THE UNIVERSITY *of* EDINBURGH

Basis Preconditioning in Interior Point Methods

Lukas Schork

Doctor of Philosophy
The University of Edinburgh
2018

Abstract

Solving normal equations $AA^T\boldsymbol{x} = \boldsymbol{b}$, where A is an $m \times n$ matrix, is a common task in numerical optimization. For the efficient use of iterative methods, this thesis studies the class of preconditioners of the form BB^T, where B is a nonsingular "basis" matrix composed of m columns of A.

It is known that for any matrix A of full row rank B can be chosen so that the entries in $|B^{-1}A|$ are bounded by 1. Such a basis is said to have "maximum volume" and its preconditioner bounds the spectrum of the transformed normal matrix in the interval $[1, 1+mn]$. The theory is extended to (numerically) rank deficient matrices, yielding a rank revealing variant of Gaussian elimination and a method for computing the minimum norm solution for $\boldsymbol{x}$ from a reduced normal system and a low-rank update. Algorithms for finding a maximum volume basis are discussed.

In the linear programming interior point method a sequence of normal equations needs to be solved, in which A changes by a column scaling from one system to the next. A heuristical algorithm is proposed for maintaining a basis of approximate maximum volume by update operations as those in the revised simplex method. Empirical results demonstrate that the approximation means no loss in the effectiveness of the preconditioner, but makes basis selection much more efficient.

The implementation of an interior point solver based on the new linear algebra is described. Features of the code include the elimination of free variables during preconditioning and the removal of degenerate variables from the optimization process once sufficiently close to a bound. A crossover method recovers a vertex solution to the linear program, starting from the basis at the end of the interior point solve. A computational study shows that the implementation is robust and of general applicability, and that its average performance is comparable to that of state-of-the-art solvers.

Declaration

The present thesis was written by myself and the work contained therein is my own except where stated otherwise in the text. The work has not been submitted for any other degree or professional qualification except as specified.

Didcot, 15 February 2019 Lukas Schork

Contents

Notations

$\mathbb{R}^n$	Space of n-dimensional real vectors.
$\mathbb{R}^n_{>0}$	Space of n-dimensional real vectors with positive entries.
$\mathbb{R}^{m\times n}$	Space of real $m \times n$ matrices.
$\mathcal{B}$, $\mathcal{J}$, $\mathcal{N}$, ...	Ordered index sets; $\mathcal{J}_k$ is the k-th index and $\lvert\mathcal{J}\rvert$ the number of indices in the set.
A_j	Column j of matrix A.
$A_{\mathcal{B}}$	If A is not diagonal matrix, $A_{\mathcal{B}}$ is composed of columns A_j for $j \in \mathcal{B}$.
$D_{\mathcal{B}}$	If D is diagonal matrix, $D_{\mathcal{B}}$ is the principal submatrix indexed by $\mathcal{B}$.
$\boldsymbol{x}_{\mathcal{B}}$	Subvector of $\boldsymbol{x}$ indexed by $\mathcal{B}$.
I, I_m	Identity matrix (of dimension m).
$\operatorname{diag}(x_i)$	Diagonal matrix with entries x_i on the diagonal.
$\boldsymbol{e}_j$	Column j of I.
$\boldsymbol{e}$	Vector of all 1s.
$\mathbf{0}$	Vector of all 0s.
$\lvert\boldsymbol{x}\rvert$, $\lvert A\rvert$	Componentwise absolute value of vector $\boldsymbol{x}$ and matrix A.
$\lVert\boldsymbol{x}\rVert$, $\lVert A\rVert$	2-norm of vector $\boldsymbol{x}$ and induced operator norm of matrix A.
$\lVert\boldsymbol{x}\rVert_\infty$	Infinity norm of $\boldsymbol{x}$.
$\lVert A\rVert_{\max}$	Maximum entry of A in absolute value.
$\lVert A\rVert_F$	Frobenius norm of A.
$\lVert A\rVert_*$	Nuclear norm of A (sum of the singular values).
$\sigma_r(A)$	r-th largest singular value of A.
$\sigma_{\min}(A)$, $\sigma_{\max}(A)$	Minimum and maximum singular value of A.
$\operatorname{cond}(A)$	2-norm condition number of A ($\sigma_{\max}(A)/\sigma_{\min}(A)$).
$\det B$	Determinant of square matrix B.
$\operatorname{vol}(A)$	Volume of A defined as the product of its singular values.
$\operatorname{null}(A)$	Null space of A.
$\operatorname{nnz}(A)$	Number of nonzero entries in A.
$O(\cdot)$	If $0 \le f(x) \le cg(x)$ for all $x \ge x_0$, where c and x_0 are positive constants, then $f(x) = O(g(x))$.
$\Omega(\cdot)$	If $0 \le cg(x) \le f(x)$ for all $x \ge x_0$, where c and x_0 are positive constants, then $f(x) = \Omega(g(x))$.
$\Theta(\cdot)$	If $f(x) = O(g(x))$ and $f(x) = \Omega(g(x))$, then $f(x) = \Theta(g(x))$.

1 Introduction

Linear programming (LP) is the minimization of a linear function subject to linear equality and inequality constraints on the variables. LP problems arise from modelling real-world processes in a variety of areas (see Vanderbei [79] for an overview) and as subproblems in algorithms for nonlinear and mixed-integer optimization. For solving large-scale problems by today's standards, which can have in the tens of millions of variables and constraints, the interior point method (IPM) is often the fastest or the only feasible option. This thesis develops the theory, computational techniques and implementation of an interior point solver that is based on a novel linear algebra kernel.

Solving sparse linear systems is the most time consuming part of the IPM. Current state-of-the-art implementations are based on Cholesky factorization (see Andersen et al. [6]), which is robust and has proved to be efficient on a wide range of problems. The issue with direct factorization, however, is the computational work and memory requirement when the sparsity pattern of the constraint matrix leads to large fill-in. Iterative methods seem to be an alternative in this case, but they have rarely found their way into production software. The challenge for iterative linear algebra is the high ill-conditioning that occurs in advanced interior point iterations and that requires effective preconditioning to obtain an acceptable convergence rate of the linear solver.

The linear systems from the IPM can be expressed as normal equations $AA^T\boldsymbol{x} = \boldsymbol{b}$ with an $m \times n$ matrix A of full row rank. A basis preconditioner for AA^T is BB^T, where the basis matrix B is nonsingular and composed of m columns of A. Employing basis preconditioning in the IPM is motivated by two observations: Firstly, basis matrices from LP models are typically "easy" to factorize because a large part of them can be permuted to triangular form. Hence computing and applying an inverse representation of the preconditioner is much less expensive than factorizing AA^T directly. Secondly, for a nondegenerate LP model $n - m$ columns of A tend to zero when the IPM approaches the solution, while the remaining m columns form a nonsingular matrix. With the obvious choice for B the basis preconditioner is then asymptotically perfect.

Basis preconditioning in the linear programming IPM was the subject of at least two previous PhD theses by Oliveira (1997) [61] and Al-Jeiroudi (2009) [2]. In both works B was chosen as the first m linearly independent columns of A after ordering the columns by decreasing magnitude. (The 1-norm was used in [61] and the IPM scaling factors were used in [2].) Their conclusion that the iterative approach performed better than direct factorization on some problem classes should be considered with care, however, firstly because the performance comparisons were based on simplicial[1] Cholesky codes, and secondly because an ad-hoc ordering method was applied to the normal matrix without taking special structures of A (such as dense columns) into account. It is clear from the reported results that the iterative approach would not be competitive to a state-of-the-art direct method. It is also unsatisfactory that the IPM implementations converged only on a small set of problems.

The issue with the previous approach for choosing B is that it guarantees effectiveness of the preconditioner only in the limit of the interior point solve and only if the LP problem is (nearly) nondegenerate. In practice LP models are often highly degenerate and asymptotic properties may not be observed until very late in the interior point solve. In either case the theoretical guarantees of the preconditioner are very conservative, see Monteiro et al. [58].

This thesis develops an IPM with basis preconditioning using a novel method for choosing B. It is known that for any matrix A of full row rank a basis matrix exists such that the entries in $|B^{-1}A|$ are bounded by 1. Such a basis is said to have "maximum volume" because $|\det B|$ will then be maximum among all neighbouring bases (i. e. among all bases obtained by

[1] The term "simplicial" is used for factorization methods that do not exploit cache memory through operations on dense submatrices.

replacing one column of B). The maximum volume criterion has been used in other areas of numerical linear algebra for decades [42, 8, 31, 64, 29, 28]. For basis preconditioning it was first investigated recently by Arioli and Duff [7], who observed that a maximum volume basis bounds the spectrum of the preconditioned normal matrix in the interval $[1, 1+mn]$; i. e. independently of the numerical properties of A. For application in the IPM a heuristical algorithm for basis selection is proposed because finding a maximum volume basis would be unacceptably expensive for large-scale problems. It will be seen that a basis of comparable quality can be maintained efficiently between interior point iterations using basis updates as those in the revised simplex method.

The thesis further explores the availability of a basis matrix in IPM implementation. The well-known issue of handling free variables is resolved by eliminating them in the preconditioning phase and the basis is utilized to remove degenerate variables from the optimization process once sufficiently close to a bound. Both techniques increase the numerical stability of the IPM. A crossover method for recovering a vertex solution starting from the basis at the end of the interior point solve is investigated. Crossover is often the dominating cost for solving large-scale LP models. In our setting the crossover time could be reduced tremendously in some cases by computing an interior solution of very high accuracy, which is enabled by the techniques mentioned before.

An interior point solver based on these ideas has been implemented in a new software package called IPX. The package is written in C++ and is of production quality. It can interface third-party codes for the sparse LU factorization/update of basis matrices, allowing to choose a factorization routine that suits the characteristics of the problem. Computational results demonstrate that the solver is robust and of general applicability, and that it performs better than the best direct solvers on several large-scale LP models.

Chapter 2 of the thesis is devoted to a general study of the maximum volume concept and its application to rank revealing factorizations. Algorithms for finding a maximum volume basis are discussed. Chapter 3 introduces basis preconditioning for least squares problems, which are important in their own right and lead to the same linear systems as the IPM. In Chapter 4 the convergence of an IPM with inexact step directions is analysed. Inexact directions result from iterative linear algebra, and the analysis gives some insights about how accurately the systems need to be solved. Chapter 5 continues with theoretical results about basis matrices in the IPM. Chapters 6 and 7 are concerned with computational techniques. They present the design and implementation of IPX and an LU factorization/update that has been developed specifically for very sparse LP matrices. Chapter 8 discusses the results of a computational study of the code. Final conclusions are drawn in Chapter 9.

2 Maximum Volume Basis

The maximum volume criterion for choosing a submatrix out of a larger matrix is a well-known concept in rank revealing factorizations. It was first used in orthogonal methods [36, 31] and later extended to Schur complement factorizations [64, 29] (i. e. Gaussian elimination and its variants). In the present work it will provide the backbone for building a preconditioner for an iterative linear solver. This chapter introduces the maximum volume criterion and develops a new rank revealing method based on it, that is suitable for sparse matrices.

Here and in the sequel a basis $\mathcal{B}$ for a matrix $A \in \mathbb{R}^{m\times n}$ of rank m is an ordered set of m indices such that $A_{\mathcal{B}}$ is nonsingular. Associated with $\mathcal{B}$ is the nonbasic set $\mathcal{N}$ with the obvious definition. $A_{\mathcal{B}}^{-1}A_{\mathcal{N}}$ is called the "tableau matrix". Given a partitioning of an arbitrary matrix A into

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{2.1}$$

in which A_{11} is square and nonsingular, let the matrix $A^{\times}$ be defined as

$$A^{\times} = \begin{bmatrix} A_{11}^{-1} & A_{11}^{-1}A_{12} \\ -A_{21}A_{11}^{-1} & A/A_{11} \end{bmatrix},$$

where $A/A_{11} = A_{22} - A_{21}A_{11}^{-1}A_{12}$ denotes the Schur complement of A_{11} in A. For reasons that will become clear below, we call $A^{\times}$ the "generalized tableau matrix".

2.1 Definition and Properties

The volume of a matrix was first defined by Ben-Israel [8] as the product of its nonzero singular values. In the present work it is more appropriate for a rank deficient matrix to have volume zero, so that the following definition is used.

Definition 2.1. *For $A \in \mathbb{R}^{m\times n}$ with singular values $\sigma_1 \geq \ldots \geq \sigma_d \geq 0$ ($d = \min(m,n)$), the volume of A is* $\operatorname{vol}(A) = \sigma_1 \cdots \sigma_d$.

The volume of a square matrix is the absolute value of its determinant. For an $m \times n$ matrix with $m < n$ we have $\operatorname{vol}(A) = (\det AA^T)^{1/2}$ and as shown by Ben-Israel [8]

$$\operatorname{vol}(A) = \left(\sum_{\mathcal{B} \text{ basis}} \operatorname{vol}(A_{\mathcal{B}})^2 \right)^{1/2}.$$

It follows that $\operatorname{vol}(A_{\mathcal{B}}) \leq \operatorname{vol}(A)$ for any basis.

Definition 2.2. *Let $A \in \mathbb{R}^{m\times n}$ and $\rho \geq 1$.*

(i) Let B be a $k \times k$ submatrix of A. B has global ρ-maximum volume in A if $\operatorname{vol}(B) \neq 0$ *and*

$$\rho \operatorname{vol}(B) \geq \operatorname{vol}(B') \tag{2.2}$$

for all $k \times k$ submatrices B' of A.

(ii) Let B be formed by k columns (rows) of A. B has local ρ-maximum volume in A if $\operatorname{vol}(B) \neq 0$ *and* (2.2) *holds for any B' that is obtained by replacing one column (row) of B by one column (row) of A that is not in B.*

(iii) Let $k < \min(m,n)$ and B be a $k \times k$ submatrix of A. B has local ρ-maximum volume in A if it has global ρ-maximum volume in all $(k+1)\times(k+1)$ submatrices of A that contain B.

The definition of local maximum volume of a row or column slice is from Pan [64]. The extension to arbitrary submatrices was introduced by the author in [73]. Note that in the latter case it is not sufficient for B to have local maximum volume in its row and column slice; the volume must be maximum up to a factor ρ even if a row and column are exchanged.

Lemma 2.3. *Let $B \in \mathbb{R}^{m\times m}$ be nonsingular.*

(i) Let B' be obtained by replacing column i of B by the vector $\boldsymbol{a}$. Then $\operatorname{vol}(B') = |v_i| \operatorname{vol}(B)$, where $\boldsymbol{v} = B^{-1}\boldsymbol{a}$.

(ii) Let B' be obtained by replacing columns $i_1 < \cdots < i_s$ of B by the columns of $A \in \mathbb{R}^{m\times s}$. Then $\operatorname{vol}(B') = |\det S| \operatorname{vol}(B)$, where S is the submatrix of $B^{-1}A$ composed of rows $i_1, \ldots, i_s$.

Proof. (i) follows immediately from Cramer's rule, by which the column replacement changes $\det B$ by the factor v_i. (ii) follows analogously from the generalized Cramer's rule [26]. □

A consequence of part (i) is that a basis matrix $A_{\mathcal{B}}$ has local ρ-maximum volume in A if and only if $\left\|A_{\mathcal{B}}^{-1}A\right\|_{\max} \le \rho$. This characterization gives the local maximum volume its importance for rank revealing factorizations and matrix approximation.

The formula for multiple column replacements allows to derive a bound on the global maximum volume in terms of a local one. Assume that $B \in \mathbb{R}^{m\times m}$ has local ρ-maximum volume in $A \in \mathbb{R}^{m\times n}$. Clearly, a B' of global maximum volume can be obtained by exchanging at most m columns of B. By Lemma 2.3 the volume change is the absolute value of the determinant of the corresponding submatrix of $B^{-1}A$. Because $\left\|B^{-1}A\right\|_{\max} \le \rho$, it follows from Hadamard's inequality [30, Section 14.18] that for any $s \times s$ submatrix S of $B^{-1}A$

$$|\det S| \le \prod_{i=1}^{s} \|S_i\| \le (\rho\sqrt{s})^s.$$

(S_i denotes the i-th column of S and $\|\cdot\|$ its 2-norm.) Therefore the global maximum volume is at most a factor $(\rho\sqrt{m})^m$ larger than a local ρ-maximum volume. This bound has already been established by Goreinov et al. [28] by the same argument. They also showed by example that the bound is sharp.

The above lemma can be generalized to row and column replacements.

Lemma 2.4. *Let the matrix A be partitioned as in (2.1) and let A_{11} be $k \times k$ nonsingular. Interchanging columns j_1 and j_2, where j_1 lies in the first block and j_2 in the second block, and rows i_1 and i_2, where i_1 lies in the first block and i_2 in the second block, changes $\operatorname{vol}(A_{11})$ by the factor $|\det S|$, where S is the submatrix of $A^{\times}$ composed of rows (j_1, i_2) and columns (i_1, j_2).*

Proof. Consider the bordered matrix $\mathbb{A} = \begin{bmatrix} A & I_m \end{bmatrix}$. The columns of A and I_m in $\mathbb{A}$ are termed "structural" and "logical", respectively. Let $\mathcal{B}$ and $\mathcal{N}$ be the index sets such that

$$\mathbb{A}_{\mathcal{B}} = \begin{bmatrix} A_{11} & 0 \\ A_{21} & I_{m-k} \end{bmatrix}, \quad \mathbb{A}_{\mathcal{N}} = \begin{bmatrix} I_k & A_{12} \\ 0 & A_{22} \end{bmatrix},$$

which are uniquely defined by the partitioning of A. Notice that $\mathcal{B}$ is a basis for $\mathbb{A}$ and that $\operatorname{vol}(A_{\mathcal{B}}) = \operatorname{vol}(A_{11})$. Exchanging a structural column of $\mathbb{A}_{\mathcal{B}}$ with a structural column of $\mathbb{A}_{\mathcal{N}}$ corresponds to a column exchange between the two blocks in A. Likewise, exchanging a logical column of $\mathbb{A}_{\mathcal{B}}$ with a logical column of $\mathbb{A}_{\mathcal{N}}$ corresponds to a row exchange between the two blocks of A. Therefore the volume change we are interested in is the volume change of $\mathbb{A}_{\mathcal{B}}$ after

replacing columns j_1 and i_2 of that matrix by columns j_2 and i_1 of $\mathbb{A}_{\mathcal{N}}$. By Lemma 2.3 the volume change is the absolute value of the determinant of the 2×2 submatrix of

$$\mathbb{A}_{\mathcal{B}}^{-1}\mathbb{A}_{\mathcal{N}} = \begin{bmatrix} A_{11}^{-1} & A_{11}^{-1}A_{12} \\ -A_{21}A_{11}^{-1} & A/A_{11} \end{bmatrix}$$

composed of rows (j_1, i_2) and columns (i_1, j_2). But this is the submatrix S of $A^{\times}$ defined in the lemma. □

Lemma 2.4 was first proved by the author in [73] by a technical computation. The simplicity of the proof given here comes from the formula for multiple column exchanges given in Lemma 2.3, which was a consequence of the generalized Cramer's rule. The proof also extends to multiple row and column exchanges if i_1, i_2, j_1 and j_2 are replaced by index sets $\mathcal{I}_1$, $\mathcal{I}_2$, $\mathcal{J}_1$ and $\mathcal{J}_2$. Then $\mathrm{vol}(A_{11})$ changes by the absolute value of the determinant of the corresponding submatrix of $A^{\times}$ of dimension $|\mathcal{I}_1| + |\mathcal{J}_1|$. Because $A^{\times}$ plays the same role in Lemma 2.4 as $B^{-1}A$ in Lemma 2.3, the term "generalized tableau matrix" is justified.

Corollary 2.5. *The submatrix A_{11} in (2.1) has local ρ-maximum volume in A if and only if it has local ρ-maximum volume in its block row and block column and the determinant of all 2×2 submatrices of $A^{\times}$ with one entry in each block is bounded by ρ.*

2.2 Finding a Submatrix of Local Maximum Volume

The applications in the following sections require us to determine a basis matrix of local maximum volume. That means, given a matrix A of full row rank and a parameter $\rho \geq 1$, a method is required for finding $\mathcal{B}$ such that $\left\|A_{\mathcal{B}}^{-1}A\right\|_{\max} \leq \rho$.

Algorithms for that task were described by Knuth [42], Pan [64] and Goreinov et al. [28]. The common scheme is to start with an arbitrary basis matrix $A_{\mathcal{B}}$, which can be found by Gaussian elimination, and then to successively exchange columns of $A_{\mathcal{B}}$ by columns of $A_{\mathcal{N}}$ under the condition that $\mathrm{vol}(A_{\mathcal{B}})$ increases with each update. The scheme is formally stated in Algorithm 1. Because each basis update increases $\mathrm{vol}(A_{\mathcal{B}})$ by a factor $|A_{\mathcal{B}}^{-1}A|_{pj} > 1$, the algorithm terminates in a finite number of iterations.

Algorithm 1 Maxvolume

Input: $A \in \mathbb{R}^{m \times n}$ of rank m, starting basis $\mathcal{B}$, $\rho \geq 1$.

1: **while** $\left\|A_{\mathcal{B}}^{-1}A\right\|_{\max} > \rho$ **do**
2: Choose (p, j) such that $|A_{\mathcal{B}}^{-1}A|_{pj} > \rho$.
3: $\mathcal{B}_p = j$

If $\rho > 1$ and the initial basis is obtained by Gaussian elimination with complete pivoting in A, an upper bound on the iteration count of Algorithm 1 can be derived. In each step of Gaussian elimination, complete pivoting chooses an entry of maximum absolute value from the active submatrix as pivot element. Let $A^{[k]}$ denote the active submatrix prior to the k-th elimination and let u_{kk} be the chosen pivot element. It follows from [64, Theorem 2.7] that

$$\sigma_k(A) \leq \left\|A^{[k]}\right\| \leq |u_{kk}|\sqrt{mn}.$$

If $\mathcal{B}$ is the basis determined by the pivotal columns, then

$$\frac{\mathrm{vol}(A)}{\mathrm{vol}(A_{\mathcal{B}})} = \frac{\prod_{k=1}^{m} \sigma_k(A)}{\prod_{k=1}^{m} |u_{kk}|} \leq (\sqrt{mn})^m.$$

Because each basis update in Algorithm 1 increases $\mathrm{vol}(A_{\mathcal{B}})$ by at least a factor ρ and the volume of any basis matrix is at most $\mathrm{vol}(A)$, the iteration count is bounded by

$$\left\lceil \log_\rho \left(\left(\sqrt{mn}\right)^m \right) \right\rceil \leq \left\lceil \log_\rho \left(\left(\sqrt{nn}\right)^m \right) \right\rceil = \left\lceil m \log_\rho n \right\rceil .$$

Hence the iteration count is polynomial in m and n. The bound does not hold if the initial basis is obtained by Gaussian elimination with partial or rook pivoting.

Knuth [42] derived a similar bound for a specific version of Algorithm 1, which adds one row of A at a time and immediately restores the maximum volume property of the enlarged basis matrix in the new row slice. If the iteration count of Algorithm 1 for $\rho = 1$ can be made polynomial in m and n by a specific choice of pivot element is an open question. Finding a submatrix of global maximum volume is known to be NP-complete [65], but that proof does not extend to finding a local maximum volume basis.

The algorithm can be extended to row and column slices. We now allow $A \in \mathbb{R}^{m \times n}$ to be rank deficient and want to determine an $m \times k$ submatrix of local ρ-maximum volume for a given $k \leq \mathrm{rank}(A)$. The idea for the algorithm is the link to the matrix $A^T A$ and was first described by Pan [64]. The presentation given here is more general by not making use of a specific matrix factorization.

Let A be partitioned into $\begin{bmatrix} A_1 & A_2 \end{bmatrix}$, where A_1 has k columns and rank k. Then

$$A^T A = \begin{bmatrix} A_1^T \\ A_2^T \end{bmatrix} \begin{bmatrix} A_1 & A_2 \end{bmatrix} = \begin{bmatrix} A_1^T A_1 & A_1^T A_2 \\ A_2^T A_1 & A_2^T A_2 \end{bmatrix} =: \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = C,$$

and a column exchange between A_1 and A_2 results in a row and column interchange in $A^T A$. Because $\mathrm{vol}(A_1) = \mathrm{vol}(C_{11})^{1/2}$, A_1 has local ρ-maximum volume in A if and only if $\mathrm{vol}(C_{11})$ cannot be increased by more than a factor ρ^2 by a symmetric row and column interchange in C. To test that condition by means of Lemma 2.4, the determinant of all 2×2 submatrices of $C^\times$ that occupy a diagonal entry in both diagonal blocks must be investigated. If $|\det S| > \rho^2$ for such a submatrix S, the corresponding columns of A_1 and A_2 are exchanged, increasing the volume of A_1 by the factor $|\det S|^{1/2} > \rho$.

The algorithm, in fact, searches for a $k \times k$ principal submatrix of local ρ^2-maximum volume in C (i. e. a submatrix determined by the same set of row and column indices). In each step there are $k(n-k)$ possible choices for an exchange, so that the complexity of the algorithm is similar to finding a $k \times k$ basis matrix in a $k \times n$ matrix. The computations require more effort, however. Implementations of both algorithms using LU factorization are described by Pan [64] and are compared in terms of arithmetic operations.

2.3 Empirical Tests

Implementing Algorithm 1 requires a rule for choosing the pivot element in case there is more than one entry of the tableau matrix eligible. Goreinov et al. [28] used the maximum absolute entry, which is the obvious choice with regard to minimizing the number of updates. But this rule requires to "scan" the entire tableau matrix after each basis change, an operation that is costly for dense matrices and impractical for sparse matrices, where the tableau cannot be stored explicitly but has to be computed one column or row at a time.

A more economic method is given in Algorithm 2. Rather than updating the tableau matrix after each basis change, the algorithm can be implemented by maintaining a representation of $A_{\mathcal{B}}^{-1}$ and computing a tableau column when it is scanned. For dense matrices either an LU or a QR factorization of $A_{\mathcal{B}}$ can be used, for which numerically stable update schemes exist (see Fletcher and Matthews [17] and Section 12.5.2 in [23]). For sparse matrices LU update techniques are readily available from implementations of the simplex method. Algorithm 2

Algorithm 2 Maxvolume Sequential

Input: $A \in \mathbb{R}^{m \times n}$ of rank m, starting basis $\mathcal{B}$, $\rho \geq 1$.

for $pass = 1, 2, \ldots$ **do**
  $updates = 0$ // count basis updates in this pass
  for $j = 1$ to n **do**
    if $j \notin \mathcal{B}$ **then**
      Compute $\boldsymbol{v} = A_{\mathcal{B}}^{-1} A_j$.
      $p = \arg\max_i |v_i|$
      if $|v_p| > \rho$ **then**
        $\mathcal{B}_p = j$
        $updates = updates + 1$
  Stop if $updates = 0$.

name	m	n	nnz(A)
bab3	22,478	393,457	3,097,799
dbic1	33,598	140,205	781,668
karted	46,501	133,114	1,770,336
nug30	52,260	379,350	1,567,800
pds-100	94,994	433,867	933,313
rail02	54,524	192,618	599,436
rail4284	4,176	1,090,526	11,174,639
rmine14	32,205	268,535	660,346
srd120	186,440	357,070	9,804,510
stp3d	97,936	137,646	500,753
watson_2	185,474	378,986	1,040,238

Table 1: Test set of LP matrices.

terminates after a complete "pass" over the columns of A without requiring an update, since then all entries of the final tableau were computed and are bounded by ρ.

To investigate the number of passes and basis updates, the algorithm was applied to a set of LP matrices that originated from an early iteration of an interior point solver. In this application the matrices have the form $A = \hat{A}D$, where $\hat{A}$ is the constraint matrix from the LP model and D is a positive definite diagonal matrix. The test problems and their dimensions are listed in Table 1. The matrices were selected from a larger set of LP models (used in Chapter 8) because they represent a variety of sparsity patterns and columns-per-row ratios n/m. An initial basis was computed by a crash procedure that preferred columns corresponding to larger diagonal entries of D. Algorithm 2 was then applied to the matrix A with $\rho = 2.0$. The sparse linear algebra operations were provided by subroutines of the author's LP solver.

