Scaling Sparse Matrices for Optimization Algorithms

Ravindra S. Gajulapalli
Leon S. Lasdon

W.P. No. 2006-08-05
August 2006

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD – 380015
INDIA

SCALING SPARSE MATRICES FOR OPTIMIZATION ALGORITHMS

Ravindra S. Gajulapalli
Leon S. Lasdon

Abstract

To iteratively solve large scale optimization problems in various contexts like planning, operations, design etc, we[1] need to generate descent directions that are based on linear system solutions. Irrespective of the optimization algorithm or the solution method employed for the linear systems, ill conditioning introduced by problem characteristics or the algorithm or both need to be addressed. In [GL01] we used an intuitive heuristic approach in scaling linear systems that improved performance of a large scale interior point algorithm significantly. We saw a factor of 10^3 improvements in condition number estimates. In this paper, given our experience with optimization problems from a variety of application backgrounds like economics, finance, engineering, planning etc., we examine the theoretical basis for scaling while solving the linear systems. Our goal is to develop reasonably "good" scaling schemes with sound theoretical basis.

We introduce concepts and define "good" scaling schemes in section (1), as well as explain related work in this area. Scaling has been studied extensively and though there is a broad agreement on its importance, the same cannot be said about what constitutes good scaling. A theoretical framework to scale an $m \times n$ real matrix is established in section (2). We use the first order conditions associated with the Euclidean metric to develop iterative schemes in section (2.3) that approximate the solution in $O(mn)$ time for real matrices.

We discuss symmetry preserving scale factors for an $n \times n$ symmetric matrix in (3). The importance of symmetry preservation is discussed in section (3.1). An algorithm to directly compute symmetry preserving scale factors in $O(n^2)$ time based on Euclidean metric is presented in section (3.4).

We also suggest scaling schemes based on rectilinear norm in section (2.4). Though all $p-$norms are theoretically equivalent, the importance of outliers increases as p increases. For barrier methods, due to large diagonal corrections, we believe that the taxicab metric ($p = 1$) may be more appropriate. We

[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.

develop a linear programming model for it and look at a "reduced" dual that can be formulated as a minimum cost flow problem on networks. We are investigating algorithms to solve it in $O(mn)$ time that we require for an efficient scaling procedure. We hope that in future special structure of the "reduced" dual could be exploited to solve it quickly. The dual information can then be used to compute the required scale factors. We discuss Manhattan metric for symmetric matrices in section (3.5) and as in the case of real matrices, we are unable to propose an efficient computational scheme for this metric. We look at a linearized ideal penalty function that only uses deviations out of the desired range in section (2.5). If we could use such a metric to generate an efficient solution, then we would like to see impact of changing the range on the numerical behavior.

Contents

List of Algorithms

1 Why Scale?

In theory, it is easy to distinguished a singular matrix from a non-singular one. This is especially true for the small problems used for illustrating concepts in text-books that have typically fewer than four variables and one or two constraints. In practice, it is difficult to say if a matrix is singular or nearly singular in presence of ill-conditioning. Hilbert matrices family from [Wat91, page 123] is illustrative. An $n \times n$ symmetric positive definite matrix H_n in this family has its (i, j) entry is given by $1/(i + j - 1)$. If we try to solve the equation

$$H_n x = b_n,$$

on a computer where the component i of the vector b_n is

$$(b_n)_i = \sum_{j=1}^{n} \frac{1}{i + j - 1},$$

we start getting random values for $n = 6$ and higher. The correct answer for above linear system is $x = [1, 1, \ldots, 1]^t$. A Hilbert's matrix for $n = 4$ looks like:

$$H_4 = \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{bmatrix}.$$

Ill conditioning occurs because of two main reasons: finite precision representations of real numbers and floating point operations errors. Some matrices like the Hilbert's matrices are inherently ill-conditioned. Note, we can easily construct integer versions (all matrix entries integers) of Hilbert's matrices with the same pathological behavior. Watkins [Wat91, page 118] suggests that while solving a linear system $Ax = b$, if A and b are a accurate to about s decimal places and if the condition number of A, $\kappa(A) \approx 10^t$, where $t < s$, then the entries of computed solution are accurate to about $s - t$ decimal places.

In addition to errors introduced solving linear systems, some optimization methods like barrier and penalty methods lead to ill conditioning even in a well conditioned problem. For instance, at each iteration of an interior point method a symmetric system of equations has to be solved. These matrices become ill-conditioned due to large diagonal corrections as variables approach their bounds in final steps. Hence scaling becomes even more important to preserve quality of solution to the linear system. The solution of the linear systems is used in computing directions for primal and dual variables and poor quality solutions delay convergence. As the problem size becomes larger, empirically more errors related to smaller pivots have been observed in [Gaj95]. A small pivot leads to a very large multiplier

and these swamp out significant bits of precision from previous entries of the row. When multiples of this row are now added to other rows, other rows suffer the same fate. Net result is that all the subsequent rows start looking similar, and even a matrix that was not singular to start with appears singular.

The following example from [Wat91, page 100] has an exact solution $[1, 1]$, but the condition number $\kappa_2(A) = 1/\epsilon$ is very large. In the example, ϵ is a very small positive number $0 < \epsilon \ll 1$.

$$\begin{bmatrix} 1 & 0 \\ 0 & \epsilon \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ \epsilon \end{bmatrix}$$

Hence any small error in the representation, say the right hand side becoming $[1, 2\epsilon]$ leads to a large error in the computed solution. For the minor change, the solution becomes $[1, 2]$ instead of the correct $[1, 1]$. If we use scale factors of 1 and $1/\epsilon$ for the first and second row, this is a well conditioned problem. In summary, scaling may not help pathological matrices like the Hilbert family but in most practical cases, it can mitigate the effects of ill-conditioning.

1.1 "Good" Scaling

The objective is to scale an $m \times n$ matrix A such that all non-zeros are between $[\beta^{-1}, 1]$ where β is the machine base, usually 16 or some integral power of 2. In Gaussian elimination, the idea is to keep the lower triangular factor L and the upper triangular factor U small and by [Wat91, Theorem 2.5.5, page 115], ensure stable behavior. Due to the small pivot problem mentioned earlier, during factorization the pivots are carefully monitored to ensure that small pivots are avoided and growth in factors L and U are kept with in bounds.

Instead of floating point numbers, if the scaling factors are chosen as nearest integer power of the machine base, $\beta = 16$, then the scaling operation is just a change in the exponent and the mantissa does not change. Thus no introduction of new errors due to floating point operations in computing scaled values. By Bauer's theorem [Wat91, 128–130], this type of scaling assures that if the same pivoting sequence, as the one used before scaling, is employed during Gaussian elimination, the solution obtained is same including roundoff errors. We call scaling procedures that do not introduce new floating point errors as *non-intrusive*. In contrast, methods that use scales that are unrestricted, lead to additional errors influencing the algorithm behavior. Thus using integral powers of β as a scale factor is *non-intrusive*. The *objective of a non-intrusive scaling is to influence the pivoting sequence but no other side effects.*

Since each iteration of an optimization algorithm solves a linear system, the scaling procedure has to be efficient both in space and time. We call a procedure

that computes in $O(mn)$ time and with $O(n)$ space, the scales for a $m \times n$ real matrix, as *efficient* for our purposes.

1.2 Related Work

The approach used by Lasdon et al. in [LPY95] was to scale each row by the largest element and then the columns next. The first stage is an $O(mn^2)$ operation while the second stage was $O(mn)$ for a sparse matrix. In addition, to being slow, it also destroyed symmetry of the matrix since column and row weights were not the same. Secondly, since scaling factors were themselves floating point numbers, this introduced additional floating point arithmetic errors. The scaling algorithm's time efficiency was $O(n^3)$ because the matrices were stored by columns and hence row oriented access was costly. In [Gaj95], the scaling method preserved symmetry by using same scale factors for rows and columns. The scale factors were integral powers of β. Lastly, the approach used in [Gaj95] is $O(n^2)$. A three pass approach to identify the largest element in each row and column was used. Thus each pass reduced the row and column norms by factors calculated in the previous pass. A three pass mechanism was found to reduce condition number estimates by a factor of 10^3, refer to section (A) for a discussion on the estimators used. Curtis and Reid [CR72] claim that such pat heuristics are unsatisfactory.

Our current approach is similar to Curtis and Reid [CR72] where they use Hamming's least squares criterion that minimizes the distance of scaled terms from 1. We feel that this is more restrictive than using the center of the range. Thus the objective function we use differs in a correction term. Secondly, we propose a Gauss-Seidel iterative scheme to solve the least squares problem. As Curtis and Reid point out, *the resulting normal equations are singular but consistent and direct solution is complicated by the need to solve large symmetric positive definite systems*. They use a conjugate gradient method to solve the system. Curtis and Reid [CR72] do not address the issue of symmetry preserving scaling. This is essential if one is to use Cholesky or other symmetric solvers. We also formulate an alternative optimization model that is based on the Manhattan metric.

Rothblum and Zenios [RZ92] address a more general problem of finding row and column scale factors satisfying constraints on column and row products. Their solution is an iterative one for the resulting constrained optimization problem. We prefer the simpler unconstrained formulation because it is easier to solve. It is also not clear how one can come up with desirable goals for column and row products in our context. The authors mention that the problem of determining a scaling that results in prescribed row and column sums has been studied for the past few decades. The symmetric version of the scaling problem they attempt requires the row and column products to be equal.

