Chapter 2

Large, Sparse Nonlinear Programming

2.1 Overview: Large, Sparse NLP Issues

Chapter 1 presents background on nonlinear programming (NLP) methods that are applicable to most problems. In this chapter, we will focus on NLP methods that are appropriate for problems that are both *large* and *sparse*. To set the stage for what follows, it is useful to define what we mean by large and sparse. The definition of "large" is closely tied with the tools available for solving linear systems of equations. For our purposes, we will consider a problem "small" if the underlying linear systems can be solved using a dense, direct matrix factorization. Solving a dense linear system using a direct factorization requires $\mathcal{O}(n^3)$ arithmetic operations. Thus, for today's computers, this suggests that the problem size is probably limited to matrices of order $n \approx 1000$. If the linear equations are solved using methods that exploit sparsity in the matrices but still use direct matrix factorizations, then with current computer hardware, the upper limit on the size of the problem is $n \approx 10^6$. This problem is considered "large" and is the focus of methods in this book. Although some of the NLP methods can be extended, as a rule we will not address "huge" problems for which $n > 10^6$. Typically, linear systems of this size are solved by iterative (as opposed to direct) methods. The second discriminator of interest concerns matrix sparsity. A matrix is said to be "sparse" when many of the elements are zero. For most applications of interest, the number of nonzero elements in the Hessian matrix $\mathbf{H}$ and Jacobian matrix $\mathbf{G}$ is less than 1%.

Many of the techniques described in Chapter 1 extend, without change, to large, sparse problems. On the other hand, there are some techniques that cannot be used for large, sparse applications. In this chapter, we will focus on those issues that are unique to the large, sparse NLP problem.

The first item concerns how to calculate Jacobian and Hessian information for large, sparse problems. All of the Newton-based methods described in Chapter 1 rely on the availability of first and second derivative information. For most practical applications, first derivatives are computed using finite difference approximations. Unfortunately, a single finite difference gradient evaluation can require n additional function evaluations, and when n is large, this computational cost can be excessive. Furthermore, computing second derivatives by naive extensions of the methods described in Chapter 1 becomes unattractive for a number of reasons. First, maintaining a sparse quasi-Newton approximation that is also positive definite appears unpromising. Second, even if we accept an approximation that is indefinite, one can expect to spend $\mathcal{O}(n)$ iterations before a "good" Hessian approximation is constructed—and if n is large, the number of iterations can be excessive! To overcome these drawbacks, current research has focused on *pointwise quasi-Newton updates* [124], [125] and *limited memory updates*. Another alternative is to abandon the quasi-Newton approach altogether and construct this information using sparse finite differences. This technique will be explained in the next section.

The second major item addressed in this chapter concerns how to efficiently construct a Newton step when the underlying matrices are large and sparse. We first concentrate on a sparse QP algorithm and present a detailed description of the method. We then address how the method can be extended to the important sparse nonlinear least squares problem. We conclude with a discussion of a sparse interior-point (barrier) algorithm.

2.2 Sparse Finite Differences

2.2.1 Background

In general, let us define the first derivatives of a set of υ functions $q_i(\mathbf{x})$ with respect to n variables $\mathbf{x}$ by the $\upsilon \times n$ matrix

$$\mathbf{D} \equiv \begin{bmatrix} (\nabla q_1)^\top \\ (\nabla q_2)^\top \\ \vdots \\ (\nabla q_\upsilon)^\top \end{bmatrix} = \frac{\partial \mathbf{q}}{\partial \mathbf{x}}. \tag{2.1}$$

It will also be of interest to compute second derivatives of a linear combination of the functions. In particular, we define the second derivatives of the function

$$\Omega(\mathbf{x}) = \sum_{i=1}^{\upsilon} \omega_i q_i(\mathbf{x}) \tag{2.2}$$

with respect to n variables $\mathbf{x}$ by the $n \times n$ matrix

$$\mathbf{E} \equiv \nabla^2 \Omega(\mathbf{x}) = \sum_{i=1}^{\upsilon} \omega_i \nabla^2 q_i(\mathbf{x}). \tag{2.3}$$

The notion of sparse finite differencing was introduced by Curtis, Powell, and Reid [65]. They proposed that the columns of $\mathbf{D}$ be partitioned into subsets (*index sets*) Γ^k such that each subset has at most one nonzero element per *row*. Then define the perturbation direction vector by

$$\boldsymbol{\Delta}^k = \sum_{j \in \Gamma^k} \delta_j \mathbf{e}_j, \tag{2.4}$$

where δ_j is the perturbation size for variable j and $\mathbf{e}_j$ is a unit vector in direction j. Using this partitioning, it can be demonstrated that the central difference estimates of the first derivatives for $i = 1,\dots,\upsilon$ and $j \in \Gamma^k$ are

$$\mathbf{D}_{ij} \approx \frac{1}{2\delta_j}[q_i(\mathbf{x}+\mathbf{\Delta}^k) - q_i(\mathbf{x}-\mathbf{\Delta}^k)]. \tag{2.5}$$

In a similar fashion, second derivative estimates for $i \in \Gamma^k$ and $j \in \Gamma^\ell$ are

$$\mathbf{E}_{ij} \approx \frac{1}{\delta_i \delta_j}[\Omega(\mathbf{x}+\mathbf{\Delta}^k+\mathbf{\Delta}^\ell) + \Omega(\mathbf{x}) - \Omega(\mathbf{x}+\mathbf{\Delta}^k) - \Omega(\mathbf{x}+\mathbf{\Delta}^\ell)] \tag{2.6}$$

and

$$\mathbf{E}_{ii} \approx \frac{1}{\delta_i^2}[\Omega(\mathbf{x}+\mathbf{\Delta}^k) + \Omega(\mathbf{x}-\mathbf{\Delta}^k) - 2\Omega(\mathbf{x})]. \tag{2.7}$$

Denote the total number of index sets Γ^k needed to span the columns of $\mathbf{D}$ by γ. Now observe that by using the same index sets for the first and second derivatives, it is possible to compute

1. central difference first derivatives using 2γ perturbations and
2. first and second derivatives using $\gamma(\gamma+3)/2$ perturbations.

Since a function evaluation can be costly, it is clear that the number of index sets γ determines the cost of constructing this derivative information. It can be demonstrated that the maximum number of nonzeros in any row of $\mathbf{D}$ is a lower bound on the number γ. Coleman and Moré [64] have also shown that computing the smallest number of index sets is a graph-coloring problem. For most applications, acceptable approximations to the minimum number of index sets can be achieved using the "greedy algorithm" suggested by Curtis, Powell, and Reid [65]. Regardless of how the index sets are constructed, the important point is that for the optimal control problems of interest, $\gamma \ll n$. In simple terms, the number of perturbations is much smaller than the number of variables.

There are two computational issues that should be emphasized with regard to a sparse differencing implementation. First, from direct inspection of (2.3), it might appear that storage is needed for all of the individual Hessian matrices $\nabla^2 q_i(\mathbf{x})$. This is not true! In fact, the calculations can be organized such that the summation is done first, thereby making it necessary to store only the result. Second, this procedure uses a single perturbation size to construct difference approximations for many functions. As such, choosing the "best" perturbation to balance truncation and roundoff errors as described in Section 1.17 for all of the functions is a compromise, and some inaccuracy can be expected. Nevertheless, for well-scaled functions, this is not a significant issue, especially because the central difference gradient information given by (2.5) is $\mathcal{O}(\delta^2)$.

2.2.2 Sparse Hessian Using Gradient Differences

Equations (2.6) and (2.7) provide a means to compute an approximation to the sparse Hessian directly from the function values $\Omega(\mathbf{x})$. On the other hand if gradients are readily available, this information can be used to construct the Hessian as proposed by Coleman, Garbow, and Moré [63]. Since the approximate Hessian $\mathbf{E}$ is a symmetric matrix of order n, the goal is to obtain a set of vectors $\Delta^1, \Delta^2, \dots, \Delta^\gamma$ such that $\mathbf{E}$ is uniquely determined by the products $\mathbf{E}\Delta^1, \mathbf{E}\Delta^2, \ldots, \mathbf{E}\Delta^\gamma$. Thus a forward difference estimate in the direction $\boldsymbol{\Delta}^k$ is expressed as

$$\mathbf{E}\Delta^k = \nabla^2\Omega(\mathbf{x})\Delta^k = \nabla\Omega(\mathbf{x}+\boldsymbol{\Delta}^k) - \nabla\Omega(\mathbf{x}) \tag{2.8}$$

and a central difference estimate is given by

$$\mathbf{E}\Delta^k = \nabla^2\Omega(\mathbf{x})\Delta^k = \frac{1}{2}\Big[\nabla\Omega(\mathbf{x}+\boldsymbol{\Delta}^k) - \nabla\Omega(\mathbf{x}-\boldsymbol{\Delta}^k)\Big]. \tag{2.9}$$

Given a sparsity pattern for the symmetric matrix $\mathbf{E}$, they determine a symmetric permutation of $\mathbf{E}$ and a partition of the columns of $\mathbf{E}$ (i.e., the index sets), consistent with determination of $\mathbf{E}$ by a lower triangular substitution method. Two techniques for constructing the index sets are implemented in their software, namely a direct method and an indirect method. Their computational experience (cf. [63]) suggests the indirect method yields a smaller number of index sets γ, whereas the direct method produces more accurate Hessian approximations.

2.2.3 Sparse Differences in Nonlinear Programming

When a finite difference method is used to construct the Jacobian, it is natural to identify the constraint functions as the quantities being differentiated in (2.1). In other words,

$$\mathbf{q} = \begin{bmatrix} \mathbf{c} \\ F \end{bmatrix} \tag{2.10}$$

and

$$\mathbf{D} = \begin{bmatrix} \mathbf{G} \\ \mathbf{g}^\mathsf{T} \end{bmatrix}. \tag{2.11}$$

It is also natural to define

$$\boldsymbol{\omega}^\mathsf{T} = (-\lambda_1, \ldots, -\lambda_m, 1), \tag{2.12}$$

where λ_k are the Lagrange multipliers with $\upsilon = m+1$, so that (2.2)

$$\Omega(\mathbf{x}) = \sum_{i=1}^{\upsilon} \omega_i q_i(\mathbf{x}) = F - \sum_{i=1}^{m} \lambda_i c_i(\mathbf{x}) = L(\mathbf{x}, \boldsymbol{\lambda}) \tag{2.13}$$

is the *Lagrangian* for the NLP problem. Clearly, it follows that the Hessian of the Lagrangian $\mathbf{H} = \mathbf{E}$, where $\mathbf{E}$ is given by (2.3).

2.3 Sparse QP Subproblem

The efficient solution of the sparse QP subproblem can be achieved using a method proposed by Gill et al. [97, 98]. In general, the calculation of a step in a QP subproblem requires the solution of the KKT system (1.66) as described in Sections 1.8 and 1.10. For convenience, recall that the *QP subproblem* (1.102)–(1.103) is as follows:

Compute $\mathbf{p}$ to minimize a quadratic approximation to the Lagrangian

$$\mathbf{g}^\mathsf{T}\mathbf{p} + \frac{1}{2}\mathbf{p}^\mathsf{T}\mathbf{H}\mathbf{p} \tag{2.14}$$

subject to the linear approximation to the constraints

$$\mathbf{b}_L \le \begin{bmatrix} \mathbf{G}\mathbf{p} \\ \mathbf{p} \end{bmatrix} \le \mathbf{b}_U \tag{2.15}$$

with bound vectors defined by

$$\mathbf{b}_L = \begin{bmatrix} \mathbf{c}_L - \mathbf{c} \\ \mathbf{x}_L - \mathbf{x} \end{bmatrix}, \qquad \mathbf{b}_U = \begin{bmatrix} \mathbf{c}_U - \mathbf{c} \\ \mathbf{x}_U - \mathbf{x} \end{bmatrix}. \tag{2.16}$$

First, the QP subproblem (2.14)–(2.15) is restated in the following *standard form*:

Compute $\widetilde{\mathbf{p}}$ to minimize

$$\widetilde{\mathbf{g}}^\mathsf{T}\widetilde{\mathbf{p}} + \frac{1}{2}\widetilde{\mathbf{p}}^\mathsf{T}\widetilde{\mathbf{H}}\widetilde{\mathbf{p}} \tag{2.17}$$

subject to the linear equality constraints

$$\widetilde{\mathbf{G}}\widetilde{\mathbf{p}} = \widetilde{\mathbf{b}} \tag{2.18}$$

and simple bounds

$$\widetilde{\mathbf{p}}_L \le \widetilde{\mathbf{p}} \le \widetilde{\mathbf{p}}_U. \tag{2.19}$$

Notice that this formulation involves only simple bounds (2.19) and equalities (2.18). This transformation can be accomplished by introducing *slack variables*, s_k. For example, a general inequality constraint of the form $\mathbf{a}_k^\mathsf{T}\mathbf{x} \ge b_k$ is replaced by the equality constraint $\mathbf{a}_k^\mathsf{T}\mathbf{x} + s_k = b_k$ and the bound $s_k \le 0$. Notice that we have introduced the tilde notation to indicate that the original variables have been augmented to include the slacks. Observe that in this formulation, when a slack variable is "fixed" on an upper or lower bound, it is equivalent to the original inequality being in the active set. Thus, for a particular QP iteration, the search direction in the "free" variables can be computed by solving the KKT system (1.66), which in this case is

$$\begin{bmatrix} \widetilde{\mathbf{H}}_f & \widetilde{\mathbf{G}}_f^\mathsf{T} \\ \widetilde{\mathbf{G}}_f & \mathbf{0} \end{bmatrix} \begin{bmatrix} -\widetilde{\mathbf{p}}_f \\ \widetilde{\widetilde{\boldsymbol{\lambda}}} \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{g}}_f \\ \mathbf{0} \end{bmatrix}. \tag{2.20}$$

We have used the f subscript to denote quantities corresponding to the "free" variables, and assume the iteration begins at a feasible point so that $\widetilde{\mathbf{b}} = \mathbf{0}$. Let us define the large, sparse symmetric indefinite KKT matrix by

$$\mathbf{K}_0 = \begin{bmatrix} \widetilde{\mathbf{H}}_f & \widetilde{\mathbf{G}}_f^\mathsf{T} \\ \widetilde{\mathbf{G}}_f & \mathbf{0} \end{bmatrix}. \tag{2.21}$$

If the initial estimate of the active set is correct, the solution of the KKT system defines the solution of the QP problem. However, in general, it will be necessary to change the active set and solve a series of equality-constrained problems. In [97, 98], it is demonstrated that the solution to a problem with a new active set can be obtained by the symmetric addition of a row and column to the original $\mathbf{K}_0$, with a corresponding augmentation of the right-hand side. In fact, after k iterations, the KKT system is of dimension n_0+k and has the form

$$\begin{bmatrix} \mathbf{K}_0 & \mathbf{U} \\ \mathbf{U}^\mathsf{T} & \mathbf{V} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_0 \\ \mathbf{w} \end{bmatrix}, \tag{2.22}$$

where $\mathbf{U}$ is $n_0 \times k$ and $\mathbf{V}$ is $k \times k$. The initial right-hand side of (2.20) is denoted by the n_0-vector $\mathbf{f}_0$. The k-vector $\mathbf{w}$ defines the additions to the right-hand side to reflect changes in the active set.

The fundamental feature of the method is that the system (2.22) can be solved using factorizations of $\mathbf{K}_0$ and $\mathbf{C}$, the $k \times k$ *Schur-complement* of $\mathbf{K}_0$:

$$\mathbf{C} \equiv \mathbf{V} - \mathbf{U}^\mathsf{T}\mathbf{K}_0^{-1}\mathbf{U}. \tag{2.23}$$

Using the Schur-complement, the values for $\mathbf{y}$ and $\mathbf{z}$ are computed by solving in turn

$$\mathbf{K}_0\mathbf{v}_0 = \mathbf{f}_0, \tag{2.24}$$

$$\mathbf{C}\mathbf{z} = \mathbf{w} - \mathbf{U}^\mathsf{T}\mathbf{v}_0, \tag{2.25}$$

$$\mathbf{K}_0\mathbf{y} = \mathbf{f} - \mathbf{U}\mathbf{z}. \tag{2.26}$$

Thus, each QP iteration requires one solve with the factorization of $\mathbf{K}_0$ and one solve with the factorization of $\mathbf{C}$. The solve for $\mathbf{v}_0$ needs to be done only once at the first iteration. Each change in the active set adds a new row and column to $\mathbf{C}$. It is relatively straightforward to update both $\mathbf{C}$ and its factorization to accommodate the change. It is important to keep $\mathbf{C}$ small enough to maintain a stable, dense factorization. This is achieved by refactoring the entire KKT matrix whenever $k > 100$. In general, the penalty for refactoring may be substantial. However, when the QP algorithm is used within the general NLP algorithm, it is possible to exploit previous estimates of the active set to give the QP algorithm a "warm start." In fact, as the NLP algorithm approaches a solution, it is expected that the active set will be correctly identified and the resulting number of iterations k for the QP subproblem will become small.