In Table 2 the number of basis updates, the computation time and the volume increase are reported for each pass of the algorithm, where the latter quantity is defined as

$$\text{volinc} = \log_2 \left(\frac{\text{vol}(B_{\text{new}})}{\text{vol}(B_{\text{old}})} \right)$$

with B_{old} and B_{new} being the basis matrix at the beginning and end of the pass, respectively. volinc was computed from the fact that $\text{vol}(B_{\text{new}})/\text{vol}(B_{\text{old}})$ is the absolute value of the product of the pivot elements v_p from the current pass (see Lemma 2.3(i)).

name		pass 1	pass 2	pass 3	pass 4	pass 5	pass 6	pass 7	pass 8	total
	updates	2913	2	0						2915
bab3	volinc	1.39E+04	2.82E+00	0.00E+00						1.39E+04
	time	8.6	7.7	7.8						24.1
	updates	9495	18	1	1	0				9515
dbic1	volinc	4.48E+04	2.12E+01	1.19E+00	1.20E+00	0.00E+00				4.48E+04
	time	9.7	3.3	3.3	3.1	3.3				22.8
	updates	26409	296	107	38	26	10	1	0	26887
karted	volinc	1.00E+05	3.51E+02	1.20E+02	4.24E+01	2.81E+01	1.05E+01	1.14E+00	0.00E+00	1.01E+05
	time	1570.3	982.2	1009.0	1018.9	955.4	985.3	931.7	981.9	8434.6
	updates	36848	82	45	14	12	8	0		37009
nug30	volinc	1.40E+05	9.03E+01	4.84E+01	1.52E+01	1.24E+01	8.25E+00	0.00E+00		1.40E+05
	time	4931.4	4484.5	4536.3	4515.8	4436.0	4566.9	4519.9		31990.8
	updates	5670	0							5670
pds-100	volinc	1.10E+04	0.00E+00							1.10E+04
	time	15.6	14.8							30.4
	updates	19808	61	16	1	3	0			19889
rail02	volinc	6.34E+04	6.60E+01	1.73E+01	1.04E+00	3.04E+00	0.00E+00			6.35E+04
	time	159.5	105.4	104.8	102.8	105.0	102.7			680.3
	updates	3531	2	0						3533
rail4284	volinc	1.69E+04	2.24E+00	0.00E+00						1.69E+04
	time	219.8	215.8	217.2						652.8
	updates	12198	4	0						12202
rmine14	volinc	2.69E+04	4.29E+00	0.00E+00						2.69E+04
	time	43.7	34.6	36.3						114.7
	updates	45741	0							45741
srd120	volinc	1.07E+05	0.00E+00							1.07E+05
	time	142.8	124.1							266.9
	updates	5688	18	2	0					5708
stp3d	volinc	1.20E+04	1.99E+01	2.06E+00	0.00E+00					1.20E+04
	time	25.4	23.9	22.7	23.2					95.2
	updates	5434	0							5434
watson_2	volinc	3.49E+04	0.00E+00							3.49E+04
	time	14.4	4.1							18.5

Table 2: Statistics for Algorithm 2. Times are reported in seconds.

The total number of basis updates ranged between $0.03m$ and $0.85m$. This quantity depends on the initial basis as well as the order in which columns of A were scanned. An ideal scanning order would yield a maximum volume basis after one pass, but such an order is, of course, not known in practice. By using the natural order in Algorithm 2, the vast majority of updates and most of the volume increase were still attributed to the first pass.

The final pass shows the time for computing all entries of $A_{\mathcal{B}}^{-1}A_{\mathcal{N}}$ without an update. For most problems this was a large fraction of the time for pass 1, meaning that the scanning operations were the dominating cost of the algorithm. The large differences in the computation times were caused by sparsity in $A_{\mathcal{B}}^{-1}A_{\mathcal{N}}$. For some LP models (such as `pds-100` and `watson_2`) the tableau matrix is sparse for all relevant bases, allowing one to compute a column in much less than order of m time. Such problems are called "hypersparse" in simplex community (see Wunderling [81] and Hall and McKinnon [33]). In other cases the $A_{\mathcal{B}}^{-1}A_{\mathcal{N}}$ are dense matrices, so that one pass of the algorithm requires at least order of $m(n-m)$ arithmetic operations.

A general conclusion is that the ratio of number of basis updates to computation time is very uneconomical in all but the first pass. Similar observations were made by I. S. Duff and J. K. Reid (private communication), who experimented with the same approach on matrices from least squares problems.

For use in practice it can be appropriate to terminate the algorithm after one pass over the tableau matrix. Although there is no guarantee for $\left\|A_{\mathcal{B}}^{-1}A_{\mathcal{N}}\right\|_{\max}$ to be bounded, in the author's experience the basis is as effective as a maximum volume basis when applied as basis preconditioner. For application in the interior point solver, where a good starting basis is available, we will reduce the computation time even further by scanning only a heuristically chosen subset of the columns (Section 6.5).

2.4 A Rank Revealing Method

This section presents an application of the local maximum volume concept that was developed by the author in [73]. Given an arbitrary matrix $A \in \mathbb{R}^{m\times n}$, the task is to determine the rank of A in the numerical sense. By that we mean to determine an index r such that $\sigma_r \geq \varepsilon$ and $\sigma_{r+1} = O(\varepsilon)$ for a given tolerance $\varepsilon > 0$, where $\sigma_1 \geq \cdots \geq \sigma_d \geq 0$ ($d = \min(m,n)$) are the singular values of A. To treat the case $r = d$ conveniently, we define $\sigma_{d+1} := 0$. Our definition of numerical rank relaxes the condition $\sigma_r \geq \varepsilon > \sigma_{r+1}$, which is only achievable by computing the singular values. Note that the numerical rank in the relaxed definition need not be unique unless there is a clear gap in the spectrum. In addition to the rank r, an $r \times r$ submatrix of A shall be identified whose minimum singular value is not too much smaller than σ_r.

The stated problem arises, for example, in solving underdetermined least squares problems in which some of the model parameters should be removed to obtain a well defined solution. In this case it is not sufficient to determine the rank or the singular values of A; a well conditioned submatrix is explicitly required to find the set of redundant parameters. Because the matrices in least squares problems are often large and sparse, methods based on orthogonal factorizations can be expensive in terms of computation time and memory requirement. We are therefore interested in solving the above problem by Gaussian elimination, which is generally better suited for sparse matrices.

It is well known that Gaussian elimination with complete pivoting may not reveal a near singularity by means of a small pivot element. For the example from Peters and Wilkinson [66],

$$A = \begin{bmatrix} 1 & -1 & \cdots & -1 & -1 \\ & 1 & & & -1 \\ & & \ddots & & \vdots \\ & & & 1 & -1 \\ & & & & 1 \end{bmatrix} \in \mathbb{R}^{m\times m},$$

complete pivoting allows to choose the diagonal entries as pivots, so that no eliminations are needed and A is determined to be of full rank. It is not revealed that $\sigma_m(A) = O(2^{-m})$ (see [64, Section 5]) and the numerical rank of A to be $m-1$ for m moderately large.

We shall see that the numerical rank of A is revealed by a basis matrix of local maximum volume in $\mathbb{A} = \begin{bmatrix} A & \beta I_m \end{bmatrix}$ for a suitably chosen β. As in the proof of Lemma 2.4, a basis $\mathcal{B}$ for $\mathbb{A}$ uniquely defines a submatrix A_{11} of A through the partitioning

$$\mathbb{A}_{\mathcal{B}} = \begin{bmatrix} A_{11} & 0 \\ A_{21} & \beta I_{m-k} \end{bmatrix}, \quad \mathbb{A}_{\mathcal{N}} = \begin{bmatrix} \beta I_k & A_{12} \\ 0 & A_{22} \end{bmatrix}, \tag{2.3}$$

where the rightmost $m-k$ columns of $\mathbb{A}_{\mathcal{B}}$ and the leftmost k columns of $\mathbb{A}_{\mathcal{N}}$ are logical (the indices in $\mathcal{B}$ and $\mathcal{N}$ can always be permuted to obtain that form). The dimension of A_{11}, i. e. the number of structural columns in the basis, is the connection to the numerical rank of A.

Lemma 2.6. *Let $\varepsilon > 0$ and $\rho \geq 1$ be given parameters. Let $\beta = \min(m,n)\varepsilon\rho$ and $\mathcal{B}$ be a local ρ-maximum volume basis for $\mathbb{A} = \begin{bmatrix} A & \beta I_m \end{bmatrix}$. The dimension of A_{11} defined through (2.3) is the numerical rank of A under tolerance ε.*

Proof. Recall that the maximum absolute entry and the 2-norm of any matrix $A \in \mathbb{R}^{m\times n}$ are related by

$$\|A\|_{\max} \leq \|A\| \leq \sqrt{mn}\,\|A\|_{\max}. \tag{2.4}$$

Let r be the dimension of A_{11}. We need to show that $\sigma_r(A) \geq \varepsilon$ and $\sigma_{r+1}(A) = O(\varepsilon)$. Because $\mathbb{A}_{\mathcal{B}}$ has local ρ-maximum volume in $\mathbb{A}$, all entries of

$$\mathbb{A}_{\mathcal{B}}^{-1}\mathbb{A}_{\mathcal{N}} = \begin{bmatrix} \beta A_{11}^{-1} & A_{11}^{-1}A_{12} \\ -A_{21}A_{11}^{-1} & \beta^{-1}A/A_{11} \end{bmatrix} \tag{2.5}$$

are bounded by ρ in absolute value. (If $r = m$, the second block row is vacuous.) In particular

$$\left\|A_{11}^{-1}\right\|_{\max} \leq \rho\beta^{-1}, \quad \|A/A_{11}\|_{\max} \leq \rho\beta. \tag{2.6}$$

For the first part we use from the interlacing property of the singular values [23, Corollary 8.6.3] that for any $r \times r$ submatrix A_{11} of A, $\sigma_r(A_{11}) \leq \sigma_r(A)$. Therefore, by (2.4),

$$\frac{1}{\sigma_r(A)} \leq \frac{1}{\sigma_{\min}(A_{11})} = \left\|A_{11}^{-1}\right\| \leq r\left\|A_{11}^{-1}\right\|_{\max} \leq r\rho\beta^{-1} \leq \frac{1}{\varepsilon},$$

so that $\sigma_r(A) \geq \varepsilon$.

The second part is trivial if $r = m$. Otherwise we use from [64, Theorem 2.7] that for any nonsingular $r \times r$ submatrix A_{11} of A, $\sigma_{r+1}(A) \leq \|A/A_{11}\|$. Therefore, by (2.4),

$$\begin{aligned}\sigma_{r+1}(A) &\leq \|A/A_{11}\| \leq \|A/A_{11}\|_{\max}\sqrt{(m-r)(n-r)} \leq \beta\rho\sqrt{(m-r)(n-r)} \\ &= \varepsilon\rho^2\min(m,n)\sqrt{(m-r)(n-r)}.\end{aligned}$$

It follows that $\sigma_{r+1}(A)$ is bounded in terms of ε for fixed dimension of A. □

Notice from (2.6) how β balances between keeping the inverse of A_{11} bounded and getting the Schur complement small. The first condition achieves that $r \leq \operatorname{rank}(A)$ and the second condition that $r \geq \operatorname{rank}(A)$ in the numerical sense. Including ρ in the definition of β guarantees that $\sigma_r(A) \geq \varepsilon$ at the cost that the bound on $\sigma_{r+1}(A)$ grows with ρ^2. In computational practice a reasonable choice can be $\varepsilon = \varepsilon_{\text{mach}}\|A\|_{\max}$, where $\varepsilon_{\text{mach}}$ is the relative machine precision. The maximum-norm condition number of A_{11} is then bounded by $(r\varepsilon_{\text{mach}})^{-1}$, so that A_{11} is nonsingular in finite precision arithmetic.

It remains to be shown that A_{11} has the desired property that $\sigma_{\min}(A_{11})$ is close to $\sigma_r(A)$. The key ingredient here is to observe that A_{11} has local $(2\rho^2)$-maximum volume in A.

Lemma 2.7. *The submatrix A_{11} obtained in Lemma 2.6 has local $(2\rho^2)$-maximum volume in A.*

Proof. Because all entries of (2.5) are bounded by ρ in absolute value, the determinant of any submatrix $\begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix}$ of $A^\times$ with an entry in each block is bounded by

$$\begin{aligned} |s_{11}s_{22} - s_{21}s_{12}| &\leq \left\|A_{11}^{-1}\right\|_{\max} \|A/A_{11}\|_{\max} + \left\|A_{21}A_{11}^{-1}\right\|_{\max} \left\|A_{11}^{-1}A_{12}\right\|_{\max} \\ &\leq \rho\beta^{-1}\rho\beta + \rho\rho = 2\rho^2. \end{aligned}$$

The claim follows from Corollary 2.5. □

The following two lemmas derive bounds on $\sigma_{\min}(A_{11})$ and $\|A/A_{11}\|_{\max}$ that hold for any submatrix A_{11} of local maximum volume in A.

Lemma 2.8. *Let $A \in \mathbb{R}^{m\times n}$ and A_{11} be a $k\times k$ submatrix ($k < \min(m,n)$) of local ρ-maximum volume. Then*

$$\|A/A_{11}\|_{\max} \leq \rho(k+1)\sigma_{k+1}(A).$$

Proof. The inequality was proved by Goreinov and Tyrtyshnikov [29, Theorem 2.1] under the assumption that A_{11} has global maximum volume in A. Because their proof actually only uses that A_{11} has maximum volume in all containing $(k+1)\times(k+1)$ submatrices of A, the result remains valid for our definition of local maximum volume. The proof is repeated here for completeness.

Consider any $(k+1)\times(k+1)$ submatrix of A of the form

$$\hat{A} = \begin{bmatrix} A_{11} & \boldsymbol{b} \\ \boldsymbol{c}^T & \alpha \end{bmatrix}.$$

Then $\gamma = \alpha - \boldsymbol{c}^T A_{11}^{-1}\boldsymbol{b}$ is an entry of A/A_{11} and each entry of A/A_{11} has this form for a particular $\hat{A}$. Therefore it suffices to show that $|\gamma| \leq \rho(k+1)\sigma_{k+1}(A)$.

If $\hat{A}$ is singular, then $\gamma = 0$ and the claim is trivial. Otherwise a straightforward computation verifies that

$$\hat{A}^{-1} = \gamma^{-1}\begin{bmatrix} H & \boldsymbol{f} \\ \boldsymbol{g}^T & 1 \end{bmatrix},$$

where

$$\boldsymbol{f} = -A_{11}^{-1}\boldsymbol{b}, \quad \boldsymbol{g} = -A_{11}^{-T}\boldsymbol{c}, \quad H = \gamma A_{11}^{-1} + \boldsymbol{f}\boldsymbol{g}^T.$$

Considering

$$\hat{A}^\times = \begin{bmatrix} A_{11}^{-1} & A_{11}^{-1}\boldsymbol{b} \\ -\boldsymbol{c}^T A_{11}^{-1} & \gamma \end{bmatrix}$$

and applying Corollary 2.5 shows that all entries of $\boldsymbol{f}$, $\boldsymbol{g}$ and H are bounded by ρ in absolute value. Hence $\left\|\hat{A}^{-1}\right\|_{\max} \leq |\gamma^{-1}|\rho$. It follows that

$$|\gamma| \leq \rho\frac{1}{\left\|\hat{A}^{-1}\right\|_{\max}} \leq \rho\frac{k+1}{\left\|\hat{A}^{-1}\right\|} = \rho(k+1)\sigma_{k+1}(\hat{A}) \leq \rho(k+1)\sigma_{k+1}(A),$$

where the last inequality comes from the interlacing property of singular values [23, Corollary 8.6.3]. □

Lemma 2.9. *Let $A \in \mathbb{R}^{m\times n}$ and A_{11} be a $k\times k$ submatrix of local ρ-maximum volume. Then*

$$\sigma_k(A) \leq \rho\sqrt{(m-k+1)(n-k+1)}k\sigma_k(A_{11}).$$

Proof. If $k = 1$, then A_{11} is scalar and because of local ρ-maximum volume it satisfies $\rho|A_{11}| \geq \|A\|_{\max}$. Therefore

$$\sigma_1(A) \leq \sqrt{mn}\,\|A\|_{\max} \leq \sqrt{mn}\rho|A_{11}| = \rho\sqrt{mn}\sigma_1(A_{11}).$$

If $k > 1$, let B be a $(k-1) \times (k-1)$ submatrix of A_{11} of 1-maximum volume in A_{11}. Consider any $k \times k$ submatrix of A of the form

$$A'_{11} = \begin{bmatrix} B & \boldsymbol{b} \\ \boldsymbol{c}^T & \alpha \end{bmatrix}.$$

Because A'_{11} differs from A_{11} by at most one row and one column, and because A_{11} has local ρ-maximum volume in A,

$$\rho\,\mathrm{vol}(A_{11}) \geq \mathrm{vol}(A'_{11}).$$

From the determinant property of the Schur complement, $\det(A_{11}) = \det(B)\det(A_{11}/B)$, it follows that for the scalar A_{11}/B

$$\rho|A_{11}/B| = \rho\frac{\mathrm{vol}(A_{11})}{\mathrm{vol}(B)} \geq \frac{\mathrm{vol}(A'_{11})}{\mathrm{vol}(B)} = |A'_{11}/B|.$$

Because A'_{11}/B is an entry of A/B and each entry of A/B has this form for a particular A'_{11}, we obtain

$$\rho|A_{11}/B| \geq \|A/B\|_{\max}.$$

Therefore

$$\begin{aligned}\sigma_k(A) \leq \|A/B\| &\leq \sqrt{(m-k+1)(n-k+1)}\,\|A/B\|_{\max} \\ &\leq \rho\sqrt{(m-k+1)(n-k+1)}|A_{11}/B| \\ &\leq \rho\sqrt{(m-k+1)(n-k+1)}k\sigma_k(A_{11}),\end{aligned}$$

where the first inequality is from [64, Theorem 2.7] and the last inequality is obtained by applying Lemma 2.8 to the $(k-1) \times (k-1)$ submatrix B of A_{11}. □

The final theorem summarizes the bounds on the singular values of A that are revealed by the submatrix A_{11} determined through a ρ-maximum volume basis in $\mathbb{A}$.

Theorem 2.10. *Let the submatrix A_{11} that is obtained from Lemma 2.6 have dimension $k \times k$. Then*

$$\sigma_k(A) \geq \sigma_{\min}(A_{11}) \geq \frac{1}{2\rho^2 k\sqrt{(m-k+1)(n-k+1)}}\sigma_k(A), \tag{2.7}$$

$$\sigma_{k+1}(A) \leq \|A/A_{11}\| \leq 2\rho^2(k+1)\sqrt{(m-k)(n-k)}\sigma_{k+1}(A). \tag{2.8}$$

Proof. The left inequalities hold for any nonsingular $k \times k$ submatrix. The left side of (2.7) follows from the interlacing property of the singular values [23, Corollary 8.6.3] and the left side of (2.8) was proved by Pan [64, Theorem 2.7]. The right inequalities require that A_{11} has $(2\rho^2)$-maximum volume in A (see Lemma 2.7) and follow from Lemmas 2.9 and 2.8. □

2.5 Implementation for Dense Matrices

The theory from the previous section leads to an algorithm for finding a square nonsingular submatrix of A that reveals the numerical rank:

Algorithm 3 Rank Revealing Method

Input: $A \in \mathbb{R}^{m\times n}$

1: Choose $\rho \geq 1$, $\beta > 0$ and set $\mathbb{A} = \begin{bmatrix} A & \beta I_m \end{bmatrix}$.
2: Initialize $\mathcal{B} = \{n+1, \ldots, n+m\}$ and use Algorithm 1 to update $\mathcal{B}$ to a ρ-maximum volume basis for $\mathbb{A}$.
3: Obtain A_{11} through the partitioning (2.3).

A tableau form implementation of Algorithm 3 has been written for testing purposes. Initially the matrix $\bar{\mathbb{A}} = \begin{bmatrix} A & I_m \end{bmatrix}$ is stored and $\mathcal{B} = \{n+1, \ldots, n+m\}$. Logical columns are not explicitly scaled by β to avoid values with very different orders of magnitude in the computation. Instead multiplications with β and β^{-1} are applied on the fly when logical columns are involved. Each iteration of the update procedure in step 2 chooses a pivot element from

$$\bar{\mathbb{A}}_{\mathcal{B}}^{-1}\bar{\mathbb{A}}_{\mathcal{N}} = \begin{bmatrix} A_{11}^{-1} & A_{11}^{-1}A_{12} \\ -A_{21}A_{11}^{-1} & A/A_{11} \end{bmatrix} \tag{2.9}$$

and transforms the corresponding column of $\bar{\mathbb{A}}_{\mathcal{B}}^{-1}\bar{\mathbb{A}}$ into a unit column by applying row operations. The pivot element is chosen in the following order:

(i) If $|\bar{\mathbb{A}}_{\mathcal{B}}^{-1}\bar{\mathbb{A}}|$ has entries corresponding to block A_{11}^{-1} that are larger than $\rho\beta^{-1}$, the maximum such entry is chosen as pivot.

(ii) If $|\bar{\mathbb{A}}_{\mathcal{B}}^{-1}\bar{\mathbb{A}}|$ has entries corresponding to block $A_{11}^{-1}A_{12}$ or $-A_{21}A_{11}^{-1}$ that are larger than ρ, the maximum such entry is chosen as pivot.

(iii) If $|\bar{\mathbb{A}}_{\mathcal{B}}^{-1}\bar{\mathbb{A}}|$ has entries corresponding to block A/A_{11} that are larger than $\rho\beta$, the maximum such entry is chosen as pivot.

The order promotes having logical columns in the basis, which is advantageous for numerical stability. If none of the cases (i)–(iii) yields a pivot element, the update procedure terminates.

A test set of real-world matrices was collected from the San Jose State University Singular Matrix Database [19], using the 327 matrices (as of January 2018) for which $\min(m,n) \leq 1000$. The matrices were transposed if necessary so that $m \leq n$. For comparison a singular value decomposition (SVD) of each matrix A was computed and the numerical rank of A was determined as the largest index s such that

$$\sigma_s(A) \geq \max(m,n)\varepsilon_{\text{mach}}\sigma_1(A).$$

This criterion is used by the MATLAB `rank` function, and all matrices in the test set were rank deficient in its sense. In our algorithm β was chosen comparably as $\max(m,n)\varepsilon_{\text{mach}} \|A\|_{\max}$ and ρ was initially set to 2.0.

For 56 matrices the numerical ranks determined by SVD and the maximum volume basis (MVB) differed. This is legitimate if there is no large gap between any two consecutive singular values. To verify that the MVB rank r is acceptable with respect to the singular values of A, the ratios $\sigma_r(A)/\sigma_s(A)$ and $\sigma_{r+1}(A)/\sigma_{s+1}(A)$ are plotted in Figure 1 for those matrices where $r \neq s$. Because the ratios are not too far away from 1.0 it can be concluded that $\sigma_{r+1}(A) = O(\sigma_{s+1}(A))$ and $\sigma_r(A) = \Omega(\sigma_s(A))$. Hence for the chosen β the rank determined by the MVB is in accordance with the spectrum of A.

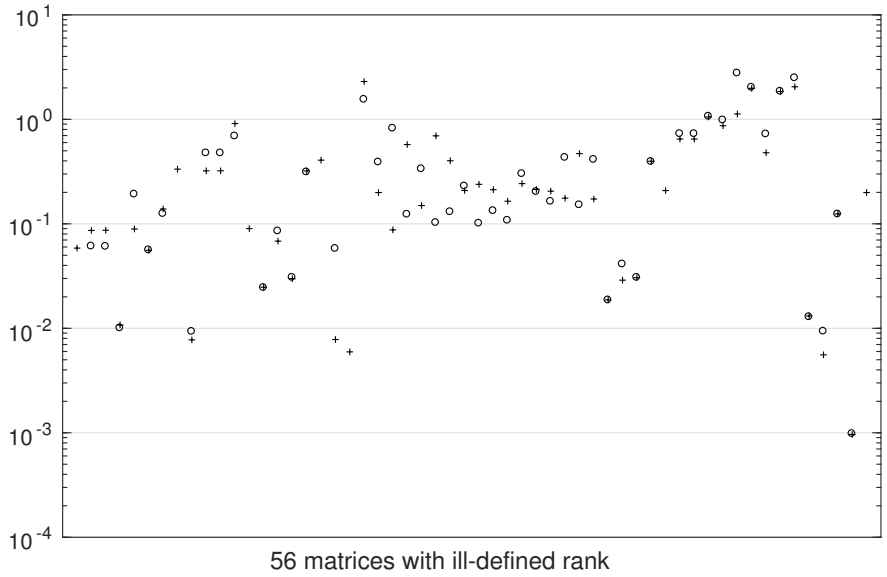


Figure 1: Ratios $\sigma_r(A)/\sigma_s(A)$ ("+") and $\sigma_{r+1}(A)/\sigma_{s+1}(A)$ ("o") for matrices with $r \neq s$. The "o" marker is missing when $r = m$ (7 matrices).

In all test cases A_{11} satisfied $\sigma_r(A_{11})/\sigma_r(A) \geq 0.004$ and the geometric mean of these ratios was 0.25, confirming that the maximum volume criterion yields a good submatrix in practice. The ratios did not improve relevantly for ρ closer to 1.

Because the update procedure in Algorithm 3 starts from the logical basis, a minimum of r basis updates is needed. For $\rho = 2.0$ the average number of basis updates (by taking the geometric mean over the 327 matrices) was $1.002r$. The computation time of the tableau form algorithm in this case is about twice the time for an LU factorization with complete pivoting (which is not rank revealing). For $\rho = 1.1$ and $\rho = 1.001$ the average number of pivots increased to $1.24r$ and $1.68r$, respectively.

2.6 Comparison to Pan's Method

Other than the method from Section 2.4, the only practical rank revealing factorization that does not require orthogonal operations is that of Pan [64]. Pan's method uses the maximum volume concept in a different manner to find a square submatrix of $A \in \mathbb{R}^{m\times n}$. Assuming that $k \leq \text{rank}(A)$ is given, the method first chooses a slice of k columns, say $A_{\mathcal{J}}$, that has local ρ-maximum volume in A. In a second step it chooses a $k \times k$ submatrix of local ρ-maximum volume in $A_{\mathcal{J}}$. Let us say that the resulting submatrix A_{11} has "normal" ρ-maximum volume in A to distinguish it from a local maximum volume submatrix in our definition. Pan proved the following bounds on the singular values for $m = n$:

$$\sigma_k(A) \geq \sigma_{\min}(A_{11}) \geq \frac{1}{k(n-k)\rho^2+1}\sigma_k(A),$$
$$\sigma_{k+1}(A) \leq \|A/A_{11}\| \leq \big(k(n-k)\rho^2+1\big)\,\sigma_{k+1}(A).$$

These bounds are almost identical to those in Theorem 2.10. Although the setting in [64] assumed the dimension of A_{11} to be given, choosing it by means of a tolerance ε on $\sigma_k(A)$ can be easily incorporated into the algorithm for finding the column subset $\mathcal{J}$. Therefore Pan's method and our method provide the same functionality.

Regarding their implementation, Pan's method has a drawback. Its first step requires computations with the normal matrix A^TA for finding the column subset $\mathcal{J}$ as outlined in Section 2.2. For dense matrices these operations are implementable but offer no advantage over rank revealing orthogonal methods, which require roughly the same number of arithmetic operations (see the comparison in [64]). For sparse matrices the algorithm would require to compute and update a sparse Cholesky factorization of $A_{\mathcal{J}}^T A_{\mathcal{J}}$. While this is possible (see Davis and Hager [14]), it can be assumed to be much more costly than operations with a basis matrix of $\begin{bmatrix} A & I_m \end{bmatrix}$.

While analysing the methods, the author suspected that normal and local maximum volume might be the same property, but such a conjecture turned out to be false. Two examples are given to prove that neither property implies the other. First, consider

$$A = \begin{bmatrix} 1 & & & 1 \\ & 1 & & 1 \\ & & 1 & 1 \\ 1 & 1 & 1 & \\ 1 & & & \\ & 1 & & \\ & & 1 & \end{bmatrix}.$$

It can be computed analytically that the singular values of the submatrix formed of any three columns are $\left(\sqrt{5}, \sqrt{2}, \sqrt{2}\right)$, so that the first three columns have local maximum volume in A. The leading 3×3 block A_{11} obviously has local maximum volume within the first three columns, so that it has normal maximum volume in A. However, A_{11} does not have global maximum volume in the leading 4×4 block because exchanging columns 3 and 4 and rows 3 and 4 increases $\mathrm{vol}(A_{11})$ by a factor 2. (This is easily verified from Lemma 2.4.) Therefore A_{11} does not have local maximum volume in A.