2 Sparse Real Matrices

Let $x \in \Re^m$ be a vector of row scale factors. Later on, we will require x to be integral. Similarly, let $y \in \Re^n$ be the column scale factors. Let A be a sparse real $m \times n$ matrix and N be the associated incidence matrix. That is, N is an $m \times n$ matrix of ones and zeros. We denote elements of matrices A and N as a_{ij} and n_{ij} respectively. Thus:

$$N = \{n_{ij} = 1 \text{ if } a_{ij} \neq 0, \text{ otherwise } n_{ij} = 0\}.$$

Let N^i denote the ith row of the non-zero incidence matrix and N_j denote the jth column of the incidence matrix N. Thus the number of non-zeros in row i and column j can be found by:

$$(N^i)^t\mathbf{e} = \left| N^i \right| \text{ and } (N_j)^t\mathbf{e} = |N_j|$$

where **e** is a vector of all 1s appropriately dimensioned.

2.1 What is the Goal?

Let U be an $m \times m$ diagonal matrix and V be an $n \times n$ diagonal matrix such that the diagonal elements are given by:

$$\begin{aligned} U_{ii} &= \beta^{x_i} \qquad \forall\, i = 1, \ldots, m \qquad \text{and} \\ V_{jj} &= \beta^{y_j} \qquad \forall\, j = 1, \ldots, n. \end{aligned}$$

The objective of scaling is to obtain matrix $A' = UAV$ such that elements of A' are within the desirable range, namely, $[\beta^{-1}, 1]$. The following lemma trivially establishes the connection between this goal and the choice of scale factors.

Lemma 2.1 (Scale Factors Range) *The desired scale factors should ideally be such that for each non-zero in matrix A the following condition holds:*

$$x_i + y_j \in \left[-1 - \log_\beta |a_{ij}|,\ -\log_\beta |a_{ij}|\right] \text{ when } n_{ij} = 1.$$

Proof of Scale Factors Range: Post scaling we would like

$$\beta^{x_i} |a_{ij}| \beta^{y_j} \in [\beta^{-1}, 1] \text{ when } n_{ij} = 1.$$

If we take logarithms to the base β, the goal can be expressed as:

$$x_i + \log_\beta |a_{ij}| + y_j \in [-1, 0] \text{ when } n_{ij} = 1.$$

The necessary conclusion follows. □

We would like to use a penalty function that measures how effective are the scale factors are chosen. Ideally, if the scaled matrix entries are in the desired range $[\beta^{-1}, 1]$, they should incur no penalty. Outside the range, the penalty incurred is a rapidly increasing function of distance from the nearest end point. Or equivalently, if α represented the logarithm of a scaled non-zero entry of the matrix A, then we would want the penalty function $\psi(\alpha)$ to be zero in the range $[-1, 0]$. Outside the range it should be increasing. An example is shown in figure 1. An L_1 penalty

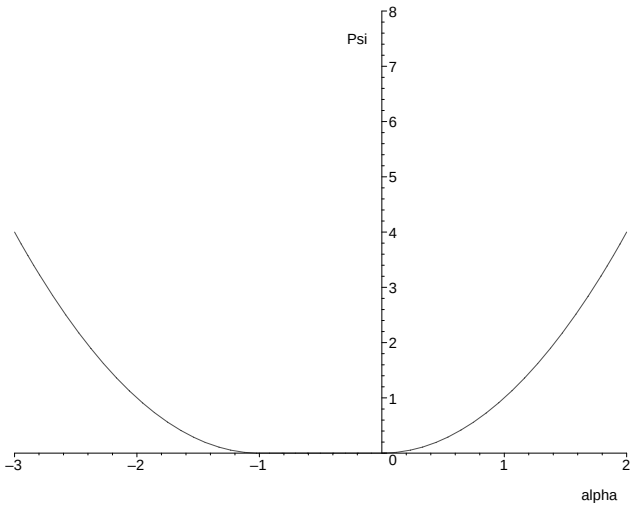

Figure 1: Penalty function for scaled matrix terms

function, of the form

$$\psi(\alpha) = \max\ \{0,\ \alpha,\ \gamma(-1-\alpha)\}$$

is a good candidate where γ is a balancing factor to prevent a few terms larger than 1 from dominating the scale factors. With diagonal corrections in barrier methods, this consideration becomes important. The penalty function, $\psi(\alpha)$, is convex but not differentiable.

2.2 Euclidean Metric

Given the difficulties with a non-smooth objective, Hamming's squared distance criterion is widely used as in Curtis and Reid [CR72], Rothblum and Zenios [RZ92] etc. We also use Euclidean distance from the center of the desired range as a smoother alternative in the following lemma.

Lemma 2.2 (The Euclidean Metric) *Let a $m \times n$ real matrix $\bar{A}$ be defined with the same sparsity pattern as matrix A. Let elements of $\bar{A}$ be the center of the range define in*

lemma (2.1), i.e.,

$$\begin{aligned} \bar{a}_{ij} &= -\log_\beta |a_{ij}| - \frac{1}{2} \qquad \textit{whenever } a_{ij} \neq 0 \textit{ and} \\ \bar{a}_{ij} &= 0 \qquad \textit{if } a_{ij} = 0. \end{aligned}$$

The L_2 norm based objective that minimizes the distance from the center of the range can be stated as:

$$\min \qquad \Pi_2 = \frac{1}{2} \sum_{ij:n_{ij}=1} (x_i + y_j - \bar{a}_{ij})^2$$

Proof of the Euclidean Metric: This is similar to Curtis and Reid [CR72] objective except that we are measuring the distance from the center of the range while they use the upper end. □

We define an $m \times m$ diagonal matrix R and an $n \times n$ diagonal matrix C such that

$$\begin{aligned} R_{ii} &= \left(N^i\right)^t \mathbf{e}, \\ C_{jj} &= \left(N_j\right)^t \mathbf{e}. \end{aligned} \tag{1}$$

Thus R_{ii} denotes count of non-zeros in row i and similarly C_{jj} gives count of non-zeros in column j. The following lemma defines a set of equations that can be used to iteratively calculate the scale factors.

Lemma 2.3 (Solutions for the Euclidean Criterion:) *The scale factors x and y satisfy:*

$$\begin{aligned} Rx + Ny &= \bar{A}\mathbf{e}, \\ Cy + N^t x &= \bar{A}^t \mathbf{e}. \end{aligned}$$

Proof of Solutions for the Euclidean Criterion: This is a standard unconstrained least squares problem and the solution is guaranteed to exist since the Hessian is positive definite. The normal equations are:

$$\begin{aligned} \frac{\partial \Pi_2}{\partial x_i} = 0 \implies \sum_{j \in N^i} (x_i + y_j - \bar{a}_{ij}) = 0 \qquad \forall i \text{ such that } \left|N^i\right| > 0, \\ \frac{\partial \Pi_2}{\partial y_j} = 0 \implies \sum_{i \in N_j} (x_i + y_j - \bar{a}_{ij}) = 0 \qquad \forall j \text{ such that } |N_j| > 0. \end{aligned}$$

Taking variables on left hand side and constants on right hand side we get:

$$\begin{aligned} \sum_{j \in N^i} x_i + \sum_{j \in N^i} y_j &= \sum_{j \in N^i} \bar{a}_{ij} \qquad \forall i \text{ such that } \left|N^i\right| > 0, \\ \sum_{i \in N_j} x_i + \sum_{i \in N_j} y_j &= \sum_{i \in N_j} \bar{a}_{ij} \qquad \forall j \text{ such that } |N_j| > 0. \end{aligned}$$

Using $\bar{A}_j$ to indicate column j of matrix $\bar{A}$ and $\bar{A}^i$ for row i, we can write the above as:

$$\begin{aligned} R_{ii}x_i + (N^i)^T y &= \mathbf{e}^T \bar{A}^i \qquad \forall i \text{ such that } \left| N^i \right| > 0, \\ (N_j)^T x + C_{jj} y_j &= \mathbf{e}^T \bar{A}_j \qquad \forall j \text{ such that } |N_j| > 0. \end{aligned}$$

The normal equations can be written in matrix form as:

$$\begin{aligned} Rx + Ny &= \bar{A}\mathbf{e}, \\ N^t x + Cy &= \bar{A}^t \mathbf{e}. \end{aligned}$$

The conclusion follows. □

Thus the row scales x, are based on a row constant term minus a term dependent on the column scales. The constant term represents the sum of non-zeros in a row divided by the row count - an "average" of the non-zeros in the rows. The variable term is average of the column scale factors that are incident for the row. Similar statement can be made about the column scale factors y.

2.3 Finding the Scale Factors

The following lemma gives the necessary conditions for convergence of the iterative scheme being suggested. Later in this section, we check if the conditions are met in general. We use an assumption on non-zero counts that is relaxed later on.

Lemma 2.4 (Gauss-Seidel Iterations) *If we assume that $R_{ii} > 0$, that is, at least one non-zero in a row and $C_{jj} > 0$ - at least one non-zero in a column, then the matrices R and C are invertible. The following iterative scheme to calculate scale factors is suggested by lemma (2.3)*

$$\begin{aligned} x_{k+1} &= R^{-1}\bar{A}\mathbf{e} - R^{-1}N y_k, \\ y_{k+1} &= C^{-1}\bar{A}^t\mathbf{e} - C^{-1}N^t x_{k+1} \end{aligned}$$

where k refers to the iteration number. Let the the matrix B be defined below:

$$B = \begin{bmatrix} I & \\ -C^{-1}N^t & I \end{bmatrix} \begin{bmatrix} 0 & -R^{-1}N \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -R^{-1}N \\ 0 & -C^{-1}N^t R^{-1}N \end{bmatrix}$$

The iterative scheme will converge if and only if matrix B has all eigenvalues less than one (absolute value) and the rate of convergence depends on the largest absolute eigenvalue.

