrm{T}}}{\left(\Delta\mathbf{x}^k - \mathbf{N}^k\boldsymbol{\gamma}^k\right)^{\mathrm{T}}\boldsymbol{\gamma}^k} \tag{3.43}$$

Equation (3.43) allows us to get a new direction matrix in terms of the previous matrix and differences in **x** and the gradient vector. We then use this to get a new search direction according to (3.29).

3.7.2.2 Example: Rank One Hessian Inverse Update

We wish to minimize the function $f\left(\mathbf{x}\right) = x_1^2 - 2x_1x_2 + 4x_2^2$

starting from $\mathbf{x}^0 = \begin{bmatrix} -3 \\ 1 \end{bmatrix} \quad \nabla f^0 = \begin{bmatrix} -8 \\ 14 \end{bmatrix}$

We let $\mathbf{N}^0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ so the search direction is

$$\mathbf{s}^0 = -\mathbf{N}\nabla f^0 = -\nabla f^0$$

We normalize the search direction to be: $\mathbf{s}^0 = \begin{bmatrix} 0.496 \\ -0.868 \end{bmatrix}$

We execute a line search in this direction (using, for example, a quadratic fit) and stop at

$$\mathbf{x}^1 = \begin{bmatrix} -2.030 \\ -0.698 \end{bmatrix} \quad \nabla f^1 = \begin{bmatrix} -2.664 \\ -1.522 \end{bmatrix}$$

Then $$\Delta\mathbf{x}^0 = \mathbf{x}^1 - \mathbf{x}^0 = \begin{bmatrix} -2.030 \\ -0.698 \end{bmatrix} - \begin{bmatrix} -3.000 \\ 1.000 \end{bmatrix} = \begin{bmatrix} 0.970 \\ -1.698 \end{bmatrix}$$

$$\boldsymbol{\gamma}^0 = \nabla f^1 - \nabla f^0 = \begin{bmatrix} -2.664 \\ -1.522 \end{bmatrix} - \begin{bmatrix} -8.000 \\ 14.000 \end{bmatrix} = \begin{bmatrix} 5.336 \\ -15.522 \end{bmatrix}$$

and
$$\Delta \mathbf{x}^0 - \mathbf{N}^0 \boldsymbol{\gamma}^0 = \begin{bmatrix} 0.970 \\ -1.698 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 5.336 \\ -15.522 \end{bmatrix} = \begin{bmatrix} -4.366 \\ 13.824 \end{bmatrix}$$

$$a\mathbf{u}\mathbf{u}^T = \frac{\left(\Delta \mathbf{x}^0 - \mathbf{N}^0 \boldsymbol{\gamma}^0\right)\left(\Delta \mathbf{x}^0 - \mathbf{N}^0 \boldsymbol{\gamma}^0\right)^{\mathrm{T}}}{\left(\Delta \mathbf{x}^0 - \mathbf{N}^0 \boldsymbol{\gamma}^0\right)^{\mathrm{T}} \boldsymbol{\gamma}^0} = \frac{\begin{bmatrix} -4.366 \\ 13.824 \end{bmatrix} \begin{bmatrix} -4.366 & 13.824 \end{bmatrix}}{\begin{bmatrix} -4.366 & 13.824 \end{bmatrix} \begin{bmatrix} 5.336 \\ -15.522 \end{bmatrix}}$$

$$= \frac{\begin{bmatrix} 19.062 & -60.364 \\ -60.364 & 191.158 \end{bmatrix}}{-237.932}$$

$$= \begin{bmatrix} -0.080 & 0.254 \\ 0.254 & -0.803 \end{bmatrix}$$

$$\mathbf{N}^1 = \mathbf{N}^0 + a\mathbf{u}\mathbf{u}^T$$

$$\mathbf{N}^1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} -0.080 & 0.254 \\ 0.254 & -0.803 \end{bmatrix}$$

$$\mathbf{N}^1 = \begin{bmatrix} 0.920 & 0.254 \\ 0.254 & 0.197 \end{bmatrix}$$

New search direction:

$$\mathbf{s}^1 = -\begin{bmatrix} 0.920 & 0.254 \\ 0.254 & 0.197 \end{bmatrix} \begin{bmatrix} -2.664 \\ -1.522 \end{bmatrix}$$

$$= \begin{bmatrix} 2.837 \\ 0.975 \end{bmatrix}$$

When we step in this direction, using again a line search, we arrive at the optimum

$$\mathbf{x}^2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \nabla f\left(\mathbf{x}^2\right) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

At this point we are done. However, if we update the direction matrix one more time, *we find it has become the inverse Hessian*. We will explain how this happens in the next section.

$$\Delta\mathbf{x}^1 = \mathbf{x}^2 - \mathbf{x}^1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} -2.030 \\ -0.698 \end{bmatrix} = \begin{bmatrix} 2.030 \\ 0.698 \end{bmatrix}$$

$$\boldsymbol{\gamma}^1 = \nabla f^2 - \nabla f^1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} -2.664 \\ -1.524 \end{bmatrix} = \begin{bmatrix} 2.664 \\ 1.524 \end{bmatrix}$$

$$\left(\Delta\mathbf{x}^1 - \mathbf{N}^1\gamma^1\right) = \begin{bmatrix} 2.030 \\ 0.698 \end{bmatrix} - \begin{bmatrix} 0.920 & 0.254 \\ 0.254 & 0.197 \end{bmatrix}\begin{bmatrix} 2.664 \\ 1.524 \end{bmatrix}$$

$$= \begin{bmatrix} 2.030 \\ 0.698 \end{bmatrix} - \begin{bmatrix} 2.838 \\ 0.977 \end{bmatrix} = \begin{bmatrix} -0.808 \\ -0.279 \end{bmatrix}$$

$$a\mathbf{u}\mathbf{u}^{\mathrm{T}} = \frac{\left(\Delta\mathbf{x}^1 - \mathbf{N}^1\boldsymbol{\gamma}^1\right)\left(\Delta\mathbf{x}^1 - \mathbf{N}^1\boldsymbol{\gamma}^1\right)^{\mathrm{T}}}{\left(\Delta\mathbf{x}^1 - \mathbf{N}^1\boldsymbol{\gamma}^1\right)^{\mathrm{T}}\boldsymbol{\gamma}^1} = \frac{\begin{bmatrix} -0.808 \\ -0.279 \end{bmatrix}\begin{bmatrix} -0.808 & -0.279 \end{bmatrix}}{\begin{bmatrix} -0.808 & -0.279 \end{bmatrix}\begin{bmatrix} 2.664 \\ 1.524 \end{bmatrix}} = \begin{bmatrix} -0.253 & -0.088 \\ -0.088 & -0.030 \end{bmatrix}$$

$$\mathbf{N}^2 = \mathbf{N}^1 + a\mathbf{u}\mathbf{u}^{\mathrm{T}} = \begin{bmatrix} 0.920 & 0.254 \\ 0.254 & 0.197 \end{bmatrix} + \begin{bmatrix} -0.253 & -0.088 \\ -0.088 & -0.030 \end{bmatrix} = \begin{bmatrix} 0.667 & 0.166 \\ 0.166 & 0.167 \end{bmatrix} = \mathbf{H}^{-1}$$

3.7.2.3 The Hereditary Property

The hereditary property is an important property of all update methods. The hereditary property states that not only will $\mathbf{N}^{k+1}$ satisfy (3.36), but

$$\begin{aligned} \mathbf{N}^{k+1}\boldsymbol{\gamma}^{k} &= \Delta\mathbf{x}^{k} \\ \mathbf{N}^{k+1}\boldsymbol{\gamma}^{k-1} &= \Delta\mathbf{x}^{k-1} \\ \mathbf{N}^{k+1}\boldsymbol{\gamma}^{k-2} &= \Delta\mathbf{x}^{k-2} \\ \mathbf{N}^{k+1}\boldsymbol{\gamma}^{k-n+1} &= \Delta\mathbf{x}^{k-n+1} \end{aligned} \tag{3.44}$$

where *n* is the number of variables. That is, (3.36) is not only satisfied for the current step, but for the *last n-1 steps.* Why is this significant? Let's write the relationship of (3.44) as follows:

$$\mathbf{N}^{k+1}\left[\gamma^{k}, \gamma^{k-1}, \gamma^{k-2} \; . \; . \; . \; \gamma^{k-n+1}\right] = \left[\Delta\mathbf{x}^{k}, \Delta\mathbf{x}^{k-1}, \Delta\mathbf{x}^{k-2}, \ldots, \Delta\mathbf{x}^{k-n+1}\right]$$

Let the matrix defined by the columns of $\boldsymbol{\gamma}$ be denoted by **G**, and the matrix defined by columns of $\Delta\mathbf{x}$ be denoted by $\Delta\mathbf{X}$. Then,

$$\mathbf{N}^{k+1}\mathbf{G} = \Delta\mathbf{X}$$

If $\gamma^k \ldots \gamma^{k-n+1}$ are independent, and if we have *n* vectors, i.e. **G** is a square matrix, then the inverse for **G** exists and is unique and,

$$\mathbf{N}^{k+1} = \Delta\mathbf{X}\mathbf{G}^{-1} \tag{3.45}$$

is uniquely defined.

Since the Hessian inverse satisfies (3.44) for a quadratic function, then we have the important result that, *after n updates the direction matrix becomes the Hessian inverse for a quadratic function*. This implies the quasi-Newton method will solve a quadratic in no more than *n+1* steps.

3.7.2.4 The Hereditary Property for the Rank One Update

In this section we will show the Hereditary property holds for a specific example. The general proof is given in the Appendix.

Suppose we look at the first two updates of a problem. The quasi-Newton condition requires that,

$$\mathbf{N}^1\gamma^0 = \Delta\mathbf{x}^0 \tag{3.46}$$

as well as,

$$\mathbf{N}^2\gamma^1 = \Delta\mathbf{x}^1$$

The question is, is the following true (as suggested by the Hereditary property)?

$$\mathbf{N}^2\gamma^0 = \Delta\mathbf{x}^0 \tag{3.47}$$

The second update is given by,

$$\mathbf{N}^2 = \mathbf{N}^1 + \frac{\left(\Delta\mathbf{x}^1 - \mathbf{N}^1\gamma^1\right)\left(\Delta\mathbf{x}^1 - \mathbf{N}^1\gamma^1\right)^{\mathrm{T}}}{\left(\Delta\mathbf{x}^1 - \mathbf{N}^1\gamma^1\right)^{\mathrm{T}}\gamma^1} \tag{3.48}$$

We can post multiply (3.48) by γ^0, giving,

$$\mathbf{N}^2\gamma^0 = \mathbf{N}^1\gamma^0 + \frac{\left(\Delta\mathbf{x}^1 - \mathbf{N}^1\gamma^1\right)\left(\Delta\mathbf{x}^1 - \mathbf{N}^1\gamma^1\right)^{\mathrm{T}}\gamma^0}{\left(\Delta\mathbf{x}^1 - \mathbf{N}^1\gamma^1\right)^{\mathrm{T}}\gamma^1} \tag{3.49}$$

To simplify things, let $\mathbf{y}^1 = \dfrac{\left(\Delta\mathbf{x}^1 - \mathbf{N}^1\gamma^1\right)}{\left(\Delta\mathbf{x}^1 - \mathbf{N}^1\gamma^1\right)^{\mathrm{T}}\gamma^1}$ so that we can write (3.49) as,

$$\mathbf{N}^2\gamma^0 = \mathbf{N}^1\gamma^0 + \mathbf{y}^1\left(\Delta\mathbf{x}^1 - \mathbf{N}^1\gamma^1\right)^{\mathrm{T}}\gamma^0 \tag{3.50}$$