For the opposite part consider

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ d & -1 & -d \\ -1 & d & -d \end{bmatrix}$$

with $d = 0.99$ and let A_{11} be the leading 2×2 block. By examining the determinant of the relevant 2×2 submatrices of $A^{\times}$ it can be verified that A_{11} has local maximum volume in A. However, by computing singular values we obtain the volume of the matrix composed of columns 1 and 2 to be ≈ 2.23 and the volume of the matrix composed of columns 1 and 3 to be ≈ 2.42. Hence the first two columns do not have local maximum volume in A, and neither does A_{11} have normal maximum volume in A.

3 Basis Preconditioning for Least Squares

The least squares problem

$$\underset{\boldsymbol{y}}{\text{minimize}} \ \left\|\boldsymbol{c} - A^T\boldsymbol{y}\right\|^2 \tag{3.1}$$

arises frequently as a subproblem in algorithms for numerical optimization. Its native interpretation is to find a vector of model parameters $\boldsymbol{y}$ for fitting model predictions $A_j^T\boldsymbol{y}$ to measurements c_j in such a way that the 2-norm of the residual $\boldsymbol{x} = \boldsymbol{c} - A^T\boldsymbol{y}$ is minimum. While stated as an optimization problem, it actually means solving the "KKT system"

$$\begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{b} \end{pmatrix} \tag{3.2}$$

for $\boldsymbol{b} = \boldsymbol{0}$. For most of this chapter we assume the $m \times n$ matrix A to have rank $m \leq n$, so that the KKT matrix is nonsingular and (3.1) has a unique minimizer.

Basis preconditioning transforms the KKT system by means of a basis matrix $A_{\mathcal{B}}$ to make its spectrum more favourable for an iterative method. For ill conditioned problems, as those arising in the linear programming IPM, preconditioning is key to obtaining an acceptable convergence rate of the linear solver. This chapter discusses basis preconditioning for (3.2) and presents features of the method that are particularly useful in the context of parameter estimation. To simplify notation, let $B = A_{\mathcal{B}}$ and $N = A_{\mathcal{N}}$ throughout.

3.1 Spectrum of the Preconditioned Matrix

It is common practice to reduce the KKT system (3.2) to normal equations

$$AA^T\boldsymbol{y} = A\boldsymbol{c} - \boldsymbol{b} =: \boldsymbol{r}. \tag{3.3}$$

If A has rank m, the normal matrix is positive definite, whereas the KKT matrix has n positive and m negative eigenvalues. A more precise characterization of their spectra is given by the following lemma.

Lemma 3.1. *Let the $m \times n$ matrix A have rank m. The eigenvalues of $\begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix}$ are 1 with multiplicity $n - m$ and $\frac{1}{2} \pm \sqrt{\frac{1}{4} + \nu_k}$, where ν_k, for $1 \leq k \leq m$, are the eigenvalues of AA^T.*

Proof. By standard argument. Consider the linear equations

$$\begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \lambda \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix}$$

for a scalar $\lambda \neq 0$. If $\boldsymbol{y} = \boldsymbol{0}$, solutions are obtained if $\boldsymbol{x} \in \text{null}(A)$ and $\lambda = 1$, yielding $n - m$ eigenvectors with eigenvalue 1. If $\boldsymbol{y} \neq \boldsymbol{0}$, it follows from

$$\begin{aligned} & A\boldsymbol{x} + AA^T\boldsymbol{y} = \lambda A\boldsymbol{x} \\ \Longrightarrow \quad & AA^T\boldsymbol{y} = (\lambda - 1)A\boldsymbol{x} \\ \Longrightarrow \quad & AA^T\boldsymbol{y} = (\lambda - 1)\lambda\boldsymbol{y} \end{aligned}$$

that $\boldsymbol{y}$ is eigenvector to AA^T with eigenvalue $(\lambda - 1)\lambda$. Solving $(\lambda - 1)\lambda = \nu_k$ for $1 \leq k \leq m$ gives the remaining $2m$ eigenvalues of the KKT matrix. □

Basis preconditioning can be applied to the KKT system or the normal equations. Given a basis matrix B and defining $\boldsymbol{u} = B^T\boldsymbol{y}$, the symmetric variant transforms (3.2) into

$$\begin{bmatrix} I & A^TB^{-T} \\ B^{-1}A & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{u} \end{pmatrix} = \begin{pmatrix} \boldsymbol{c} \\ B^{-1}\boldsymbol{b} \end{pmatrix} \tag{3.4}$$

and (3.3) into

$$B^{-1}AA^TB^{-T}\boldsymbol{u} = B^{-1}\boldsymbol{r}. \tag{3.5}$$

Notice that (3.5) are the normal equations corresponding to (3.4), so that the spectra of the preconditioned matrices are related through Lemma 3.1. It therefore suffices to analyse the numerical properties of one system.

The preconditioned normal matrix is symmetric positive definite and writing it as

$$C := B^{-1}AA^TB^{-T} = I_m + B^{-1}NN^TB^{-T}$$

shows that its eigenvalues are bounded from below by 1. The preconditioning will therefore be effective if $\sigma_{\max}(C) = \left\|B^{-1}A\right\|^2$ is of moderate size. This need not be the case if B is obtained through LU factorization of A, as it is often done in practice. For example, the partitioning into

$$B = \begin{bmatrix} 1 & -1 & & \cdots & -1 \\ & 1 & & & -1 \\ & & \ddots & & \vdots \\ & & & 1 & -1 \\ & & & & 1 \end{bmatrix}, \quad N = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}$$

could have been obtained by LU factorization with any type of pivoting; yet $\left\|B^{-1}A\right\|$ grows exponentially with m.

Arioli and Duff [7] were the first to consider the maximum volume criterion for choosing a basis preconditioner. A (local) ρ-maximum volume basis bounds the entries in $B^{-1}A$ by $\rho \geq 1$ in absolute value, so that

$$\left\|B^{-1}A\right\|^2 = 1 + \left\|B^{-1}N\right\|^2 \leq 1 + \left\|B^{-1}N\right\|_F^2 \leq 1 + \rho^2 \operatorname{nnz}(B^{-1}N) \leq 1 + \rho^2 m(n-m).$$

Because a maximum volume basis exists for any matrix A of full row rank, basis preconditioning can guarantee a $1 + m(n-m)$ bound on the condition number of the preconditioned normal matrix, independently of the singular values of A. If the tableau matrix is sparse, the relevant quantity for the bound is the number of nonzero entries.

The convergence rate of an iterative method actually depends on the eigenvalue distribution of the matrix rather than its condition number, so that minimizing $\left\|B^{-1}A\right\|$ should not be the ultimate goal for a preconditioner. A quantity that characterizes the whole spectrum of C is its nuclear norm $\|C\|_*$, which is the sum of its eigenvalues. Because the nuclear norm of a symmetric positive semidefinite matrix is the sum of its diagonal entries, we have

$$\|C\|_* = m + \left\|B^{-1}NN^TB^{-T}\right\|_* = m + \left\|B^{-1}N\right\|_F^2.$$

Hence minimizing the Frobenius norm of $B^{-1}N$ minimizes the nuclear norm of C, which is likely to avoid a uniform spectrum. While Arioli and Duff argued for the maximum volume basis because they could not find an algorithmic approach for choosing B such that $\left\|B^{-1}A\right\|$ is minimum, the maximum volume criterion seems to be preferable anyway because it directly bounds $\left\|B^{-1}N\right\|_F$.

It is insightful to derive an update formula for C when basis index $\mathcal{B}_p$ is replaced by $j \in \mathcal{N}$. Denote $\boldsymbol{w}^T = \boldsymbol{e}_p^TB^{-1}A$, $\boldsymbol{v} = B^{-1}A_j$ and B' the updated basis matrix. The update of $B^{-1}A$

proceeds by adding multiples of row $\boldsymbol{w}^T$ to other rows such that column j becomes unit column $\boldsymbol{e}_p$:

$$(B')^{-1}A = B^{-1}A + \begin{pmatrix} -v_1/v_p \\ \vdots \\ -(v_p-1)/v_p \\ \vdots \\ -v_m/v_p \end{pmatrix} \boldsymbol{w}^T =: B^{-1}A + \boldsymbol{\eta}\boldsymbol{w}^T.$$

Here $\boldsymbol{\eta}$ has been defined as the vector of multipliers. The pivot operation changes C to

$$\begin{aligned} C' &= C + B^{-1}A\boldsymbol{w}\boldsymbol{\eta}^T + \boldsymbol{\eta}\boldsymbol{w}^T A^T B^{-T} + \boldsymbol{\eta}\boldsymbol{w}^T\boldsymbol{w}\boldsymbol{\eta}^T \\ &= C + B^{-1}AA^T B^{-T}\boldsymbol{e}_p\boldsymbol{\eta}^T + \boldsymbol{\eta}\boldsymbol{e}_p^T B^{-1}AA^T B^{-T} + \|\boldsymbol{w}\|^2 \boldsymbol{\eta}\boldsymbol{\eta}^T \\ &= C + \boldsymbol{c}\boldsymbol{\eta}^T + \boldsymbol{\eta}\boldsymbol{c}^T + \|\boldsymbol{w}\|^2 \boldsymbol{\eta}\boldsymbol{\eta}^T \\ &= C + \frac{1}{2}(\boldsymbol{c}+\boldsymbol{\eta})(\boldsymbol{c}+\boldsymbol{\eta})^T - \frac{1}{2}(\boldsymbol{c}-\boldsymbol{\eta})(\boldsymbol{c}-\boldsymbol{\eta})^T + \|\boldsymbol{w}\|^2 \boldsymbol{\eta}\boldsymbol{\eta}^T, \end{aligned} \tag{3.6}$$

where $\boldsymbol{c} = C_p$. From the final form it is obvious that C changes by a rank-2 operation and $\|C\|_*$ changes by

$$\frac{1}{2}\|\boldsymbol{c}+\boldsymbol{\eta}\|^2 - \frac{1}{2}\|\boldsymbol{c}-\boldsymbol{\eta}\|^2 + \|\boldsymbol{w}\|^2\|\boldsymbol{\eta}\|^2 = 2\boldsymbol{c}^T\boldsymbol{\eta} + \|\boldsymbol{w}\|^2\|\boldsymbol{\eta}\|^2. \tag{3.7}$$

The formula shows that $\|C\|_*$ decreases with a basis update only if $\boldsymbol{c}^T\boldsymbol{\eta}$ is sufficiently negative. This need not be the case for an update that increases $\mathrm{vol}(B)$. For example, consider

$$A = \begin{bmatrix} B & N \end{bmatrix} = \begin{bmatrix} 1 & & & +2 & 1 & \cdots & 1 \\ & 1 & & -2 & 0 & \cdots & 0 \\ & & 1 & -2 & 0 & \cdots & 0 \\ & & & 1 & -2 & 0 & \cdots & 0 \end{bmatrix},$$

where the last column is repeated 10 times. It is easily verified that the spectrum of C is $\{21, 7, 1, 1\}$. After pivoting on entry $+2$, all entries in the tableau are bounded by 1, so that the new basis has maximum volume. The update changes the spectrum of C to $\{36.75, 1, 1, 1\}$ and therefore increases $\|C\|_*$. The gist of the example is that the pivot operation fills in a large part of the tableau matrix and thereby increases its Frobenius norm. Although in this particular case the spectrum after the update is desirable (C has only two distinct eigenvalues), in general we can expect that a larger $\|C\|_*$ leads to slower convergence of an iterative method.

It seems possible to derive an algorithm from (3.7) that works similarly to the maximum volume algorithms from Chapter 2 but yields a basis that minimizes $\|B^{-1}N\|_F$ among all neighbouring bases. Choosing the updates would be significantly more expensive, but the smaller Frobenius norm might lead to a more favourable spectrum of the preconditioned matrix. In particular, the method is likely to yield a sparse tableau matrix if it exists. The idea has not been implemented for the present work, but is kept in mind for further investigation.

3.2 Numerical Investigation

It is informative to compare the bounds on the spectrum of C discussed above on some real-world problems. A set of 15 matrices was obtained from an LP interior point solver, where $A = \hat{A}D$ with $\hat{A}$ being the constraint matrix from the LP model and D a positive definite diagonal matrix. D was recorded in the first interior point iteration in which the iterative method for solving $AA^T\boldsymbol{x} = \boldsymbol{b}$ did not converge in 500 iterations with a diagonal preconditioner. The

name	m	n	$\mathrm{nnz}(A)$	$\sigma_{\min}(AA^T)$	$\sigma_{\max}(AA^T)$
bab3	22,478	415,935	3,120,277	1.30E−07	1.89E+02
buildingenergy	225,031	353,726	908,875	9.54E−02	5.46E+06
datt256	9,863	206,010	1,134,485	1.81E−04	4.40E+02
L1_sixm250obs	154,168	462,452	794,069	7.19E−01	1.69E+06
mining	661,094	1,010,052	3,415,524	6.96E−07	4.04E+06
netdiversion	99,581	228,549	595,459	7.07E−09	3.41E−02
ns1853823	223,144	436,320	1,570,068	5.39E−05	5.71E+02
nug20	14,098	86,644	296,004	7.91E−07	1.34E−01
pds-100	94,994	528,861	1,028,307	2.35E−06	7.67E+01
rail03	129,647	696,742	1,479,165	2.22E−04	1.94E+02
srd060	93,200	271,710	2,889,410	2.80E+04	2.17E+10
stp3d	97,936	235,582	598,689	3.68E−05	2.37E+01
ts-palko	22,002	69,237	1,098,905	2.27E+00	1.48E+07
vpphard2	136,399	275,633	698,357	5.98E−05	3.87E+02
watson_2	185,474	564,460	1,225,712	1.11E+01	2.67E+07

Table 3: Test set of scaled LP matrices from an interior point solver.

name	$\sigma_{\max}(C)$	$\left\|B^{-1}N\right\|_F^2$	$\rho^2\,\mathrm{nnz}(B^{-1}N)$	$\rho^2 m(n-m)$
bab3	1.10E+04	3.12E+05	3.86E+08	3.54E+10
buildingenergy	2.16E+06	7.52E+06	4.07E+10	1.16E+11
datt256	1.44E+06	1.45E+07	4.14E+09	7.74E+09
L1_sixm250obs	6.62E+04	3.86E+05	1.39E+10	1.90E+11
mining	3.08E+03	2.85E+05	5.89E+08	9.23E+11
netdiversion	3.70E+04	1.62E+06	6.37E+07	5.14E+10
ns1853823	7.46E+05	3.17E+06	2.17E+10	1.90E+11
nug20	1.95E+06	2.11E+07	4.03E+09	4.09E+09
pds-100	1.00E+05	1.16E+06	4.75E+08	1.65E+11
rail03	1.36E+05	2.24E+06	1.60E+09	2.94E+11
srd060	3.00E+01	1.32E+05	2.99E+06	6.65E+10
stp3d	2.41E+05	2.81E+06	6.63E+09	5.39E+10
ts-palko	5.29E+04	4.30E+05	4.16E+09	4.16E+09
vpphard2	2.53E+03	3.19E+04	7.51E+06	7.60E+10
watson_2	2.50E+03	4.76E+05	2.32E+07	2.81E+11

Table 4: Basis preconditioning with a 2.0-maximum volume basis.

matrices and their dimensions are listed in Table 3. The lower and upper ends of the spectrum of AA^T show that the normal matrices are moderately ill conditioned.

For each matrix A a ρ-maximum volume basis with $\rho = 2.0$ was computed. In Table 4 the quantities discussed in the previous section are compared, for which we have

$$\sigma_{\max}(C) \leq 1 + \left\|B^{-1}N\right\|_F^2 \leq 1 + \rho^2 \operatorname{nnz}(B^{-1}N) \leq 1 + \rho^2 m(n-m).$$

Because $\sigma_{\min}(C) \geq 1$, it is certain for all problems except `nug20` that the preconditioning decreased the 2-norm condition number of the normal matrix. When $\sigma_{\min}(AA^T)$ was close to zero, the whole spectrum was shifted to the right and the maximum eigenvalue increased. For most of the matrices $\left\|B^{-1}N\right\|_F^2$ was about one order of magnitude larger than $\sigma_{\max}(C)$, whereas the bound given by $\rho^2 \operatorname{nnz}(B^{-1}N)$ was loose. This means that the bound on $\operatorname{cond}(C)$ given by the maximum volume property is far too conservative in practice, even if sparsity of the tableau matrix is taken into account.

3.3 Additional Columns

For some matrices it can be necessary or advantageous to treat a set of columns separately. These can be columns that were added to the problem after a (good) basis was determined, or columns that have significantly more nonzeros than the columns in the other part. The first case arises in parameter estimation when a few measurements are added to the least squares problem after computing an initial estimate. The second case frequently occurs for LP matrices.

Assume that

$$A = \begin{bmatrix} A_{\mathcal{B}} & A_{\mathcal{N}} & A_{\mathcal{R}} \end{bmatrix},$$

where $\mathcal{R}$ is the set of n_r "remaining" columns. For the moment it is assumed that a basis matrix exists after dropping $A_{\mathcal{R}}$ from A. The basis preconditioned normal matrix becomes

$$C = A_{\mathcal{B}}^{-1} A A^T A_{\mathcal{B}}^{-T} = I_m + A_{\mathcal{B}}^{-1} A_{\mathcal{R}} A_{\mathcal{R}}^T A_{\mathcal{B}}^{-T} + A_{\mathcal{B}}^{-1} A_{\mathcal{N}} A_{\mathcal{N}}^T A_{\mathcal{B}}^{-T} =: M + A_{\mathcal{B}}^{-1} A_{\mathcal{N}} A_{\mathcal{N}}^T A_{\mathcal{B}}^{-T}.$$

Because $A_{\mathcal{R}}$ was ignored while choosing the basis, C can have n_r large eigenvalues. The idea to bound the spectrum of C is using M as preconditioner for the system $C\boldsymbol{u} = A_{\mathcal{B}}^{-1}\boldsymbol{r}$. Assuming that n_r is small, the second term of M is a low-rank matrix and an inverse representation is obtained from the Sherman-Morrison-Woodbury formula as

$$M^{-1} = I_m - A_{\mathcal{B}}^{-1} A_{\mathcal{R}} (I_{n_r} + A_{\mathcal{R}}^T A_{\mathcal{B}}^{-T} A_{\mathcal{B}}^{-1} A_{\mathcal{R}})^{-1} A_{\mathcal{R}}^T A_{\mathcal{B}}^{-T}.$$

Operations with M^{-1} can be implemented either by storing the matrix $A_{\mathcal{B}}^{-1} A_{\mathcal{R}}$ explicitly (usually in dense format) or by performing matrix-vector products with $A_{\mathcal{R}}$ and $A_{\mathcal{R}}^T$ and a forward and transpose solve with $A_{\mathcal{B}}$. In any case the columns of $A_{\mathcal{B}}^{-1} A_{\mathcal{R}}$ need to be computed to assemble the matrix in parenthesis for its Cholesky factorization.

For analysing the spectrum of the preconditioned system we can assume left-preconditioning with M, recalling that the eigenvalues of $M^{-1}C$ are those of $M^{-1/2} C M^{-1/2}$ and therefore real and positive. From

$$M^{-1}C = I_m + M^{-1} A_{\mathcal{B}}^{-1} A_{\mathcal{N}} A_{\mathcal{N}}^T A_{\mathcal{B}}^{-T}$$

it follows that $\sigma_{\min}(M^{-1}C) \geq 1$, so that the lower bound on the smallest eigenvalue established by basis preconditioning is maintained. Furthermore

$$\begin{aligned} \left\|M^{-1}C\right\| &= 1 + \left\|M^{-1} A_{\mathcal{B}}^{-1} A_{\mathcal{N}} A_{\mathcal{N}}^T A_{\mathcal{B}}^{-T}\right\| \\ &\leq 1 + \left\|M^{-1}\right\| \left\|A_{\mathcal{B}}^{-1} A_{\mathcal{N}} A_{\mathcal{N}}^T A_{\mathcal{B}}^{-T}\right\| \\ &\leq 1 + \left\|A_{\mathcal{B}}^{-1} A_{\mathcal{N}} A_{\mathcal{N}}^T A_{\mathcal{B}}^{-T}\right\|, \end{aligned}$$

where the last inequality holds because the eigenvalues of M are ≥ 1. Hence the maximum eigenvalue of $M^{-1}C$ is bounded by that of the basis preconditioned system without the columns from $\mathcal{R}$.

The technique is attractive if separating a small number of "dense" columns leads to a sparse tableau matrix, since then $\left\|A_{\mathcal{B}}^{-1}A_{\mathcal{N}}\right\|_F$ obtained by a maximum volume basis can be much smaller than if the whole matrix A was used. Excluding dense columns may also lead to faster linear algebra operations in the search for a basis. Prominent examples are LP matrices with dual block angular structure, in which the constraint matrix becomes block diagonal after removing a set of linking columns (see, for example, Lubin et al. [49]). In this case separating $A_{\mathcal{R}}$ also leads to a straightforward parallelization of the maximum volume algorithms from Chapter 2.

The above method assumed that A remains of full row rank after dropping $A_{\mathcal{R}}$. Otherwise a slight modification can be used. Let $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$, where $\mathcal{R}_1 \subseteq \mathcal{R}$ is a maximum subset of columns that can be removed from A without loosing full row rank. Then the partitioning

$$A = \begin{bmatrix} A_{\mathcal{B}} & A_{\mathcal{N}} & A_{\mathcal{R}_1} \end{bmatrix}$$

is used and M is built of columns in $\mathcal{R}_1$. The important observation is that columns in $\mathcal{R}_0 \subseteq \mathcal{B}$ do not cause nonzero entries in $A_{\mathcal{B}}^{-1}A_{\mathcal{N}}$. For, if $(A_{\mathcal{B}}^{-1}A_{\mathcal{N}})_{pq} \neq 0$ for some $\mathcal{B}_p \in \mathcal{R}_0$, then index $\mathcal{N}_q$ could replace index $\mathcal{B}_p$ in the basis and $\mathcal{B}_p$ could be added to $\mathcal{R}_1$. This is in contradiction to $\mathcal{R}_1 \subseteq \mathcal{R}$ being a maximum subset. Hence the rows of the tableau matrix corresponding to columns in $\mathcal{R}_0$ are all zero.

The observation can be exploited further. After dropping $A_{\mathcal{R}_1}$ from A, the columns of $A_{\mathcal{N}}$ are linearly dependent on $A_{\mathcal{B}\setminus\mathcal{R}_0}$ for any basic-nonbasic partition. Therefore $A_{\mathcal{R}_0}$ can be replaced in the basis matrix by any "auxiliary" columns that keep the matrix nonsingular, without affecting the search for a maximum volume basis. The obvious choice is to replace $A_{\mathcal{R}_0}$ by unit columns to increase sparsity in the LU factorization. After a basis has been found, the actual $A_{\mathcal{B}}$ is factorized once in preparation for applying the preconditioner.

3.4 Error Control for Iterative Methods

A frequent question in practice is to what accuracy a linear system needs to be solved by an iterative method. For least squares problems a meaningful termination criterion can be derived from the transformed residual if basis preconditioning is used.

Assume that an iterative method computes an approximate solution to (3.5) that satisfies $C\boldsymbol{u} = B^{-1}\boldsymbol{r} + \boldsymbol{\delta}$, and that a solution to the KKT system is recovered from

$$\boldsymbol{y} = B^{-T}\boldsymbol{u}, \quad \boldsymbol{x}_{\mathcal{N}} = \boldsymbol{c}_{\mathcal{N}} - N^T\boldsymbol{y}, \quad \boldsymbol{x}_{\mathcal{B}} = B^{-1}(\boldsymbol{b} - N\boldsymbol{x}_{\mathcal{N}}). \tag{3.8}$$

By definition of $\boldsymbol{x}$, a residual occurs only in the basic components of the first block equation in (3.2). That residual is exactly $\boldsymbol{\delta}$ because

$$\begin{aligned} \boldsymbol{x}_{\mathcal{B}} + B^T\boldsymbol{y} &= B^{-1}(\boldsymbol{b} - N\boldsymbol{x}_{\mathcal{N}}) + \boldsymbol{u} \\ &= B^{-1}\boldsymbol{b} - B^{-1}N(\boldsymbol{c}_{\mathcal{N}} - N^TB^{-T}\boldsymbol{u}) + \boldsymbol{u} \\ &= C\boldsymbol{u} + B^{-1}(\boldsymbol{b} - N\boldsymbol{c}_{\mathcal{N}}) \\ &= B^{-1}\boldsymbol{r} + \boldsymbol{\delta} + B^{-1}(\boldsymbol{b} - N\boldsymbol{c}_{\mathcal{N}}) \\ &= \boldsymbol{c}_{\mathcal{B}} + \boldsymbol{\delta}. \end{aligned}$$

For a least squares problem the relevant quantity for the accuracy of the solution is the relative error in the objective, $(\|\boldsymbol{x}\| - \|\boldsymbol{x}^*\|)/\|\boldsymbol{x}^*\|$, where $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ denotes the exact solution

to the KKT system. From

$$\begin{bmatrix} I & & B^T \\ & I & N^T \\ B & N & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{x}_{\mathcal{B}} - \boldsymbol{x}^*_{\mathcal{B}} \\ \boldsymbol{x}_{\mathcal{N}} - \boldsymbol{x}^*_{\mathcal{N}} \\ \boldsymbol{y} - \boldsymbol{y}^* \end{pmatrix} = \begin{pmatrix} \boldsymbol{\delta} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{pmatrix} \tag{3.9}$$

it follows that $\boldsymbol{x} - \boldsymbol{x}^*$ is the projection of $(\boldsymbol{\delta}, \boldsymbol{0})$ onto the null space of A and therefore $\|\boldsymbol{x} - \boldsymbol{x}^*\| \leq \|\boldsymbol{\delta}\|$. Now assume that $\|\boldsymbol{\delta}\| \leq \varepsilon \|\boldsymbol{x}\|$ for some $\varepsilon \in (0, 1)$. Using the expression obtained from the triangle inequality,

$$\|\boldsymbol{x}^*\| = \|\boldsymbol{x} - (\boldsymbol{x} - \boldsymbol{x}^*)\| \geq \|\boldsymbol{x}\| - \|\boldsymbol{x} - \boldsymbol{x}^*\| \geq \|\boldsymbol{x}\| - \|\boldsymbol{\delta}\|,$$

it follows that

$$\frac{\|\boldsymbol{x}\| - \|\boldsymbol{x}^*\|}{\|\boldsymbol{x}^*\|} \leq \frac{\|\boldsymbol{x} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}^*\|} \leq \frac{\|\boldsymbol{\delta}\|}{\|\boldsymbol{x}\| - \|\boldsymbol{\delta}\|} \leq \frac{\|\boldsymbol{\delta}\|}{1/\varepsilon \|\boldsymbol{\delta}\| - \|\boldsymbol{\delta}\|} = \frac{\varepsilon}{1 - \varepsilon}. \tag{3.10}$$

Hence the criterion $\|\boldsymbol{\delta}\| \leq \varepsilon \|\boldsymbol{x}\|$ controls the relative error in $\boldsymbol{x}$ and $\|\boldsymbol{x}\|$. Because $\varepsilon/(1-\varepsilon) \approx \varepsilon$ for $\varepsilon \ll 1$, $\|\boldsymbol{\delta}\|$ needs to be k orders of magnitude smaller than $\|\boldsymbol{x}\|$ to obtain k-digit accuracy in the least squares objective and the least squares residual. The criterion might seem unusual by involving the approximate solution to the KKT system rather than the right-hand side, but it is implementable since $\boldsymbol{x}$ is either available in the iterative method or can be computed at reasonable cost. Notice that the derivation of (3.10) required that the approximate solution $\boldsymbol{x}$ satisfied $A\boldsymbol{x} = \boldsymbol{b}$, which is why we had to choose $\boldsymbol{x}_{\mathcal{B}} = B^{-1}(\boldsymbol{b} - N\boldsymbol{x}_N)$ in (3.8) and not $\boldsymbol{x}_B = \boldsymbol{c}_{\mathcal{B}} - B^T\boldsymbol{y}$.