Proof of Gauss-Seidel Iterations: Under the scheme, in each iteration, previous iteration values of the ys along with newly calculated xs are used. We can write

the normal equations in lemma (2.3), since R and C are invertible under the assumptions of the lemma, as:

$$\begin{aligned} x + R^{-1}Ny &= R^{-1}\bar{A}\mathbf{e}, \\ y + C^{-1}N^t x &= C^{-1}\bar{A}^t\mathbf{e}. \end{aligned}$$

This can be recast in the form $(I - L - U)z = b$ where the vector b represents the constant right hand sides, vector $z^t = (x^t,\ y^t)$ and L and U are lower and upper triangular matrices with null principal diagonals and I is an $m+n$-rowed identity matrix. The matrices L and U are defined by:

$$\begin{aligned} L &= \begin{bmatrix} 0 & 0 \\ -C^{-1}N^t & 0 \end{bmatrix}, \\ U &= \begin{bmatrix} 0 & -R^{-1}N \\ 0 & 0 \end{bmatrix}. \end{aligned}$$

In this form, suggested in [Kre85, pages 811-812], upper triangular U has non-zeros only in those positions where "old" values from previous iterations have to be used because the corresponding "new" ones are not available - in our case y scale factors. Similarly, the lower triangular matrix L has non-zeros in positions where "new" approximations are already available -in our case current iteration x values. Gauss-Seidel formula is:

$$(I - L)z_{k+1} = b + Uz_k.$$

The convergence of the method depends on the eigenvalues of matrix $B = (I - L)^{-1}U$. The matrix $I - L$ and its inverse for the scale factors are given by:

$$\begin{aligned} I - L &= \begin{bmatrix} I & 0 \\ C^{-1}N^t & I \end{bmatrix}, \\ (I - L)^{-1} &= \begin{bmatrix} I & 0 \\ -C^{-1}N^t & I \end{bmatrix}. \end{aligned}$$

Inverse can be readily verified by forming the product. Thus forming the product $B = (I - L)^{-1}U$ gives the desired result. The necessary condition follows from [Kre85, page 811] where it is stated that if matrix B has all eigenvalues with absolute value less than one then it will converge from any starting point. The convergence rate depends on the spectral radius (largest eigenvalue in magnitude). □

In the following lemma, eigenvalues of the matrix B used in Gauss-Seidel iterations are shown to follow the necessary condition.

Lemma 2.5 (Eigenvalues of Iteration Matrix) *The iteration matrix* B*, defined in lemma (2.4), has eigenvalues with absolute value less than 1.*

Proof for Eigenvalues of Iteration Matrix: The eigenvalues of a triangular matrix are given by the diagonal entries. Hence both the matrices involved in forming B have eigenvalues less than or equal to one. The first matrix

$$\begin{bmatrix} I & \\ -C^{-1}N^t & I \end{bmatrix}$$

has all eigenvalues of 1. The second matrix

$$\begin{bmatrix} 0 & -R^{-1}N \\ 0 & 0 \end{bmatrix}$$

has eigenvalues of 0. Let the product of eigenvalues of a square matrix C be denoted by $\phi(C)$. From [GMW81, page 24] the matrix product CD, where C and D are arbitrary square matrices, satisfy the property:

$$\phi(CD) = \phi(C)\phi(D)$$

We conclude that all eigenvalues are less than 1.

Alternatively, we can examine non-zero eigenvalues of matrix B and come to a similar conclusion. For any vector, $z \in \Re^{m+n}$, it can be partitioned as $z^t = (x^t, y^t)$. Hence if vector z was an eigenvector of matrix B associated with a non-zero real eigenvalue $\lambda \neq 0$, then it would satisfy $Bz = \lambda z$. In other words,

$$Bz = \begin{bmatrix} -R^{-1}Ny \\ -C^{-1}N^tR^{-1}Ny \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \end{bmatrix}.$$

Thus the x part of the z vector has to be equal to $-1/\lambda R^{-1}Ny$. The y part indicates that vector y and eigenvalue λ also solve the eigenvalue equation for the matrix $-C^{-1}N^tR^{-1}N$. The matrix N is an $m \times n$ matrix of 1s and 0s. The product $N^tR^{-1}N$ is symmetric and positive semidefinite. For any $n-$ dimensional vector y, using $s = Ny$,

$$y^tN^tR^{-1}Ny = s^tR^{-1}s = \sum_{i=1}^{m} R_{ii}s_i^2 \geq 0.$$

If N was non-singular, then the inequality holds strictly and the matrix is positive definite, but in practice it may not be. An extreme example would be a dense matrix that has ones in all rows and columns. Because the product $N^tR^{-1}N$ is symmetric, it has n real eigenvalues, not necessarily distinct, but the eigenvectors are distinct and form an orthonormal basis. The matrix C^{-1} is positive definite with it eigenvalues given by the diagonal elements that are all smaller than unity. Thus we see that the matrix $-C^{-1}N^tR^{-1}N$ is a product of a positive definite and a positive semidefinite system but it itself is not symmetric. And this also leads to the conclusion of the bounded eigenvalues. □

The setup procedure shown in algorithm (1) initializes the constants and right hand sides used in the iterations during a single pass on the A matrix. Hence it is efficient.

For the iterative scheme to work, we had insisted that there be at least one non-zero in each row and column of A. While it seems reasonable to insist that a matrix not be structurally singular, this requirement can be relaxed. Often, modeling languages like GAMS, AMPL have conditional operators and in a large optimization model, it may happen that an empty row is generated because of these operators. Similarly, before a solver is called, some variables may be fixed for some solve statements as the modelers try various scenarios. This leads to columns with zero counts. Hence in the following lemma we relax the requirement.

Lemma 2.6 (Structurally Singular Matrix) *The requirement in lemma (2.4) that the matrices R and C be invertible can be relaxed by ignoring zero count rows and columns during the inversion operations.*

Proof of Structurally Singular Matrix: Any value assigned to the scale factors associated such columns and rows does not matter while solving the linear systems. The corresponding normal equations are consistent and shown below:

$$\begin{aligned} 0x_i + \mathbf{0}^t y &= 0 \qquad \forall\, i \text{ where } \left| N^i \right| = 0, \\ \mathbf{0}^t x + 0y_j &= 0 \qquad \forall\, j \text{ where } |N_j| = 0. \end{aligned}$$

Thus if we skip the associated rows or columns during the divisions performed for R^{-1} or C^{-1} operation, the effect is as if we are working on a reduced model without these rows and columns. □

Algorithm 1 Setup for Gauss-Seidel Iterations

Require: $n > 0 \wedge m > 0$

$R \leftarrow 0$ { Non-zero row counts }
$C \leftarrow 0$ { Non-zero column counts}
$a \leftarrow 0$ { Row right hand sides }
$b \leftarrow 0$ { Column right hand sides}
for all non-zeros (i, j) in matrix A **do**
  $t \leftarrow \left(-\log_\beta |A_{ij}| - 1/2\right)$
  $R_{ii} \leftarrow R_{ii} + 1$
  $C_{jj} \leftarrow C_{jj} + 1$
  $a_i \leftarrow a_i + t$
  $b_j \leftarrow b_j + t$
end for
{Skip zero count columns and rows from lemma (2.6) during inversion}
$a \leftarrow R^{-1}a$
$b \leftarrow C^{-1}b$
return a, b, R, C

The Gauss-Seidel iterations are shown in algorithm (2). As can be seen, each iteration scans the A matrix twice for non-zero incidence information only. It is tempting to combine the two scans to one. This can be done by saving previous x values and using them to update y instead of the current x. This method is the Jacobi iteration and the values are updated according to:

$$\begin{aligned} x_{k+1} &= R^{-1}\bar{A}\mathbf{e} - R^{-1}Ny_k, \\ y_{k+1} &= C^{-1}\bar{A}^t\mathbf{e} - C^{-1}N^t x_k \end{aligned}$$

As mentioned in [Kre85, page 813], this method of simultaneous corrections instead of successive corrections is mainly of theoretical interest.

Algorithm 2 Scale Factors using Gauss-Seidel Iterations

Require: $k > 0$ { Maximum iterations allowed }
1: $y \leftarrow 0$ { Only old y needed }
2: $l \leftarrow 0$ { Iteration count }
3: **repeat**
4: $\quad l \leftarrow l + 1$
5: $\quad t \leftarrow a$ { Compute row factors }
6: $\quad$ **for all** non-zeros (i, j) in matrix A **do**
7: $\quad\quad t_i \leftarrow t_i - y_j/R_{ii}$
8: $\quad$ **end for**
9: $\quad x \leftarrow \text{Round}(t)$
10: $\quad t \leftarrow b$ { Compute column factors }
11: $\quad$ **for all** non-zeros (i, j) in matrix A **do**
12: $\quad\quad t_j \leftarrow t_j - x_i/C_{jj}$
13: $\quad$ **end for**
14: $\quad y \leftarrow \text{Round}(t)$
15: **until** $l = k$ or convergence.
16: **return** integer vectors x, y

We conclude with an observation about the efficiency of the proposed scheme. The set up cost is dominated by the sweep across the matrix A and hence is $O(|N|)$ while the actual scale factors require $2k$ sweeps across the incidence structure of A. Hence, its cost is $O(k \times |N|)$. In the completely dense matrix case, it reduces to $O(kmn)$. In terms of space we need:

- Two real vectors a, b for row and column right hand sides.
- Two integer vectors R, C for row and column counts.
- One real vector t to store temporary values of x or y before rounding off. Hence t is of dimension $\max(m, n)$.