We can distribute the transpose on the last term, and distribute the post multiplication γ^0 to give,[1]

$$\mathbf{N}^2\gamma^0 = \mathbf{N}^1\gamma^0 + \mathbf{y}^1\left[\left(\Delta\mathbf{x}^1\right)^{\mathrm{T}}\gamma^0 - \left(\gamma^1\right)^{\mathrm{T}}\mathbf{N}^1\gamma^0\right] \tag{3.51}$$

Replacing $\mathbf{N}^1\gamma^0$ with $\Delta\mathbf{x}^0$:

$$\mathbf{N}^2\gamma^0 = \Delta\mathbf{x}^0 + \mathbf{y}^1\left[\left(\Delta\mathbf{x}^1\right)^{\mathrm{T}}\gamma^0 - \left(\gamma^1\right)^{\mathrm{T}}\Delta\mathbf{x}^0\right] \tag{3.52}$$

Now we examine the term in brackets. We note that,

$$\left(\gamma^1\right)^{\mathrm{T}}\Delta\mathbf{x}^0 = \left(\mathbf{H}\Delta\mathbf{x}^1\right)^{\mathrm{T}}\Delta\mathbf{x}^0 = \left(\Delta\mathbf{x}^1\right)^{\mathrm{T}}\mathbf{H}\Delta\mathbf{x}^0 = \left(\Delta\mathbf{x}^1\right)^{\mathrm{T}}\gamma^0 \tag{3.53}$$

So the term in brackets in (3.52) vanishes, giving the desired result,

$$\mathbf{N}^2\gamma^0 = \Delta\mathbf{x}^0 \tag{3.54}$$

This result, although specific to this example, holds in general for the last *n* updates, as given in (3.44).

From the numerical example in that last section we had,

$$\mathbf{N}^2 = \begin{bmatrix} 0.667 & 0.166 \\ 0.166 & 0.167 \end{bmatrix} \quad \gamma^0 = \begin{bmatrix} 5.336 \\ -15.522 \end{bmatrix} \quad \Delta\mathbf{x}^0 = \begin{bmatrix} 0.970 \\ -1.698 \end{bmatrix}$$

Within round-off, (3.54) is satisfied for this data.

3.7.3 Conjugacy

3.7.3.1 *Definition*

Quasi-Newton methods are also methods of *conjugate directions.* A set of search directions, $\mathbf{s}^0, \mathbf{s}^1, \ldots, \mathbf{s}^k$ are said to be *conjugate* with respect to a square, symmetric matrix, **H**, if,

[1] Recall that when you take the transpose inside a product, the order of the product is reversed; also because **N** is symmetric, $\mathbf{N}^T = \mathbf{N}$. Thus: $\left(\mathbf{N}^1\gamma^1\right)^{\mathrm{T}}\gamma^0 = \left(\gamma^1\right)^{\mathrm{T}}\mathbf{N}^1\gamma^0$),

$$\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^i = 0 \text{ for all } i \neq k \tag{3.55}$$

A set of conjugate directions possesses an important property: If minimizing line searches are used along each conjugate direction, a method of conjugate directions is guaranteed to minimize a quadratic function of *n* variables in at most *n* steps. Himmelblau [5] indicates the excellent convergence properties of quasi-Newton methods on general functions may be due more to their conjugate direction properties than to their ability to approximate the Hessian inverse. Because of the importance of conjugate directions, we will prove two results here.

PROPOSITION. If **H** is positive definite and the set of non-zero vectors $\mathbf{s}^0, \mathbf{s}^1, ..., \mathbf{s}^{n-1}$ are conjugate to **H,** then these vectors are linearly independent.

PROOF. Suppose we have constants, α^i, $i = 0, 2, ..., n-1$ such that

$$\alpha^0\mathbf{s}^0 + \alpha^1\mathbf{s}^1 + ... + \alpha^k\mathbf{s}^k + ... + \alpha^{n-1}\mathbf{s}^{n-1} = \mathbf{0} \tag{3.56}$$

Now we multiply each term by $\left(\mathbf{s}^k\right)^T \mathbf{H}$:

$$\alpha^0 \underbrace{\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^0}_{=0} + \alpha^1 \underbrace{\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^1}_{=0} + ... + \alpha^k \underbrace{\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^k}_{positive} + ... + \alpha^{n-1} \underbrace{\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^{n-1}}_{=0} = \mathbf{0} \tag{3.57}$$

From conjugacy, all of the terms except $\alpha^k \left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^k$ are zero. Since **H** is positive definite, then the only way for this remaining term to be zero is for α^k to be zero. In this way we can show that for (3.57) to be satisfied all the α coefficients must be zero. This is the definition of linear independence.

3.7.3.2 Conjugate Direction Theorem

We will now show that a method of conjugate directions will solve a quadratic function in *n* steps, if minimizing steps are taken.

THEOREM. Let $\mathbf{s}^0, \mathbf{s}^1, ..., \mathbf{s}^{n-1}$ be a set of non-zero **H** conjugate vectors, with **H** a positive definite matrix. For the function,

$$f^{k+1} = f^k + \left(\nabla f^k\right)^{\mathrm{T}} \left(\mathbf{x}^{k+1} - \mathbf{x}^k\right) + \frac{1}{2}\left(\mathbf{x}^{k+1} - \mathbf{x}^k\right)^{\mathrm{T}} \mathbf{H}\left(\mathbf{x}^{k+1} - \mathbf{x}^k\right) \tag{3.58}$$

the sequence,

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k\mathbf{s}^k \tag{3.59}$$

with,

$$\alpha^k = -\frac{\left(\nabla f^k\right)^T \mathbf{s}^k}{\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^k} \tag{3.60}$$

converges to the unique solution, $\mathbf{H}(\mathbf{x}^* - \mathbf{x}^k) = -\nabla f^k$ after n steps, that is $\mathbf{x}^* = \mathbf{x}^n$.

Note: In the discussion which follows, recall that in general for the function (3.58),

$$\nabla f^{k+1} = \nabla f^k + \mathbf{H}\left(\mathbf{x}^{k+1} - \mathbf{x}^k\right)$$

and at the optimum,

$$0 = \nabla f^k + \mathbf{H}\left(\mathbf{x}^* - \mathbf{x}^k\right) \tag{3.61}$$

PROOF. Based on (3.59) above we note that,

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

Likewise for $\mathbf{x}^2$:

$$\mathbf{x}^2 = \mathbf{x}^1 + \alpha^1 \mathbf{s}^1 = \mathbf{x}^0 + \alpha^0 \mathbf{s}^0 + \alpha^1 \mathbf{s}^1$$

Or, in general

$$\left(\mathbf{x}^k - \mathbf{x}^0\right) = \alpha^0 \mathbf{s}^0 + \alpha^1 \mathbf{s}^1 + \ldots + \alpha^{k-1} \mathbf{s}^{k-1} \tag{3.62}$$

After n steps, we can write the optimum (assuming the directions are independent, which we just showed) as,

$$\left(\mathbf{x}^* - \mathbf{x}^0\right) = \alpha^0 \mathbf{s}^0 + \alpha^1 \mathbf{s}^1 + \ldots + \alpha^k \mathbf{s}^k + \ldots + \alpha^{n-1} \mathbf{s}^{n-1} \tag{3.63}$$

The rest of the proof is focused on determining the α coefficients in (3.63). Because the **s** vectors are conjugate directions, when we multiply both sides of (3.63) by $\left(\mathbf{s}^k\right)^T \mathbf{H}$, we have,

$$\left(\mathbf{s}^k\right)^T \mathbf{H}\left(\mathbf{x}^* - \mathbf{x}^0\right) = \alpha^0 \underbrace{\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^0}_{=0} + \alpha^1 \underbrace{\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^1}_{=0} + \ldots + \alpha^k \underbrace{\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^k}_{positive} + \ldots + \alpha^{n-1} \underbrace{\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^{n-1}}_{=0}$$

which allows us to solve for α^k:

$$\alpha^k = \frac{\left(\mathbf{s}^k\right)^T \mathbf{H}(\mathbf{x}^* - \mathbf{x}^0)}{\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^k} \tag{3.64}$$

Unfortunately (3.64) is in terms of $\mathbf{x}^*$, which we presumably don't know. However, if we multiply (3.62) by $\left(\mathbf{s}^k\right)^T \mathbf{H}$, we have,

$$\left(\mathbf{s}^k\right)^T \mathbf{H}\left(\mathbf{x}^k - \mathbf{x}^0\right) = \alpha^0 \underbrace{\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^0}_{=0} + \alpha^1 \underbrace{\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^1}_{=0} + \ldots + \alpha^{k-1} \underbrace{\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^{k-1}}_{=0} \tag{3.65}$$

which gives,

$$\left(\mathbf{s}^k\right)^T \mathbf{H}(\mathbf{x}^k - \mathbf{x}^0) = 0 \tag{3.66}$$

Substituting this result into (3.64), we have

$$\alpha^k = \frac{\left(\mathbf{s}^k\right)^T \mathbf{H}(\mathbf{x}^* - \mathbf{x}^k)}{\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^k} \tag{3.67}$$

So far we have not enforced any condition that associates $\mathbf{x}^*$ with an optimum. We do so now. Recalling that $\mathbf{H}(\mathbf{x}^* - \mathbf{x}^k) = -\nabla f^k$ at the optimum, we can write α^k as,

$$\alpha^k = -\frac{\left(\nabla f^k\right)^T \mathbf{s}^k}{\left(\mathbf{s}^k\right)^T \mathbf{H}\mathbf{s}^k}$$

which is identical with (3.60).

We notice that (3.60) is the same as the minimizing step we derived in the Appendix. Thus the conjugate direction theorem relies on taking minimizing steps.

3.7.3.3 Quasi-Newton Methods and Conjugacy

We stated earlier that quasi-Newton methods are also methods of conjugate directions. Thus for the example given in Section 7.3, we should have,

$$\left(\mathbf{s}^0\right)^T \mathbf{H}\mathbf{s}^1 = 0$$

Substituting the search directions and Hessian of that problem,

$$\begin{bmatrix} 0.496 & -0.868 \end{bmatrix} \begin{bmatrix} 2. & -2. \\ -2. & 8. \end{bmatrix} \begin{bmatrix} 2.837 \\ 0.975 \end{bmatrix} = 0.0017$$

Within the round-off of the data, we see this is verified.

It can be shown that any quasi-Newton method that has the hereditary property is also a method of conjugate directions. This is shown in the Appendix.

3.7.3.4 Some Insight into Conjugacy

As discussed by Nocedal and Wright [13] and Fletcher [6], there is an interesting graphical interpretation of the properties of conjugate directions. If the Hessian in (3.58) is a diagonal matrix, the contours of the function are aligned with the coordinate directions. We can optimize the function by conducting minimizing line searches along the coordinate directions, each one in turn, as shown in Fig. 3.11.

However, if the Hessian is not a diagonal matrix, the strategy of minimizing along the coordinate directions no longer works. In order to use such a method, we must first transform the Hessian into a diagonal matrix and then minimize along new, transformed coordinate directions.