In practice some care must be taken to handle the case $\|\boldsymbol{x}^*\| \approx 0$ (or even $\|\boldsymbol{x}^*\| = 0$), in which the overdetermined system $A^T\boldsymbol{y} = \boldsymbol{c}$ is (close-to) consistent. Because the solution to a consistent system is $\boldsymbol{y} = B^{-T}\boldsymbol{c}_{\mathcal{B}}$ for any basis, the criterion $\left\|\boldsymbol{c}_{\mathcal{N}} - N^T B^{-T}\boldsymbol{c}_{\mathcal{B}}\right\|_\infty \leq \varepsilon_{\text{abs}}$ can be used to test for that case prior to starting the iterative solver.

3.5 Rank Deficient Matrices

When A has rank less than m, the least squares solution to $A^T\boldsymbol{y} = \boldsymbol{c}$ is no longer unique. A slight modification of the basis preconditioning approach allows to compute one specific solution: As for the rank revealing method from Section 2.4 we determine a maximum volume basis for $\begin{bmatrix} A & \beta I_m \end{bmatrix}$ for a small $\beta > 0$. The set of logical columns in the basis defines a partitioning of A into $\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$, where A_{11} is nonsingular and its dimension is the numerical rank r of A. Let $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$ and $\boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2)$ be partitioned accordingly. After removing the second block row from A, the remaining system defines a full-rank least squares problem with unknowns $\boldsymbol{y}_1$. Because A_{11} has maximum volume in its block row, a basis preconditioner for solving the reduced problem is readily available. A solution to the original problem is then obtained by setting $\boldsymbol{y}_2 = \boldsymbol{0}$.

The sketched procedure is, of course, well known and comes naturally with basis preconditioning. In some applications it is desired to obtain the least squares solution $\boldsymbol{y}^*$ of minimum norm. It will be shown that $\boldsymbol{y}_1^*$ differs from the above $\boldsymbol{y}_1$ by a rank $m - r$ correction and can be computed by the same approach.

Let U and V be defined by

$$A \underbrace{\begin{bmatrix} -A_{11}^{-1}A_{12} \\ I_{n-r} \end{bmatrix}}_{=:U} = 0, \quad A^T \underbrace{\begin{bmatrix} -A_{11}^{-T}A_{21}^T \\ I_{m-r} \end{bmatrix}}_{=:V} = 0,$$

so that their columns span the respective null spaces of A and A^T. It follows from [9, Section 5.6, Corollary 7] that $\boldsymbol{y}^*$ are the solution components to

$$\begin{bmatrix} A^T & U \\ V^T & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{y}^* \\ \boldsymbol{\lambda}^* \end{pmatrix} = \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{0} \end{pmatrix}.$$

Substituting the expressions for U and V yields

$$\begin{bmatrix} A_{11}^T & A_{21}^T & -A_{11}^{-1}A_{12} \\ A_{12}^T & A_{22}^T & I_{n-r} \\ -A_{21}A_{11}^{-1} & I_{m-r} & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{y}_1^* \\ \boldsymbol{y}_2^* \\ \boldsymbol{\lambda}^* \end{pmatrix} = \begin{pmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \\ \boldsymbol{0} \end{pmatrix}. \tag{3.11}$$

The lower right 2×2 block is nonsingular and pivoting on it reduces (3.11) to

$$\left(A_{11}^T - \begin{bmatrix} A_{21}^T & -A_{11}^{-1}A_{12} \end{bmatrix} \begin{bmatrix} 0 & I_{m-r} \\ I_{n-r} & -A_{22}^T \end{bmatrix} \begin{bmatrix} A_{12}^T \\ -A_{21}A_{11}^{-1} \end{bmatrix} \right) \boldsymbol{y}_1^* =$$
$$\boldsymbol{c}_1 - \begin{bmatrix} A_{21}^T & -A_{11}^{-1}A_{12} \end{bmatrix} \begin{bmatrix} 0 & I_{m-r} \\ I_{n-r} & -A_{22}^T \end{bmatrix} \begin{pmatrix} \boldsymbol{c}_2 \\ \boldsymbol{0} \end{pmatrix}.$$

Denoting the Schur complement matrix by S and evaluating the right-hand side yields

$$S\boldsymbol{y}_1^* = \boldsymbol{c}_1 + A_{11}^{-1}A_{12}\boldsymbol{c}_2.$$

After solving the Schur complement system, the remaining components of $\boldsymbol{y}^*$ are obtained from the last block equation in (3.11) as $\boldsymbol{y}_2^* = A_{21}A_{11}^{-1}\boldsymbol{y}_1^*$.

The linear system with S can be further decomposed by writing S in product form:

$$\begin{aligned} S &= A_{11}^T + A_{21}^T A_{21} A_{11}^{-1} + A_{11}^{-1} A_{12} A_{12}^T + A_{11}^{-1} A_{12} A_{22}^T A_{21} A_{11}^{-1} \\ &= A_{11}^T + A_{21}^T A_{21} A_{11}^{-1} + A_{11}^{-1} A_{12} A_{12}^T + A_{11}^{-1} A_{12} (A_{12}^T A_{11}^{-T} A_{21}^T) A_{21} A_{11}^{-1} \\ &= \left(A_{11}^T + A_{11}^{-1} A_{12} A_{12}^T \right) + \left(A_{21}^T A_{21} A_{11}^{-1} + A_{11}^{-1} A_{12} A_{12}^T A_{11}^{-T} A_{21}^T A_{21} A_{11}^{-1} \right) \\ &= A_{11}^{-1} \left(A_{11} A_{11}^T + A_{12} A_{12}^T \right) + A_{11}^{-1} \left(A_{11} A_{11}^T + A_{12} A_{12}^T \right) A_{11}^{-T} A_{21}^T A_{21} A_{11}^{-1} \\ &= A_{11}^{-1} \left(A_{11} A_{11}^T + A_{12} A_{12}^T \right) \left(I_r + A_{11}^{-T} A_{21}^T A_{21} A_{11}^{-1} \right), \end{aligned}$$

where the second line used that $A_{22} - A_{21}A_{11}^{-1}A_{12} = 0$ when A has rank r. (Here A is required to be truly rank deficient.) Computing $\boldsymbol{y}_1^*$ now proceeds in two stages,

$$(A_{11}A_{11}^T + A_{12}A_{12}^T)\boldsymbol{y}_1 = A_{11}\boldsymbol{c}_1 + A_{12}\boldsymbol{c}_2, \tag{3.12a}$$
$$(I_r + A_{11}^{-T}A_{21}^T A_{21}A_{11}^{-1})\boldsymbol{y}_1^* = \boldsymbol{y}_1. \tag{3.12b}$$

The first stage computes the specific least squares solution to $A^T\boldsymbol{y} = \boldsymbol{c}$ that corresponds to $\boldsymbol{y}_2 = \boldsymbol{0}$. The second stage applies a correction of rank at most $m - r$ to $\boldsymbol{y}_1$ to obtain the least squares solution of minimum norm.

The above form is computationally interesting because $m-r$ will often be small in practice, so that the second stage can be obtained inexpensively through the Sherman-Morrison-Woodbury formula. The total cost for computing the minimum norm solution is then comparable to solving an ordinary least squares problem. For sparse matrices this can be much faster than computing a singular value decomposition or a complete orthogonal factorization of A.

The above form also yields a bound on $\|\boldsymbol{y}\|$ in terms of $\|\boldsymbol{y}^*\|$ by the assumption that A_{11} has local maximum volume in its block column; for

$$\begin{aligned}\|\boldsymbol{y}\| = \|\boldsymbol{y}_1\| &\leq \left\|I_r + A_{11}^{-T} A_{21}^T A_{21} A_{11}^{-1}\right\| \|\boldsymbol{y}_1^*\| \\ &\leq (1 + \left\|A_{21} A_{11}^{-1}\right\|_F^2) \|\boldsymbol{y}_1^*\| \\ &\leq (1 + m(m-r)) \|\boldsymbol{y}_1^*\| \\ &\leq (1 + m(m-r)) \|\boldsymbol{y}^*\| \,.\end{aligned}$$

Combining this with the trivial inequality $\|\boldsymbol{y}^*\| \leq \|\boldsymbol{y}\|$, we obtain

$$\|\boldsymbol{y}^*\| \leq \|\boldsymbol{y}\| \leq (1 + m(m-r)) \|\boldsymbol{y}^*\| \,.$$

The bound seems to be rarely useful in practice, however, because $m(m-r)$ can be in the same order of magnitude as $\|\boldsymbol{y}^*\|$.

Addendum

The author discovered the product form of S while trying to write (3.11) into symmetric form (which turned out not to be possible); he then found that (3.12a) and (3.12b) are obtained directly from the Moore-Penrose inverse of a 2×2 partitioned matrix derived by Hung and Markham [39].

4 Analysis of an Inexact Interior Point Method

For the theoretical part of this work, the linear programming (LP) problem is stated in standard form of a primal-dual pair

$$\underset{\boldsymbol{x}}{\text{minimize}}\ \boldsymbol{c}^T\boldsymbol{x} \quad \text{subject to } A\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \geq \boldsymbol{0}, \tag{4.1a}$$

$$\underset{\boldsymbol{y},\boldsymbol{z}}{\text{maximize}}\ \boldsymbol{b}^T\boldsymbol{y} \quad \text{subject to } A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c},\ \boldsymbol{z} \geq \boldsymbol{0}, \tag{4.1b}$$

where $A \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{c} \in \mathbb{R}^n$ are given. It can be assumed without loss of generality that A has full row rank; otherwise, if $\operatorname{rank}(A) < \operatorname{rank}(\begin{bmatrix} A & \boldsymbol{b}\end{bmatrix})$, then the equations $A\boldsymbol{x} = \boldsymbol{b}$ are inconsistent, and if $\operatorname{rank}(A) = \operatorname{rank}(\begin{bmatrix} A & \boldsymbol{b}\end{bmatrix})$, then $m - \operatorname{rank}(A)$ constraints are redundant and can be removed. Checking for consistency and identifying redundant constraints can be done by Gaussian elimination in $O(m^2n)$ time.

The LP problem (4.1) is said to be "primal feasible" if there exists an $\boldsymbol{x}$ satisfying $A\boldsymbol{x} = \boldsymbol{b}$, $\boldsymbol{x} \geq \boldsymbol{0}$, and "dual feasible" if there exists $(\boldsymbol{y}, \boldsymbol{z})$ such that $A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c}$, $\boldsymbol{z} \geq \boldsymbol{0}$. If the problem is primal and dual feasible, then it has at least one optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ and each such solution satisfies $\boldsymbol{c}^T\boldsymbol{x}^* = \boldsymbol{b}^T\boldsymbol{y}^*$. For this and other results about linear programming the reader is referred to [59, Chapter 13].

The simplex method and the interior point method (IPM) are the two commonly used tools for solving LP problems in practice. While the simplex method identifies an optimal vertex by moving along edges of the feasible region, the IPM generates iterates that lie in the relative interior of the feasible region (assuming that it is non-empty) and that approach the solution set in the limit. The theoretical development of IPMs dates back to the work of Karmarkar [40] and has led to a range of algorithms that can be classified as "interior point methods" in the broad sense; see Section "Background" in Wright [80, Chapter 2, pp. 40–45] for a historical overview. The common feature of the evolved algorithms is that their iteration count is bounded polynomially in n. The best known complexity result for computing an ε-accurate solution (defined below) is $O(\sqrt{n}\ln(1/\varepsilon))$ and is achieved, for example, by the short-step path-following methods of Kojima et al. [45] and of Monteiro and Adler [56]. Path-following methods keep the iterates in a certain neighbourhood of a central trajectory, which connects the analytic center of the feasible region to the analytic center of the optimal face; see Gonzaga [27] for an extensive discussion of such algorithms. Practical implementations of IPMs are more closely related to long-step path-following methods, such as the algorithm of Kojima et al. [46], which use a wider neighbourhood than the short-step methods. Although their best known complexity bound is worse, namely $O(n\ln(1/\varepsilon))$ in [46], they actually require fewer iterations than short-step methods in practice.

In this chapter the convergence of an interior point method that works with inexactly computed step directions is analysed. A "feasible" and an "infeasible" method are formulated and are shown to retain the complexity bounds of their exact counterparts. The conditions on the residuals in the linear systems are designed to be utilizable in practice as stopping criterion for an iterative solver. The analysis is carried out in the potential reduction framework, which leaves more flexibility for inexact directions than a path-following method. The work was originally presented by the author in [69].

4.1 Background

Before stating the inexact interior point algorithms in the next section, a brief summary of the main concepts of IPMs is given. A primal-dual IPM applied to (4.1) generates a sequence of iterates $\left(\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{z}^k\right)$ with $\left(\boldsymbol{x}^k, \boldsymbol{z}^k\right) > \boldsymbol{0}$. We say that an iterate is an ε-approximate solution if

$(\boldsymbol{x}^k)^T\boldsymbol{z}^k \le \varepsilon$ and

$$\left\|\begin{pmatrix} A\boldsymbol{x}^k - \boldsymbol{b} \\ A^T\boldsymbol{y}^k + \boldsymbol{z}^k - \boldsymbol{c}\end{pmatrix}\right\| \le \frac{\varepsilon}{(\boldsymbol{x}^0)^T\boldsymbol{z}^0}\left\|\begin{pmatrix} A\boldsymbol{x}^0 - \boldsymbol{b} \\ A^T\boldsymbol{y}^0 + \boldsymbol{z}^0 - \boldsymbol{c}\end{pmatrix}\right\|. \tag{4.2}$$

Measuring the primal and dual residuals as a combined vector is justified because the considered methods will decrease each component of $A\boldsymbol{x}^k - \boldsymbol{b}$ and $A^T\boldsymbol{y}^k + \boldsymbol{z}^k - \boldsymbol{c}$ at the same rate.

Interior point methods compute the next iterate by taking a step along the Newton direction for the nonlinear system

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{4.3a}$$
$$A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c}, \tag{4.3b}$$
$$x_i z_i = \mu \quad \text{for } 1 \le i \le n, \tag{4.3c}$$

for some $\mu > 0$. These equations along with $(\boldsymbol{x}, \boldsymbol{z}) \ge \boldsymbol{0}$ express the optimality conditions for (4.1) with the complementarity requirement $x_i z_i = 0$ perturbed by μ. The Newton direction at the iterate $(\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{z}^k)$ is the solution to the linear system

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z^k & 0 & X^k \end{bmatrix}\begin{pmatrix}\Delta\boldsymbol{x}^* \\ \Delta\boldsymbol{y}^* \\ \Delta\boldsymbol{z}^*\end{pmatrix} = \begin{pmatrix} \boldsymbol{b} - A\boldsymbol{x}^k \\ \boldsymbol{c} - A^T\boldsymbol{y}^k - \boldsymbol{z}^k \\ -X^kZ^k\boldsymbol{e} + \mu\boldsymbol{e}\end{pmatrix},$$

where the convention is used that for any lower-case letter representing a vector in $\mathbb{R}^n$ (such as $\boldsymbol{x}$, $\boldsymbol{z}^k$), the corresponding upper-case letter (such as X, Z^k) denotes the diagonal matrix of dimension n with the vector components on the diagonal. Defining $D = (X^k)^{1/2}(Z^k)^{-1/2}$, $W = (X^kZ^k)^{1/2}$ and $\boldsymbol{w} = W\boldsymbol{e}$, the Newton system can be written in the scaled quantities $\Delta\boldsymbol{u}^* = D^{-1}\Delta\boldsymbol{x}^*$ and $\Delta\boldsymbol{v}^* = D\Delta\boldsymbol{z}^*$ as

$$\begin{bmatrix} AD & 0 & 0 \\ 0 & DA^T & I \\ I & 0 & I \end{bmatrix}\begin{pmatrix}\Delta\boldsymbol{u}^* \\ \Delta\boldsymbol{y}^* \\ \Delta\boldsymbol{v}^*\end{pmatrix} = \begin{pmatrix} \boldsymbol{b} - A\boldsymbol{x}^k \\ D(\boldsymbol{c} - A^T\boldsymbol{y}^k - \boldsymbol{z}^k) \\ -\boldsymbol{w} + \mu W^{-1}\boldsymbol{e}\end{pmatrix} =: \begin{pmatrix}\boldsymbol{p} \\ \boldsymbol{q} \\ \boldsymbol{r}\end{pmatrix}. \tag{4.4}$$

Potential reduction methods are a subclass of IPMs that choose the step size at each iteration to reduce a "potential function" by at least a certain value. The analysis presented below uses the Tanabe-Todd-Ye potential function [75, 78]

$$\phi(\boldsymbol{x}, \boldsymbol{z}) = (n + \sqrt{n})\ln(\boldsymbol{x}^T\boldsymbol{z}) - \sum_{i=1}^{n}\ln(x_i z_i) - n\ln(n).$$

By means of the following lemma, decreasing $\phi(\boldsymbol{x}^k, \boldsymbol{z}^k)$ toward $-\infty$ forces $\boldsymbol{x}^k$ and $\boldsymbol{z}^k$ to become complementary.

Lemma 4.1. *$\phi(\boldsymbol{x}, \boldsymbol{z}) \ge \sqrt{n}\ln(\boldsymbol{x}^T\boldsymbol{z})$ for all $(\boldsymbol{x}, \boldsymbol{z}) > \boldsymbol{0}$.*

Proof. Given by Kojima et al. [47, Section 1]. □

4.2 Two Inexact Potential Reduction Methods

Instead of $(\Delta\boldsymbol{u}^*, \Delta\boldsymbol{y}^*, \Delta\boldsymbol{v}^*)$ the inexact methods work with step directions of the form

$$\begin{bmatrix} AD & 0 & 0 \\ 0 & DA^T & I \\ I & 0 & I \end{bmatrix}\begin{pmatrix}\Delta\boldsymbol{u} \\ \Delta\boldsymbol{y} \\ \Delta\boldsymbol{v}\end{pmatrix} = \begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{q} \\ \boldsymbol{r} + \boldsymbol{\xi}\end{pmatrix}, \tag{4.5}$$

where a residual $\boldsymbol{\xi}$ occurs in the scaled complementarity equations. The primal and dual feasibility equations must be satisfied exactly. Let $\kappa \in [0, 1)$ be a parameter that is chosen independently of the problem dimension. The algorithms below make use of the following conditions on the residual:

$$-\boldsymbol{r}^T\boldsymbol{\xi} \leq \kappa \|\boldsymbol{r}\|^2, \tag{4.6a}$$

$$\|\boldsymbol{\xi}\| \leq \kappa \min(\|\Delta\boldsymbol{u}\|, \|\Delta\boldsymbol{v}\|), \tag{4.6b}$$

$$-\boldsymbol{w}^T\boldsymbol{\xi} \leq \kappa n/(n+\sqrt{n}) \|\boldsymbol{w}\|^2. \tag{4.6c}$$

Algorithm 4 is the inexact version of the method analysed by Kojima et al. [47] for the linear complementarity problem and formulated by Wright [80, Chapter 4] for the LP problem. The method is called "feasible" because all iterates belong to the strictly feasible set

$$\mathcal{F}^o = \left\{(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \mid A\boldsymbol{x} = \boldsymbol{b}, A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c}, (\boldsymbol{x}, \boldsymbol{z}) > \boldsymbol{0}\right\},$$

which is assumed to be non-empty. Algorithm 4 does not require condition (4.6c).

Algorithm 4 Feasible Potential Reduction Method
(adapted from Algorithm PR in Wright [80, Chapter 4, p. 67])

Input: $(\boldsymbol{x}^0, \boldsymbol{y}^0, \boldsymbol{z}^0) \in \mathcal{F}^o$, $\varepsilon > 0$.

1: Set $\delta = 0.15(1-\kappa)^4$ and $k = 0$.
2: If $(\boldsymbol{x}^k)^T\boldsymbol{z}^k \leq \varepsilon$, then stop.
3: Let $\mu = (\boldsymbol{x}^k)^T\boldsymbol{z}^k/(n+\sqrt{n})$. Compute a solution to (4.5) with residual $\boldsymbol{\xi}$ that satisfies (4.6a)–(4.6b). Set $\Delta\boldsymbol{x} = D\Delta\boldsymbol{u}$ and $\Delta\boldsymbol{z} = D^{-1}\Delta\boldsymbol{v}$.
4: Find a step size α^k such that

$$\phi(\boldsymbol{x}^k + \alpha^k\Delta\boldsymbol{x}, \boldsymbol{z}^k + \alpha^k\Delta\boldsymbol{z}) \leq \phi(\boldsymbol{x}^k, \boldsymbol{z}^k) - \delta. \tag{4.7}$$

5: Set $(\boldsymbol{x}^{k+1}, \boldsymbol{y}^{k+1}, \boldsymbol{z}^{k+1}) = (\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{z}^k) + \alpha^k(\Delta\boldsymbol{x}, \Delta\boldsymbol{y}, \Delta\boldsymbol{z})$, $k = k+1$ and go to 2.

The following theorem, which is proved in Section 4.3, states that Algorithm 4 retains the complexity bound of its exact counterpart (i. e. for $\kappa = 0$).

Theorem 4.2. *Let $(\boldsymbol{x}^0, \boldsymbol{y}^0, \boldsymbol{z}^0) \in \mathcal{F}^o$ and $L \geq 0$ be such that $\phi(\boldsymbol{x}^0, \boldsymbol{z}^0) = O(\sqrt{n}L)$ and $\ln(1/\varepsilon) = O(L)$. Then Algorithm 4 terminates in $O(\sqrt{n}L)$ iterations.*

Algorithm 5 is an "infeasible" inexact potential reduction method, as its iterates do not, in general, belong to $\mathcal{F}^o$. It extends Algorithm 1 from Mizuno et al. [55] to work with inexact directions. Given positive parameters ρ and ε, it finds an ε-accurate approximation to a solution $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ to (4.1), if it exists, such that $\|(\boldsymbol{x}^*, \boldsymbol{z}^*)\|_\infty \leq \rho$.

The following theorem, which is proved in Section 4.4, states that Algorithm 5 retains the complexity bound of its exact counterpart.

Theorem 4.3. *Let $L \geq \ln(n)$ be such that $\ln(\rho) = O(L)$ and $\ln(1/\varepsilon) = O(L)$. Then Algorithm 5 terminates in $O(\sqrt{n}(n+\sqrt{n})^2L)$ iterations. If the algorithm stops in step 2, then the iterate is an ε-approximate solution; otherwise it stops in step 4 proving that there is no optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ to (4.1) with $\|(\boldsymbol{x}^*, \boldsymbol{z}^*)\|_\infty \leq \rho$.*

When Algorithm 5 stops in step 4 and ρ is sufficiently large, then the LP problem must be primal and/or dual infeasible. Todd [77] studied criteria to decide from the IPM iterates which of (4.1a) and (4.1b) are infeasible. He showed exemplarily for the path-following method of Kojima et al. [44] how the iterates generate a certificate of primal or dual infeasibility in

Algorithm 5 Infeasible Potential Reduction Method
(adapted from Algorithm 1 in Mizuno et al. [55])

Input: $\rho > 0$, $\varepsilon > 0$.

1: Set $\left(\boldsymbol{x}^0, \boldsymbol{y}^0, \boldsymbol{z}^0\right) = \rho\left(\boldsymbol{e}, \boldsymbol{0}, \boldsymbol{e}\right)$, $\delta = (1-\kappa)^4/(800(n+\sqrt{n})^2)$ and $k = 0$.

2: If $(\boldsymbol{x}^k)^T\boldsymbol{z}^k \leq \varepsilon$, then stop.

3: Let $\mu = (\boldsymbol{x}^k)^T\boldsymbol{z}^k/(n+\sqrt{n})$. Compute a solution to (4.5) with residual $\boldsymbol{\xi}$ that satisfies (4.6a)–(4.6c). Set $\Delta\boldsymbol{x} = D\Delta\boldsymbol{u}$ and $\Delta\boldsymbol{z} = D^{-1}\Delta\boldsymbol{v}$.

4: Find a step size α^k such that

$$\phi(\boldsymbol{x}^k + \alpha^k\Delta\boldsymbol{x}, \boldsymbol{z}^k + \alpha^k\Delta\boldsymbol{z}) \leq \phi(\boldsymbol{x}^k, \boldsymbol{z}^k) - \delta, \tag{4.8a}$$

$$(\boldsymbol{x}^k + \alpha^k\Delta\boldsymbol{x})^T(\boldsymbol{z}^k + \alpha^k\Delta\boldsymbol{z}) \geq (1-\alpha^k)(\boldsymbol{x}^k)^T\boldsymbol{z}^k. \tag{4.8b}$$

If no such step size exists, then stop.

5: Set $\left(\boldsymbol{x}^{k+1}, \boldsymbol{y}^{k+1}, \boldsymbol{z}^{k+1}\right) = \left(\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{z}^k\right) + \alpha^k\left(\Delta\boldsymbol{x}, \Delta\boldsymbol{y}, \Delta\boldsymbol{z}\right)$, $k = k+1$ and go to 2.

the limit (see Section 5.2 in [77]). Todd's theory is based on a generic IPM formulation which includes Algorithm 5 with $\kappa = 0$. If the results also hold for the inexact method (i. e. $\kappa > 0$) has not been investigated for the present work.

The following lemma is needed for the convergence analysis of the inexact potential reduction methods. It uses the particular form of the scaled Newton system to prove that condition (4.6b) bounds the relative error in the inexact solution.

Lemma 4.4. *Given solutions to* (4.4) *and* (4.5)*, suppose that* (4.6b) *holds for* $\kappa \in [0, 1)$*. Then*

$$\|\Delta\boldsymbol{u} - \Delta\boldsymbol{u}^*\| \leq \kappa/(1-\kappa)\,\|\Delta\boldsymbol{u}^*\|\,,$$
$$\|\Delta\boldsymbol{v} - \Delta\boldsymbol{v}^*\| \leq \kappa/(1-\kappa)\,\|\Delta\boldsymbol{v}^*\|\,.$$

Proof. Let $P = DA^T(AD^2A^T)^{-1}AD$. The solution to (4.4) is

$$\Delta\boldsymbol{u}^* = DA^T(AD^2A^T)^{-1}\boldsymbol{p} - (I-P)\boldsymbol{q} + (I-P)\boldsymbol{r},$$
$$\Delta\boldsymbol{y}^* = (AD^2A^T)^{-1}(\boldsymbol{p} + AD\boldsymbol{q} - AD\boldsymbol{r}),$$
$$\Delta\boldsymbol{v}^* = -DA^T(AD^2A^T)^{-1}\boldsymbol{p} + (I-P)\boldsymbol{q} + P\boldsymbol{r}.$$

It follows that

$$\Delta\boldsymbol{u} - \Delta\boldsymbol{u}^* = (I-P)\boldsymbol{\xi},$$
$$\Delta\boldsymbol{v} - \Delta\boldsymbol{v}^* = P\boldsymbol{\xi}.$$

Because P and $I-P$ are projection operators, $\|P\| \leq 1$ and $\|I-P\| \leq 1$. Therefore the absolute errors are bounded by the norm of the residual,

$$\|\Delta\boldsymbol{u} - \Delta\boldsymbol{u}^*\| \leq \|\boldsymbol{\xi}\|\,,$$
$$\|\Delta\boldsymbol{v} - \Delta\boldsymbol{v}^*\| \leq \|\boldsymbol{\xi}\|\,.$$

On the other hand, it follows from the triangle inequality and (4.6b) that

$$\|\Delta\boldsymbol{u}^*\| = \|\Delta\boldsymbol{u} - (I-P)\boldsymbol{\xi}\| \geq \|\Delta\boldsymbol{u}\| - \|\boldsymbol{\xi}\| \geq (1-\kappa)\,\|\Delta\boldsymbol{u}\|\,, \tag{4.10a}$$
$$\|\Delta\boldsymbol{v}^*\| = \|\Delta\boldsymbol{v} - P\boldsymbol{\xi}\| \geq \|\Delta\boldsymbol{v}\| - \|\boldsymbol{\xi}\| \geq (1-\kappa)\,\|\Delta\boldsymbol{v}\|\,. \tag{4.10b}$$

Combining both inequalities and (4.6b) gives

$$\|\Delta\boldsymbol{u} - \Delta\boldsymbol{u}^*\| \leq \|\boldsymbol{\xi}\| \leq \kappa\,\|\Delta\boldsymbol{u}\| \leq \kappa/(1-\kappa)\,\|\Delta\boldsymbol{u}^*\|\,,$$
$$\|\Delta\boldsymbol{v} - \Delta\boldsymbol{v}^*\| \leq \|\boldsymbol{\xi}\| \leq \kappa\,\|\Delta\boldsymbol{v}\| \leq \kappa/(1-\kappa)\,\|\Delta\boldsymbol{v}^*\|$$

as claimed. □

4.3 Proof of the Complexity Bound for the Feasible Method

Analysing Algorithm 4 requires two technical results from Mizuno et al. [55], which are stated in the following two lemmas.