The Schur-complement method derives its efficiency from two facts. First, the KKT matrix is factored only *once* using a very efficient *multifrontal algorithm* [4]. This software solves $\mathbf{A}\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$, where $\mathbf{A}$ is an $n \times n$ real *symmetric indefinite* sparse matrix. Since $\mathbf{A}$ is symmetric, it can be factored as $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^\mathsf{T}$, where $\mathbf{L}$ is a unit lower-triangular matrix and $\mathbf{D}$ is a block-diagonal matrix composed solely of 1×1 and 2×2 blocks. Since $\mathbf{A}$ is not necessarily positive definite, pivoting to preserve stability is required. The package uses the threshold-pivoting generalization of Bunch and Kaufman with 2×2 block pivoting for sparse symmetric indefinite matrices. The software requires storage for the nonzero elements in the lower-triangular portion of the matrix and a work array. Second, subsequent changes to the QP active set can be computed using a *solve* with the previously factored KKT matrix and a *solve* with the small, dense Schur-complement matrix. Since the factorization of the KKT matrix is significantly more expensive than the solve operation, the overall method is quite effective. The algorithm is described in [32].

2.4 Merit Function

When a QP algorithm is used to approximate a general nonlinearly constrained problem, it may be necessary to adjust the steplength α in order to achieve "sufficient reduction" in a merit function as discussed in Section 1.11. The merit function we use is a modified version of (1.110), which was proposed by Gill et al. in [95]. It is related to the function given by Rockafellar in [151]:

$$\begin{aligned} M(\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\nu},\mathbf{s},\mathbf{t}) = F - \boldsymbol{\lambda}^{\mathsf{T}}(\mathbf{c}-\mathbf{s}) - \boldsymbol{\nu}^{\mathsf{T}}(\mathbf{x}-\mathbf{t}) \\ + \frac{1}{2}(\mathbf{c}-\mathbf{s})^{\mathsf{T}}\boldsymbol{\Theta}(\mathbf{c}-\mathbf{s}) + \frac{1}{2}(\mathbf{x}-\mathbf{t})^{\mathsf{T}}\boldsymbol{\Xi}(\mathbf{x}-\mathbf{t}). \end{aligned} \tag{2.27}$$

The diagonal penalty matrices are defined by $\boldsymbol{\Theta}_{ii} = \theta_i$ and $\boldsymbol{\Xi}_{ii} = \xi_i$ for $\theta_i > 0$ and $\xi_i > 0$. The merit function is written to explicitly include terms for the bounds that were not present in the original formulation (1.110) [95]. For this merit function, the slack variables **s** and **t** at the beginning of a step are defined by

$$s_i = \begin{cases} c_{Li} & \text{if } c_{Li} > c_i - \lambda_i/\theta_i, \\ c_i - \lambda_i/\theta_i & \text{if } c_{Li} \le c_i - \lambda_i/\theta_i \le c_{Ui}, \\ c_{Ui} & \text{if } c_i - \lambda_i/\theta_i > c_{Ui}; \end{cases} \tag{2.28}$$

$$t_i = \begin{cases} x_{Li} & \text{if } x_{Li} > x_i - \nu_i/\xi_i, \\ x_i - \nu_i/\xi_i & \text{if } x_{Li} \le x_i - \nu_i/\xi_i \le x_{Ui}, \\ x_{Ui} & \text{if } x_i - \nu_i/\xi_i > x_{Ui}. \end{cases} \tag{2.29}$$

These expressions for the slack variables yield a minimum value for the merit function M for given values of the variables $\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\nu}$ and penalty weights subject to the bounds on the slacks. The search direction in the real variables **x** is augmented to permit the multipliers and the slack variables to vary according to

$$\begin{bmatrix} \bar{\mathbf{x}} \\ \bar{\boldsymbol{\lambda}} \\ \bar{\boldsymbol{\nu}} \\ \bar{\mathbf{s}} \\ \bar{\mathbf{t}} \end{bmatrix} = \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \\ \boldsymbol{\nu} \\ \mathbf{s} \\ \mathbf{t} \end{bmatrix} + \alpha \begin{bmatrix} \mathbf{p} \\ \Delta\boldsymbol{\lambda} \\ \Delta\boldsymbol{\nu} \\ \Delta\mathbf{s} \\ \Delta\mathbf{t} \end{bmatrix}. \tag{2.30}$$

The multiplier search directions, $\Delta\boldsymbol{\lambda}$ and $\Delta\boldsymbol{\nu}$, are defined using the QP multipliers $\widehat{\boldsymbol{\lambda}}$ and $\widehat{\boldsymbol{\nu}}$ according to

$$\Delta\boldsymbol{\lambda} \equiv \widehat{\boldsymbol{\lambda}} - \boldsymbol{\lambda} \tag{2.31}$$

and

$$\Delta\boldsymbol{\nu} \equiv \widehat{\boldsymbol{\nu}} - \boldsymbol{\nu}. \tag{2.32}$$

As in (1.107), the predicted slack variables are just

$$\bar{\mathbf{s}} = \mathbf{G}\mathbf{p} + \mathbf{c} = \mathbf{s} + \Delta\mathbf{s}. \tag{2.33}$$

Using this expression, the slack-vector step analogous to (1.108) is just

$$\Delta\mathbf{s} = \mathbf{G}\mathbf{p} + (\mathbf{c} - \mathbf{s}). \tag{2.34}$$

A similar technique defines the bound slack-vector search direction

$$\Delta \mathbf{t} = \mathbf{p} + (\mathbf{x} - \mathbf{t}). \tag{2.35}$$

Note that when a full step is taken ($\alpha = 1$), the updated estimates for the Lagrange multipliers $\overline{\boldsymbol{\lambda}}$ and $\overline{\boldsymbol{\nu}}$ are just the QP estimates $\widehat{\boldsymbol{\lambda}}$ and $\widehat{\boldsymbol{\nu}}$. The slack variables $\mathbf{s}$ and $\mathbf{t}$ are just the linear estimates of the constraints. The terms $(\mathbf{c}-\mathbf{s})$ and $(\mathbf{x}-\mathbf{t})$ in the merit function are measures of the *deviation* from linearity. In [95], Gill et al. prove global convergence for an SQP algorithm that uses this merit function, provided the QP subproblem has a solution, which requires bounds on the derivatives and Hessian condition number.

It is also necessary to define the penalty weights $\boldsymbol{\Theta}$ and $\boldsymbol{\Xi}$. In [95], it is shown that convergence of the method assumes the weights are chosen such that

$$M_0' \le -\frac{1}{2}\mathbf{p}^{\mathsf{T}}\mathbf{H}\mathbf{p}, \tag{2.36}$$

where M_0' denotes the directional derivative of the merit function (2.27) with respect to the steplength α evaluated at $\alpha = 0$. To achieve this, let us define the vector

$$\Psi_i = \begin{cases} \theta_i - \psi_0 & \text{if} \quad 1 \le i \le m, \\ \xi_{i-m} - \psi_0 & \text{if} \quad m < i \le (m+n), \end{cases}$$

where $\psi_0 > 0$ is a strictly positive "threshold." Since (2.36) provides a single condition for the $(m+n)$ penalty parameters, we make the choice unique by minimizing the norm $\|\boldsymbol{\Psi}\|_2$. This yields

$$\boldsymbol{\Psi} = \mathbf{a}(\mathbf{a}^{\top}\mathbf{a})^{-1}\varsigma, \tag{2.37}$$

where

$$a_i = \begin{cases} (c_i - s_i)^2 & \text{if} \quad 1 \le i \le m, \\ (x_{i-m} - t_{i-m})^2 & \text{if} \quad m < i \le (m+n), \end{cases}$$

and

$$\begin{aligned} \varsigma = &-\frac{1}{2}\mathbf{p}^{\mathsf{T}}\mathbf{H}\mathbf{p} + \widehat{\boldsymbol{\lambda}}^{\mathsf{T}}\Delta\mathbf{s} + \widehat{\boldsymbol{\nu}}^{\mathsf{T}}\Delta\mathbf{t} - 2(\Delta\boldsymbol{\lambda})^{\mathsf{T}}(\mathbf{c}-\mathbf{s}) - 2(\Delta\boldsymbol{\nu})^{\mathsf{T}}(\mathbf{x}-\mathbf{t}) \\ &-\psi_0(\mathbf{c}-\mathbf{s})^{\mathsf{T}}(\mathbf{c}-\mathbf{s}) - \psi_0(\mathbf{x}-\mathbf{t})^{\mathsf{T}}(\mathbf{x}-\mathbf{t}). \end{aligned}$$

Typically, the threshold parameter ψ_0 is set to machine precision and increased only if the minimum norm solution is zero. In essence, then, the penalty weights are chosen to be as small as possible consistent with the descent condition (2.36).

2.5 Hessian Approximation

A positive definite Hessian matrix ensures that the solution to the QP subproblem is unique and also makes it possible to compute $\boldsymbol{\Theta}$ and $\boldsymbol{\Xi}$ to satisfy the descent condition (2.36). For NLP applications, the Hessian of the Lagrangian is

$$\mathbf{H}_L = \nabla_x^2 F - \sum_{i=1}^{m} \lambda_i \nabla_x^2 c_i. \tag{2.38}$$

An approximation to $\mathbf{H}_L$ can be constructed using the methods described in Section 2.2; however, in general, it is not positive definite. (This does not imply that a finite difference approximation is poor, since the true Hessian may also be indefinite.) In fact, it is necessary only that the reduced Hessian of the Lagrangian be positive definite at the solution with the correct active set of constraints. Similar restrictions are required at $\mathbf{x} \neq \mathbf{x}^*$ to ensure that *each* QP subproblem has a solution. Consequently, for the QP subproblem, we use the modified matrix

$$\mathbf{H} = \mathbf{H}_L + \tau(|\sigma| + 1)\mathbf{I}. \tag{2.39}$$

The parameter τ is chosen such that $0 \le \tau \le 1$ and is normalized using the *Gerschgorin bound* for the most negative eigenvalue of $\mathbf{H}_L$, i.e.,

$$\sigma = \min_{1 \le i \le n} \left\{ h_{ii} - \sum_{i \neq j}^{n} |h_{ij}| \right\}. \tag{2.40}$$

h_{ij} is used to denote the nonzero elements of $\mathbf{H}_L$. An approach for modifying an approximation to the Hessian for least squares problems by the matrix $\bar{\tau}\mathbf{I}$ was originally suggested by Levenberg [131] and, because of this similarity, we refer to τ as the *Levenberg parameter*. As a practical matter, normalization, using the Gerschgorin bound, is useful even though the accuracy of the Gerschgorin estimate is not critical. It is also instructive to recall the trust-region interpretation of the parameter as defined by (1.93).

The proper choice for the Levenberg parameter τ can greatly affect the performance of the NLP algorithm. A fast rate of convergence can be obtained only when $\tau = 0$ and the correct active set has been identified. On the other hand, if $\tau = 1$, in order to guarantee a positive definite Hessian, the search direction $\mathbf{p}$ is significantly biased toward a gradient direction and convergence is degraded. A strategy similar to that used for adjusting a trust region (cf. [82]) is employed by the algorithm to maintain a current value for the Levenberg parameter τ and adjust it from iteration to iteration. The *inertia* (i.e., the number of positive, negative, and zero eigenvalues) of the related KKT matrix (2.21) is used to infer that the reduced Hessian is positive definite. Using results from Gould [104], Gill et al. [97, 98] show that the reduced Hessian will be positive definite if the inertia of $\mathbf{K}_0$ (2.21) is

$$In(\mathbf{K}_0) = (n_f, m, 0), \tag{2.41}$$

where n_f is the number of rows in $\widetilde{\mathbf{H}}_f$ and m is the number of rows in $\widetilde{\mathbf{G}}_f$. Basically, the philosophy is to reduce the Levenberg parameter when the predicted reduction in the merit function agrees with the actual reduction and increase the parameter when the agreement is poor. The process is accelerated by making the change in τ proportional to the observed rate of change in the gradient of the Lagrangian. To be more precise, at iteration k, three quantities are computed, namely

1. the actual reduction
$$\varrho_1 = M^{(k-1)} - M^{(k)}, \tag{2.42}$$

2. the predicted reduction
$$\varrho_2 = M^{(k-1)} - \tilde{M}^{(k)} = -M_0' - \frac{1}{2}\mathbf{p}^\mathsf{T}\mathbf{H}\mathbf{p}, \tag{2.43}$$

   where $\tilde{M}^{(k)}$ is the predicted value of the merit function, and

3. the rate of change in the norm of the gradient of the Lagrangian

$$\varrho_3 = \frac{\|\boldsymbol{\vartheta}^{(k)}\|_\infty}{\|\boldsymbol{\vartheta}^{(k-1)}\|_\infty}, \tag{2.44}$$

where the error in the gradient of the Lagrangian is

$$\boldsymbol{\vartheta} = \mathbf{g} - \mathbf{G}^\mathsf{T}\boldsymbol{\lambda} - \boldsymbol{\nu}. \tag{2.45}$$

Then, if $\varrho_1 \le 0.25\varrho_2$, the actual behavior is much worse than predicted, so bias the step toward the gradient by setting $\tau^{(k+1)} = \min(2\tau^{(k)}, 1)$. On the other hand, if $\varrho_1 \ge 0.75\varrho_2$, then the actual behavior is sufficiently close to predicted, so bias the step toward a Newton direction by setting $\tau^{(k+1)} = \tau^{(k)}\min(0.5, \varrho_3)$. It is important to note that this strategy does not ensure that the reduced Hessian is positive definite. In fact, it may be necessary to supersede this adaptive adjustment and increase $\tau^{(k+1)}$ whenever the inertia of the KKT matrix is incorrect. The inertia is easily computed as a byproduct of the symmetric indefinite factorization by counting the number of positive and negative elements in the diagonal matrix (with a positive and negative contribution coming from each 2×2 block).

2.6 Sparse SQP Algorithm

2.6.1 Minimization Process

Let us now summarize the steps in the algorithm. The iteration begins at the point $\mathbf{x}$, with $k = 1$, and proceeds as follows:

1. *Gradient Evaluation.* Evaluate gradient information $\mathbf{g}$ and $\mathbf{G}$ and then
   (a) evaluate the error in the gradient of the Lagrangian from (2.45);
   (b) terminate if the KKT conditions are satisfied;
   (c) compute $\mathbf{H}_L$ from (2.38); if this is the first iteration, go to step 2; otherwise
   (d) *Levenberg modification:*
      i. compute the rate of change in the norm of the gradient of the Lagrangian from (2.44);
      ii. if $\varrho_1 \le 0.25\varrho_2$, then set $\tau^{(k)} = \min(2\tau^{(k-1)}, 1)$; otherwise
      iii. if $\varrho_1 \ge 0.75\varrho_2$, then set $\tau^{(k)} = \tau^{(k-1)}\min(0.5, \varrho_3)$.
2. *Search Direction.* Construct the optimization search direction:
   (a) compute $\mathbf{H}$ from (2.39);
   (b) compute $\mathbf{p}$ by solving the QP subproblem (2.14)–(2.15);
   (c) *Inertia control:* if inertia of $\mathbf{K}$ is incorrect and
      i. if $\tau^{(k)} < 1$, then set $\tau^{(k)} \leftarrow \min(10\tau^{(k)}, 1)$ and return to step 2(a);
      ii. if $\tau^{(k)} = 1$ and $\mathbf{H} \ne \mathbf{I}$, then set $\tau^{(k)} = 0$ and $\mathbf{H} = \mathbf{I}$ and return to step 2(a);
      iii. if $\mathbf{H} = \mathbf{I}$, then QP constraints are locally inconsistent—terminate the minimization process and attempt to locate a feasible point for the nonlinear constraints;

   (d) compute $\Delta\boldsymbol{\lambda}$ and $\Delta\boldsymbol{\nu}$ from (2.31) and (2.32);

   (e) compute $\Delta\mathbf{s}$ and $\Delta\mathbf{t}$ from (2.34) and (2.35);

   (f) compute penalty parameters to satisfy (2.36); and

   (g) initialize $\alpha = 1$.

3. *Prediction:*

   (a) compute the predicted point for the variables, the multipliers, and the slacks from (2.30);

   (b) evaluate the constraints $\overline{\mathbf{c}} = \mathbf{c}(\overline{\mathbf{x}})$ at the predicted point.

4. *Line Search.* Evaluate the merit function $M(\overline{\mathbf{x}},\overline{\boldsymbol{\lambda}},\overline{\boldsymbol{\nu}},\overline{\mathbf{s}},\overline{\mathbf{t}}) = \bar{M}$ and

   (a) if the merit function $\bar{M}$ is "sufficiently" less than M, then $\overline{\mathbf{x}}$ is an improved point—terminate the line search and go to step 5;

   (b) else change the steplength α to reduce M and return to step 3.