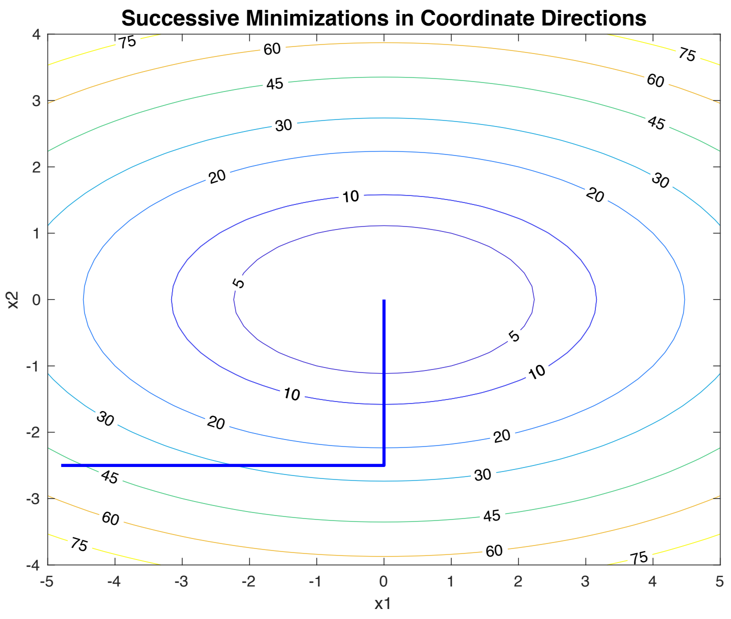

Fig. 3.11. For a Hessian comprised of a diagonal matrix, the function contours are aligned with the coordinate axes. The function can be optimized by minimizing along each coordinate axis in turn.

Suppose we have the quadratic function,

$$\phi(\mathbf{x}) = \mathbf{b}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} \tag{3.68}$$

where **A** is a symmetric, positive definite, but not diagonal matrix. If we define **S** as an *n* x *n* matrix given by,

$$\mathbf{S} = [\mathbf{s}^0, \mathbf{s}^1, \ldots, \mathbf{s}^{n-1}]$$

where the vectors **s** are conjugate with respect to **A**, and we define a new set of variables,

$$\hat{\mathbf{x}} = \mathbf{S}^{-1}\mathbf{x} \tag{3.69}$$

The quadratic function (3.68) becomes,

$$\phi(\mathbf{S}\hat{\mathbf{x}}) = (\mathbf{S}^T\mathbf{b})^T\hat{\mathbf{x}} + \frac{1}{2}\hat{\mathbf{x}}^T(\mathbf{S}^T\mathbf{A}\mathbf{S})^T\hat{\mathbf{x}}$$

From the definition of conjugacy (3.55) all of the cross-product terms of the matrix $\mathbf{S}^T\mathbf{A}\mathbf{S}$ disappear, leaving us with a diagonal matrix, so we can find the optimum of ϕ by conducting a minimizing step along the coordinate directions given by $\hat{\mathbf{x}}$. By (3.69), the *i*th coordinate direction in $\hat{\mathbf{x}}$-space corresponds to the $\mathbf{s}_i$ direction in **x** space.

Another result of conjugacy is that at the *k+1* step,

$$\left(\nabla f^{k+1}\right)^T \mathbf{s}^i = 0 \quad \text{for all} \quad i \le k \tag{3.70}$$

Equation (3.70) indicates 1) that the current gradient is orthogonal to all the past search directions, and 2) at the current point we have zero slope with respect to all past search directions, i.e.,

$$\frac{\partial f}{\partial \alpha^i} = 0 \quad \text{for all} \quad i \le k$$

meaning we have minimized the function in the "subspace" of the previous directions.

3.7.4 Rank 2 Updates

3.7.4.1 The DFP Method

Although the rank one update does have the hereditary property (and is a method of conjugate directions), it does not guarantee that at each stage the direction matrix, **N**, is positive definite. It is important that the update remain positive definite because this insures the search direction will always go downhill. It has been shown that (3.43) is the only rank one update which satisfies the quasi-Newton condition. For more flexibility, rank 2 updates have been proposed. These are of the form,

$$\mathbf{N}^{k+1} = \mathbf{N}^k + a\mathbf{u}\mathbf{u}^{\mathrm{T}} + b\mathbf{v}\mathbf{v}^{\mathrm{T}} \tag{3.71}$$

If we substitute this into the quasi-Newton condition,

$$\mathbf{N}^{k+1}\boldsymbol{\gamma}^k = \Delta\mathbf{x}^k \tag{3.72}$$

we have,

$$\mathbf{N}^k\boldsymbol{\gamma}^k + a\mathbf{u}\mathbf{u}^{\mathrm{T}}\boldsymbol{\gamma}^k + b\mathbf{v}\mathbf{v}^{\mathrm{T}}\boldsymbol{\gamma}^k = \Delta\mathbf{x}^k \tag{3.73}$$

There are a number of possible choices for **u** and **v**. One choice is to try,

$$\mathbf{u} = \Delta\mathbf{x}^k, \quad \mathbf{v} = \mathbf{N}^k\boldsymbol{\gamma}^k \tag{3.74}$$

Substituting (3.74) into (3.73),

$$\mathbf{N}^k \gamma^k + a \Delta \mathbf{x}^k \underbrace{\left(\Delta \mathbf{x}^k\right)^{\mathrm{T}} \gamma^k}_{scalar} + b \mathbf{N}^k \gamma^k \underbrace{\left(\mathbf{N}^k \gamma^k\right)^{\mathrm{T}} \gamma^k}_{scalar} = \Delta \mathbf{x}^k \tag{3.75}$$

In (3.75) we note that the dot products result in scalars. If we choose *a* and *b* such that,

$$a\left(\Delta \mathbf{x}^k\right)^{\mathrm{T}} \boldsymbol{\gamma}^k = 1 \text{ and } b\left(\mathbf{N}^k \boldsymbol{\gamma}^k\right)^{\mathrm{T}} \boldsymbol{\gamma}^k = -1 \tag{3.76}$$

Equation (3.73) becomes,

$$\mathbf{N}^k \boldsymbol{\gamma}^k + \Delta \mathbf{x}^k - \mathbf{N}^k \boldsymbol{\gamma}^k = \Delta \mathbf{x}^k \tag{3.77}$$

and is satisfied.

Combining (3.76), (3.74) and (3.71), the update is,

$$\mathbf{N}^{k+1} = \mathbf{N}^k + \frac{\Delta \mathbf{x}^k \left(\Delta \mathbf{x}^k\right)^{\mathrm{T}}}{\left(\Delta \mathbf{x}^k\right)^{\mathrm{T}} \boldsymbol{\gamma}^k} - \frac{\mathbf{N}^k \boldsymbol{\gamma}^k \left(\mathbf{N}^k \boldsymbol{\gamma}^k\right)^{\mathrm{T}}}{\left(\mathbf{N}^k \boldsymbol{\gamma}^k\right)^{\mathrm{T}} \boldsymbol{\gamma}^k} \tag{3.78}$$

Or, with some rearranging, as it is more commonly given,

$$\mathbf{N}^{k+1} = \mathbf{N}^k + \frac{\Delta \mathbf{x}^k \left(\Delta \mathbf{x}^k\right)^{\mathrm{T}}}{\left(\Delta \mathbf{x}^k\right)^{\mathrm{T}} \boldsymbol{\gamma}^k} - \frac{\mathbf{N}^k \boldsymbol{\gamma}^k \left(\boldsymbol{\gamma}^k\right)^{\mathrm{T}} \mathbf{N}^k}{\left(\boldsymbol{\gamma}^k\right)^{\mathrm{T}} \mathbf{N}^k \boldsymbol{\gamma}^k} \tag{3.79}$$

Davidon [7] was the first one to propose this update. Fletcher and Powell [8] further developed his method; thus this method came to be known as the Davidon-Fletcher-Powell (DFP) update. This update has the following properties,

For quadratic functions:

1. It has the hereditary property; after *n* updates, $\mathbf{N}^n = \mathbf{H}^{-1}$.
2. It is a method of conjugate directions and therefore terminates after at most n steps.

For general functions (including quadratics):

3. The direction matrix **N** remains positive definite if we do exact line searches. This guarantees the search direction points downhill at every step.

The DFP update was popular for many years. As mentioned, we need to take a minimizing step to insure **N** stays positive definite. The DFP method is more sensitive to errors in α^* than the BFGS update, described in the next section, and can degrade if α^* is not accurate.

3.7.5 The Broyden Fletcher Goldfarb Shanno (BFGS) Update

The most widely used update today is known as the Broyden [9], Fletcher [10], Goldfarb [11], Shanno [12] or "BFGS" update, suggested by all four authors independently in 1970. It is also a rank 2 update. It has the same properties as the DFP update but does not require a minimizing step to remain positive definite, but only requires $\Delta\mathbf{x}^T\gamma > 0$. (If for a particular step this condition is not met, the update is skipped.) It also works better for inaccurate line searches. This update is,

$$\mathbf{N}^{k+1} = \mathbf{N}^k + \left(1 + \frac{\left(\boldsymbol{\gamma}^k\right)^{\mathrm{T}} \mathbf{N}^k \boldsymbol{\gamma}^k}{\left(\Delta\mathbf{x}^k\right)^{\mathrm{T}} \boldsymbol{\gamma}^k}\right)\left(\frac{\Delta\mathbf{x}^k \left(\Delta\mathbf{x}^k\right)^{\mathrm{T}}}{\left(\Delta\mathbf{x}^k\right)^{\mathrm{T}} \boldsymbol{\gamma}^k}\right) - \frac{\Delta\mathbf{x}^k \left(\boldsymbol{\gamma}^k\right)^{\mathrm{T}} \mathbf{N}^k + \mathbf{N}^k \boldsymbol{\gamma}^k \left(\Delta\mathbf{x}^k\right)^{\mathrm{T}}}{\left(\Delta\mathbf{x}^k\right)^{\mathrm{T}} \boldsymbol{\gamma}^k} \tag{3.80}$$

This update is currently considered to be the best update for use in optimization. It is the update inside MATLAB, Excel and many other optimization packages.

3.7.6 Comments About Quasi-Newton Methods

The quasi-Newton methods explained here combine the advantages of steepest descent and Newton's method without the disadvantages. They start out as steepest descent, which works well far from the optimum, and gradually become Newton's method, which works well near the optimum. They do this without requiring the evaluation of second derivatives, which can be computationally expensive. By insuring the update is positive definite, the search direction will always go downhill.

Note that these methods use information the previous methods "threw away." Quasi-Newton methods use differences in gradients and differences in **x** to estimate second derivatives according to (3.35). This allows information from previous steps to correct (or update) the current step.

3.7.7 Hessian Updates Vs. Hessian Inverse Updates

All of the updates we have presented so far are updates for the Hessian Inverse. We can easily develop updates for the Hessian itself, as will be required for the SQP and IP algorithms, starting from the condition

$$\boldsymbol{\gamma}^k = \mathbf{H}^k \Delta\mathbf{x}^k \tag{3.81}$$

instead of $\left(\mathbf{H}^{-1}\right)^k \boldsymbol{\gamma}^k = \Delta\mathbf{x}^k$ which we used before. The BFGS Hessian approximation ((3.80) is the Hessian inverse approximation) is given by,

$$\mathbf{H}^{k+1} = \mathbf{H}^k + \frac{\boldsymbol{\gamma}^k \left(\boldsymbol{\gamma}^k\right)^{\mathrm{T}}}{\left(\boldsymbol{\gamma}^k\right)^{\mathrm{T}} \Delta\mathbf{x}^k} - \frac{\mathbf{H}^k \Delta\mathbf{x}^k \left(\Delta\mathbf{x}^k\right)^{\mathrm{T}} \mathbf{H}^k}{\left(\Delta\mathbf{x}^k\right)^{\mathrm{T}} \mathbf{H}^k \Delta\mathbf{x}^k} \tag{3.82}$$

You will note that this looks a lot like the DFP Hessian inverse update but with **H** interchanged with **N** and **γ** interchanged with Δ**x**. In fact these two formulas are said to be complementary to each other.