Lemma 4.5 ([55, Lemma 2])**.** *Let $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^n_{>0}$, $\Delta\boldsymbol{x}, \Delta\boldsymbol{z} \in \mathbb{R}^n$, $\alpha > 0$ and $\tau \in (0,1)$ be such that $\left\|\alpha X^{-1}\Delta\boldsymbol{x}\right\|_\infty \leq \tau$ and $\left\|\alpha Z^{-1}\Delta\boldsymbol{z}\right\|_\infty \leq \tau$. Then*

$$\phi(\boldsymbol{x} + \alpha\Delta\boldsymbol{x}, \boldsymbol{z} + \alpha\Delta\boldsymbol{z}) \leq \phi(\boldsymbol{x}, \boldsymbol{z}) + g_1\alpha + g_2\alpha^2$$

with coefficients

$$g_1 = \left(\frac{n+\sqrt{n}}{\boldsymbol{x}^T\boldsymbol{z}}\boldsymbol{e} - (XZ)^{-1}\boldsymbol{e}\right)^T (Z\Delta\boldsymbol{x} + X\Delta\boldsymbol{z}),$$
$$g_2 = (n+\sqrt{n})\frac{\Delta\boldsymbol{x}^T\Delta\boldsymbol{z}}{\boldsymbol{x}^T\boldsymbol{z}} + \frac{\left\|X^{-1}\Delta\boldsymbol{x}\right\|^2 + \left\|Z^{-1}\Delta\boldsymbol{z}\right\|^2}{2(1-\tau)}.$$

Lemma 4.6 ([55, Lemma 3])**.** *Let $\boldsymbol{w} \in \mathbb{R}^n_{>0}$ and $w_{min} = \min_i w_i$. Then*

$$\left\|W^{-1}\boldsymbol{e} - \frac{n+\sqrt{n}}{\boldsymbol{w}^T\boldsymbol{w}}\boldsymbol{w}\right\| \geq \frac{\sqrt{3}}{2w_{min}}.$$

In the sequel let $\boldsymbol{w} = (X^kZ^k)^{1/2}\boldsymbol{e}$ as defined earlier and $w_{min} = \min_i w_i$. Applying Lemma 4.6 to the vector $\boldsymbol{r}$ defined in (4.4) for $\mu = (\boldsymbol{x}^k)^T\boldsymbol{z}^k/(n+\sqrt{n})$ yields

$$\|\boldsymbol{r}\| = \left\|-\boldsymbol{w} + \mu W^{-1}\boldsymbol{e}\right\| = \mu\left\|-\frac{1}{\mu}\boldsymbol{w} + W^{-1}\boldsymbol{e}\right\| \geq \mu\frac{\sqrt{3}}{2w_{min}}. \tag{4.11}$$

Lemma 4.7. *In iteration k of Algorithm 4 the potential reduction* (4.7) *is achieved by*

$$\alpha = \frac{w_{min}}{2\|\boldsymbol{r}\|}(1-\kappa)^3.$$

Proof. (adapted from Wright [80, Chapter 4]) From the first two block equations in (4.4) and $\boldsymbol{p} = \boldsymbol{0}$, $\boldsymbol{q} = \boldsymbol{0}$ we have that

$$(\Delta\boldsymbol{u}^*)^T\Delta\boldsymbol{v}^* = -(\Delta\boldsymbol{u}^*)^TDA^T\Delta\boldsymbol{y}^* = -(AD\Delta\boldsymbol{u}^*)^T\Delta\boldsymbol{y}^* = 0.$$

Therefore $\|\Delta\boldsymbol{u}^*\|^2 + \|\Delta\boldsymbol{v}^*\|^2 = \|\boldsymbol{r}\|^2$. Combining this with (4.10) and the definition of α yields

$$\left\|\alpha(X^k)^{-1}\Delta\boldsymbol{x}\right\|_\infty \leq \alpha\left\|W^{-1}\right\|\|\Delta\boldsymbol{u}\| \leq \frac{\alpha}{w_{min}}\frac{\|\Delta\boldsymbol{u}^*\|}{1-\kappa} \leq \frac{\alpha}{w_{min}}\frac{\|\boldsymbol{r}\|}{1-\kappa} \leq \frac{1}{2},$$
$$\left\|\alpha(Z^k)^{-1}\Delta\boldsymbol{z}\right\|_\infty \leq \alpha\left\|W^{-1}\right\|\|\Delta\boldsymbol{v}\| \leq \frac{\alpha}{w_{min}}\frac{\|\Delta\boldsymbol{v}^*\|}{1-\kappa} \leq \frac{\alpha}{w_{min}}\frac{\|\boldsymbol{r}\|}{1-\kappa} \leq \frac{1}{2}.$$

Hence $\tau = 1/2$ satisfies the assumptions of Lemma 4.5, so that

$$\Delta\phi := \phi(\boldsymbol{x}^k + \alpha\Delta\boldsymbol{x}, \boldsymbol{z}^k + \alpha\Delta\boldsymbol{z}) - \phi(\boldsymbol{x}^k, \boldsymbol{z}^k) \leq g_1\alpha + g_2\alpha^2 \tag{4.12}$$

with coefficients

$$g_1 = \left(\frac{n+\sqrt{n}}{\boldsymbol{w}^T\boldsymbol{w}}\boldsymbol{e} - W^{-2}\boldsymbol{e}\right)^T W(\Delta\boldsymbol{u} + \Delta\boldsymbol{v}),$$
$$g_2 = \left\|W^{-1}\Delta\boldsymbol{u}\right\|^2 + \left\|W^{-1}\Delta\boldsymbol{v}\right\|^2.$$

(The first term of g_2 from Lemma 4.5 vanishes because $\Delta \boldsymbol{x}^T \Delta \boldsymbol{z} = 0$ by the same argument as above.)

To show that ϕ is sufficiently reduced along the direction $(\Delta \boldsymbol{x}, \Delta \boldsymbol{z})$ we need to show that g_1 is negative and bounded away from zero, while g_2 is bounded. From the definition of $\boldsymbol{r}$ and condition (4.6a) we have

$$g_1 = \left(\frac{n + \sqrt{n}}{\boldsymbol{w}^T \boldsymbol{w}} \boldsymbol{w} - W^{-1} \boldsymbol{e} \right)^T (\Delta \boldsymbol{u} + \Delta \boldsymbol{v}) = -\frac{n + \sqrt{n}}{\boldsymbol{w}^T \boldsymbol{w}} \boldsymbol{r}^T (\boldsymbol{r} + \boldsymbol{\xi}) \tag{4.13}$$

$$\leq -(1 - \kappa) \frac{n + \sqrt{n}}{\boldsymbol{w}^T \boldsymbol{w}} \left\| \boldsymbol{r} \right\|^2. \tag{4.14}$$

For the second order term it follows from (4.10) that

$$g_2 = \left\| W^{-1} \Delta \boldsymbol{u} \right\|^2 + \left\| W^{-1} \Delta \boldsymbol{v} \right\|^2 \leq \frac{1}{w_{min}^2} \left(\left\| \Delta \boldsymbol{u} \right\|^2 + \left\| \Delta \boldsymbol{v} \right\|^2 \right)$$

$$\leq \frac{\left\| \Delta \boldsymbol{u}^* \right\|^2 + \left\| \Delta \boldsymbol{v}^* \right\|^2}{w_{min}^2 (1 - \kappa)^2} = \frac{\left\| \boldsymbol{r} \right\|^2}{w_{min}^2 (1 - \kappa)^2}.$$

Inserting the bounds on g_1 and g_2 into (4.12) and using the definition of α gives

$$\Delta \phi \leq -(1 - \kappa) \frac{n + \sqrt{n}}{\boldsymbol{w}^T \boldsymbol{w}} \left\| \boldsymbol{r} \right\|^2 \alpha + \frac{\left\| \boldsymbol{r} \right\|^2}{w_{min}^2 (1 - \kappa)^2} \alpha^2$$

$$= -(1 - \kappa)^4 \frac{n + \sqrt{n}}{\boldsymbol{w}^T \boldsymbol{w}} \frac{w_{min}}{2} \left\| \boldsymbol{r} \right\| + \frac{(1 - \kappa)^4}{4}.$$

Finally, using the bound on $\|\boldsymbol{r}\|$ from (4.11) yields

$$\Delta \phi \leq (1 - \kappa)^4 \left(-\frac{\sqrt{3}}{4} + \frac{1}{4} \right) \leq -0.15 (1 - \kappa)^4 = -\delta.$$

□

Proof of Theorem 4.2. Let $K \geq 0$ be the smallest index such that $\phi(\boldsymbol{x}^K, \boldsymbol{z}^K) \leq \sqrt{n} \ln(\varepsilon)$. From Lemma 4.1 we then have

$$\sqrt{n} \ln \left((\boldsymbol{x}^K)^T \boldsymbol{z}^K \right) \leq \phi(\boldsymbol{x}^K, \boldsymbol{z}^K) \leq \sqrt{n} \ln(\varepsilon),$$

so that Algorithm 4 terminates at the latest in iteration K. Because each iteration of the algorithm reduces $\phi(\boldsymbol{x}^k, \boldsymbol{z}^k)$ by at least δ, we have $K \leq \lceil (\phi(\boldsymbol{x}^0, \boldsymbol{z}^0) - \sqrt{n} \ln(\varepsilon)) / \delta \rceil$. It follows from $\phi(\boldsymbol{x}^0, \boldsymbol{z}^0) = O(\sqrt{n} L)$ and $\ln(1/\varepsilon) = O(L)$ that $K = O(\sqrt{n} L)$ as claimed. □

4.4 Proof of the Complexity Bound for the Infeasible Method

The analysis of Algorithm 5 is based on the work of Mizuno et al. [55]. Let the sequence $\{\theta^k\}$ be defined by

$$\theta^0 = 1, \quad \theta^{k+1} = (1 - \alpha^k) \theta^k \quad \text{for } k \geq 0.$$

Because of the form of the step directions we have

$$\left(A \boldsymbol{x}^k - \boldsymbol{b}, A^T \boldsymbol{y}^k + \boldsymbol{z}^k - \boldsymbol{c} \right) = \theta^k \left(A \boldsymbol{x}^0 - \boldsymbol{b}, A^T \boldsymbol{y}^0 + \boldsymbol{z}^0 - \boldsymbol{c} \right), \tag{4.15}$$

and because of the step size restriction (4.8b) we have

$$(\boldsymbol{x}^k)^T \boldsymbol{z}^k \geq \theta^k (\boldsymbol{x}^0)^T \boldsymbol{z}^0. \tag{4.16}$$

The following lemma is obtained by setting $\gamma_0 = 1$ and $\gamma_1 = 1$ in [55, Lemma 4].

Lemma 4.8. *Let $\rho > 0$ and suppose that*

$$\begin{aligned}\left(\boldsymbol{x}^0, \boldsymbol{y}^0, \boldsymbol{z}^0\right) &= \rho\left(\boldsymbol{e}, \boldsymbol{0}, \boldsymbol{e}\right), \\ \left(A\boldsymbol{x}^k - \boldsymbol{b}, A^T\boldsymbol{y}^k + \boldsymbol{z}^k - \boldsymbol{c}\right) &= \theta^k\left(A\boldsymbol{x}^0 - \boldsymbol{b}, A^T\boldsymbol{y}^0 + \boldsymbol{z}^0 - \boldsymbol{c}\right), \\ (\boldsymbol{x}^k)^T\boldsymbol{z}^k &\geq \theta^k(\boldsymbol{x}^0)^T\boldsymbol{z}^0. \end{aligned} \tag{4.17}$$

If there exists a solution $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ to (4.1) with $\|(\boldsymbol{x}^*, \boldsymbol{z}^*)\|_\infty \leq \rho$, then the solution to (4.4) at $\left(\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{z}^k\right)$ satisfies

$$\max(\|\Delta\boldsymbol{u}^*\|, \|\Delta\boldsymbol{v}^*\|) \leq \frac{5(\boldsymbol{x}^k)^T\boldsymbol{z}^k}{w_{min}}.$$

We can now prove that Algorithm 5 finds the required step sizes when the LP problem has an optimal solution.

Lemma 4.9. *If there exists an optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ to (4.1) with $\|(\boldsymbol{x}^*, \boldsymbol{z}^*)\|_\infty \leq \rho$, then (4.8a) and (4.8b) hold for*

$$\alpha = \frac{(1-\kappa)^3 w_{min}^2}{200(n+\sqrt{n})(\boldsymbol{x}^k)^T\boldsymbol{z}^k}.$$

Proof. It is easily seen that by (4.8b) and the definition of $\left(\boldsymbol{x}^0, \boldsymbol{z}^0\right)$ the assumptions of Lemma 4.8 are satisfied. Combining the lemma with (4.10) yields

$$\max(\|\Delta\boldsymbol{u}\|, \|\Delta\boldsymbol{v}\|) \leq \frac{5(\boldsymbol{x}^k)^T\boldsymbol{z}^k}{(1-\kappa)w_{min}}. \tag{4.18}$$

It follows that

$$\begin{aligned}\left\|\alpha(X^k)^{-1}\Delta\boldsymbol{x}\right\| &\leq \alpha\left\|W^{-1}\right\|\|\Delta\boldsymbol{u}\| \leq \alpha\frac{5(\boldsymbol{x}^k)^T\boldsymbol{z}^k}{(1-\kappa)w_{min}^2} = \frac{(1-\kappa)^2}{40(n+\sqrt{n})} \leq \frac{1}{40}, \\ \left\|\alpha(Z^k)^{-1}\Delta\boldsymbol{z}\right\| &\leq \alpha\left\|W^{-1}\right\|\|\Delta\boldsymbol{v}\| \leq \alpha\frac{5(\boldsymbol{x}^k)^T\boldsymbol{z}^k}{(1-\kappa)w_{min}^2} = \frac{(1-\kappa)^2}{40(n+\sqrt{n})} \leq \frac{1}{40}.\end{aligned}$$

Hence $\tau = 1/40$ satisfies the assumptions of Lemma 4.5, so that

$$\Delta\phi := \phi(\boldsymbol{x}^k + \alpha\Delta\boldsymbol{x}, \boldsymbol{z}^k + \alpha\Delta\boldsymbol{z}) - \phi(\boldsymbol{x}^k, \boldsymbol{z}^k) \leq g_1\alpha + g_2\alpha^2$$

with coefficients

$$\begin{aligned}g_1 &= \left(\frac{n+\sqrt{n}}{\boldsymbol{w}^T\boldsymbol{w}}\boldsymbol{e} - W^{-2}\boldsymbol{e}\right)^T W(\Delta\boldsymbol{u} + \Delta\boldsymbol{v}), \\ g_2 &= \left((n+\sqrt{n})\frac{\Delta\boldsymbol{u}^T\Delta\boldsymbol{v}}{\boldsymbol{w}^T\boldsymbol{w}} + \frac{\left\|W^{-1}\Delta\boldsymbol{u}\right\|^2 + \left\|W^{-1}\Delta\boldsymbol{v}\right\|^2}{2(1-\tau)}\right).\end{aligned}$$

It will be shown that g_1 is negative and bounded away from zero, while g_2 is bounded. Combining (4.14), (4.11) and the definition of μ gives

$$g_1 \leq -(1-\kappa)\frac{1}{\mu}\|\boldsymbol{r}\|^2 \leq -(1-\kappa)\mu\frac{3}{4w_{min}^2} = -(1-\kappa)\frac{\boldsymbol{w}^T\boldsymbol{w}}{n+\sqrt{n}}\frac{3}{4w_{min}^2}.$$

Next, from (4.18) we have

$$|\Delta\boldsymbol{u}^T\Delta\boldsymbol{v}| \leq \|\Delta\boldsymbol{u}\|\|\Delta\boldsymbol{v}\| \leq \left(\frac{5\boldsymbol{w}^T\boldsymbol{w}}{(1-\kappa)w_{min}}\right)^2 \tag{4.19}$$

and therefore

$$(n+\sqrt{n})\frac{\Delta \boldsymbol{u}^T\Delta \boldsymbol{v}}{\boldsymbol{w}^T\boldsymbol{w}} \le \frac{n+\sqrt{n}}{\boldsymbol{w}^T\boldsymbol{w}}\left(\frac{5\boldsymbol{w}^T\boldsymbol{w}}{(1-\kappa)w_{min}}\right)^2 \le \frac{n+\sqrt{n}}{n}\left(\frac{5\boldsymbol{w}^T\boldsymbol{w}}{(1-\kappa)w_{min}^2}\right)^2, \tag{4.20}$$

where the last inequality was obtained by multiplying with $\boldsymbol{w}^T\boldsymbol{w}/(nw_{min}^2) \ge 1$. Furthermore, (4.18) also implies that

$$\frac{\left\|W^{-1}\Delta \boldsymbol{u}\right\|^2 + \left\|W^{-1}\Delta \boldsymbol{v}\right\|^2}{2(1-\tau)} \le \frac{1}{1-\tau}\left(\frac{5\boldsymbol{w}^T\boldsymbol{w}}{(1-\kappa)w_{min}^2}\right)^2. \tag{4.21}$$

Summing up (4.20) and (4.21) gives

$$g_2 \le \left(\frac{n+\sqrt{n}}{n} + \frac{1}{1-\tau}\right)\left(\frac{5\boldsymbol{w}^T\boldsymbol{w}}{(1-\kappa)w_{min}^2}\right)^2 \le 4\left(\frac{5\boldsymbol{w}^T\boldsymbol{w}}{(1-\kappa)w_{min}^2}\right)^2.$$

Inserting g_1, g_2 and the definition of α into the quadratic form yields

$$\begin{aligned}\Delta\phi &\le -(1-\kappa)\frac{\boldsymbol{w}^T\boldsymbol{w}}{n+\sqrt{n}}\frac{3}{4w_{min}^2}\alpha + 4\left(\frac{5\boldsymbol{w}^T\boldsymbol{w}}{(1-\kappa)w_{min}^2}\right)^2\alpha^2 \\ &= \frac{(1-\kappa)^4}{(n+\sqrt{n})^2}\left(-\frac{3}{4\cdot 200} + 4\left(\frac{5}{200}\right)^2\right) = -\delta.\end{aligned}$$

Hence α satisfies (4.8a).

To verify that α satisfies (4.8b), a straightforward calculation yields

$$\Delta \boldsymbol{z}^T\boldsymbol{x}^k + \Delta \boldsymbol{x}^T\boldsymbol{z}^k = \Delta \boldsymbol{v}^T\boldsymbol{w} + \Delta \boldsymbol{u}^T\boldsymbol{w} = \boldsymbol{w}^T(\boldsymbol{r}+\boldsymbol{\xi}) = \left(\frac{n}{n+\sqrt{n}} - 1\right)\boldsymbol{w}^T\boldsymbol{w} + \boldsymbol{w}^T\boldsymbol{\xi}$$

and consequently

$$\begin{aligned}(\boldsymbol{x}^k + \alpha\Delta \boldsymbol{x})^T(\boldsymbol{z}^k + \alpha\Delta \boldsymbol{z}) &= (\boldsymbol{x}^k)^T\boldsymbol{z}^k + \alpha(\Delta \boldsymbol{z}^T\boldsymbol{x}^k + \Delta \boldsymbol{x}^T\boldsymbol{z}^k) + \alpha^2\Delta \boldsymbol{x}^T\Delta \boldsymbol{z} \\ &= (1-\alpha)\boldsymbol{w}^T\boldsymbol{w} + \alpha\left(\frac{n}{n+\sqrt{n}}\boldsymbol{w}^T\boldsymbol{w} + \boldsymbol{w}^T\boldsymbol{\xi} + \alpha\Delta \boldsymbol{u}^T\Delta \boldsymbol{v}\right).\end{aligned}$$

By (4.19) and (4.6c) the term in parenthesis satisfies

$$\begin{aligned}\frac{n}{n+\sqrt{n}}\boldsymbol{w}^T\boldsymbol{w} + \boldsymbol{w}^T\boldsymbol{\xi} + \alpha\Delta \boldsymbol{u}^T\Delta \boldsymbol{v} &\ge \frac{(1-\kappa)n}{n+\sqrt{n}}\boldsymbol{w}^T\boldsymbol{w} - \alpha\left(\frac{5\boldsymbol{w}^T\boldsymbol{w}}{(1-\kappa)w_{min}}\right)^2 \\ &= \frac{(1-\kappa)\boldsymbol{w}^T\boldsymbol{w}}{n+\sqrt{n}}\left(n - \frac{1}{8}\right) > 0.\end{aligned}$$

Therefore α satisfies (4.8b), which completes the proof. □

Proof of Theorem 4.3. The theorem follows from the previous lemma by the same argumentation as in [55]. By definition of $\boldsymbol{x}^0$ and $\boldsymbol{z}^0$ we have

$$\phi(\boldsymbol{x}^0, \boldsymbol{z}^0) = (n+\sqrt{n})\ln(n\rho^2) - \sum_{i=1}^{n}\ln(\rho^2) - n\ln(n) = \sqrt{n}\ln(n\rho^2) = \sqrt{n}(\ln(n) + 2\ln(\rho)).$$

Under the hypothesis of the theorem $\phi(\boldsymbol{x}^0, \boldsymbol{z}^0) = O(\sqrt{n}L)$ and $\ln(1/\varepsilon) = O(L)$. Because $\phi(\boldsymbol{x}^k, \boldsymbol{z}^k) \ge \sqrt{n}\ln((\boldsymbol{x}^k)^T\boldsymbol{z}^k)$ and the potential function decreases by at least δ in each iteration,

Algorithm 5 terminates in $O(\sqrt{n}L/\delta) = O(\sqrt{n}(n+\sqrt{n})^2 L)$ iterations. When the algorithm stops in step 2, then $(\boldsymbol{x}^k)^T\boldsymbol{z}^k \le \varepsilon$ and because of (4.15) and (4.16)

$$\left\|\begin{pmatrix} A\boldsymbol{x}^k - \boldsymbol{b} \\ A^T\boldsymbol{y}^k + \boldsymbol{z}^k - \boldsymbol{c}\end{pmatrix}\right\| = \theta^k \left\|\begin{pmatrix} A\boldsymbol{x}^0 - \boldsymbol{b} \\ A^T\boldsymbol{y}^0 + \boldsymbol{z}^0 - \boldsymbol{c}\end{pmatrix}\right\| \le \frac{\varepsilon}{(\boldsymbol{x}^0)^T\boldsymbol{z}^0}\left\|\begin{pmatrix} A\boldsymbol{x}^0 - \boldsymbol{b} \\ A^T\boldsymbol{y}^0 + \boldsymbol{z}^0 - \boldsymbol{c}\end{pmatrix}\right\|.$$

Hence (4.2) also holds. Therefore the final iterate is indeed an ε-approximate solution. On the other hand, if there exists an optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ to (4.1) with $\|(\boldsymbol{x}^*, \boldsymbol{z}^*)\| \le \rho$, it follows from Lemma 4.9 that a step size satisfying (4.8a) and (4.8b) exists. Hence, if the algorithm stops in step 4, then there are no such solutions. □

4.5 Discussion

The analysis gave some insights into the conditions (4.6a)–(4.6c). From (4.13) it is seen that $(\Delta\boldsymbol{x}, \Delta\boldsymbol{z})$ is a descent direction for ϕ if and only if $-\boldsymbol{r}^T\boldsymbol{\xi} < \|\boldsymbol{r}\|^2$. A condition of the form (4.6a) is therefore required in a potential reduction method. (4.6b) bounds the curvature of ϕ along $(\Delta\boldsymbol{x}, \Delta\boldsymbol{z})$. When the iterate is feasible, this condition can be replaced by $\|\boldsymbol{\xi}\| \le c\,\|\boldsymbol{r}\|$ for an arbitrary constant c, since then

$$\|\Delta\boldsymbol{u}\|^2 + \|\Delta\boldsymbol{v}\|^2 = \|\boldsymbol{r} + \boldsymbol{\xi}\|^2 \le (1+c)^2\,\|\boldsymbol{r}\|^2$$

gives the required bound on g_2 in Lemma 4.7 (α needs to be adjusted). For an infeasible iterate (4.6b) is needed in its written form for bounding $\|\Delta\boldsymbol{u}\|$ and $\|\Delta\boldsymbol{v}\|$. (4.6c) guarantees that in the infeasible algorithm the step size restriction (4.8b) can be satisfied.

The choice of α made in Lemma 4.7 and Lemma 4.9 guarantees to reduce ϕ by at least the predefined value δ at each iteration, which yields the polynomial complexity bound. As pointed out by one of the examiners, it prevents superlinear convergence, however. This is because $\left\|\alpha(X^k)^{-1}\Delta\boldsymbol{x}\right\|_\infty$ and $\left\|\alpha(Z^k)^{-1}\Delta\boldsymbol{z}\right\|_\infty$ are bounded by $\tau < 1$, so that if $x_j^k \to 0$, then

$$\frac{x_j^{k+1}}{x_j^k} = \frac{x_j^k + \alpha\Delta x_j}{x_j^k} = 1 + \alpha\left(x_j^k\right)^{-1}\Delta x_j \ge 1 - \tau,$$

and similarly if $z_j^k \to 0$. Hence the convergence rate is only linear. A necessary condition for superlinear convergence is to have $\tau \to 1$ so that x_j^{k+1}/x_j^k can approach zero. Whether superlinear convergence is achieved by a less conservative choice of α has not been answered by the analysis.

Inexact directions of the form (4.5) have been analysed by Monteiro and O'Neil [57] and by Al-Jeiroudi and Gondzio [3] for the long-step path-following method, which sets $\mu = \sigma\boldsymbol{x}^T\boldsymbol{z}/n$, where $\sigma \in [0,1]$, and chooses the step size to keep $x_i z_i \ge \gamma\boldsymbol{x}^T\boldsymbol{z}/n$ for a constant $\gamma \in (0,1)$. Both papers use a basic-nonbasic partitioning of the variables and assume that a residual occurs only in the basic components of the complementarity equations; i. e. the residual in (4.5) takes the form $\boldsymbol{\xi} = (\boldsymbol{\xi}_\mathcal{B}, \boldsymbol{\xi}_\mathcal{N}) = (\boldsymbol{\xi}_\mathcal{B}, \boldsymbol{0})$. Monteiro and O'Neil [57] use the condition

$$\|\boldsymbol{\xi}_\mathcal{B}\| \le \frac{(1-\gamma)\sigma}{4\sqrt{n}}\sqrt{\boldsymbol{x}^T\boldsymbol{z}/n}, \tag{4.22}$$

whereas Al-Jeiroudi and Gondzio [3] require

$$\|W_\mathcal{B}\boldsymbol{\xi}_\mathcal{B}\|_\infty \le \eta\boldsymbol{x}^T\boldsymbol{z}/n \tag{4.23}$$

with $\eta < 1$ depending on σ and γ. Notice that (4.23) measures the residual in the unscaled complementarity equations. A similar condition was used by Gondzio [25] for a feasible long-step path-following IPM for convex QP, requiring that

$$\|W\boldsymbol{\xi}\|_\infty \le \delta\,\|W\boldsymbol{r}\|_\infty \tag{4.24}$$

with $\delta < 1$ depending on σ and γ. The issue with convergence analysis in the path-following framework is that keeping the iterates within the central path neighbourhood limits the amount of inexactness that can be tolerated. Indeed, (4.22) obviously becomes restrictive for large-scale problems; the constant in (4.23) is constrained by at least $\eta < 0.5$ (see [3, Section 4]); and a possible choice for δ in (4.24) is given in [25, Section 3.2] as 0.05.

The obvious question regarding practicality is why not allowing residuals in the primal and dual feasibility equations. The condition $A\Delta\boldsymbol{x} = \boldsymbol{b} - A\boldsymbol{x}^k$ is a significant restriction to the set of available solution methods because it typically requires a basis matrix of A to be realized. Mizuno and Jarre [54] considered an inexact path-following IPM that allowed residuals in the feasibility rather than the complementarity equations. Their analysis showed that the residuals must be measured in a norm depending on A that is not accessible in practice. It therefore seems unlikely that a practical algorithm with a theoretical foundation can be derived.

Having the residuals in the complementarity equations is also more natural from the standpoint of Newton's method, which linearizes the perturbed optimality conditions (4.3) at the current iterate. Because a linearization error $\mu - (x_i + \Delta x_i)(z_i + \Delta z_i)$ occurs in the nonlinear part of the system, there seems to be no reason for solving the corresponding step equations exactly. As long as the computation error does not dominate the linearization error, Newton's method converges.