- Two integer vectors x, y to return the result.

Without loss of generality, if we assume $n > m$, the space efficiency is $O(n)$.

The rounding off to produce integer scale factors is deferred. The computation of right hand sides during set up is done using full precision available. Row factors are accumulated using real arithmetic in real vector t. Just before x values are needed in computing y, we round off the results to integers. Similarly, column factors are accumulated using real precision. This allows *iterations to adapt to rounding errors*. We feel that this produces better integer approximations than computing everything in real precision and rounding off end results.

We feel that the limit on iterations, k, can be set to three based on our experience in [GL01] but empirical work is needed to come up with appropriate values for the new scaling scheme.

We conclude section (2.3) with a pathological example from [Gaj95, page 67]. We have been using such examples as "smelling salts" whenever we get carried away by our ideas. The example illustrates how the Gauss-Seidel iterations work and it is seen that it converges in two iterations. Like the proverbial successful operation, but a dead patient, the scaled matrix seems to be worse than what we started with.

Example 2.1 (Pathological Square Matrix) *In the following, assume machine base $\beta = 10$. This will allow human readable iteration output. Given the following "almost" symmetric matrix*

$$A = \begin{bmatrix} 1 & 10^{10} & 10^{20} \\ 10^{10} & 10^{30} & 10^{50} \\ 10^{20} & 10^{40} & 10^{80} \end{bmatrix},$$

what are the appropriate column and row scale factors?

Solution of Pathological Square Matrix: The incidence matrix, N, is a 3×3 matrix of all ones since this is a completely dense matrix. We execute steps of the setup algorithm (1) and get the following output:

$$\begin{array}{rcll} R & = & \text{Diag}(3,3,3) & \text{row non-zero count} \\ C & = & \text{Diag}(3,3,3) & \text{column non-zero count} \\ a^t & = & (-31.5, -91.5, -141.5) & \text{row right hand side} \\ b^t & = & (-31.5, -81.5, -151.5) & \text{column right hand side.} \end{array}$$

The output can be verified easily by stepping through each non-zero entry (i, j) and incrementing appropriate counts and adding to right hand sides. Note, this is based on a single scan of the matrix A. Next we execute steps in algorithm (2) to compute scale factors.

Iteration 1, Pass 1: The temporary vector t is assigned row right hand sides in $t \leftarrow a$. Vector y is zero before the start of this iteration. No change in t values from the loop based on old y values. Hence rounding off t gives row scale factors $x^t = (-32, -92, -142)$.

Iteration 1, Pass 2: Temporary vector t is assigned column right hand sides. The loop scanning through A matrix non-zeros adjusts t values based on the new x values and leaves $t^T = (57.167, 7.167, -62.83)$. After rounding, the column scale factors are $y^t = (57, 7, -63)$.

Iteration 2, Pass 1: The temporary vector t is assigned row right hand sides and the steps are repeated. The main difference from first iteration being the non-zero values for column scale factors y. The $t^T = (-31.8333, -91.8333, -141.833)$ and hence row scale factors remain unchanged at $x^t = (-32, -92, -142)$. This convergence implies that there will be no change in y values either and hence we can stop.

Using scale factors $x^t = (-32, -92, -142)$ for rows and $y^t = (57, 7, -63)$ for columns, the scaled matrix looks like:

$$A' = XAY = \begin{bmatrix} 10^{25} & 10^{-15} & 10^{-75} \\ 10^{-25} & 10^{-55} & 10^{-105} \\ 10^{-65} & 10^{-95} & 10^{-125} \end{bmatrix}.$$

If we use condition number estimates suggested in section (A), it seems to have deteriorated with the scaled matrix. □

In defense of the iteration scheme, no scaling can help pathological instances like this or Hilbert family matrices. Secondly, for any scaling scheme, given sufficient time, one can come up with pathological instances where the scaling will not do well.

2.4 Manhattan Metric

Theoretically all p−norms are equivalent. Two vector norms $\| \quad \|_{(1)}$ and $\| \quad \|_{(2)}$ are equivalent if there exist positive scalars α and δ such that

$$\alpha \| x \|_{(1)} \leq \| x \|_{(2)} \leq \delta \| x \|_{(1)}.$$

In practice this does not translate to identical results. For example, in a curve fitting exercise, when the integer p increases, the impact of outliers increases. In the limit, $p = \infty$, large elements determine the curve.

In the heuristic used in [GL01], we were implicitly using an $\| \quad \|_\infty$ norm in choosing the largest column and row elements. While working with penalty and barrier algorithms, this can be a disadvantage. For instance, in barrier methods,

as variables approach the bounds, we have observed diagonal corrections become larger than 10^{20}. *Allowing these terms to dominate the scale factors is not advisable.* In these situations $\| \quad \|_1$ may be a more appropriate choice. Traditionally Euclidean norm is more popular in scaling literature because of differentiability. Curtis and Reid [CR72] use Hamming's least squares criterion in that manner. The following lemma suggests a linear programming formulation for the Manhattan metric.

Lemma 2.7 (Manhattan Metric) *Let a $m \times n$ real matrix $\bar{A}$, with sparsity pattern of matrix A, be defined as in lemma (2.2). The L_1 norm based objective that minimizes the distance from the center of the range can be stated as:*

$$\min \qquad \Pi_1 = \sum_{ij:n_{ij}=1} z_{ij},$$

subject to constraints:

$$\begin{aligned} -z_{ij} + x_i + y_j &\leq \bar{a}_{ij} \qquad \forall\, i,j \text{ such that } n_{ij} = 1, \\ -z_{ij} - x_i - y_j &\leq -\bar{a}_{ij} \qquad \forall\, i,j \text{ such that } n_{ij} = 1. \end{aligned}$$

Proof of Manhattan Metric: Similar to the L_2 objective, we can state the goal of minimizing the distance from the center of the desired range in lemma (2.1) as:

$$\Pi_1 = \sum_{ij:n_{ij}=1} |\, x_i + y_j - \bar{a}_{ij}\,| = \sum_{ij:n_{ij}=1} \max\{x_i + y_j - \bar{a}_{ij},\ \bar{a}_{ij} - x_i - y_j\}.$$

Defining variables z_{ij} to be the result of the max operator, we get the constraints

$$\begin{aligned} z_{ij} &\geq -\bar{a}_{ij} + x_i + y_j \qquad \forall\, i,j \text{ such that } n_{ij} = 1, \\ z_{ij} &\geq \bar{a}_{ij} - x_i - y_j \qquad \forall\, i,j \text{ such that } n_{ij} = 1. \end{aligned}$$

Rearranging the terms by moving variables to the left hand side, constants to the right hand side and converting inequalities to less than form gives the desired result. □

Both metrics considered so far, Euclidean and Manhattan metrics, measure deviations from the center of the desired range given in lemma (2.1). They are shown in figure (2), with the thicker line representing the Euclidean metric.

Since, an $m \times n$ matrix A has $|N|$ non-zeros, the linear programming formulation has $2 \times |N|$ constraints, $|N|$ variables for the penalty terms and $m+n$ variables for the row and column scale factors. The following lemma yields a smaller model involving $|N|$ variables, $m+n$ explicit equality constraints by using duality.

Lemma 2.8 (Manhattan Dual) *Dual multipliers of the following linear program solve the primal problem in lemma (2.7).*

$$\max \qquad \bar{\Pi}_1^D = \sum_{ij:n_{ij}=1} \bar{a}_{ij} u_{ij}.$$

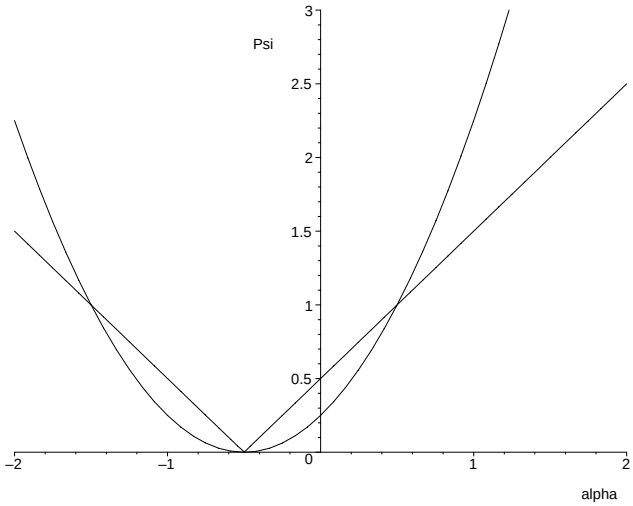

Figure 2: Euclidean and Manhattan metric penalty function for scaled matrix terms