3.8 The Conjugate Gradient Method

3.8.1 Definition

There is one more method we will learn, called the *Conjugate Gradient* method. We will present the results for this method primarily because it is almost as simple as steepest descent but is a method of conjugate directions, making it much more powerful. Because it uses very little storage, it is often used in large-scale problems.

The search direction for conjugate gradient is given by,

$$\mathbf{s}^{k+1} = -\nabla f^{k+1} + \beta^{\mathrm{k}}\mathbf{s}^{\mathrm{k}} \tag{3.83}$$

Where β^{k}, a scalar, is given by

$$\beta^k = \frac{\left(\nabla f^{k+1}\right)^{\mathrm{T}} \nabla f^{k+1}}{\left(\nabla f^{k}\right)^{\mathrm{T}} \nabla f^{k}} \tag{3.84}$$

We will motivate the theory behind this formula and solve for β by developing it for the second step (given that the first step is steepest descent) i.e.,

$$\mathbf{s}^0 = -\nabla f(\mathbf{x}^0)$$

$$\mathbf{s}^1 = -\nabla f(\mathbf{x}^1) + \beta^0\mathbf{s}^0 \tag{3.85}$$

For a method of conjugate directions,

$$(\mathbf{s}^0)^T\mathbf{H}\mathbf{s}^1 = 0 \tag{3.86}$$

We know for a quadratic function,

$$\nabla f^1 - \nabla f^0 = \mathbf{H}(\mathbf{x}^1 - \mathbf{x}^0) = \mathbf{H}\alpha^0\mathbf{s}^0$$

taking the transpose (and recognizing that $\mathbf{H}^T = \mathbf{H}$),

$$(\nabla f^1 - \nabla f^0)^T = (\mathbf{x}^1 - \mathbf{x}^0)^T\mathbf{H} = \alpha^0(\mathbf{s}^0)^T\mathbf{H}$$

post multiplying by $\mathbf{H}^{-1}$,

$$(\nabla f^1 - \nabla f^0)^T \mathbf{H}^{-1} = (\mathbf{x}^1 - \mathbf{x}^0)^T = \alpha^0 (\mathbf{s}^0)^T$$

from which we can write,

$$(\mathbf{s}^0)^T = \frac{(\nabla f^1 - \nabla f^0)^T \mathbf{H}^{-1}}{\alpha^0} \tag{3.87}$$

Substituting (3.87) and (3.85) into (3.86) gives,

$$\left[\frac{(\nabla f^1 - \nabla f^0)^T \mathbf{H}^{-1}}{\alpha^0}\right]\mathbf{H}(-\nabla f^1 + \beta^0 \mathbf{s}^0) = 0 \tag{3.88}$$

We can multiply both sides by α^0 so it drops out, and the Hessian inverse and Hessian combine to give the identity matrix. Thus we are left with,

$$(\nabla f^1 - \nabla f^0)^T (-\nabla f^1 - \beta \nabla f^0) = 0$$

If we take a minimizing step, α^*, then the cross products of (3.88) disappear and we have,

$$\beta = \frac{\left(\nabla f^1\right)^T \nabla f^1}{\left(\nabla f^0\right)^T \nabla f^0} \tag{3.89}$$

3.8.2 Example: Conjugate Gradient Method

We will optimize our usual function, $f = x_1^2 - 2x_1x_2 + 4x_2^2$

starting from $\mathbf{x}^0 = \begin{bmatrix} -3 \\ 1 \end{bmatrix}$ $\nabla f^0 = \begin{bmatrix} -8 \\ 14 \end{bmatrix}$

We take a minimizing step in the negative gradient direction and stop at

$$\mathbf{x}^1 = \begin{bmatrix} -2.03 \\ -0.7 \end{bmatrix} \quad \nabla f^1 = \begin{bmatrix} -2.664 \\ -1.522 \end{bmatrix}$$

Now we calculate β^0 as

$$\beta^0 = \frac{\left(\nabla f^1\right)^{\mathrm{T}} \nabla f^1}{\left(\nabla f^0\right)^{\mathrm{T}} \nabla f^0} = \frac{\begin{bmatrix} -2.664 & -1.522 \end{bmatrix}\begin{bmatrix} -2.664 \\ -1.522 \end{bmatrix}}{\begin{bmatrix} -8 & 14 \end{bmatrix}\begin{bmatrix} -8 \\ 14 \end{bmatrix}} = \frac{9.413}{260} = 0.0362$$

We calculate the new search direction as,

$$\mathbf{s}^1 = -\nabla f^1 + \beta \mathbf{s}^0 = -\begin{bmatrix} -2.664 \\ -1.522 \end{bmatrix} + 0.0362 \begin{bmatrix} 8 \\ -14 \end{bmatrix} = \begin{bmatrix} 2.954 \\ 1.015 \end{bmatrix}$$

when we step in this direction, we arrive at the optimum, $\mathbf{x}^2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \nabla f^2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

The main advantage of the conjugate gradient method, as compared to quasi-Newton methods, is computation and storage. The conjugate gradient method only requires that we store the last search direction and last gradient, instead of a full matrix.

Although both conjugate gradient and quasi-Newton methods will optimize quadratic functions in *n* steps, on non-quadratic problems quasi-Newton methods are better. Further, small errors can build up in the conjugate gradient method so some researchers recommend restarting the algorithm periodically (such as every *n* steps) to be steepest descent.

3.9 Inexact Line Searches

Up until now we have used a line search with a quadratic fit to try to locate the minimum of the objective function in the search direction. If desired, we could refit the quadratic estimate several times to try to locate the exact minimum.

However, if we are far away from the optimum, so that we have lots of iterations ahead of us, then finding the minimum precisely is not very efficient. We incur significant expense to find the minimum for the current step only to move away from it. It would be useful to have methods that allow us to be "sloppy" in our line search until we are close to the optimum.

3.9.1 Sufficient Decrease

Several criteria define what it means to have a *sufficient decrease* in the objective. If this condition is met, we terminate the line search and start a new iteration. All of these criteria require the following decrease in the objective function,

$$f(\mathbf{x}^{k+1}) \le f(\mathbf{x}^k) + c_1 \alpha^k \underbrace{\left(\nabla f^k\right)^T \mathbf{s}^k}_{direct.\,deriv.} \tag{3.90}$$

where c_1 is a constant (between 0 and ½; Nocedal and Wright [13] indicate a typical value is quite small: 1 e-4) and the term in brackets is the directional derivative. This condition states that a new point, to be acceptable, must be less than the starting value of the objective plus a fraction of the step length times the directional derivative (note that the directional derivative is negative). This expression is also known as the *Armijo condition.*

This condition is necessary but not sufficient. In particular it prohibits large steps but not very small ones. This can be seen in Fig. 3.12. Expression (3.90) would allow all steps between 0 and α_a.

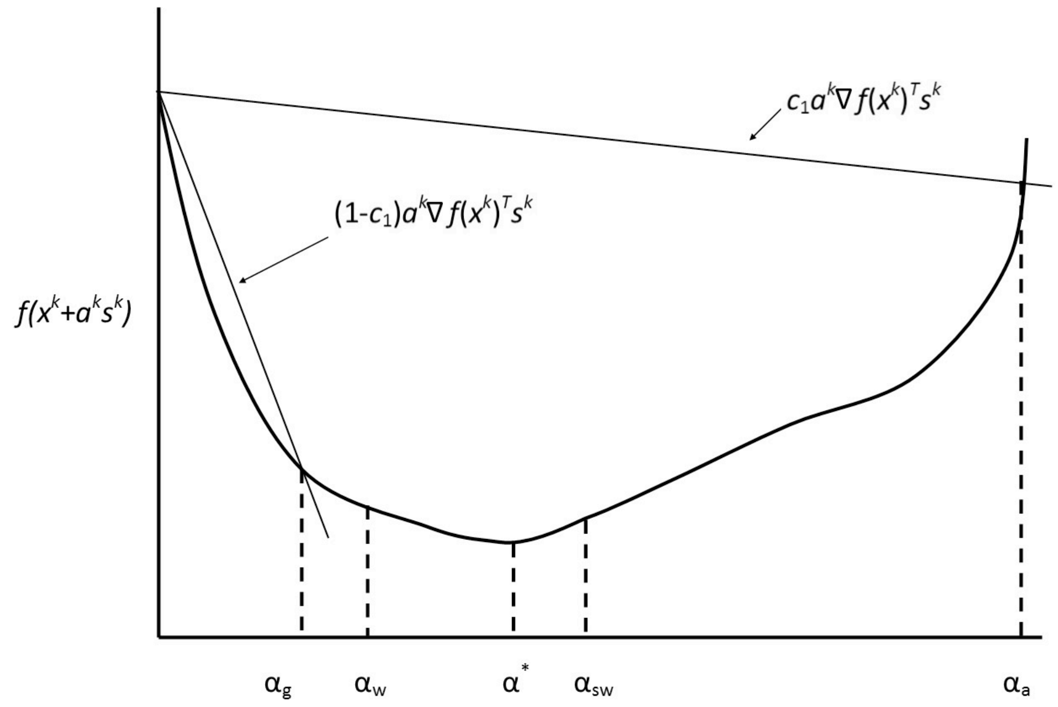

Fig. 3.12. Function values for various step lengths for an example function. After Biegler [14].

To preclude very small steps, we have three possible criteria:

$$\nabla f(\mathbf{x}^{k+1})^T \mathbf{s}^k \geq c_2 \nabla f(\mathbf{x}^k)^T \mathbf{s}^k \tag{3.91}$$

where $c_2 \in (c_1, 1)$. This statement requires the slope at the proposed point to be greater than some fraction of the slope at the beginning point. Since the slope at the starting point is negative (we are going downhill), this requires the slope to be more shallow or flatter than at the starting point, as it might be as we get close to the minimum in the search direction. Equations (3.90) and (3.91) are known as the *Wolfe conditions.* These conditions limit acceptable step lengths to be between α_w and α_a in the figure.

The *strong Wolfe conditions* are even more restrictive,

$$\left|\nabla f(\mathbf{x}^{k+1})^T \mathbf{s}\right| \leq c_2 \left|\nabla f(\mathbf{x}^k)^T \mathbf{s}^k\right| \tag{3.92}$$

Here we are taking the absolute value of the slopes and restricting an acceptable step to have a negative or positive slope less than the absolute value of the slope at the starting point. This restricts the step to be in a bowl. This would require the step to be between α_w and α_{sw} (note that the absolute value of the slope at α_w should equal the absolute value of the slope at α_{sw}.)

Finally we have the *Goldstein conditions*. These require (3.90) to be met and,

$$f(\mathbf{x}^k)+(1-c_1)\alpha^k\nabla f(\mathbf{x}^k)^T\mathbf{s}^k \le f(\mathbf{x}^{k+1}) \tag{3.93}$$

The condition given by (3.93) keeps the proposed step from being too small by setting a minimum acceptable step length, as shown in the figure. When combined with (3.90) we can write the Goldstein conditions as,

$$f(\mathbf{x}^k)+(1-c_1)\alpha^k\nabla f(\mathbf{x}^k)^T\mathbf{s}^k \le f(\mathbf{x}^{k+1}) \le f(\mathbf{x}^k)+c_1\alpha^k\nabla f(\mathbf{x}^k)^T\mathbf{s}^k$$

In terms of Fig. 3.11, these conditions require the step to be between α_g and α_a. The Goldstein conditions are easier to check than the Wolfe conditions because they do not involve evaluating the directional derivative at the proposed point.

