Avoiding Numerical Cancellation in the Interior Point Method for Solving Semidefinite Programs

Jos F. Sturm

Department of Econometrics, Tilburg University, The Netherlands.
j.f.sturm@kub.nl
http://fewcal.kub.nl/sturm

August 5, 2002

Abstract

The matrix variables in a primal-dual pair of semidefinite programs are getting increasingly ill-conditioned as they approach a complementary solution. Multiplying the primal matrix variable with a vector from the eigenspace of the non-basic part will therefore result in heavy numerical cancellation. This effect is amplified by the scaling operation in interior point methods. A complete example illustrates these numerical issues. In order to avoid numerical problems in interior point methods, we propose to maintain the matrix variables in a Cholesky form. We discuss how the factors of the v-space Cholesky form can be updated after a main iteration of the interior point method with Nesterov-Todd scaling. An analogue for second order cone programming is also developed. Numerical results demonstrate the success of this approach.

Keywords: Semidefinite Programming, Second Order Cone Programming.

AMS subject classification: 90C22, 90C20.

JEL codes: C61, C63.

1   Introduction

This paper addresses numerical issues in interior point methods for semidefinite programming, particularly focusing on the final iterates. The numerical issues that we encounter belong to the category of *numerical cancellation*. In general, the phenomenon of numerical cancellation refers to the following situation: the computer adds two numbers, say α and β, and the result is much smaller in magnitude, i.e. $|\alpha + \beta|/(|\alpha| + |\beta|) << 1$. For instance, in a floating point system with 4 digits of accuracy, the computation '`-0.1350 + 0.1357 = 0.7E-3`' suffers from numerical cancellation, since the result has merely one significant digit: we do not know whether it is say `0.65E-3` or `0.74E-3`, because '`-0.1350`' and '`0.1357`' are merely 4-digit floating point representations of real values.

In linear programming, we know that some of the nonnegative decision variables will have to approach zero in order to obtain optimality; these are the so-called non-basic variables. In

semidefinite programming, the eigenvalues of a symmetric matrix variable X are restricted to be nonnegative, and some of these nonnegative eigenvalues approach zero for near optimal solutions. In fact, associated with a semidefinite programming problem is a matrix Q_N of full column rank such that $X^{(k)}Q_N \to 0$ for any solution sequence $X^{(1)}, X^{(2)}, \ldots$ that approaches optimality [7, 17]. An example is

$$X^{(k)} = \left[\begin{array}{cc} 1 + 1/k & 1 \\ 1 & 1 + 1/k \end{array} \right], \quad Q_N = \left[\begin{array}{c} 1 \\ -1 \end{array} \right].$$

As the example illustrates, the entries in the matrix $X^{(k)}$ itself will in general *not* approach zero. Therefore, the computation '$X^{(k)}Q_N$' will increasingly suffer from numerical cancellation when optimality is approached. A specific example of such a computation is the matrix product 'XZ', where Z is a nearly optimal dual slack variable associated with a nearly optimal matrix variable X. We will show later that the problem also appears when computing the search direction for a degenerate problem, and in the evaluation of a primal-dual scaling. Particularly dangerous are computations involving X^{-1}, since X is getting nearly singular, and the small eigenvalues cannot be reliably computed from X. However, these type of computations typically appear in interior point methods, as part of the so-called scaling operation. The loss of accuracy due to numerical cancellation will be even more pronounced if the sequence $X^{(1)}, X^{(2)}, \ldots$ is unbounded (i.e. the entries become arbitrarily large), as is true for problems in which the optimal value cannot be attained.

We remark that the problems of numerical cancellation as discussed so far do not occur in linear programming, because there the quantities that approach zero are stored as separate entries, viz. the non-basic variables. Therefore, we propose to store the primal and dual iterates (X and Z) in a *Cholesky form* that allows us to maintain the contributions from all eigenvalues with high accuracy. The technique has been implemented as Versions 1.04 and 1.05 of the popular semidefinite programming solver SeDuMi [25].

Numerical issues in semidefinite programming have been addressed in several research papers [3, 15, 33]. Todd, Toh and Tütüncü [33] investigated the so-called AHO [3], HRVW/KSH/M [10], and NT [21, 22, 30] search directions in a numerical experiment, using their SDPT3 software [35]. By and large, they obtained the highest accuracy with the AHO direction. Alizadeh, Haeberly and Overton [3] provided a possible explanation why the AHO direction may have better numerical performance in the final stage of the interior point method than the HRVW/KSH/M and NT search directions. Very recently, Kruk [15] observed that the Gauss-Newton method may suffer less from numerical problems than the interior point method, and he also provides a possible explanation. But, unlike the interior point approach, the Gauss-Newton method is not known to have polynomial convergence, and the computational experience on practical semidefinite programming models is still relatively limited.

The organization of the paper is as follows. In Section 2, we discuss the definitions and notational conventions which are used in the remaining sections. We also state some well known facts from the interior point method for semidefinite programming. Section 3 presents an example which is used to illustrate the main sources of numerical problems in the interior point method for semidefinite programming. A formal error analysis shows that more accurate results can be obtained if the floating point representation of the Cholesky factors of the iterates is known, instead of the iterates themselves. The example illustrates this fact. For Nesterov-Todd scaling, we therefore propose to

maintain the iterates in their V-space Cholesky form [30]. Section 4 provides a numerically stable update formula for the v-space Cholesky form. We propose a procedure with similar favorable characteristics for second order cone programming (SOCP) in Section 5. This section uses concepts from Euclidean Jordan algebra, and can be skipped by readers who are not specifically interested in SOCP. In Section 6, we discuss the matter of solving the scaled normal equations system, which appears in each iteration of the interior point method. We address not only numerical issues, but also efficiency issues. An outline of the specific strategy which has been implemented in SeDuMi 1.05 is given. Numerical results are reported in Section 7. The paper in concluded in Section 8.

2 Preliminaries

We consider semidefinite programming problems in the following standard form:

$$
\begin{aligned}
&\text{minimize} && \operatorname{tr} CX \\
&\text{such that} && \operatorname{tr} A_i X = b_i \text{ for } i = 1, 2, \ldots, m \\
& && X \text{ is symmetric and positive semidefinite,}
\end{aligned}
\tag{1}
$$

where $A_1, A_2, \ldots, A_m$ and C are given $\nu \times \nu$ matrices, b is a vector with components $b_1, b_2, \ldots, b_m$, and the decision variable is an $\nu \times \nu$ matrix X.

The standard $\operatorname{vec}(\cdot)$ operator [12] stacks the columns of a matrix into a long vector, i.e.

$$
\operatorname{vec}(X) = \operatorname{vec}\left(\begin{bmatrix} x_1 & \cdots & x_\nu \end{bmatrix}\right) = \begin{bmatrix} x_1 \\ \vdots \\ x_\nu \end{bmatrix},
$$

where $x_i \in \Re^\nu$, $i = 1, \ldots, \nu$, are the columns of X.

Letting $a_i = \operatorname{vec}(A_i)$, $i = 1, \ldots, m$, $A := \begin{bmatrix} a_1, & \cdots, & a_m \end{bmatrix}^{\mathrm{T}}$, and $c := \operatorname{vec}(C)$, we can reformulate the standard semidefinite programming problem as a conic linear program in $\Re^n$ with $n = \nu^2$ as follows:

$$
\inf\{c^{\mathrm{T}} x \mid Ax = b, \ x \in \operatorname{PSD}(\nu)\}.
\tag{2}
$$

The decision variable is now $x = \operatorname{vec}(X) \in \Re^n$. The convex cone $\operatorname{PSD}(\nu)$ is defined as follows:

$$
\operatorname{PSD}(\nu) := \{\operatorname{vec}(X) \mid X \in \Re^{\nu \times \nu} \text{ is symmetric and positive semidefinite}\}.
$$

The dual cone of $\operatorname{PSD}(\nu)$ in $\Re^n$ is then

$$
\operatorname{PSD}(\nu)^* = \{\operatorname{vec}(Z) \mid Z \in \Re^{\nu \times \nu}, \ Z + Z^{\mathrm{T}} \text{ is positive semidefinite}\}.
$$

Associated with (2) is a dual problem, viz.

$$
\sup\{b^{\mathrm{T}} y \mid c - A^{\mathrm{T}} y \in \operatorname{PSD}(\nu)^*\}.
\tag{3}
$$

The vector of dual decision variables is $y \in \Re^m$.

We may replace 'C' by $(C + C^{\mathrm{T}})/2$ and similarly 'A_i' by $(A_i + A_i^{\mathrm{T}})/2$, $i = 1, 2, \ldots, m$, without affecting problems (2) and (3). Therefore, we will assume without loss of generality that the matrices A_i, $i = 1, \ldots, m$ and C are symmetric. Under this assumption, we have

$$c - A^{\mathrm{T}}y \in \mathrm{PSD}(\nu)^* \iff c - A^{\mathrm{T}}y \in \mathrm{PSD}(\nu).$$

Given a dual solution y we define the dual slack as $Z = C - \sum_{i=1}^{m} y_i A_i$, or $z = \mathrm{vec}\,(Z) = c - A^{\mathrm{T}}y$ in vectorized form.

Given matrices that are identified by upper-case symbols X, Z and C, we implicitly define the lower case symbols x, z and c as $x := \mathrm{vec}\,(X)$, $z := \mathrm{vec}\,(Z)$, and $c := \mathrm{vec}\,(C)$, We will use the matrix form (1) and vector form (2) of a semidefinite program interchangeably, depending on the context. Observe that $\|x\|_2^2 = \mathrm{tr}\, X^{\mathrm{T}}X$; we let $\|X\|_F := \sqrt{\mathrm{tr}\, X^{\mathrm{T}}X}$ denote the Frobenius norm of X.

2.1 Path-Following Direction

In each iteration of the (feasible) primal-dual interior point method, a primal feasible solution X and a dual feasible solution Z are computed, such that X and Z are positive definite. Whenever such solutions exist, it holds that positive *semi*-definite feasible solutions X and Z are optimal if and only if $XZ = 0$. We refer to Alizadeh [1], Vandenberghe and Boyd [37] and Todd [31, 32] for general introductions to semidefinite programming. For an introduction to path-following methods, we refer to Gonzaga [8, 9].

The central path, which is parameterized by a parameter $\mu > 0$, is characterized by the so-called central-path equations:

$$\begin{cases} X(\mu)Z(\mu) = \mu I \\ Ax(\mu) = b \\ A^{\mathrm{T}}y(\mu) + z(\mu) = c \\ X(\mu) \in \mathrm{PSD}(\nu),\, Z(\mu) \in \mathrm{PSD}(\nu),\, y(\mu) \in \mathrm{Img}\, A, \end{cases} \tag{4}$$

where $\mathrm{Img}\, A$ denotes the image (or range space) of A. For simplicity, we will assume that A has full row rank, so that $\mathrm{Img}\, A = \Re^m$.

The distance to optimality for a feasible solution is measured by the duality gap, which is $c^{\mathrm{T}}x - b^{\mathrm{T}}y = x^{\mathrm{T}}z = \mathrm{tr}\, XZ$. Obviously, $\mathrm{tr}\, X(\mu)Z(\mu) = \nu\mu$.

We let $\lambda_{\min}(XZ)$ denote the smallest eigenvalue of XZ, which is real since $XZ \sim X^{1/2}ZX^{1/2}$. If (X, y, Z) is a given iterate in the interior point method, then X and Z are positive definite. Furthermore, in path-following and central region following methods, we have that $\lambda_{\min}(XZ)/(x^{\mathrm{T}}z)$ is bounded from below by a positive constant which is independent of the duality gap; this is the so-called centrality condition. In particular, one has

$$\lambda_{\min}(XZ) \geq \frac{(1 - \beta)(x^{\mathrm{T}}z)}{\nu}$$

for $N_2(\beta)$ and $N_\infty^-(\beta)$ path-following methods [20, 14], and

$$\lambda_{\min}(XZ) \geq \frac{(1 - \beta)\theta(x^{\mathrm{T}}z)}{\nu} \tag{5}$$

for central region following methods [29, 26] with parameters $0 < \beta < 1$ and $0 < \theta \leq 1$. Observe that $\lambda_{\min}(XZ)/(x^{\mathrm{T}}z) = 1/\nu$ if and only if (X, y, Z) is on the central path. The centrality property is important for our numerical investigation.

