View metadata, citation and similar papers at core.ac.uk brought to you by CORE

provided by Research Papers in Economics

Scaling Sparse Constrained Nonlinear Problems for Iterative Solvers

Ravindra S. Gajulapalli
Leon S. Lasdon

W.P. No. 2006-08-06
August 2006

The main objective of the Working Paper series of IIMA is to help faculty members, research staff, and doctoral students to speedily share their research findings with professional colleagues and to test out their research findings at the pre-publication stage.

**INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD – 380015
INDIA**

SCALING SPARSE CONSTRAINED NONLINEAR PROBLEMS FOR ITERATIVE SOLVERS

Ravindra S. Gajulapalli
Leon S. Lasdon

Abstract

We[1] look at scaling a nonlinear optimization problem for iterative solvers that use at least first derivatives. These derivatives are either computed analytically or by differencing. We ignore iterative methods that are based on function evaluations only and that do not use any derivative information. We also exclude methods where the full problem structure is unknown like variants of delayed column generation.

We look at related work in section (1). Despite its importance as evidenced in widely used implementations of nonlinear programming algorithms, scaling has not received enough attention from a theoretical point of view. What do we mean by scaling a nonlinear problem itself is not very clear. In this paper we attempt a scaling framework definition. We start with a description of a nonlinear problem in section (2). Various authors prefer different forms, but all forms can be converted to the form we show. We then describe our scaling framework in section (3). We show the equivalence between the original problem and the scaled problem. The correctness results of section (3.3) play an important role in the dynamic scaling scheme suggested.

In section (4), we develop a prototypical algorithm that can be used to represent a variety of iterative solution methods. Using this we examine the impact of scaling in section (5). In the last section (6), we look at what the goal should be for an ideal scaling scheme and make some implementation suggestions for nonlinear solvers.

[1] Professor Leon Lasdon holds the David Bruton Jr. Chair in Business Decision Support Systems at the Department of Information, Risk, and Operations Management, the University of Texas at Austin. He is the author of several widely used NLP codes, is a co-developer of the Microsoft Excel Solver, and has published over 100 journal articles and three books.

Contents

1 Related Work

We quote Gill, Murray and Wright [GMW81, page 273]:

> The term “scaling” is invariably used in a vague sense to discuss numerical difficulties whose existence is universally acknowledged, but cannot be described precisely in general terms. Therefore, it is not surprising that much confusion exists about scaling, and that authors tend to avoid all but its most elementary aspects.

While discussing scaling linear systems, at least broad approaches and some concordance on issues is seen in published literature. For nonlinear optimization, we could not find definitive articles similar to Curtis and Reid [CR72] for Gaussian elimination. Almost two decades since Gill et al. made their observation, we find that they still seem to be on target with their remark. Even recent textbooks do not mention it [NS95], others dismiss it cursorily [Fle87, pages 59 and 148]. A recent second edition, in 2003, of Luenberger [Lue84] has the same coverage on this topic as the first one. Searching for articles on this topic is of little avail.

Luenberger [Lue84, pages 222 – 225] discusses how the convergence rates of steepest descent method applied to unconstrained optimization problem is highly sensitive to eigenvalues of the Hessian of the objective function. In this context he talks about scaling the variables using a generic linear transformation of the type $Ty = x$ where T as an invertible matrix. He talks about scale invariance of some unconstrained optimization methods like the Newton’s method. This is true in theory with exact arithmetic but in practice, the invariance property may not hold. We talk about scale invariance in more detail while discussing the scaling framework. Luenberger [Lue84, pages 402 – 403] briefly mentions that scaling variables does not affect the dual canonical convergence rate while changing the primal rate. On the other hand, scaling the constraints impacts dual rate with no change in the primal. The dual being discussed is based on local duality theorem [Lue84, page 399].

As mentioned earlier, good coverage to scaling in the context of constrained nonlinear optimization is provided by Gill, Murray and Wright [GMW81]. They motivate the need to transform variables by using a heat exchanger example [GMW81, pages 273 – 274] where the typical values range from $11,000$ to 5.4×10^{-10}. They also suggest that simple diagonal scaling alone may not be enough in some cases. They recommend an offset scheme that they acknowledge will fail if the modeler provides too wide a range. We incorporate the offset suggestion but with an additional safeguard. In a large model, their scheme based on a modeler specifying acceptable ranges for variables is impractical. They look at some specific cases like nonlinear least squares problems on [GMW81, page 275]. They discuss [GMW81, pages 325–327] how poor function scaling may result in insufficient decrease in the merit function. Schemes to scale variables, objective function and linear constraints are developed in [GMW81, pages 346 – 354]. They make some suggestions for nonlinear constraints based on the Jacobian matrix of the constraint functions in the working set. In our unified approach, we look at a dynamic scaling of constraints and objective function and do not make these distinctions.

In most of the iterative solvers we consider while developing our framework, a linear system is solved in each iteration. We presume that some reasonably efficient scheme like those suggested in [GL06] are used in solving these linear systems. Hence in this article we ignore that aspect of scaling dependence.

In practice, we have seen source code, user documentation etc associated with a variety of nonlinear programming solvers to see that scaling is indeed important. For instance, established reduced gradient methods like CONOPT2, LSGRG2 as well as augmented Lagrangian methods like MINOS do use scaling or provide options for it. Among all the robust general purpose solvers, CONOPT2 is probably the most rigorous about scaling and preprocessing. In INTOPT [GL01], we did not use scaling for the nonlinear optimization problem itself.

2 Nonlinear Optimization Problem

In this section, we define a nonlinear optimization problem and define the terminology for derivatives etc. We look at associated dual variables and the associated Lagrangian saddle point conditions.

2.1 Problem Statement and Derivatives

Without loss of generality, a constrained nonlinear problem, Γ, can be expressed as:

$$\begin{array}{rcl} \min & & f(x), \\ \text{subject to} & & \\ h(x) & = & b, \\ x & \leq & u, \\ x & \geq & l, \end{array} \tag{1}$$

where x is an n-dimensional vector, i.e., $x \in \Re^n$. The scalar objective function, f is being minimized and $f : \Re^n \mapsto \Re$. The function h represents m nonlinear equations, $h : \Re^n \mapsto \Re^m$. The constraint right hand sides are represented by a constant vector b where $b \in \Re^m$. Any inequality constrained problem can be transformed into this form by adding extra variables to denote slack or surplus.

The only inequalities present in the nonlinear model are explicit bounds on the variables. If a variable does not have a bound, its corresponding entry in vectors l or u may be $-\infty$ or $+\infty$ as the case may be. Without loss of generality, we assume that the bounds on all variables are finite. This can be accomplished by using a very large positive number like 10^{35} for ∞, and its negative counter part for $-\infty$ on cases where the variables have no finite bounds. The following assumption ensures that variable bounds in the given problem are consistent.

Assumption 2.1 (Consistent variable bounds) *We assume that in the nonlinear problem, Γ, $l < u$, that is the lower bounds are strictly lower than the upper bounds.*

Justification for Consistent variable bounds: We assume that some rudimentary preprocessing prevents $l_j > u_j$ for some variable j being posed to the nonlinear solver. There is no feasible solution in this case. In addition, we are ignoring the cases where $l_j = u_j$ because this implies that the variable remains fixed through out the solution process and hence can be ignored since the gradients etc. will always be zero with respect to such a variable. □

In addition, we make the following continuity assumption about the problem functions:

Assumption 2.2 (Continuity of functions) *All functions used in problem, Γ, are twice differentiable, i.e., $f, h \in \mathcal{C}^2$.*

Justification for Continuity of functions: If a function is not continuous then the solution method may not converge from certain starting points. □

The following assumption highlights constraint violations allowed during a solution process.

Assumption 2.3 (Variables within bounds) *Given a starting point x_0 for the nonlinear problem, Γ, such that $l \leq x_0 \leq u$, at all iterations, a solution method will respect the variable bounds.*

Justification for Variables within bounds: A nonlinear optimization method may ensure that all constraints are satisfied at all iterations like reduced gradient methods or it may allow constraint violations like penalty and barrier methods for the intermediate iterations. But in the following discussion, we assume that the explicit bounds are always respected by the algorithm in question. Not making this assumption, may lead to scenarios where some problem functions like $\sqrt{x - a}$ may be undefined outside the given range. □

For scaling functions, we consider the objective function also as a candidate for scaling. Hence for notational simplicity, let g denote the combination of objective function, f and m constraint functions h. Let $g_i(x) = h_i(x)$ for the m constraints and let $g_{m+1}(x) = f(x)$, that is the last function is the objective function. In other words, $g : \mathfrak{R}^n \mapsto \mathfrak{R}^{m+1}$. We let the matrix $J(x)$ denote the Jacobian matrix of the nonlinear problem evaluated at point x. We will usually suppress the argument to reduce notational clutter if there is no ambiguity. The matrix of first partials,

$$J(x) : \mathfrak{R}^n \mapsto \mathfrak{R}^{(m+1)\times n},$$

has $m+1$ rows - one for each function including the last row for objective function and n columns - one for each variable. An entry in row i and column j of this matrix signifies:

$$J_{i,j}(x) = \frac{\partial g_i}{\partial x_j} \qquad \forall i = 1, \ldots, m+1,\ j = 1 \ldots n.$$

We make an assumption about the derivative information used by an iterative solver.

Assumption 2.4 (Derivative information usage) *We assume that an iterative solver for the nonlinear problem, Γ, will* use at least the first partials *represented by the Jacobian matrix.*

Justification of Derivative information usage: Without using derivative information, termination criterion based on first order necessary conditions cannot be used. In practice, most algorithms also use it to determine descent directions, step size determination, scaling etc. □

Optionally, some solvers like those based on barrier methods may either use the Lagrangian Hessian based on second partials or an approximation of that. Let $H(x)$ define a rank three tensor of the second partials, i.e.,

$$H(x) : \mathfrak{R}^n \mapsto \mathfrak{R}^{(m+1)\times n\times n},$$

Hence the Hessian of the function g_i is an $n \times n$ symmetric matrix $H^i(x)$ of second partials for function g_i. An Hessian element corresponding to variables j and k for function i is given by:

$$H^i_{j,k}(x) = \frac{\partial^2 g_i}{\partial x_j \partial x_k} \qquad \forall i = 1, \ldots, m+1,\ j = 1 \ldots n,\ k = 1 \ldots n.$$

2.2 Dual Multipliers

In this section we look at the Lagrangian and associated dual variables. First order and second order conditions for optimality are stated and they will be used to find a corresponding solution in a scaled problem in section (3.3).

Let y_i be the dual multiplier associated with constraint i in the nonlinear problem, Γ, of equation (1). We call these multipliers *constraint duals*. To simplify the notation, with the objective function f, we associate a dual multiplier y_{m+1} that is always set to unity. Thus vector of constraint duals has $m+1$ components corresponding to each function in $g(x)$. We associate dual multipliers w_j and z_j with the upper bound and lower bound constraints respectively on variable x_j. We call these multipliers *upper* and *lower bound duals* respectively and collectively refer to them as *bound duals*. The Lagrangian associated with the nonlinear problem in equation (1) can be written as:

$$\mathcal{L}(x; y, w, z) = f(x) + \sum_{i=1}^{m} y_i h_i(x) - \sum_{j=1}^{n} w_j \left(u_j - x_j\right) - \sum_{j=1}^{n} z_j \left(x_j - l_j\right).$$

Using $y_{m+1} = 1$ as the multiplier for the objective function, we can write this as:

$$\mathcal{L}(x; y, w, z) = y^t g(x) - w^t(u - x) - z^t(x - l). \tag{2}$$

In forming the Lagrangian, we took the signs of the variable bound constraints to ensure that the bound duals remain positive in sign. The first order necessary conditions of optimality [Lue84, page 314] at a regular point x^* are given by:

$$\begin{aligned} \nabla f(x^*) + \sum_{i=1}^{m} y_i^* \nabla h_i(x^*) + w^* - z^* &= \mathbf{0}, \\ w_j^*(u_j - x_j^*) &= 0 \qquad \forall\, j = 1, \ldots, n, \\ z_j^*(x_j^* - l_j) &= 0 \qquad \forall\, j = 1, \ldots, n. \end{aligned}$$

We will abbreviate Karush-Kuhn-Tucker conditions as KKT conditions. As mentioned earlier, using $y_{m+1}^* = 1$, and diagonal matrices W and Z, we can write the above conditions as:

$$\begin{aligned} J(x^*)^t y^* + w^* - z^* &= \mathbf{0}, \\ W(u - x^*) &= \mathbf{0}, \\ Z(x^* - l) &= \mathbf{0}, \\ w^* &\geq \mathbf{0}, \\ z^* &\geq \mathbf{0}. \end{aligned} \tag{3}$$