3.9.2 Global Convergence of Line Search Methods

For a method where $\mathbf{s}=-(\mathbf{B}^k)^{-1}\nabla f^k$ with **B** a positive definite matrix, and α^k that satisfies the Wolfe or Goldstein conditions, it can be shown that a sequence of iterations will converge to a point where the norm of the gradient goes to zero. This is based on a theorem by Zoutendick [15].

3.9.3 A Backtracking Strategy

If we are using a quasi-Newton or Newton method, we often will begin with $\alpha=1$. If the Wolfe condition (3.90) is not met we then cut back the step size until it is. We don't always have to check the second condition because we are starting with a long step. Alternately, we can check the Goldstein conditions.

3.10 Trust Region Methods

3.10.1 Introduction

In the unconstrained methods we have studied so far, we used a line search to determine how far to go in some particular direction. The direction was independent of the step length. With *trust region* methods, the search direction can change as a function of the step length. Nocedal and Wright [13] explain, "Trust region methods define a region around the current point within which they *trust* the model to be an adequate representation of the objective function, and then choose the step to be the approximate minimizer of the model in this region. In effect, they choose the direction and length of the step simultaneously. If a step is not acceptable, they reduce the size of the region and find a new minimizer. In general, the direction of the step changes whenever the size of the trust region is altered."

This additional freedom gives trust region methods superior convergence properties over line search methods. However, the computational expense for the trust region calculations may exceed that for line search iterations.

Choosing the size of the trust region is an important part of the method. If the region is too small, progress towards the optimum is delayed. If the region is too large, the approximate model may not be an adequate representation of the actual function, and the step will be inaccurate, also delaying progress.

The trust region problem can be stated as,

$$\begin{aligned}&\text{Find } \Delta\mathbf{x} \text{ to,}\\ &\text{Min } f_a(\mathbf{x}^k+\Delta\mathbf{x}) = f(\mathbf{x}^k)+\nabla f(\mathbf{x}^k)^T\Delta\mathbf{x}+\frac{1}{2}\Delta\mathbf{x}^T\mathbf{B}^k\Delta\mathbf{x} \qquad (3.94)\\ &\text{s.t. } \left\|\Delta\mathbf{x}\right\| \le \Delta_{TR}\end{aligned}$$

As before, $\Delta\mathbf{x}$ is a vector with a magnitude and direction. The matrix **B** is either $\nabla^2 f(\mathbf{x}^k)$ or a quasi-Newton approximation. Thus we can state the problem as, "Minimize a quadratic approximation of the objective subject to a constraint that the magnitude (L_2 norm) of the step is less than the trust region, Δ_{TR}. Depending on how good the approximation is, we will expand or contract the trust region iteration by iteration.

Fig. 3.13 shows how the search direction and step length both change as we change the size of the trust region. The circles with dashed lines show the trust region radius. The arrows show the optimal step for a particular radius. The solid contours are the contours of the quadratic approximation we are trying to minimize.

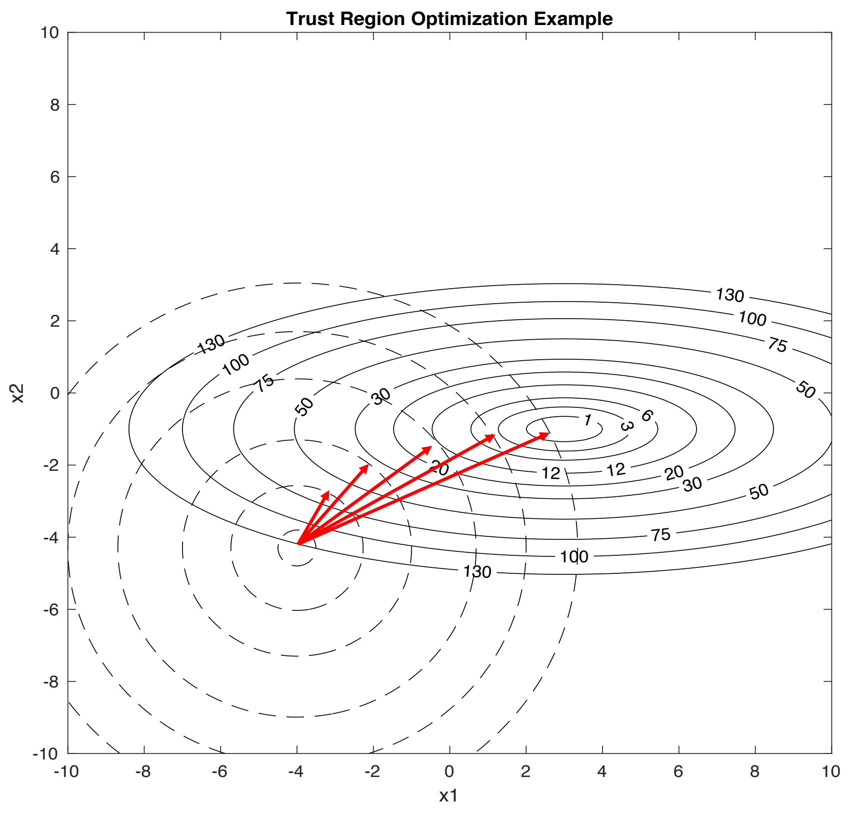

Fig. 3.13. Illustration of how step length and direction change with trust region size.

3.10.2 Solution

The trust region problem is an optimization problem which must be solved once each iteration. Based on the theory for the exact solution to (3.94) (see Biegler [14]), it can be shown that if the trust region is larger than the distance to the minimum of the approximation, the search direction is given by the Newton step, $\mathbf{s} = -\mathbf{B}^{-1}\nabla f$. When the trust region is small relative to the Newton step, the search direction becomes steepest descent, $\mathbf{s} = -\nabla f(\mathbf{x}^k)$. For cases in between, but still assuming the trust region is less than the distance to the minimum (i.e. $\left\|\Delta\mathbf{x}^*\right\| = \Delta_{TR}$), the optimal search direction is somewhere between these two cases. Based on Fig. 3.13, Fig. 3.14 shows how the optimal steps trace out an arc between steepest descent and the Newton step.

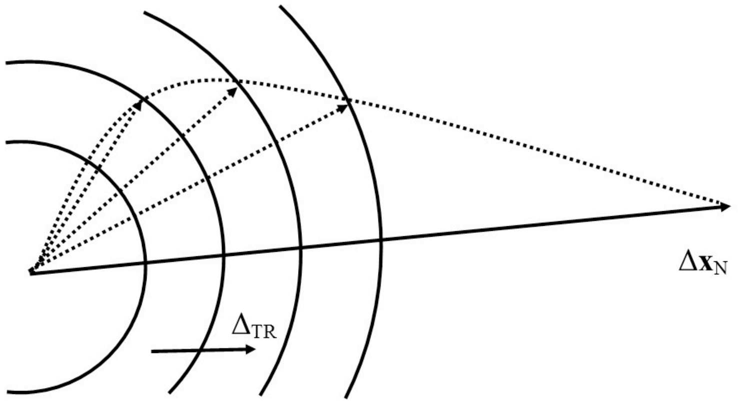

Fig. 3.14. Arc of optimal step and search direction for various trust region sizes. When the trust region approaches zero, the search direction is the negative gradient.

Solving the exact problem requires an iterative process. To reduce computation, approximate methods are often used. One such method is referred to as the *dogleg method*. The dogleg method approximates the arc of Fig. 3.12 with two limiting steps: the "Cauchy step," explained below, and the Newton step, $\Delta\mathbf{x}_N = -\mathbf{B}^{-1}\nabla f$. In between, we use a linear combination of these two steps where they are blended together such that the step length is equal to the trust region, Δ_{TR}. This dogleg method is shown in Fig. 3.15.

The Cauchy step is given by,

$$\Delta\mathbf{x}_C = -\left[\frac{\nabla f(\mathbf{x}^k)^T \nabla f(\mathbf{x}^k)}{\nabla f(\mathbf{x}^k)^T \mathbf{B}^k \nabla f(\mathbf{x}^k)}\right]\nabla f(\mathbf{x}^k) \qquad \text{if } \nabla f^T \mathbf{B} \nabla f > 0 \tag{3.95}$$

or,

$$\Delta\mathbf{x}_C = -\frac{\Delta_{TR}^k}{\left\|\nabla f(\mathbf{x}^k)\right\|}\nabla f(\mathbf{x}^k) \qquad \text{otherwise} \tag{3.96}$$

The term in brackets in (3.95) is a scalar. If we consider the search direction to be $\mathbf{s} = -\nabla f^k$, then we see that this scalar can be written as, $\alpha^* = -\dfrac{\left(\nabla f^k\right)^T \mathbf{s}}{\mathbf{s}^T \mathbf{B} \mathbf{s}}$, i.e. it is equal to the minimizing step length in this direction (see (3.13)). If $\nabla f^T \mathbf{B} \nabla f$ is not positive, then the Cauchy step is the normalized steepest descent search direction with a step length equal to the trust region, as given by (3.96).

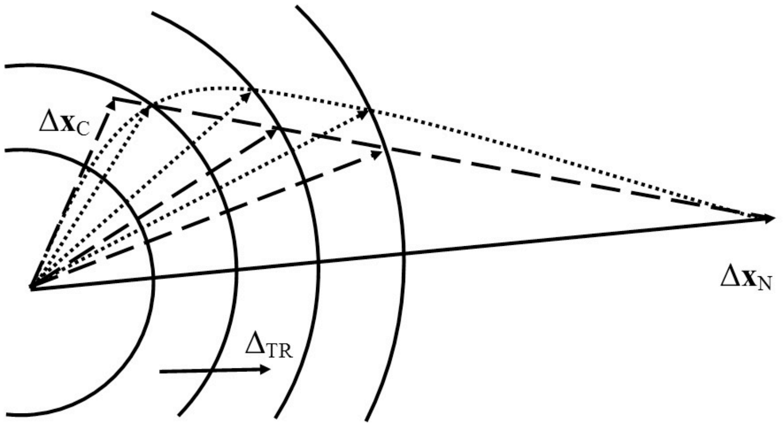

Fig. 3.15. The steps of the dogleg and exact methods for various trust region sizes. The dogleg steps are a linear combination of the Cauchy step and the Newton step.