The analysis presented in the present work suggests to measure the residual from an iterative linear solver in the scaled complementarity equations $\Delta\boldsymbol{u} + \Delta\boldsymbol{v} = \boldsymbol{r}$, because then the error in the scaled step directions $\Delta\boldsymbol{u}$ and $\Delta\boldsymbol{v}$ remains bounded even for weak tolerances (Lemma 4.4). Boundedness of $\|\Delta\boldsymbol{u}\|$ and $\|\Delta\boldsymbol{v}\|$ in terms of $\boldsymbol{x}^T\boldsymbol{z}/w_{min}$ (potential reduction method, see (4.18)) or $\sqrt{\boldsymbol{x}^T\boldsymbol{z}/n}$ (path-following method, see Wright [80, Chapter 6]) is key to the convergence of the IPM. Because implementations are typically related to path-following methods, the criterion $\|\boldsymbol{\xi}\| \leq \kappa\sqrt{\boldsymbol{x}^T\boldsymbol{z}/n}$ is preferable. The latter is used with the infinity norm in the author's LP code.

5 Basis Matrices in the Interior Point Method

Basis matrices will be used in three components of the LP solver presented in Chapter 6: preconditioning, crossover and for the dynamic elimination of variables. This chapter covers the theoretical background for the computational techniques.

The LP problem is stated in standard form of a primal-dual pair

$$\underset{\boldsymbol{x}}{\text{minimize}}\ \boldsymbol{c}^T\boldsymbol{x} \quad \text{subject to } A\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \geq \boldsymbol{0}, \tag{5.1a}$$

$$\underset{\boldsymbol{y},\boldsymbol{z}}{\text{maximize}}\ \boldsymbol{b}^T\boldsymbol{y} \quad \text{subject to } A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c},\ \boldsymbol{z} \geq \boldsymbol{0}, \tag{5.1b}$$

where A is an $m \times n$ matrix of full row rank. A basis $\mathcal{B}$ defines a "vertex solution" $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ to the primal and dual equations by setting $\hat{\boldsymbol{x}}_{\mathcal{N}} = \boldsymbol{0}$ and $\hat{\boldsymbol{z}}_{\mathcal{B}} = \boldsymbol{0}$. The vertex is primal feasible if $\hat{\boldsymbol{x}}_{\mathcal{B}} = A_{\mathcal{B}}^{-1}\boldsymbol{b} \geq \boldsymbol{0}$ and dual feasible if $\hat{\boldsymbol{z}}_{\mathcal{N}} = \boldsymbol{c}_{\mathcal{N}} - A_{\mathcal{N}}^T\hat{\boldsymbol{y}} \geq \boldsymbol{0}$, where $\hat{\boldsymbol{y}} = A_{\mathcal{B}}^{-T}\boldsymbol{c}_{\mathcal{B}}$. A basis is said to be "optimal" if its vertex is primal and dual feasible.

5.1 Background

Throughout the chapter it is assumed that a strictly feasible primal-dual point exists, i. e. a point $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ satisfying

$$A\boldsymbol{x} = \boldsymbol{b}, \quad A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c}, \quad (\boldsymbol{x}, \boldsymbol{z}) > \boldsymbol{0}.$$

The existence of such a point is a sufficient condition for (5.1a) and (5.1b) to have optimal solutions. In this case, the Goldman-Tucker theorem [22] states that there exists at least one strictly complementary solution; i. e. an optimal solution for which exactly one of x_i and z_i is positive for $1 \leq i \leq n$. The following two properties of strictly complementary solutions will be used repeatedly in the remainder of this chapter.

Lemma 5.1. *(i) The partitioning into positive primal and positive dual variables is identical for all strictly complementary solutions.*

(ii) If $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ is a strictly complementary solution and $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ is an arbitrary solution, then $x_i^* = 0 \implies x_i = 0$ and $z_i^* = 0 \implies z_i = 0$ for $1 \leq i \leq n$.

Proof. For example in Wright [80, Chapter 2]. □

The existence of a strictly feasible primal-dual point also implies the existence of the central path $\mathcal{C}$ (see [80, Theorem 2.8]), which is the set of points $\left(\boldsymbol{x}_\mu, \boldsymbol{y}_\mu, \boldsymbol{z}_\mu\right)$ parametrized by $\mu > 0$ that satisfy

$$\begin{aligned} A\boldsymbol{x}_\mu &= \boldsymbol{b}, \\ A^T\boldsymbol{y}_\mu + \boldsymbol{z}_\mu &= \boldsymbol{c}, \\ (\boldsymbol{x}_\mu, \boldsymbol{z}_\mu) &> \boldsymbol{0}, \\ (\boldsymbol{x}_\mu)_i(\boldsymbol{z}_\mu)_i &= \mu \quad \text{for } 1 \leq i \leq n. \end{aligned}$$

As $\mu \to 0$, the trajectory $\left(\boldsymbol{x}_\mu, \boldsymbol{y}_\mu, \boldsymbol{z}_\mu\right)$ converges to a strictly complementary solution to (5.1). Each point in $\mathcal{C}$ defines a positive definite diagonal matrix $D_\mu = \operatorname{diag}\left(\sqrt{(\boldsymbol{x}_\mu)_i/(\boldsymbol{z}_\mu)_i}\right)$, which will become important below. For a derivation of the properties of the central path see Megiddo [51].

A path-following IPM solves (5.1) by generating a sequence of iterates $\left(\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{z}^k\right)$ that lie in some vicinity of the central path while driving the complementarity measures $\mu^k = \left(\boldsymbol{x}^k\right)^T\boldsymbol{z}^k/n$

to zero. In each iteration a damped Newton step is taken toward the central point whose complementarity measure is a factor $\sigma \in [0,1]$ times μ^k. Assuming that the current iterate is feasible, the Newton direction is the solution to the linear system

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ Z^k & 0 & X^k \end{bmatrix} \begin{pmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{y} \\ \Delta \boldsymbol{z} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ -X^k Z^k \boldsymbol{e} + \sigma \mu^k \boldsymbol{e} \end{pmatrix}.$$

After eliminating $\Delta \boldsymbol{z}$ and scaling by $D^k = \operatorname{diag}\left(\boldsymbol{d}^k\right)$, where $d_i^k = \sqrt{x_i^k / z_i^k}$, this system becomes

$$\begin{bmatrix} I & D^k A^T \\ AD^k & 0 \end{bmatrix} \begin{pmatrix} (D^k)^{-1} \Delta \boldsymbol{x} \\ -\Delta \boldsymbol{y} \end{pmatrix} = \begin{pmatrix} D^k \left(-\boldsymbol{z}^k + \sigma \mu^k \left(X^k\right)^{-1} \boldsymbol{e} \right) \\ \mathbf{0} \end{pmatrix} =: \begin{pmatrix} D^k \boldsymbol{r}^a \\ \mathbf{0} \end{pmatrix}.$$

Therefore $-\Delta \boldsymbol{y}$ is the solution to the least squares problem

$$\underset{\boldsymbol{\lambda}}{\text{minimize}} \left\| D^k A^T \boldsymbol{\lambda} - D^k \boldsymbol{r}^a \right\|. \tag{5.3}$$

The matrix D^k is called "scaling matrix" in interior point terminology and usually attains unfavourable numerical properties in advanced iterations. Because each limit point of the iteration sequence of the IPM is a strictly complementary solution to (5.1) (see Wright [80, Theorem 5.14] and Tapia et al. [76]), the diagonal entries of D^k either tend to zero or to infinity. If the columns of A corresponding to large scaling factors have rank less than m, AD^k eventually approaches a rank deficient matrix and its condition number becomes unbounded. But even if AD^k remains of full rank, the wide spread of the scaling factors almost always causes the least squares problem to become very ill conditioned in advanced iterations.

Definition 5.2. *Let $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ be any strictly complementary solution to* (5.1)*. Given any basic-nonbasic partitioning $(\mathcal{B}, \mathcal{N})$ of the variables, we define the index sets*

$$\begin{aligned} \mathcal{B}^o &= \{i \in \mathcal{B} \mid z_i^* = 0\}, & \mathcal{B}^+ &= \{i \in \mathcal{B} \mid z_i^* > 0\}, \\ \mathcal{N}^o &= \{i \in \mathcal{N} \mid x_i^* = 0\}, & \mathcal{N}^+ &= \{i \in \mathcal{N} \mid x_i^* > 0\}, \end{aligned}$$

and call the variables in $\mathcal{N}^+$ "primal superbasic" and the variables in $\mathcal{B}^+$ "dual superbasic". The basis $\mathcal{B}$ is said to have "maximum cardinality"[2] if

$$\operatorname{rank}(A_{\mathcal{B}^o}) = \operatorname{rank}(A_{\mathcal{B}^o \cup \mathcal{N}^+}).$$

The definition of maximum cardinality says that $\mathcal{B}$ contains the maximum number of variables for which $x_i^* > 0$. By means of Lemma 5.1(i) the definition of the index sets $\mathcal{B}^o$, $\mathcal{B}^+$, $\mathcal{N}^o$ and $\mathcal{N}^+$ is independent of the particular choice of the strictly complementary solution.

In the following let $\pi : \mathcal{B} \to \{1, \ldots, m\}$ be the mapping that assigns each basic variable i to its position in $\mathcal{B}$; i. e. $\mathcal{B}_{\pi(i)} = i$ for all $i \in \mathcal{B}$.

Lemma 5.3. *$\mathcal{B}$ has maximum cardinality if and only if $(A_{\mathcal{B}}^{-1} A_j)_{\pi(i)} = 0$ for all $i \in \mathcal{B}^+$ and all $j \in \mathcal{N}^+$.*

Proof. $\mathcal{B}$ has maximum cardinality

$\iff \operatorname{rank}(A_{\mathcal{B}^o}) = \operatorname{rank}(A_{\mathcal{B}^o \cup \mathcal{N}^+})$

$\iff$ Each column in $A_{\mathcal{N}^+}$ is linearly dependent on the columns in $A_{\mathcal{B}^o}$.

$\iff$ For each $j \in \mathcal{N}^+$ there exists a vector $\boldsymbol{\lambda}$ such that $A_{\mathcal{B}^o} \boldsymbol{\lambda} = A_j$.

$\iff$ For each $j \in \mathcal{N}^+$ we have $(A_{\mathcal{B}}^{-1} A_j)_{\pi(i)} = 0$ for all $i \in \mathcal{B}^+$. □

[2] Because a basis always contains m indices, there should be no confusion with the cardinality of a set.

If $\mathcal{B}$ has maximum cardinality, $|\mathcal{B}^+|$ and $|\mathcal{N}^+|$ are said to be the number of primal and dual degeneracies of the LP model. (Notice that a primal superbasic variable means a dual degeneracy and vice versa.) By definition, LP degeneracy is a property of the problem data, whereas the number of degenerate variables at a vertex $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$, i. e. those $i \in \mathcal{B}$ for which $\hat{x}_i = 0$ and those $i \in \mathcal{N}$ for which $\hat{z}_i = 0$, are a property of the basis.

Lemma 5.4. *When the vertex $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ is optimal, then the number of primal and dual degenerate variables are lower bounded by the respective number of degeneracies of the LP problem.*

Proof. Let $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ be any strictly complementary solution. By means of Lemma 5.1(ii) we have $x_i^* = 0 \implies \hat{x}_i = 0$ and $z_i^* = 0 \implies \hat{z}_i = 0$. Hence, if $\mathcal{B}$ is a basis corresponding to the vertex, then $\hat{\boldsymbol{x}}_{\mathcal{B}^+} = \boldsymbol{0}$ and $\hat{\boldsymbol{z}}_{\mathcal{N}^+} = \boldsymbol{0}$. □

5.2 Comparison of Maximum Volume and Maximum Weight Bases

This section compares two criteria for choosing a basis matrix as preconditioner in the linear programming IPM. Consider the normal equations resulting from (5.3), $A(D^k)^2A^T\Delta\boldsymbol{y} = -A(D^k)^2\boldsymbol{r}^a$, which are transformed by basis preconditioning into

$$C\Delta\boldsymbol{u} := (A_{\mathcal{B}}D_{\mathcal{B}}^k)^{-1}A(D^k)^2A^T(A_{\mathcal{B}}D_{\mathcal{B}}^k)^{-T}\Delta\boldsymbol{u} = -(A_{\mathcal{B}}D_{\mathcal{B}}^k)^{-1}A(D^k)^2\boldsymbol{r}^a.$$

Al-Jeiroudi et al. [4] implemented an iterative linear solver with basis preconditioning in an IPM. They motivated the idea by the fact that close to the solution of a nondegenerate LP model exactly m scaling factors become large and the corresponding columns of A form a nonsingular matrix. Hence $A(D^k)^2A^T$ converges to $A_{\mathcal{B}}(D_{\mathcal{B}}^k)^2A_{\mathcal{B}}^T$, where $\mathcal{B}$ is the unique optimal LP basis. The obvious approach was building the basis from the first m linearly independent columns of A in decreasing order of the scaling factors. Such a basis is said to have "maximum weight". Formally, a maximum weight basis for $\left(A, \boldsymbol{d}^k\right)$ is defined through the following algorithm:

Algorithm 6 Maxweight (from Monteiro et al. [58, Section 2])

Input: $A \in \mathbb{R}^{m\times n}$ of rank m, $\boldsymbol{d} \in \mathbb{R}^n_{>0}$

1: Order the elements of $\boldsymbol{d}$ such that $d_1 \geq \cdots \geq d_n$; order the columns of A accordingly.
2: Let $\mathcal{B} = \emptyset$ and $l = 1$.
3: **while** $|\mathcal{B}| < m$ **do**
4: If A_l is linearly independent of the columns in $A_{\mathcal{B}}$, set $\mathcal{B} = \mathcal{B} \cup \{l\}$.
5: $l = l + 1$

Monteiro et al. [58] analysed the maximum weight basis preconditioner without assumptions on the scaling matrix D^k. They proved that it bounds the condition number of C by $m\bar{\chi}_A^2$, where

$$\bar{\chi}_A = \max_{\mathcal{B} \text{ basis}} \left\{ \left\| A_{\mathcal{B}}^{-1}A \right\| \right\}.$$

The key point in the proof of [58, Lemma 2.1] is that

$$|(D_{\mathcal{B}}^k)^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}}^k| \leq |A_{\mathcal{B}}^{-1}A_{\mathcal{N}}| \tag{5.4}$$

when $\mathcal{B}$ is a maximum weight basis for $\left(A, \boldsymbol{d}^k\right)$; i. e. scaling with $(D_{\mathcal{B}}^k)^{-1}$ and $D_{\mathcal{N}}^k$ does not increase the entries in the tableau matrix. Consequently, a maximum weight basis bounds the condition number of C independently of D^k. The bound is not of much use in practice, however, because $\bar{\chi}_A$ may grow exponentially with the dimension of A (see the example in Section 3.1).

A stronger bound is obtained if $\mathcal{B}$ is chosen instead such that $A_{\mathcal{B}}D^k_{\mathcal{B}}$ has ρ-maximum volume in AD^k for some $\rho \geq 1$. The defining property of a maximum volume basis is that $\left\|(D^k_{\mathcal{B}})^{-1}A^{-1}_{\mathcal{B}}A_{\mathcal{N}}D^k_{\mathcal{N}}\right\|_{\max} \leq \rho$, which yields the bound $\operatorname{cond}(C) \leq 1 + \rho^2 m(n-m)$ derived in Section 3.1. The maximum volume basis directly bounds the entries in the scaled tableau matrix, whereas the maximum weight basis only guarantees (5.4), yielding a bound that depends on $\bar{\chi}_A$.

The following lemma shows that a maximum weight basis for $\left(A, \boldsymbol{d}^k\right)$ and a maximum volume basis for AD^k have maximum cardinality when the scaling factors correspond to a sufficiently late iterate of a path-following method.

Lemma 5.5. *Let $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ be a strictly complementary solution to (5.1) and $\mathcal{I} \cup \mathcal{J} = \{1, \ldots, n\}$ be the partitioning of variables such that $\boldsymbol{x}^*_{\mathcal{I}} > \boldsymbol{0}$ and $\boldsymbol{x}^*_{\mathcal{J}} = \boldsymbol{0}$. Let $\left(\boldsymbol{d}^k\right)_{k\in\mathbb{N}}$ be a sequence of vectors $\boldsymbol{d}^k \in \mathbb{R}^n_{>0}$ for which $\boldsymbol{d}^k_{\mathcal{I}} \to \boldsymbol{\infty}$ and $\boldsymbol{d}^k_{\mathcal{J}} \to \boldsymbol{0}$ as $k \to \infty$. Then there exists $\bar{k} \in \mathbb{N}$ such that the following holds true for all $k \geq \bar{k}$:*

(i) Any maximum weight basis for $\left(A, \boldsymbol{d}^k\right)$ has maximum cardinality.

(ii) Any maximum volume basis for AD^k has maximum cardinality.

Proof. For part (i) choose $\bar{k} \in \mathbb{N}$ large enough such that $\max_{j\in\mathcal{J}} d^k_j < \min_{i\in\mathcal{I}} d^k_i$ for all $k \geq \bar{k}$. Any maximum weight basis for $\left(A, \boldsymbol{d}^k\right)$ then contains the maximum number of variables from $\mathcal{I}$, and therefore has maximum cardinality.

For part (ii) let

$$\sigma := \min_{\mathcal{B} \text{ basis}} \left\{ \min_{p,q} \left\{ \alpha \mid \alpha = |A^{-1}_{\mathcal{B}}A_{\mathcal{N}}|_{pq},\, \alpha \neq 0 \right\} \right\}$$

and choose $\bar{k} \in \mathbb{N}$ large enough such that

$$\min_{j\in\mathcal{J}} \left\{ (d^k_j)^{-1} \right\} \cdot \sigma \cdot \min_{i\in\mathcal{I}} \left\{ d^k_i \right\} > 1 \tag{5.5}$$

for all $k \geq \bar{k}$. Now let $\mathcal{B}$ be a maximum volume basis for AD^k, where $k \geq \bar{k}$. Consider any $j \in \mathcal{B}^+$ and $i \in \mathcal{N}^+$, and let p and q be such that $j = \mathcal{B}_p$ and $i = \mathcal{N}_q$. Because of the maximum volume property we have

$$(d^k_j)^{-1}|A^{-1}_{\mathcal{B}}A_{\mathcal{N}}|_{pq}\, d^k_i = |(D^k_{\mathcal{B}})^{-1}A^{-1}_{\mathcal{B}}A_{\mathcal{N}}D^k_{\mathcal{N}}|_{pq} \leq 1.$$

It follows from (5.5), $\mathcal{B}^+ \subseteq \mathcal{J}$ and $\mathcal{N}^+ \subseteq \mathcal{I}$ that $(A^{-1}_{\mathcal{B}}A_{\mathcal{N}})_{pq} = 0$. By Lemma 5.3 $\mathcal{B}$ has maximum cardinality.

To complete the proof of the lemma, choose $\bar{k}$ as the maximum of the values constructed for part (i) and part (ii). □

It follows from the lemma that if $\boldsymbol{d}^k$ corresponds to a sufficiently late IPM iterate and $\mathcal{B}$ is a maximum weight basis for $\left(A, \boldsymbol{d}^k\right)$, then the scaled tableau matrix can be partitioned as

$$(D^k_{\mathcal{B}})^{-1}A^{-1}_{\mathcal{B}}A_{\mathcal{N}}D^k_{\mathcal{N}} = \begin{array}{c} \\ \mathcal{B}^o \\ \mathcal{B}^+ \end{array} \begin{array}{c} \begin{array}{cc} \mathcal{N}^o & \mathcal{N}^+ \end{array} \\ \begin{bmatrix} \to 0 & O(\bar{\chi}_A) \\ O(\bar{\chi}_A) & 0 \end{bmatrix} \end{array}. \tag{5.6}$$

Because the off-diagonal blocks may contain large entries, the matrix is likely to have a wide spectrum unless the number of degeneracies of the LP problem is small. In practice LP models

are often highly degenerate, meaning that $|\mathcal{B}^+|$ and $|\mathcal{N}^+|$ are of order m and $n-m$, respectively; see Achterberg [1] for a statistic on LP models from combinatorial optimization. But even on a nondegenerate problem the effectiveness of the preconditioner is only guaranteed asymptotically and may not be observed until very late in the interior point solve.

For the maximum volume basis, the scaled tableau matrix at a sufficiently late IPM iterate can also be partitioned as in (5.6), but now the entries in the off-diagonal blocks are $\leq \rho$ in absolute value. This leads to the asymptotic bound

$$\operatorname{cond}(C) \leq 1 + \rho^2 \left(|\mathcal{B}^+||\mathcal{N}^o| + |\mathcal{B}^o||\mathcal{N}^+| \right)$$

because there are at most $|\mathcal{B}^+||\mathcal{N}^o| + |\mathcal{B}^o||\mathcal{N}^+|$ entries in the scaled tableau matrix which do not tend to zero.

A further advantage of the maximum volume criterion is to be invariant to a column scaling. Let $R \in \mathbb{R}^{m \times m}$ and $S \in \mathbb{R}^{n \times n}$ be positive definite diagonal matrices and consider the LP problem with $\tilde{A} = RAS$, $\tilde{\boldsymbol{b}} = R\boldsymbol{b}$ and $\tilde{\boldsymbol{c}} = S\boldsymbol{c}$ instead of A, $\boldsymbol{b}$ and $\boldsymbol{c}$. Then the central path $\mathcal{C}$ is scaled to $\tilde{\mathcal{C}}$ by

$$\left(\boldsymbol{x}_\mu, \boldsymbol{y}_\mu, \boldsymbol{z}_\mu\right) \in \mathcal{C} \implies \left(S^{-1}\boldsymbol{x}_\mu, R^{-1}\boldsymbol{y}_\mu, S\boldsymbol{z}_\mu\right) \in \tilde{\mathcal{C}}$$

and the scaling matrix at the central point corresponding to the same μ becomes $\tilde{D}_\mu = D_\mu S^{-1}$. Hence the scaled tableau matrix

$$(\tilde{D}_\mu)_{\mathcal{B}}^{-1} \tilde{A}_{\mathcal{B}}^{-1} \tilde{A}_{\mathcal{N}} (\tilde{D}_\mu)_{\mathcal{N}} = S_{\mathcal{B}} (D_\mu)_{\mathcal{B}}^{-1} S_{\mathcal{B}}^{-1} A_{\mathcal{B}}^{-1} R^{-1} R A_{\mathcal{N}} S_{\mathcal{N}} (D_\mu)_{\mathcal{N}} S_{\mathcal{N}}^{-1} = (D_\mu)_{\mathcal{B}}^{-1} A_{\mathcal{B}}^{-1} A_{\mathcal{N}} (D_\mu)_{\mathcal{N}}$$

remains unchanged, and so does the maximum volume property of $\mathcal{B}$.

The maximum weight basis is invariant to a row scaling only. Consider the point on the central path for a fixed μ and its associated scaling matrix D_μ. Given any basis $\mathcal{B}$, we can scale the columns of A and thereby the central path such that $(D_\mu)_{\mathcal{B}}$ become the m largest entries of D_μ. Hence a scaling operation can transform any basis into a maximum weight basis, despite the fact that the relevant matrices $(D_\mu)_{\mathcal{B}}^{-1} A_{\mathcal{B}}^{-1} A_{\mathcal{N}} (D_\mu)_{\mathcal{N}}$ remain unchanged for all $\mathcal{B}$.

The scaling property does not play a role for totally unimodular matrices, which intrinsically have a "correct" column scaling. If A is totally unimodular, the tableau matrices only have entries ± 1 and 0, so that maximum volume and maximum weight are the same property. For the special case of the node-arc incidence matrix of a directed graph Monteiro et al. [58] already derived the stronger bound on the condition number of C.

It might seem a disadvantage of the maximum volume criterion that finding such a basis requires an iterative update procedure, whereas a maximum weight basis can be determined by a variant of LU factorization. This is hardly relevant in practice, however, as we will see that a good approximation to a maximum volume basis can be obtained efficiently in the IPM. Furthermore, the maximum weight criterion assumes that linear dependency is a strictly combinatorial notion and it is unlikely that such a method can be implemented robustly while maintaining its theoretical properties. In contrast, the maximum volume algorithm from Chapter 2 was straightforwardly translated into a numerically stable computer program.

Another criterion for basis selection was used by Oliveira and Sorensen [62], who composed the basis matrix of the first m linearly independent columns of A, after ordering the columns by decreasing 1-norm of $A_j d_j$. They justified this choice to be an "inexpensive heuristic" for minimizing $\left\| D_{\mathcal{B}}^{-1} A_{\mathcal{B}}^{-1} A_{\mathcal{N}} D_{\mathcal{N}} \right\|$ (see [62, Section 4.3]). Regarding the scaling property of the central path discussed above, the 1-norm criterion is invariant to column scaling (because AD_μ remains unchanged), but in general not invariant to row scaling. It also gives no useful bound

on $\left\|D_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}}\right\|$, as seen by adapting the example from Section 3.1 to

$$A = \begin{bmatrix} 1 & -1 & & \cdots & -1 & 1 \\ & 1 & & & -1 & 1 \\ & & \ddots & & \vdots & \vdots \\ & & & 1 & -1 & 1 \\ & & & & 1 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 1 & & & & & \\ & \frac{1}{2} & & & & \\ & & \ddots & & & \\ & & & \frac{1}{m} & & \\ & & & & \frac{1}{m} & \end{bmatrix},$$

where A has dimension $m \times (m+1)$. Because the 1-norm of each column of AD is 1, the criterion allows to choose the first m columns as basis, resulting in $\left\|D_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}}\right\|$ to grow proportionally to $2^m/m$.

5.3 Comparison of Maximum Volume and Optimal Bases

Intuitively one might expect that a maximum volume basis for AD_μ would eventually become an optimal LP basis if we follow the central path toward $\mu = 0$. While this is obviously true for a nondegenerate problem, the conjecture will turn out to be false in general. The following analysis reveals the common features and the differences between the two types of bases. Throughout, we consider points $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ on the central path with complementarity measure $\mu = \boldsymbol{x}^T\boldsymbol{z}/n$ and scaling matrix $D = D_\mu$. For such a point and any basis $\mathcal{B}$ the scaled tableau matrix can be expressed alternatively as

$$D_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}} = \frac{1}{\sqrt{\mu}}D_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}}\sqrt{\mu} = X_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}X_{\mathcal{N}},$$
$$D_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}} = \sqrt{\mu}D_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}}\frac{1}{\sqrt{\mu}} = Z_{\mathcal{B}}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}Z_{\mathcal{N}}^{-1}.$$

The following lemma provides a connection between any point on the central path and the vertex corresponding to any basis.

Lemma 5.6. *Let $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in \mathcal{C}$, $D = \operatorname{diag}\left(\sqrt{x_i/z_i}\right)$. Let $\mathcal{B}$ be a basis with vertex $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$. Then*

(i) $D_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}}\boldsymbol{e}_{\mathcal{N}} = X_{\mathcal{B}}^{-1}\hat{\boldsymbol{x}}_{\mathcal{B}} - \boldsymbol{e}_{\mathcal{B}}$,

(ii) $\boldsymbol{e}_{\mathcal{B}}^T D_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}} = \boldsymbol{e}_{\mathcal{N}}^T - \hat{\boldsymbol{z}}_{\mathcal{N}}^T Z_{\mathcal{N}}^{-1}$,

(iii) $\sum_{i\in\mathcal{B}} \hat{x}_i(x_i)^{-1} + \sum_{j\in\mathcal{N}} \hat{z}_j(z_j)^{-1} = n$.