The $n \times n$ diagonal matrices, W and Z are defined using the bound duals w and z respectively, namely,

$$\begin{aligned} W &= \text{Diag } \{w\} = \text{Diag } \{w_1, w_2, \ldots, w_n\} \text{ and} \\ Z &= \text{Diag } \{z\} = \text{Diag } \{z_1, z_2, \ldots, z_n\}. \end{aligned}$$

The KKT conditions associated with the bound duals are also called *complementary slackness* conditions. If a variable x_j is not at either of its bound, then both the bound duals are zero since the slacks in the bound constraints are non-zero. Let the set L and U denote the set of active upper and lower bound conditions respectively, i.e.,

$$\begin{aligned} U &= \left\{j : x_j^* = u_j,\ w_j^* \geq 0\right\} \\ L &= \left\{j : x_j^* = l_j,\ z_j^* \geq 0\right\}. \end{aligned}$$

Then the tangent subspace of the active conditions can be stated as:

$$M = \{d : \nabla h(x^*)d = \mathbf{0},\ d_j = 0\ \forall j \in U \cup L\}.$$

The second order necessary conditions for a regular point x^* to be a relative minimum point for the problem, Γ, in equation (1) is that the Lagrangian Hessian, $L(x^*)$ be *positive semi-definite* on the tangent subspace of the active constraints. The second order sufficiency conditions [Lue84, page 316] are met if the Lagrangian Hessian matrix is *positive definite* on the following subspace:

$$M' = \{d : \nabla h(x^*)d = \mathbf{0},\ d_j = 0\ \forall j \in (U \text{ and } w_j > 0) \cup (L \text{ and } z_j > 0)\}.$$

The subspace $M' \supseteq M$ and it excludes degenerate bound conditions where a variable is at a bound but the associated dual is also zero. The Lagrangian Hessian is an $n \times n$ symmetric matrix of second partials of the Lagrangian and in terms of the rank three tensor $H(x)$, it is defined by:

$$L(x) = \nabla_{xx}^2 \mathcal{L}(x, y, w, z) = \nabla^2 f(x) + \sum_{i=1}^{m} y_i \nabla^2 h_i(x) = \sum_{i=1}^{m+1} y_i H^i(x) = y^t H(x). \quad (4)$$

Thus the Lagrangian Hessian is a product of a vector of $m + 1$ components and a rank three tensor.

3 Scaling Framework

We address what we mean by scaling in section (3.1) and develop a scaled problem in section (3.2). We neither discuss *how to obtain the scale factors* themselves nor *when to do the scaling*. These issues are delved in section (6). We show that the scaled problem is equivalent to the original problem in section (3.3). This result is important in guiding the dynamic scaling procedure that we suggest later on.

3.1 Scaling Scheme and Scaled Derivatives

Unlike generic linear transformations suggested in [Lue84], we consider only a limited set of transformations on *variables and functions* though some of the results we state are applicable to generic linear transformations based on *invertible matrices*. Examples where nonlinear transformations are more beneficial can be easily constructed, see [GMW81, page 279] for an example, but we restrict ourselves to linear transformations.

For variables, the transformations can be interpreted as changing units. For instance, instead of measuring in meters, we measure in kilometers. The idea being that if all variables are roughly similar in magnitude, we get a more stable numerical behavior like in the case of matrices used in linear systems. More specifically a variable, x_j, is scaled as:

$$\overline{x}_j = \frac{x_j}{v_j} - a_j \qquad \forall\, j = 1, \dots, n,$$

where v_j is strictly positive scale factor, a_j is an offset and $\overline{x}_j$ is the new transformed variable. In general, we use the notation $\overline{t}$ to denote a new entity that has been generated by scaling entity t. If V denotes an $n \times n$ diagonal matrix, whose diagonal element $V_{jj} = v_j$, then the variable transformation can be expressed as:

$$\overline{x} = \mathcal{T}(x) = V^{-1}x - a. \tag{5}$$

Since the matrix is a nonsingular diagonal matrix with strictly positive quantities, it is also positive definite. Hence the transformation $\mathcal{T}(x)$ is a strict one-to-one mapping and can be reversed.

Similar to variables, we hope that scaling functions results in all functions and their derivatives being “well” balanced. Thus $g_i : \Re^n \mapsto \Re$ is equivalently transformed to $\overline{g}_i$ as shown below:

$$\overline{g}_i(\overline{x}) = r_i\, g_i(\mathcal{T}^{-1}(\overline{x})) \qquad \forall\, i = 1, \dots, m+1,$$

where r_i denotes function scale factor. Note, we are including the objective function also in the same scaling scheme. If an $(m+1) \times (m+1)$ diagonal matrix R is used to represent the row scale factors, i.e., $R_{ii} = r_i$, then we can express function transformations, $\mathcal{G}(y) = Ry$, as:

$$\overline{g}(\overline{x}) = \mathcal{G}(g(\mathcal{T}^{-1}(\overline{x}))) = Rg(\mathcal{T}^{-1}(\overline{x})). \tag{6}$$

Let $\overline{J}(\overline{x})$ be the matrix of first partials of the transformed functions with respect to the scaled variables. Given scaled variables $\overline{x}$, we can use the inverse of the transformation in equation (5) to get unscaled values, i.e., $x = \mathcal{T}^{-1}(\overline{x}) = V(\overline{x} + a)$. Then an entry, $\overline{J}_{i,j}$, for function i with respect to variable j, is given by:

$$\overline{J}_{i,j}(\overline{x}) = \frac{\partial \overline{g}_i(\overline{x})}{\partial \overline{x}_j} = r_i\, \frac{\partial g_i(\mathcal{T}^{-1}(\overline{x}))}{\partial x_j}\, v_j \qquad \forall i = 1, \dots, m+1,\ j = 1 \dots n.$$

In matrix notation, we can write the new Jacobian $\overline{J}(\overline{x})$ in terms of the original Jacobian evaluated at $x = \mathcal{T}^{-1}(\overline{x})$ as:

$$\overline{J}(\overline{x}) = R\; J(x)\; V. \tag{7}$$

Let $\overline{H}^i(\overline{x})$ denote the transformed Hessian of the function g_i. Using the transformations, an Hessian element corresponding to variables j and k gets scaled as:

$$\overline{H}^i_{j,k}(\overline{x}) = \frac{\partial^2 \overline{g}_i(\overline{x})}{\partial \overline{x}_j \partial \overline{x}_k} = r_i \, \frac{\partial^2 g_i(\mathcal{T}^{-1}(\overline{x}))}{\partial x_j \partial x_k} \, v_j \, v_k \qquad \forall i = 1, \ldots, m+1, \; j, k \; = 1 \ldots n.$$

Similar to the Jacobian matrix, a scaled function i's Hessian matrix $\overline{H}^i(\overline{x})$, can be expressed in terms of original Hessian evaluated at $x = \mathcal{T}^{-1}(\overline{x})$ as:

$$\overline{H}^i(\overline{x}) = r_i \, V^t \, H^i(x) \, V \qquad \forall i = 1, \ldots, m+1. \tag{8}$$

Since by assumption (2.2), the functions f, h were twice differentiable, the scaled functions are too. This is seen in expressions for first and second order partials in equations (7) and (8). In other words, $\overline{f}, \overline{h} \in \mathcal{C}^2$.

3.2 Scaled Optimization Problem

To generate a scaled nonlinear problem from the original problem, Γ, in equation (1), we scale the functions and right hand sides using the row transformations $\mathcal{G}$. Since the objective function is g_{m+1} and it has no right hand side, we can ignore the last row when applying row transformations to generate the equality constraints' right hand sides. In other words, the right hand sides are given by

$$\overline{b}_i = r_i b_i \qquad \forall \, i = 1, \ldots, m.$$

For the variables and bounds we use column transformations $\mathcal{T}$. In this case, the modified upper and lower bounds are given by:

$$\begin{aligned} \overline{u} &= \mathcal{T}(u) = V^{-1}u - a \qquad \text{and} \\ \overline{l} &= \mathcal{T}(l) = V^{-1}l - a. \end{aligned}$$

Using these changes, the scaled nonlinear problem, $\Gamma\langle V, R\rangle$, can be stated as:

$$\begin{array}{rcl} \min & & \overline{f}(\overline{x}), \\ \text{subject to} & & \\ \overline{h}(\overline{x}) & = & \overline{b}, \\ \overline{x} & \leq & \overline{u}, \\ \overline{x} & \geq & \overline{l}, \end{array} \tag{9}$$

To emphasize the dependence of the scaled problem on variable and function scale factors, we use notation $\Gamma\langle V, R\rangle$ to denote the scaled problem. Can domain violations occur for functions like $\sqrt{x - c}$ when a solution method is applied to the scaled problem, $\Gamma\langle V, R\rangle$, of equation (9) instead of the original problem, Γ? In the following lemma we state the conclusion that follows from our choice of linear transformations.

Lemma 3.1 (Variables within bounds) *Let a starting point $\overline{x}_0$ for the scaled problem, $\Gamma\langle V, R\rangle$, be such that $\overline{l} \leq \overline{x}_0 \leq \overline{u}$. If a solution method that follows assumption (2.3) is applied to the scaled problem, then all function evaluations will be inside original variable ranges* $l \leq x \leq u$ as well.

Proof of Variables within bounds: Since the method is being applied to scaled problem, by assumption (2.3), at all iterations the condition $\overline{l} \leq \overline{x} \leq \overline{u}$ will hold. This implies the condition $V(\overline{l} + a) \leq V(\overline{x} + a) \leq V(\overline{u} + a)$ is invariant through out the solution process. In other words, since $x = \mathcal{T}^{-1}(\overline{x}) = V(\overline{x} + a)$ we get $l \leq \mathcal{T}^{-1}(\overline{x}) \leq u$ and there is no risk of violating explicit variable bounds. □

The "correctness" of solving the scaled problem is addressed in section (3.3) . All we have stated in lemma (3.1) is that if a solution method can be applied *safely*, i.e., without domain violations, to the original problem then it can be applied safely to the scaled problem too.

3.3 Correctness

We say the scaled problem, $\Gamma\langle V, R\rangle$, of equation (9) is equivalent to the original nonlinear problem, Γ, of equation (1) if solving one provides a solution to the other. Thus we define correctness in terms of equivalence relationships. Solving a scaled problem is a *correct* procedure if the solutions are equivalent. The results in this section provide a foundation for the dynamic scaling scheme suggested in section (6). They provide mechanisms:

- To convert a scaled solution to a solution of the original problem.
- To adjust for changes in scaling scheme.

As we have been using earlier, through out this section, a notation $\overline{t}$ means a value from the scaled problem, $\Gamma\langle V, R\rangle$, while a plain t denotes *corresponding* value from the original problem, Γ, for any entity of interest like variables, duals etc.

Sometimes, when we talk about adjustments for scale changes, we need to introduce a new scaled problem and we denote values from the new scaled problems as $\tilde{t}$. Let $\tilde{\mathcal{T}}(x) = \tilde{V}^{-1}x - \tilde{a}$ and $\tilde{\mathcal{G}}(y) = \tilde{R}y$ represent a new scaling scheme. A new scaled problem, $\Gamma\langle \tilde{V}, \tilde{R}\rangle$ is defined in an analogous manner as the scaled problem, $\Gamma\langle V, R\rangle$, was defined by equation (9). Hence variables and functions in this new scheme are scaled as:

$$\begin{aligned} \tilde{x} &= \tilde{\mathcal{T}}(x) = \tilde{V}^{-1}x - \tilde{a} \qquad \text{and} \\ \tilde{g}(\tilde{x}) &= \tilde{\mathcal{G}}(g(\tilde{\mathcal{T}}^{-1}(\tilde{x}))) = \tilde{R}g(\tilde{\mathcal{T}}^{-1}(\tilde{x})). \end{aligned}$$

What do we mean by equivalence? Given a solution to either the original problem or the scaled problem we should be able to map it to the other uniquely and verify that the mapped point is indeed a solution. The first step is therefore to develop a mapping. By the scaling scheme, we can transform variables uniquely. For instance, given a vector $\overline{x}$ in the scaled problem space we can construct a vector $x = \mathcal{T}^{-1}(\overline{x})$ for the original problem space and vice versa. A similar scheme needs to be developed for the dual multipliers. We start with a mapping for the constraint and bound duals in the following lemma but prove the results at the end of this section after all the necessary pieces have been shown to be correct.

Lemma 3.2 (Equivalence of Solutions) *Given a solution $(\overline{x}\,; \overline{y}, \overline{w}, \overline{z})$ to the scaled problem, $\Gamma\langle V, R\rangle$, a solution $(x\,; y, w, z)$ to the original problem, Γ, can be constructed as:*

$$\begin{aligned} x &= \mathcal{T}^{-1}(\overline{x}) = V\,(\overline{x} + a)\,, \\ y &= \frac{1}{r_{m+1}} R\overline{y}, \\ w &= \frac{1}{r_{m+1}} V^{-1}\overline{w} \qquad \textit{and} \\ z &= \frac{1}{r_{m+1}} V^{-1}\overline{z}. \end{aligned}$$

Conversely, any solution $(x\,; y, w, z)$ to the original problem, Γ, can be transformed to a solution of the scaled problem, $\Gamma\langle V, R\rangle$, by using inverse of the transformations shown above.