The trust region algorithm using the dogleg method can be summarized as follows,

1. Solve the trust region problem (3.94) using the follow equations,

$$\Delta\mathbf{x}^k = \Delta\mathbf{x}_N \text{ if } \Delta_{TR} \ge \left\|\Delta\mathbf{x}_N\right\|$$

$$\Delta\mathbf{x}^k = \frac{\Delta_{TR}^k}{\left\|\Delta\mathbf{x}_C\right\|}\Delta\mathbf{x}_C \text{ if } \Delta_{TR} \le \left\|\Delta\mathbf{x}_C\right\|$$

Otherwise,

$$\Delta\mathbf{x}^k = \eta\Delta\mathbf{x}_N + (1-\eta)\Delta\mathbf{x}_C$$

where $\Delta\mathbf{x}_N$ is the Newton step, $\Delta\mathbf{x}_C$ is the Cauchy step, and

$$\eta = \frac{-(\Delta\mathbf{x}_N - \Delta\mathbf{x}_C)^T\Delta\mathbf{x}_C + \left[((\Delta\mathbf{x}_N - \Delta\mathbf{x}_C)^T\Delta\mathbf{x}_C)^2 + (\Delta_{TR}^2 - \left\|\Delta\mathbf{x}_C\right\|^2)\left\|\Delta\mathbf{x}_N - \Delta\mathbf{x}_C\right\|^2\right]^{1/2}}{\left\|\Delta\mathbf{x}_N - \Delta\mathbf{x}_C\right\|^2}$$

This value of η results in a step length of magnitude $\left\|\Delta\mathbf{x}^k\right\|$ equal to Δ_{TR}^k

2. Compute the ratio, ρ, of the actual change in the function to the predicted change in the function. This ratio is given by,

$$\rho^k = \frac{\left(f(\mathbf{x}^k) - f(\mathbf{x}^k + \Delta\mathbf{x}^k)\right)}{\left(f_a(\mathbf{x}^k) - f_a(\mathbf{x}^k + \Delta\mathbf{x}^k)\right)}$$

Note that it is possible for this ratio to be negative (actual function got worse) or greater than one (actual improvement was greater than predicted).

3. If $\rho^k < \frac{1}{4}$, then $\Delta_{TR}^{k+1} < \frac{1}{4}\Delta_{TR}^k$

   else if $\rho^k > \frac{3}{4}$ and $\left\|\Delta\mathbf{x}^k\right\| = \Delta_{TR}^k$, then $\Delta_{TR}^{k+1} = Min(2\Delta_{TR}^k, \Delta_{\max})$

   else $\Delta_{TR}^{k+1} = \Delta_{TR}^k$

4. If $\rho^k > \beta$, then $\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k$
   else $\mathbf{x}^{k+1} = \mathbf{x}^k$

5. Continue until convergence is reached ($\left\|\nabla f\right\| < \varepsilon$).

In the above, the constant β has a value between 0 and ¼ . The values of ¼ and ¾ used above are typical values. The parameter $\Delta_{\max}$ is the maximum limit on the trust region.

3.10.3 <u>Example</u>

We will apply the trust region method to the non-quadratic problem,

$$\text{Min} \quad f(\mathbf{x}) = x_1^4 - 2x_2x_1^2 + x_2^2 + x_1^2 - 2x_1 + 5$$

starting at the point $\mathbf{x}^T = [-1, 4]$. A contour plot of the problem is shown in Fig. 3.16.

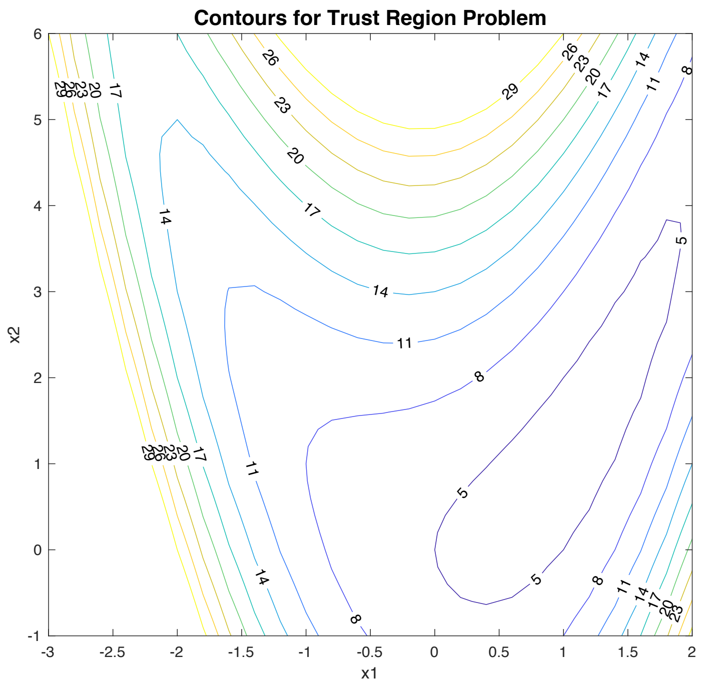

Fig. 3.16 Contour plot for example problem.

We begin with a Hessian set to the identity matrix and use the BFGS update. The table below shows the progress of the algorithm for the first 12 steps. In the table, we compare the approximate solution using the dogleg method to the exact solution of the trust region problem. The exact solution was obtained using `fmincon` in MATLAB. The beginning trust region was somewhat arbitrarily set to be the norm of the Newton step divided by 8. This turned out to be 1.25. From step 5 on, the Newton step was smaller than the trust region.

Trust Region Example				Approximate		Exact				
Iter	x_1	x_2	f	Δx_1	Δx_2	Δx_1	Δx_2	ρ	$\lVert\Delta \mathbf{x}\rVert$	*Comments*
1	-1.000	4.000	17	-1.00	-0.750	-1.00	-0.750	0.29	1.25	= trust region
2	-2.000	3.250	13.56	0.993	-0.759	0.941	-0.823	0.55	1.25	= trust region
3	-1.006	2.491	10.21	0.297	-1.214	0.333	-1.205	1.29	1.25	= trust region
4	-0.709	1.277	7.52	0.784	-1.840	0.667	-1.886	0.79	2.00	= max trust reg
5	0.075	-0.563	5.17	-0.022	0.292	-0.022	0.291	1.40	0.29	= Newton step
6	0.052	-0.271	4.97	0.293	0.279	0.293	0.279	1.53	0.41	= Newton step
7	0.345	0.008	4.44	0.451	0.286	0.450	0.286	0.97	0.53	= Newton step
8	0.796	0.295	4.15	0.026	0.342	0.026	0.342	1.14	0.34	= Newton step
9	0.822	0.637	4.03	0.131	0.250	0.131	0.249	1.23	0.28	= Newton step
10	0.953	0.888	4.002	0.041	0.095	0.041	0.094	1.12	0.10	= Newton step
11	0.995	0.983	4.000	0.005	0.016	0.004	0.016	0.99	0.01	= Newton step
12	0.999	0.999	4.000	0.001	0.001	0.000	0.000	0.86	0.001	= Newton step

The steps of the table are shown graphically in Fig. 3.17. Note that from step 5 onwards, where the step to the optimum of the model was smaller than the radius of the trust region, we have just shown the optimum (Newton) step. Fig. 3.18 shows the path of `fmincon`.

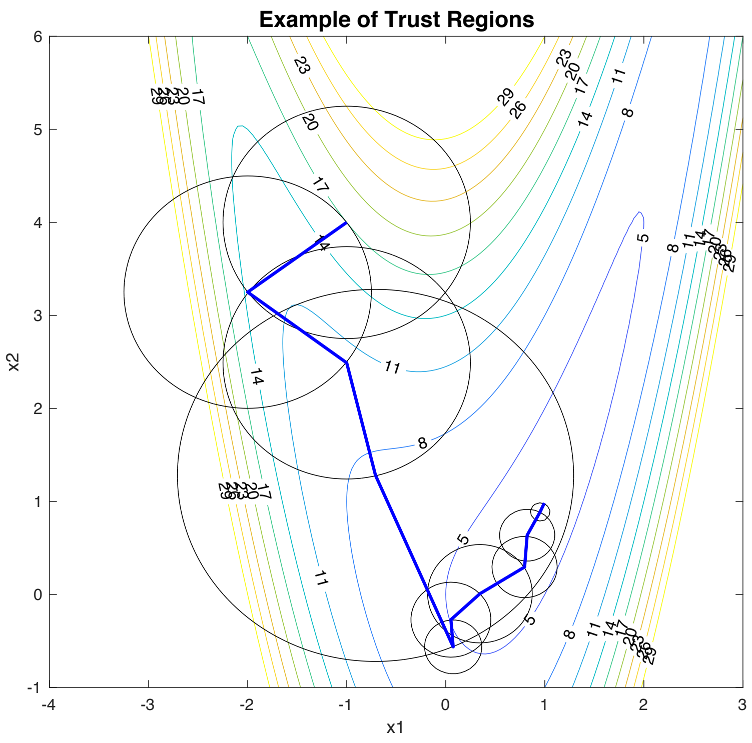

Fig. 3.17. The steps of the algorithm, showing the trust regions.

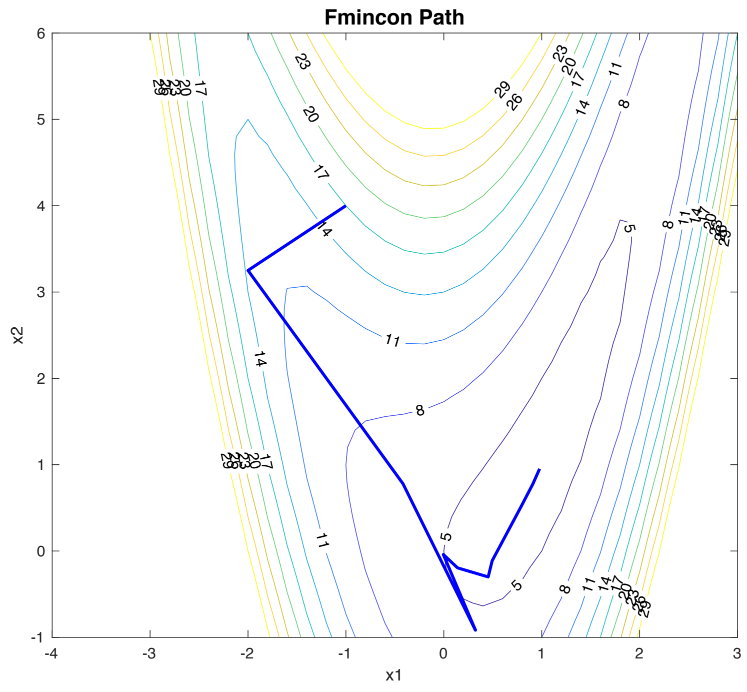

Fig. 3.18 The path of `fmincon` for ten steps.

3.10.4 Comparison of Trust Region to Exact Line Search

Trust regions, which allow the search direction and step length to change together, should be more efficient than methods that keep the search direction fixed while changing the step length. However, this gain in efficiency to be balanced against the property that when paired with exact line searches, quasi-Newton methods solve a quadratic function of *n* variables in *n* steps. The BFGS method loses this property when inexact line searches are used. Also, methods using exact line searches will tend to take fewer iterations, with more function calls per iteration. If derivatives are being evaluated numerically (since we take derivatives each iteration), the gain in efficiency of the Trust Region method may be canceled out by the computation required to take more derivatives. In the table below we provide some very limited data comparing the two.

In the table, the number of objective function evaluations ("Obj Evals") and gradient evaluations ("Grad Evals") are given. Gradients were evaluated analytically; if these had been done numerically, using, for example, a finite forward different scheme requiring one evaluation per partial derivative, then the total evaluations would be given by "Total Evals."

Name	# Vars	Exact Obj. Evals	Exact Grad. Evals (Iterations)	Exact Total Evals (including gradient)	Trust Region Obj. Evals.	Trust Region Grad. Evals (Iterations)	Trust Total Evals (including gradient)
Quad 1	2	21	3	27	7	7	**21**
Quad 2	2	23	3	29	8	8	**24**
Rosenbrock	2	134	22	178	37	37	**111**
4th order	2	54	8	70	16	16	**48**
Rosenbrock	3	178	32	274	47	47	**188**
Quad 3	3	36	4	**48**	18	18	72
Raydan 1 (exp)	4	42	7	**70**	16	16	80
Raydan 1 (exp)	8	60	10	**140**	31	31	279

The table shows that for functions of two variables, trust region methods are somewhat more efficient. For functions for four or more variables exact line searches are somewhat more efficient if we are taking gradients numerically using a finite forward difference. Of course, the efficiency of both methods relies on a number of things, such as any heuristics employed and the values of constants used (such as max trust region size, etc.) that are specific to a particular implementation scheme.