A main iteration of the interior point method consists of computing a search direction that is added to the current iterate with a certain step length $t > 0$, yielding the next iterate:

$$X^+ = X + t\Delta X, \tag{6}$$

$$y^+ = y + t\Delta y, \ Z^+ = Z + t\Delta Z. \tag{7}$$

The primal-only and dual-only path-following methods use only (6) and (7), respectively.

The search direction $(\Delta x, \Delta y, \Delta z)$ is implicitly defined by a system of equations, as follows:

$$\begin{cases} \Delta x + \Pi\Delta z = r \\ A\Delta x = 0 \\ A^{\mathrm{T}}\Delta y + \Delta z = 0. \end{cases} \tag{8}$$

The system is parameterized by an invertible $n \times n$ matrix Π and a vector $r \in \Re^n$, which depend not only on the iterate, but also on the specific algorithmic choices of the interior point method. Choices for Π and r which correspond to the so-called predictor (or affine scaling) direction are as follows:

	Π	r
primal-only	$X \otimes X$	$-\Pi z$
dual-only	$Z^{-1} \otimes Z^{-1}$	$-x$
primal-dual	$\Pi z = x$	$-x.$

Here, '$\otimes$' denotes the standard Kronecker product [12], i.e.

$$(D \otimes D)\,\mathrm{vec}\,(X) = \ \mathrm{vec}\,(DXD^{\mathrm{T}}).$$

The vectors z in the primal-only and x in the dual-only rows need not be in $\mathrm{PSD}(\nu)$ and can be replaced by c and $A^{\mathrm{T}}(AA^{\mathrm{T}})^{-1}b$, respectively; the specific choice does not affect the search direction. However, the relation $\Pi z = x$ in the primal-dual row does *not* uniquely define the matrix Π, and the corresponding primal-dual search direction. A possible choice of Π is $\Pi = (X \otimes Z^{-1} + Z^{-1} \otimes X)/2$, resulting in the so-called HRVW/KSH/M-direction [10].

In this paper, we are mainly concerned with the so-called NT-direction [21, 22]. The NT-direction corresponds to $\Pi = D \otimes D$, where D is the unique positive definite matrix satisfying $\Pi z = (D \otimes D)z = x$. In other words, D is characterized by

$$D \succ 0, \ DZD = X. \tag{9}$$

In fact, D is the metric geometric mean [4, 33] of X and Z^{-1}. An explicit formula for D is:

$$D = X^{1/2}(X^{1/2}ZX^{1/2})^{-1/2}X^{1/2}. \tag{10}$$

Observe that on the central path, we have in all approaches that Π is a multiple of $X(\mu) \otimes X(\mu)$. Furthermore, r is a multiple of $-x$ for the affine scaling directions on the central path. In fact, the resulting search direction is known to be the tangent direction to the central path.

We remark from (8) that

$$0 = A\Pi(A^{\mathrm{T}}\Delta y + \Delta z) = A\Pi A^{\mathrm{T}}\Delta y + A(r - \Delta x) = A\Pi A^{\mathrm{T}}\Delta y + Ar,$$

yielding

$$A\Pi A^{\mathrm{T}} \Delta y = -Ar. \tag{11}$$

Notice that for the case that $r = -x$ we have $-Ar = b$, and (11) can be further simplified to

$$A\Pi A^{\mathrm{T}} \Delta y = b. \tag{12}$$

Once the Δy direction is known, the Δz and Δx directions then follow easily as

1. $\Delta z = -A^{\mathrm{T}}\Delta y$, and

2. $\Delta x = r - \Pi\Delta z$.

The main computational effort in the interior point method lies therefore in solving a system of the form (12). From a numerical point of view, evaluating '$r - \Pi\Delta z$' is also a challenge (even in the case of linear programming). Namely, if the iterates converge then $\Delta x\ (= r - \Pi\Delta z)$ approaches zero, whereas r does not; hence the accuracy in Δx suffers from numerical cancellation. The error can be reduced by one step of iterative refinement, based on the residual $A \cdot \mathrm{fl}(r - \Pi\Delta z)$. Here, we use the notation 'fl(x operation y)' to denote the result of a floating point operation [11].

Degeneracy can make the system (12) ill conditioned. In the setting of this paper, we consider a problem to be degenerate if for any primal optimal solution X^* there exists a $\psi \in \Re^m$ such that

$$A^{\mathrm{T}}\psi \neq 0, \ (X^* \otimes X^*)A^{\mathrm{T}}\psi = 0. \tag{13}$$

If a semidefinite program is primal degenerate in the sense of [2] then it is also degenerate in the sense of (13). The converse is not necessarily true, see Theorem 6 in [2] for details. Practical optimization problems are often degenerate.

In actual implementations of the interior point method, the right hand side in (12) may not be exactly the b vector, because

- an infeasible interior point method is used, i.e. Ax is not necessarily equal to b, see Lustig, Marsten and Shanno [18], or

- a self-dual embedding is used, see Ye, Todd and Mizuno [39], and

- the direction is not simply a predictor direction, but includes e.g. a second order correction, see Mehrotra [19].

Nevertheless, for the purpose of this paper it suffices to focus on the system (12). This systems stays the same throughout the interior point process, except for the linear operator Π. Thus, if the operator Π cannot be evaluated with sufficient accuracy, then solving (12) becomes pointless due to the 'garbage in, garbage out' effect, more formally known as error propagation. This issue is discussed in more detail in the next section.

3 Benefit of Cholesky Form

In this section, we consider the numerical accuracy of evaluating the Π-operator in floating point arithmetic, for instance in the computation of $A\Pi A^{\mathrm{T}}$.

In Sections 3.1 and 3.2, we assume that Π is of the form $\Pi = X \otimes X$, where X is positive definite. If X is a primal iterate, then this choice of Π corresponds to primal scaling. By letting X denote the inverse of the dual matrix iterate ($\Pi = Z^{-1} \otimes Z^{-1}$) or the NT scaling point ($\Pi = D \otimes D$), we can treat dual scaling and NT-type primal-dual scaling in the same way. Since $X(\mu) = \mu Z(\mu)^{-1} = \sqrt{\mu} D(\mu)$, a single example illustrates both primal scaling, dual scaling and primal-dual scaling on the central path.

The upper-triangular Cholesky factor of X is denoted U, i.e. $U^{\mathrm{T}} U = X$. We compare two models: one in which we know $\mathrm{fl}(X)$, versus one in which we know $\mathrm{fl}(U)$. In Section 3.3, we discuss how Cholesky factors can be used in primal-dual scaling. Furthermore, we discuss the issue of error propagation in the case of NT-scaling.

3.1 An example

Consider the following example:

$$\begin{cases} \min\{ \operatorname{tr} CX \mid \operatorname{tr} A_1 X = 0, \ \operatorname{tr} A_2 X = b_2, X \in \mathrm{PSD}(3)\} \\ \max\{b_2 y_2 \mid Z = C - y_1 A_1 - y_2 A_2 \in \mathrm{PSD}(3)\}, \end{cases}$$

where

$$C = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, A_1 = \begin{bmatrix} 3 & 1 & 5 \\ 1 & 0 & 2 \\ 5 & 2 & 8 \end{bmatrix}, A_2 = \begin{bmatrix} 3 & 1 & 5 \\ 1 & 1 & 2 \\ 5 & 2 & 8 \end{bmatrix}.$$

An optimal solution is given as

$$X^* = b_2 \begin{bmatrix} 4 & -2 & -2 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix}, y^* = \begin{bmatrix} -1 \\ 0 \end{bmatrix} /2, Z^* = \begin{bmatrix} 5 & 3 & 7 \\ 3 & 2 & 4 \\ 7 & 4 & 10 \end{bmatrix} /2.$$

In this simple example, the central path is affine, viz.

$$X(\mu) = X^* + \mu \begin{bmatrix} 20 & 0 & -14 \\ 0 & 0 & 0 \\ -14 & 0 & 10 \end{bmatrix},$$
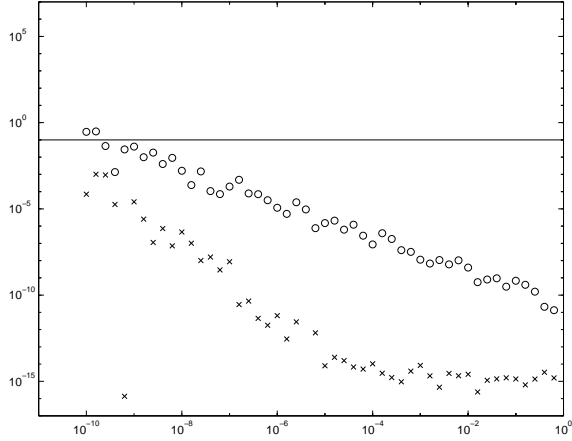
7

Figure 1: Relative errors on central path for tr $A_1 X(\mu) A_1 X(\mu)$ with $b_2 = 1000\pi$. '×'=relerr($U(\mu)$), 'o'=relerr($X(\mu)$). The x-axis is the gap $\nu\mu$.
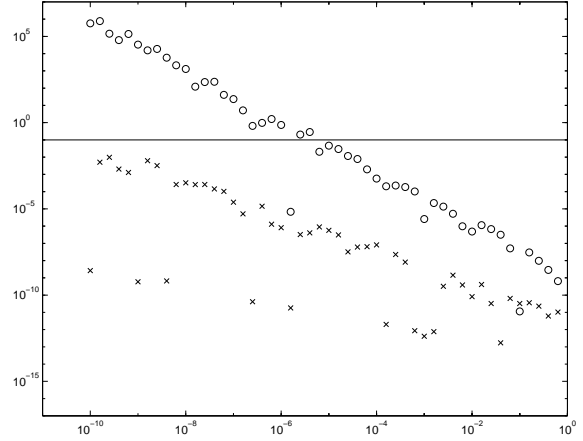
Figure 2: Relative errors outside central path for tr $A_1 X A_1 X$ with $b_2 = 1000\pi$. '×'=relerr(U), 'o'=relerr(X). The x-axis is the gap $\nu\mu$.

$$y(\mu) = y^* + (\mu/b_2) \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \ Z(\mu) = Z^* + (\mu/b_2) \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

The Cholesky factor $U(\mu)$ with $U(\mu)^{\mathrm{T}} U(\mu) = X(\mu)$ is

$$\phi := \sqrt{b_2 + 5\mu}, \ U(\mu) = \begin{bmatrix} 2\phi & -b_2/\phi & -(b_2 + 7\mu)/\phi \\ 0 & \sqrt{5\mu b_2}/\phi & -2\sqrt{\mu b_2/5}/\phi \\ 0 & 0 & \sqrt{\mu/5} \end{bmatrix}. \tag{14}$$

The upper-left entry in the 2×2 matrix $A\Pi A^{\mathrm{T}} = A(X(\mu) \otimes X(\mu))A^{\mathrm{T}}$ is:

$$\mathrm{tr} \ A_1 X(\mu) A_1 X(\mu) = \ \mathrm{tr} \ (A_1 X(\mu))^2 = 8\mu^2.$$

Figure 1 shows the relative errors in computing this quantity for $b_2 = 1000\pi$ as a function of the gap $\nu\mu$, $10^{-10} \le \nu\mu \le 1$. (There is nothing magic about this choice of b_2, except that it is irrational.) The relative errors are defined as:

$$\mathrm{relerr}(X(\mu)) := \frac{|8\mu^2 - \mathrm{fl}(\ \mathrm{tr} \ (A_1 X(\mu))^2)|}{8\mu^2}, \tag{15}$$

$$\mathrm{relerr}(U(\mu)) := \frac{|8\mu^2 - \mathrm{fl}(\ \mathrm{tr} \ (U(\mu) A_1 U(\mu)^{\mathrm{T}})^2)|}{8\mu^2}. \tag{16}$$

The numerical results were obtained using MATLAB 5.3 with IEEE double precision floating point arithmetic.