Proof of Equivalence of Solutions: We defer the proof until the end of this section. □

Since we want the dual multiplier of the objective function to be always one, when we map from the scaled values we need to ensure that post mapping, the value remains one. This is the motivation for the factor r_{m+1} in the above mapping for constraint duals. Thus our choice of constraint dual mapping implies that *the relative importance of individual functions in the Lagrangian is same* in both the scaled problem as well as the original problem. The factor r_{m+1}, used in bound duals' mapping, arises because the KKT conditions need to be equivalent. The following corollary addresses how to adjust the scaled problem duals when scaling scheme changes.

Corollary 3.1 (Scale Adjustment for Solutions) *Let $\Gamma\langle \tilde{V}, \tilde{R}\rangle$ be a new scaled problem associated with a new scaling scheme defined by $\tilde{\mathcal{T}}$ and $\tilde{\mathcal{G}}$ as mentioned earlier. Given a solution $(\overline{x}\,; \overline{y}, \overline{w}, \overline{z})$ of the scaled problem, $\Gamma\langle V, R\rangle$, we can get a solution $(\tilde{x}\,; \tilde{y}, \tilde{w}, \tilde{z})$ for* *the new scaled problem, $\Gamma\langle\tilde{V},\tilde{R}\rangle$, as shown below:*

$$\begin{aligned} \tilde{x} &= \tilde{V}^{-1}V(\overline{x}+a)-\tilde{a}, \\ \tilde{y} &= \frac{\tilde{r}_{m+1}}{r_{m+1}}\tilde{R}^{-1}R\overline{y}, \\ \tilde{w} &= \frac{\tilde{r}_{m+1}}{r_{m+1}}\tilde{V}V^{-1}\overline{w} \qquad \textit{and} \\ \tilde{z} &= \frac{\tilde{r}_{m+1}}{r_{m+1}}\tilde{V}V^{-1}\overline{z}. \end{aligned}$$

Proof of Scale Adjustment for Solutions: We use lemma (3.2) to get a solution to the original problem, Γ, from a solution $(\overline{x}\,;\overline{y},\overline{w},\overline{z})$ to the scaled problem, $\Gamma\langle V,R\rangle$. Then the converse part of the lemma can be used to construct a solution to the new scaled problem, $\Gamma\langle\tilde{V},\tilde{R}\rangle$ from the solution to the original problem, Γ.

To illustrate, we first transform $\overline{x}$ to the original problem solution, $x=\mathcal{T}^{-1}(\overline{x})$. Then, a solution, $\tilde{x}$, to the new scaled problem, $\Gamma\langle\tilde{V},\tilde{R}\rangle$ is given by:

$$\tilde{x}=\tilde{\mathcal{T}}(\mathcal{T}^{-1}(\overline{x}))=\tilde{V}^{-1}\left(V(\overline{x}+a)\right)-\tilde{a}=\tilde{V}^{-1}V(\overline{x}+a)-\tilde{a}.$$

Similarly, the constraint duals are first mapped to the original problem, Γ, and then to the new scaled problem, $\Gamma\langle\tilde{V},\tilde{R}\rangle$, in the following:

$$\tilde{y}=\tilde{r}_{m+1}\tilde{R}^{-1}y=\tilde{r}_{m+1}\tilde{R}^{-1}\frac{1}{r_{m+1}}R\overline{y}=\frac{\tilde{r}_{m+1}}{r_{m+1}}\tilde{R}^{-1}R\overline{y}.$$

Reader can verify that bound duals can be deduced similarly. □

The corollary shows that the ratio of objective function scale factors under the two scaling schemes is important for adjustment of all the duals. In addition, the constraint duals depend on the ratios of individual function scale factors, $\tilde{r}_i/\overline{r}_i$ under the two scaling schemes. Similarly, the bound duals get adjusted based on individual variable scale factors ratios, $\tilde{v}_j/\overline{v}_j$. The primal variables get adjusted by reverse of the ratios used for the bound duals, but there is an additional dependence on an offset factor.

3.3.1 Lagrangian Hessian

In the following lemma we establish the relationship of the scaled problem Lagrangian Hessian to the original problem Hessian.

Lemma 3.3 (Lagrangian Hessian Equivalence) *The Lagrangian Hessian of the scaled problem, $\Gamma\langle V,R\rangle$, evaluated at any point $\overline{x}$ such that $\overline{l}\le\overline{x}\le\overline{u}$ and constraint duals $\overline{y}$, can be stated in terms of the original problem Hessian as shown below:*

$$\overline{L}(\overline{x})=r_{m+1}VL(x)V,$$

where $x = \mathcal{T}^{-1}(\overline{x})$ and the constraint duals y for the original problem are defined by lemma (3.2). And the converse holds too, i.e., the Lagrangian Hessian of the original problem, Γ, evaluated at any point $l \le x \le u$ and constraint duals y can be stated in terms of the scaled problem as:

$$L(x) = \frac{1}{r_{m+1}} V^{-1} \overline{L}(\overline{x}) V^{-1}.$$

Proof of Lagrangian Hessian Equivalence: Similar to the expression for Lagrangian Hessian of the original problem in equation (4), we can write one for the scaled problem as:

$$\overline{L}(\overline{x}) = \overline{y}^t \overline{H}(\overline{x}) = \sum_{i=1}^{m+1} \overline{y}_i \overline{H}^i(\overline{x}).$$

The individual scaled function Hessians are expressed in terms of the unscaled Hessians in equation (8) as:

$$\overline{H}^i(\overline{x}) = r_i \, V^t \, H^i(x) \, V,$$

where $x = \mathcal{T}^{-1}(\overline{x})$. The constraint duals defined by lemma (3.2) imply that $r_i \overline{y}_i / r_{m+1} = y_i$ or equivalently, $r_i \overline{y}_i = r_{m+1} y_i$. Hence, combining this with the expression for scaled individual function Hessians,$\overline{H}^i$, we can rewrite the scaled Lagrangian Hessian as:

$$\overline{L}(\overline{x}) = \sum_{i=1}^{m+1} \overline{y}_i r_i \, V^t \, H^i(x) \, V = r_{m+1} V^t \left(\sum_{i=1}^{m+1} y_i H^i(x) \right) V = r_{m+1} V^t \, L(x) \, V.$$

This yields first part of the desired result.

To prove the converse based on this is trivial since (a) the diagonal matrix V is invertible, (b) the mapping $\mathcal{T}(x)$ and mappings for constraint duals are one-to-one, and (c) r_{m+1} is strictly positive. □

We see that the scaled Lagrangian Hessian is *almost independent of the function scale factors* except for a scalar r_{m+1} that is based on the objective function's scale factor. More formally, the following corollary provides a basis for adjusting the Lagrangian Hessian when the scaling changes. This becomes critical with devices like quasi-Newtonian approximations of the Lagrangian Hessian.

Corollary 3.2 (Scale Adjustment for Lagrangian Hessian) *Let $\Gamma\langle \tilde{V}, \tilde{R} \rangle$ be a new scaled problem associated with a new scaling scheme defined by $\tilde{\mathcal{T}}$ and $\tilde{\mathcal{G}}$ as mentioned earlier. Then the new scaled Lagrangian Hessian $\tilde{L}(\tilde{x})$ evaluated at $\tilde{x} = \tilde{\mathcal{T}}(\mathcal{T}^{-1}(\overline{x}))$ can be expressed as*

$$\tilde{L}(\tilde{x}) = \frac{\tilde{r}_{m+1}}{r_{m+1}} \tilde{V} V^{-1} \overline{L}(\overline{x}) V^{-1} \tilde{V}.$$

Proof of Scale Adjustment for Lagrangian Hessian: Let $x = \mathcal{T}^{-1}(\overline{x})$ and $y = 1/r_{m+1}R\overline{y}$. Then from the converse part of lemma (3.3) we get

$$L(x) = \frac{1}{r_{m+1}}V^{-1}\overline{L}(\overline{x})V^{-1}.$$

From the lemma we also have that when $\tilde{x} = \tilde{\mathcal{T}}(x)$ and $\tilde{y} = \tilde{r}_{m+1}\tilde{R}^{-1}y$ the scaled Lagrangian Hessian is:

$$\tilde{L}(\tilde{x}) = \tilde{r}_{m+1}\tilde{V}L(x)\tilde{V}.$$

Combining the two expressions for the Lagrangian Hessian gives the required result. □

Note, the scale adjustment requires multiplying previously scaled Hessian with the ratio of objective function factors and ratios of variable factors in the two scaling schemes. Thus allowing a change of scaling scheme is $O(n^2)$ and hence is not expensive.

In the following corollary, we show that if the Lagrangian Hessian of the original problem is positive semi-definite on a subspace, then the property holds for the scaled problem too on a scaled subspace. This is required to establish second order conditions for a local optimum.

Corollary 3.3 (Positive Definiteness of Lagrangian Hessian) *If the Lagrangian Hessian, $L(x; y)$, of the original problem, Γ, is positive definite (or semi-definite) on a subspace $M \subseteq \Re^n$, then the Hessian, $\overline{L}(\overline{x}; \overline{y})$, of the scaled problem, $\Gamma\langle V, R\rangle$ evaluated at $\overline{x}$ and with constraint duals $\overline{y}$ as defined in lemma (3.2) is positive definite (or semi-definite) on a scaled subspace $\overline{M}$ defined by:*

$$\overline{M} = \left\{\overline{d} \in \Re^n : V\overline{d} \in M\right\}.$$

The converse holds too, i.e., if the Lagrangian Hessian of the scaled problem $\overline{L}(\overline{x}; \overline{y})$ is positive (semi) definite on the subspace $\overline{M} \subseteq \Re^n$, then Hessian of the original problem $L(x; y)$ is positive (semi) definite on the subspace M defined by:

$$M = \left\{d \in \Re^n : V^{-1}d \in \overline{M}\right\}.$$

Proof of Positive Definiteness of Lagrangian Hessian: We drop the arguments for Lagrangian Hessian since they are unambiguous from the context. For any $d \in M$, positive definiteness implies $d^tLd > 0$ (for semi-definite case ≥ 0). Using lemma (3.3), we can express the original problem Lagrangian Hessian in the scaled problem terms. Hence, for $d \in M$,

$$d^tLd = \frac{r_{m+1}}{r_{m+1}}d^tV^{-1}VLVV^{-1}d = \frac{1}{r_{m+1}}d^tV^{-1}\overline{L}V^{-1}d = \frac{1}{r_{m+1}}\overline{d}^t\,\overline{L}\,\overline{d}.$$

The first step follows from $r_{m+1}/r_{m+1} = 1$ and $V^{-1}V = VV^{-1} = I$. The second step follows from equivalence of Hessians in lemma (3.3), defining $\overline{L} = r_{m+1}VLV$. The last step follows from the substitution $\overline{d} = V^{-1}d$. Since $V\overline{d} = d \in M$ we conclude that $\overline{d} \in \overline{M}$ and the result follows.

Proving the converse is a similar exercise. □

3.3.2 Active Inequalities and Complementary Slackness

The only inequalities present in the nonlinear problem of equation (1) are the simple bounds on the variables. First we show that the set of active bounds is identical in both the scaled problem as well as original problem. Then we show that the complementary slackness conditions are equivalent.

Lemma 3.4 (Equivalence of Active Bounds) *For any point $l \le x \le u$ of the original problem, Γ, let sets U and L denote variables that are at their upper and lower bounds respectively. In other words,*

$$\begin{aligned} U &= \{j : x_j = u_j \qquad 1 \le j \le n\} \\ L &= \{j : x_j = l_j \qquad 1 \le j \le n\}. \end{aligned}$$

Let sets $\overline{U}$ and $\overline{L}$ of upper and lower bounded variables be defined analogously at $\overline{x} = \mathcal{T}(x)$ for the scaled problem, $\Gamma\langle V, R\rangle$. Then the sets are identical, i.e., $\overline{U} = U$ and $\overline{L} = L$. Conversely, the sets of active lower and upper bounds in a scaled problem, $\Gamma\langle V, R\rangle$, at $\overline{x}$ are identical to those in the original problem at $x = \mathcal{T}^{-1}(\overline{x})$.

Proof of Equivalence of Active Bounds: Sets U and L are mutually exclusive, i.e., $L \cap U = \emptyset$ by consistent bounds assumption (2.1) that assumes $l < u$. Hence the bounds for any variable j can never be equal, i.e.,$l_j = u_j$. In addition, since $l \le x \le u$, then $\mathcal{T}(l) \le \mathcal{T}(x) \le \mathcal{T}(u)$. Hence, $\overline{l} \le \overline{x} \le \overline{u}$.

For any variable at its upper bound, $j \in U$, the statement $x_j = u_j$ implies $x_j/v_j - a_j = u_j/v_j - a_j$. In other words, $\overline{x}_j = \overline{u}_j$ or $j \in \overline{U}$. Hence, $U \subseteq \overline{U}$.