3.11 Appendix

3.11.1 The Gradient of a Quadratic Function in Vector Form

We define the coordinate vector to be,

$$\mathbf{e}_i = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow \text{A single 1 in the } i^{th} \text{ position}$$

We note that $\nabla x_i = \mathbf{e}_i$ so

$$\begin{aligned}\nabla \mathbf{x}^{\mathrm{T}} &= \left[\nabla x_1 \ ,\ldots,\ \nabla x_n\right] \\ &= \left[\mathbf{e}_1 \ ,\ldots,\ \mathbf{e}_n\right] = \mathbf{I}\end{aligned}$$

Suppose we have a linear function:

$$f\left(\mathbf{x}\right) = a + \mathbf{b}^{\mathrm{T}}\mathbf{x}$$

then

$$\nabla f\left(\mathbf{x}\right) = \nabla\left(a + \ \mathbf{b}^{\mathrm{T}}\mathbf{x}\right) = \underbrace{\nabla a}_{term\,1} + \underbrace{\nabla\left(\mathbf{b}^{\mathrm{T}}\mathbf{x}\right)}_{term\,2}$$

For the first term, since a is a constant, $\nabla a = 0$. Looking at the second term, from the rule for differentiation of a product,

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

but $\nabla \mathbf{b}^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}}$ and $\nabla \mathbf{x}^{\mathrm{T}} = \mathbf{I}$

Thus

$$\begin{aligned}\nabla f\left(\mathbf{x}\right) &= \nabla a + \nabla\left(\mathbf{b}^{\mathrm{T}}\mathbf{x}\right) \\ &= 0 + \left(\nabla \mathbf{b}^{\mathrm{T}}\right)\mathbf{x} + \left(\nabla \mathbf{x}^{\mathrm{T}}\right)\mathbf{b} \\ &= 0 + 0 + \mathbf{I}\mathbf{b} \\ &= \mathbf{b}\end{aligned}$$

Now suppose we have a quadratic function of the form:

$$q\left(\mathbf{x}\right) = a + \mathbf{b}^{\mathrm{T}}\mathbf{x} + \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x}$$

We wish to evaluate the gradient in vector form. We will do this term by term,

$$\nabla q(\mathbf{x}) = \nabla a + \nabla\left(\mathbf{b}^{\mathrm{T}}\mathbf{x}\right) + \frac{1}{2}\nabla\left(\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x}\right)$$

Applying the results from a linear function,

$$\begin{aligned}\nabla q(\mathbf{x}) &= \nabla a + \nabla\left(\mathbf{b}^{\mathrm{T}}\mathbf{x}\right) + \frac{1}{2}\nabla\left(\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x}\right)\\ &= 0 + \mathbf{b} + \frac{1}{2}\nabla\left(\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x}\right)\end{aligned}$$

So we only need to evaluate the term, $\frac{1}{2}\nabla\left(\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x}\right)$. If we split this into two vectors, i.e. $\mathbf{u} = \mathbf{x},\ \mathbf{v} = \mathbf{H}\mathbf{x}$, then

$$\nabla\left(\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x}\right) = \left(\nabla\mathbf{x}^{\mathrm{T}}\right)\mathbf{v} + \left(\nabla\mathbf{v}^{\mathrm{T}}\right)\mathbf{x}$$

We know $\left(\nabla\mathbf{x}^{\mathrm{T}}\right)\mathbf{v} = \mathbf{I}\mathbf{H}\mathbf{x} = \mathbf{H}\mathbf{x}$, so we must only evaluate $\left(\nabla\mathbf{v}^{\mathrm{T}}\right)\mathbf{x} = \left(\nabla\left(\mathbf{H}\mathbf{x}\right)^{\mathrm{T}}\right)\mathbf{x}$. We can write,

$$\left(\mathbf{H}\mathbf{x}\right)^{\mathrm{T}} = [\mathbf{h}_{r1}^{\mathrm{T}}\mathbf{x}\ ,\ \mathbf{h}_{r2}^{\mathrm{T}}\mathbf{x}\ ,\ldots,\ \mathbf{h}_{rn}^{\mathrm{T}}\mathbf{x}]$$

Applying the gradient operator,

$$\begin{aligned}\nabla\left(\mathbf{H}\mathbf{x}\right)^{\mathrm{T}} &= \left[\mathbf{h}_{r1},\ \mathbf{h}_{r2}\ ,\ldots,\ \mathbf{h}_{rn}\right]\\ &= \mathbf{H}^{\mathrm{T}}\end{aligned}$$

Returning now to the gradient of the expression, $q(\mathbf{x}) = a + \mathbf{b}^{\mathrm{T}}\mathbf{x} + \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x}$

$$\begin{aligned}\nabla q(\mathbf{x}) &= \nabla a + \nabla\left(\mathbf{b}^{\mathrm{T}}\mathbf{x}\right) + \nabla\left(\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x}\right)\\ &= 0 + \mathbf{b} + \frac{1}{2}\left\{\nabla\left(\mathbf{x}^{\mathrm{T}}\right)\mathbf{H} + \nabla\left(\mathbf{H}\mathbf{x}\right)^{\mathrm{T}}\right\}\mathbf{x}\\ &= \mathbf{b} + \frac{1}{2}\left(\mathbf{H} + \mathbf{H}^{\mathrm{T}}\right)\mathbf{x}\\ &= \mathbf{b} + \mathbf{H}\mathbf{x} \qquad (3.97)\end{aligned}$$

If the quadratic we are approximating is a Taylor expansion,

$$f^{k+1} = f^{k} + \left(\nabla f^{k}\right)^{\mathrm{T}}\Delta\mathbf{x}^{k} + \frac{1}{2}\left(\Delta\mathbf{x}^{k}\right)^{\mathrm{T}}\mathbf{H}^{k}\Delta\mathbf{x}^{k}$$

Then (3.97) is:

$$\nabla f^{k+1} = \nabla f^{k} + \mathbf{H}^{k}\Delta\mathbf{x}^{k} \qquad (3.98)$$

3.11.2 Optimal Step Length for Quadratic Function

In this section we will derive (3.12). If we start with a Taylor expansion,

$$f^{k+1} = f^k + \left(\nabla f^k\right)^{\mathrm{T}} \Delta\mathbf{x}^k + \frac{1}{2}\left(\Delta\mathbf{x}^k\right)^{\mathrm{T}} \mathbf{H}\Delta\mathbf{x}^k \tag{3.99}$$

When we do a line search,

$$\Delta\mathbf{x}^k = \alpha\mathbf{s} \tag{3.100}$$

Substituting (3.100) into (3.99) gives

$$f^{k+1} = f^k + \left(\nabla f^k\right)^{\mathrm{T}} \alpha\mathbf{s} + \frac{1}{2}\left(\alpha\mathbf{s}\right)^{\mathrm{T}} \mathbf{H}\alpha\mathbf{s}$$

If we take the derivative of this expression with respect to α (a scalar),

$$\frac{df^{k+1}}{d\alpha} = \left(\nabla f^k\right)^{\mathrm{T}} \mathbf{s} + \alpha\mathbf{s}^{\mathrm{T}}\mathbf{H}\mathbf{s} \tag{3.101}$$

Setting the derivative equal to zero and solving for α gives:

$$\alpha^* = -\frac{\left(\nabla f^k\right)^{\mathrm{T}} \mathbf{s}}{\mathbf{s}^T\mathbf{H}\mathbf{s}} \tag{3.102}$$

3.11.3 Proof of the Hereditary Property for the Rank One Update

THEOREM. Let $\mathbf{H}$ be a constant symmetric matrix and suppose that $\Delta\mathbf{x}^0, \Delta\mathbf{x}^1, \ldots, \Delta\mathbf{x}^k$ and $\gamma^0, \gamma^1, \ldots, \gamma^k$ are given vectors, where $\gamma^i = \mathbf{H}\Delta\mathbf{x}^i, \quad i = 0,1,\ldots,k$, where $k < n$. Starting with any initial symmetric matrix $\mathbf{N}^0$, let

$$\mathbf{N}^{k+1} = \mathbf{N}^k + \frac{\left(\Delta\mathbf{x}^k - \mathbf{N}^k\boldsymbol{\gamma}^k\right)\left(\Delta\mathbf{x}^k - \mathbf{N}^k\boldsymbol{\gamma}^k\right)^{\mathrm{T}}}{\left(\Delta\mathbf{x}^k - \mathbf{N}^k\boldsymbol{\gamma}^k\right)^{\mathrm{T}}\boldsymbol{\gamma}^k} \tag{3.103}$$

then

$$\mathbf{N}^{k+1}\boldsymbol{\gamma}^i = \Delta\mathbf{x}^i \quad \text{for} \quad i \le k \tag{3.104}$$

PROOF. The proof is by induction. We will show that if (3.104) holds for previous direction matrix, it holds for the current direction matrix. We know that at the current point, *k*, the following is true,

$$\mathbf{N}^{k+1}\boldsymbol{\gamma}^k = \Delta\mathbf{x}^k \tag{3.105}$$

because we enforced this condition when we developed the update. Now, *suppose* it is true that (notice the superscript on **N** is *k*, not *k+1*),

$$\mathbf{N}^k\boldsymbol{\gamma}^i = \Delta\mathbf{x}^i \quad \text{for} \quad i \le k\text{-}1 \tag{3.106}$$

i.e., that the hereditary property holds for the *previous* direction matrix. We can post multiply (3.103) by $\boldsymbol{\gamma}^i$, giving,

$$\mathbf{N}^{k+1}\boldsymbol{\gamma}^i = \mathbf{N}^k\boldsymbol{\gamma}^i + \frac{\left(\Delta\mathbf{x}^k - \mathbf{N}^k\boldsymbol{\gamma}^k\right)\left(\Delta\mathbf{x}^k - \mathbf{N}^k\boldsymbol{\gamma}^k\right)^{\mathrm{T}}\boldsymbol{\gamma}^i}{\left(\Delta\mathbf{x}^k - \mathbf{N}^k\boldsymbol{\gamma}^k\right)^{\mathrm{T}}\boldsymbol{\gamma}^k} \tag{3.107}$$

To simplify things, let $\mathbf{y}^k = \dfrac{\left(\Delta\mathbf{x}^k - \mathbf{N}^k\boldsymbol{\gamma}^k\right)}{\left(\Delta\mathbf{x}^k - \mathbf{N}^k\boldsymbol{\gamma}^k\right)^{\mathrm{T}}\boldsymbol{\gamma}^k}$ so that we can write (3.107) as,

$$\mathbf{N}^{k+1}\boldsymbol{\gamma}^i = \mathbf{N}^k\boldsymbol{\gamma}^i + \mathbf{y}^k\left(\Delta\mathbf{x}^k - \mathbf{N}^k\boldsymbol{\gamma}^k\right)^{\mathrm{T}}\boldsymbol{\gamma}^i \tag{3.108}$$