5. *Update.* Update all quantities and set $k = k+1$;

   (a) compute the actual reduction from (2.42);

   (b) compute the predicted reduction from (2.43), where $\tilde{M}^{(k)}$ is the predicted value of the merit function; and

   (c) return to step 1.

The steps outlined describe the fundamental elements of the optimization process; however, a number of points deserve additional clarification. First, note that the algorithm requires a line search in the direction defined by (2.30) with the steplength α adjusted to reduce the merit function. Adjusting the value of the steplength α, as required in step 4(b), is accomplished using a line-search procedure that constructs a quadratic and cubic model of the merit function. The reduction is considered "sufficient" when $M(\alpha) - M(0) < \kappa_1 \alpha M'(0)$. Instead of (1.89), the Wolfe rule, $M'(\alpha) < \kappa_2 M'(0)$ for $0 < \kappa_1 < \kappa_2 < 1$, is imposed to prevent steplengths from becoming too small.

In order to evaluate the Hessian matrix (2.38), an estimate of the Lagrange multipliers is needed. The values obtained by solving the QP problem with $\mathbf{H} = \mathbf{I}$ are used for the first iteration and, thereafter, the values $\overline{\boldsymbol{\lambda}}$ from (2.30) are used. Note that, at the very first iteration, *two* QP subproblems are solved—one to compute first order multiplier estimates and the second to compute the step. Furthermore, for the very first iteration, the multiplier search directions are $\Delta\boldsymbol{\lambda} = 0$ and $\Delta\boldsymbol{\nu} = 0$, so that the multipliers will be initialized to the QP estimates $\overline{\boldsymbol{\lambda}} = \boldsymbol{\lambda} = \widehat{\boldsymbol{\lambda}}$ and $\overline{\boldsymbol{\nu}} = \boldsymbol{\nu} = \widehat{\boldsymbol{\nu}}$. The multipliers are reset in a similar fashion, after a *defective QP subproblem* is encountered, in step 2(c)iii. The subject of defective subproblems will be covered in Section 2.7. The Levenberg parameter τ in (2.39) and the penalty weights θ_i and ξ_i in (2.27) are initialized to zero and, consequently, the merit function is initially just the Lagrangian.

2.6.2 Algorithm Strategy

The basic algorithm described above has been implemented in FORTRAN as part of the SOCS library and is documented in [29]. In the software, the preceding approach is referred to as strategy M since the iterates follow a path from the initial point to the solution. However, in practice it may be desirable and/or more efficient to first locate a feasible point. Consequently, the software provides four different algorithm strategies:

M Minimize. Beginning at $\mathbf{x}^0$, solve a sequence of quadratic programs until the solution $\mathbf{x}^*$ is found.

FM Find a Feasible point and then Minimize. Beginning at $\mathbf{x}^0$, solve a sequence of quadratic programs to locate a feasible point $\mathbf{x}^f$ and then, beginning from $\mathbf{x}^f$, solve a sequence of quadratic programs until the solution $\mathbf{x}^*$ is found.

FME Find a Feasible point and then Minimize subject to Equalities. Beginning at $\mathbf{x}^0$, solve a sequence of quadratic programs to locate a feasible point $\mathbf{x}^f$ and then, beginning from $\mathbf{x}^f$, solve a sequence of quadratic programs while maintaining feasible equalities until the solution $\mathbf{x}^*$ is found.

F Find a Feasible point. Beginning at $\mathbf{x}^0$, solve a sequence of quadratic programs to locate a feasible point $\mathbf{x}^f$.

The default strategy in the software is FM since computational experience suggests that it is more robust and efficient. Details on the FME strategy can be found in [32]. The fourth strategy, F, to locate a feasible point only, is also useful when debugging a new problem formulation.

2.7 Defective Subproblems

The fundamental step in the optimization algorithm requires the solution of a QP subproblem. In Section 1.16, we discussed a number of things that can go wrong that will prevent the solution of the subproblem. In particular, the QP subproblem can be *defective* because

1. the linear constraints (2.15) are inconsistent (i.e., have no solution);
2. the Jacobian matrix $\mathbf{G}$ is rank deficient;
3. the linear constraints (2.15) are redundant or extraneous, which can correspond to Lagrange multipliers that are zero at the solution;
4. the quadratic objective (2.14) is unbounded in the null space of the active constraints.

Unfortunately, because the QP problem is a subproblem within the overall NLP, it is not always obvious how to determine the cause of the difficulty. In particular, the QP subproblem may be defective *locally* simply because the quadratic/linear model does not approximate the nonlinear behavior. On the other hand, the QP subproblem may be defective because the original NLP problem is inherently ill-posed. Regardless of the cause of the defective QP subproblem, the overall algorithm behavior can be significantly impacted. In particular, a defective QP subproblem often produces a large increase in the solution time because

1. there is a large amount of *fill* caused by pivoting for stability in the sparse linear algebra software, which makes the sparse method act like a dense method;

2. there are many iterations in the QP subproblem, which, in turn, necessitates the factorization of a large, dense Schur-complement matrix and/or repeated KKT sparse matrix factorizations.

In order to avoid these detrimental effects, the strategy employed by the NLP algorithm has been constructed to minimize the impact of a defective QP subproblem. The first premise in designing the NLP strategy is to segregate difficulties caused by the constraints from difficulties caused by the objective function. The second basic premise is to design an NLP strategy that will eliminate a defective subproblem rather than solve an ill-conditioned system. This philosophy has been implemented in the default FM strategy. Specifically, we find a feasible point first. During this phase, difficulties that could be attributed to the objective function, Lagrange multipliers, and Hessian matrix are ignored. Instead, difficulties are attributed solely to the constraints during this phase. After a feasible point has been located with a full-rank Jacobian, it is assumed that the constraints are OK. Thus, during the optimization phase, defects related to the objective function are treated. The primary mechanism for dealing with a defective QP subproblem during the optimization process was the Levenberg Hessian modification technique. In particular, note that in step 2(c)i of the algorithm, first the Levenberg parameter is increased. If that fails, in step 2(c)ii, the Hessian is reset to the identity matrix. Only after the above two steps fail is it concluded that the defect in the QP subproblem must be caused by the constraints, and an attempt is made to locate a (nearby) feasible point.

It is worth noting that a defective subproblem is not something unique to SQP methods. In fact, all of the globalization strategies discussed in Section 1.11 can be considered remedies for a defective subproblem. However, it is curious that the remedies we will discuss are computationally attractive for large, sparse problems, but are generally not used for small, dense applications!

2.8 Feasible Point Strategy

2.8.1 QP Subproblem

Finding a point that is feasible with respect to the constraints is the first phase in either the FM or FME strategy and is often used when attempting to deal with a defective problem. The approach employed is to take a series of steps of the form given by (1.87) with the search direction computed to solve a *least distance program* (LDP). This can be accomplished if we impose the requirement that the search direction have minimum norm, i.e., $\|\mathbf{p}\|_2$. The *primary* method for computing the search direction is to minimize

$$\frac{1}{2}\mathbf{p}^{\mathsf{T}}\mathbf{p} \tag{2.46}$$

subject to the linear constraints

$$\mathbf{b}_L \le \begin{bmatrix} \mathbf{G}\mathbf{p} \\ \mathbf{p} \end{bmatrix} \le \mathbf{b}_U. \tag{2.47}$$

For problems with no inequality constraints, the LDP search direction is equivalent to

$$\mathbf{p} = -\mathbf{G}^{\#}\mathbf{b}, \tag{2.48}$$

where $\mathbf{G}^{\#}$ is the *pseudoinverse* of $\mathbf{G}$ and can be viewed as a generalization of the basic Newton step (1.28).

However, as previously suggested, it is possible to encounter a defective subproblem and, in this case, it is useful to construct the search direction from a problem that has a solution even if the Jacobian is singular and/or the linear constraints are inconsistent. Thus, the *relaxation* method requires finding the augmented set of variables $(\mathbf{p}, \mathbf{u})$ to minimize

$$\frac{1}{2}\mathbf{p}^{\mathsf{T}}\mathbf{p} + \frac{\rho}{2}\mathbf{u}^{\mathsf{T}}\mathbf{u} \tag{2.49}$$

subject to the linear constraints

$$\mathbf{b}_L \leq \begin{bmatrix} \mathbf{Gp}+\mathbf{u} \\ \mathbf{p} \end{bmatrix} \leq \mathbf{b}_U, \tag{2.50}$$

where the constant $\rho \gg 0$. Typically, $\rho = 10^6$. Notice that, by adding the residual or slack variables $\mathbf{u}$, the linear equations (2.50) always have a solution. Although the size of the QP problem has been increased, both the Hessian and Jacobian for the augmented problem are sparse. Although the condition number of the KKT matrix for the relaxation QP problem is $\mathcal{O}(\rho^2)$, an accurate solution can often be obtained using a few steps of iterative refinement. It is interesting that the notion of adding variables is *not* usually considered attractive for dense problems because the associated linear algebra cost becomes excessive. However, when sparse matrix techniques are employed, this technique is, in fact, quite reasonable.

Since the solution of this subproblem is based on a linear model of the constraint functions, it may be necessary to adjust the steplength α in (1.87) to produce a reduction in the constraint error. Specifically, a line search is used to adjust α to achieve "sufficient decrease" (1.89) in the constraint violation merit function as defined by

$$M_v(\mathbf{x}) = \sum_{i=1}^{m} \chi^2(c_{Li}, c_i, c_{Ui}) + \sum_{i=1}^{n} \chi^2(x_{Li}, x_i, x_{Ui}) \tag{2.51}$$

with $\chi(l, y, u) \equiv \max[0, l-y, y-u]$.

2.8.2 Feasible Point Strategy

The overall strategy for locating a feasible point can now be described. Beginning at the point $\mathbf{x}$, with the "primary strategy," the procedure is as follows:

1. *Gradient Evaluation.* Evaluate the constraints and Jacobian. Terminate if the constraints are feasible.

2. *Search Direction.* Compute search direction:

   (a) if (*primary strategy*), solve the QP subproblem (2.46)–(2.47), and

      i. if QP solution is possible, then go to step 3; otherwise

      ii. change to *relaxation strategy* and go to step 2(b);

   (b) if (*relaxation strategy*), solve the QP subproblem (2.49)–(2.50).

3. *Line Search.* Choose the steplength α to reduce the constraint violation $M_v(\overline{\mathbf{x}})$ given by (2.51). If relaxation strategy is used, do an accurate line search.

4. *Strategy Modification:*

   (a) if the primary strategy is being used, then return to step 1;

   (b) if the relaxation strategy is being used and if $\alpha = 1$, then switch to primary strategy and return to step 1.

This overall algorithm gives priority to the primary method for computing the search direction. If the primary strategy fails, then presumably there is something defective with the QP subproblem and the relaxation strategy is used. If a switch to the relaxation strategy is made, then subsequent steps use this approach and perform an accurate line search. When full-length steps are taken with the relaxation strategy (i.e., $\alpha = 1$), the primary strategy is again invoked. This logic is motivated by the fact that the relaxation step with $\alpha = 1$ is "approximately" a Newton step and, therefore, it is worthwhile to switch back to the primary (LDP) step. Although this strategy is somewhat ad hoc, it has been quite effective in practice.

Detecting failure of the primary strategy in step 2(a) is based on a number of factors. Specifically, the primary strategy is abandoned whenever

1. the amount of "fill" in the sparse linear system exceeds expectations (implying an ill-conditioned linear system), or

2. the condition number of the KKT matrix is large, or

3. the inertia of the KKT matrix is incorrect, or

4. the number of QP iterations is excessive.

2.8.3 An Illustration

Example 2.1. The performance of the feasible point algorithm is illustrated on an example derived from an ill-conditioned two-point boundary value problem (BVP) (Burgers' equation). The basic problem is to solve

$$\begin{aligned}\dot{y}_1 &= y_2,\\ \dot{y}_2 &= \epsilon^{-1} y_1 y_2,\\ 0 &\le y_1(t)\end{aligned}$$

subject to the boundary conditions $y_1(0) = 2\tanh(\epsilon^{-1})$ and $y_1(1) = 0$. For this illustration, the parameter $\epsilon = 10^{-3}$. The continuous problem is replaced by a discrete approximation with $M = 50$ grid points. Thus, it is required to compute the values of $\mathbf{x}^\mathsf{T} = (y_1(0), y_2(0), \ldots, y_1(1), y_2(1))$ such that the constraints

$$\mathbf{c}(\mathbf{x}) = \mathbf{y}_{j+1} - \mathbf{y}_j - \frac{h}{2}\left[\dot{\mathbf{y}}_{j+1} + \dot{\mathbf{y}}_j\right] = 0$$

for $j = 1,\dots,(M-1)$ are satisfied in addition to the boundary conditions. For this example, the iterations began with a linear initial guess between the boundary conditions

$$\mathbf{y}_k = \mathbf{y}(0) + \frac{(k-1)}{(M-1)}\left[\mathbf{y}(1) - \mathbf{y}(0)\right]$$

for $k = 1,\dots,M$ with $y_2(0) = -2\epsilon^{-1}[1-\tanh(\epsilon^{-1})]$ and $y_2(1) = -2\epsilon^{-1}$. We defer details of the discretization process to subsequent chapters and simply view this as a system of nonlinear equations to be solved by proper choice of the variables $\mathbf{x}$.

The behavior of the algorithm is summarized in Table 2.1. At the first iteration, an attempt to compute the search direction using the primary least distance programming method (ldp) failed, and the relaxation (r) strategy was used. A step of length $\alpha = 1$ reduced the constraint error $\|\mathbf{c}\|$ from 97.3976 to 1.83373. Since a full Newton step was used, the second iteration began with the primary ldp strategy, which also failed, forcing the use of the relaxation method. For the second iteration, the steplength α was 0.317 (which is not a Newton step) and, consequently, the relaxation strategy was employed for iteration 3. At iteration 6, an attempt was made to switch back to the primary strategy, but again it was unsuccessful. Finally, at iteration 9, it was possible to return to the primary strategy, which was then used for all subsequent iterations. Notice that the condition number of the symmetric indefinite KKT system is rather moderate at the solution, even though it is very large for some of the early iterations.

Table 2.1. *Burgers' equation example.*

Iter.	Method	KKT Cond.	α	$\|\mathbf{c}\|$
1	ldp:r	$0.48 \times 10^{+12}$	1.000	97.3976
2	ldp:r	$0.54 \times 10^{+11}$	0.317	1.83373
3	r	$0.26 \times 10^{+13}$	0.149	1.31122
4	r	$0.38 \times 10^{+13}$	0.149	1.16142
5	r	$0.31 \times 10^{+13}$	1.000	1.02214
6	ldp:r	$0.22 \times 10^{+12}$	0.46×10^{-1}	0.337310
7	r	$0.16 \times 10^{+13}$	0.35×10^{-1}	0.323168
8	r	$0.16 \times 10^{+13}$	1.000	0.308115
9	ldp	$0.14 \times 10^{+09}$	0.37×10^{-2}	0.182173
10	ldp	$0.27 \times 10^{+08}$	0.10×10^{-1}	0.181395
11	ldp	$0.50 \times 10^{+06}$	0.88×10^{-1}	0.179771
12	ldp	$0.15 \times 10^{+06}$	0.457	0.165503
13	ldp	$0.78 \times 10^{+05}$	1.000	0.892×10^{-1}
14	ldp	$0.69 \times 10^{+05}$	1.000	0.251×10^{-1}
15	ldp	$0.69 \times 10^{+05}$	1.000	0.748×10^{-4}
16	ldp	$0.69 \times 10^{+05}$	1.000	0.212×10^{-8}

2.9 Computational Experience

2.9.1 Large, Sparse Test Problems

The NLP algorithm described has been tested on problems derived from optimal control and data-fitting applications. An extensive collection of test results for trajectory optimization and optimal control problems is found in [37]. The test set includes simple quadratic programs, nonlinear root solving, and poorly posed problems. Also included are examples with a wide range in the number of degrees of freedom and function nonlinearity. All problems in the test set

1. exhibit a *banded* sparsity pattern for the Jacobian and Hessian and

2. have some active constraints (i.e., no unconstrained problems).

With regard to the first attribute, all problems have some nonzero elements that are not along the diagonal (i.e., they are not separable). On the other hand, none of the problems is characterized by matrices with truly random structure. In general, the results described in [30] are very positive and, consequently, it seems worthwhile to understand the reasons for this promising behavior.