The reverse is also true by a similar argument. If $j \in \overline{U}$ then $\overline{x}_j = \overline{u}_j$. This in turn implies, $v_j(\overline{x}_j + a_j) = v_j(\overline{u_j} + a_j)$ or $x_j = u_j$. Thus $\overline{U} \subseteq U$. This, together with the previous statement, $U \subseteq \overline{U}$, we conclude $U = \overline{U}$.

The argument for active lower bounds will be identical with u_j and $\overline{u}_j$ replaced by l_j and $\overline{l}_j$ respectively. Hence we conclude that $L = \overline{L}$.

The proof also indicates that the converse holds too. □

We examine complementary slackness conditions in the following lemma.

Lemma 3.5 (Equivalence of Complementary Slackness Conditions) *Let a solution to the scaled problem, $\Gamma\langle V, R\rangle$ be $(\overline{x}\,; \overline{y}, \overline{w}, \overline{z})$ and let $(x\,; y, w, z)$ be a corresponding point* *for the original problem, Γ, defined in lemma (3.2). Then, complementary slackness conditions for the scaled problem, namely,*

$$\begin{aligned}\overline{W}(\overline{u}-\overline{x}) &= \mathbf{0},\\ \overline{Z}(\overline{x}-\overline{l}) &= \mathbf{0}.\end{aligned}$$

imply that these conditions hold for the original problem, Γ, and vice versa.

Proof of Equivalence of Complementary Slackness Conditions: We first examine the complementary slackness conditions associated with upper bounds on the variables. Since $x = \mathcal{T}^{-1}(\overline{x})$ and $u = \mathcal{T}^{-1}(\overline{u})$ we can write the upper bound conditions as:

$$\overline{W}\left(\overline{u}-\overline{x}\right) = \overline{W}\left(\mathcal{T}(u)-\mathcal{T}(x)\right) = \overline{W}\left(V^{-1}u - a - V^{-1}x + a\right) = \mathbf{0}.$$

Since $\overline{W}$ and V are diagonal matrices, the product $\overline{W}V^{-1}$ is same as $V^{-1}\overline{W}$. From the lemma (3.2) we have $w = 1/r_{m+1}V^{-1}\overline{W}$. Hence using the diagonal matrix W constructed from components of vector w we get:

$$\overline{W}V^{-1}\left(u-x\right) = r_{m+1}W\left(u-x\right) = \mathbf{0}.$$

Dividing both sides by the positive scalar $1/r_{m+1}$ we see that the complementary slackness conditions associated with the upper bound constraints are met for the original problem too. The reader can verify that the lower bound complementary slackness conditions are also met in a similar manner, i.e.,

$$\overline{Z}\left(\overline{x}-\overline{l}\right) = \mathbf{0} \Rightarrow Z\left(x-l\right) = \mathbf{0}.$$

Readers can verify that proving the converse is a similar exercise. □

3.3.3 Lagrangian Gradient

We can express the Lagrangian gradient of the scaled problem in terms of gradient of the original problem and vice versa.

Lemma 3.6 (Lagrangian Gradient Equivalence) *The Lagrangian gradient of the scaled problem, $\Gamma\langle V, R\rangle$, can be expressed in terms of gradient of the original problem, as:*

$$\nabla\overline{\mathcal{L}}\left(\overline{x}\,;\overline{y},\overline{w},\overline{z}\right) = r_{m+1}V\;\nabla\mathcal{L}\left(x\,;y,w,z\right),$$

where x and original problem duals are defined according to lemma (3.2). Conversely,

$$\nabla\mathcal{L}\left(x\,;y,w,z\right) = \frac{1}{r_{m+1}}V^{-1}\;\nabla\overline{\mathcal{L}}\left(\overline{x}\,;\overline{y},\overline{w},\overline{z}\right),$$

expresses original problem Lagrangian gradient in terms of the scaled problem gradient.

Proof of Lagrangian Gradient Equivalence: The Lagrangian gradient of the scaled problem with respect to $\overline{x}$ is given by:

$$\nabla\overline{\mathcal{L}}\left(\overline{x}\,;\overline{y},\overline{w},\overline{z}\right)=\overline{J}(\overline{x})^t\overline{y}+\overline{w}-\overline{z}.$$

Equation (7) defines the scaled problem Jacobian in terms of the original Jacobian, using $x=\mathcal{T}^{-1}(\overline{x})$, as:

$$\overline{J}(\overline{x})=RJ(x)V.$$

Thus we get an expression involving only the original Jacobian:

$$\nabla\overline{\mathcal{L}}\left(\overline{x}\,;\overline{y},\overline{w},\overline{z}\right)=V^t\;J(x)^t\;R^t\overline{y}+\overline{w}-\overline{z}=V\;J(x)^t\;R\overline{y}+\overline{w}-\overline{z}.$$

The last step follows because diagonal matrices are symmetrical, hence for V and R the transpose operation results in no change, i.e., $V=V^t$ and $R=R^t$. From lemma (3.2) we can replace $R\overline{y}$ with $r_{m+1}y$, $\overline{w}$ with $r_{m+1}Vw$ and $\overline{z}$ with $r_{m+1}Vz$ to get:

$$\nabla\overline{\mathcal{L}}\left(\overline{x}\,;\overline{y},\overline{w},\overline{z}\right)=r_{m+1}V\;\left(J(x)^t\;y+w-z\right)=r_{m+1}V\;\nabla\mathcal{L}\left(x\,;y,w,z\right).$$

The scale factors are strictly positive, $r_{m+1}\neq 0$ and V is invertible. Hence the converse can be proved easily. □

Similar to the Lagrangian Hessian, the gradient is almost independent of the function scale factors except for the objective function scale, r_{m+1}. In the following corollary we state an obvious conclusion that if Lagrangian gradient is zero in the scaled problem, then the original problem gradient for a point mapped according to lemma (3.2) is zero.

Corollary 3.4 (Equivalence of Vanishing Lagrangian Gradient) *If the Lagrangian gradient* $\nabla\overline{\mathcal{L}}\left(\overline{x}\,;\overline{y},\overline{w},\overline{z}\right)$ *of the scaled problem,* $\Gamma\langle V,R\rangle$*, is zero then so is the gradient* $\nabla\mathcal{L}\left(x\,;y,w,z\right)$ *of the original problem,* Γ*, when evaluated at a corresponding point as defined in lemma (3.2) and vice versa.*

Proof of Equivalence of Vanishing Lagrangian Gradient: Trivially established using lemma (3.6). The matrix V is nonsingular and $r_{m+1}\neq 0$. □

We show adjustment of Lagrangian gradient for scaling scheme changes in the following corollary that is similar to the result about adjusting Lagrangian Hessian in corollary (3.2).

Corollary 3.5 (Scale Adjustment for Lagrangian Gradient) *Let* $\Gamma\langle\tilde{V},\tilde{R}\rangle$ *be a new scaled problem associated with a new scaling scheme defined by* $\tilde{\mathcal{T}}$ *and* $\tilde{\mathcal{G}}$ *mentioned earlier. The new scaled Lagrangian gradient* $\nabla\tilde{\mathcal{L}}$ *evaluated at* $\tilde{x}=\tilde{\mathcal{T}}(\mathcal{T}^{-1}(\overline{x}))$ *can be expressed as*

$$\nabla\tilde{\mathcal{L}}(\tilde{x};\tilde{y},\tilde{w},\tilde{z})=\frac{\tilde{r}_{m+1}}{r_{m+1}}\tilde{V}V^{-1}\overline{\mathcal{L}}(\overline{x};\overline{y},\overline{w},\overline{z}).$$

Proof of Scale Adjustment for Lagrangian Gradient: The proof is almost identical to the result shown for Hessians in corollary (3.2). Let $x = \mathcal{T}^{-1}(\overline{x})$ and $y = 1/r_{m+1} R\overline{y}$. Then from the converse part of lemma (3.6) we get

$$\nabla \mathcal{L}\left(x\,; y, w, z\right) = \frac{1}{r_{m+1}} V^{-1}\ \nabla \overline{\mathcal{L}}\left(\overline{x}\,; \overline{y}, \overline{w}, \overline{z}\right),$$

From the lemma we also have that when $\tilde{x} = \tilde{\mathcal{T}}(x)$ and $\tilde{y} = \tilde{r}_{m+1}\tilde{R}^{-1}y$ the scaled Lagrangian gradient is:

$$\nabla \tilde{\mathcal{L}}\left(\tilde{x}\,; \tilde{y}, \tilde{w}, \tilde{z}\right) = \tilde{r}_{m+1}\tilde{V}\ \nabla \mathcal{L}\left(x\,; y, w, z\right),$$

Combining the two expressions for the Lagrangian gradient gives the result. □

3.3.4 Regular Points and Tangent Subspaces

We establish regularity conditions by showing that a regular point in the scaled problem maps to a regular point in the original problem and vice versa. This also leads to the conclusion that the associated tangent subspaces are equivalent. Tangent subspaces assume significance for second order conditions of local optimality. We need to show that the Lagrangian Hessian is positive (semi) definite on the tangent subspace for these conditions to hold. We first establish an obvious result that row or column scaling does not change the rank of a matrix using singular value decomposition in the following lemma.

Lemma 3.7 (Matrix Rank under Scaling) *Given a real $m \times n$ matrix A with a rank $r \leq \min\{m, n\}$, its rank does not change if its rows or columns are multiplied by non-zero values.*

Proof of Matrix Rank under Scaling: We use singular value decomposition to prove the result. Since A has rank r, by [Wat91, Theorem 7.1.12, page 395], there exist isometries $U \in \mathfrak{R}^{m\times r}$ and $V \in \mathfrak{R}^{n\times r}$ and an $r \times r$ diagonal matrix Σ with singular values of A as its main diagonal entries $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ such that

$$A = U\Sigma V^t.$$

Multiplying each row by a non-zero is equivalent to premultiplying matrix A by an $m\times m$ diagonal matrix D_1 whose diagonal entries are the non-zero row multipliers. Similarly, multiplying each column by a non-zero is equivalent to post multiplying matrix A with an $n\times n$ diagonal matrix D_2 whose diagonal entries are the non-zero multipliers of the columns. Let $\hat{A} = D_1AD_2$. Using singular value decomposition, we will show that its rank does not change.

$$\hat{A} = D_1AD_2 = D_1U\Sigma V^tD_2 = \left(D_1U\right)\Sigma\left(D_2V\right)^t = \hat{U}\Sigma\hat{V}.$$

If $\hat{U} = D_1U$ and $\hat{V} = D_2V$ are isometries, then the rank of matrix $\hat{A}$ is still r by the theorem. Let u_i and u_j be any two distinct columns i and j of the isometry U. Since U is an isometry, $u_i^t u_j = 0 \ \forall \ i \neq j$. Let d_i^1 and d_j^1 be the corresponding non-zero diagonal entries in D_1. Then the corresponding columns in $\hat{U}$ are given by $\hat{u}_i = d_i^1 u_i$ and $\hat{u}_j = d_j^1 u_j$. But $d_i^1 d_j^1 u_i^t u_j = d_i^1 d_j^1 .0 = 0 \ \forall \ i \neq j$. Hence, $\hat{u_i}^t \hat{u}_j = 0 \ \forall \ i \neq j$ in matrix $\hat{U}$. In other words, $\hat{U}$ is an isometry. By similar reasoning, $\hat{V}$ is an isometry. Hence rank of matrix $\hat{A}$ is the same because Σ has not changed. □

If it was possible to change a matrix rank by scaling rows or columns, then rank deficiency would never pose a problem to solvers. We were unable to solve some difficult problems because of this issue in [Gaj95].

A point $x \in \Re^n$ is a *regular point* of the nonlinear problem, Γ, defined in equation (1) if it satisfies all the constraints, namely, nonlinear equations $h(x) = b$, explicit variable bounds $l \leq x \leq u$ and in addition, the gradients of *active* constraints are linearly independent. A constraint is *active* or *binding* if it holds as an equality. Hence, the set of active constraints will always include $h(x) = b$. In addition, some of the variables that are at their bounds, either lower or upper, also form part of the active set. In the following lemma we show that the regular point of the original problem can be mapped to a regular point of the scaled problem and vice versa.

Lemma 3.8 (Equivalence of Regular Points) *If x is a regular point of the original nonlinear problem, Γ, defined by equation (1) then $\overline{x} = \mathcal{T}(x)$ is a regular point of the scaled problem, $\Gamma\langle V, R\rangle$, defined by equation (9) and vice versa, i.e., given a regular point $\overline{x}$ of the scaled problem, $x = \mathcal{T}^{-1}(x)$ is a regular point of the original problem.*

Proof of Equivalence of Regular Points: Objective function can be ignored while discussing regularity conditions. Let an $m \times m$ diagonal matrix R' be a restriction of the row scale factors matrix R where the $m+1$ row and column related to the objective function has been omitted. Similarly, let an $m \times n$ matrix $J'(x) = \nabla h(x)$ denote the gradients of constraint functions only. Hence, J' is the same as problem Jacobian $J(x)$ without the last row of objective function gradients.