Proof. Part (i) and (ii) are computed from

$$\begin{aligned} D_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}}\boldsymbol{e}_{\mathcal{N}} &= X_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}X_{\mathcal{N}}\boldsymbol{e}_{\mathcal{N}} \\ &= X_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}\boldsymbol{x}_{\mathcal{N}} \\ &= X_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}(\boldsymbol{b} - A_{\mathcal{B}}\boldsymbol{x}_{\mathcal{B}}) \\ &= X_{\mathcal{B}}^{-1}\hat{\boldsymbol{x}}_{\mathcal{B}} - \boldsymbol{e}_{\mathcal{B}}, \\ \boldsymbol{e}_{\mathcal{B}}^T D_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}} &= \boldsymbol{e}_{\mathcal{B}}^T Z_{\mathcal{B}}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}Z_{\mathcal{N}}^{-1} \\ &= \boldsymbol{z}_{\mathcal{B}}^T A_{\mathcal{B}}^{-1}A_{\mathcal{N}}Z_{\mathcal{N}}^{-1} \\ &= (\boldsymbol{c}_{\mathcal{B}}^T A_{\mathcal{B}}^{-1} - \boldsymbol{y}^T)A_{\mathcal{N}}Z_{\mathcal{N}}^{-1} \\ &= (\boldsymbol{c}_{\mathcal{B}}^T A_{\mathcal{B}}^{-1}A_{\mathcal{N}} - \boldsymbol{c}_{\mathcal{N}}^T + \boldsymbol{z}_{\mathcal{N}}^T)Z_{\mathcal{N}}^{-1} \\ &= \boldsymbol{e}_{\mathcal{N}}^T - \hat{\boldsymbol{z}}_{\mathcal{N}}^T Z_{\mathcal{N}}^{-1}. \end{aligned}$$

Part (iii) is obtained by setting equal the two expressions

$$\begin{aligned}
\boldsymbol{e}_{\mathcal{B}}^T D_{\mathcal{B}}^{-1} A_{\mathcal{B}}^{-1} A_{\mathcal{N}} D_{\mathcal{N}} \boldsymbol{e}_{\mathcal{N}} &\overset{(i)}{=} \boldsymbol{e}_{\mathcal{B}}^T X_{\mathcal{B}}^{-1} \hat{\boldsymbol{x}}_{\mathcal{B}} - \boldsymbol{e}_{\mathcal{B}}^T \boldsymbol{e}_{\mathcal{B}}, \\
\boldsymbol{e}_{\mathcal{B}}^T D_{\mathcal{B}}^{-1} A_{\mathcal{B}}^{-1} A_{\mathcal{N}} D_{\mathcal{N}} \boldsymbol{e}_{\mathcal{N}} &\overset{(ii)}{=} \boldsymbol{e}_{\mathcal{N}}^T \boldsymbol{e}_{\mathcal{N}} - \hat{\boldsymbol{z}}_{\mathcal{N}}^T Z_{\mathcal{N}}^{-1} \boldsymbol{e}_{\mathcal{N}}.
\end{aligned}$$

□

From part (i) and (ii) of the lemma the difference between a maximum volume basis and an optimal basis can be illustrated in terms of the scaled tableau matrix. Let $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in \mathcal{C}$ and let $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ be the vertex corresponding to basis $\mathcal{B}$. Because $\boldsymbol{x}_{\mathcal{B}}$ and $\boldsymbol{z}_{\mathcal{N}}$ are positive, $\mathcal{B}$ is primal feasible if and only if $X_{\mathcal{B}}^{-1} \hat{\boldsymbol{x}}_{\mathcal{B}} \geq \mathbf{0}$, and dual feasible if and only if $Z_{\mathcal{N}}^{-1} \hat{\boldsymbol{z}}_{\mathcal{N}} \geq \mathbf{0}$. By means of Lemma 5.6 (i) and (ii) this is equivalent to $D_{\mathcal{B}}^{-1} A_{\mathcal{B}}^{-1} A_{\mathcal{N}} D_{\mathcal{N}} \boldsymbol{e}_{\mathcal{N}} \geq -\boldsymbol{e}_{\mathcal{B}}$ and $\boldsymbol{e}_{\mathcal{B}}^T D_{\mathcal{B}}^{-1} A_{\mathcal{B}}^{-1} A_{\mathcal{N}} D_{\mathcal{N}} \leq \boldsymbol{e}_{\mathcal{N}}^T$; i. e. $\mathcal{B}$ is an optimal basis if and only if the row sums of the scaled tableau matrix are bounded from below by -1 and the column sums are bounded from above by 1. In contrast, for a maximum volume basis each entry in the scaled tableau matrix is bounded by 1 in absolute value.

It will be shown by example that a maximum volume basis need not become optimal and that an optimal basis need not attain maximum volume if we follow the central path toward $\mu = 0$. Consider the linear program

$$\min_{\boldsymbol{x}} 0 \qquad \text{s.t.} \quad \begin{bmatrix} 1 & 0 & 1 & & & \\ 0 & 1 & & 1 & & \\ 1 & 1 & & & 1 & \\ -1 & -1 & & & & 1 \end{bmatrix} \boldsymbol{x} = \begin{pmatrix} \delta \\ \delta \\ 3/2 \\ -1/2 \end{pmatrix}, \quad \boldsymbol{x} \geq \mathbf{0}$$

for $\delta = \frac{4509}{3275} \approx 1.4$, whose feasible region is illustrated in Figure 2 by treating x_3, x_4, x_5 and x_6 as slack variables. Because the objective function is zero, the primal components of the central path are constant. From the optimality conditions it can be computed analytically that $x_1 = x_2 = 0.54$, $x_3 = x_4 = \delta - 0.54$, $x_5 = 0.42$ and $x_6 = 0.58$. For any basis $\mathcal{B}$ the scaled tableau matrix is the same for all central points.

The basis $\mathcal{B} = \{2, 3, 5, 6\}$ defining the vertex $\hat{\boldsymbol{x}}$ is optimal. Computing

$$X_{\mathcal{B}}^{-1} A_{\mathcal{B}}^{-1} A_1 x_1 = \begin{pmatrix} 0 \\ x_1/x_3 \\ x_1/x_5 \\ -x_1/x_6 \end{pmatrix}$$

and using that $x_5 < x_1$ shows that $\mathcal{B}$ does not have maximum volume. On the other hand, the basis $\mathcal{B}' = \{2, 3, 4, 6\}$ defining the vertex $\hat{\boldsymbol{x}}'$ is primal infeasible. Computing

$$X_{\mathcal{B}'}^{-1} A_{\mathcal{B}'}^{-1} A_1 x_1 = \begin{pmatrix} x_1/x_2 \\ x_1/x_3 \\ -x_1/x_4 \\ 0 \end{pmatrix}, \quad X_{\mathcal{B}'}^{-1} A_{\mathcal{B}'}^{-1} A_5 x_5 = \begin{pmatrix} x_5/x_2 \\ 0 \\ -x_5/x_4 \\ x_5/x_6 \end{pmatrix}$$

and substituting the values for $\boldsymbol{x}$ verifies that $\mathcal{B}'$ has maximum volume.

The example also proves that a maximum weight basis need not be optimal and that an optimal basis need not have maximum weight. It is readily verified that $\mathcal{B}'$ as defined above and $\mathcal{B}'' = \{1, 3, 4, 6\}$ are the only maximum weight bases for any point on the central path. Their vertices $\hat{\boldsymbol{x}}'$ and $\hat{\boldsymbol{x}}''$ are primal infeasible.

The common feature of maximum volume and maximum weight bases in the limit $\mu \to 0$ is maximum cardinality (Lemma 5.3). The same holds true for an optimal basis.

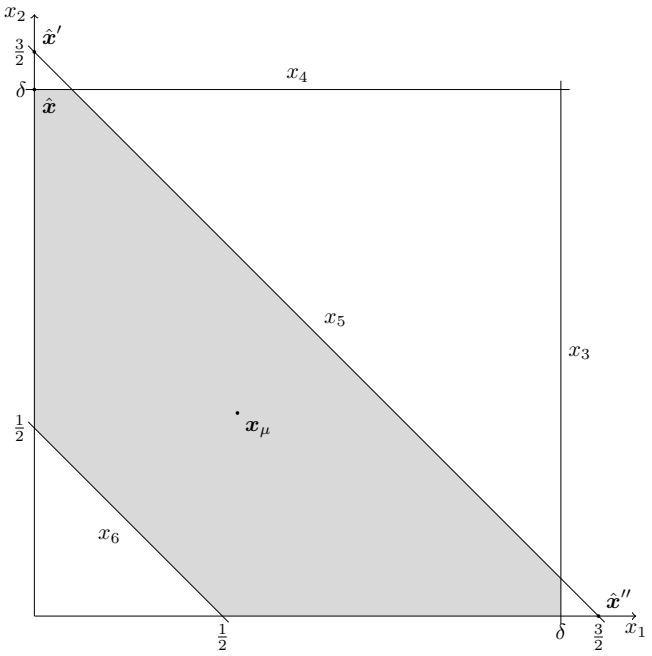


Figure 2: Feasible region of example problem. Constraints are labelled by their slack variable.

Lemma 5.7. *An optimal basis has maximum cardinality.*

Proof. Let $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ be a strictly complementary solution to (5.1) and let $\mathcal{B}$ be an optimal basis. We need to show that if $i = \mathcal{B}_p$ and $j \in \mathcal{N}$ are such that $x_i^* = 0$ and $x_j^* > 0$, then $(A_{\mathcal{B}}^{-1} A_j)_p = 0$.

We can assume without loss of generality that there exists exactly one $j \in \mathcal{N}$ for which $x_j^* > 0$. (Otherwise consider the LP problem obtained by fixing all primal superbasic variables except j at their optimal solution value.) Let $i = \mathcal{B}_p$ be such that $x_i^* = 0$. Because the vertex $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ defined by $\mathcal{B}$ is optimal, we have $\hat{x}_i = 0$ by means of Lemma 5.1 (ii). Hence

$$0 = \hat{x}_i = (A_{\mathcal{B}}^{-1}\boldsymbol{b})_p = (A_{\mathcal{B}}^{-1} A \boldsymbol{x}^*)_p = x_i^* + (A_{\mathcal{B}}^{-1} A_{\mathcal{N}} \boldsymbol{x}_{\mathcal{N}}^*)_p = 0 + (A_{\mathcal{B}}^{-1} A_j)_p \, x_j^*,$$

so that $(A_{\mathcal{B}}^{-1} A_j)_p = 0$. □

Addendum

In [70] the author defined a basis $\mathcal{B}$ to be of "correct degeneracy" if its vertex $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ satisfies $x_i^* = 0 \implies \hat{x}_i = 0$ and $z_i^* = 0 \implies \hat{z}_i = 0$ for a strictly complementary solution $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$. It was proved using Lemma 5.6(iii) that a maximum volume/weight basis attains correct degeneracy in the limit $\mu \to 0$. During the writing up of the present work the author realized that any basis of maximum cardinality also has correct degeneracy, so that the latter concept provides no additional characterization of a maximum volume basis.

5.4 Crossover

Some LP applications require vertex solutions. The most prominent example is classical integer programming, where cutting plane generation and reoptimization with the simplex method rely

on an optimal basis. Because the IPM converges to strictly complementary points, the obtained solution is not a vertex if the LP problem is degenerate. IPMs are therefore supplemented by a crossover procedure, which recovers an optimal basis through simplex-type iterations.

Megiddo [52] described a polynomial-time algorithm for constructing an optimal basis from any primal-dual solution. Bixby and Saltzman [10] demonstrated how Megiddo's method can be formulated equivalently as primal and dual "push phases". The push method additionally requires a starting basis as input, which is transformed into an optimal one. This form of the algorithm is commonly used in crossover implementations and is described in the following.

Let $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ be a primal-dual solution to (5.1) (not necessarily strictly complementary) and $\mathcal{B}$ be a basis of maximum cardinality. Recall that variables $j \in \mathcal{N}$ for which $x_j \neq 0$ and $i \in \mathcal{B}$ for which $z_i \neq 0$ are said to be primal superbasic and dual superbasic, respectively, and that their index sets are denoted by $\mathcal{N}^+$ and $\mathcal{B}^+$.

The primal push phase manipulates $\boldsymbol{x}$ and $\mathcal{B}$ to remove primal superbasics:

[P1] If $\mathcal{N}^+ = \emptyset$, then stop. Otherwise choose $j \in \mathcal{N}^+$.

[P2] Let $\Delta \boldsymbol{x}_\mathcal{B} = A_\mathcal{B}^{-1} A_j x_j$ and $0 \leq \alpha \leq 1$ be maximum subject to $\boldsymbol{x}_\mathcal{B} + \alpha \Delta \boldsymbol{x}_\mathcal{B} \geq \boldsymbol{0}$. Set $\boldsymbol{x}_\mathcal{B} = \boldsymbol{x}_\mathcal{B} + \alpha \Delta \boldsymbol{x}_\mathcal{B}$ and $x_j = (1 - \alpha) x_j$. If $\alpha = 1$, then go to [P1].

[P3] Choose $i \in \mathcal{B}$ for which $x_i = 0$ and $\Delta x_i < 0$. (i is a blocking index.)

[P4] Replace i by j in the basis and go to [P1].

The dual push phase manipulates $(\boldsymbol{y}, \boldsymbol{z})$ and $\mathcal{B}$ to remove dual superbasics:

[D1] If $\mathcal{B}^+ = \emptyset$, then stop. Otherwise choose $i \in \mathcal{B}^+$ and let p be the position of i in $\mathcal{B}$.

[D2] Let $\Delta \boldsymbol{y} = A_\mathcal{B}^{-T} \boldsymbol{e}_p z_i$, $\Delta \boldsymbol{z}_\mathcal{N} = -A_\mathcal{N}^T \Delta \boldsymbol{y}$ and $0 \leq \alpha \leq 1$ be maximum subject to $\boldsymbol{z}_\mathcal{N} + \alpha \Delta \boldsymbol{z}_\mathcal{N} \geq \boldsymbol{0}$. Set $\boldsymbol{y} = \boldsymbol{y} + \alpha \Delta \boldsymbol{y}$, $\boldsymbol{z}_\mathcal{N} = \boldsymbol{z}_\mathcal{N} + \alpha \Delta \boldsymbol{z}_\mathcal{N}$ and $z_i = (1 - \alpha) z_i$. If $\alpha = 1$, then go to [D1].

[D3] Choose $j \in \mathcal{N}$ for which $z_j = 0$ and $\Delta z_j < 0$. (j is a blocking index.)

[D4] Replace i by j in the basis and go to [D1].

Lemma 5.8. *Assume that $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ is an optimal solution to (5.1) and $\mathcal{B}$ a basis of maximum cardinality. After executing the primal and dual push phases $\mathcal{B}$ is an optimal basis and $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ its associated vertex.*

Proof. Obviously each push maintains $A\boldsymbol{x} = \boldsymbol{b}$, $A^T \boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c}$ and $(\boldsymbol{x}, \boldsymbol{z}) \geq \boldsymbol{0}$. It is also clear that each pivot operation maintains maximum cardinality of the basis.

We need to show that $\boldsymbol{x}$ and $\boldsymbol{z}$ remain complementary during the push operations. Assume that in the primal phase x_j is pushed toward zero for some $j \in \mathcal{N}^+$ and consider the update to $\boldsymbol{x}_\mathcal{B}$. If $\Delta x_i \neq 0$ for some $i \in \mathcal{B}$, then $(A_\mathcal{B}^{-1} A_j)_{\pi(i)} \neq 0$. Because $\mathcal{B}$ has maximum cardinality, we then must have $i \notin \mathcal{B}^+$ by Lemma 5.3 and it follows from Lemma 5.1(ii) that $z_i = 0$. Hence complementarity of $\boldsymbol{x}$ and $\boldsymbol{z}$ is maintained in [P2]. Complementarity is trivially maintained in [D2] because now $\boldsymbol{x}_\mathcal{N} = \boldsymbol{0}$. The argument is easily adjusted if the dual push phase is executed first.

Because each iteration of the primal and dual push phases reduces the number of superbasic variables by 1, the algorithm terminates after at most $n - m$ primal and m dual pushes with an optimal basis. □

Notice that the number of push operations is determined from the beginning by the number of superbasic variables, whereas the number of pivot operations may depend on the order in

which superbasics are processed. If the initial solution is strictly complementary, then the number of pushes is the number of degeneracies of the LP model.

Implementing algorithms as the one above is prone to numerical instability. The problem is that after choosing an x_j to be moved to zero, there is no flexibility about which variable leaves the basis unless a tie exists in [P3]. If the pivot element corresponding to the first blocking variable is small, the new basis will be ill conditioned and severe round-off errors may occur in subsequent computations. The same issue is well known from the simplex method, where a two-pass ratio test that promotes larger pivot elements at the cost of accepting slight infeasibilities is often used in practice [34]. The same technique can be used for implementing the push method. Here, however, we can also gain more flexibility in the pivot choice by a modification of the algorithm.

Observe that [P4] can be replaced by

[P4'] Let p be the position of i in $\mathcal{B}$. Choose $j' \in \mathcal{N}^+$ such that $(A_{\mathcal{B}}^{-1}A_{j'})_p \neq 0$. Replace i by j' in the basis and go to [P1].

That means, a blocking index $i \in \mathcal{B}$ can be replaced by any $j' \in \mathcal{N}^+$ that keeps the basis nonsingular. Because the number of superbasics still reduces by 1 in each iteration, the proof of Lemma 5.8 remains valid. If the variable j that was pushed in [P1] is not exchanged, it remains superbasic and will be pushed again in a subsequent iteration. The obvious method to improve numerical stability is computing the tableau row corresponding to $i \in \mathcal{B}$ and choosing the maximum absolute entry among all primal superbasics as pivot.

The analogue for the dual push phase is to replace [D4] by

[D4'] Choose $i' \in \mathcal{B}^+$ such that $(A_{\mathcal{B}}^{-1}A_j)_{p'} \neq 0$, where p' is the position of i' in $\mathcal{B}$. Replace i' by j in the basis and go to [D1].

Here we may compute the tableau column corresponding to $j \in \mathcal{N}$ and choose the pivot as the maximum absolute entry among all dual superbasics.

The cost for the modified push method is computing an additional tableau row or column prior to a pivot operation. If the method is used for numerical stability, the extra cost can be reduced by applying the technique only when the first choice of pivot element (i. e. the pivot in [P4] or [D4]) is smaller than a threshold. The flexibility in the pivot choice can also be used to keep the basis matrix as sparse as possible.

A crossover implementation is described in Section 6.8. In practice it must be taken into account that the solution from an interior point method is neither exactly feasible nor complementary, and that linear dependency relies on numerical tolerances. This does not affect the number of push operations, however, so that the complexity of the algorithm is maintained.

5.5 Fixing Primal and Dual Variables

To reduce the problem dimension and computation time, it seems appealing to drop columns from A during an interior point solve as soon as primal variables are indicated to be at a bound in the solution. The downside of the idea is that we would no longer obtain a dual solution to the original problem. To illustrate, consider

$$\begin{aligned} \text{minimize} \quad & x_1 + x_2 - c_3 x_3 \\ \text{subject to} \quad & x_1 + x_2 + x_3 = 1, \\ & x_1 - x_4 = 1, \\ & \boldsymbol{x} \geq \mathbf{0}, \end{aligned}$$

where c_3 is a parameter. The primal solution is unique with $\boldsymbol{x} = (1, 0, 0, 0)$. After dropping x_3, the reduced problem has the (unbounded) dual solution set

$$y_1 \leq 1, \quad y_2 = 1 - y_1, \quad z_1 = 0, \quad z_2 = z_4 = y_2.$$

For any dual solution of the reduced problem, z_3 can be computed for the original problem from $z_3 = -c_3 - y_1$. Hence a dual solution of the reduced problem does not yield a dual solution of the original problem when $c_3 > -y_1$. Because c_3 is an arbitrary parameter which does not affect the solution computed for the reduced problem, the example shows that in general we do not obtain a dual solution after dropping variables.

Variables can be removed from the optimization process in a different manner, however, founded by Lemmas 5.9 and 5.10.

Lemma 5.9. *Let $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ be a strictly complementary solution to* (5.1) *and let $\mathcal{B}$ be a basis of maximum cardinality. After fixing the primal superbasic variables $\mathcal{N}^+ \subseteq \mathcal{N}$ at $\boldsymbol{x}^*_{\mathcal{N}^+}$ and solving the reduced problem, a solution to* (5.1) *is obtained by setting $\boldsymbol{z}_{\mathcal{N}^+} = \boldsymbol{0}$.*

Proof. Let $\mathcal{R} = \{1, \ldots, n\} \setminus \mathcal{N}^+$ be the indices of variables in the reduced problem and let $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ be the primal-dual point obtained from the stated procedure. Obviously $A\boldsymbol{x} = \boldsymbol{b}$, $A^T_{\mathcal{R}}\boldsymbol{y} + \boldsymbol{z}_{\mathcal{R}} = \boldsymbol{c}_{\mathcal{R}}$, $\boldsymbol{x} \geq \boldsymbol{0}$, $\boldsymbol{z} \geq \boldsymbol{0}$ and $\boldsymbol{x}^T\boldsymbol{z} = 0$. For $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ to be a solution to (5.1), we need to prove that $A^T_{\mathcal{N}^+}\boldsymbol{y} = \boldsymbol{c}_{\mathcal{N}^+}$.

Let $j \in \mathcal{N}^+$ and $\boldsymbol{v} = A_{\mathcal{B}}^{-1}A_j$. We claim that for each $i \in \mathcal{B}$ either $z_i = 0$ or $v_{\pi(i)} = 0$; for if $z_i > 0$, then $z^*_i > 0$ by Lemma 5.1(ii), so that $v_{\pi(i)} = 0$ by Lemma 5.3. Consequently $\boldsymbol{v}^T\boldsymbol{z}_{\mathcal{B}} = 0$ and therefore

$$A_j^T\boldsymbol{y} = A_j^T A_{\mathcal{B}}^{-T}(\boldsymbol{c}_{\mathcal{B}} - \boldsymbol{z}_{\mathcal{B}}) = \boldsymbol{v}^T(\boldsymbol{c}_{\mathcal{B}} - \boldsymbol{z}_{\mathcal{B}}) = \boldsymbol{v}^T\boldsymbol{c}_{\mathcal{B}} = A_j^T A_{\mathcal{B}}^{-T}\boldsymbol{c}_{\mathcal{B}}.$$

The same argument holds for $(\boldsymbol{y}^*, \boldsymbol{z}^*)$ in place of $(\boldsymbol{y}, \boldsymbol{z})$. Hence $A_j^T\boldsymbol{y} = A_j^T A_{\mathcal{B}}^{-T}\boldsymbol{c}_{\mathcal{B}} = A_j^T\boldsymbol{y}^*$. Because $j \in \mathcal{N}^+$ we have $z^*_j = 0$ so that $A_j^T\boldsymbol{y} = A_j^T\boldsymbol{y}^* = c_j$. □

Lemma 5.10. *Let $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ be a strictly complementary solution to* (5.1) *and let $\mathcal{B}$ be a basis of maximum cardinality. Define a modified LP problem by reducing $\boldsymbol{c}_{\mathcal{B}^+}$ to $\boldsymbol{c}_{\mathcal{B}^+} - \boldsymbol{z}^*_{\mathcal{B}^+}$ for dual superbasic variables $\mathcal{B}^+ \subseteq \mathcal{B}$ and removing the lower bound on $\boldsymbol{x}_{\mathcal{B}^+}$. From a solution $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ to the modified problem, a solution to* (5.1) *is obtained by setting $\boldsymbol{z}_{\mathcal{B}^+} = \boldsymbol{z}^*_{\mathcal{B}^+}$.*

Proof. Let $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ be the suggested solution to (5.1) obtained from the stated procedure. Obviously $A\boldsymbol{x} = \boldsymbol{b}$, $A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c}$, $x_j \geq 0$ for $j \notin \mathcal{B}^+$ and $\boldsymbol{z} \geq \boldsymbol{0}$. It remains to show that $\boldsymbol{x}_{\mathcal{B}^+} = \boldsymbol{0}$, which implies that $\boldsymbol{x}^T\boldsymbol{z} = 0$.

Let $i = \mathcal{B}_p \in \mathcal{B}^+$. Because $\mathcal{B}$ has maximum cardinality, for any $j \in \mathcal{N}$ such that $x^*_j > 0$ we have $(A_{\mathcal{B}}^{-1}A_j)_p = 0$. Observe that $x_j > 0 \implies x^*_j > 0$ for $j \notin \mathcal{B}^+$ because $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ yields a strictly complementary solution to the modified problem by setting $\boldsymbol{z}^*_{\mathcal{B}^+} = \boldsymbol{0}$. (Free variables do not contribute to the notion of strict complementarity.) It follows that

$$x_i = (A_{\mathcal{B}}^{-1}(\boldsymbol{b} - A_{\mathcal{N}}\boldsymbol{x}_{\mathcal{N}}))_p = (A_{\mathcal{B}}^{-1}\boldsymbol{b})_p.$$

The same argument holds for $\boldsymbol{x}^*$ in place of $\boldsymbol{x}$. Therefore $x_i = x^*_i$ and because $i \in \mathcal{B}^+$, $x_i = x^*_i = 0$. □

The gist of Lemma 5.9 is that primal superbasic variables can be fixed at any value in the interior of the solution set and be removed from the optimization. The gist of Lemma 5.10 is that in the same way dual superbasic variables can be fixed by decreasing the objective coefficients and removing the corresponding bounds from $\boldsymbol{x}$. In both cases a primal-dual solution can be recovered for the original problem.

Applying the lemmas in practice requires an optimal solution value for the variables to be fixed. Obtaining such a value is not as unrealistic as it seems. When the basis has maximum cardinality, the optimal solution for primal and dual superbasic variables is not unique. Let $[\underline{x}_j, \overline{x}_j]$ be the range of optimal solution values for $j \in \mathcal{N}^+$; i. e. for each $\xi \in [\underline{x}_j, \overline{x}_j]$ there exists an optimal LP solution with $x_j = \xi$. Denote $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ the limit point of the central path. Because $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ is the analytic center of the optimal face, we have $\underline{x}_j < x_j^* < \overline{x}_j$. Hence there exists a positive μ^* such that x_j is inside the optimal solution set for all central points with complementarity measure $\leq \mu^*$. For the IPM it means that sufficiently late iterates provide values at which superbasic variables can be fixed. Of course, there is no criterion to guarantee "sufficiently late" in practice.

The technique has proved to be viable for resolving two computational issues. The first problem were round-off errors in the linear solver if nonbasic scaling factors became large or basic scaling factors became small. This situation corresponds precisely to the presence of superbasic variables in the solution. The second problem was caused by unbounded solution sets, in which case the IPM tends to push $\|\boldsymbol{x}\|$ or $\|(\boldsymbol{y}, \boldsymbol{z})\|$ to infinity. An unbounded primal solution requires some x_j for $j \in \mathcal{N}$ to become large, because otherwise $\|\boldsymbol{x}_\mathcal{B}\| = \left\|A_\mathcal{B}^{-1}(\boldsymbol{b} - A_\mathcal{N}\boldsymbol{x}_\mathcal{N})\right\|$ would be bounded as soon as the iterates approach feasibility. Similarly, an unbounded dual solution requires z_j to become large for some $j \in \mathcal{B}$. Both problems could be solved by dynamically fixing variables at their current values. Details of the implementation are given in Section 6.7.

6 Implementation of the Interior Point Solver

This chapter[3] describes the implementation of a linear programming interior point solver based on the ideas discussed so far. The main features of the solver are the following: the IPM uses iterative linear algebra with basis preconditioning, where the basis is updated from one interior point iteration to the next to maintain a large volume; for the early IPM iterations a less sophisticated (and less expensive) preconditioner is used; at the switch to basis preconditioning a starting basis is constructed by a "crash" procedure; degenerate variables are eliminated during the optimization process to improve numerical stability and to allow running the IPM to high accuracy; the final basis from the preconditioner serves as starting basis for crossover.

The resulting software package is called IPX and comprises about 10,000 lines of executable C++ code. The package does not implement a sparse *LU* factorization for the basis matrices by itself. Instead it can interface any code that provides the well-known simplex operations INVERT, FTRAN, BTRAN and UPDATE. By default the author's BASICLU package is used, which is presented in Chapter 7.

The following sections describe the algorithmic components of IPX from a high-level perspective. Data structures and coding details are discussed only if they are novel and non-obvious. A test set of 165 LP models is used to present statistics and to illustrate the effect of algorithmic decisions. The set was composed for benchmarking (see Chapter 8) and represents a wide range of medium to large-scale problems. The models and their solution times are given in Appendix A.