The calculation and treatment of the Hessian matrix are fundamental to the observed performance of the algorithm. As stated, the basic algorithm requires that the Hessian matrix $\mathbf{H}_L$ be computed from (2.38). Observe that the evaluation of $\mathbf{H}_L$ requires an estimate for both the variables and the Lagrange multipliers, i.e., $(\mathbf{x}, \boldsymbol{\lambda})$. Since the accuracy of the Hessian is affected by the accuracy of the multipliers, it seems desirable to use an NLP strategy that tends to produce "accurate" values for $\boldsymbol{\lambda}$. The default FM strategy, which first locates a feasible point and then stays "near" the constraints, presumably benefits from multiplier estimates $\boldsymbol{\lambda}$ that are more accurate near the constraints.

A summary of the results for the test problem set in [30] is given in Figure 2.1. All 109 problems were run using the three optimization strategies (M, FM, FME). The algorithm performance was measured in terms of the number of function evaluations (the number of times $f(\mathbf{x})$ and $\mathbf{c}(\mathbf{x})$ are evaluated) and the solution time. For each test problem, a first-, second-, and third-place strategy has been selected, where the first-place strategy required the smallest number of function evaluations. If a particular strategy failed to solve the problem, this was counted as a failure. It is clear from Figure 2.1 that strategy FM was in first place over 63% of the time. Furthermore, FM was either the best or second-best strategy nearly 89% of the time. Finally, notice that strategy FM solved all 109 problems (no failures). For all but 7 cases, the least number of function evaluations corresponds to the shortest solution time. Consequently, comparing strategies based on run time leads to the same conclusions. These results clearly indicate why strategy FM has been selected as the default. We note that for 3 problems, there are no degrees of freedom, in which case F is the only possible strategy and these cases were eliminated from the comparison.

There are at least three alternatives for computing the Hessian matrix. For some applications, it is possible to evaluate the relevant matrices analytically. Unfortunately, this is often cumbersome. For all of the test results in [30], the sparse finite difference approximations described in Section 2.2 were used. Additional information on sparse differencing is given in [35]. The third alternative, which is often effective for small, dense problems, is to use a quasi-Newton approximation to the Hessian. In general, the errors introduced by finite difference approximations are far smaller than those for quasi-Newton estimates and,

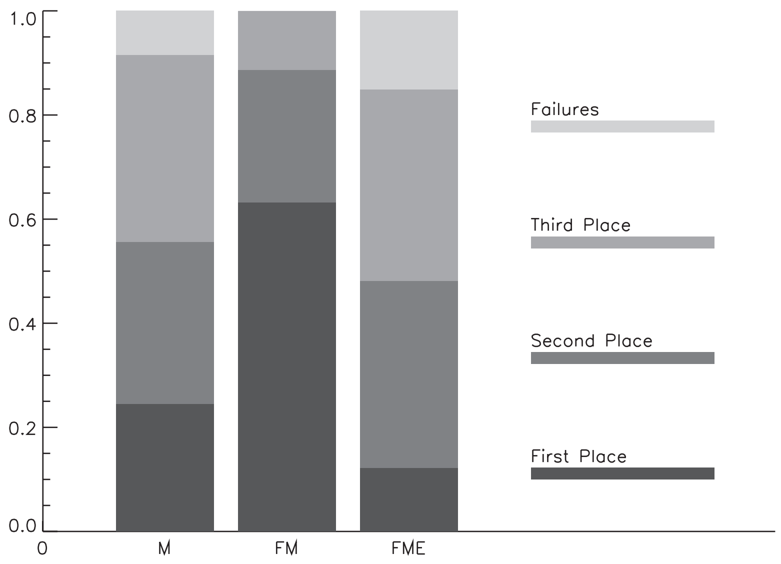

Figure 2.1. *Strategy comparison.*

for all practical purposes, one can consider the first two alternatives to be "exact." Probably the most significant advantage of methods with an exact Hessian is that ultimately one can expect quadratic convergence. The adaptive Levenberg strategy described in (2.42)–(2.45) is designed so that ultimately $\tau = 0$ and $\mathbf{H} = \mathbf{H}_L$ in the QP subproblem.

2.9.2 Small, Dense Test Problems

Although the strategy described was developed to accommodate sparse NLP applications, it is interesting to consider whether the same techniques may also be worthwhile for small, dense problems. One significant feature developed for large, sparse problems is the Levenberg modification technique, which permits using the exact Hessian without altering the matrix sparsity. A second feature is the default FM strategy, which first locates a feasible point. In contrast to the large, sparse case, for small, dense problems it is common to use a quasi-Newton approximation for the Hessian matrix. One possible approximation is the SR1 formula given by (1.49). The update formula requires the change in position given by $\Delta\mathbf{x} = \overline{\mathbf{x}} - \mathbf{x}$. For the SR1 update, it is appropriate to use $\Delta\mathbf{g} = \nabla_x L(\overline{\mathbf{x}}) - \nabla_x L(\mathbf{x})$. If the denominator is zero, making the SR1 correction undefined, we simply skip the update. Now recall that the SR1 recursive estimate is symmetric but not necessarily positive definite. Since the reduced Hessian must be positive definite at the solution, the most common approximation is the BFGS update formula (1.50). In this case, it is possible to keep the entire approximate Hessian positive definite (thereby ensuring the reduced Hessian is positive definite) provided the update also is constructed such that $\Delta\mathbf{x}^\mathsf{T}\Delta\mathbf{g} > 0$. We make an attempt to satisfy this condition by adjusting the Lagrange multiplier estimates used to construct the gradient difference $\Delta\mathbf{g}$. If this fails, the update is skipped. Thus, there are two alternative approaches for incorporating a recursive estimate into the NLP framework described. Since the update generated by the SR1 formula is symmetric, but indefinite, one might expect to generate a "more accurate" approximation to the Hessian $\mathbf{H}_L$. However, as in the case of an exact Hessian, it will be necessary to modify the approximation using the Levenberg strategy. In contrast, the BFGS update will not require any modification to maintain positive definiteness. Nevertheless, the BFGS approximation may not be an accurate estimate for an indefinite Hessian.

Table 2.2 summarizes the results of these different strategies on a set of small, dense test problems. The test set given in [108] consists of 68 test problems, nearly all of them found in the collection by Hock and Schittkowski [114]. The NLP algorithm described was used with three different methods to construct the Hessian. Both the FM and M strategies were employed. Finally, the NPSOL [96] algorithm was used as a benchmark. The *baseline* strategy referred to as SR1-FM incorporates the SR1 update in conjunction with the FM option. All results in Table 2.2 are relative—thus, the second column labeled FDH-FM compares the results for a finite difference Hessian and FM option to the baseline SR1-FM performance. The number of function evaluations (including finite difference perturbations) is the quantitative measure used to assess algorithm performance. By definition, all computed quantities (objective and constraints) are evaluated on a single function evaluation. Thus, when comparing FDH-FM and SR1-FM (reading down the second column of the table), one finds that better results were obtained on 9 problems out of 68, where "better" means that the solution was obtained with fewer function evaluations. Worse results were obtained on 48 of the 68 problems using the FDH-FM option and 3 cases were the same. The FDH-FM option also failed on 1 problem that was solved by the baseline and did not solve any more problems than the baseline. Both options failed to find a solution in 7 cases. The final row in the table presents the *average percentage change* in the number of function evaluations. Thus, on average, the FDH-FM option required 70.89% more function evaluations to obtain a solution than the SR1-FM option. It should be noted that a "failure" can occur because of either an algorithmic factor (e.g., maximum iterations) or a problem characteristic (e.g., no solution exists).

Table 2.2. *Dense test summary.*

Algor.	FDH-FM	BFGS-FM	SR1-M	FDH-M	BFGS-M	NPSOL
Better	9	19	19	7	25	25
Worse	48	25	13	51	33	34
Same	3	15	28	2	7	1
Fail	1	2	1	1	4	2
Solve	0	0	1	1	0	3
Both	7	7	6	6	7	4
% Δ	70.89	22.48	2.016	69.83	22.82	16.88

An analysis of the results in Table 2.2 suggests a number of trends. First, the use of a finite difference Hessian approximation for small, dense problems is much more expensive than a recursive quasi-Newton method. Second, the SR1 update seems to be somewhat better on average than the BFGS update. Presumably this is because the SR1 update yields a better approximation to the Hessian because it does not require positive definiteness. Third, although there is a slight benefit to the FM strategy in comparison to the M strategy, the advantage is not nearly as significant as it is for large, sparse applications. One can speculate that since a recursive Hessian estimate is poor during early iterations, there is no particular advantage to having "good" multiplier estimates. Finally, it is interesting to note that the results in the last two columns comparing NPSOL with the BFGS-M strategy are quite similar, which is to be expected since they employ nearly identical methods. The minor differences in performance are undoubtedly due to different line-search algorithms and other subtle implementation issues.

2.10 Nonlinear Least Squares

2.10.1 Background

In contrast to the general NLP problem as defined by (1.98)–(1.100) in Section 1.12, the nonlinear least squares (NLS) problem is characterized by an objective function

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{r}^{\mathsf{T}}(\mathbf{x})\mathbf{r}(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{\ell}\mathbf{r}_i^2, \tag{2.52}$$

where $\mathbf{r}(\mathbf{x})$ is an ℓ-vector of *residuals*. As for the general NLP, it is necessary to find an n-vector $\mathbf{x}$ to minimize $F(\mathbf{x})$ while satisfying the constraints (1.99) and bounds (1.100). The $\ell \times n$ *residual Jacobian* matrix $\mathbf{R}$ is defined by

$$\mathbf{R}^{\mathsf{T}} = [\nabla\mathbf{r}_1, \ldots, \nabla\mathbf{r}_\ell] \tag{2.53}$$

and the gradient vector is

$$\mathbf{g} = \mathbf{R}^{\mathsf{T}}\mathbf{r} = \sum_{i=1}^{\ell} r_i \nabla\mathbf{r}_i. \tag{2.54}$$

And finally, the Hessian of the Lagrangian is given by

$$\mathbf{H}_L(\mathbf{x}, \boldsymbol{\lambda}) = \sum_{i=1}^{\ell} r_i \nabla^2 r_i - \sum_{i=1}^{m} \lambda_i \nabla^2 c_i + \mathbf{R}^{\mathsf{T}}\mathbf{R} \tag{2.55}$$

$$\equiv \mathbf{V} + \mathbf{R}^{\mathsf{T}}\mathbf{R}. \tag{2.56}$$

The matrix $\mathbf{R}^{\mathsf{T}}\mathbf{R}$ is referred to as the *normal matrix* and we shall refer to the matrix $\mathbf{V}$ as the *residual Hessian*.

2.10.2 Sparse Least Squares

Having derived the appropriate expressions for the gradient and Hessian of the least squares objective function, one obvious approach is to simply use these quantities as required within the framework of a general NLP problem. Unfortunately, there are a number of well-known difficulties that are related to the normal matrix $\mathbf{R}^{\mathsf{T}}\mathbf{R}$. First, it is quite possible that the Hessian $\mathbf{H}_L$ may be dense even when the residual Jacobian $\mathbf{R}$ is sparse. For example, this can occur when there is one dense row in $\mathbf{R}$. Even if the Hessian is not completely dense, in general, formation of the normal matrix introduces "fill" into an otherwise sparse problem.

Second, it is well known that the normal matrix approach is subject to ill-conditioning even for small, dense problems. In particular, formation of $\mathbf{R}^\mathsf{T}\mathbf{R}$ is prone to cancellation, and the condition number of $\mathbf{R}^\mathsf{T}\mathbf{R}$ is the square of the condition number of $\mathbf{R}$. It is for this reason that the use of the normal matrix is not recommended.

For linearly constrained, linear least squares problems, the augmented matrix of particular significance is

$$\widetilde{\mathbf{D}} = \begin{bmatrix} \widetilde{\mathbf{G}} \\ \mathbf{R} \end{bmatrix}, \tag{2.57}$$

where $\widetilde{\mathbf{G}}$ is the $n_d \times n$ Jacobian of the active constraints and bounds. For small, dense problems the preferred solution technique (cf. [127]) is to introduce an orthogonal decomposition for (2.57) without forming the normal matrix. Furthermore the solution is unique if $\widetilde{\mathbf{D}}$ has rank n. Conversely, as example (5.6) illustrates, the linear least squares problem has no unique solution when the number of degrees of freedom $n_d = n - \hat{m}$ exceeds the number of residuals ℓ. Unfortunately, these techniques do not readily generalize to large, sparse systems when it is necessary to repeatedly modify the active set (and, therefore, the factorization). A survey of the various alternatives is found in [110]. In order to ameliorate the difficulties associated with the normal matrix, while maintaining the benefits of the general sparse NLP Schur-complement method, a "sparse tableau" approach has been adopted.

In general, the objective function is approximated by a quadratic model of the form

$$\mathbf{g}^\mathsf{T}\mathbf{p} + \frac{1}{2}\mathbf{p}^\mathsf{T}\mathbf{H}\mathbf{p}.$$

Combining (2.14) with (2.56), let us proceed formally to "complete the square" and derive an alternative representation for the term

$$\begin{aligned}
\mathbf{p}^\mathsf{T}\mathbf{H}\mathbf{p} &= \mathbf{p}^\mathsf{T}\left[\mathbf{V} + \mathbf{R}^\mathsf{T}\mathbf{R}\right]\mathbf{p} \\
&= \mathbf{p}^\mathsf{T}\mathbf{V}\mathbf{p} + \mathbf{p}^\mathsf{T}\mathbf{R}^\mathsf{T}\mathbf{R}\mathbf{p} \\
&= \mathbf{p}^\mathsf{T}\mathbf{V}\mathbf{p} + \mathbf{p}^\mathsf{T}\mathbf{R}^\mathsf{T}\mathbf{R}\mathbf{p} + \mathbf{p}^\mathsf{T}\mathbf{R}^\mathsf{T}\mathbf{R}\mathbf{p} - \mathbf{p}^\mathsf{T}\mathbf{R}^\mathsf{T}\mathbf{R}\mathbf{p} \\
&= \mathbf{p}^\mathsf{T}\mathbf{V}\mathbf{p} + \mathbf{p}^\mathsf{T}\mathbf{R}^\mathsf{T}\mathbf{w} + \mathbf{w}^\mathsf{T}\mathbf{R}\mathbf{p} - \mathbf{w}^\mathsf{T}\mathbf{w} \\
&= [\mathbf{p}, \mathbf{w}]^\mathsf{T} \begin{bmatrix} \mathbf{V} & \mathbf{R}^\mathsf{T} \\ \mathbf{R} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \mathbf{w} \end{bmatrix}.
\end{aligned} \tag{2.58}$$

Notice that in the last two lines, we have introduced ℓ new variables $\mathbf{w} = \mathbf{R}\mathbf{p}$.

As in the general NLP problem, it is necessary that the QP subproblem be well-posed. Consequently, to correct a defective QP subproblem, the *residual Hessian* is modified:

$$\overline{\mathbf{V}} = \mathbf{V} + \tau(|\sigma| + 1)\mathbf{I}, \tag{2.59}$$

where σ is the Gerschgorin bound for the most negative eigenvalue of $\mathbf{V}$. Notice that it is not necessary to modify the entire Hessian matrix for the augmented problem but simply the portion that can contribute to directions of negative curvature. However, since the artificial variables $\mathbf{w}$ are introduced, the inertia test (2.41) must become

$$In(\mathbf{K}_0) = (n_f - \ell, m + \ell, 0) \tag{2.60}$$

to account for the artificial directions.

We then solve an augmented subproblem for the variables $\mathbf{q}^{\mathsf{T}} = [\mathbf{p},\mathbf{w}]^{\mathsf{T}}$, i.e., minimize

$$\frac{1}{2}[\mathbf{p},\mathbf{w}]^{\mathsf{T}}\begin{bmatrix}\overline{\mathbf{V}} & \mathbf{R}^{\mathsf{T}}\\ \mathbf{R} & -\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{p}\\ \mathbf{w}\end{bmatrix} + \mathbf{g}^{\mathsf{T}}\mathbf{p} \tag{2.61}$$

subject to

$$\mathbf{b}_L \le \begin{bmatrix}\mathbf{G} & \mathbf{0}\\ \mathbf{I} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{p}\\ \mathbf{w}\end{bmatrix} \le \mathbf{b}_U. \tag{2.62}$$

An alternative form for the augmented problem is suggested in [14] and [16] that requires the explicit introduction of the constraints $\mathbf{w} = \mathbf{R}\mathbf{p}$. However, the technique defined by (2.61) and (2.62) is more compact than the approach in [14] and still avoids formation of the normal matrix. As for the general NLP problem, we choose the Levenberg parameter, $0 \le \tau \le 1$, such that the projected Hessian of the augmented problem is positive definite. We modify the Levenberg parameter at each iteration such that ultimately $\tau \to 0$. Observe that for linear residuals and constraints, the residual Hessian $\mathbf{V} = 0$ and, consequently, $|\sigma| = 0$. Thus, in the linear case, no modification is necessary unless $\mathbf{R}$ is rank deficient. Furthermore, for linearly constrained problems with small residuals at the solution as $\|\mathbf{r}\| \to 0$, $|\sigma| \to 0$, so the modification to the Hessian is “small” even when the Levenberg parameter $\tau \ne 0$. Finally, for large residual problems, i.e., even when $\|\mathbf{r}^*\| \ne 0$, the accelerated trust-region strategy used for the general NLP method adjusts the modification $\tau \to 0$, which ultimately leads to quadratic convergence.