We first show that the scaled nonlinear constraints are satisfied. Given x is a regular point of the original problem, Γ, let $\overline{x} = \mathcal{T}(x)$ be its corresponding equivalent for the scaled problem, $\Gamma\langle V, R\rangle$. Since x is regular, the nonlinear equations, $h(x) = b$, are satisfied. Since $x = \mathcal{T}^{-1}(\overline{x})$ by the reverse transformations and we can multiply both, left hand side, $h(x)$, and right hand side, b, by nonsingular R', we can write:

$$h(x) = b \quad \Longrightarrow \quad R'h(x) = R'h(\mathcal{T}^{-1}(\overline{x})) = \overline{h}(\overline{x}) = R'b = \overline{b}.$$

In other words, for the scaled problem $\overline{h}(\overline{x}) = \overline{b}$.

Similarly, $l \leq x \leq u$ implies, $\mathcal{T}(l) \leq \mathcal{T}(x) \leq \mathcal{T}(u)$ and hence we conclude that $\overline{l} \leq \overline{x} \leq \overline{u}$. Thus the scaled point $\overline{x}$ satisfies the constraints of the scaled problem, $\Gamma\langle V, R\rangle$,.

To show that $\overline{x}$ is a regular point given x is, we need to show that gradients of active constraints form a linearly independent set. As mentioned earlier, the nonlinear equations $h(x) = b$ are always included in the set of active constraints. Hence the equality constraint gradients, $J'(x)$ should be included in determining the linear independence of the active set.

The sets of active upper and lower bounds are identical in both the scaled problem and the original problem from lemma (3.4). Hence, we will use sets U and L to denote active upper and lower bounds of the scaled problem also. For any variable $j \in U$, the active constraint implies $x_j = u_j$ and similarly, for any variable $j \in L$, we have $x_j = l_j$ in the original problem. In the scaled problem same conditions hold but for scaled values of corresponding quantities. Hence, the gradient of an active bound is the appropriate row of an $n \times n$ identity matrix I_n.

Let $b = |U \cup L|$ be the total number of variables at their bounds, hence $b \leq n$. Let I_B be a $b \times n$ sub-matrix of identity matrix I_n that has been formed by selecting only rows that correspond to a variable at its bound. That is, I_B has a row e_j of the identity matrix if and only if $j \in U \cup L$. Matrix I_B is of full rank b since its rows are linearly independent.

Given that x is a regular point of the original problem, Γ, then the $(m + b) \times n$ matrix A of gradients of the active set of constraints defined by

$$A = \begin{bmatrix} J'(x) \\ I_B \end{bmatrix},$$

has full rank. We need to show that corresponding matrix, $\overline{A}$, of gradients of active set of constraints in the scaled problem, $\Gamma\langle V, R\rangle$, also has full rank. Analogous to the scaled problem Jacobian in equation (7) we can express the scaled problem's constraint gradients in terms of the original gradient, using $\overline{x} = \mathcal{T}(x)$. This allows us to write $\overline{A}$ as:

$$\overline{A} = \begin{bmatrix} \overline{J}'(\overline{x}) \\ I_B \end{bmatrix} = \begin{bmatrix} R'\ J'(x)\ V \\ I_B \end{bmatrix}.$$

By lemma (3.7), the rank of $R'\ J'(x)\ V$ is same as rank of $J'(x)$. Hence we conclude that if A has full rank, then so does $\overline{A}$. Thus if x is a regular point, then so is $\overline{x} = \mathcal{T}(x)$.

The converse holds in a similar manner. Instead of R', V and the mapping $\overline{x} = \mathcal{T}(x)$, we use R'^{-1}, V^{-1} and the inverse mapping $x = \mathcal{T}^{-1}(\overline{x})$. □

At a regular point the tangent subspace is the same as the null space of the gradients of the active constraints [Lue84, page 298]. For the original problem, Γ, the tangent subspace M at a regular point x is defined by:

$$M = \{d \in \Re^n : \nabla h(x) d = \mathbf{0} \quad \wedge d_j = 0 \quad \forall j \in U \cup L\}. \tag{10}$$

Note, components corresponding to variables at their bounds are zero. The following lemma shows the relationship between tangent subspaces of the original problem and the scaled problem.

Lemma 3.9 (Equivalence of Tangent Subspaces) *Let x be a regular point of the original problem Γ, with the associated tangent subspace M, defined by equation (10). Hence, by lemma (3.8) $\overline{x} = \mathcal{T}(x)$ is the regular point of the scaled problem, $\Gamma\langle V, R\rangle$. Let its tangent subspace $\overline{M}$ be defined as in equation (10). Then for any vector $d \in M$, the vector $V^{-1}d \in \overline{M}$. Conversely, for every vector $\overline{d} \in \overline{M}$, the vector $V\overline{d} \in M$.*

Proof of Equivalence of Tangent Subspaces: The set of active bounds is identical in both the original and scaled problems from lemma (3.4). Hence, using equation (10), we can write the tangent subspace of the scaled problem as:

$$\overline{M} = \left\{\overline{d} \in \mathfrak{R}^n : \nabla\overline{h}(\overline{x})\overline{d} = \mathbf{0} \quad \wedge \overline{d}_j = 0 \;\; \forall j \in U \cup L\right\}.$$

As was shown in the proof of lemma (3.8), similar to expression for scaled problem Jacobian in equation (7), we can write

$$\nabla\overline{h}(\overline{x})\overline{d} = R'\nabla h(x)V\overline{d} = \mathbf{0} \quad\Longrightarrow\quad \nabla h(x)V\overline{d} = \mathbf{0}.$$

The implication follows by multiplying both sides by R'^{-1} . This allows us to simplify the tangent subspace for the scaled problem as:

$$\overline{M} = \left\{\overline{d} \in \mathfrak{R}^n : \nabla h(x)V\overline{d} = \mathbf{0} \quad \wedge \overline{d}_j = 0 \;\; \forall j \in U \cup L\right\}.$$

In both the scaled problem as well as the original problem, the same components corresponding to active bounds have to be zero. Hence if $d \in M$, then

$$\nabla h(x)d = \nabla h(x)VV^{-1}d = \nabla h(x)V\left(V^{-1}d\right) = \mathbf{0},$$

implying, $\overline{d} = V^{-1}d \in \overline{M}$. Similarly, if $\overline{d} \in \overline{M}$, we see that $V\overline{d} \in M$. □

We alluded to scaled subspaces in corollary (3.3) while discussing positive semidefiniteness property of the Lagrangian Hessian. We can interpret corresponding tangent subspace of the scaled problem as a scaled version of tangent subspace of the original problem.

3.3.5 Equivalence of Solutions

We provide proof of equivalence of solutions that was deferred earlier. The proof is just a summarization of the results shown earlier.

Proof of Equivalence of Solutions in lemma (3.2): We will show that a solution $(\overline{x}\,;\overline{y}, \overline{w}, \overline{z})$ to the scaled problem, $\Gamma\langle V, R\rangle$, implies that $(x\,; y, w, z)$ obtained by the transformations of lemma (3.2) is a solution to the original problem, Γ. The converse also holds since each step listed in the following holds in both directions.

Since $\overline{x}$ is a regular point of $\Gamma\langle V, R\rangle$, by equivalence of regular points in lemma (3.8), x is also a regular point of Γ.

Since, $\overline{x}$ is a solution of $\Gamma\langle V, R\rangle$, Lagrangian gradient $\nabla\overline{\mathcal{L}}\,(\overline{x}\,;\overline{y},\overline{w},\overline{z}) = \mathbf{0}$ and by the vanishing gradients corollary (3.4), $\nabla\mathcal{L}\,(x\,;y,w,z) = \mathbf{0}$ for Γ too.

Since complementary slackness conditions hold for $\Gamma\langle V, R\rangle$, at $(\overline{x}\,;\overline{y},\overline{w},\overline{z})$, by the equivalence of complementary slackness conditions in lemma (3.5), they hold at $(x\,;y,w,z)$ for Γ too.

Thus we have shown that the first order KKT conditions for the scaled problem, $\Gamma\langle V, R\rangle$, imply they are met for the original problem, Γ, too.

From equivalence of tangent subspaces in lemma (3.9), for every vector $\overline{d}$ in the tangent subspace $\overline{M}$ of $\Gamma\langle V, R\rangle$, the vector $V\overline{d} \in M$ where M is the tangent subspace of Γ. From the equivalence of positive (semi) definiteness property in corollary (3.3), we have that positive (semi) definiteness of Lagrangian Hessian of $\Gamma\langle V, R\rangle$ implies the same for Γ. Thus second order sufficient (necessary) conditions of the scaled problem imply the same for the original problem.

As mentioned earlier, each step listed above holds for the converse. Hence, the converse holds too. □

4 Solution Methods

We use a generic nonlinear programming algorithm to represent a variety of solution methods like successive quadratic programming, barrier methods, reduced gradient methods etc. Our objective is to identify important steps that would be affected by scaling the nonlinear problem. Since our emphasis is on the numerical behavior, we ignore critical steps like manipulating sparse data structures used for managing derivative data. Needless to say, the generic algorithm is an abstraction and actual implementations will be far more complex.

In the following algorithm, not all steps need to be used by a method. For instance, only successive quadratic programming (SQP) type of methods would be approximating the Lagrangian Hessian L with $\hat{L}$. These methods would not be using second derivatives to compute the true Hessian. Barrier methods may operate with true Hessian or may approximate it slightly differently. We have used interior point methods with both variants [GL01]. The quasi-Newtonian approximation used by barrier methods uses gradients of the barrier problem's Lagrangian unlike the SQP method. Hence the terms $w - z$ do not appear in the gradients used to update the approximate Hessian.

For Newton like methods, to prevent divergence from poor starting points we have used line search based on merit functions, but trust region methods are also commonly used. In trust region methods, instead of adjusting step length, size of the trust region (a sphere for the L_2 norm and an n-cube for L_1 norm) is reduced if the actual function values are not predicted well by model, usually a quadratic, of the function. Thus the step length remains one but the direction changes. In practice, one often ends up using a hybrid approach. For instance, in [GL01], whenever we did not meet sufficient slope or step size criteria, we added a trust region constraint $d^t d \leq \Delta$.

For merit function, in SQP and interior point methods L_1 exact penalty is often used. Other choices include the differentiable L_2 penalty method. For reduced gradient methods, since feasibility is always maintained, the objective function serves as a good candidate for merit function.

For reduced gradient methods, finding a descent direction and step size determination are not independent as algorithm (1) seems to suggest. For a full rank constraint gradients matrix $J'(x) = \nabla h(x)$, null space of J', and range space of J'^t span the complete space $\Re^n$. Hence the direction vector can be construed as a component in the null space of J' and another component in the range space of $(J')^t$. First a component, d_1, in the null space is constructed based on a reduced system which requires finding factors of an $m \times m$ basis matrix. Then a system $h(x + \alpha d_1 + d_2(\alpha)) = b$ needs to be solved for each step size to find m components of d_2. Thus the movement is not along a straight line $x + \alpha d$ as in SQP or interior point methods, but along a surface defined by d_1 and the problem constraints.