8

In practice, the iterates are of course *not* exactly on the central path. An example of a primal interior solution X satisfying $\operatorname{tr} CX = \operatorname{tr} CX(\mu)$ is

$$
X = \varphi \left(\begin{bmatrix} 4 & -2 & -2 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix} + \delta \begin{bmatrix} 0 & 4 & -4 \\ 4 & -4 & 0 \\ -4 & 0 & 4 \end{bmatrix} + \delta^2 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 53 & -18 \\ 0 & -18 & 9 \end{bmatrix} \right),
$$

where

$$
\delta := \frac{\sqrt{13b_2 - 49\mu} - 2\sqrt{\mu}}{13b_2 - 53\mu} \sqrt{\mu}, \qquad \varphi := \frac{b_2}{1 - 4\delta + 53\delta^2}.
$$

The Cholesky factor U of X is

$$
U = \sqrt{\varphi} \begin{bmatrix} 2 & -1 + 2\delta & -1 - 2\delta \\ 0 & 7\delta & -2\delta \\ 0 & 0 & \delta \end{bmatrix}. \tag{17}
$$

The upper-left entry in the 2×2 matrix $A\Pi A^{\mathrm{T}} = A(X \otimes X)A^{\mathrm{T}}$ is:

$$
\operatorname{tr} A_1 X A_1 X = \operatorname{tr} ((A_1 X)^2) = 1224(\delta^2 \varphi)^2 = O(\mu^2). \tag{18}
$$

Figure 2 shows the relative errors in computing this quantity for $b_2 = 1000\pi$ and a gap $10^{-10} \leq \nu\mu \leq 1$.

The numerical results indicate that much more accurate results can be obtained by using the Cholesky factor U than by using X itself. In Figure 2, the solid horizontal line is drawn at a relative error of 10^{-1}, i.e. merely one significant digit. This critical line is crossed at $\nu\mu \approx 10^{-5}$ for relerr(X) and $\nu\mu \approx 10^{-10}$ for relerr(U). We give a theoretical explanation for this phenomenon below.

3.2 Analysis of the numerical error

A crucial quantity in analyzing floating point arithmetic is the so-called *unit roundoff* or *machine epsilon*, denoted by ϵ. In double precision IEEE floating point arithmetic, we have $\epsilon \approx 10^{-16}$, and the range of the floating point system is $[2^{-1021}, 2^{1024}]$. For $x \in \Re$ with $2^{-1021} \leq |x| \leq 2^{1024}$, we have $\mathrm{fl}(x) = (1 + \delta)x$ for some $|\delta| \leq \epsilon$. Similarly, for any two floating point numbers x and y, we have

$$\mathrm{fl}(x \text{ operation } y) = (1 + \delta)(x \text{ operation } y) \text{ for some } |\delta| \leq \epsilon.$$

At this point, at is useful to recall some elementary results from numerical analysis. The numerical error in computing the inner product between two vectors $x, y \in \Re^n$ is

$$
|x^{\mathrm{T}} y - \mathrm{fl}(x^{\mathrm{T}} y)| \leq \gamma_n \sum_{i=1}^{n} |x_i||y_i|, \tag{19}
$$

where

$$
\gamma_n \leq 1.01 \, n \, \epsilon \text{ for } n = 1, 2, \ldots, \left\lfloor \frac{0.01}{\epsilon} \right\rfloor. \tag{20}
$$

9

Using the Cauchy-Schwarz inequality, one further has that

$$|x^{\mathrm{T}}y - \mathrm{fl}(x^{\mathrm{T}}y)| \leq \gamma_n \|x\|_2 \|y\|_2.$$

Similarly, for $A \in \Re^{m \times n}$, $B \in \Re^{n \times q}$ and $C \in \Re^{m \times q}$ with $C = AB$, one has

$$\|\hat{C} - C\|_p \leq \gamma_n \|A\|_p \|B\|_p \text{ for } p = 1, 2, F, \tag{21}$$

where

$$\hat{c}_{ij} := \mathrm{fl}(\sum_{k=1}^{n} a_{ik} b_{kj}).$$

See Chapter 3 in Higham [11] for details.

In the following, we are interested in the dependence of numerical errors with respect to the duality gap $\nu\mu$ and the unit roundoff ϵ. This allows us to treat n and ν as constants, so that $\gamma_n = O(\epsilon)$ and $\gamma_\nu = O(\epsilon)$.

Theorem 1 *Let M be a given $\nu \times \nu$ matrix, let $\{X^{(1)}, X^{(2)}, \ldots\} \subset \Re^{\nu \times \nu}$ be a bounded sequence, i.e. $\|X^{(k)}\| = O(1)$, and let $\tilde{\epsilon} \geq \epsilon$ be a given scalar constant. Let $\rho_k := \mathrm{tr}\,(MX^{(k)})^2$. If $\hat{X}^{(1)}, \hat{X}^{(2)}, \ldots$ is a sequence with*

$$\|\hat{X}^{(k)} - X^{(k)}\| = O(\tilde{\epsilon})$$

then

$$\hat{\rho}_k = \rho_k + O\left(\|MX^{(k)}\|\tilde{\epsilon} + \tilde{\epsilon}^2\right), \tag{22}$$

where $\hat{\rho}_k$ is the result of the floating point computations:

$$\hat{T}^{(k)} = \mathrm{fl}(M\hat{X}^{(k)}), \quad \hat{\rho}_k = \mathrm{fl}\left(\sum_{i=1}^{\nu}\sum_{j=1}^{\nu} \hat{t}_{ij}^{(k)} \hat{t}_{ji}^{(k)}\right).$$

Proof. Let $E^{(k)} := \hat{X}^{(k)} - X^{(k)}$ and $T^{(k)} := MX^{(k)}$. Applying (21) we have $\|\hat{T}^{(k)} - M\hat{X}^{(k)}\| = O(\epsilon)$. Furthermore, $M\hat{X}^{(k)} = T^{(k)} + ME^{(k)}$ with $\|E^{(k)}\| = O(\tilde{\epsilon})$. Therefore,

$$\|\hat{T}^{(k)} - T^{(k)}\| = O(\tilde{\epsilon}).$$

We conclude using (19) that

$$
\begin{aligned}
\hat{\rho}_k &= \mathrm{fl}\left(\sum_{i=1}^{\nu}\sum_{j=1}^{\nu} \hat{t}_{ij}^{(k)} \hat{t}_{ji}^{(k)}\right) \\
&= \mathrm{tr}\,(\hat{T}^{(k)})^2 + O(\epsilon\|\hat{T}^{(k)}\|^2) \\
&= \rho_k + O\left(\|T^{(k)}\|\tilde{\epsilon} + \tilde{\epsilon}^2\right).
\end{aligned}
$$

Q.E.D.

10

In any feasible interior point method, the primal iterates X are bounded, so that $\|\text{fl}(X) - X\| = O(\epsilon)$. In particular, we may apply the above theorem to the example of Section 3.1 with $\hat{X} = \text{fl}(X)$, $\tilde{\epsilon} = \epsilon$, and $M = A_1$. Furthermore, we have in this example that

$$\|A_1 X(\mu)\| = O(\mu), \quad \|A_1 X\| = O(\sqrt{\mu}).$$

Therefore, relation (22) yields

$$\begin{cases} \text{comp} \left(\text{ tr } (A_1 X(\mu))^2 \right) - \text{ tr } (A_1 X(\mu))^2 = O(\mu\epsilon + \epsilon^2) \\ \text{comp} \left(\text{ tr } (A_1 X)^2 \right) - \text{ tr } (A_1 X)^2 = O(\sqrt{\mu}\epsilon + \epsilon^2), \end{cases}$$

where 'comp$(\cdot)$' stands for the calculated version, as outlined by the computation of $\hat{\rho}_k$ in Theorem 1. Referring to (15) and using the fact that

$$\text{tr } (A_1 X(\mu))^2 = 8\mu^2, \quad \frac{1}{\text{tr } (A_1 X)^2} = \frac{1}{1224(\delta^2\varphi)^2} = O(\frac{1}{\mu^2}),$$

the relative errors based on $X(\mu)$ and X with $\mu > \epsilon$ are thus as follows:

$$\text{relerr}(X(\mu)) = O(\frac{\epsilon}{\mu}), \quad \text{relerr}(X) = O(\frac{\epsilon}{\mu\sqrt{\mu}}).$$

Indeed, in our numerical example we have

$$\text{relerr}(X(\mu)) \approx \frac{10^{-11}}{\mu}, \quad \text{relerr}(X) \approx \frac{10^{-9}}{\mu\sqrt{\mu}}.$$

Theorem 2 *Let $M = M^T$ be a given $\nu \times \nu$ symmetric matrix, let $\{F^{(1)}, F^{(2)}, \ldots\} \subset \Re^{p\times\nu}$ and $\{G^{(1)}, G^{(2)}, \ldots\} \subset \Re^{\nu\times q}$ be bounded sequences, i.e. $\|F^{(k)}\| = O(1)$ and $\|G^{(k)}\| = O(1)$, and let $\epsilon_k^f \geq \epsilon$ and $\epsilon_k^g \geq \epsilon$ be given scalar constants. Let $\rho_k := \|F^{(k)} M G^{(k)}\|_F^2$. If $\hat{F}^{(1)}, \hat{F}^{(2)}, \ldots$ and $\hat{G}^{(1)}, \hat{G}^{(2)}, \ldots$ are sequences with*

$$\|\hat{F}^{(k)} - F^{(k)}\| = O(\epsilon_k^f), \quad \|\hat{G}^{(k)} - G^{(k)}\| = O(\epsilon_k^g) \tag{23}$$

then

$$\hat{\rho}_k - \rho_k = O(\eta_k \sqrt{\rho_k} + \eta_k^2 + \epsilon\rho_k), \tag{24}$$

where

$$\eta_k := \epsilon_k^f \|G^{(k)}\| + \epsilon_k^g \|F^{(k)}\| + \epsilon_k^f \epsilon_k^g,$$

and $\hat{\rho}_k$ is the result of the floating point computations:

$$\hat{P}^{(k)} = \text{fl}(\hat{F}^{(k)} M), \quad \hat{T}^{(k)} = \text{fl}(\hat{P}^{(k)} \hat{G}^{(k)}), \quad \hat{\rho}_k = \text{fl}(\sum_{i,j} (\hat{t}_{ij}^{(k)})^2).$$

Furthermore, it holds that $\eta_k = O(\epsilon_k^f + \epsilon_k^g)$.