We can distribute the transpose on the last term, and distribute the post multiplication $\boldsymbol{\gamma}^i$ to give,

$$\mathbf{N}^{k+1}\boldsymbol{\gamma}^i = \mathbf{N}^k\boldsymbol{\gamma}^i + \mathbf{y}^k\left[\left(\Delta\mathbf{x}^k\right)^{\mathrm{T}}\boldsymbol{\gamma}^i - \left(\boldsymbol{\gamma}^k\right)^{\mathrm{T}}\mathbf{N}^k\boldsymbol{\gamma}^i\right] \tag{3.109}$$

Since we have assumed (3.106) is true, we can replace $\mathbf{N}^k\boldsymbol{\gamma}^i$ with $\Delta\mathbf{x}^i$:

$$\mathbf{N}^{k+1}\boldsymbol{\gamma}^i = \Delta\mathbf{x}^i + \mathbf{y}^k\left[\left(\Delta\mathbf{x}^k\right)^{\mathrm{T}}\boldsymbol{\gamma}^i - \left(\boldsymbol{\gamma}^k\right)^{\mathrm{T}}\Delta\mathbf{x}^i\right] \tag{3.110}$$

Now we examine the term in brackets. We note that,

$$\left(\boldsymbol{\gamma}^k\right)^{\mathrm{T}}\Delta\mathbf{x}^i = \left(\mathbf{H}\Delta\mathbf{x}^k\right)^{\mathrm{T}}\Delta\mathbf{x}^i = \left(\Delta\mathbf{x}^k\right)^{\mathrm{T}}\mathbf{H}\Delta\mathbf{x}^i = \left(\Delta\mathbf{x}^k\right)^{\mathrm{T}}\boldsymbol{\gamma}^i \tag{3.111}$$

So the term in brackets in (3.110) vanishes, giving,

$$\mathbf{N}^{k+1}\gamma^i = \Delta\mathbf{x}^i \quad \textit{for } i \le k-1 \tag{3.112}$$

and since (3.105) is satisfied by definition for the update, i.e, for *i=k,* we have

$$\mathbf{N}^{k+1}\gamma^i = \Delta\mathbf{x}^i \quad \textit{for } i \le k \tag{3.113}$$

Thus, if the hereditary property holds for the previous direction matrix, it holds for the current direction matrix. The expression (3.105) is all that is needed to have the hereditary property for the first update, $\mathbf{N}^1$. The second update, $\mathbf{N}^2$, will then have the hereditary property since $\mathbf{N}^1$ does, and so on.

3.11.4 Proof that an Update with the Hereditary Property is Also a Method of Conjugate Directions

THEOREM. An update with the hereditary property and exact line searches is a method of conjugate directions and therefore terminates after $m \le n$ iterations on a quadratic function.

We assume that the hereditary property holds for $k = 1, 2, ..., m$

$$\mathbf{N}^{k+1}\boldsymbol{\gamma}^{i} = \Delta\mathbf{x}^{i} \quad \text{for all } i \le k \tag{3.114}$$

We need to show that conjugacy holds as well,

$$\left(\mathbf{s}^{\mathrm{k}}\right)^{T}\mathbf{H}\mathbf{s}^{i} = 0 \quad \text{for all } i \le k-1 \tag{3.115}$$

The proof is by induction. We will show that if $\mathbf{s}^{k}$ is conjugate then $\mathbf{s}^{k+1}$ is as well, i.e.,

$$\left(\mathbf{s}^{\mathrm{k+1}}\right)^{T}\mathbf{H}\mathbf{s}^{i} = 0 \quad \text{for all } i \le k \tag{3.116}$$

We note that

$$\mathbf{s}^{k+1} = -\mathbf{N}^{k+1}\nabla f^{k+1} \tag{3.117}$$

by definition of the quasi-Newton method. Or taking the transpose,

$$\left(\mathbf{s}^{k+1}\right)^{T} = -\left(\nabla f^{k+1}\right)^{T}\mathbf{N}^{k+1} \tag{3.118}$$

Substituting (3.118) into (3.117);

$$\left(\mathbf{s}^{\mathrm{k+1}}\right)^{T}\mathbf{H}\mathbf{s}^{i} = -\left(\nabla f^{k+1}\right)^{T}\mathbf{N}^{k+1}\mathbf{H}\mathbf{s}^{i} \quad \text{for all } i \le k \tag{3.119}$$

Also,

$$\mathbf{H}\mathbf{s}^{i} = \frac{\mathbf{H}\Delta\mathbf{x}^{i}}{\alpha^{i}} = \frac{\boldsymbol{\gamma}^{i}}{\alpha^{i}}$$

so (3.119) becomes,

$$\left(\mathbf{s}^{\mathrm{k+1}}\right)^{T}\mathbf{H}\mathbf{s}^{i} = -\frac{\left(\nabla f^{k+1}\right)^{T}\mathbf{N}^{k+1}\boldsymbol{\gamma}^{i}}{\alpha^{i}} \quad \text{for all } i \le k \tag{3.120}$$

From the hereditary property we have $\mathbf{N}^{k+1}\boldsymbol{\gamma}^{i} = \Delta\mathbf{x}^{i} \quad i \le k$, so (3.106) can be written,

$$\left(\mathbf{s}^{k+1}\right)^T \mathbf{H}\mathbf{s}^i = -\left[\frac{\left(\nabla f^{k+1}\right)^T \Delta\mathbf{x}^i}{\alpha^i}\right] = 0 \quad \text{for all } i \le k$$

The term in brackets is zero for all values of $i = 1, 2, ..., k-1$ from the assumption the previous search direction was conjugate. It is zero for $i = k$ from the definition of α^*. Thus if we have conjugate directions at *k*, and the hereditary property holds, we have conjugate directions at *k+1*.

3.11.5 Proof the DFP Update Stays Positive Definite

THEOREM. If $\left(\Delta\mathbf{x}^k\right)^T \boldsymbol{\gamma} > 0$ for all steps of the algorithm, and if we start with any symmetric, positive definite matrix, $\mathbf{N}^0$, then the DFP update preserves the positive definiteness of $\mathbf{N}^{\mathrm{k}}$ for all k.

PROOF. The proof is inductive. We will show that if $\mathbf{N}^{\mathrm{k}}$ is positive definite, $\mathbf{N}^{\mathrm{k+1}}$ is also. From the definition of positive definiteness,

$$\mathbf{z}^{\mathrm{T}}\mathbf{N}^{k+1}\mathbf{z} > 0 \quad \text{for all } \mathbf{z} \ne 0$$

For simplicity we will drop the superscript *k* on the update terms. From (3.66),

$$\mathbf{z}^{\mathrm{T}}\mathbf{N}^{k+1}\mathbf{z} = \underbrace{\mathbf{z}^{\mathrm{T}}\mathbf{N}^{k}\mathbf{z}}_{term\,1} + \underbrace{\mathbf{z}^{\mathrm{T}}\left(\frac{\Delta\mathbf{x}\Delta\mathbf{x}^{\mathrm{T}}}{\Delta\mathbf{x}^{\mathrm{T}}\gamma}\right)\mathbf{z}}_{term\,2} - \underbrace{\mathbf{z}^{\mathrm{T}}\left(\frac{\mathbf{N}\gamma\gamma^{\mathrm{T}}\mathbf{N}}{\gamma^{\mathrm{T}}\mathbf{N}\gamma}\right)\mathbf{z}}_{term\,3} \tag{3.121}$$

We need to show that all the terms on the right hand side are positive. We will focus for a moment on the first and third terms on the right hand side. Noting that $\mathbf{N}$ can be written as $\mathbf{N} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$ via Choleski decomposition, and if we substitute $\mathbf{a} = \mathbf{L}^{\mathrm{T}}\mathbf{z},\ \mathbf{a}^{\mathrm{T}} = \mathbf{z}^{\mathrm{T}}\mathbf{L}$, $\mathbf{b} = \mathbf{L}^{\mathrm{T}}\boldsymbol{\gamma},\ \mathbf{b}^{\mathrm{T}} = \boldsymbol{\gamma}^{\mathrm{T}}\mathbf{L}$ the first and third terms are,

$$\mathbf{z}^{\mathrm{T}}\mathbf{N}\mathbf{z} - \mathbf{z}^{\mathrm{T}}\left(\frac{\mathbf{N}\boldsymbol{\gamma}\boldsymbol{\gamma}^{\mathrm{T}}\mathbf{N}}{\boldsymbol{\gamma}^{\mathrm{T}}\mathbf{N}\boldsymbol{\gamma}}\right)\mathbf{z} = \mathbf{a}^{\mathrm{T}}\mathbf{a} - \frac{\left(\mathbf{a}^{\mathrm{T}}\mathbf{b}\right)^2}{\mathbf{b}^{\mathrm{T}}\mathbf{b}} \tag{3.122}$$

The Cauchy-Schwarz inequality states that for any two vectors, **x** and **y**,

$$\mathbf{x}^{\mathrm{T}}\mathbf{x} \ge \frac{\left(\mathbf{x}^{\mathrm{T}}\mathbf{y}\right)^2}{\mathbf{y}^{\mathrm{T}}\mathbf{y}} \qquad \text{thus} \qquad \mathbf{a}^{\mathrm{T}}\mathbf{a} - \frac{\left(\mathbf{a}^{\mathrm{T}}\mathbf{b}\right)^2}{\mathbf{b}^{\mathrm{T}}\mathbf{b}} \ge 0 \tag{3.123}$$

So the first and third terms of (3.78) are positive. Now we need to show this for the second term,

$$\mathbf{z}^{\mathrm{T}}\left(\frac{\Delta\mathbf{x}\Delta\mathbf{x}^{\mathrm{T}}}{\Delta\mathbf{x}^{\mathrm{T}}\boldsymbol{\gamma}}\right)\mathbf{z} = \frac{\mathbf{z}^{\mathrm{T}}\Delta\mathbf{x}\Delta\mathbf{x}^{\mathrm{T}}\mathbf{z}}{\Delta\mathbf{x}^{\mathrm{T}}\boldsymbol{\gamma}} = \frac{\left(\mathbf{z}^{\mathrm{T}}\Delta\mathbf{x}\right)^{2}}{\Delta\mathbf{x}^{\mathrm{T}}\boldsymbol{\gamma}} \tag{3.124}$$

The numerator of the right-most expression is obviously positive. The denominator can be written,

$$\Delta\mathbf{x}^{\mathrm{T}}\gamma = \left(\Delta\mathbf{x}^{k}\right)^{\mathrm{T}}\nabla f^{k+1} - \left(\Delta\mathbf{x}^{k}\right)^{\mathrm{T}}\nabla f^{k} = \underbrace{\alpha\left(\mathbf{s}^{k}\right)^{\mathrm{T}}\nabla f^{k+1}}_{term\,1} - \underbrace{\alpha\left(\mathbf{s}^{k}\right)^{\mathrm{T}}\nabla f^{k}}_{term\,2} \tag{3.125}$$

The second term in (3.125), $\left(\mathbf{s}^{k}\right)^{\mathrm{T}}\nabla f^{k}$, is negative if the search direction goes downhill which it does if $\mathbf{N}^{k}$ is positive definite, and with the minus sign is therefore positive. The first term in (3.125), $\alpha\left(\mathbf{s}^{k}\right)^{\mathrm{T}}\nabla f^{k+1}$, can be positive or negative; however, it is zero if we are at α^*; thus the entire expression in (3.125) is positive if we take a minimizing step, α^*.

We have now shown that all three terms of (3.121) are positive if we take a minimizing step. Thus, if $\mathbf{N}^{k}$ is positive definite, $\mathbf{N}^{k+1}$ is positive definite, etc.