In most of the nonlinear programming methods that we are discussing, each iteration of a method involves one or more linear systems. For example, in a method that is solving a quadratic subproblem by active set strategy in each iteration, the first order conditions involve a linear system. There may be as many linear systems generated as different active sets are used by the quadratic solver. On the other hand, in a barrier problem, only one linear system may be solved per iteration. In reduced gradient methods, factors of an $m \times m$ basis matrix are generated to find the direction in the null space of the J' and also a system of nonlinear equations in terms of m basic variables need to be solved for each step size tried. While discussing scaling effects in section (5), we would like to focus on nonlinear optimization issues primarily. Hence we make the following unrealistic assumption:

Assumption 4.1 (Perfect Linear System Solver) *We assume that a linear system is solved using exact arithmetic with a device that can represent real numbers and incurs no floating point operation errors.*

Justification for Perfect Linear System Solver: We want to look at scaling effects only as far as it impacts preparing data for a linear system. By making this assumption, we ignore propagation of errors that happen in practice from the nonlinear method to the linear system solver and vice versa. In most implementations, linear systems are often scaled before each solve and this scaling is independent of the scaling adopted by the nonlinear method. For instance, schemes suggested in [GL06] may be used for such a purpose. □

Algorithm 1 Nonlinear Solver Algorithm

```
Require: $(l \leq x_0 \leq u) \wedge (l < u)$
$(x; y, w, z) \leftarrow (x_0; \mathbf{1}, \mathbf{1}, \mathbf{1})$ { Prefer non-zero duals initially }
$\hat{L} \leftarrow I_n$ { Approximate Lagrangian Hessian }
for $i = 1$ to 300 do {Main iteration loop}
  $q^- \leftarrow J^t y + w - z$ { Current duals but old Jacobian $J(x)$ when $i > 1$ }
  $J \leftarrow \nabla g$ { Jacobian }
  $L \leftarrow y^t H = y^t \nabla^2 g$ { Lagrangian Hessian }
  $q \leftarrow J^t y + w - z$ { Lagrangian gradient }
  Check for termination
  Update $\hat{L}$ with $(x^-, x)$ and $(q^-, q)$ { BFGS type of rank 2 update when $i > 1$ }
  Method specific adjustments { Like barrier parameter $\mu$ }
  for $p = 1$ to 5 do {Descent direction $d$}
    Find direction $d$ and new duals $(y^+, w^+, z^+)$ { Trial values of duals }
Ensure: $l \leq x + \alpha_{\max} d \leq u$
    $\rho^+ \leftarrow \max\{\rho, 1.5|y^+|\}$ {Trial penalties}
    if $\phi'(0) < -\epsilon_\phi$ then {Sufficient slope}
      $\alpha \leftarrow \alpha_{\max}$ { Largest step size }
      for $l = 1$ to 10 do {Line search}
        if $\phi(\alpha) \leq \phi(0) + \epsilon_A \alpha \phi'(0)$ then {Armijjo's test: sufficient decrease}
          exit Line search
        end if
        Find new $\alpha \in (0, \alpha_{\max}]$
      end for
      if $\alpha > \epsilon_\alpha$ then {Sufficient step size}
        exit Descent direction $d$
      end if
    end if{ Sufficient slope }
  end for
  $x^- \leftarrow x$ { Previous value for quasi-Newton approximations }
  $(x; y, w, z; \rho) \leftarrow (x + \alpha d; y^+, w^+, z^+; \rho^+)$ { Commit }
end for
return $(x; y, w, z)$
```

5 Scaling Effects

Using algorithm (1) we identify critical steps that would be influenced by scaling. We look at termination criteria, behavior of merit functions, Hessian approximations and derivative calculations when differencing is used. Analytical derivatives are available to the same precision as function evaluations.

5.1 Termination Criteria

Before getting into criteria commonly used to terminate the solution process, we suggest that L_∞ should be used for medium to large problems, where typically the number of constraints is more than 5,000 and number of variables is more than 10,000. We have seen some authors use L_1or L_2 norms [LPY95]. For instance, if we were to test $\| d \|_1 \leq \epsilon$ with a tolerance $\epsilon = 10^{-5}$, and if each component was bounded $\epsilon_1 \leq | d_i | \leq \epsilon_2$, then $n\epsilon_1 \leq \| d \|_1 \leq n\epsilon_2$. Thus, if $\epsilon_1 = 10^{-7}$, for a thousand variable problem we will not able to meet the tolerance criterion. To get around this, some authors use scaled values of norms for tests where the scaling is a function of the dimension n. We believe that a simpler solution is to use L_∞ norm and it works well in practice [GL01]. Nash and Sofer [NS95, page 375] also suggest this.

We focus on only three criteria commonly used for termination checks mentioned in step (8) of algorithm (1). In actual implementations, because a variety of exceptions that need to be handled, termination tests are rarely simple. For instance, we [Gaj95] keep track of the best solution found so far, have to handle reset operations that involve starting back from previous commit point for some types of failures. More than 18 different causes for termination have been used in [GL01].

To test feasibility, we examine only the equations $h(x) - b$. Since by assumption (2.3) a solution method always respects explicit variable bounds, we are assured that step size calculation and descent direction estimation ensure this. Hence to test feasibility of constraints, a common test used is:

$$| h_i(x) - b_i | \leq \epsilon_f(1 + | b_i |).$$

Typically, a value of $\epsilon_f = 10^{-4}$ would be used. This test becomes unreliable when $| b_i |$ is small compared to operations defining $h_i(x)$. For example, in the function $h_i(x) = x_1^2 - x_2/x_3 = 0$, the test becomes an absolute test from a relative test since b_i has vanished. If the two operands involved in the subtraction are large, say order of 10^8, then the test is looking for a relative precision of about 10^{-12}. In this case, looking at relative sizes of operands involved in defining a constraint becomes important. Thus under the proposed scaling scheme, though $\overline{b_i} = 0$, it is hoped that the function scaling factor r_i and variable scaling factors v_1, v_2 and v_3 will make magnitudes of the operands relatively smaller. When two points are infeasible, we use the count of infeasibilities and if these are equal, then we look at the largest magnitude of infeasibility, $\| h(x) - b \|_\infty$ to decide which of the two points is better. This criteria becomes important in comparing the current best solution with a proposed solution.

For feasible points, the second criterion used is based on residuals of KKT conditions. Many authors, especially in interior point methods literature, use complex strategies that emphasize the three components of the residuals, namely, Lagrangian gradient, constraints and complementary slackness conditions in different ways. We had experimented with some of these termination criteria in [Gaj95], but we saw only marginal difference. A priori, we do not know what should be the correct weights for each component. In keeping with our philosophy of using L_∞ norm we have successfully used the following simpler criterion:

$$\max\{\| \nabla \mathcal{L}(x; y, w, z) \|_\infty , \| h(x) - b \|_\infty , \| W(u - x) \|_\infty , \| Z(x - l) \|_\infty\} \leq \epsilon_0.$$

The last two components related to complementary slackness conditions are unaffected by scaling except for a constant factor r_{m+1} as seen while proving equivalence of complementary slackness conditions in lemma (3.5). Scaling factor for a bound dual is a reciprocal of that used for the corresponding primal variable. The residuals of Lagrangian gradient get scaled by $r_{m+1}V$ as shown in equivalence of gradients in lemma (3.6) while the constraint residuals get scaled by R from the definition of scaled problem in equation (9).

The third important criterion we use is based on detecting no progress for feasible iterations. If for $k = 5$ consecutive iterations, the solution remains feasible but no change in the objective function value then a termination is called for. The test is of the type:

$$| f(x) - f(x^-) | \leq \epsilon_0(1 + | f(x) |).$$

This test is going to be impacted by scale factor r_{m+1} associated with the objective function.

For infeasible cases, we let a solution run to its iteration limit of 300. Often in such cases, their is no change in the solution since the solver is unable to either determine a downhill direction or step size is not sufficient. A criterion preferred by other authors that we would like to use in future is sensitive to variable scale factors and is of the type:

$$\| x - x^- \|_\infty \leq \epsilon_x(1 + \| x \|_\infty .$$

5.2 Hessian Approximation

Quasi-Newton approximations of the Lagrangian Hessian are based on rank two updates. A common method used is based on Powell's variant of the BFGS method for constrained problems. According to Luenberger [Lue84, page 269] the BFGS method has been empirically observed to be better than the DFP method. The update is shown step (9) of algorithm (1)

We use s for the displacement vector and p for gradient difference that are used in the updates. The vector $s = x - x^-$ denotes change in position between current iteration and previous one. Previous values of the variables, x^-, are saved just before committing in step (27) of algorithm (1). Hence vector s is sensitive to variable scaling.

Similarly, vector $p = q - q^-$ is used for change in the Lagrangian gradient and is shown in steps (7) and (4) of algorithm (1). From the equivalence of Lagrangian gradients lemma (3.6), we see that vector p is sensitive to variable scaling as well as scale factor associated with objective function.

The true Lagrangian Hessian itself is sensitive to variable scaling as seen in equivalence of Lagrangian Hessians lemma (3.3). The scaled Hessian is given by $\overline{L} = r_{m+1} VLV$. Hence we expect that in the limit, the approximation $\hat{L}$ will also be similarly sensitive.

5.3 Merit Functions

As mentioned earlier, merit functions are used to prevent divergence when a Newton type method is started from poor starting points. Since reduced gradient methods maintain feasibility through out the solution process, the objective function is a good candidate. Hence, slope of the merit function and the function values will be influenced by scale factor, r_{m+1}, associated with objective function and variable scale factors, V, in a manner similar to what has been discussed earlier. In the remainder, we focus on merit functions for methods that allow constraint violations at intermediate points.

We consider two candidates for the merit function, $\phi(\alpha)$, based on either the L_1 exact penalty or the L_2 quadratic penalty. Merit function's slope, $\nabla\phi(0)$, is used to test if the given direction is a descent direction in step (14) of algorithm (1) while Armijjo's criterion or some other suitable criteria is used in step (17) to ensure sufficient decrease in the merit function. Thus, line search, to find a step size requires calculation of slope once and one or more merit function evaluations for each step size tried.

A merit function is defined based on a penalty function, $\pi(x; \rho, \gamma, \beta) : \Re^n \mapsto \Re$, that penalizes constraint violations. The penalty weights, $\rho > 0$ are associated with equality constraints, $\gamma > 0$ and $\beta > 0$ are associated with upper and lower bound conditions respectively. For an equality constraint i, $h_i(x) - b_i$ measures constraint violation while $\max\{0, x_j - u_j\}$ and $\max\{0, l_j - x_j\}$ measure upper and lower bound violations for variable j respectively. The quadratic loss penalty function, based on the Euclidean metric, is defined by:

$$\begin{aligned}\pi(x;\rho,\gamma,\beta) &= f(x)+\frac{1}{2}\sum_{i=1}^{m}\rho_i\,(h_i(x)-b_i)^2+ \\ &\quad \frac{1}{2}\sum_{j=1}^{n}\gamma_j\,(\max\{0,x_j-u_j\})^2+\frac{1}{2}\sum_{j=1}^{n}\beta_j\,(\max\{0,l_j-x_j\})^2.\end{aligned} \tag{11}$$

The exact penalty function, based on the Manhattan metric, is similarly defined by:

$$\begin{aligned}\pi(x;\rho,\gamma,\beta) &= f(x)+\sum_{i=1}^{m}\rho_i\,|\,h_i(x)-b_i\,|+ \\ &\quad \sum_{j=1}^{n}\gamma_j\max\{0,x_j-u_j\}+\sum_{j=1}^{n}\beta_j\max\{0,l_j-x_j\}.\end{aligned} \tag{12}$$

A merit function , $\phi(\alpha;x,d):\mathfrak{R}^+\mapsto\mathfrak{R}$, measures progress along a direction $d\in\mathfrak{R}^n$ from a given point x based on a step size $\alpha>0$. It is defined in terms of the penalty function as:

$$\phi(\alpha;x,d)=\pi(x+\alpha d;\rho,\gamma,\beta).$$

If there is no ambiguity, we may refer to a merit function simply as $\phi(\alpha)$ since we are talking about it in the context of a line search along a direction d from the current point. We are also interested in its slope, $\nabla_\alpha(0)$, for judging sufficient decrease condition in step (14) of the algorithm (1). Before we can discuss sensitivity of merit function and its slope to scaling, we first develop simplified expressions for these values.

To write merit function expressions in matrix notation, for the equality constraint penalty weights, we define an $(m+1)\times(m+1)$ diagonal matrix Λ with a weight of one for the objective function row defined by:

$$\Lambda_{i,i}=\left\{\begin{array}{ll}\rho_i & \forall\, i=1,\ldots,m \\ 1 & \quad i=m+1\end{array}\right. .$$

We modify slightly the function $g:\mathfrak{R}^n\mapsto\mathfrak{R}^{m+1}$ to include the constraint right hand sides:

$$g_i(x)=\left\{\begin{array}{ll}h_i(x)-b_i & \forall\, i=1,\ldots,m \\ f(x) & \quad i=m+1\end{array}\right. .$$

The problem Jacobian $J(x):\mathfrak{R}^n\mapsto\mathfrak{R}^{(m+1)\times n}$ is same as before and it includes objective function gradient. When we want to exclude objective function, we use $g'(x)$ to denote just the equality constraints $h(x)-b$ and similarly, the restricted Jacobian that excludes the objective function gradient by $J'(x)$. Individual function gradients are row vectors to conform to the function row of the Jacobian.

In the following lemma under the assumption that a solution method respects explicit variable bounds, we develop simplified expressions for the merit function based on quadratic loss penalty. Penalties associated with variable bound violations can be ignored by the following lemma.