Proof. Let $P^{(k)} := F^{(k)} M$ and $T^{(k)} := F^{(k)} M G^{(k)}$. Applying (21), we have that $\hat{P}^{(k)} - \hat{F}^{(k)} M =$

$O(\epsilon \|\hat{F}^{(k)}\|)$. Since $\|\hat{F}^{(k)} - F^{(k)}\| = O(\epsilon_k^f)$ we also have that $\hat{F}^{(k)}M - P^{(k)} = O(\hat{\epsilon}_1)$. Using the boundedness of the sequence $F^{(1)}, F^{(2)}, \dots$, it thus follows that

$$\|\hat{P}^{(k)} - P^{(k)}\| = O(\hat{\epsilon}_1). \tag{25}$$

Similarly, we have from (21) that $\hat{T}^{(k)} - \hat{P}^{(k)}\hat{G}^{(k)} = O(\epsilon \|\hat{P}^{(k)}\|\|\hat{G}^{(k)}\|)$. Since $\|P^{(k)}\| = \|F^{(k)}M\| = O(\|F^{(k)}\|)$, we have from (25) and (23) that

$$\begin{cases} \|\hat{P}^{(k)}\hat{G}^{(k)} - T^{(k)}\| = O(\epsilon_k^f\|G^{(k)}\| + \epsilon_k^g\|F^{(k)}\| + \epsilon_k^f\epsilon_k^g) = O(\eta_k) \\[2mm] \|\hat{P}^{(k)}\|\|\hat{G}^{(k)}\| - \|P^{(k)}\|\|G^{(k)}\| = O(\epsilon_k^f\|G^{(k)}\| + \epsilon_k^g\|F^{(k)}\| + \epsilon_k^f\epsilon_k^g) = O(\eta_k) \end{cases} \tag{26}$$

It follows that

$$\|\hat{T}^{(k)} - T^{(k)}\| \le \|\hat{T}^{(k)} - \hat{P}^{(k)}\hat{G}^{(k)}\| + \|\hat{P}^{(k)}\hat{G}^{(k)} - T^{(k)}\| = O(\eta_k + \epsilon(\|F^{(k)}\|\|G^{(k)}\|)) = O(\eta_k),$$

so that also

$$\|\hat{T}^{(k)}\|_F - \sqrt{\rho_k} = O(\eta_k). \tag{27}$$

Finally, we have from (19) that

$$\hat{\rho}_k - \|\hat{T}^{(k)}\|_F^2 = O(\epsilon \|\hat{T}^{(k)}\|^2)$$

which together with (27) yields

$$\hat{\rho}_k - \rho_k = O(\eta_k\sqrt{\rho_k} + \eta_k^2 + \epsilon\rho_k).$$

Q.E.D.

We may apply the above theorem to the example of Section 3.1 with

$$M = A_1, \ F = G^{\mathrm{T}} = U, \ \hat{F} = \hat{G}^{\mathrm{T}} = \mathrm{fl}(U), \ \epsilon_k^f = \epsilon_k^g = \epsilon,$$

and U as in (17). By choosing a sequence $\mu_1 > \mu_2 > \cdots > 0$, a sequence of Us is implied. We obtain from (24) and (18) that

$$\mathrm{comp}\left(\|UA_1U^{\mathrm{T}}\|_F^2\right) - \|UA_1U^{\mathrm{T}}\|_F^2 = O(\mu\epsilon + \epsilon^2).$$

Referring to Definition 16, the relative error based on U with $\mu > \epsilon$ can thus be estimated as

$$\mathrm{relerr}(U) = O(\frac{\epsilon}{\mu}).$$

Indeed, in Figure 2 we have

$$\mathrm{relerr}(U) \approx \frac{10^{-11}}{\mu}.$$

We need the more analysis to explain the phenomenon in Figure 1. The matrix $U(\mu)$ in (refexdefUmu) is naturally partitioned as

$$U_1(\mu) = \left[\begin{array}{ccc} u_{11}(\mu) & u_{12}(\mu) & u_{13}(\mu) \end{array} \right], U_2(\mu) = \left[\begin{array}{ccc} 0 & u_{22}(\mu) & u_{23}(\mu) \\ 0 & 0 & u_{33}(\mu) \end{array} \right],$$

12

so that
$$\|U_1(\mu)\| = O(1), \quad \|U_2(\mu)\| = O(\sqrt{\mu}). \tag{28}$$
Furthermore, we have in this example that
$$\|U_1(\mu)A_1U_1(\mu)^{\mathrm{T}}\| = O(\mu^2), \quad \|U_1(\mu)A_1U_2(\mu)^{\mathrm{T}}\| = O(\mu\sqrt{\mu}), \quad \|U_2(\mu)A_1U_2(\mu)^{\mathrm{T}}\| = O(\mu). \tag{29}$$

Applying Theorem 2 with
$$M = A_1, \; F^{(k)} = (G^{(k)})^{\mathrm{T}} = U_1(\mu_k), \; \hat{F}^{(k)} = \mathrm{fl}(F^k), \; \hat{G}^{(k)} = \mathrm{fl}(G^{(k)}), \; \epsilon_k^f = \epsilon_k^g = \epsilon,$$
yields for $\rho_{11}^{(k)} := \|U_1(\mu_k)^{\mathrm{T}}A_1U_1(\mu_k)\|_F^2 = O(\mu_k^4)$ that
$$\hat{\rho}_{11}^{(k)} - \rho_{11}^{(k)} = O(\epsilon\mu_k^2 + \epsilon^2). \tag{30}$$

Setting
$$G^{(k)} = U_2(\mu_k)^{\mathrm{T}}, \; \epsilon_k^g = \sqrt{\mu_k}\epsilon$$
yields for $\rho_{12}^{(k)} := \|U_1(\mu_k)^{\mathrm{T}}A_1U_2(\mu_k)\|_F^2 = O(\mu_k^3)$ that $\eta_k = O(\epsilon\sqrt{\mu_k})$ and
$$\hat{\rho}_{12}^{(k)} - \rho_{12}^{(k)} = O(\epsilon\mu_k^2 + \epsilon^2\mu_k). \tag{31}$$
Keeping this choice of $G^{(k)}$ and setting also
$$F^{(k)} = U_2(\mu_k), \; \epsilon_k^f = \sqrt{\mu_k}\epsilon$$
yields for $\rho_{22}^{(k)} := \|U_2(\mu_k)^{\mathrm{T}}A_1U_2(\mu_k)\|_F^2 = O(\mu_k^2)$ that $\eta_k = O(\epsilon\mu_k)$ and
$$\hat{\rho}_{22}^{(k)} - \rho_{22}^{(k)} = O(\epsilon\mu_k^2). \tag{32}$$

Combining relations (30)–(32) yields
$$\mathrm{comp}\left(\|U(\mu_k)A_1U(\mu_k)^{\mathrm{T}}\|_F^2\right) = \mathrm{fl}(\hat{\rho}_{11}^{(k)} + 2\hat{\rho}_{12}^{(k)} + \hat{\rho}_{22}^{(k)}) = \|U(\mu)A_1U(\mu)^{\mathrm{T}}\|_F^2 + O(\mu^2\epsilon + \epsilon^2).$$

Referring to Definition 16, the relative error based on $U(\mu)$ is thus as follows:
$$\mathrm{relerr}(U(\mu)) = O(\epsilon + \frac{\epsilon^2}{\mu^2}).$$

Indeed, in Figure 1 we have
$$\mathrm{relerr}(U(\mu)) \approx 10^{-15} + \frac{10^{-23}}{\mu^2}.$$

3.3 Primal-Dual Scaling

In the preceding, we have illustrated the potential benefit of using a Cholesky form representation for Π. We have seen that for the primal scaling $\Pi = X \otimes X = (U_x \otimes U_x)^{\mathrm{T}}(U_x \otimes U_x)$, we need to represent the primal iterate X by its Cholesky factor U_x. Similarly, the dual scaling benefits from representing the dual iterate Z by its Cholesky factor U_z.

In the case of primal-dual HRVW-scaling, we need both U_x and U_z. Namely, we have $\Pi = (X \otimes Z^{-1} + Z^{-1} \otimes X)/2$ with

$$X \otimes Z^{-1} = (U_x^{\mathrm{T}} U_x) \otimes (U_z^{-1} U_z^{-\mathrm{T}}) = (U_x \otimes U_z^{-\mathrm{T}})^{\mathrm{T}}(U_x \otimes U_z^{-\mathrm{T}}).$$

The actual primal-dual iterate (X, Z) can be constructed from U_x and U_z in the obvious way.

For the case of NT-scaling, Figures 1–2 show the benefit of working with the Cholesky factor U_d of D; the associated Π matrix is then expressed as

$$\Pi = D \otimes D = (U_d \otimes U_d)^{\mathrm{T}}(U_d \otimes U_d) = U(\Pi)^{\mathrm{T}}U(\Pi),$$

where $U(\Pi) = (U_d \otimes U_d)$ is the Cholesky factor of Π. However, the iterates X and Z cannot be computed from U_d alone. The missing information is the so-called 'v-solution' [13]. We refer to (9) before actually defining the concept of v-solutions. If we pre-multiply on both sides of the identity in (9) by $U_d^{-\mathrm{T}}$ and post-multiply by U_d^{-1}, we obtain that $U_d Z U_d^{\mathrm{T}} = U_d^{-\mathrm{T}} X U_d^{-1}$. The resulting matrix is the Cholesky-based v-solution,

$$V := U_d Z U_d^{\mathrm{T}} = U_d^{-\mathrm{T}} X U_d^{-1}. \tag{33}$$

The pair (U_d, V) is known as the v-space Cholesky form, which was introduced for semidefinite programming by [30]. Obviously, the matrices X and Z can be constructed from the pair (U_d, V) using the formulas

$$X = U_d^{\mathrm{T}} V U_d, \ Z = U_d^{-1} V U_d^{-\mathrm{T}}.$$

The numerical benefit of having the pair $(\mathrm{fl}(U_d), \mathrm{fl}(V))$ instead of $(\mathrm{fl}(X), \mathrm{fl}(Z))$ is significant. Figures 1–2 already demonstrate the benefit of working with U_d instead of D. The numerical errors studied in that particular example were due to numerical cancellation in the $a_1^{\mathrm{T}} \Pi a_1$ computation. In NT-scaling however, there is a potentially more dangerous source of numerical problems. Namely, it is in general not possible to compute D accurately from nearly optimal X and Z. This means that *none* of the entries in the matrix $A\Pi A^{\mathrm{T}}$ can be computed accurately, if we have to compute D from nearly optimal X and Z. To see why D cannot be computed accurately from X and Z, observe that a sequence $(X^{(k)}, Z^{(k)})$, $k = 1, 2, \ldots$, approaches optimality if and only if

$$\mathrm{tr}\ X^{(k)} Z^{(k)} = \ \mathrm{tr}\ (X^{(k)})^{1/2} Z^{(k)} (X^{(k)})^{1/2} \to 0,$$

or equivalently,

$$(X^{(k)})^{1/2} Z^{(k)} (X^{(k)})^{1/2} \to 0.$$

Since the actual entries of X and Z do no approach zero in general, it follows that $\mathrm{fl}(X^{1/2} Z X^{1/2})$ is typically inaccurate due to numerical cancellation. Therefore, D cannot be computed accurately from (10). (The same is true for Cholesky-based variants of (10) as proposed in [33].) The effect is illustrated in Figure 3.
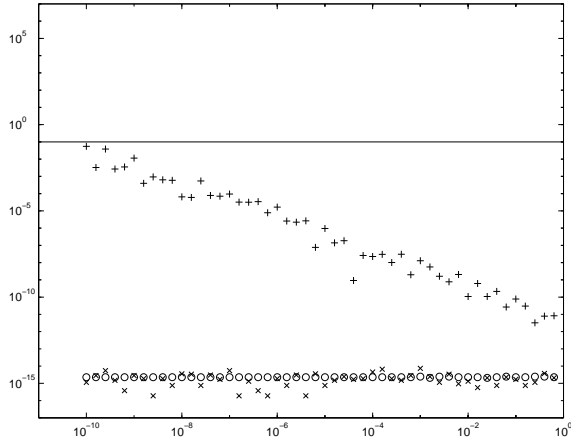
Figure 3: Relative errors for $\operatorname{tr} A_2 D(\mu) A_2 D(\mu)$ on central path with $b_2 = 1000\pi$. '×'=relerr($U_d(\mu)$), 'o'=relerr($D(\mu)$), '+'=relerr($\hat{D}(\mu)$), where $\hat{D}(\mu)$ is computed from $X(\mu)$ and $Z(\mu)$ by formula (10), using floating point arithmetic. The x-axis is the gap $\nu\mu$.
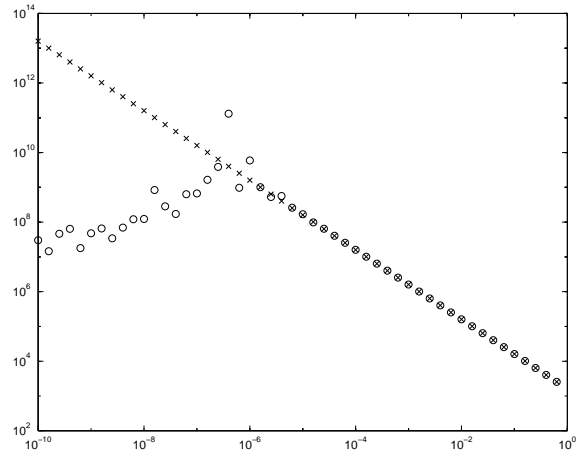
Figure 4: Instability of Cholesky factor without pivoting. Shown is $(a_1^{\mathrm{T}}\Pi a_2)/(a_1^{\mathrm{T}}\Pi a_1)$ with $\Pi = X \otimes X$ outside central path, on the example with $b_2 = 1000\pi$. '×'=computations based on U, 'o'=computations based on X. The x-axis is the gap $\nu\mu$.