Subject to constraints:

$$\begin{aligned}
\sum_{j \in N^i} u_{ij} &= \frac{R_{ii}}{2} \qquad \forall\, i = 1, \ldots, m, \\
\sum_{i \in N_j} u_{ij} &= \frac{C_{jj}}{2} \qquad \forall\, j = 1, \ldots, n, \\
0 \leq u_{ij} &\leq 1 \qquad \forall\, i, j \textit{ such that } n_{ij} = 1.
\end{aligned}$$

Proof of Manhattan Dual: We can convert the linear program in lemma (2.7) to canonical form by writing the objective with sign reversed as:

$$\max \qquad -\Pi_1 = -\sum_{ij:n_{ij}=1} z_{ij}.$$

Associating non-negative dual multipliers u_{ij} and v_{ij} with the first and second set of constraints respectively, we can write the dual linear program. Note, all the variables in the primal program are unrestricted. The dual of the canonical form is:

$$\min \qquad -\Pi_1^D = \sum_{ij:n_{ij}=1} \bar{a}_{ij} u_{ij} - \sum_{ij:n_{ij}=1} \bar{a}_{ij} v_{ij}.$$

subject to constraints:

$$\begin{aligned}
-u_{ij} - v_{ij} &= -1 \qquad \leftarrow z_{ij} \qquad \forall\, i, j \text{ such that } n_{ij} = 1, \\
\sum_{j \in N^i} u_{ij} - \sum_{j \in N^i} v_{ij} &= 0 \qquad \leftarrow x_i \qquad \forall\, i = 1, \ldots, m \text{ and} \\
\sum_{i \in N_j} u_{ij} - \sum_{i \in N_j} v_{ij} &= 0 \qquad \leftarrow y_j \qquad \forall\, j = 1, \ldots, n.
\end{aligned}$$

Using the substitution $v_{ij} = 1 - u_{ij}$, we can re-write the dual objective as:

$$\min \qquad -\Pi_1^D = 2 \sum_{ij:n_{ij}=1} \bar{a}_{ij} u_{ij} - \sum_{ij:n_{ij}=1} \bar{a}_{ij}.$$

And the constraints can be written as:

$$\begin{aligned} \sum_{j \in N^i} u_{ij} &= \frac{R_{ii}}{2} \qquad \forall\, i = 1, \ldots, m, \\ \sum_{i \in N_j} u_{ij} &= \frac{C_{jj}}{2} \qquad \forall\, j = 1, \ldots, n, \\ 0 \;\leq\; u_{ij} \;&\leq\; 1 \qquad \forall\, i, j \text{ such that } n_{ij} = 1. \end{aligned}$$

Rewriting the objective as a maximization and dropping the constant additive term as well as the constant 2 we get the objective stated in this lemma. We can construct solution to the dual of the original problem from the modified dual solution by using $v_{ij} = 1 - u_{ij}$. It is trivial to verify that the reconstructed solution will meet optimality criteria for the original dual. The multipliers associated with the dual constraints are the primal variables, namely, scale factors. □

Lemma (2.8) gives a simple linear programming problem with a special structure. If we use the substitution $\bar{u}_{ij} = 2u_{ij}$ and multiply each constraint by 2, it becomes a minimum cost flow problem on an acyclic network. It can be viewed as a capacitated transportation problem with a capacity of 2 units on each arc. Flows are represented by variables $\bar{u}_{ij}$ while $\bar{a}_{ij}$ denote the costs associated with the arcs. The supply nodes $\{1, 2, \ldots, m\}$ correspond to rows of A matrix and the demand nodes $\{1, 2, \ldots, n\}$ correspond to columns of A matrix while the arcs correspond to non-zeros in the matrix. The supplies at each node are half the outgoing capacity available at the supply node. Similarly, the demands at each node are half the incoming capacity. Solving it like a generic minimum cost flow problem will not meet our $O(mn)$ efficiency requirement mentioned in section (1.1). In practice, we would like any algorithm used for scaling to require not more than three to four scans of matrix A in computing scale factors.

2.5 Ideal Metric?

Both, Euclidean and Manhattan, metrics do not resemble the ideal penalty shown in figure (1). In the following lemma we modify the model used for Manhattan metric to reflect a linearized version of the ideal penalty that is shown in figure (3). The idea is to minimize deviations from the end points of the desired range but ignore anything in between. With this type of measure, we can explore sensitivity of linear system solutions to the desired range itself. For instance, what if we change the desired range from $[\beta^{-1}, 1]$ to $[\beta^{-2}, \beta]$? Secondly, if a matrix entries are all within the desired range, both Euclidean and Manhattan metrics measure

deviations from a single point and hence would compute scale factors that are not one. We would prefer to *minimize interventions like scaling only when absolutely necessary*.

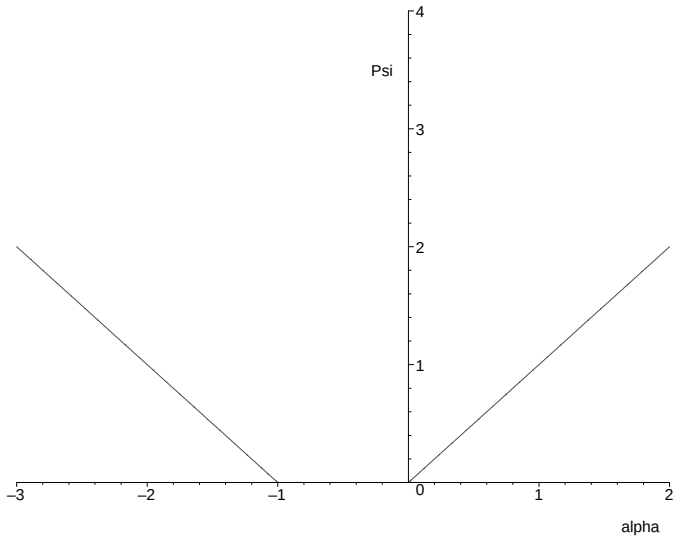

Figure 3: Linearized Penalty function for scaled matrix terms

Lemma 2.9 (Linearized Ideal Metric) *Let an* $m \times n$ *real matrix* $\bar{A}$*, with sparsity pattern of matrix* A*, be defined same as in lemma (2.2). The objective that minimizes the distance from the end points of the range in lemma (2.1) can be stated as:*

$$\min \qquad \Pi_P = \sum_{ij:n_{ij}=1} z_{ij},$$

subject to constraints:

$$\begin{aligned} -z_{ij} + x_i + y_j &\leq \bar{a}_{ij} + \frac{1}{2} \qquad \forall\, i,j \text{ such that } n_{ij} = 1, \\ -z_{ij} - x_i - y_j &\leq -\bar{a}_{ij} + \frac{1}{2} \qquad \forall\, i,j \text{ such that } n_{ij} = 1, \\ z_{ij} &\geq 0 \qquad \forall\, i,j \text{ such that } n_{ij} = 1. \end{aligned}$$

Proof of Linearized Ideal Metric: This is very similar to the model used for Manhattan metric in lemma (2.7) except for non-negativity conditions on z_{ij} and a constant term $1/2$ on the right hand sides of the constraints. From lemma (2.1) we should penalize only when either of the following conditions hold for an associated non-zero in matrix A.

$$\begin{aligned} x_i + y_j &< -1 - \log_\beta |a_{ij}| \text{ or} \\ x_i + y_j &> -\log_\beta |a_{ij}|. \end{aligned}$$

From the definition in lemma (2.2), we have, for non-zeros in A matrix:

$$\log_\beta |a_{ij}| = -\left(\bar{a}_{ij} + \frac{1}{2}\right).$$

Hence we want to penalize deviations if either of the following conditions hold:

$$\begin{aligned} x_i + y_j &< \bar{a}_{ij} - \frac{1}{2} \qquad \text{or} \\ x_i + y_j &> \bar{a}_{ij} + \frac{1}{2}. \end{aligned}$$

This is equivalent to saying, the penalty, z_{ij}, should be defined as:

$$z_{ij} = \max\left\{0,\ \bar{a}_{ij} - \frac{1}{2} - x_i - y_j,\ x_i + y_j - \bar{a}_{ij} - \frac{1}{2}\right\} \text{ when } n_{ij} = 1.$$

Replacing the max operator with three inequalities for each non-zero in matrix A, we get:

$$\begin{aligned} z_{ij} &\geq \bar{a}_{ij} - \frac{1}{2} - x_i - y_j \\ z_{ij} &\geq x_i + y_j - \bar{a}_{ij} - \frac{1}{2} \\ z_{ij} &\geq 0. \end{aligned}$$

Multiplying the first two inequalities by -1 converts them to the desired form. □

We conclude the discussion on linearized ideal metric by discussing the dual. Though we would have preferred to use the ideal, the dual is not easily reducible because of the non-negativity conditions associated with the variables z_{ij}. This was not necessary when we were penalizing deviations from the midpoint of the range.

$$\max \qquad \Pi_P^D = \sum_{ij:n_{ij}=1} \left(\bar{a}_{ij} + \frac{1}{2}\right) u_{ij} - \sum_{ij:n_{ij}=1} \left(\bar{a}_{ij} - \frac{1}{2}\right) v_{ij}.$$

subject to constraints:

$$\begin{aligned} -u_{ij} - v_{ij} &\geq -1 \qquad \leftarrow z_{ij} \qquad \forall\, i,j \text{ such that } n_{ij} = 1, \\ \sum_{j \in N^i} u_{ij} - \sum_{j \in N^i} v_{ij} &= 0 \qquad \leftarrow x_i \qquad \forall\, i = 1,\ldots,m \text{ and} \\ \sum_{i \in N_j} u_{ij} - \sum_{i \in N_j} v_{ij} &= 0 \qquad \leftarrow y_j \qquad \forall\, j = 1,\ldots,n. \end{aligned}$$

The substitution $v_{ij} = 1 - u_{ij}$ cannot be used now since the constraint associated with z_{ij} has become $u_{ij} + v_{ij} \leq 1$.