Lemma 5.1 (Merit Function based on Quadratic Loss Penalty) *For the merit function based on the quadratic loss penalty function of equation (11), under the assumption (2.3) that explicit variable bounds hold for a solution method, if there exists* $\alpha_{\max} > 0$ *such that* $l \leq x + \alpha_{\max} d \leq u$ *after step (12) of algorithm (1) then for the range* $\alpha \in [0, \alpha_{\max}]$*, the merit function can be simplified as:*

$$\phi(\alpha) = \frac{1}{2} \begin{pmatrix} g'(x + \alpha d) \\ 2 \end{pmatrix}^t \Lambda \; g(x + \alpha d),$$

and its slope is given by

$$\nabla_\alpha \phi(0) = \begin{pmatrix} g'(x) \\ 1 \end{pmatrix}^t \Lambda \; J(x) d.$$

Proof of Merit Function based on Quadratic Loss Penalty: By assumption (2.3), the explicit variable bounds, $l \leq x \leq u$, hold always for the solution method, and by the conditions of the lemma, $l \leq x + \alpha_{\max} d \leq u$. Hence, for the range, $\alpha \in [0, \alpha_{\max}]$, we can ignore the penalty terms related to variable bounds in the penalty function of equation (11) and write it as:

$$\phi(\alpha; x, d) = \pi(x + \alpha d; \rho) = f(x) + \frac{1}{2} \sum_{i=1}^{m} \rho_i \left(h_i(x + \alpha d) - b_i\right)^2.$$

This yields the simplified expression stated in the lemma for the merit function. To calculate the slope, we look at the gradient of the penalty function of equation (11). Its gradient, see [NS95, page 539], is given by:

$$\begin{aligned} \nabla_x \pi(x; \rho, \gamma, \beta) \;\; = \;\; & \nabla f(x) + \sum_{i=1}^{m} \rho_i \left(h_i(x) - b_i\right) \nabla h_i(x) + \\ & \sum_{j=1}^{n} \gamma_j \left(\max\left[0, x_j - u_j\right]\right) - \sum_{j=1}^{n} \beta_j \left(\max\left[0, l_j - x_j\right]\right). \end{aligned}$$

Since gradient of $l_j - x_j$ is -1, the last summation term has a minus sign in the slope calculation. Again by assumption (2.3), the explicit variable bounds hold at each iteration of the solution method, hence we can simplify the gradient row vector as:

$$\nabla_x \pi(x; \rho) = \nabla f(x) + \sum_{i=1}^{m} \rho_i \left(h_i(x) - b_i\right) \nabla h_i(x).$$

But the slope, $\nabla_\alpha \phi(0; x, d) = \nabla_x \pi(x; \rho) d$. Hence from the above we get the desired result for the slope. □

Using the merit function based on L_1 penalty is a similar exercise except for the fact that the penalty function is not differentiable everywhere. When explicit variable bounds hold by assumption (2.3), then the bound related terms can be removed from the penalty function definition as shown in the following lemma.

Lemma 5.2 (Merit Function based on Exact Penalty) *Let the merit function,* $\phi(\alpha; x, d)$ *be based on the exact penalty function of equation (12). Let an* m *dimensional vector* σ *be defined as:*

$$\sigma_i = \begin{cases} 1 & \textit{if } h_i(x) > b_i \\ -1 & \textit{if } h_i(x) < b_i \\ 1 & \textit{if } h_i(x) = b_i \ \wedge \ \nabla h_i(x) d > 0 \\ -1 & \textit{if } h_i(x) = b_i \ \wedge \ \nabla h_i(x) d < 0 \end{cases} \qquad \forall i = 1, \ldots, m.$$

Under the assumption (2.3) that explicit variable bounds hold for a solution method, if there exists $\alpha_{\max} > 0$ *such that* $l \le x + \alpha_{\max} d \le u$ *after step (12) of algorithm (1) then for the range* $\alpha \in [0, \alpha_{\max}]$, *the merit function can be simplified as:*

$$\phi(\alpha) = \begin{pmatrix} \sigma \\ 1 \end{pmatrix}^t \Lambda\, g(x)$$

and its slope is given by

$$\nabla_\alpha \phi(0) = \begin{pmatrix} \sigma \\ 1 \end{pmatrix}^t \Lambda\, J(x) d.$$

Proof of Merit Function based on Exact Penalty: Similar to the quadratic loss penalty case, by assumption (2.3), the explicit variable bounds, $l \le x \le u$, always hold, and $l \le x + \alpha_{\max} d \le u$ by the lemma. Hence, for the range, $\alpha \in [0, \alpha_{\max}]$, leaving out the penalty terms related to variable bounds in equation (12) we get:

$$\begin{aligned} \phi(\alpha; x, d) = \pi(x + \alpha d; \rho) &= f(x) + \sum_{i=1}^{m} \rho_i \,|\, h_i(x + \alpha d) - b_i \,| \\ &= f(x) + \sum_{i=1}^{m} \sigma_i \rho_i \left(h_i(x + \alpha d) - b_i \right). \end{aligned}$$

Hence the merit function can be expressed in matrix notation as shown in the lemma.

For slope calculations, we cannot use the scheme used earlier for quadratic loss penalty in ignoring the bound terms. The exact penalty function does not meet the criterion for a differentiable penalty function in equation (21) of Luenberger [Lue84, page 372]. If we use $p^+ = \max\{0, x\}$ for the bound terms, then the exact penalty is $\psi(s) = s$. But $\nabla\psi \neq 0$ when $s = 0$. The slope is always one. For quadratic loss penalty, the function $\psi(s) = s^2$ and the derivative at $s = 0$ is indeed zero as required.

Hence instead of gradient of the penalty function $\nabla_x \pi$, we only state the directional derivatives based on [Fle87, page 298]. For the upper and lower bounds we define n dimensional vectors μ and τ respectively as follows:

$$\mu_j = \begin{cases} 1 & \text{if } x_j > u_j \\ 0 & \text{if } x_j < u_j \\ 1 & \text{if } x_j = u_j \wedge d_j > 0 \\ 0 & \text{if } x_j = u_j \wedge d_j < 0 \end{cases} \qquad \forall j = 1, \ldots n,$$

$$\tau_j = \begin{cases} 1 & \text{if } x_j < l_j \\ 0 & \text{if } x_j > u_j \\ 1 & \text{if } x_j = l_j \wedge d_j < 0 \\ 0 & \text{if } x_j = l_j \wedge d_j > 0 \end{cases} \qquad \forall j = 1, \ldots n.$$

Then the directional derivative can be defined by:

$$\mathcal{D}\pi(x; \rho, \gamma, \beta\, ; d) = \nabla f(x) d + \sum_{i=1}^{m} \sigma_i \rho_i \nabla h_i(x) d + \sum_{j=1}^{n} \mu_j \gamma_j d_j + \sum_{j=1}^{n} \tau_j \beta_j d_j.$$

Since by the lemma conditions, if there is an $\alpha_{\max} > 0$ such that $l \leq x + \alpha_{\max} d \leq u$, then for any variable already at its upper bound, $j \in U$, the direction component $d_j \leq 0$ and similarly for a variable at its lower bound $j \in L$, its direction component $d_j \geq 0$. This together with the condition that $l \leq x \leq u$ by the assumption on explicit bounds implies μ_j and τ_j will always be zero for the solution methods we are discussing and the bound terms can be ignored from the directional derivative calculations. Hence the directional derivative can be written as:

$$\mathcal{D}\pi(x; \rho\, ; d) = \nabla f(x) d + \sum_{i=1}^{m} \sigma_i \rho_i \nabla h_i(x) d.$$

Since the slope of the merit function is given by $\nabla_\alpha \phi(0; x, d) = \mathcal{D}\pi(x; \rho\, ; d)$, writing the above in matrix notation, we get the desired result. □

From lemma (5.1) and lemma (5.2), it is clear that both the merit function and its slope at zero are sensitive to scaling. The merit function is more influenced by function scaling while the slope is impacted by variable scaling in addition because of the Jacobian term. From equation (7) we have the scaled problem Jacobian as $\bar{J}(\bar{x}) = R\ J(x)\ V$. The computations of slope for merit functions based on both types of penalties get simplified for Newton type methods where the constraint $\nabla h(x) d = b - h(x)$ is satisfied by the direction vector d.

For exact penalty function, as shown in Luenberger [Lue84, page 389], having $\rho_i \geq |y_i^*|$ ensures that the local minimum of the exact penalty is same as the original nonlinear problem in equation (1). Since the optimal duals are not known before the solution is obtained, the update rule suggested by Powell is used in step (13) of algorithm (1) with a slight modification. The factor 1.5 used in the adjustment differs from Powell's value of 1. It has been found to be more effective in our tests. Some similar scheme may be used for the quadratic penalty too.

For a scaled problem, since $\overline{\rho}_i \geq |\overline{y}_i^*|$, it can be shown easily that the relation between scaled and original penalties should mirror the relation between the scaled and original constraint duals that was proved in equivalence of solutions lemma (3.2). Hence, whenever scaling changes, we can use the constraint duals adjustment mechanism of corollary (3.1) for merit function penalties.

5.4 Derivatives by Differencing

Given the relation between scaled and original Jacobian in equation (7) and a similar one for Hessians in equation (8) the obvious value dependence on scaling exists. This value dependence is true whether derivatives were computed analytically to the same precision as the function evaluations or they were based on finite differencing. But in differencing, an additional source of error creeps in because of the approximations being used. We would like to investigate if scaling could mitigate these errors. We have [GL01] used differencing when analytic derivatives were not available or only first derivatives were available analytically. Our scheme is similar to a scheme that was used in [LPY95] with some additional safeguards.

For first derivatives, when no analytic derivatives are available, we allow both forward and central differences. The perturbation interval h is given by:

$$h = (1 + |x_i|)\, \delta_1$$

where $\delta_1 = 1.05 \times 10^{-8}$ is the default value. For the central differencing option, using e_i for column i of the identity matrix, I_n, we compute the derivative of a function $f : \mathfrak{R}^n \mapsto \mathfrak{R}$ according to:

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x + he_i) - f(x - he_i)}{2h}.$$

Hence central differencing requires $2n$ function evaluations for the gradient of a function ∇f. For forward differencing option we require only $n + 1$ function evaluations. The computation in this case is:

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x + he_i) - f(x)}{h}.$$

When first derivatives are available analytically, from interfaces of modeling systems like GAMS, AMPL, etc., we use forward or central differencing for second derivatives. The perturbation interval, h, is same as before and the first partials replace the function evaluations in the computations as shown for central differencing option below:

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \approx \frac{1}{2h}\left[\frac{\partial f(x+he_i)}{\partial x_j} - \frac{\partial f(x-he_i)}{\partial x_j}\right].$$

Similarly, the forward differencing computation is:

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \approx \frac{1}{h}\left[\frac{\partial f(x+he_i)}{\partial x_j} - \frac{\partial f(x)}{\partial x_j}\right].$$

When no analytic derivatives are available and the solution method needs second derivatives, then we compute larger perturbation intervals, h_1 and h_2, corresponding to the two variables i and j, as:

$$\begin{aligned} h_1 &= (1+|x_i|)\,\delta_2 \qquad \text{and} \\ h_2 &= (1+|x_j|)\,\delta_2. \end{aligned}$$

where $\delta_2 = 1.0 \times 10^{-4}$ is the default value. Since this depends on differenced first derivatives where already some precision is lost, we do not allow forward differencing option. In this case, the second partials are given by using central differencing the first partials:

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \approx \frac{1}{2h_1}\left[\frac{\partial f(x+h_1e_i)}{\partial x_j} - \frac{\partial f(x-h_1e_i)}{\partial x_j}\right]$$

and this can be computed as:

$$\begin{aligned} a &= f(x+h_1e_i+h_2e_j) + f(x-h_1e_i-h_2e_j), \\ b &= f(x-h_1e_i+h_2e_j) + f(x+h_1e_i-h_2e_j), \\ \frac{\partial^2 f}{\partial x_i \partial x_j} &\approx \frac{a-b}{4h_1h_2}. \end{aligned}$$

When $i = j$, the expression gets simplified and only three function evaluations instead of four are needed. Thus for $n^2/2$ terms in a function Hessian, we need roughly $2n^2$ function evaluations.

Since the perturbation intervals are based on variable values, variable scaling is important. Gill et al. [GMW81, pages 128 – 130] describe how each computation of a first derivative is subject to truncation error that is proportional to the second derivative and a condition error that is dependent on the function value and the first derivative that is being computed. In this regard we would like to investigate various schemes for computing the intervals that take function and derivative values also into account rather than just relying on the variable being scaled.

6 Goal and Suggestions

Suggestions made in this section need to be validated empirically. The correctness results only show us the mechanics of scaling and as we saw in section (5), it affects behavior of the nonlinear method significantly. Much more work needs to be done in quantifying these impacts and using them as a criterion in selection of scale factors similar to what we did for linear systems scaling [GL06]. We discuss goals, when to do scaling and how to do scaling in this section.

6.1 Goals?

Before we discuss scaling a nonlinear problem, we would like to reiterate a point we made in justifying our unrealistic assumption (4.1) of a perfect linear system. We have empirically observed that a good scaling of the linear system before a solve is always beneficial. Hence we strongly recommend *scaling the linear system independently before each solution* irrespective of whether scaling is used or not for the nonlinear problem itself. Even when the nonlinear problem is scaled, the complexities and requirements of the nonlinear problem are quite different from the linear system. In short, scaling the nonlinear problem is not a substitute for scaling the linear system and vice versa.