3.4 Stable Cholesky factors

The numerical need for pivoting in the Cholesky factorization of a positive definite matrix is often denied. Indeed, the $A\Pi A^{\mathrm{T}}$ matrix can safely be factored with an arbitrary pivot ordering for linear programming, even though $A\Pi A^{\mathrm{T}}$ becomes ill conditioned [38]. Unfortunately, the Cholesky factorization without numerical pivoting can lead to highly unstable factors in the setting of semidefinite programming.

Definition 1 *A $\nu \times \nu$ upper triangular matrix U is called a stable U-factor if for all rows $i = 1, 2, \ldots, \nu$ it holds that*

$$u_{ii} \geq \max\{|u_{ij}| \mid j = i+1, i+2, \ldots, \nu\}. \tag{34}$$

See Lemma 8.6 in Higham [11] for a motivation of the above definition. Loosely speaking, if the quantity

$$\max_{i,j} \frac{|u_{ij}|}{u_{ii}} \tag{35}$$

is large, then forward and backward solving procedures will suffer from excessive error propagation.

Consider now the example of Section 3.1. The unmodified pivot ordering $(1,1) \to (2,2)$ of the 2×2 positive definite matrix $A\Pi A^{\mathrm{T}}$ can lead to an undesirably large quantity $|u_{12}|/u_{11}$ for iterates outside the central path, as shown in Figure 4. (Observe that $u_{12}/u_{11} = \operatorname{tr} A_1 X A_2 X / \operatorname{tr}(A_1 X)^2$.)

4 Maintaining the v-Space Cholesky Form

It is clear that the Nesterov-Todd scaling point D can easily be computed from the V-space factor U_d, viz. $D = U_d^{\mathrm{T}} U_d$. Now suppose that we can solve system (12) accurately. It is then straightforward to compute the primal and dual directions ΔX and ΔZ, and the new iterate (X^+, y^+, Z^+) as in (6)–(7). However, we will then face numerical problems in the next iterate, since the new NT-scaling D^+ cannot be accurately computed from the floating point representations of the entries in ill-conditioned matrices X^+ and Z^+, as discussed before. Therefore, we propose to take the step in a scaled space, in which the primal and dual solutions remain (locally) well conditioned. Namely, the scaling maps the current primal and dual solutions onto the Cholesky-based v-solution as in (33). Since $XZ = U_d^{\mathrm{T}} V^2 U_d^{-\mathrm{T}}$, the matrices XZ and V^2 share the same spectrum of eigenvalues. The eigenvalues within this spectrum are all of the same order of magnitude due to the centrality property (5). On the central path, V is a multiple of the identity matrix and hence ideally conditioned.

The scaled search direction, denoted $(\overline{\Delta X}, \overline{\Delta Z})$, is implicitly defined as

$$U_d^{\mathrm{T}} \overline{\Delta X} U_d = \Delta X, \quad \overline{\Delta Z} = U_d \Delta Z U_d^{\mathrm{T}}.$$

In fact, since we use only the scaled search directions, it is not necessary to compute ΔX. Namely, once Δy and ΔZ have been computed we let $\overline{\Delta Z} = U_d \Delta Z U_d^{\mathrm{T}}$. Then, instead of solving Δx from

$$\Delta x + \Pi \Delta z = -x$$

as in (8), we solve $\overline{\Delta X}$ from

$$\overline{\Delta X} + \overline{\Delta Z} = -V.$$

Given the scaled search direction, we arrive at the new iterate under the current scaling,

$$\bar{X}^+ = V + t\overline{\Delta X}, \ \bar{Z}^+ = V + t\overline{\Delta Z}.$$

Since V is well conditioned, we can in principle control the step length t in such a way that $\bar{X}^+$ and $\bar{Z}^+$ are also well conditioned. However, in practice we will give priority to fast convergence, so that the conditioning of $\bar{X}^+$ and $\bar{Z}^+$ can be poor on those iterates where the rate of linear convergence is close to zero. The procedure for updating U_d will then be less accurate. Such inaccuracy can possibly lead to larger residual vectors that are included in a self-dual or infeasible interior point formulation; however, the increase will usually be offset by a reduction in the same residuals that is caused by the small (hence fast) rate of linear convergence.

The procedure to update the U_d and V factors is as follows:

Procedure 1 (Update of v-factors)

Input: $\bar{X}^+$, $\bar{Z}^+$, U_d.

Output: Upper triangular matrix U_d^+ and positive definite matrix V^+ such that for
$$X^+ := U_d^T \bar{X}^+ U_d, \quad Z^+ := U_d^{-1} \bar{Z}^+ U_d^{-T},$$
we have
$$X^+ = (U_d^+)^T V^+ U_d^+, \quad V^+ = U_d^+ Z^+ (U_d^+)^T.$$

1. *Compute an upper triangular matrix $\bar{U}_x^+$ as the Cholesky factorization of $\bar{X}^+$, i.e.*

$$\bar{X}^+ = (\bar{U}_x^+)^T \bar{U}_x^+.$$

2. *Let $W := \bar{U}_x^+ \bar{Z}^+ (\bar{U}_x^+)^T$; compute an orthogonal matrix Q_w and a diagonal matrix Λ_v^+ as the symmetric eigenvalue decomposition of $W^{1/2}$, i.e.*

$$W = Q_w (\Lambda_v^+)^2 Q_w^T.$$

3. *Let $T := (\Lambda_v^+)^{-1/2} Q_w^T$; compute an orthogonal matrix Q_v^+ and an upper triangular matrix R as the QR-factorization of T, i.e.*

$$T = (Q_v^+)^T R.$$

4. *OUTPUT:*

$$U_d^+ = R \bar{U}_x^+ U_d,$$
$$V^+ = Q_v^+ \Lambda_v^+ (Q_v^+)^T.$$

The above procedure does not guarantee in itself that U_d^+ is a stable upper triangular factor; hence a re-ordering of rows and columns may be necessary. A few Givens rotations will make the U_d^+-matrix upper-triangular in the new pivot ordering; see e.g. Dennis and Schnabel [5] for this technique.

Theorem 3 *Procedure 1 is correct.*

Proof.   The proof is straightforward. We have

$$R^{\mathrm{T}} V^+ R = T^{\mathrm{T}} \Lambda_v^+ T = Q_w Q_w^{\mathrm{T}} = I,$$

so that

$$(U_d^+)^{\mathrm{T}} V^+ U_d^+ = U_d^{\mathrm{T}} (\bar{U}_x^+)^{\mathrm{T}} \bar{U}_x^+ U_d = X^+.$$

Furthermore, we have

$$W = Q_w (\Lambda_v^+)^2 Q_w^{\mathrm{T}} = T^{-1} \Lambda_v^+ T^{-\mathrm{T}},$$

so that

$$U_d^+ Z^+ (U_d^+)^{\mathrm{T}} = R \bar{U}_x^+ \bar{Z}^+ (\bar{U}_x^+)^{\mathrm{T}} R^{\mathrm{T}} = R W R^{\mathrm{T}} = R T^{-1} \Lambda_v^+ T^{-\mathrm{T}} R^{\mathrm{T}} = V^+,$$

where we used that $R T^{-1} = R((Q_v^+)^{-1} R)^{-1} = Q_v^+$.   **Q.E.D.**

5 Second Order Cone Programming

A second order cone program is an optimization problem of the form (2), where the $\text{PSD}(\nu)$-cone is replaced by a Cartesian product of second order cones. The second order cone in $\Re^n$ is formally defined as

$$\text{SOC}(n) := \left\{ x \in \Re^n \;\middle|\; x_1 \geq \sqrt{\sum_{i=2}^{n} x_i^2} \right\}. \tag{36}$$

For an introduction to second order cone programming, we refer to Lobo et al. [16] and Tsuchiya [36].

Exactly as in semidefinite programming, it is possible that a vector approaches the boundary of the second order cone without any of its components approaching zero. Consider for instance the sequence $\{x^{(1)}, x^{(2)}, \ldots\} \subset \text{SOC}(3)$ defined as

$$x^{(k)} = \begin{bmatrix} 5 + 1/k \\ -4 \\ 3 \end{bmatrix}.$$

In fact, calculating the constraint violation

$$x_1^{(k)} - \sqrt{(x_2^{(k)})^2 + (x_3^{(k)})^2} = (5 + 1/k) - 5$$

already leads to numerical cancellation as $k \to \infty$. This phenomenon is typical when optimality is approached in a second order cone program. In this section, we propose a procedure for updating a v-space Cholesky form for second order cone programming, which avoids this type of numerical cancellation.

Standard notation from Euclidean Jordan algebra will be used throughout this section. An excellent reference on this topic is Faraut and Korányi [6]. We will also use the similarity relations as developed in [27].

Define a vector $\iota \in \Re^n$ as
$$\iota_1 = \sqrt{2}, \; \iota_2 = \iota_3 = \cdots = \iota_n = 0;$$
this vector is known as the identity of the $\text{SOC}(n)$ Jordan algebra, We let $J := \iota\iota^{\mathrm{T}} - I \in \Re^{n \times n}$. Any vector $x \in \Re^n$ has two spectral values, characterized by their sum $\text{tr}\,x$ and product $\det(x)$. We have

$$\text{tr}\,x = \iota^{\mathrm{T}}x, \quad \det(x) = \frac{x^{\mathrm{T}}Jx}{2} = \frac{(\text{tr}\,x)^2 - \|x\|_2^2}{2}. \tag{37}$$

Consequently, the two spectral values of ι are both 1. The Jordan product operator $L(x) \in \Re^{n \times n}$ is defined as

$$L(x) = \frac{1}{2}(\iota x^{\mathrm{T}} + x\iota^{\mathrm{T}} - (\text{tr}\,x)J);$$

observe that $L(\iota) = I$. The quadratic representation $P(x)$ is

$$P(x) = xx^{\mathrm{T}} - \det(x)J.$$

We let $x^2 := L(x)x$. It is easily verified that

$$x^2 = L(x)x = P(x)\iota,$$

18

and
$$\operatorname{tr} x^2 = \|x\|_2^2 = (\operatorname{tr} x)^2 - 2\det(x), \quad \det(x^2) = \det(x)^2. \tag{38}$$

Therefore, if λ_1 and λ_2 are the two spectral values of x, then λ_1^2 and λ_2^2 are the two spectral values of x^2. Given $x \in \operatorname{SOC}(n)$, there is a unique vector $x^{1/2} \in \operatorname{SOC}(n)$ such that $(x^{1/2})^2 = x$. Namely

$$x^{1/2} = \frac{1}{\operatorname{tr} x^{1/2}} (x + \det(x^{1/2})\iota),$$

where, due to (38),

$$\det(x^{1/2}) = \sqrt{\det(x)}, \quad \operatorname{tr} x^{1/2} = \sqrt{\operatorname{tr} x + 2\det(x^{1/2})}.$$

Any $x \in \Re^n$ with $\det(x) \neq 0$ has an inverse

$$x^{-1} = \frac{1}{\det(x)} Jx, \tag{39}$$

so that $L(x)x^{-1} = \iota$.