3 Sparse Symmetric Matrices

Let A be an $n \times n$ symmetric matrix and let $x \in \Re^n$ be an n-dimensional vector that represents the scale factors. Let U represent an $n \times n$ diagonal matrix such that:

$$U_{ii} = \beta^{x_i} \qquad \forall\, i = 1, \ldots, n.$$

Similar to $m \times n$ real matrices, the objective of scaling is to obtain matrix $A' = UAU$ such that elements of A' are within the desirable range, namely, $[\beta^{-1}, 1]$.

3.1 Why Preserve Symmetry?

Many authors like Rothblum and Zenios [RZ92] define symmetric scaling as generating A' by using $A' = UAU^{-1}$. It is easy to see that this does not preserve symmetry and leaves diagonal elements untouched. For instance, the element in row i, column j of the scaled matrix under their scheme would like:

$$a'_{ij} = \beta^{x_i} a_{ij} \beta^{-x_j} = \beta^{x_i - x_j} a_{ij}$$

Hence $a'_{ij} \neq a'_{ji}$ unless $x_i = x_j$ for off diagonal elements where $i \neq j$. For diagonal elements, the terms $x_i - x_i$ become zero and hence they remain untouched.

While solving linear systems that are symmetric, one would not like to destroy the symmetry because it precludes usage of Cholesky decomposition or symmetric indefinite system solvers. For instance, in each iteration of interior point method in [GL01], we solve the resulting linear system using symmetric indefinite system solvers. We prefer to retain symmetry due to the following reasons:

Space efficiency: Symmetric solvers use half the space required for storing the input matrix as well as less space for factors generated. For example, for LU factorization one needs store both L and U while Cholesky factorization, $A = GG^T$, requires space for only one factor G.

Time efficiency: Symmetric solvers take advantage of the symmetry in pivoting, element access etc. For instance, only half the non-zeros need to be examined for Cholesky decomposition when compared to the asymmetric case. This also may make the elementary row and column operations during factorization more efficient.

Numerical stability: Symmetric methods are more stable. For instance, Cholesky decomposition is numerically stable, whatever be the pivoting sequence.

When the general solver used in [LPY95] was replaced by a symmetric solver in [GL01], we saw significant improvements along all the above dimensions.

3.2 Goal for Symmetric Matrices

For symmetric matrices, without loss of generality we assume that *only the lower half is stored* for space efficiency. We also denote incidence of non-zeros with two incidence matrices. We define the incidence matrix N *to contain only sub-diagonal non-zeros* and a separate incidence matrix D for the non-zeros on the diagonal. Hence N is defined by:

$$N = \{n_{ij} = 1 \text{ if } i > j \text{ and } a_{ij} \neq 0, \text{ otherwise } n_{ij} = 0\}.$$

Similarly the diagonal non-zeros are indicated by:

$$D = \{d_{ii} = 1 \text{ if } a_{ii} \neq 0, \text{ otherwise } d_{ii} = 0\}.$$

The reason for breaking the incidence matrix in two parts becomes apparent once we examine desired ranges for the scale factors and look at the normal equations associated with the objective based on Euclidean metric. Let N_j be the jth column of the incidence matrix N. Thus the number of *sub-diagonal* non-zeros in column j can be found by:

$$(N_j)^t \mathbf{e} = |N_j|$$

where **e** is the vector of all 1s.

The objective of scaling is to obtain matrix $A' = UAU$ such that elements of A' are within the desirable range, namely, $[\beta^{-1}, 1]$. The following lemma, similar to lemma (2.1), connects the goal with the choice of scale factors. Note the diagonal non-zero elements impose a different requirement.

Lemma 3.1 (Symmetric Scale Factors Range) *The desired scale factors should ideally be such that for each* off-diagonal non-zero *in symmetric matrix* A *the following condition holds:*

$$x_i + x_j \in \left[-1 - \log_\beta |a_{ij}|, \ -\log_\beta |a_{ij}|\right] \ \textit{when } n_{ij} = 1.$$

And for each diagonal non-zero *in* A*, the following condition holds:*

$$x_i \in \left[\frac{-1 - \log_\beta |a_{ii}|}{2}, \ \frac{-\log_\beta |a_{ii}|}{2}\right] \ \textit{when } d_{ii} = 1.$$

Proof of Symmetric Scale Factors Range: The proof for off-diagonal non-zeros is identical to that in lemma (2.1). The second condition can be proved by looking at the post scaling scenario. We want

$$\beta^{x_i} |a_{ii}| \beta^{x_i} \in [\beta^{-1}, 1] \text{ when } d_{ii} = 1.$$

If we take logarithms to the base β, the goal can be expressed as:

$$2x_i + \log_\beta |a_{ii}| \in [-1, 0] \text{ when } d_{ii} = 1.$$

Hence the conclusion follows. □

3.3 Euclidean Metric for Symmetric Matrices

Similar to real matrices discussed in section (2.2) we use Euclidean distance from the center of the desired range in the following lemma.

Lemma 3.2 (Symmetry Preserving Euclidean Metric) *Similar to real matrices, let an $n \times n$ real symmetric matrix $\bar{A}$ be defined with the same sparsity pattern as matrix A. Let elements of $\bar{A}$ be the center of the ranges defined in lemma (3.1), i.e.,*

$$\begin{aligned} \bar{a}_{ij} &= -\log_\beta |a_{ij}| - \frac{1}{2} \qquad \textit{whenever } n_{ij} = 1\ , \\ \bar{a}_{jj} &= \frac{-\log_\beta |a_{jj}| - 1/2}{2} \qquad \textit{whenever } d_{jj} = 1 \textit{ and} \\ \bar{a}_{ij} &= 0 \qquad \textit{if } a_{ij} = 0. \end{aligned}$$

The L_2 norm based objective that minimizes distance from the center of the respective ranges can be stated as:

$$\min \qquad \Pi_2 = \sum_{j=1}^{n} \sum_{i \in N_j} (x_i + x_j - \bar{a}_{ij})^2 + \frac{1}{2} \sum_{j=1}^{n} D_{jj} (x_j - \bar{a}_{jj})^2 .$$

Proof of the Symmetry Preserving Euclidean Metric: The off-diagonal entries of matrix $\bar{A}$ defined here are the same as in lemma (2.2) for real matrices. Diagonal entries are an exception and they denote the center of the range defined in lemma (3.1).

First term of the sum represents deviations in the off-diagonal entries. The inner summation of the first term gives deviations for a column of sub-diagonal entries while the outer summation sums over all the columns. The second term of the summation represents the deviations in diagonal elements. If the diagonal element D_{jj} is zero, it is not considered in the sum.

Since only the lower half, i.e., sub-diagonal elements of the symmetric matrix A, are represented by non-zero entries, the weight given to diagonal elements should be half the weights given to off-diagonal entries. □

Let the $n \times n$ diagonal matrix C such that

$$C_{jj} = (N_j)^t \mathbf{e}. \tag{2}$$

Thus C_{jj} gives count of *sub-diagonal* non-zeros in column j unlike the case of $m \times n$ real matrices where we used it for complete column count. Similar to lemma (2.3) that provided a way to iteratively compute the scale factors for $m \times n$ real matrices, the following lemma defines a set of equations for symmetry preserving scale factors.

Lemma 3.3 (Symmetry Preserving Solutions for Euclidean Criterion:) *The scale factors* x *satisfy:*

$$(2C + D)\,x + 2N^t x = 2\bar{A}^t\mathbf{e} - D\bar{A}\mathbf{e}$$

Proof of Symmetry Preserving Solutions for Euclidean Criterion: As in the $m \times n$ real matrix case, this is an unconstrained least squares problem and the solution exists since Hessian is positive definite. The normal equations are:

$$\frac{\partial \Pi_2}{\partial x_j} = 0 \implies 2\sum_{i \in N_j} (x_i + x_j - \bar{a}_{ij}) + D_{jj}(x_j - \bar{a}_{jj}) = 0$$

First term in the partial follows from the definition of incidence matrix N. Only sub-diagonal elements, i.e., $i > j$, are likely to be non-zero in column j. The second term follows from diagonal element deviation. Collecting variable terms on one side, we can rewrite the normal equations as:

$$2\sum_{i \in N_j} x_i + 2\sum_{i \in N_j} x_j + D_{jj}x_j = 2\sum_{i \in N_j} \bar{a}_{ij} + D_{jj}\bar{a}_{jj}$$

Like $N_{j\_}$ denotes the column j of the incidence matrix, let $\bar{A}_j$ be the column j of matrix $\bar{A}$. Since the column j of the incidence matrix, N_j, has only zeros above the diagonal, and $|N_j| = C_{jj}$, we can simplify the first order conditions to:

$$2N_j^t x + (2C_{jj} + D_{jj})\,x_j = 2N_j^t\bar{A}_j + D_{jj}\bar{a}_{jj}$$

The incidence matrix N contains only sub-diagonal elements by definition while the matrix $\bar{A}$ includes diagonal elements too. Hence the term $2N_j^t\bar{A}_j$ is equivalent to:

$$2N_j^t\bar{A}_j = 2\bar{A}_j^t\mathbf{e} - 2D_{jj}\bar{a}_{jj}.$$

Hence the first order conditions are:

$$2N_j^t x + (2C_{jj} + D_{jj})\,x_j = 2\bar{A}_j^t\mathbf{e} - D_{jj}\bar{a}_{jj}$$

The normal equations can be now written in matrix form as:

$$2N^t x + (2C + D)\,x = 2\bar{A}^t\mathbf{e} - D\bar{A}\mathbf{e}.$$

This gives the result. □

We can interpret conditions of lemma (3.3) as defining scale factors by column. Each scale factor is an average of the non-zero entries of the column (including super-diagonal entries) but adjusted by values assigned to other scale factors incident to the column.