2.10.3 Residual Hessian

Because NLS problems require the residual Hessian $\mathbf{V}$, a number of special techniques have been developed specifically for this purpose. One approach is to construct a quasi-Newton approximation for $\mathbf{V}$ itself rather than the full Hessian $\mathbf{H}$. Inserting the definition of the least squares Hessian (2.56) into the secant equation (1.47) gives

$$\overline{\mathbf{B}}\Delta\mathbf{x} = (\overline{\mathbf{V}} + \mathbf{R}^{\mathsf{T}}\mathbf{R})\Delta\mathbf{x} = \Delta\mathbf{g}. \tag{2.63}$$

Rearranging this expression leads to

$$\overline{\mathbf{V}}\Delta\mathbf{x} = \Delta\mathbf{g} - \mathbf{R}^{\mathsf{T}}\mathbf{R}\Delta\mathbf{x} \equiv \Delta\mathbf{g}^{\#}. \tag{2.64}$$

The idea of using an SR1 update (1.49) to construct a quasi-Newton approximation to $\mathbf{V}$ was proposed in [10] and [11]. Dennis, Gay, and Welsch [70] suggest a similar technique in which the DFP update (1.51) is used and the quantity

$$\Delta\mathbf{g}^{\#} \equiv (\overline{\mathbf{R}} - \mathbf{R})^{\mathsf{T}}\overline{\mathbf{r}}. \tag{2.65}$$

Other variations have also been suggested and these methods are generally quite effective for dense least squares problems, especially when $\|\mathbf{r}^*\| \gg 0$.

A finite difference method can also be used to construct gradient information for NLS problems. Since both the residual Jacobian and the constraint Jacobian are needed, the quantities being differentiated in (2.1) are defined as

$$\mathbf{q} = \begin{bmatrix}\mathbf{c}\\ \mathbf{r}\end{bmatrix}, \tag{2.66}$$

and then it follows that

$$\mathbf{D} = \begin{bmatrix} \mathbf{G} \\ \mathbf{R} \end{bmatrix}. \tag{2.67}$$

It is also natural to define

$$\boldsymbol{\omega}^{\mathsf{T}} = (-\lambda_1, \dots, -\lambda_m, r_1, \dots, r_\ell), \tag{2.68}$$

where λ_k are the Lagrange multipliers with $\upsilon = m + \ell$, so that (2.2) becomes

$$\Omega(\mathbf{x}) = \sum_{i=1}^{\upsilon} \omega_i q_i(\mathbf{x}) = -\sum_{i=1}^{m} \lambda_i c_i(\mathbf{x}) + \sum_{i=1}^{\ell} [r_i] r_i(\mathbf{x}). \tag{2.69}$$

It should be recalled that elements of $\boldsymbol{\omega}$ are not perturbed during the finite difference operation. To emphasize this, we have written the second term above as $[r_i]r_i(\mathbf{x})$ since the quantities $[r_i]$ do *not* change during the perturbations. Then it follows that the residual Hessian $\mathbf{V} = \mathbf{E}$, where $\mathbf{E}$ is given by (2.3). This technique plays an important role in large-scale parameter estimation as discussed in Section 5.4.

2.11 Barrier Algorithm

2.11.1 External Format

For simplicity in presentation, in this section we temporarily drop the notational conventions used elsewhere in the book. The general NLP problem was defined in Section 1.12 and for convenience can be restated using slightly different notation as follows.

Determine the set of n variables to minimize the nonlinear function F

$$\mathsf{F}(\mathsf{x}) \tag{2.70}$$

subject to $\mathsf{m_C}$ nonlinear constraints

$$\underline{\mathsf{a}} \le \mathsf{a}(\mathsf{x}) \le \overline{\mathsf{a}} \tag{2.71}$$

and $\mathsf{m_B}$ simple variable bounds

$$\underline{\mathsf{x}} \le \mathsf{x} \le \overline{\mathsf{x}}. \tag{2.72}$$

This is referred to as the *external format* of the problem. Equality constraints or fixed variables are specified by setting the lower and upper limits to the same value. One-sided constraints or bounds can be specified as lower or upper bounds by using an "infinite" value for the unconstrained side of the inequalities. When solving a series of related problems it also may be convenient to ignore specified constraints or bounds. With this generality in mind, denote the set of indices of subscripts corresponding to nonignored or *included* equality constraints by $\mathcal{E}$ and the subset of constraints themselves by

$$\underline{\mathsf{a}}_{\mathcal{E}} = \mathsf{a}_{\mathcal{E}}(\mathsf{x}) = \overline{\mathsf{a}}_{\mathcal{E}}.$$

Denote the set of indices for included inequality constraints by $\mathcal{I}$ and the inequality constraints by

$$\underline{\mathsf{a}}_{\mathcal{I}} \le \mathsf{a}_{\mathcal{I}}(\mathsf{x}) \le \overline{\mathsf{a}}_{\mathcal{I}}.$$

Similarly, denote the subset of the $\mathsf{n_{FX}}$ simple bounds that are not ignored and that fix variables as

$$\underline{\mathsf{x}}_{\mathcal{F}} = \mathsf{x}_{\mathcal{F}} = \overline{\mathsf{x}}_{\mathcal{F}}$$

and the subset of included simple bounds on the $\mathsf{n_F}$ free variables as

$$\underline{\mathsf{x}}_{\mathcal{A}} \leq \mathsf{x}_{\mathcal{A}} \leq \overline{\mathsf{x}}_{\mathcal{A}}.$$

The relevant dimensions of the problem, as they may appear later, are

n	total number of variables
$\mathsf{n_{FX}}$	number of fixed variables
$\mathsf{n_F}$	number of free variables, also the number of included inequality simple bounds
$\mathsf{m_C}$	total number of nonlinear constraints
$\mathsf{m_E}$	number of included equality constraints
$\mathsf{m_I}$	number of included inequality constraints
$\mathsf{m_B}$	total number of simple bounds

Note that the dimensions as seen by the external format are always denoted in a sans serif font. In contrast, dimensions (and variables) of the problem transformed to internal form will be given in a conventional mathematics font.

2.11.2 Internal Format

The external problem format is converted to an internal form that is more convenient for algorithm description. The transformation to internal form includes the following steps.

- Eliminate all constraints marked as ignored or having infinite bounds on both sides. (Further transformations take place only on constraints that are really part of the problem.)
- Eliminate all fixed variables. Only the free variables $x = \mathsf{x}_{\mathcal{A}}$ appear explicitly.
- Introduce one slack variable for each nonlinear inequality constraint to transform the inequality constraint into an equality constraint. Thus the inequality constraint

  $$\underline{\mathsf{a}}_k \leq \mathsf{a}_k(\mathsf{x}) \leq \overline{\mathsf{a}}_k$$

  is replaced by a nonlinear equality constraint

  $$c_k(\mathbf{x}) \equiv \mathsf{a}_k(\mathsf{x}) - s_k = 0,$$

  and the simple bound

  $$\underline{\mathsf{a}}_k \leq s_k \leq \overline{\mathsf{a}}_k.$$

  There are $\mathsf{m_I}$ slack variables, which we denote by $\mathbf{s}$.
- Replace two-sided bounds by one-sided bounds. In doing so, eliminate one-sided bounds when the limit is $\pm\infty$. The naming convention for the bounds that are not eliminated is as follows. Denote by $\mathcal{B}_1$ the subset of indices corresponding to the

$\mathsf{n_F}$ inequality simple bounds when a finite lower bound is given. The similar subset when a finite upper bound appears is $\mathcal{B}_2$. The corresponding subsets of the bounds on slack variables (or of the nonlinear inequality constraints) are $\mathcal{B}_3$ and $\mathcal{B}_4$. Each slack variable appears in one or two one-sided simple bounds.

- Modify the form of the problem so that all bounds are simple positivity bounds.

The NLP transformed to internal format is as follows:

Minimize the function

$$F(\mathbf{y}) \equiv \mathsf{F}(\mathsf{x}) \tag{2.73}$$

of the $n = \mathsf{n_F} + \mathsf{m_I}$ variables

$$\mathbf{y} \equiv \begin{pmatrix} \mathbf{x} \\ \mathbf{s} \end{pmatrix} \tag{2.74}$$

subject to the $m_E = \mathsf{m_E} + \mathsf{m_I}$ nonlinear equality constraints

$$\mathbf{c}(\mathbf{y}) \equiv \begin{pmatrix} \mathsf{a}_{\mathcal{E}}(\mathsf{x}) - \underline{\mathsf{a}}_{\mathcal{E}} \\ \mathsf{a}_{\mathcal{I}}(\mathsf{x}) - \mathbf{s} \end{pmatrix} = \mathbf{0} \tag{2.75}$$

and the m_B linear bounds

$$\mathbf{b}(\mathbf{y}) \equiv \begin{pmatrix} (\mathbf{x} - \underline{\mathsf{x}}_{\mathcal{A}})_{\mathcal{B}_1} \\ (\overline{\mathsf{x}}_{\mathcal{A}} - \mathbf{x})_{\mathcal{B}_2} \\ (\mathbf{s} - \underline{\mathsf{a}}_{\mathcal{I}})_{\mathcal{B}_3} \\ (\overline{\mathsf{a}}_{\mathcal{I}} - \mathbf{s})_{\mathcal{B}_4} \end{pmatrix} \geq \mathbf{0}. \tag{2.76}$$

The relevant dimensions of the problem in the internal format are:

n	$\equiv \mathsf{n_F} + \mathsf{m_I}$	total number of internal variables
m_E	$\equiv \mathsf{m_E} + \mathsf{m_I}$	number of nonlinear equality constraints
m_B	$\equiv \lvert\mathcal{B}_1\rvert + \lvert\mathcal{B}_2\rvert + \lvert\mathcal{B}_3\rvert + \lvert\mathcal{B}_4\rvert$	number of simple positivity bounds

There are a number of features of the transformed problem (2.73)–(2.76) that deserve comment in light of the developments to follow. Observe that the equality constraints (2.75) may be nonlinear functions of the variables $\mathbf{y}$, whereas the inequality constraints (2.76) are strictly linear functions of $\mathbf{y}$. This property makes it straightforward to construct an initial guess $\mathbf{y}^{(0)}$ that is strictly feasible with respect to the inequalities $\mathbf{b}(\mathbf{y}^{(0)}) > \mathbf{0}$. Furthermore, the method to be described constructs iterates that maintain feasibility $\mathbf{b}(\mathbf{y}^{(k)}) > \mathbf{0}$ for all k. However, this also implies that the original (external format) inequalities (2.71) may not be strictly feasible until a solution is found. Consequently, let us refer to the approach as an *infeasible barrier method*. With the problem transformed to the internal format (2.73)–(2.76) let us now proceed to develop the details of the interior-point algorithm.

2.11.3 Definitions

The *Lagrangian* for the internal format NLP is defined as

$$L(\mathbf{y},\boldsymbol{\eta},\boldsymbol{\lambda}) = F(\mathbf{y}) - \boldsymbol{\eta}^{\mathsf{T}}\mathbf{c}(\mathbf{y}) - \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{b}(\mathbf{y}), \tag{2.77}$$

where $\boldsymbol{\eta}$ is the m_E-vector of Lagrange multipliers corresponding to the equality constraints, and $\boldsymbol{\lambda}$ is the m_B-vector of Lagrange multipliers corresponding to the inequality (bound) constraints. The gradient of the Lagrangian is given by

$$\nabla_y L(\mathbf{y},\boldsymbol{\eta},\boldsymbol{\lambda}) = \mathbf{g} - \mathbf{C}^{\mathsf{T}}\boldsymbol{\eta} - \mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda}, \tag{2.78}$$

where the gradient of the objective is just

$$\mathbf{g} = \nabla_y F(\mathbf{y}) = \begin{pmatrix} \nabla_x F(\mathbf{x}) \\ \mathbf{0} \end{pmatrix}, \tag{2.79}$$

and the $m_E \times n$ Jacobian of equalities is

$$\mathbf{C} \equiv \begin{pmatrix} \mathbf{A}_{\mathcal{E}} & \mathbf{0} \\ \mathbf{A}_{\mathcal{I}} & -\mathbf{I} \end{pmatrix}, \tag{2.80}$$

with the $m_B \times n$ Jacobian of the bounds given by

$$\mathbf{B} \equiv \begin{pmatrix} \mathbf{I}_{\mathcal{B}_1} & \mathbf{0} \\ -\mathbf{I}_{\mathcal{B}_2} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{\mathcal{B}_3} \\ \mathbf{0} & -\mathbf{I}_{\mathcal{B}_4} \end{pmatrix}. \tag{2.81}$$

The matrices $\mathbf{I}_{\mathcal{B}_2}$, $\mathbf{I}_{\mathcal{B}_4}$, $\mathbf{I}_{\mathcal{B}_1}$, and $\mathbf{I}_{\mathcal{B}_3}$ are rectangular submatrices of an identity matrix, each row having a single nonzero entry of 1.

The Hessian of the Lagrangian is given by

$$\nabla^2_{yy} L(\mathbf{y},\boldsymbol{\eta},\boldsymbol{\lambda}) = \nabla^2_{yy} F(\mathbf{y}) - \sum_{k=1}^{m_E} \eta_k \nabla^2_{yy} c_k(\mathbf{y}) - \sum_{k=1}^{m_B} \lambda_k \nabla^2_{yy} b_k(\mathbf{y}) \tag{2.82}$$

$$= \nabla^2_{yy} F(\mathbf{y}) - \sum_{k=1}^{m_E} \eta_k \nabla^2_{yy} c_k(\mathbf{y}) \tag{2.83}$$

$$= \begin{pmatrix} \mathbf{H}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{2.84}$$

where

$$\mathbf{H}_L \equiv \nabla^2_{xx} F(\mathbf{x}) - \sum_{k=1}^{m_E} \eta_k \nabla^2_{xx} c_k(\mathbf{x}). \tag{2.85}$$

When the projected Hessian is not positive definite consider a modified Hessian matrix that is defined as

$$\mathbf{W} = \begin{pmatrix} \mathbf{H} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{H}_L + \tau(|\sigma|+1)\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{2.86}$$

where $0 \le \tau \le 1$ is a Levenberg parameter and σ is the Gerschgorin bound for the most negative eigenvalue of $\mathbf{H}_L$.

Let us denote the solution to (2.73)–(2.76) by $(\mathbf{y}^*, \boldsymbol{\eta}^*, \boldsymbol{\lambda}^*)$. The solution is characterized by the following:

1. **Feasibility.** $\mathbf{c}(\mathbf{y}^*) = \mathbf{0}$ and $\mathbf{b}(\mathbf{y}^*) \ge \mathbf{0}$.

2. **Constraint Qualification.** The gradients of all constraints active at $\mathbf{y}^*$ are linearly independent.

3. **First Order KKT Condition.** The point $(\mathbf{y}^*, \boldsymbol{\eta}^*, \boldsymbol{\lambda}^*)$ is a stationary point of the Lagrangian, i.e., $\nabla_y L = \mathbf{0}$, which means

$$\mathbf{g} - \mathbf{C}^{\mathsf{T}}\boldsymbol{\eta} - \mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda} = \mathbf{0} \tag{2.87}$$

and

$$\lambda_k^* \ge 0, \qquad \lambda_k b_k = 0 \tag{2.88}$$

for $k = 1, \ldots, m_B$.

4. **Strict Complementarity.** $\lambda_k^* > 0$ if $b_k = 0$ for $k = 1, \ldots, m_B$.

5. **Second Order KKT Condition.** The projected Hessian of the Lagrangian $\mathbf{Z}^{\mathsf{T}}\mathbf{W}^*\mathbf{Z}$ is positive definite, where $\mathbf{Z}$ is a basis for the null space of the Jacobian of the constraints that are equal to zero at $\mathbf{y}^*$, and $\mathbf{W}^* \equiv \mathbf{W}(\mathbf{y}^*, \boldsymbol{\eta}^*, \boldsymbol{\lambda}^*)$.