Some standard relations are:

$$\det(P(x)y) = \det(x)^2 \det(y), \tag{40}$$

$$P(x)^{-1} = P(x^{-1}) \text{ with } \det(x) \neq 0, \tag{41}$$

$$(P(x)y)^{-1} = P(x^{-1})y^{-1} \text{ with } \det(x) \neq 0, \det(y) \neq 0, \tag{42}$$

and

$$P(x)P(y)P(x) = P(P(x)y), \tag{43}$$

see [6], Propositions III.4.2, II.3.1, and II.3.3. Applying (43) with $y = \iota$ yields

$$P(x)^2 = P(x^2). \tag{44}$$

Based on the above notation, we can now define the meaning of Π in the setting of second order cone programming. In general, Π is block diagonal, where each $\operatorname{SOC}(n)$-constraint has a corresponding $n \times n$ diagonal block. In this section, we focus on an individual $\operatorname{SOC}(n)$ constraint with its diagonal block Π.

	Π	r
primal-only	$P(x)$	$-P(x)z$
dual-only	$P(z^{-1})$	$-x$
primal-dual	$\Pi z = x$	$-x$.

Again, there are various possible choices for Π in the setting of primal-dual scaling, see e.g. Tsuchiya [36]. A possible choice of Π is $\Pi = L(x)L(z)^{-1}$, resulting in the so-called AHO-direction [3]. The NT-direction corresponds to $\Pi = P(d)$, where d is characterized by

$$d \in \operatorname{SOC}(n), \quad P(d)z = x, \tag{45}$$

19

for x and z in the interior of the second order cone. An explicit formula for d is:

$$d = P(x^{1/2})(P(x^{1/2})z)^{-1/2}. \tag{46}$$

The second order cone has a much simpler structure than the positive semi-definite cone. Since in particular a vector has merely two spectral values, one needs only one extra scalar to compute x^{-1} from x accurately, namely $\det(x)$. Having $\det(x)$ also allows us to compute $P(x)u$ accurately for any $u \in \Re^n$. In the case of primal-only scaling, it is therefore a good idea to update $\det(x)$ in each iteration using the product formula (40). A dual-only algorithm should maintain $\det(z)$ in a similar fashion. In the setting of NT-scaling, we prefer to work with the v-factors $(d, \det(d), v)$ instead of the actual iterates x and z. The v-solution associated with x and z is defined as

$$v = P(d^{1/2})z = P(d^{-1/2})x. \tag{47}$$

We remark that we define the v-solution for second order cone programming based on the symmetric square root $P(d^{1/2})$ of Π, whereas the Cholesky v-solution in semidefinite programming is based on the upper triangular factor $U(\Pi)$.

Two vectors (matrices) are said to be similar if they share the same set of spectral values (eigenvalues). In the setting of second order cones, we write $x \sim y$ if $\operatorname{tr} x = \operatorname{tr} y$ and $\det(x) = \det(y)$. Relation (40) in Sturm [27] states for x, z, d and v as in (46) and (47) that

$$v \sim (P(x)^{1/2}z)^{1/2}. \tag{48}$$

Corollary 1 in [27] also states for arbitrary $x, y \in \mathrm{SOC}(n)$, $z \in \Re^n$ that

$$P(x^{1/2})P(y^{1/2})z \sim P(P(y^{1/2})x)^{1/2}z. \tag{49}$$

Letting $\tilde{z} = P(y^{1/2})z$, one can rewrite (49) for $\det(y) \neq 0$ as

$$P(x^{1/2})\tilde{z} \sim P(P(y^{1/2})x)^{1/2}P(y^{-1/2})\tilde{z}. \tag{50}$$

As discussed in Section 4, the v-space version of the interior point method solves the scaled (predictor) search direction from

$$\overline{\Delta x} + \overline{\Delta z} = -v,$$

together with the feasibility requirements

$$AP(d^{1/2})\overline{\Delta x} = 0, \quad P(d^{1/2})A^{\mathrm{T}}\Delta y + \overline{\Delta z} = 0.$$

Given the scaled search direction, we arrive at the new iterate under the current scaling,

$$\bar{x}^+ = v + t\overline{\Delta x}, \ \bar{z}^+ = v + t\overline{\Delta z}.$$

Due to the centrality condition (5), v is well-conditioned; an appropriate condition measure with respect to $\mathrm{SOC}(n)$ is $(\operatorname{tr} v)^2/\det(v)$. The scaled iterates $\bar{x}^{\mathrm{new}}$ and $\bar{z}^+$ are also much better conditioned than their unscaled counterparts. However, the scaling vector d can become increasingly ill-conditioned. Straightforward use of formulas (46) and (47) can therefore quickly result in massive numerical cancellation. The following procedure effectively avoids such numerical cancellation.

Procedure 2 (Update of SOC v-factors)

Input: $\bar{x}^+$, $\bar{z}^+$, d, $\det(d)$.

Output: d^+, $\det(d^+)$, v^+ such that for

$$x^+ := P(d^{1/2})\bar{x}^+, \quad z^+ := P(d^{-1/2})\bar{z}^+,$$

we have

$$x^+ = P(d^+)^{1/2}v^+, \quad v^+ = P(d^+)^{1/2}z^+.$$

1. *Let*
$$\begin{cases} \det(v^+) = \sqrt{\det(\bar{x}^+)\det(\bar{z}^+)} \\ \operatorname{tr} v^+ = \sqrt{(\bar{x}^+)^T\bar{z}^+ + 2\det(v^+)}. \end{cases}$$

2. *Let*
$$\chi = \frac{1}{\operatorname{tr} v^+}(\bar{x}^+ + \det(v^+)(\bar{z}^+)^{-1}), \quad \det(\chi) = \frac{\det(v^+)}{\det(\bar{z}^+)},$$

   and compute
$$d^+ = P(d^{1/2})\chi, \quad \det(d^+) = \det(d)\det(\chi).$$

3. *Let*
$$\psi = \bar{x}^+ - \det(v^+)(\bar{z}^{\mathrm{new}})^{-1}, \quad \alpha = \frac{d^T\psi}{(\operatorname{tr}(d^+)^{1/2})^2},$$

$$\phi = \frac{1}{2\sqrt{\det(\chi)}}(\psi - \alpha\chi), \quad \gamma = \frac{\alpha + \operatorname{tr}\phi}{(\operatorname{tr} d^{1/2})^2},$$

   and compute
$$\begin{cases} v_1^+ = \operatorname{tr} v^+/\sqrt{2} \\ v_i^+ = \phi_i + \gamma d_i \text{ for } i = 2, 3, \dots, n. \end{cases}$$

Theorem 4 *Procedure 2 is correct.*

Proof. Let
$$x^+ := P(d^{1/2})\bar{x}^+, \quad z^+ := P(d^{-1/2})\bar{z}^+.$$
Using (48) and (50), we have
$$(v^+)^2 \sim P(x^+)^{1/2}z^+ \sim P(\bar{x}^+)^{1/2}\bar{z}^+.$$
This implies that
$$\operatorname{tr}(v^+)^2 = \iota^T P(\bar{x}^+)^{1/2}\bar{z}^{\mathrm{new}} = (\bar{x}^+)^T\bar{z}^+, \tag{51}$$
and, using (40),
$$\det(v^+)^2 = \det(P(\bar{x}^+)^{1/2}\bar{z}^{\mathrm{new}}) = \det(\bar{x}^+)\det(\bar{z}^+). \tag{52}$$
The correctness of Step 1 now follows immediately.

From the definition (37), we have that

$$\det(\chi) = \frac{(\bar{x}^+ + \det(v^+)(\bar{z}^+)^{-1})^{\mathrm{T}} J(\bar{x}^+ + \det(v^+)(\bar{z}^+)^{-1})}{2(\operatorname{tr} v^+)^2}.$$

Using (37) and (39), it further follows that

$$\det(\chi) = \frac{\det(\bar{x}^+) + \det(v^+)^2/\det(\bar{z}^{\mathrm{new}}) + (\det(v^+)/\det(\bar{z}^+))(\bar{x}^{\mathrm{new}})^{\mathrm{T}}\bar{z}^+}{(\operatorname{tr} v^+)^2}.$$

Applying (51)–(52) to the above relation yields

$$\det(\chi) = \frac{2\det(v^+)^2 + \det(v^+)\operatorname{tr}(v^{\mathrm{new}})^2}{(\operatorname{tr} v^+)^2 \det(\bar{z}^+)}.$$

Therefore, using (38) we obtain

$$\det(\chi) = \frac{\det(v^+)}{\det(\bar{z}^+)}. \tag{53}$$

We shall now show that $x^+ = P(d^+)z^+$. We have, using the definition of d^+ and relation (43) that

$$P(d^+)z^+ = P(P(d^{1/2})\chi)z^+ = P(d^{1/2})P(\chi)P(d^{1/2})z^+.$$

Together with the definition of z^+, this yields

$$P(d^+)z^+ = P(d^{1/2})P(\chi)\bar{z}^+.$$

Using the definition of $P(\cdot)$ we have

$$P(\chi)\bar{z}^+ = (\chi^{\mathrm{T}}\bar{z}^+)\chi - \det(\chi)\det(\bar{z}^+)(\bar{z}^+)^{-1}.$$

Moreover, using the definition of χ and relations (51) and (38), we have

$$\chi^{\mathrm{T}}\bar{z}^+ = \frac{\operatorname{tr}(v^+)^2 + 2\det(v^+)}{\operatorname{tr} v^+} = \operatorname{tr} v^+.$$

Hence,

$$P(\chi)\bar{z}^+ = (\bar{x}^+ + \det(v^+)(\bar{z}^+)^{-1}) - \det(v^+)(\bar{z}^+)^{-1} = \bar{x}^+.$$

It thus follows that

$$P(d^+)z^+ = P(d^{1/2})P(\chi)\bar{z}^+ = P(d^{1/2})\bar{x}^+ = x^+,$$

completing the correctness proof of Step 2.

The correctness of v_1^+ in Step 3 is obvious. For the remaining entries v_i^+, $i = 2, 3, \ldots, n$, observe that

$$v^+ = -(Jv^+)_i = -\det(v^+)((v^{\mathrm{new}})^{-1})_i.$$

Since $v^+ = P((d^+)^{1/2})z^+$, we have using (42) that

$$(v^+)^{-1} = P((d^+)^{-1/2})(z^+)^{-1} = P((d^+)^{-1/2})P(d^{1/2})(\bar{z}^+)^{-1}.$$

But also,
$$v^+ = P((d^+)^{-1/2})P(d^{1/2})\bar{x}^+.$$

Hence,
$$v_i^+ = \frac{1}{2}\left(P((d^+)^{-1/2})P(d^{1/2})\psi\right)_i.$$

After some tedious but straightforward calculations, it further follows that
$$v_i^+ = \phi_i + \gamma d_i.$$

<div align="right">**Q.E.D.**</div>

After this intermezzo on second order cone programming, we return to the study of the semidefinite program (1) in the next section.

6 Solving the Scaled Normal Equations System

In order to compute Δy from (12), we need to obtain the Cholesky factorization of $A\Pi A^{\mathrm{T}}$. It is well known that if $U(\Pi)$ is such that $\Pi = U(\Pi)^{\mathrm{T}}U(\Pi)$, then the Cholesky factor of $A\Pi A^{\mathrm{T}}$ is identical to the R-factor in the QR-factorization of the matrix $U(\Pi)A^{\mathrm{T}}$. In principle, it is therefore not needed to build the matrix $A\Pi A^{\mathrm{T}}$ itself. However, in most practical models we see that A is a very sparse matrix whereas $U(\Pi)A^{\mathrm{T}}$ can be dense. Furthermore, the number of rows in A is typically *much* smaller than the number of columns. Due to these properties, it is often computationally cheaper to compute $A\Pi A^{\mathrm{T}}$ than $U(\Pi)A^{\mathrm{T}}$, both in terms of storage and number of operations. Therefore, the matrix $A\Pi A^{\mathrm{T}}$ is explicitly formed in all implementations of the interior point method that the author is aware of.