3.4 Finding Scale Factors for Symmetric Matrices

Similar to lemma (2.4) that defined a mechanism to compute scale factors for an $m \times n$ real matrix, the following lemma allows the scale factors of a symmetric matrix to be computed directly.

Lemma 3.4 (Triangular Solves) *The equations that scale factors need to satisfy in lemma (3.3) define a triangular system that is consistent even if it is not invertible.*

Proof of Triangular Solves: If we look at a row of the linear system

$$2N_j^t x + (2C_{jj} + D_{jj})\, x_j = 2N_j^t \bar{A}_j + D_{jj} \bar{a}_{jj}$$

By definition of the incidence matrix, N, only sub-diagonal elements, i.e., $i > j$ are represented in the column N_j. Hence, when $j = 1$, then all the scale factors involved in the equation. On the other hand, when $j = n$, then only x_n is involved. This is an upper triangular system that can be solved in $O(n^2)$ time by back substitution if the matrix $2C + D + 2N^t$ was invertible. If a column j has no diagonal element, i.e., $D_{jj} = 0$ and no sub-diagonal elements implying that N_j contains all zeros and hence sub-diagonal count $C_{jj} = 0$, then the right hand side as well as the left hand side are zero. The equation looks like:

$$2\mathbf{0}^t x + (2 \times 0 + 0) x_j = 2\mathbf{0}^t \mathbf{0} + 0.$$

This scenario is likely to occur in barrier methods where symmetric indefinite matrices are encountered. Such an occurrence means that non-zeros in the column, if any at all, occur above the diagonal and by symmetry they will be scaled when the columns in which they occur below the diagonal are encountered. Thus we set $x_j = 0$ and continue with backward substitution. □

The computations to find symmetry preserving scale factors are shown in algorithm (3). As can be seen, there is only one A matrix scan for the process and it only requires scalars to compute the factors. Similar to the algorithm (2), we defer conversion to integer, by rounding operation, to the last step before the factors are needed.

Like in section (3), we conclude this section (3.4) with an example to illustrate computation of symmetry preserving scale factors. This is another "smelling salts" type of example from [Gaj95, page 67].

Example 3.1 (Pathological Symmetric Matrix) *Similar to example (2.1), we use machine base $\beta = 10$ for better readability. Given the following symmetric matrix*

$$A = \begin{bmatrix} 1 & 10^{20} & 10^{10} & 1 \\ 10^{20} & 10^{20} & 1 & 10^{40} \\ 10^{10} & 1 & 10^{40} & 10^{50} \\ 1 & 10^{40} & 10^{50} & 1 \end{bmatrix},$$

Algorithm 3 Symmetry Preserving Scale Factors using Triangular Solves

Require: $n > 0$

for $j = n$ going down to 1 **do** {Backward substitution}
  $c \leftarrow 0$ { For $2C_{jj} + D_{jj} - 2N_j^t x$ }
  $t \leftarrow a$ { For $2N_j^t \bar{A}_j + D_{jj}\bar{a}_{ij}$ }
  if $D_{jj} = 1$ **then** {diagonal element}
    $t \leftarrow \frac{1}{2}\left(-\log_\beta |a_{jj}| - 1/2\right)$
    $c \leftarrow 1$
  end if
  for all sub-diagonal non-zeros (i, j) in column j of matrix A **do** $\{i > j\}$
    $t \leftarrow t + 2\left(-\log_\beta |a_{ij}| - 1/2\right) - 2x_i$
    $c \leftarrow c + 2$
  end for{end for column j}
  if $c > 0$ **then** {positive count, otherwise let $x_j = 0$ }
    $x_j \leftarrow \text{Round}(t/c)$
  end if
end for
return integer vector x

what are the appropriate symmetry preserving scale factors?

Solution of Pathological Symmetric Matrix: The sub-diagonal incidence matrix, N, corresponding to dense matrix A, is a 4×4 matrix of all ones below the diagonal and zeros everywhere else. The diagonal incidence matrix has ones on its principal diagonal and zeros everywhere else. We execute steps of the symmetry preserving scale factors algorithm (3) and start with x_4 in the backward substitution.

Scale factor x_4*:* The count $c = 1$ since only a diagonal element with 0 sub-diagonal elements. The value of $t = -1/4$. Hence $x_4 = 0$ after rounding off.

Scale factor x_3*:* Count $c = 1 + 2 = 3$ since there is one sub-diagonal element. The value of $t = -121.25$ and hence $x_3 = -40$.

Scale factor x_2*:* Count $c = 1+2+2 = 5$ since there are two sub-diagonal elements. The value of $t = -92.25 - 2 \times (x_3 = 40) - 2 \times (x_4 = 0) = -12.25$ and hence $x_2 = -2$.

Scale factor x_1*:* Count $c = 1 + 2 + 2 + 2 = 7$ since there are three sub-diagonal elements. The value of $t = -63.25 - 2 \times -40 - 2 \times -2 = 20.75$ and hence $x_1 = -3$.

Using the symmetry preserving scale factors $x^t = (3, -2, -40, 0)$ the scale matrix A' looks like:

$$A' = XAX = \begin{bmatrix} 10^{6} & 10^{21} & 10^{-27} & 10^{3} \\ 10^{21} & 10^{16} & 10^{-42} & 10^{38} \\ 10^{-27} & 10^{-42} & 10^{-40} & 10^{10} \\ 10^{3} & 10^{38} & 10^{10} & 1 \end{bmatrix}.$$

Using estimates suggested in section (A), we see that $\| A \|_1 \approx 10^{50}$ and $\left\| A^{-1} \right\|_1 \geq 10^{20}$ and hence the condition number of A, $\kappa_1(A) \geq 10^{70}$. For the scaled matrix A' we see the lower bound is 10^{48}. Hence it seems to have improved the condition number. □

3.5 Manhattan Metric for Symmetric Matrices

Similar to lemma (2.7) for $m \times n$ matrix, the following lemma suggests a linear programming formulation based on Manhattan metric for symmetric matrices.

Lemma 3.5 (Symmetry Preserving Manhattan Metric) *Let a $n \times n$ real symmetric matrix $\bar{A}$, with sparsity pattern of matrix A, be defined as in lemma (3.2). The L_1 norm based objective that minimizes the distance from the center of the range can be stated as:*

$$\min \qquad \Pi_1 = \sum_{j=1}^{n} \sum_{i \in N_j} z_{ij} + \sum_{j=1:D_{jj}=1}^{n} z_{jj},$$

subject to constraints:

$$\begin{aligned} -z_{ij} + x_i + x_j &\leq \bar{a}_{ij} && \forall\, i, j \text{ such that } n_{ij} = 1, \\ -z_{ij} - x_i - x_j &\leq -\bar{a}_{ij} && \forall\, i, j \text{ such that } n_{ij} = 1, \\ -z_{jj} + x_j &\leq \bar{a}_{jj} && \forall\, j \text{ such that } d_{jj} = 1, \\ -z_{jj} - x_j &\leq -\bar{a}_{jj} && \forall\, j \text{ such that } d_{jj} = 1. \end{aligned}$$

Proof of Symmetry Preserving Manhattan Metric: The proof of correctness of objective function term and constraints of the sub-diagonal elements is identical to lemma (2.7) for $m \times n$ matrix.

For diagonal non-zeros, goal of minimizing distance from center of the desired range in lemma (3.1) can be written as:

$$\sum_{j=1:D_{jj}=1}^{n} |\, x_j - \bar{a}_{jj} \,| = \sum_{j=1:D_{jj}=1}^{n} \max\{x_j - \bar{a}_{jj},\ \bar{a}_{jj} - x_j\}.$$

Defining variables z_{jj} to be the result of the max operator we get the constraints in the same manner as sub-diagonal elements. □

To define associated dual, we need rows of the incidence matrix N. Since it is defined for sub-diagonal elements, row N^j, contains non-zeros for sub-diagonal non-zeros of row j. To construct a full row or column j of matrix A, given non-zeros below and on the diagonal, we can use the set $N_j \cup N^j \cup D_{jj}$. Similar to lemma (2.8) for $m \times n$ matrices, the following lemma yields a smaller model involving $|N|+|D|$ variables, n explicit equality constraints by using duality.