2.11.4 Logarithmic Barrier Function

The interior-point approach is based on the definition of the *logarithmic barrier function*

$$\beta(\mathbf{y}, \mu) = F(\mathbf{y}) - \mu \sum_{k=1}^{m_B} \ln b_k(\mathbf{y}), \tag{2.89}$$

where μ is called the *barrier parameter*. Fiacco and McCormick [79] showed that the original constrained problem (2.73)–(2.76) could be replaced by a sequence of problems for successively smaller values of the barrier parameter μ. A classical *primal-barrier* approach would solve a sequence of equality constrained problems, namely minimize

$$\beta(\mathbf{y}, \mu) \tag{2.90}$$

subject to the equalities

$$\mathbf{c}(\mathbf{y}) = \mathbf{0}. \tag{2.91}$$

The necessary optimality conditions for the barrier subproblem (2.90)–(2.91) can be stated in terms of the *barrier Lagrangian*

$$L_\beta(\mathbf{y}, \boldsymbol{\eta}_\mu, \mu) = \beta(\mathbf{y}, \mu) - \boldsymbol{\eta}_\mu^{\mathsf{T}}\mathbf{c}(\mathbf{y}). \tag{2.92}$$

The gradient of the barrier Lagrangian is given by

$$\begin{aligned}\nabla_y L_\beta &= \nabla_y \beta - \mathbf{C}^\mathsf{T} \boldsymbol{\eta}_\mu &(2.93)\\ &= \mathbf{g} - \mu \mathbf{B}^\mathsf{T} \mathbf{D}_b^{-1} \mathbf{e} - \mathbf{C}^\mathsf{T} \boldsymbol{\eta}_\mu &(2.94)\\ &= \mathbf{g} - \mathbf{B}^\mathsf{T} \boldsymbol{\pi}_b - \mathbf{C}^\mathsf{T} \boldsymbol{\eta}_\mu, &(2.95)\end{aligned}$$

where $\mathbf{D}_b$ is a diagonal matrix

$$\mathbf{D}_b = \operatorname{Diag}(b_1, b_2, \dots, b_{m_B}), \tag{2.96}$$

and $\mathbf{e}$ is a vector of ones. Observe that the components of the vector

$$\boldsymbol{\pi}_b = \mu \mathbf{D}_b^{-1} \mathbf{e} \tag{2.97}$$

are just $(\pi_b)_k = \mu / b_k$.

The classical approach is to solve a sequence of equality constrained problems (2.90)–(2.91) for successively smaller barrier parameters μ. Ill-conditioning in the solution process is reduced by solving a sequence of problems rather than just beginning with a small value for μ as illustrated in the example (1.116). Let us denote the local minimizer by $\mathbf{y}_\mu$. Then it follows that the necessary conditions $\nabla_y L_\beta = \mathbf{0}$ and $\nabla_\eta L_\beta = \mathbf{0}$ are just

$$\begin{aligned}\mathbf{g} - \mathbf{B}^\mathsf{T} \boldsymbol{\pi}_b - \mathbf{C}^\mathsf{T} \boldsymbol{\eta}_\mu &= \mathbf{0}, &(2.98)\\ \mathbf{c}(\mathbf{y}_\mu) &= \mathbf{0}. &(2.99)\end{aligned}$$

Furthermore, it can be shown that

$$\begin{aligned}\lim_{\mu \to 0} \mathbf{y}_\mu &= \mathbf{y}^*, &(2.100)\\ \lim_{\mu \to 0} \boldsymbol{\eta}_\mu &= \boldsymbol{\eta}^*, &(2.101)\\ \lim_{\mu \to 0} \boldsymbol{\pi}_b &= \boldsymbol{\lambda}^*, &(2.102)\end{aligned}$$

where $(\mathbf{y}^*, \boldsymbol{\eta}^*, \boldsymbol{\lambda}^*)$ is the solution of the original inequality-constrained problem (2.73)–(2.76). This limiting behavior can be exploited to form a modified set of necessary conditions for the unknown quantities $(\mathbf{y}, \boldsymbol{\eta}, \boldsymbol{\lambda})$. In particular if we treat $(\mathbf{y}_\mu, \boldsymbol{\eta}_\mu, \boldsymbol{\pi}_b)$ as estimates for the true values $(\mathbf{y}^*, \boldsymbol{\eta}^*, \boldsymbol{\lambda}^*)$ and explicitly include the condition (2.102), we obtain the *modified primal-dual necessary conditions*

$$\boldsymbol{\Phi}_\mu \equiv \begin{pmatrix} \mathbf{g} - \mathbf{C}^\mathsf{T} \boldsymbol{\eta} - \mathbf{B}^\mathsf{T} \boldsymbol{\lambda} \\ \mathbf{c} \\ \mathbf{D}_b (\boldsymbol{\lambda} - \boldsymbol{\pi}_b) \end{pmatrix} = \mathbf{0}. \tag{2.103}$$

The conditions in the last row of (2.103) often written as

$$b_k(\mathbf{y}) \lambda_k - \mu = 0 \tag{2.104}$$

for $k = 1, \dots, m_B$ are referred to as the "centering" or "approximate complementarity" conditions, since as $\mu \to 0$, the centering condition (2.104) approaches the complementarity condition $b_k(\mathbf{y}) \lambda_k = 0$. Like other primal-dual methods (e.g., [173], [88], [87]), this condition is explicitly included here in contrast to a classical approach based solely on (2.98)–(2.99).

2.11.5 Computing a Search Direction

The equations (2.103) form a nonlinear system of equations $\mathbf{\Phi}_\mu = \mathbf{0}$ in the variables $(\mathbf{y}, \boldsymbol{\eta}, \boldsymbol{\lambda})$. A Taylor series expansion about the current point leads to the Newton equations for the modified primal-dual necessary conditions

$$\mathbf{W}\Delta\mathbf{y} - \mathbf{C}^\mathsf{T}\Delta\eta - \mathbf{B}^\mathsf{T}\Delta\lambda = -(\mathbf{g} - \mathbf{C}^\mathsf{T}\boldsymbol{\eta} - \mathbf{B}^\mathsf{T}\boldsymbol{\lambda}); \tag{2.105}$$

$$\nabla\mathbf{c}_k^\mathsf{T}(\mathbf{y})\Delta\mathbf{y} = -c_k(\mathbf{y}), \qquad k = 1,\dots,m_E; \tag{2.106}$$

$$\lambda_k \nabla\mathbf{b}_k^\mathsf{T}(\mathbf{y})\Delta\mathbf{y} + b_k(\mathbf{y})\Delta\lambda_k = -(b_k(\mathbf{y})\lambda_k - \mu), \qquad k = 1,\dots,m_B, \tag{2.107}$$

where $\mathbf{W}$ is the Hessian matrix defined by (2.86). Rewriting these equations yields the following *unsymmetric primal-dual KKT system*:

$$\begin{pmatrix} \mathbf{W} & \mathbf{C}^\mathsf{T} & \mathbf{B}^\mathsf{T} \\ \mathbf{C} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}_\lambda\mathbf{B} & \mathbf{0} & -\mathbf{D}_b \end{pmatrix} \begin{pmatrix} \Delta\mathbf{y} \\ -\Delta\eta \\ -\Delta\lambda \end{pmatrix} = - \begin{pmatrix} \mathbf{g} - \mathbf{C}^\mathsf{T}\boldsymbol{\eta} - \mathbf{B}^\mathsf{T}\boldsymbol{\lambda} \\ \mathbf{c} \\ \mathbf{D}_b\,(\boldsymbol{\lambda} - \boldsymbol{\pi}_b) \end{pmatrix}, \tag{2.108}$$

where $\mathbf{D}_\lambda = \operatorname{Diag}(\lambda_1, \lambda_2, \dots, \lambda_{m_B})$.

In principle the unsymmetric system (2.108) can be solved for the Newton step; however, for large, sparse linear systems it may be preferable to solve a KKT system that is scaled so that it is symmetric. Here consider using both row and column scaling. Let us apply scaling only to the third block row and the third block column of (2.108). Scale the rows of the coefficient matrix by the diagonal matrix $\mathbf{D}_r = \operatorname{Diag}(r_1, r_2, \dots, r_{m_B})$. Scale the columns (or the variables) by the diagonal matrix $\mathbf{D}_v = \operatorname{Diag}(v_1, v_2, \dots, v_{m_B})$. Thus application of the scaling matrices to (2.108) yields

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D}_r \end{pmatrix} \begin{pmatrix} \mathbf{W} & \mathbf{C}^\mathsf{T} & \mathbf{B}^\mathsf{T} \\ \mathbf{C} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}_\lambda\mathbf{B} & \mathbf{0} & -\mathbf{D}_b \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D}_v \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D}_v^{-1} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{y} \\ -\Delta\eta \\ -\Delta\lambda \end{pmatrix}$$
$$= - \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D}_r \end{pmatrix} \begin{pmatrix} \mathbf{g} - \mathbf{C}^\mathsf{T}\boldsymbol{\eta} - \mathbf{B}^\mathsf{T}\boldsymbol{\lambda} \\ \mathbf{c} \\ \mathbf{D}_b(\boldsymbol{\lambda} - \boldsymbol{\pi}_b) \end{pmatrix}. \tag{2.109}$$

Multiplying (2.109) through by the scaling matrices gives the *symmetric primal-dual KKT system*

$$\begin{pmatrix} \mathbf{W} & \mathbf{C}^\mathsf{T} & \mathbf{B}^\mathsf{T}\mathbf{D}_v \\ \mathbf{C} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}_r\mathbf{D}_\lambda\mathbf{B} & \mathbf{0} & -\mathbf{D}_r\mathbf{D}_b\mathbf{D}_v \end{pmatrix} \begin{pmatrix} \Delta\mathbf{y} \\ -\Delta\eta \\ -\mathbf{D}_v^{-1}\Delta\lambda \end{pmatrix} = - \begin{pmatrix} \mathbf{g} - \mathbf{C}^\mathsf{T}\boldsymbol{\eta} - \mathbf{B}^\mathsf{T}\boldsymbol{\lambda} \\ \mathbf{c} \\ \mathbf{D}_r\mathbf{D}_b(\boldsymbol{\lambda} - \boldsymbol{\pi}_b) \end{pmatrix}. \tag{2.110}$$

There are a number of possible ways to choose the matrices $\mathbf{D}_r$ and $\mathbf{D}_v$ such that the resulting KKT system is symmetric. In order to achieve symmetry of the KKT system we must have $\mathbf{D}_v = \mathbf{D}_r\mathbf{D}_\lambda$, implying that

$$r_k\lambda_k = v_k \tag{2.111}$$

for $k = 1,\dots,m_B$. Here consider three choices of diagonal entries for D_r and D_v that satisfy (2.111). One alternative is to choose

$$\begin{cases} r_k = \frac{1}{\lambda_k}, \\ v_k = 1, \end{cases} \tag{2.112}$$

which leads to

$$\mathbf{B}_k^{\mathsf{T}} \Delta \mathbf{y} - \frac{b_k(\mathbf{y})}{\lambda_k} (-\Delta\lambda_k) = -\frac{b_k(\mathbf{y})}{\lambda_k} (\boldsymbol{\lambda} - \boldsymbol{\pi}_b)_k \,. \tag{2.113}$$

Note that the lower-right diagonal block entry of the KKT matrix is $\frac{b_k(\mathbf{y})}{\lambda_k}$. Unfortunately, for inactive inequality constraints $\lambda_k \to 0$, leading to an entry that becomes infinite. A second alternative is

$$\begin{cases} r_k = 1, \\ v_k = \lambda_k, \end{cases} \tag{2.114}$$

which leads to

$$\lambda_k \mathbf{B}_k^{\mathsf{T}} \Delta \mathbf{y} - b_k(\mathbf{y}) \lambda_k \left(-\frac{\Delta\lambda_k}{\lambda_k} \right) = -b_k(\mathbf{y}) (\boldsymbol{\lambda} - \boldsymbol{\pi}_b)_k \,. \tag{2.115}$$

In this case, the lower-right diagonal block entry is $b_k(\mathbf{y})\lambda_k$ and the lower left block entry is $\lambda_k \mathbf{B}_k^{\mathsf{T}}$. This situation may also cause difficulties for inactive inequalities, since both of the above entries approach zero as $\mu \to 0$. Finally, consider

$$\begin{cases} r_k = \frac{1}{\sqrt{\lambda_k}}, \\ v_k = \sqrt{\lambda_k}, \end{cases} \tag{2.116}$$

which leads to

$$\sqrt{\lambda_k} \mathbf{B}_k^{\mathsf{T}} \Delta \mathbf{y} - b_k(\mathbf{y}) \left(-\frac{\Delta\lambda_k}{\sqrt{\lambda_k}} \right) = -\frac{b_k(\mathbf{y})}{\sqrt{\lambda_k}} (\boldsymbol{\lambda} - \boldsymbol{\pi}_b)_k \,. \tag{2.117}$$

In this case, the lower-right diagonal block entry is $b_k(\mathbf{y})$, which should be well behaved. It is for this reason that (2.116) is the preferred scaling technique.

Further simplification is possible by choosing the row and column scaling matrices $\mathbf{D}_v = \mathbf{D}_r = \mathbf{I}$. By eliminating $\Delta\lambda$ from (2.108) one obtains the *condensed primal-dual KKT system*

$$\begin{pmatrix} \mathbf{W} + \mathbf{B}^{\mathsf{T}} \mathbf{D}_b{}^{-1} \mathbf{D}_\lambda \mathbf{B} & \mathbf{C}^{\mathsf{T}} \\ \mathbf{C} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Delta \mathbf{y} \\ -\Delta \eta \end{pmatrix} = - \begin{pmatrix} \mathbf{g} - \mathbf{C}^{\mathsf{T}} \boldsymbol{\eta} - \mathbf{B}^{\mathsf{T}} \boldsymbol{\pi}_b \\ \mathbf{c} \end{pmatrix}, \tag{2.118}$$

where

$$\Delta\lambda = -\mathbf{D}_b{}^{-1} \mathbf{D}_\lambda \mathbf{B} \Delta \mathbf{y} - \boldsymbol{\lambda} + \boldsymbol{\pi}_b. \tag{2.119}$$

A Newton iteration requires solving either the linear system (2.110) or (2.118) for the search direction $(\Delta\mathbf{y}, \Delta\eta, \Delta\lambda)$. These sparse symmetric indefinite linear systems can be solved efficiently using the same multifrontal algorithm [4] described in Section 2.3. This technique also provides the *inertia* of the system (2.110) given as a triple defining the number of positive, negative, and zero eigenvalues, respectively. It can be shown that if

$$In(\mathbf{K}) = (n, m, 0), \tag{2.120}$$

where $\mathbf{K}$ is the matrix on the left-hand side of (2.110), then the projected Hessian matrix $\mathbf{Z}^{\mathsf{T}}\mathbf{W}\mathbf{Z}$ is positive definite as required by the necessary conditions. The Levenberg parameter τ in (2.86) can be adjusted to obtain the correct inertia using the procedure described in Section 2.5 and [17]. This creates a well-defined subproblem for the NLP. It is shown in the next section that the above inertia requirements on the barrier method KKT matrix are consistent with the optimality requirements for the original problem.

The search direction computed is a linear prediction for the nonlinear behavior, and as such may not satisfy the inequality constraints $\mathbf{b}(\mathbf{y}) \geq \mathbf{0}$ and $\boldsymbol{\lambda} \geq \mathbf{0}$. However, both of these conditions are linear functions of the variables, and consequently we can modify the *length* of the step such that these conditions are met. Specifically, using a linear estimate for the boundary, i.e., $\mathbf{b}(\overline{\mathbf{y}}) = \mathbf{0}$, the steplength is given by

$$\sigma_y = \min_k \left[\frac{-b_k(\mathbf{y})}{\nabla \mathbf{b}_k^\mathsf{T}(\mathbf{y}) \Delta \mathbf{y}} \right], \tag{2.121}$$

where the minimization is over all constraints in the downhill direction, that is, for all k such that $\mathbf{b}_k^\mathsf{T}(\mathbf{y})\Delta\mathbf{y} < 0$. In a similar fashion the length of the step in the multipliers must be changed so that $\boldsymbol{\lambda} \geq \mathbf{0}$. Again, making a linear estimate for the boundary at $\overline{\boldsymbol{\lambda}} = \mathbf{0}$ one finds

$$\sigma_\lambda = \min_k \left[\frac{-\lambda_k}{\Delta\lambda_k} \right] \tag{2.122}$$

for all k such that $\Delta\lambda_k < 0$. Now, a full length step as defined by the scalars σ_y and σ_λ corresponds to a move to the boundary of the feasible region. In order to maintain strict feasibility it is necessary to take some fraction of this step. Following the approach in Gay, Overton, and Wright [88] let us define a fraction of the full step according to

$$\varphi = 1 - \min(.01, 100\mu^2). \tag{2.123}$$

Thus, a "pad" has been introduced that ensures all subproblem iterates remain strictly feasible. Note that because the format of the problem has only *linear* inequalities this computation is particularly simple. In contrast, a formulation that permits nonlinear inequality constraints to appear directly in the barrier function would require a more complex line-search procedure to guarantee the nonlinear constraints remain feasible.