The (i,j)th-entry in the matrix $A\Pi A^{\mathrm{T}}$ with NT-scaling $\Pi = D \otimes D$ is
$$a_i^{\mathrm{T}}\Pi a_j = \operatorname{tr} A_i D A_j D. \tag{54}$$

Sparsity in the A matrix can be exploited throughout the computation of the entries in $A\Pi A^{\mathrm{T}}$. For instance, one may apply the following scheme:

1. Let $T := DA_j$.

2. Compute $(TD)_{p,q}$ for those (p,q) where the (q,p)th entry in A_i is nonzero for some $i \in \{1,2,\dots,j\}$.

3. Compute $\operatorname{tr} A_i(TD)$ for $i = 1,2,\dots,j$.

Unfortunately, some entries in the $A\Pi A^{\mathrm{T}}$ matrix that results from the above sparsity exploiting scheme may be inaccurate due to numerical cancellation. This phenomenon is illustrated in Figures 1–2. These figures also show that numerical cancellation can be avoided, to a large extent, by using the Cholesky factor U_d of D. This is equivalent to computing the matrix as
$$A\Pi A^{\mathrm{T}} = (AU(\Pi)^{\mathrm{T}})(U(\Pi)A^{\mathrm{T}}).$$

However, this order of computation destroys the sparsity in A, and is thus not suited for large sparse problems.

Numerical cancellation in *forming* $A\Pi A^{\mathrm{T}}$ should not bother us; it is in fact a blessing in disguise. Namely this situation corresponds to degeneracy where ψ in (13) is simply a column of the identity matrix; in the example of Section 3.1 one has $\psi = \left[\begin{array}{cc} 1, & 0 \end{array}\right]^{\mathrm{T}}$. Usually, degeneracy is only revealed during the Cholesky factorization of $A\Pi A^{\mathrm{T}}$. Therefore, the pivots become unreliable even if the entries in $A\Pi A^{\mathrm{T}}$ are completely accurate. This means that we should exploit sparsity as much as possible during the formation of $A\Pi A^{\mathrm{T}}$. Techniques for improving numerical accuracy should come into play at a later stage, namely where we apply the preconditioned conjugate gradient method.

Before arriving at this later stage, we have to address another trade-off between accuracy and efficiency. For linear and second order cone programming, $A\Pi A^{\mathrm{T}}$ is almost always a sparse matrix. For semidefinite programs with multiple matrix variables, the $A\Pi A^{\mathrm{T}}$ matrix can also be sparse. The $A\Pi A^{\mathrm{T}}$ matrix should therefore be factored in a sparsity preserving pivot ordering, such as determined by the minimum degree heuristic. Unfortunately, an ordering based on symbolic considerations may be undesirable from a numerical point of view. This is illustrated in Figure 4. Even if we could compute $A\Pi A^{\mathrm{T}}$ and its Cholesky factor U with complete accuracy, the quality of the computed solution Δy to (11) will be poor due to excessive error propagation in the forward and backward solves.

In SeDuMi 1.05, the user can set a threshold quantity `maxu`, with a default value of `5E5`, such that the modified upper triangular Cholesky factor of $A\Pi A^{\mathrm{T}}$ is restricted to satisfy

$$\frac{u_{ij}}{u_{ii}} \leq \texttt{maxu} \text{ for all } i, j = 1, \ldots, m. \tag{55}$$

Such a modified Cholesky factor is in general merely a preconditioner to (12). In SeDuMi 1.05, the modification amounts to an 'adding strategy' for *undesirable* pivots and a 'skipping' strategy for *unreliable* pivots. An undesirable pivot is so small that it makes the factorization unstable, whereas an unreliable pivot has suffered so much numerical cancelation that it makes the factorization inaccurate. These concepts can be explained with the example of Section 3.1. Figure 2 shows that the $(1,1)$ pivot becomes unreliable for $\nu\mu < 10^{-5}$. In this case, we simply make the first row of the Cholesky factor identical to the first row of the identity matrix ('skipping'). Figure 2 shows that the $(1,1)$ pivot is already undesirable for $10^{-5} < \nu\mu < 0.005$, while still being reliable. In this case, we add a quantity to the $(1,1)$ entry which is just large enough to satisfy (55). In both cases, the modified Cholesky factor of $A\Pi A^{\mathrm{T}}$ is the usual Cholesky factor of $A\Pi A^{\mathrm{T}} + E$, where E is a low rank matrix. Using this Cholesky factor as a preconditioner in the preconditioned conjugate gradient (PCG) method theoretically leads to convergence in k steps, where k is the rank of E. Moreover, we can exploit the v-space Cholesky form in the PCG calculations to achieve high accuracy. However, the current implementation of this approach has its own shortcomings.

On massively degenerate problems, the modified Cholesky factorization introduces a modification of a substantial rank. To prevent excessive computational time, the number of PCG steps should however be limited. Unfortunately, the modified Cholesky factorization as described in the previous paragraph is a poor preconditioner, since the produced eigenvalues outside the ideal cluster at 1 are spread at a level near zero. The convergence of the PCG method can therefore be quite disappointing, even for moderately degenerate problems. Therefore, other modified Cholesky

strategies have to be investigated. The development and analysis of improved modification and conditioning schemes are subjects of future research.

7 Experimental Results

The techniques as discussed in this paper have been implemented in a solver, called SeDuMi [25]. We evaluated our approach on semidefinite programs and second order cone programs, some collected at New York University [23] and others for the Seventh Dimacs Challenge [24]. As a measure of numerical performance, we used the following quantities:

$$\mathtt{relgap} = \frac{c^\mathrm{T}x - b^\mathrm{T}y}{1 + |b^\mathrm{T}y|}$$

and

$$\mathtt{relerr} = \max\left\{\mathtt{relgap}, [\lambda_{\min}(x)]_-, \frac{[\lambda_{\min}(c - A^\mathrm{T}y)]_-}{1 + \|c\|_\infty}, \frac{\|Ax - b\|_2}{1 + \|b\|_\infty}\right\},$$

where $[\alpha]_- := \max(-\alpha, 0)$ denotes the negative part. In a pure interior point method as implemented in SeDuMi, one always has $[\lambda_{\min}(x)]_- = 0$.

Table 1 shows results on feasible problems from [24] with semi-definiteness constraints. The results were obtained with IEEE double precision floating point arithmetic, implemented in an Intel Pentium III processor at 933MHz, with a level 2 cache of size 256KB and a main memory of 256MB. The two optimization tools SeDuMi 1.05 [25] and SDPT3 3.0 [35] were invoked from MATLAB 5.3 under Windows 2000. The problems were solved with NT-scaling to maximal accuracy, using otherwise default parameters. In particular, we set `pars.eps=0`, `pars.maxiter=250` in SeDuMi, and `OPTIONS.gaptol=1e-20`, `OPTIONS.maxit=250`, `OPTIONS.vers=2` in SDPT3. The columns labeled '$c^\mathrm{T}x$' and '$b^\mathrm{T}y$' show 10 digits of the computed approximately optimal values of (2) and (3), respectively. The attained accuracy is quite satisfactory, except perhaps for the `minphase` problem. This problem does not seem to have a (finite) dual optimal solution. The dual solution is however strictly feasible, and is therefore a strict lower bound on the primal optimal value. This allows us to deduce that the dual objective value obtained by SDPT3, viz. $b^\mathrm{T}y = 5.978 < 5.981$, is further away from the true optimal value than the one computed by SeDuMi. The computational times needed by SDPT3 vary from 6.2 seconds for solving `minphase` to 12095.4 seconds for solving `copo23`. Not included in the table is the infeasible problem `filtinf1`, for which SeDuMi found a strictly feasible Farkas-type dual solution, whereas SDPT3 failed. Instances `hinf12` and `hinf13` are included in Table 2; the other excluded semidefinite programs from [24] have a size and structure for which the two solvers under consideration are not suited.

Table 2 shows results on semidefinite models arising in H^∞ control, collected in [23]. In these models, there are no positive definite primal feasible solutions, and there does not appear to be a finite dual optimal solution. The situation is therefore similar as for the `minphase` problem discussed before. The objective values reported by SeDuMi can differ substantially from those generated by SDPT3 and other solvers; see the table below.

| | SDPT3 | SeDuMi | | | | |
problem	relerr	relerr	relgap	$c^\mathrm{T}x$	$b^\mathrm{T}y$	sec
copo14	2E-008	1E-012	1E-013	-2.566884205E-012	-2.711390990E-012	46.6
copo23	2E-008	2E-012	-7E-013	-3.936694720E-012	-3.284431946E-012	4691.7
filter48_socp	2E-005	5E-011	-3E-009	1.416129188E+000	1.416129195E+000	27.0
minphase	2E-008	2E-008	-2E-005	5.980989890E+000	5.981110313E+000	5.2
truss5	4E-007	8E-013	-3E-015	1.326356780E+002	1.326356780E+002	4.5
truss8	2E-006	5E-013	-1E-014	1.331145892E+002	1.331145892E+002	38.3

Table 1: Results on (mixed) semidefinite programs using SeDuMi 1.05 and SDPT3 3.0.

| | SDPT3 | SeDuMi | | | | |
problem	relerr	relerr	relgap	$c^\mathrm{T}x$	$b^\mathrm{T}y$	sec
hinf1	6E-008	5E-011	-2E-007	-2.032606431E+000	-2.032605919E+000	1.9
hinf2	4E-007	6E-009	-4E-007	-1.096709353E+001	-1.096708906E+001	0.4
hinf3	4E-006	2E-010	-6E-008	-5.694086865E+001	-5.694086497E+001	0.7
hinf4	3E-007	4E-009	-3E-008	-2.747639871E+002	-2.747639792E+002	0.6
hinf5	3E-003	7E-008	-2E-006	-3.622192518E+002	-3.622184948E+002	0.4
hinf6	4E-005	5E-008	-2E-006	-4.489357520E+002	-4.489349136E+002	0.4
hinf7	6E-006	8E-008	-9E-005	-3.907786962E+002	-3.907429957E+002	0.3
hinf8	1E-005	3E-007	-5E-006	-1.161479635E+002	-1.161473575E+002	0.4
hinf9	7E-004	4E-010	2E-011	-2.362492583E+002	-2.362492583E+002	0.6
hinf10	3E-007	2E-007	-2E-005	-1.087395148E+002	-1.087373010E+002	0.8
hinf11	2E-007	5E-008	-2E-005	-6.587598042E+001	-6.587479701E+001	1.0
hinf12	2E-008	6E-012	-8E-004	-3.980298894E-003	-3.132031854E-003	0.7
hinf13	1E-004	2E-004	-6E-003	-4.547645933E+001	-4.520038496E+001	0.9
hinf14	9E-005	1E-006	-8E-005	-1.299722214E+001	-1.299605985E+001	1.8
hinf15	1E-004	1E-004	-1E-002	-2.505904332E+001	-2.481198770E+001	2.1

Table 2: Results on semidefinite programs from H^∞ control using SeDuMi 1.05 and SDPT3 3.0.

problem	SDPT3 relerr	SeDuMi relerr	relgap	$c^{\mathrm{T}}x$	$b^{\mathrm{T}}y$	sec
ladder1	2E-010	9E-014	-4E-014	-2.353439750E+001	-2.353439750E+001	0.5
ladder2	3E-010	4E-014	-1E-014	-2.346762362E+001	-2.346762362E+001	0.6
nb	2E-004	6E-012	6E-013	-5.070309465E-002	-5.070309465E-002	90.0
nb_L1	2E-004	1E-013	-5E-014	-1.301227067E+001	-1.301227067E+001	48.5
nb_L2	9E-007	9E-013	-6E-014	-1.628971981E+000	-1.628971981E+000	132.0
nb_L2_bessel	7E-007	2E-013	-2E-014	-1.025695112E-001	-1.025695112E-001	87.1
nql30 (new)	2E-006	2E-011	-3E-010	-9.460284898E-001	-9.460284893E-001	17.1
nql60 (new)	3E-006	4E-012	-7E-011	-9.350529481E-001	-9.350529480E-001	79.1
nql180 (new)	7E-005	4E-010	-2E-006	-9.277274775E-001	-9.277237150E-001	1146.9
qssp30 (new)	3E-007	7E-013	-2E-013	-6.496675734E+000	-6.496675734E+000	16.3
qssp60 (new)	5E-006	1E-013	1E-013	-6.562706469E+000	-6.562706469E+000	126.2
qssp180 (new)	1E-004	2E-001	-2E-003	-6.282258034E+000	-6.267906497E+000	3977.7
sched_50_50_scaled	6E-005	2E-009	-3E-013	7.852038440E+000	7.852038440E+000	25.2
sched_100_50_scaled	7E-004	6E-009	9E-011	6.716503110E+001	6.716503110E+001	51.3
sched_100_100_scaled	3E-002	9E-007	-2E-010	2.733078561E+001	2.733078561E+001	140.5
sched_200_100_scaled	3E-003	9E-007	-2E-012	5.181196103E+001	5.181196103E+001	359.7