Based on the correctness results in section (3.3) we recommend a *dynamic scaling mechanism* that scales problem functions and variables in each iteration. Static scaling employed by many algorithms in practice is typically based on the Jacobian matrix at the start of the solution process. In a nonlinear problem, the Jacobian may change significantly as the solution progresses and hence the static scaling scheme may make the matters worse. We use the Jacobian matrix of the *original problem* at the current point to generate scale factors. This ensures that the scaled Jacobian entries are well balanced.

A second strategy we would like to explore is to use the Jacobian matrix of the original problem for function scaling and initial variable scales. We then adjust variable scale factors based on the Lagrangian Hessian of the *original problem*. As we have mentioned earlier after proving equivalence of scaled Lagrangian Hessian in lemma (3.3), the Hessian is almost independent of function scaling. Since the second derivatives are less likely to change fast, this may make the scaling more stable. The mechanisms we are suggesting in this section will work with either strategy.

6.2 Generating Scale Factors

We use machine base, typically $\beta = 16$, to determine scale factors as integer powers of β as suggested in [GL06]. This allows the scaled value to be generated by an adjustment of the exponent of the original value. Using unrestricted scale factors will lead to floating point operations while generating scaled values and hence introduce a new source of errors.

We first discuss how to determine the offsets a in the mapping

$$\overline{x} = \mathcal{T}(x) = V^{-1}x - a.$$

To compute the offsets, we determine $t \in \Re^n$ from the original problem information, mainly related to bounds. The offset is defined in terms of this vector as $a = V^{-1}t$ for the first scaling scheme. Since this information is not based on a specific scaling scheme, we would like to keep $t = Va$ *invariant* through all scaling schemes. Each iteration uses a different scaling scheme. In the following lemma we show that offsets can be ignored while updating primal variables when the invariance property holds.

Lemma 6.1 (Scale Adjustments for Primal Variables under Offset Invariance) *Let a solution of the scaled problem, $\Gamma\langle V, R\rangle$, be $(\overline{x}\ ;\overline{y}, \overline{w}, \overline{z})$ where the offset is defined by $a = V^{-1}t$ for some vector $t \in \Re^n$. Let $\Gamma\langle \tilde{V}, \tilde{R}\rangle$ be a new scaled problem associated with a new scaling scheme defined by $\tilde{\mathcal{T}}$ and $\tilde{\mathcal{G}}$ as mentioned earlier that also defines its offset by $\tilde{a} = \tilde{V}^{-1}t$. Then the primal variables, $\tilde{x}$ for the new scaled problem, $\Gamma\langle \tilde{V}, \tilde{R}\rangle$, can be obtained by ignoring the offsets as shown below:*

$$\tilde{x} = \tilde{V}^{-1}V\overline{x}$$

Proof of Scale Adjustments for Primal Variables under Offset Invariance: Using the invariance, based on adjustment of solutions corollary (3.1), we can write the change of primal variables as:

$$\begin{aligned} \tilde{x} &= \tilde{V}^{-1}V(\overline{x} + a) - \tilde{a} \\ &= \tilde{V}^{-1}V\overline{x} + \tilde{V}^{-1}Va - \tilde{a}. \end{aligned}$$

For the scaled problem, $\Gamma\langle V, R\rangle$, $Va = t$. Hence we get

$$\begin{aligned} \tilde{x} &= \tilde{V}^{-1}V\overline{x} + \tilde{V}^{-1}t - \tilde{a} \\ &= \tilde{V}^{-1}V\overline{x} + \tilde{a} - \tilde{a} \\ &= \tilde{V}^{-1}V\overline{x}. \end{aligned}$$

The second step follows from the assumption in lemma that the new scaled problem, $\Gamma\langle \tilde{V}, \tilde{R}\rangle$, also defines offsets by $\tilde{a} = \tilde{V}^{-1}t$. The updates for x and by the same token, the bounds l and u are simplified. □

Thus, by lemma (6.1), it suffices if the offsets for the first iteration scaling scheme are $a = V^{-1}t$ and ignored there after in all iterations. To find t, we suggest the following procedure:

- If the lower bound is not finite, $l_j = -\infty$, then it is ignored by setting $t_j = 0$.
- If the lower bound is finite but upper bound is not finite, $u_j = \infty$ then $t_j = l_j$.
- If both bounds are finite and $u_j - l_j > \beta^2$ then this is too big a range, ignore by letting $t_j = 0$.
- Otherwise both are finite bounds and $u_j - l_j \le \beta^2$ hence use the mid-point of the range as an offset, i.e., $t_j = (u_j + l_j)/2$.

Thus, the above offset scheme is based on Gill et al. [GMW81, page 274] with an additional safeguard. If the user has provided realistic ranges, then we want to keep the scaled variables with in $[-1, 1]$ range if possible. In practice, user often does not know what a realistic range is, especially for large models with thousands of constraints and variables.

The variable scale factors are based on the ranges alone in [GMW81, page 274]. In our case, they depend on the functions the variables appear in. At each iteration, new scale factors are generated from *the original problem Jacobian*, $J(x)$, that is an $(m + 1) \times n$ matrix and it is scaled using Gauss-Seidel iterative scheme suggested in [GL06].

6.3 Mechanics of Scaling

We discuss initializations related to scaling and dynamic scaling adjustments. We show that most of the steps are of time and space complexity $O(\max\{m, n\})$. Only generation of scale factors R and V by Gauss-Seidel iterations that is of time complexity $O(mn)$, and updating the approximate Hessian that is $O(n^2)$ are *relatively* expensive operations. The time requirements are of the same order as a matrix vector multiplication.

6.3.1 Initialization

The scale factors should be initialized *before the main iterations begin* using identity matrices and offsets defined in section (6.2). Following scaling related initializations need to be done after step (1) of algorithm (1):

$$\begin{aligned} R &\leftarrow I_{m+1}, \\ V &\leftarrow I_n, \\ x_0 &\leftarrow x_0 - t, \\ u &\leftarrow u - t \qquad \text{and} \\ l &\leftarrow l - t. \end{aligned}$$

The initializations are $O(m)$ or $O(n)$. The diagonal matrices R and V can be represented by integer vectors since the scale factors are integral powers of β. The offsets a are never stored explicitly. We only keep the float vector t and the offsets can be generated any time by $a = V^{-1}t$. Thus the space complexity is also $O(\max\{m, n\})$.

6.3.2 Scale Adjustments for Nonlinear Problem

In each iteration, immediately after step (5) of algorithm (1) the Jacobian matrix at the current point becomes available. Using the Jacobian, new scale factors R^+ and V^+ should be computed. By adjustment of primal variables lemma (6.1), since the initialized scaling scheme with scale factors of unity, ensures $a = V^{-1}t = I_n t = t$, new offsets a^+ can be ignored in updates of primal variables and variable bounds.

To reduce computations we introduce the following temporary entities:

$$\begin{aligned} c &\leftarrow \frac{r^+_{m+1}}{r_{m+1}} && \text{objective correction,} \\ \Delta R &\leftarrow (R^+)^{-1}R && \text{function corrections and} \\ \Delta V &\leftarrow (V^+)^{-1}V && \text{variable corrections.} \end{aligned}$$

Since the scale factors are integral powers of machine base β, divisions and multiplications are just integer subtractions and additions for the temporary entities. These operations are $O(\max\{m, n\})$. Space requirements are two integer vectors for the diagonal matrices ΔR and ΔV, hence $O(\max\{m, n\})$. In procedure shown in algorithm (2), we generate new scale factors and compute the corrections.

The current point (x, y, w, z) should be adjusted for the new scale factors. Based on adjustment of primal variables lemma (6.1), using the temporary entities defined above, we can write the adjustments to change of scaling for primal variables and the bounds as:

$$\begin{aligned} x &\leftarrow (\Delta V)\, x, \\ u &\leftarrow (\Delta V)\, u \qquad \text{and} \\ l &\leftarrow (\Delta V)\, l. \end{aligned}$$

Algorithm 2 Generate Scale Factors and Corrections

```
$(\Delta R, \Delta V) \leftarrow (R, V)$ { Save old values }
Gauss-Seidel Iterations, refer to [GL06] { Reuse $(R, V)$ for output}
$(R, V) \leftarrow (R^+, V^+)$
$c \leftarrow r_{m+1}/\Delta r_{m+1}$ { objective correction }
$\Delta R \leftarrow R^{-1}\Delta R$ { function corrections }
$\Delta V \leftarrow V^{-1}\Delta V$ { variable corrections }
```

The above step is an $O(n)$ operation.

In constraint duals we also include a dual for the objective function that always has unity as its value, $y_{m+1} = 1$. This is maintained as an invariant by the scaling adjustments. The adjustments for the constraint duals and bound duals based on adjustment of solutions corollary (3.1) are:

$$\begin{aligned} y &\leftarrow c\,(\Delta R)\,y, \\ w &\leftarrow c\,(\Delta V)^{-1}\,w \qquad \text{and} \\ z &\leftarrow c\,(\Delta V)^{-1}\,z. \end{aligned}$$

The above step is an $O(\max\{m, n\})$ operation.

We need to adjust the constraint right hand sides too. For this, we restrict ourselves to only the first m entries of the diagonal matrix ΔR. Like earlier, we use the notation, $\Delta R'$ to indicate a sub-matrix with the last $m+1$ row and column removed corresponding to the objective function. Since the right hand sides were multiplied by R' earlier and now they need to be multiplied by $(R^+)'$, the adjustment for right hand sides is reverse of the one employed for constraint duals.

$$b \leftarrow c\,(\Delta R')^{-1}\,b.$$

This is an $O(n)$ operation.

6.3.3 Adjustments for Merit Functions

For solution methods using merit functions based on penalty functions, the penalty multipliers need to be adjusted. As mentioned in section (5.3), the requirement that these weights be at least as large as the dual multipliers, translates to an update mechanism similar to constraint duals. Like the constraint duals, using weight of unity for the objective function weight, $\rho_{m+1} = 1$, we can write this as:

$$\rho \leftarrow c\,(\Delta R)\,\rho.$$

This is an $O(m)$ operation.

6.3.4 Updating Quasi-Newton Approximations

For solution methods that approximate the Lagrangian Hessian, previous values of primal variables x^-, previous Lagrangian gradient q^- and the approximate Hessian $\hat{L}$ need to be adjusted.

Previous Lagrangian gradient is computed in step (4) of algorithm (1) before scaling is performed because one needs values of gradients at the previous point and the new Jacobian has not yet been computed. But the new scale factors are available only after step (5). Hence the requirement for adjustment. Using Lagrangian gradient adjustment corollary (3.5) we can update the previous gradient in an $O(n)$ operation as:

$$q^- \leftarrow c\,(\Delta V)^{-1}\,q^-.$$

Similarly, previous values of primal variables are saved in step (27) of algorithm (1). Similar to the primal variables adjustment suggested in lemma (6.1), we can adjust them as:

$$x^- \leftarrow \quad (\Delta V)\,x^-.$$

As mentioned toward the end of section (5.2), quasi-Newton approximations in the limit will behave like true Lagrangian Hessian and hence have same sensitivity to scaling. Hence we use scale adjustments of corollary (3.2) for true Lagrangian Hessians for the approximations too that is an $O(n^2)$ operation.

$$\hat{L} \leftarrow c\,(\Delta V)^{-1}\,\hat{L}\,(\Delta V)^{-1}.$$

References

[CR72] A. R. Curtis and J. K. Reid. On the Automatic Scaling of Matrices for Gaussian Elimination. *IMA Journal of Applied Mathematics*, 10(1):118–124, 1972.

[Fle87] R. Fletcher. *Practical Methods of Optimization*. Series in Statistics. John Wiley & Sons Ltd., The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, England, 2nd edition, 1987.

[Gaj95] Ravindra S. Gajulapalli. *INTOPT: An interior point algorithm for large scale nonlinear optimization*. PhD thesis, University of Texas at Austin, 1995.

[GL01] Ravindra S. Gajulapalli and Leon S. Lasdon. Computational experience with a safeguarded barrier algorithm for sparse nonlinear programming. *Computational Optimization and Applications*, 19:107–120, 2001.

[GL06] Ravindra S. Gajulapalli and Leon S. Lasdon. Scaling sparse matrices for optimization algorithms. Technical Report 2006-08-5, Indian Institute of Management, Ahmedabad, Vastrapur, Ahmedabad - 380015, India, August 2006.

[GMW81] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, Inc., Orlando, Florida 32887, 1981. Reprinted in 1987.

[LPY95] Leon S. Lasdon, John Plummer, and Gang Yu. Primal-dual and primal interior point algorithms for general nonlinear programs. *ORSA Journal on Computing*, 7(3):321–332, 1995.

[Lue84] David G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, second edition, January 1984. Last reprinted in 1989. There is a newer second editon from Springer with ISBN 1402075936 dated 30 September 2003.

[NS95] Stephen Nash and Ariela Sofer. *Linear and Nonlinear Programming*. Science / Engineering / Math. McGraw-Hill, Inc., December 1995.

[Wat91] David S. Watkins. *Fundamentals of Matrix Computation*. John Wiley & Sons, Inc., New York, first edition, 1991.