In keeping with most primal-dual methods [173], we attempt to use different steplengths in the primal variables $\mathbf{y}$ and the dual variables $\boldsymbol{\eta}$ and $\boldsymbol{\lambda}$. Specifically, consider a correction of the form

$$\begin{pmatrix} \overline{\mathbf{y}} \\ \overline{\boldsymbol{\eta}} \\ \overline{\boldsymbol{\lambda}} \end{pmatrix} = \begin{pmatrix} \mathbf{y} \\ \boldsymbol{\eta} \\ \boldsymbol{\lambda} \end{pmatrix} + \alpha \begin{pmatrix} \Delta\mathbf{y} \\ \gamma\Delta\eta \\ \gamma\Delta\lambda \end{pmatrix}. \tag{2.124}$$

Different primal and dual step scaling is defined by setting $\alpha = \hat{\alpha}$, where

$$\hat{\alpha} = \min(1, \varphi\sigma_y), \tag{2.125}$$

$$\gamma = \frac{1}{\hat{\alpha}} \min(1, \varphi\sigma_\lambda). \tag{2.126}$$

The primal and dual steps have the same scaling when

$$\alpha = \tilde{\alpha} = \min(1, \varphi\sigma_y, \varphi\sigma_\lambda), \tag{2.127}$$

with $\gamma = 1$, and in this case the step is just a multiple of the Newton step. Clearly it is desirable to take a Newton step when converging to a solution. Furthermore a scalar multiple of the Newton step must provide local improvement on the KKT conditions. On the other hand computational experience suggests using different primal-dual scaling may improve efficiency. Consequently we first try to use the scaled primal-dual step. If an acceptable point is obtained with $\alpha^{(0)} = \hat{\alpha}$ and $\gamma \neq 1$, it is accepted. If the point is not accepted, we force the same scaling by setting $\alpha^{(0)} = \tilde{\alpha}$ with $\gamma = 1$ and then choose subsequent steps so that $\alpha^{(k)} \leq \hat{\alpha}$ based on some globalization strategy.

2.11.6 Inertia Requirements for the Barrier KKT System

Using terminology from an active set method, one can show that the barrier algorithm asymptotically requires the Hessian positive definiteness properties indicated by optimality theory. The analysis also provides some insight into the workings of the interior-point method.

First, consider an active set method. When an active set SQP method solves a QP subproblem, inequalities go in and out of the active set (and the KKT matrix) until the correct active set, say

$$\widehat{\mathbf{A}} \equiv \begin{pmatrix} \mathbf{A}_{\mathcal{E}} \\ \widehat{\mathbf{A}}_{\mathcal{I}} \end{pmatrix}, \tag{2.128}$$

is identified. We know that optimality conditions require only that the projected Hessian $\mathbf{Z}^{\mathsf{T}}\mathbf{H}\mathbf{Z}$ be positive definite, where $\mathbf{Z}$ is a basis for the null space of $\hat{A}$, and $\mathbf{H}$ is the modified Hessian of the Lagrangian with respect to the original problem variables. Note that the Hessian may need to be modified as in [32] even when $\mathbf{Z}^{\mathsf{T}}\mathbf{H}\mathbf{Z}$ is positive definite. This is because the initial active set may result in a projected Hessian that extends outside $\mathbf{Z}^{\mathsf{T}}\mathbf{H}\mathbf{Z}$.

Now, consider an interior-point algorithm. The questions under consideration are the following:

- How does the KKT system change from one iteration to the next?
- Asymptotically, what ensures that the interior-point method does not require more than $\mathbf{Z}^{\mathsf{T}}\mathbf{H}\mathbf{Z}$ to be positive definite?

Note that the only part of the interior-point KKT matrix in (2.118) that changes during the iterations is the $\mathbf{B}^{\mathsf{T}}\mathbf{D}_b{}^{-1}\mathbf{D}_\lambda\mathbf{B}$ portion of the Hessian. The hope is that it changes in a way such that only $\mathbf{Z}^{\mathsf{T}}\mathbf{H}\mathbf{Z}$ need be positive definite near a solution. In (2.118), the matrix $\mathbf{M} \equiv \mathbf{W} + \mathbf{B}^{\mathsf{T}}\mathbf{D}_b{}^{-1}\mathbf{D}_\lambda\mathbf{B}$ becomes the Hessian of the barrier Lagrangian $\nabla^2_{yy}\beta = \mathbf{W} + \mu\mathbf{B}^{\mathsf{T}}\mathbf{D}_b{}^{-2}\mathbf{B}$ when $b_k(y)\lambda_k = \mu$, $k = 1,\ldots,m_B$. Also, the upper block of the right-hand-side vector in (2.118) becomes the gradient of the barrier Lagrangian (2.95) when $b_k(y)\lambda_k = \mu$, $k = 1,\ldots,m_B$.

Consider the positive definiteness requirements for $\mathbf{M}$ that are imposed by the interior-point method. The inertia requirements for the interior-point KKT system imply that $\mathbf{N}^{\mathsf{T}}\mathbf{M}\mathbf{N}$ is positive definite, where $\mathbf{N}$ is the null space of the $\mathbf{C}$ matrix in (2.118). For any

$$\mathbf{y} \equiv \begin{pmatrix} \mathbf{x} \\ \mathbf{s} \end{pmatrix},$$

we have

$$\mathbf{y}^{\mathsf{T}}(\mathbf{W} + \mathbf{B}^{\mathsf{T}}\mathbf{D}_b{}^{-1}\mathbf{D}_\lambda\mathbf{B})\mathbf{y} = \mathbf{x}^{\mathsf{T}}\mathbf{H}\mathbf{x} + \mathbf{s}^{\mathsf{T}}(\mathbf{B}^{\mathsf{T}}\mathbf{D}_b{}^{-1}\mathbf{D}_\lambda\mathbf{B})\mathbf{s}\,. \tag{2.129}$$

Suppose **y** is a vector in the null space **N**. Then, from the definition of **C** we have

$$\mathbf{s} = \mathbf{A}_{\mathcal{I}}\mathbf{x} \,. \tag{2.130}$$

Now, positive definiteness requirements for the interior-point method can be examined by considering the two types of vectors **y** that can be in the null space **N**. First, assume that **x** belongs to **Z**, the null space of $\hat{A}$. In this case, the optimality conditions ensure that $\mathbf{x}^{\mathsf{T}}\mathbf{H}\mathbf{x} > 0$. Since the entries in $\mathbf{D}_b$ and $\mathbf{D}_\lambda$ are positive, the second term in (2.129) is nonnegative. Thus, the right-hand side of (2.129) is positive and these null space vectors follow the optimality theory.

Next, suppose that $\mathbf{y} \in \mathbf{N}$ but **x** does not belong to **Z**. Since $\mathbf{y} \in \mathbf{N}$, we must have $\mathbf{A}_{\mathcal{E}}\mathbf{x} = \mathbf{0}$ and $\widehat{\mathbf{s}} = \widehat{\mathbf{A}}_{\mathcal{I}}\mathbf{x} \neq \mathbf{0}$. The definition of **B**, combined with $\widehat{\boldsymbol{\lambda}} > \mathbf{0}$, $\widehat{\mathbf{b}}(\mathbf{y}) \to \mathbf{0}$, and $\widehat{\mathbf{s}} \neq \mathbf{0}$, ensures that the second term in (2.129) will become more positive than any negative value for $\mathbf{x}^{\mathsf{T}}\mathbf{H}\mathbf{x}$. Thus, the Hessian positive definiteness requirements for the interior-point method asymptotically match those for the QP theory.

2.11.7 Filter Globalization

The role of a globalization strategy within the context of a line-search method is essentially to decide whether to accept or reject a particular step. Presumably, the point $(\overline{\mathbf{y}}, \overline{\boldsymbol{\eta}}, \overline{\boldsymbol{\lambda}})$ defined by (2.124) should be accepted if something "good" happens, and otherwise rejected. Many approaches have been proposed for the step acceptance criteria. One obvious technique is to monitor the error in the KKT conditions, i.e., the right-hand side of (2.108)

$$\|\boldsymbol{\Phi}_\mu\| \equiv \left\| \begin{matrix} \mathbf{g} - \mathbf{C}^{\mathsf{T}}\boldsymbol{\eta} - \mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda} \\ \mathbf{c} \\ \mathbf{D}_b(\boldsymbol{\lambda} - \boldsymbol{\pi}_b) \end{matrix} \right\|_\infty . \tag{2.131}$$

Unfortunately, it is well known that reducing $\|\boldsymbol{\Phi}_\mu\|$ does not necessarily correspond to finding a minimizer of $\beta(\mathbf{y}, \mu)$. Gay, Overton, and Wright [88] define an augmented Lagrangian merit function corresponding to the equality constrained barrier subproblem (2.90)–(2.91), and use a "watchdog" technique to force descent while monitoring $\|\boldsymbol{\Phi}_\mu\|$. Forsgren and Gill [87] propose an alternate merit function. However, both of these techniques require the computation of penalty weights, a task that can prove to be problematic. Instead let us consider use of a filter (cf. Section 1.11.4) as an alternative to constructing a merit function. We apply the nonlinear filter as a globalization strategy for the equality constrained barrier subproblem. In particular, for *fixed* μ we would like to do the following:

Minimize

$$\beta(\mathbf{y}, \mu) = F(\mathbf{y}) - \mu \sum_{k=1}^{m_B} \ln b_k(\mathbf{y}) \tag{2.132}$$

and minimize

$$\left\| \begin{matrix} \mathbf{c}(\mathbf{y}) \\ \mathbf{D}_b(\boldsymbol{\lambda} - \boldsymbol{\pi}_b) \end{matrix} \right\|_\infty . \tag{2.133}$$

Denote the values of the objective and constraint violation at the point $(\mathbf{y}^{(k)}, \boldsymbol{\lambda}^{(k)})$ by

$$\left\{\beta^{(k)}, v^{(k)}\right\} \equiv \left\{\beta(\mathbf{y}^{(k)}, \mu), \left\| \begin{matrix} \mathbf{c}(\mathbf{y}^{(k)}) \\ \mathbf{D}_b(\boldsymbol{\lambda}^{(k)} - \boldsymbol{\pi}_b) \end{matrix} \right\|_\infty \right\} . \tag{2.134}$$

When comparing the information at two different points $(\mathbf{y}^{(k)}, \boldsymbol{\lambda}^{(k)})$ and $(\mathbf{y}^{(j)}, \boldsymbol{\lambda}^{(j)})$, a pair $\left\{\beta^{(k)}, v^{(k)}\right\}$ is said to *dominate* another pair $\left\{\beta^{(j)}, v^{(j)}\right\}$ if and only if both $\beta^{(k)} \le \beta^{(j)}$ and $v^{(k)} \le v^{(j)}$. Using this definition we can then define a *filter* as a list of pairs

$$\mathcal{F} = \begin{bmatrix} \beta^{(1)}, v^{(1)} \\ \beta^{(2)}, v^{(2)} \\ \vdots \\ \beta^{(K)}, v^{(K)} \end{bmatrix} \tag{2.135}$$

such that no pair dominates any other. A new point $\left\{\beta^{(\ell)}, v^{(\ell)}\right\}$ is said to be acceptable for inclusion in the filter if it is not dominated by any point in the filter. Conversely, if a new point $\left\{\beta^{(\ell)}, v^{(\ell)}\right\}$ is dominated by any point in the filter, it is not acceptable. Thus, if a trial point produces an improvement in *either* the objective *or* the constraint violation over previous iterates, it is accepted. In keeping with [84], two special entries are included in the filter corresponding to the "northwest" and the "southeast" corners. For the northwest corner, the filter includes $\{\beta_{NW}, 0\}$, where β_{NW} is a liberal estimate for the upper bound on the objective function. At the southeast corner, an entry $\{-\infty, c_{max}\}$ is included, where c_{max} is an upper bound on the absolute constraint violation. An estimate for c_{max} must be specified by the user, and may be reduced if necessary to ensure convergence of the algorithm. It is important to note that the filter globalization strategy is used within a *single* barrier subproblem. When the barrier parameter is changed, the filter is restarted. Consequently, we are comparing only iterates with the same value for the barrier parameter.

We would like to choose the steplength $\alpha^{(k)}$ such that the new point given by (2.124) is "acceptable" to the filter. If the point is accepted, the information is used to augment the filter for subsequent iterations. If the point is not accepted, we utilize a special line-search strategy that forces step contraction, i.e., $\alpha^{(k)} < \alpha^{(k-1)}$, in order to find a point that is acceptable for inclusion in the filter. This step contraction is utilized only when the primal-dual step scaling is equal, i.e., $\gamma = 1$ in (2.124).

In contrast to a traditional line search, a filter has two different quantities that determine acceptance. Consequently, we compute two different estimates for the steplength—one chosen to reduce the objective, and one chosen to reduce the constraint error. Since $\nabla_y \beta = \mathbf{g} - \mathbf{B}^{\mathsf{T}} \boldsymbol{\pi}_b$, the slope of the barrier function evaluated at $\alpha = 0$ is

$$\beta_0' = (\Delta \mathbf{y})^{\mathsf{T}} \nabla_y \beta = \mathbf{g}^{\mathsf{T}} \Delta \mathbf{y} - (\mathbf{B} \Delta \mathbf{y})^{\mathsf{T}} \boldsymbol{\pi}_b. \tag{2.136}$$

Let us assume that the current iterate $\bar{\alpha} \equiv \alpha^{(k)}$ has been rejected by the filter. Three pieces of information are available: the function and slope at $\alpha = 0$ and the function value at $\bar{\alpha}$. For a standard merit function, it is common to estimate the new step as the minimizer of a polynomial model that interpolates the known information. However, Murray and Wright [136] suggest a model of the following form:

$$q(\alpha) = a + b\alpha + c\alpha^2 - \mu \ln(d - \alpha), \tag{2.137}$$

where the constant d defines the location of the known singularity in the objective. Clearly, one can choose $d = \sigma_y$ from (2.121) and then compute the remaining coefficients a, b, c so the quadratic log barrier model (2.137) interpolates the known data. Thus, by requiring

$q(0) = \beta(0) = \beta_0$, $q(\bar\alpha) = \beta(\bar\alpha) = \bar\beta$, and $q'(0) = \beta_0'$, one obtains

$$a = \beta_0 + \mu \ln(d); \tag{2.138}$$

$$b = \beta_0' - \mu/d; \tag{2.139}$$

$$c = \frac{1}{\bar\alpha^2}\left[\bar\beta - a - b\bar\alpha + \mu \ln(d - \bar\alpha)\right]. \tag{2.140}$$

From this information, the minimizer of the quadratic log barrier model (2.137) is

$$\alpha_q^* = \begin{cases} \frac{1}{4c}\left[2cd - b - \sqrt{(b+2cd)^2 + 8\mu c}\right] & \text{if } c \neq 0 \text{ and } \beta_0' < 0, \\ d + \frac{\mu}{b} & \text{if } c = 0 \text{ and } \beta_0' < 0, \\ 0 & \text{if } \beta_0' > 0. \end{cases} \tag{2.141}$$

As a second alternative, we would like to choose a steplength to reduce the constraint violation (2.133). For simplicity in constructing the steplength, let us model the violation by

$$\phi(\alpha) = \frac{1}{2}\left[\mathbf{c}^{\mathrm{T}}\mathbf{c} + (\mathbf{D}_b(\boldsymbol{\lambda} - \boldsymbol{\pi}_b))^{\mathrm{T}}(\mathbf{D}_b(\boldsymbol{\lambda} - \boldsymbol{\pi}_b))\right]. \tag{2.142}$$

The behavior by a quadratic model can be approximated by

$$q(\alpha) = a + b\alpha + c\alpha^2 \tag{2.143}$$

and the model coefficients can be computed to interpolate the function and slope at $\alpha = 0$ as well as the function value at $\bar\alpha$. After simplification one finds that the step that minimizes the constraint violation is

$$\alpha_v^* = \frac{\bar\alpha^2 \phi_0}{\left[\bar\phi - \phi_0 + 2\bar\alpha\phi_0\right]}. \tag{2.144}$$

As with any line search it is necessary to introduce some heuristics that safeguard the procedure. The basic goal is to choose the largest step when both models predict a step, use one of the models if possible, and resort to bisection otherwise. This philosophy can be implemented as follows:

$$\alpha^* = \begin{cases} \max\left[\alpha_v^*, \alpha_q^*, \kappa_2\bar\alpha\right] & \text{if } 0 < \alpha_v^* < \kappa_1\bar\alpha \text{ and } 0 < \alpha_q^* < \kappa_1\bar\alpha, \\ \max\left[\alpha_v^*, \kappa_2\bar\alpha\right] & \text{if } 0 < \alpha_v^* < \kappa_1\bar\alpha \text{ and } 0 = \alpha_q^* \text{ or } \alpha_q^* > \kappa_1\bar\alpha\ , \\ \max\left[\alpha_q^*, \kappa_2\bar\alpha\right] & \text{if } 0 < \alpha_q^* < \kappa_1\bar\alpha \text{ and } 0 = \alpha_v^* \text{ or } \alpha_v^* > \kappa_1\bar\alpha\ , \\ \max\left[\bar\alpha/2, \kappa_2\bar\alpha\right] & \text{otherwise.} \end{cases} \tag{2.145}$$

Typically, we choose the constants $\kappa_1 = .99$ and $\kappa_2 = \frac{1}{10}$. The computed value α^* from (2.145) provides an estimate for the steplength $\alpha^{(k)}$ in (2.124) which will construct a strictly contracting sequence until finding a point that is acceptable to the filter. Computational experience suggests that the line search seldom requires more than one or two steps.