Table 3: Results on second order cone programs using SeDuMi 1.05 and SDPT3 3.0.

problem	SDPT3 $c^{\mathrm{T}}x$	$b^{\mathrm{T}}y$	SeDuMi 1.04β $c^{\mathrm{T}}x$	$b^{\mathrm{T}}y$	SeDuMi 1.05 $c^{\mathrm{T}}x$	$b^{\mathrm{T}}y$
hinf12	-1.573E+000	-7.871E-001	-2.314E-002	-2.040E-002	-3.980E-003	-3.132E-003
hinf13	-4.770E+001	-4.600E+001	-4.437E+001	-4.436E+001	-4.548E+001	-4.520E+001

For `hinf12`, SeDuMi found a strictly feasible dual solution y, i.e. $C - \sum_{i=1}^{m} y_i A_i$ is positive definite. The accompanying x-solution is also positive definite, and has a constraint violation of $\|Ax - b\|_2 =$`1E-11`. One may be surprised that a substantial negative duality gap $c^{\mathrm{T}}x - b^{\mathrm{T}}y$ can be obtained with such a small constraint violation. In fact, it was shown in [34, 28] that given a solution x with a constraint violation $\|Ax - b\| > 0$, and a quantity $\delta > 0$, there cannot exist a dual solution y with an objective value $b^{\mathrm{T}}y \geq c^{\mathrm{T}}x + \delta$, unless $\|y\|_2 \geq \delta/\|Ax - b\|_2$. This implies for the y solution computed by SeDuMi that

$$\|y\|_2 \geq \frac{b^{\mathrm{T}}y - c^{\mathrm{T}}x}{\|Ax - b\|} \approx \frac{\texttt{8E}-4}{\texttt{1E}-11} = \texttt{8E7}.$$

Indeed, the actual size of the computed y solution is $\|y\|_2 = $ `9E9`. So despite the tiny constraint violation, it is possible that the optimal value to `hinf12` is simply zero.

Table 3 shows results for second order cone problems. On most instances, the primal and dual objective values agree to their ten reported digits. However, SeDuMi 1.05 failed to produce a sensible result on `qssp180`. The approximately optimal objective values reported by SDPT3 for this problem are $c^{\mathrm{T}}x = -6.639$ and $b^{\mathrm{T}}y = -6.640$. The solution times of SDPT3 varied between 0.2 seconds for `ladder1` to 10800.3 seconds for `qssp180`.

In general, Tables 1–3 show that our implementation of the v-space factors, SeDuMi 1.05, produced less accurate results than SDPT3 on only two of the instances under consideration, namely `hinf13` and `qssp180`. In most cases, however, our results were substantially more accurate. We conclude that the numerical results support the use of v-space factors.

8   Conclusions

We have identified various sources of numerical problems in the interior point method, that are specific to semidefinite programming. Since these specific issues have not been dealt with before, we faced a heavy numerical challenge in our effort to develop a reliable implementation of the interior point method for semidefinite programming.

An important technique to avoid numerical cancellation in computing the Nesterov-Todd scaling has been proposed in this paper: the v-space Cholesky form. The v-space Cholesky form was already used as an elegant approach to derive the Nesterov-Todd direction in Sturm and Zhang [30], but the importance of this Cholesky form for numerical computing has not been recognized before. Moreover, we have proposed an effective algorithm to update the v-space factors in an interior point framework. A numerically stable v-space factor updating rule for second order cone programming has also been derived.

On sparse problems with a large number of positive semidefinite or second order cone constraints, the $A\Pi A^{\mathrm{T}}$ matrix is typically sparse. The pivot order for factoring the $A\Pi A^{\mathrm{T}}$ matrix should therefore be chosen using a sparsity preserving heuristic, where numerical information is not taken into account. We have seen that without modification, the resulting Cholesky factor can be highly instable on degenerate problems. To resolve this difficulty, we have proposed to calculate a stable but modified Cholesky factor as a preconditioner in a conjugate gradient method. Further research has to be done on the method of modification.

The numerical results are quite encouraging. The primal and dual objective values correspond to 10 or more digits for most semidefinite programs in Table 1 and second order cone programs in Table 3. In fact, the duality gap of the computed solutions was positive on merely five of the 37 instances in Tables 1–3.

Throughout this paper, we have focused on the search direction problem in a standard form feasible interior point method, as given by (8). However, general purpose interior point methods such as SDPT3 and SeDuMi use either the infeasible interior point framework or the self-dual embedding framework. This should be taken into account in a more comprehensive numerical study.

References

[1] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization problems. *SIAM Journal on Optimization*, 5:13–51, 1995.

[2] F. Alizadeh, J.A. Haeberly, and M. Overton. Complementarity and nondegeneracy in semidefinite programming. *Mathematical Programming*, 77(2):111–128, 1997.

[3] F. Alizadeh, J.A. Haeberly, and M. Overton. Primal–dual interior–point methods for semidefinite programming: convergence rates, stability and numerical results. *SIAM Journal on Optimization*, 8(3):746–768, 1998.

[4] T. Ando. Concavity of certain maps and positive definite matrices and applications to Hadamard products. *Linear Algebra and its Applications*, 26:203–241, 1979.

[5] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1983.

[6] J. Faraut and A. Korányi. *Analysis on Symmetric Cones*. Oxford Mathematical Monographs. Oxford University Press, New York, 1994.

[7] D. Goldfarb and K. Scheinberg. Interior point trajectories in semidefinite programming. *SIAM Journal on Optimization*, 8:871–886, 1998.

[8] C.C. Gonzaga. Path-following methods for linear programming. *SIAM Review*, 34(2):167–224, 1992.

[9] C.C. Gonzaga. Interior path following algorithms. In J.R. Birge and K.G. Murty, editors, *Mathematical Programming, State of the Art 1994*, pages 93–101, Ann Arbor, 1994. University of Michigan.

[10] C. Helmberg, F. Rendl, R.J. Vanderbei, and H. Wolkowicz. An interior–point method for semidefinite programming. *SIAM Journal on Optimization*, 6:342–361, 1996.

[11] N.J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, 1996.

[12] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, New York, 1985. Corrected reprint 1990.

[13] M. Kojima, N. Megiddo, T. Noma, and A. Yoshise. *A Unified Approach to Interior Point Algorithms for Linear Complementarity Problems*. Springer-Verlag, Berlin, 1991.

[14] M. Kojima, S. Shindoh, and S. Hara. Interior–point methods for the monotone semidefinite linear complementarity problem in symmetric matrices. *SIAM Journal on Optimization*, 7(1):86–125, 1997.

[15] S. Kruk. *High Accuracy Algorithms for the Solutions of Semidefinite Linear Programs*. PhD thesis, University of Waterloo, Waterloo, Ontario, Canada, February 2001.

[16] M.S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second–order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998. Special Issue on Linear Algebra in Control, Signals and Image Processing.

[17] Z.-Q. Luo, J.F. Sturm, and S. Zhang. Superlinear convergence of a symmetric primal–dual path following algorithm for semidefinite programming. *SIAM Journal on Optimization*, 8(1):59–81, 1998.

[18] I.J. Lustig, R.E. Marsten, and D.F. Shanno. Computational experience with a primal–dual interior point method for linear programming. *Linear Algebra and its Applications*, 152:191–222, 1991.

[19] S. Mehrotra. On the implementation of a primal–dual interior point method. *SIAM Journal on Optimization*, 2:575–601, 1992.

[20] S. Mizuno, M.J. Todd, and Y. Ye. On adaptive-step primal-dual interior-point algorithms for linear programming. *Mathematics of Operations Research*, 18:964–981, 1993.

[21] Y. Nesterov and M.J. Todd. Self–scaled barriers and interior–point methods for convex programming. *Mathematics of Operations Research*, 22(1):1–42, 1997.

[22] Y. Nesterov and M.J. Todd. Primal–dual interior–point methods for self–scaled cones. *SIAM Journal on Optimization*, 8:324–364, 1998.

[23] M.L. Overton. Test problems: LMI, truss and Steiner tree problems. Located at URL `http://cs.nyu.edu/cs/faculty/overton/sdppack/sdppack.html`, 1997.

[24] G. Pataki and S. Schmieta. The DIMACS library of semidefinite-quadratic-linear programs. Technical report, Computational Optimization Research Center, Columbia University, New York, NY, USA, 1999. `http://dimacs.rutgers.edu/Challenges/Seventh/Instances/`.

[25] J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999. Special issue on Interior Point Methods (CD supplement with software).

[26] J.F. Sturm. Central region method. In J.B.G. Frenk, C. Roos, T. Terlaky, and S. Zhang, editors, *High Performance Optimization*, pages 157–194. Kluwer Academic Publishers, 2000.

[27] J.F. Sturm. Similarity and other spectral relations for symmetric cones. *Linear Algebra and its Applications*, 312:135–154, 2000.

[28] J.F. Sturm. Theory and algorithms for semidefinite programming. In J.B.G. Frenk, C. Roos, T. Terlaky, and S. Zhang, editors, *High Performance Optimization*, pages 3–196. Kluwer Academic Publishers, 2000.

[29] J.F. Sturm and S. Zhang. On a wide region of centers and primal–dual interior point algorithms for linear programming. *Mathematics of Operations Research*, 22(2):408–431, 1997.

[30] J.F. Sturm and S. Zhang. Symmetric primal–dual path following algorithms for semidefinite programming. *Applied Numerical Mathematics*, 29:301–315, 1999.

[31] M.J. Todd. A study of search directions in interior–point methods for semidefinite programming. *Optimization Methods and Software*, 11–12:1–46, 1999.

[32] M.J. Todd. Semidefinite optimization. *Acta Numerica*, 10:515–560, 2001.

[33] M.J. Todd, K.C. Toh, and R.H. Tütüncü. On the Nesterov–Todd direction in semidefinite programming. *SIAM Journal on Optimization*, 8(3):769–796, 1998.

[34] M.J. Todd and Y. Ye. Approximate Farkas lemmas and stopping rules for iterative infeasible–point algorithms for linear programming. *Mathematical Programming*, 81:1–21, 1998.

[35] K.C. Toh, M.J. Todd, and R.H. Tütüncü. SDPT3 - a MATLAB package for semidefinite programming. version 1.3. *Optimization Methods and Software*, 11–12:545–581, 1999. Special issue on Interior Point Methods (CD supplement with software).

[36] T. Tsuchiya. A convergence analysis of the scaling–invariant primal–dual path–following algorithm for second–order cone programming. *Optimization Methods and Software*, 11–12:141–182, 1999. Special issue on Interior Point Methods (CD supplement with software).

[37] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1996.

[38] S.J. Wright. Modified Cholesky factorizations in interior point algorithms for linear programming. Technical Report ANL/MCS-P600-0596, Argonne National Laboratory, Argonne, IL, 1996.

[39] Y. Ye, M.J. Todd, and S. Mizuno. An $O(\sqrt{n}L)$-iteration homogeneous and self–dual linear programming algorithm. *Mathematics of Operations Research*, 19:53–67, 1994.