Lemma 3.6 (Symmetry Preserving Manhattan Dual) *Dual multipliers of the following linear program solve the primal problem in lemma (3.5).*

$$\max \qquad \bar{\Pi}_1^D = \sum_{j=1}^{n} \sum_{i \in N_j} \bar{a}_{ij} u_{ij} + \sum_{j=1:D_{jj}=1}^{n} \bar{a}_{jj} s_j.$$

Subject to constraints:

$$\begin{aligned} \sum_{i \in N^j \cup N_j} u_{ij} + d_{jj} s_j &= \frac{|N_j| + \left|N^i\right| + D_{jj}}{2} \qquad \forall\, j = 1, \ldots, n, \\ 0 \;\leq\; u_{ij} \;&\leq\; 1 \qquad \forall\, i, j \text{ such that } n_{ij} = 1 \\ 0 \;\leq\; s_j \;&\leq\; 1 \qquad \forall\, j = 1, \ldots, n. \end{aligned}$$

Proof of Symmetry Preserving Manhattan Dual: We can write the dual following the same approach as in lemma (2.8). In the linear program in lemma (3.5) we associate non-negative dual multipliers u_{ij} and v_{ij} with first and second set of constraints respectively representing sub-diagonal non-zeros. In addition, we need dual multipliers s_j and t_j for the third and fourth set of constraints respectively for diagonal non-zeros. The dual can be written as:

$$\max \qquad \Pi_1^D = \sum_{j=1}^{n} \sum_{i \in N_j} \bar{a}_{ij} \left(u_{ij} - v_{ij}\right) + \sum_{j=1:D_{jj}=1}^{n} \bar{a}_{jj} \left(s_j - t_j\right),$$

subject to constraints:

$$\begin{aligned} u_{ij} + v_{ij} &= 1 \qquad \leftarrow z_{ij} \qquad \forall\, i, j \;: n_{ij} = 1, \\ s_j + t_j &= 1 \qquad \leftarrow z_{jj} \qquad \forall\, j \;: d_{jj} = 1, \\ \sum_{i \in N^j \cup N_j} \left(u_{ij} - v_{ij}\right) + d_{jj} \left(s_j - t_j\right) &= 0 \qquad \leftarrow x_j \qquad \forall\, j = 1, \ldots, n. \end{aligned}$$

Using the substitutions $v_{ij} = 1 - u_{ij}$ and $t_j = 1 - s_j$, and getting rid of constants in the objective we get the required result. □

A Condition Number Estimates

In this section we let A be a square $n \times n$ invertible matrix. We develop bounds that can be *computed efficiently* on condition numbers before the linear system is solved. Ideally, we would like the computations to be $O(|A|)$ where $|A|$ denotes the number of non-zeros in A matrix. We briefly mention the post-solution methods that could be used in diagnostics. A matrix norm induced by any vector norm, $\|\ \|$, is called the subordinate norm. Thus for any $p-$norm, the corresponding matrix $p-$norm induced by it is given by:

$$\|A\|_p = \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \sup_{\|x\|_p = 1} \|Ax\|_p .$$

The condition number of matrix A, denoted by $\kappa_p(A)$ under any induced matrix $p-$norm is defined to be

$$\kappa_p(A) = \|A\|_p \left\|A^{-1}\right\|_p .$$

Note, for any non-zero vector x, using substitution $Ay = x$, we get

$$\begin{aligned} \left\|A^{-1}\right\|_p &= \sup_{x \neq 0} \frac{\left\|A^{-1}x\right\|_p}{\|x\|_p} = \sup_{y \neq 0} \frac{\|y\|_p}{\|Ay\|_p} = \sup_{y \neq 0} \frac{1}{\|Ay\|_p / \|y\|_p} \\ &= \frac{1}{\inf_{y \neq 0} \|Ay\|_p / \|y\|_p}. \end{aligned}$$

Since, A is invertible, both the supremum and infimum are attained.

Using the terminology from [Wat91, page 96], we can get a geometric picture by defining two terms

$$\begin{aligned} \text{maxmag}(A) &= \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} \\ \text{minmag}(A) &= \inf_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}. \end{aligned}$$

Thus $\|A\|_p$ signifies maximum magnification of a vector by the matrix A and it follows from the preceding discussion that

$$\begin{aligned} \text{maxmag}(A) &= \|A\|_p , \\ \text{maxmag}(A^{-1}) &= \frac{1}{\text{minmag}(A)}, \\ \text{maxmag}(A) &= \frac{1}{\text{minmag}(A^{-1})}. \end{aligned}$$

Using the new terms defined, the condition number is the ratio of maximum to minimum magnification done by matrix A on any vector x.

$$\kappa_p(A) = \frac{\text{maxmag}(A)}{\text{minmag}(A)}. \tag{3}$$

A lower bound on the condition number [Wat91, page 101] can be trivially established. If A_j denotes column j, then

$$\kappa_p(A) \geq \frac{\| A_i \|_p}{\| A_j \|_p} \qquad \forall\, i, j = 1, \ldots, n. \tag{4}$$

Using the columns of identity matrix as x in computing maximum and minimum magnification definition gives the result. Thus, the condition number will be large if the columns differ widely in magnitude under all $p-$norms.

A.1 L_2 Norm

For the Euclidean norm, we can write we have

$$\| A \|_2 = \sup_{\| x \|_2 = 1} \| Ax \|_2 \Longleftrightarrow \max_{x^t x = 1} (Ax)^t Ax.$$

The Kuhn-Tucker conditions imply $2(A - \lambda I)x = 0$. Let λ_{max} be the largest eigenvalue of matrix A and let λ_{min} be the smallest eigenvalue. Then $\text{maxmag}(A) = \lambda_{max}$ and $\text{minmag}(A) = \lambda_{min}$. Thus the condition number of matrix A under the Euclidean metric from equation (3) is

$$\kappa_2(A) = \frac{\lambda_{max}}{\lambda_{min}}.$$

Since eigenvalues are usually not known before hand, the lower bound based on column norms in equation (4) may be appropriate at the pre-solution stage.

A.2 L_∞ Norm

For the infinity norm, $p = \infty$, the induced matrix norm is called *row sum norm* because $\text{maxmag}(A)$ is the row with largest sum of magnitudes [Wat91, page 93]), i.e.,

$$\| A \|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^{n} | a_{ij} | .$$

To estimate a lower bound on the condition number, we try to guess an upper bound on the minimum magnification. Using A_j for column j, and e_j for a corresponding column of the identity matrix we have:

$$\text{minmag}(A) = \min_{\|x\|_\infty = 1} \|Ax\|_\infty \leq \min_{1 \leq j \leq n} \|Ae_j\|_\infty .$$

Since Ae_j is column j of A, an upper bound on the minimum magnification is given by:

$$\text{minmag}(A) \leq \min_{1 \leq j \leq n} \|A_j\|_\infty .$$

In other words, a lower bound on condition number can be estimated by equation (3):

$$\kappa_\infty(A) \geq \|A\|_\infty / \min_{1 \leq j \leq n} \|A_j\|_\infty .$$

This is likely to be a tighter bound than the one computed in equation (4) since at least the numerator is exact. It can be estimated by a single scan of the matrix A and requires finding the largest magnitude in each column, and then finding the column with smallest such magnitude. This corresponds to a solution of setting $x_k = 1$ for the column k with smallest infinity norm and other components to zero.

A.3 L_1 Norm

Similar to infinity norm, the matrix norm induced by the taxicab metric is called *column sum norm* because maxmag(A) is the column with largest sum of magnitudes [Wat91, page 93]),i.e.,

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^{n} |a_{ij}| .$$

For the taxicab metric, $p = 1$, using A^i as row i of matrix A,

$$\begin{aligned}
\text{minmag}(A) &= \min_{\|x\|_1 = 1} \|Ax\|_1 = \min_{\|x\|_1 = 1} \sum_{i=1}^{n} \left| (A^i)^t x \right| \\
&= \min_{\|x\|_1 = 1} \sum_{i=1}^{n} \left| \sum_{j=1}^{n} a_{ij} x_j \right| \\
&\leq \min_{\|x\|_1 = 1} \sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}||x_j| = \min_{\|x\|_1 = 1} \sum_{j=1}^{n} |x_j| \sum_{i=1}^{n} |a_{ij}| \\
&= \min_{\|x\|_1 = 1} \sum_{j=1}^{n} |x_j| \|A_j\|_1 .
\end{aligned}$$

Hence for the rectilinear norm, to find the upper bound on the minimum magnification, we find the L_1 norm of each column. The smallest such value forms an upper bound on minmag(A). Hence for the L_1 metric we seem to do no better than equation (4).

A.4 Post Solution Estimates

In addition to the above pre-solution methods, a post-solution bound on $\| A^{-1} \|_p$ can be found. Since the induced matrix norm is consistent, we have

$$\| A^{-1}b \|_p \leq \| A^{-1} \|_p \| b \|_p .$$

In other words, we have

$$\| A^{-1} \|_p \geq \frac{\| A^{-1}b \|_p}{\| b \|_p} = \frac{\| x \|_p}{\| b \|_p},$$

where x is the computed solution. This bound is thus obtained by two vector norms and does not involve any matrix scans. In addition, this gives *an upper bound on the condition number* unlike the pre-solution methods. More tighter bounds can be obtained by using the factors generated. This can be formulated as another optimization problem.

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.

[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.

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

[Kre85] Erwin Kreyszig. *Advanced Engineering Mathematics*. Wiley Eastern Limited, fifth edition, November 1985.

[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.

[RZ92] Uriel G. Rothblum and Stavros A. Zenios. Scaling of matrices satisfying line-product constraints and generalizations. *Linear Algebra and its Applications*, 175:159–175, 1992.

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