It should be noted that the approximate complementarity equations (2.104) appear in the filter. The solution of (2.108) is guaranteed to have the properties of a solution to the barrier KKT system only if the approximate complementarity equations are satisfied. Thus, filter convergence theory [83] can be relied on only when (2.104) is satisfied.

2.11.8 Barrier Parameter Update Strategy

An important practical matter when implementing a barrier method is the choice for the barrier parameter μ. Clearly, μ must converge to zero, in order that $\mathbf{y}_\mu \to \mathbf{y}^*$. Typical early implementations of interior-point methods, as described in Fiacco and McCormick [79], calculate the solution to the barrier subproblem very accurately. Recent computational experience suggests that not only is this unnecessary, but it is computationally expensive. Instead, the preferred technique is to simply get "close" to the central path and then reduce the barrier parameter. Obviously, some quantitative definition of "close" is required, and we have adopted the approach in Gay, Overton, and Wright [88]. Specifically, we will consider a point "close" to the central path if

$$\|\boldsymbol{\Phi}_\mu\| < \min[\kappa\mu, \epsilon_c], \tag{2.146}$$

where (2.131) defines the error in the KKT conditions, and ϵ_c is a user-specified central path tolerance. Typically, $\epsilon_c = \kappa = 10$. We will use this relation to decide when the barrier parameter should be reduced.

Let us now describe the procedure for updating the barrier parameter estimate. The following test is applied after completing every step and is essentially a modified version of the procedure described in [88] for computing the new barrier parameter $\widehat{\mu}$:

if $(\alpha \geq .1)$ **and** $\|\boldsymbol{\Phi}_\mu\| < \min[\kappa\mu, \epsilon_c]$ **then**

$$\widehat{\mu} = \begin{cases} 10\mu^2 & \text{if } \mu < 10^{-4}, \\ \mu/10 & \text{if } \mu \geq 10^{-4}, \end{cases} \tag{2.147}$$

elseif μ unchanged for N_u iterations

$$\widehat{\mu} = .9\mu. \tag{2.148}$$

The philosophy of the procedure is to aggressively reduce the barrier parameter when it appears promising to do so, based on the observed behavior of the KKT error $\|\boldsymbol{\Phi}_\mu\|$. Conversely, if it appears that progress is slow and the barrier parameter is unchanged after N_u iterations, a modest reduction is made. Typically, we use $N_u = 10$ and $\epsilon_c = \kappa = 10$.

2.11.9 Initialization

A fundamental property of the barrier algorithm is that the sequence of iterates remain strictly feasible with respect to the bounds (2.76) and inequalities $\boldsymbol{\lambda} \geq \mathbf{0}$, as required by (2.88). In this section, we describe how to initialize the interior-point method. The primal variables $\mathbf{y}$ are simply reset to lie strictly within their bounds as a part of the transformation to internal format (2.74). However, in general, good values for the dual variables $(\boldsymbol{\eta}, \boldsymbol{\lambda})$ and the related barrier parameter μ are not available. There are a number of possibilities and we suggest three options.

Option 1. Suppose values for $\mathbf{y}$ and μ are known and the corresponding gradient information is also available. In this case, we can choose to use the central path estimate

$$\boldsymbol{\lambda} = \mu \mathbf{D}_b^{-1}\mathbf{e} \tag{2.149}$$

and then compute estimates $\widetilde{\boldsymbol{\eta}}$ for the multipliers $\boldsymbol{\eta}$ to minimize the error in the KKT conditions (2.87), i.e., minimize

$$\|\mathbf{g}-\mathbf{C}^{\mathsf{T}}\widetilde{\boldsymbol{\eta}}-\mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda}\| = \|\mathbf{C}^{\mathsf{T}}\widetilde{\boldsymbol{\eta}}-(\mathbf{g}-\mu\mathbf{B}^{\mathsf{T}}\mathbf{D}_b^{-1}\mathbf{e})\|. \tag{2.150}$$

This is an overdetermined linear least squares problem. Since $\mathbf{b} > \mathbf{0}$, it is clear from (2.149) that $\boldsymbol{\lambda} > \mathbf{0}$ (provided $\mu > 0$). On the other hand, we cannot expect that $\boldsymbol{\lambda}$ is a good approximation to the optimal multipliers $\boldsymbol{\lambda}^*$ unless $\mathbf{y}$ is "near" the central path.

Option 2. A second possible approach is to use the central path estimate (2.149) for $\boldsymbol{\lambda}$ and the gradient information at $\mathbf{y}$ and then compute estimates $\tilde{\mu}$ and $\widetilde{\boldsymbol{\eta}}$ to minimize

$$\|\mathbf{g}-\mathbf{C}^{\mathsf{T}}\widetilde{\boldsymbol{\eta}}-\mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda}\| = \left\|\begin{pmatrix}\mathbf{C}^{\mathsf{T}} & \mathbf{B}^{\mathsf{T}}\mathbf{D}_b^{-1}\mathbf{e}\end{pmatrix}\begin{pmatrix}\widetilde{\boldsymbol{\eta}}\\ \tilde{\mu}\end{pmatrix}-\mathbf{g}\right\|. \tag{2.151}$$

Unfortunately, the value of $\tilde{\mu}$ that solves this linear least squares problem may not be positive. So, if the computed estimate $\tilde{\mu} < \mu_L$, where $\mu_L = \max(\mu_\ell, \|c\|_\infty/\bar{\kappa})$ for a user-specified value μ_ℓ, we set $\mu = \mu_L$ and recompute using Option 1. The constant $\bar{\kappa}$ is chosen slightly larger than κ (say $\bar{\kappa} = 11$).

Option 3. The third possibility is to compute estimates $\tilde{\mu}$, $\widetilde{\boldsymbol{\eta}}$, and $\widetilde{\boldsymbol{\lambda}}$ to minimize

$$\left\|\begin{matrix}\mathbf{g}-\mathbf{C}^{\mathsf{T}}\widetilde{\boldsymbol{\eta}}-\mathbf{B}^{\mathsf{T}}\widetilde{\boldsymbol{\lambda}}\\ \mathbf{D}_b\widetilde{\boldsymbol{\lambda}}-\tilde{\mu}\mathbf{e}\end{matrix}\right\| = \left\|\begin{pmatrix}\mathbf{C}^{\mathsf{T}} & \mathbf{B}^{\mathsf{T}} & \mathbf{0}\\ \mathbf{0} & \mathbf{D}_b & -\mathbf{e}\end{pmatrix}\begin{pmatrix}\widetilde{\boldsymbol{\eta}}\\ \widetilde{\boldsymbol{\lambda}}\\ \tilde{\mu}\end{pmatrix}-\begin{pmatrix}\mathbf{g}\\ \mathbf{0}\end{pmatrix}\right\|. \tag{2.152}$$

As with Option 2, if the computed solution $\tilde{\mu} < \mu_L$, where $\mu_L = \max(\mu_\ell, \|c\|_\infty/\bar{\kappa})$, then we set $\mu = \mu_L$ and recompute using Option 1. Otherwise, we use $\tilde{\mu}$ and simply truncate the multipliers $\lambda_k = \max(\epsilon_m, \widetilde{\boldsymbol{\lambda}}_k)$, where ϵ_m is machine precision, to ensure they are strictly positive.

All three initialization options require the solution of a sparse linear least squares problem. The multifrontal algorithm [4] used for solving the KKT system can also be used to construct the minimum norm solution of the linear least squares problem

$$\|\mathbf{A}\mathbf{x}-\mathbf{b}\|. \tag{2.153}$$

Since the multifrontal method performs a symmetric factorization, if we assume $\mathbf{A}$ is full rank, then we can solve

$$\begin{pmatrix}\mathbf{I} & \mathbf{A}^{\mathsf{T}}\\ \mathbf{A} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ -\mathbf{r}\end{pmatrix} = \begin{pmatrix}\mathbf{0}\\ \mathbf{b}\end{pmatrix} \tag{2.154}$$

in the *underdetermined* case and

$$\begin{pmatrix}\mathbf{I} & \mathbf{A}\\ \mathbf{A}^{\mathsf{T}} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{r}\\ \mathbf{x}\end{pmatrix} = \begin{pmatrix}\mathbf{b}\\ \mathbf{0}\end{pmatrix} \tag{2.155}$$

when the system is *overdetermined*.

2.11.10 Outline of the Primary Algorithm

The preceding sections described the main features of the interior-point algorithm. Let us now outline the main algorithmic operations. The primary algorithm proceeds as follows:

1. Initialization: Compute external to internal transformation; (2.73)–(2.76), define sparsity of **B**, **C**, **W**, permutations, etc. Evaluate function and gradient information at the initial point.

2. Multiplier and Barrier Parameter Initialization: Compute initial values for $\boldsymbol{\eta}$, $\boldsymbol{\lambda}$, and μ using (2.150), (2.151), or (2.152).

3. Gradient Evaluation and Convergence Tests: Evaluate gradients and then check error in KKT conditions (2.131). Terminate if $\|\boldsymbol{\Phi}_\mu\| \le \epsilon$ and $\mu \le \epsilon$ for a tolerance ϵ (go to step 9).

4. Step Calculations:
   (a) Compute log-barrier function. Initialize filter if necessary.
   (b) Iteration print. Hessian evaluation.
   (c) Levenberg modification.
   (d) Barrier parameter update using (2.147) or (2.148). Restart filter when μ changes.
   (e) Multiplier reset. If $\|\boldsymbol{\eta}\|_\infty$ is too large, then recompute multipliers using (2.151), and, if this fails, using (2.150). If they are still too large, terminate (go to step 9).

5. Search Direction: Solve the KKT system (2.110) or (2.118) and (2.119) and modify the Hessian (2.86) by adjusting the Levenberg parameter if necessary.

6. Step Calculations: Compute the step scaling (2.121), (2.122), and (2.123).

7. Line Search:
   (a) Compute predicted point (2.124) and then evaluate functions and log-barrier function.
   (b) Check point against filter—if acceptable, then update filter (2.135) and go on; otherwise reduce α using (2.145) and repeat.

8. Update Information: Set $\mathbf{y} \leftarrow \overline{\mathbf{y}}$, $\boldsymbol{\eta} \leftarrow \overline{\boldsymbol{\eta}}$, and $\boldsymbol{\lambda} \leftarrow \overline{\boldsymbol{\lambda}}$, etc. Return to step 3.

9. Barrier Algorithm Termination.

2.11.11 Computational Experience

First, let us revisit the Powell example (1.116) introduced previously. Figure 2.2 illustrates the example for $m = 20$ with $\epsilon_c = 10^{-2}$. Notice that the central path is a vertical line in the x_1x_2-space. For the sake of illustration a number of points on the central path have been computed accurately, that is, by accurately solving the equality constrained subproblem, and the results are summarized in Table 2.3. Observe that as $\mu \to 0$ the computed values approach a solution at $(0, -1)$ more and more accurately.

The iteration history followed by the interior-point algorithm is presented in Table 2.4 and is shown in Figure 2.2. Note that 12 iterations were required to compute the solution. If the number of constraints in this simple example are increased, it can be used to illustrate another important aspect of interior-point methods. Table 2.5 presents a comparison of the

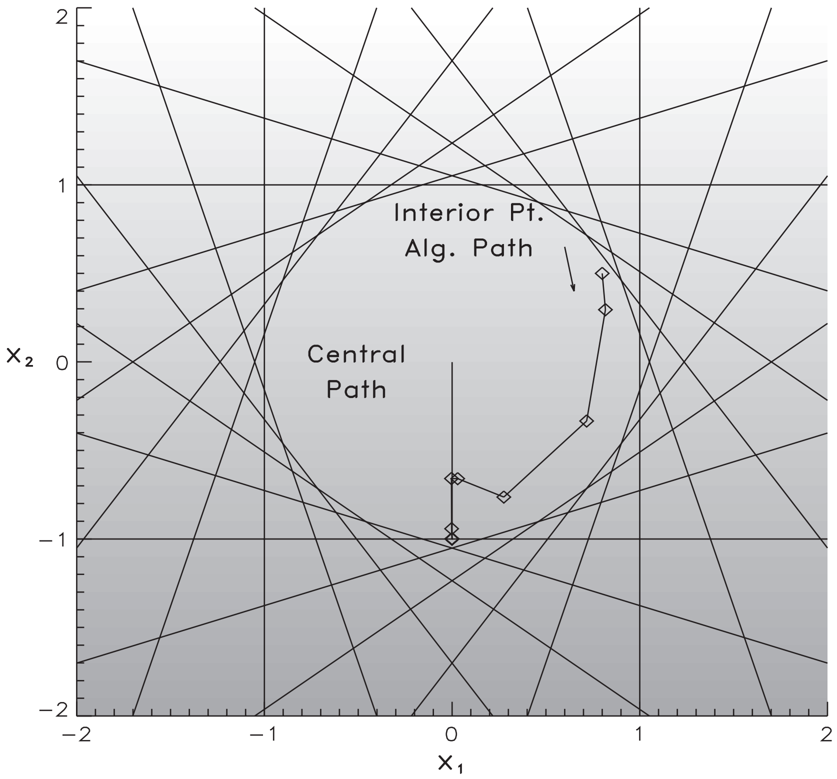

Figure 2.2. *Interior-point algorithm behavior.*

Table 2.3. *Central path.*

μ	x_1	x_2
1	$.443143 \times 10^{-16}$	$-.992604 \times 10^{-1}$
10^{-1}	$-.326292 \times 10^{-10}$	$-.658967$
10^{-2}	$-.687390 \times 10^{-9}$	$-.983966$
10^{-3}	$-.159620 \times 10^{-11}$	$-.998951$
10^{-4}	$.865020 \times 10^{-6}$	$-.999900$
10^{-5}	$.353902 \times 10^{-5}$	$-.999990$
10^{-6}	$.220346 \times 10^{-3}$	$-.999999$

interior-point method with the sparse SQP method presented in Section 2.6, which employs an active set strategy. In particular, we present the number of iterations needed to obtain a solution for three different sized problems. Observe that the number of iterations grows substantially for the active set method, whereas the barrier method computes the solution in nearly the same number of iterations.

Table 2.4. *Interior-point iteration history.*

x_1	x_2	$\|\mathbf{\Phi}_\mu\|$	μ
.800000	.500000	.441090	10^{-1}
.817167	.294690	.200266	10^{-1}
.717009	$-.333391$	.113042	10^{-1}
.275835	$-.763470$	9.703972×10^{-2}	10^{-1}
$.299099 \times 10^{-1}$	$-.659337$	5.747828×10^{-2}	10^{-1}
$-.350958 \times 10^{-2}$	$-.658608$	9.259092×10^{-2}	10^{-2}
$-.254854 \times 10^{-2}$	$-.942010$	2.195237×10^{-2}	10^{-2}
$-.496161 \times 10^{-3}$	$-.999420$	1.244132×10^{-2}	10^{-3}
$-.468141 \times 10^{-3}$	$-.998461$	1.379102×10^{-3}	10^{-4}
$-.406328 \times 10^{-3}$	$-.999949$	1.096671×10^{-4}	10^{-5}
$-.370598 \times 10^{-3}$	$-.999990$	1.011066×10^{-5}	10^{-9}
$-.370515 \times 10^{-3}$	-1.00000	1.925492×10^{-9}	10^{-9}

Table 2.5. *Iteration comparison.*

Constraints	Barrier Iterations	QP Iterations
20	12	11
200	12	65
2000	13	593

One final observation with regard to barrier methods is illustrated by this simple example. At the solution the limiting constraint (1.116) takes the form

$$x_1 \cos\left(\frac{\pi}{2}\right) + x_2 \sin\left(\frac{\pi}{2}\right) = x_2 \geq -1. \tag{2.156}$$

Observe from Figure 2.2 that when $x_2 = -1$, there are many points on the limiting constraint with the same value for the objective function. In other words, for this example, the optimal solution point is not unique, even though the optimal objective value is. An active set method such as the simplex algorithm would compute a solution with two active constraints, whereas the optimal solution computed by the barrier algorithm at $(-.370515 \times 10^{-3}, -1)$ has only one limiting constraint. Thus, the barrier method which does not utilize an active set has computed a solution on the optimal *facet*, whereas an active-set method would compute a solution that is at a *vertex*.