IPX internally stores the LP model in computational form of a primal-dual pair,

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{x}^l,\boldsymbol{x}^u}{\text{minimize}} \quad & \boldsymbol{c}^T\boldsymbol{x} && && \text{(6.1a)}\\ \text{subject to} \quad & A\boldsymbol{x} &&= \boldsymbol{b},\\ & \boldsymbol{x}-\boldsymbol{x}^l &&= \boldsymbol{l},\\ & \boldsymbol{x}+\boldsymbol{x}^u &&= \boldsymbol{u},\\ & \boldsymbol{x}^l,\boldsymbol{x}^u &&\geq \boldsymbol{0}, \end{aligned}$$

and

$$\begin{aligned} \underset{\boldsymbol{y},\boldsymbol{z}^l,\boldsymbol{z}^u}{\text{maximize}} \quad & \boldsymbol{b}^T\boldsymbol{y}+\boldsymbol{l}^T\boldsymbol{z}^l-\boldsymbol{u}^T\boldsymbol{z}^u && && \text{(6.1b)}\\ \text{subject to} \quad & A^T\boldsymbol{y}+\boldsymbol{z}^l-\boldsymbol{z}^u &&= \boldsymbol{c},\\ & \boldsymbol{z}^l,\boldsymbol{z}^u &&\geq \boldsymbol{0}, \end{aligned}$$

where A is an $m\times(n+m)$ matrix whose rightmost m columns form the identity matrix. The first n and the last m components of $\boldsymbol{x}$ are termed "structural" and "slack" variables, respectively, although the objective coefficients of slack variables do not need to be zero. Entries of $-\boldsymbol{l}$ and $\boldsymbol{u}$ are allowed to be infinity, in which case the corresponding dual variable is fixed at zero and its term is dropped from the dual objective. We define the index sets

$$\mathcal{L}=\{j \mid l_j>-\infty\},\quad \mathcal{U}=\{j \mid u_j<\infty\}.$$

A variable x_j is termed "free" if it has no finite bound and "fixed" if its lower and upper bounds are equal. It is assumed that only structural variables can be free and only slack variables can be fixed (otherwise a column and/or row of A can be trivially eliminated). Although fixed slack variables could be eliminated as well, we keep them as variables in the problem formulation until the switch to basis preconditioning (Section 6.6). This guarantees that A has full row rank and, as the following presentation will show, makes the implementation of several parts of the LP solver more convenient.

[3] Parts of the material were published by the author in the form of a technical report [72].

6.1 Preprocessing

IPX accepts user input in the form

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \bar{\boldsymbol{c}}^T \boldsymbol{x} \\ \text{subject to} \quad & \bar{A}\boldsymbol{x} \,\{=, \leq, \geq\}\, \bar{\boldsymbol{b}}, \\ & \bar{\boldsymbol{l}} \leq \boldsymbol{x} \leq \bar{\boldsymbol{u}}, \end{aligned}$$

where $\bar{A}$ is an $\bar{m} \times \bar{n}$ matrix. By default a row and column scaling is applied to the input data. If $\bar{A}$ contains entries of a wide range of magnitudes, scaling leads to more flexibility for pivoting in the LU factorization of basis matrices. It also reduces the condition number of $\begin{bmatrix}\bar{A} & I\end{bmatrix}$, which is decisive for the convergence rate of the linear solver in early interior point iterations.

A variant of the recursive matrix equilibration from Knight et al. [41] has shown to be effective on badly scaled models. Their algorithm iteratively scales the rows and columns of a matrix by the reciprocal of the square root of their infinity norm. It was proved in [41] that the scheme converges to an equilibrated matrix (i. e. each row and column attains infinity norm 1 in the limit). In IPX the method is applied to $\bar{A}$. In each scaling iteration the scaling factors are truncated to powers of 2, so that no round-off errors occur. The algorithm is terminated when the infinity norm of each row and column is contained in the interval $[0.5, 8)$. On many well-posed models from our test set no scaling operations were required, but for some badly scaled models equilibration was necessary to avoid numerical trouble in the linear algebra.

The user model is then transformed into computational form with optional dualization. If the input becomes the primal problem, then $A = \begin{bmatrix}\bar{A} & I_{\bar{m}}\end{bmatrix}$; and if it becomes the dual problem, then $A = \begin{bmatrix}\bar{A}^T & (-I_{\bar{n}})_{\mathcal{J}} & I_{\bar{n}}\end{bmatrix}$, where $\mathcal{J}$ is the set of variables with finite lower and upper bounds. The reason for dualization is that the linear algebra implementation is optimized for matrices A with (many) more structural columns than rows. If the user does not make an explicit choice, the model is dualized if $\bar{m} > 2\bar{n}$.

6.2 Interior Point Algorithm

At an iterate $(\boldsymbol{x}, \boldsymbol{x}^l, \boldsymbol{x}^u, \boldsymbol{y}, \boldsymbol{z}^l, \boldsymbol{z}^u)$ of the computational form LP model, the primal and dual residuals are

$$\boldsymbol{r}^b = \boldsymbol{b} - A\boldsymbol{x}, \tag{6.2a}$$

$$\boldsymbol{r}^l = \boldsymbol{l} - \boldsymbol{x} + \boldsymbol{x}^l, \tag{6.2b}$$

$$\boldsymbol{r}^u = \boldsymbol{u} - \boldsymbol{x} - \boldsymbol{x}^u, \tag{6.2c}$$

$$\boldsymbol{r}^c = \boldsymbol{c} - A^T\boldsymbol{y} - \boldsymbol{z}^l + \boldsymbol{z}^u, \tag{6.2d}$$

and the complementarity measure is

$$\mu = \frac{1}{|\mathcal{L}| + |\mathcal{U}|} \left(\sum_{j \in \mathcal{L}} x_j^l z_j^l + \sum_{j \in \mathcal{U}} x_j^u z_j^u \right). \tag{6.3}$$

The Newton step toward the central point with complementarity measure $\sigma\mu$ for some $\sigma \in [0, 1]$ is defined by the linear system

$$\begin{bmatrix} & A^T & & & I & -I \\ A & & & & & \\ I & & -I & & & \\ I & & & I & & \\ & & Z^l & & X^l & \\ & & & Z^u & & X^u \end{bmatrix} \begin{pmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{y} \\ \Delta\boldsymbol{x}^l \\ \Delta\boldsymbol{x}^u \\ \Delta\boldsymbol{z}^l \\ \Delta\boldsymbol{z}^u \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}^c \\ \boldsymbol{r}^b \\ \boldsymbol{r}^l \\ \boldsymbol{r}^u \\ \boldsymbol{s}^l \\ \boldsymbol{s}^u \end{pmatrix} \tag{6.4}$$

with $\boldsymbol{s}^l = -X^l Z^l \boldsymbol{e} + \sigma\mu\boldsymbol{e}$ and $\boldsymbol{s}^u = -X^u Z^u \boldsymbol{e} + \sigma\mu\boldsymbol{e}$.

Typically IPMs solve (6.4) for multiple right-hand sides at each iteration to compose a step direction that makes better progress in reducing μ and/or keeps the iterates closer to the central path. IPX implements a version of Mehrotra's method [53], which composes the step from a predictor and a corrector direction. Mehrotra's method is widely considered to be the most efficient method using two linear system solves per iteration. For implementations based on Cholesky factorization it is common practice to compute more than one corrector direction to reduce the iteration count of the IPM (see Gondzio [24]). This technique is not used in IPX because iterative linear algebra does not offer large savings when solving multiple systems with the same matrix. We have also not tried to implement an IPM with only one linear system solve per iteration. The issue with a single system solve is not only that the step direction makes less progress in reducing μ, but also that established rules for choosing the centering parameter (σ in Algorithm 7) require the "affine scaling" direction obtained for $\boldsymbol{s}^l = -X^l Z^l \boldsymbol{e}$, $\boldsymbol{s}^u = -X^u Z^u \boldsymbol{e}$.

IPX's interior point scheme is outlined in Algorithm 7. The computation of the starting point and the heuristics for choosing the centering parameter σ and the step sizes are those from Mehrotra [53] with the obvious modifications to accommodate lower and upper bounds on the variables. Computing the starting point requires solving (6.4) for two right-hand sides.

The usual requirement in linear programming is to obtain 8-digit accuracy in the objective, which leads to the termination criterion

$$|f_p - f_d| \le 10^{-8}\,(1 + 0.5|f_p + f_d|)\,, \tag{6.5}$$

where f_p and f_d are the primal and dual objective value. By the choice of the starting point and the behaviour of the algorithm, primal and dual feasibility are practically always satisfied when the objective criterion is reached.

After eliminating $\Delta\boldsymbol{x}^l$, $\Delta\boldsymbol{x}^u$, $\Delta\boldsymbol{z}^l$ and $\Delta\boldsymbol{z}^u$ from (6.4), the linear system attains KKT form

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{pmatrix} \Delta\boldsymbol{x} \\ -\Delta\boldsymbol{y} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}^a \\ \boldsymbol{r}^b \end{pmatrix}, \tag{6.6}$$

where $\boldsymbol{r}^a = -\boldsymbol{r}^c + (X^l)^{-1}(\boldsymbol{s}^l + Z^l\boldsymbol{r}^l) - (X^u)^{-1}(\boldsymbol{s}^u - Z^u\boldsymbol{r}^u)$ and G is the diagonal matrix with entries $g_j = z_j^l/x_j^l + z_j^u/x_j^u$. Notice that g_j is zero if x_j is a free variable. For the LP model with lower and upper bounds on the variables, the scaling matrix D and an indicator matrix H are defined by

$$d_j = \begin{cases} g_j^{-1/2} & \text{if } g_j \neq 0, \\ 1 & \text{if } g_j = 0, \end{cases} \qquad h_j = \begin{cases} 1 & \text{if } g_j \neq 0, \\ 0 & \text{if } g_j = 0, \end{cases}$$

so that (6.6) can be scaled to

$$\begin{bmatrix} H & DA^T \\ AD & 0 \end{bmatrix} \begin{pmatrix} D^{-1}\Delta\boldsymbol{x} \\ -\Delta\boldsymbol{y} \end{pmatrix} = \begin{pmatrix} D\boldsymbol{r}^a \\ \boldsymbol{r}^b \end{pmatrix}. \tag{6.7}$$

The iterative methods implemented in IPX compute an approximate solution to (6.7) of the form

$$\begin{bmatrix} H & DA^T \\ AD & 0 \end{bmatrix} \begin{pmatrix} D^{-1}\Delta\boldsymbol{x} \\ -\Delta\boldsymbol{y} \end{pmatrix} = \begin{pmatrix} D\boldsymbol{r}^a \\ \boldsymbol{r}^b \end{pmatrix} + \begin{pmatrix} \boldsymbol{\delta} \\ \boldsymbol{0} \end{pmatrix}, \tag{6.8}$$

where $\boldsymbol{\delta}$ is the residual vector. As termination criterion for the linear solver it is required that

$$\|\boldsymbol{\delta}\|_\infty \le \tau := \varepsilon_\tau \sqrt{\mu}, \tag{6.9}$$

where ε_τ is a problem independent parameter. The criterion is motivated by the theory from Chapter 4 because it maintains boundedness of $\left\|D^{-1}\Delta\boldsymbol{x}\right\|$ in terms of $\sqrt{\mu}$, which is key to

Algorithm 7 Interior Point Algorithm

1: Compute starting point $\left(\boldsymbol{x}, \boldsymbol{x}^l, \boldsymbol{x}^u, \boldsymbol{y}, \boldsymbol{z}^l, \boldsymbol{z}^u\right)^0$.
2: **for** $k = 0, 1, \ldots$ **do**
3: Compute $\boldsymbol{r}^b$, $\boldsymbol{r}^l$, $\boldsymbol{r}^u$, $\boldsymbol{r}^c$ from (6.2) and μ from (6.3).
4: Stop if a termination criterion is satisfied.
5: Fix degenerate variables for which $\min(x_j^l, x_j^u)$ or $\max(z_j^l, z_j^u)$ is close to zero.
6: Compute preconditioner.
7: Set

$$\boldsymbol{s}^l = -X^l Z^l \boldsymbol{e}, \quad \boldsymbol{s}^u = -X^u Z^u \boldsymbol{e}$$

and solve (6.4) for the solution $\left(\Delta\boldsymbol{x}_p, \Delta\boldsymbol{y}_p, \Delta\boldsymbol{x}_p^l, \Delta\boldsymbol{x}_p^u, \Delta\boldsymbol{z}_p^l, \Delta\boldsymbol{z}_p^u\right)$.
8: Choose centering parameter $0 \le \sigma \le 1$.
9: Set

$$\begin{aligned}\boldsymbol{s}^l &= \sigma\mu\boldsymbol{e} - X^l Z^l \boldsymbol{e} - \Delta X_p^l \Delta Z_p^l \boldsymbol{e},\\ \boldsymbol{s}^u &= \sigma\mu\boldsymbol{e} - X^u Z^u \boldsymbol{e} - \Delta X_p^u \Delta Z_p^u \boldsymbol{e}\end{aligned}$$

and solve (6.4) for the solution $\left(\Delta\boldsymbol{x}, \Delta\boldsymbol{y}, \Delta\boldsymbol{x}^l, \Delta\boldsymbol{x}^u, \Delta\boldsymbol{z}^l, \Delta\boldsymbol{z}^u\right)$.
10: Choose step sizes $0 \le \alpha^{\text{primal}} \le 1$ and $0 \le \alpha^{\text{dual}} \le 1$.
11: Make step

$$\begin{aligned}\left(\boldsymbol{x}, \boldsymbol{x}^l, \boldsymbol{x}^u\right)^{k+1} &= \left(\boldsymbol{x}, \boldsymbol{x}^l, \boldsymbol{x}^u\right)^k + \alpha^{\text{primal}} \left(\Delta\boldsymbol{x}, \Delta\boldsymbol{x}^l, \Delta\boldsymbol{x}^u\right),\\ \left(\boldsymbol{y}, \boldsymbol{z}^l, \boldsymbol{z}^u\right)^{k+1} &= \left(\boldsymbol{y}, \boldsymbol{z}^l, \boldsymbol{z}^u\right)^k + \alpha^{\text{dual}} \left(\Delta\boldsymbol{y}, \Delta\boldsymbol{z}^l, \Delta\boldsymbol{z}^u\right).\end{aligned}$$

the convergence of the IPM. Choosing a smaller tolerance increases the computational cost per linear system but reduces the number of interior point iterations as the step directions become more accurate. The effect on the total computation time is quite moderate, however. On our test set varying ε_τ between its default value 0.3 and within the range $[0.05, 0.5]$ changed the geometric mean of the IPX runtimes by 8%.

Given an approximate solution to the KKT system, the order in which the remaining step components are recovered determines to which of the block equations in (6.4) the residual propagates. To satisfy the dual feasibility equations exactly, IPX sets

$$\begin{aligned}\Delta\boldsymbol{x}^l &= -\boldsymbol{r}^l + \Delta\boldsymbol{x},\\ \Delta\boldsymbol{x}^u &= \quad \boldsymbol{r}^u - \Delta\boldsymbol{x},\\ \Delta z_j^l &= (s_j^l - z_j^l \Delta x_j^l)/x_j^l && \text{for all } j \text{ for which } z_j^l/x_j^l < z_j^u/x_j^u,\\ \Delta z_j^u &= (s_j^u - z_j^u \Delta x_j^u)/x_j^u && \text{for all } j \text{ for which } z_j^l/x_j^l \ge z_j^u/x_j^u,\\ \Delta z_j^l &= \quad r_j^c - (A^T\Delta\boldsymbol{y})_j + \Delta z_j^u && \text{for all } j \text{ for which } z_j^l/x_j^l \ge z_j^u/x_j^u,\\ \Delta z_j^u &= -r_j^c + (A^T\Delta\boldsymbol{y})_j + \Delta z_j^l && \text{for all } j \text{ for which } z_j^l/x_j^l < z_j^u/x_j^u.\end{aligned}$$

By this choice the first four block equations in (6.4) are solved exactly and the residual is shifted between the last two blocks. If a variable has two finite bounds and is active at its lower bound in the solution, the residual eventually occurs in the equation $z_j^l \Delta x_j^l + x_j^l \Delta z_j^l = s_j^l$. Because in this case $x_j^l \to 0$ but $x_j^u \to (x_j^u)^* > 0$, adjusting Δz_j^l to satisfy dual feasibility causes a smaller residual in the right-hand side to (6.4) than adjusting Δz_j^u.

6.3 Initial Iterations

At the early interior point iterations the KKT matrix is usually well conditioned by itself and a simple and inexpensive preconditioner suffices to achieve fast convergence of the linear solver. This is exploited by performing a few "initial iterations" of the IPM before starting basis preconditioning.

The KKT matrix in (6.6) is nonsingular if and only if A has full row rank and the columns of A corresponding to zeros on the diagonal of G are linearly independent. While the first requirement is satisfied after adding slack variables to all constraints, the latter requirement might not be. To overcome the problem at the beginning, zeros on the diagonal of G are replaced by the regularization value $\min\{\{g_j \,|\, g_j \neq 0\}, \mu\}$, where μ is given by (6.3). Regularization can significantly change the solution to the linear system, but using it in the initial iterations has shown to be unproblematic for the convergence of the IPM. The regularization value is chosen to yield a well conditioned KKT matrix as long as the remaining diagonal entries of G are balanced and μ is sufficiently large. Including μ in the definition guarantees that regularization is eventually reduced to zero even if none of the remaining g_j becomes small (i. e. if all non-free variables are at a bound in the solution).

Regularization also ensures that the KKT system can always be reduced to normal equations

$$AG^{-1}A^T\Delta\boldsymbol{y} = \boldsymbol{r}^b - AG^{-1}\boldsymbol{r}^a =: \boldsymbol{r}. \tag{6.10}$$

From an approximate solution for $\Delta\boldsymbol{y}$ the solution components $\Delta\boldsymbol{x}$ to the KKT system are recovered from

$$\Delta\boldsymbol{x}_{\mathcal{N}} = G_{\mathcal{N}}^{-1}(\boldsymbol{r}^a + A^T\Delta\boldsymbol{y})_{\mathcal{N}}, \quad \Delta\boldsymbol{x}_{\mathcal{B}} = \boldsymbol{r}^b - A_{\mathcal{N}}\Delta\boldsymbol{x}_{\mathcal{N}},$$

where G is the regularized matrix and $\mathcal{B} = \{n+1, \ldots, n+m\}$ is the slack basis. By definition of $\Delta\boldsymbol{x}$, the solution satisfies (6.8) with $\boldsymbol{\delta}_{\mathcal{N}} = \boldsymbol{0}$. Using that $H = I_{n+m}$, $A_{\mathcal{B}} = I_m$ and $G = D^{-2}$, we obtain

$$\begin{aligned}
\boldsymbol{\delta}_{\mathcal{B}} &= D_{\mathcal{B}}^{-1}\Delta\boldsymbol{x}_{\mathcal{B}} - D_{\mathcal{B}}A_{\mathcal{B}}^T\Delta\boldsymbol{y} - D_{\mathcal{B}}\boldsymbol{r}_{\mathcal{B}}^a \\
&= D_{\mathcal{B}}^{-1}(\boldsymbol{r}^b - A_{\mathcal{N}}\Delta\boldsymbol{x}_{\mathcal{N}}) - D_{\mathcal{B}}A_{\mathcal{B}}^T\Delta\boldsymbol{y} - D_{\mathcal{B}}\boldsymbol{r}_{\mathcal{B}}^a \\
&= D_{\mathcal{B}}^{-1}(\boldsymbol{r}^b - A_{\mathcal{N}}G_{\mathcal{N}}^{-1}(\boldsymbol{r}_{\mathcal{N}}^a + A_{\mathcal{N}}^T\Delta\boldsymbol{y})) - D_{\mathcal{B}}A_{\mathcal{B}}^T\Delta\boldsymbol{y} - D_{\mathcal{B}}\boldsymbol{r}_{\mathcal{B}}^a \\
&= D_{\mathcal{B}}^{-1}\left(\boldsymbol{r}^b - A_{\mathcal{N}}G_{\mathcal{N}}^{-1}\boldsymbol{r}_{\mathcal{N}}^a - A_{\mathcal{N}}G_{\mathcal{N}}^{-1}A_{\mathcal{N}}^T\Delta\boldsymbol{y} - D_{\mathcal{B}}^2\Delta\boldsymbol{y} - D_{\mathcal{B}}^2\boldsymbol{r}_{\mathcal{B}}^a\right) \\
&= D_{\mathcal{B}}^{-1}\left(\boldsymbol{r}^b - AG^{-1}\boldsymbol{r}^a - AG^{-1}A^T\Delta\boldsymbol{y}\right).
\end{aligned}$$

Therefore the accuracy requirement (6.9) is satisfied if

$$\left\| G_{\mathcal{B}}^{1/2}(\boldsymbol{r} - AG^{-1}A^T\Delta\boldsymbol{y}) \right\|_\infty \leq \tau.$$

IPX uses the Conjugate Residual (CR) method [68, Section 6.8] for solving positive definite linear systems. The CR method is a Krylov subspace method, originally proposed by Luenberger [50] for the symmetric indefinite case, whose iterates minimize the 2-norm of the residual. Algorithm 8 states the CR method for solving $C\boldsymbol{y} = \boldsymbol{r}$ while incorporating a preconditioner given by a symmetric positive definite matrix M. The iterates $\boldsymbol{y}$ of the algorithm are related to the iterates $\boldsymbol{u}$ generated by the unpreconditioned CR method applied to

$$M^{-1/2}CM^{-1/2}\boldsymbol{u} = M^{-1/2}\boldsymbol{r} \tag{6.11}$$

by $\boldsymbol{y} = M^{-1/2}\boldsymbol{u}$. Consequently the residual $\boldsymbol{\rho}$ and the preconditioned residual $\boldsymbol{\pi} = M^{-1}\boldsymbol{\rho}$ in Algorithm 8 satisfy

$$\boldsymbol{\pi}^T\boldsymbol{\rho} = \left\| M^{-1/2}(\boldsymbol{r} - C\boldsymbol{y}) \right\|^2,$$

and this quantity is minimized by $\boldsymbol{y}$ over a Krylov subspace.

It turned out that the update to $\boldsymbol{\pi}$ in line 13 might not be sufficiently accurate when M is ill conditioned. It happened that $\|\boldsymbol{\pi}\|$ converged to zero while $\|\boldsymbol{\rho}\|$ stagnated. To safeguard the implementation, $\boldsymbol{\pi}$ is recomputed from $\boldsymbol{\pi} = M^{-1}\boldsymbol{\rho}$ every 5 iterations at the cost of an extra operation with M^{-1}. The issue does not arise with basis preconditioning, where a different implementation of the CR method will be used.

Algorithm 8 Preconditioned CR Method

Input: $C, M \in \mathbb{R}^{m\times m}$ symmetric positive definite, $\boldsymbol{r} \in \mathbb{R}^m$
Output: approximate solution to $C\boldsymbol{y} = \boldsymbol{r}$

1: $\boldsymbol{y} = \boldsymbol{0}$ // or an initial guess
2: $\boldsymbol{\rho} = \boldsymbol{r} - C\boldsymbol{y}$ // residual
3: $\boldsymbol{\pi} = M^{-1}\boldsymbol{\rho}$ // preconditioned residual
4: $\boldsymbol{\omega} = C\boldsymbol{\pi}$
5: $\Delta\boldsymbol{y} = \boldsymbol{\pi}$ // step direction
6: $\Delta\boldsymbol{\rho} = \boldsymbol{\omega}$ // update to negative residual
7: $\gamma = \boldsymbol{\pi}^T\boldsymbol{\omega}$
8: **repeat**
9: $\quad\Delta\boldsymbol{\pi} = M^{-1}\Delta\boldsymbol{\rho}$ // can be stored in $\boldsymbol{\omega}$
10: $\quad\alpha = \gamma/\Delta\boldsymbol{\pi}^T\Delta\boldsymbol{\rho}$
11: $\quad\boldsymbol{y} = \boldsymbol{y} + \alpha\Delta\boldsymbol{y}$
12: $\quad\boldsymbol{\rho} = \boldsymbol{\rho} - \alpha\Delta\boldsymbol{\rho}$
13: $\quad\boldsymbol{\pi} = \boldsymbol{\pi} - \alpha\Delta\boldsymbol{\pi}$
14: $\quad\boldsymbol{\omega} = C\boldsymbol{\pi}$ // restores $\boldsymbol{\omega}$
15: $\quad\gamma_{\text{new}} = \boldsymbol{\pi}^T\boldsymbol{\omega}$
16: $\quad\beta = \gamma_{\text{new}}/\gamma$
17: $\quad\Delta\boldsymbol{y} = \boldsymbol{\pi} + \beta\Delta\boldsymbol{y}$
18: $\quad\Delta\boldsymbol{\rho} = \boldsymbol{\omega} + \beta\Delta\boldsymbol{\rho}$
19: $\quad\gamma = \gamma_{\text{new}}$
20: **until** a termination criterion is satisfied

In an early version of the code the initial IPM iterations were performed with diagonal preconditioning, where M is obtained by dropping all off-diagonal entries from $AG^{-1}A^T$. The method has been refined for matrices A with "dense" columns; i. e. columns whose nonzero count is much higher than the average. Let A_s and A_d be the sparse and dense part of A, respectively, and let G_s and G_d be the corresponding diagonal blocks of G. The preconditioner used in IPX is

$$M = \operatorname{diag}(A_sG_s^{-1}A_s^T) + A_dG_d^{-1}A_d^T =: M_s + A_dG_d^{-1}A_d^T.$$

By treating the second summand as a low-rank update, an inverse representation of M can be derived from the Sherman-Morrison-Woodbury formula as

$$M^{-1} = M_s^{-1} - M_s^{-1}A_d(G_d + A_d^TM_s^{-1}A_d)^{-1}A_d^TM_s^{-1}.$$

Computing the representation requires a Cholesky factorization of a dense matrix of dimension equal to the number of columns in A_d. Applying M^{-1} requires two multiplications with the diagonal matrix M_s^{-1}, matrix-vector products with A_d and A_d^T, and a forward and backward solve with the dense Cholesky factor. The splitting into A_s and A_d is made by placing the maximum number of columns into A_d such that each column in A_d contains more than 40 nonzeros and more than 10 times the number of nonzeros of any column in A_s; if this yields more than 1500 columns in A_d, no columns are treated as dense.

The performance of the "augmented" to the pure diagonal preconditioner is compared in Table 5 on the 34 problems from our test set for which A_d was not vacuous. For each preconditioner the IPM was run for the same number of iterations ("ipiter"); the number was chosen as the maximum that could be performed with both methods before switching to basis preconditioning (see below). On 29 problems the augmented preconditioner led to fewer CR iterations, and on 26 problems it also reduced the computation time. The geometric mean of the ratios of CR iterations was 0.35 and the mean of the runtime ratios was 0.47, meaning that the augmented method was a factor 2 faster on average. In those cases where it was relevantly slower (30% for problem `in` and almost a factor 2 for `ns2118727` and `ns1687037`), the iteration counts were comparable but a large number of dense columns made the computations expensive. From the results on these 34 problems, the maximum number of columns treated as dense by IPX was adjusted to 1000.

Other than applying M^{-1}, the major cost per CR iteration is computing the matrix-vector product $\boldsymbol{\omega} = AG^{-1}A^T\boldsymbol{\pi}$. Without forming the normal matrix explicitly, there are two obvious implementations. The first is to compute the intermediate result $\boldsymbol{v} = G^{-1}A^T\boldsymbol{\pi}$ and then $\boldsymbol{\omega} = A\boldsymbol{v}$. It is called "two-pass" method because it requires two passes over the matrix A, either by rows or by columns. The second option is to initialize $\boldsymbol{\omega} = \mathbf{0}$ and then for each column j to compute $\alpha = g_j^{-1}A_j^T\boldsymbol{\pi}$ and to update $\boldsymbol{\omega} = \boldsymbol{\omega} + \alpha A_j$. It is called "one-pass" method because it makes one pass over a columnwise storage of A. The one-pass method and several versions of the two-pass method using rowwise or columnwise access to A have been implemented in IPX and benchmarked over a diverse set of LP matrices. On average the one-pass variant was 17% faster than the best two-pass variant and it was also the fastest method on most of the matrices. The advantage of the one-pass technique is that the matrix A is fetched into cache memory only once. The method is now used for all matrix-vectors products with normal matrices in IPX.

For terminating the initial interior point phase an iteration limit of $\min(500, 10+m/20)$ is set for the linear solver. If convergence is not achieved either while computing the predictor or the corrector direction, the current interior point iteration is restarted with basis preconditioning. Taking the geometric mean over the 87 problems from our test set that took longer than 10 seconds to solve, the initial iterations accounted for 8% of the IPM runtime.

6.4 Basis Preconditioned CR Method

Basis preconditioning enables a specific reduction of the KKT system to a smaller, positive definite one. Assume first that the computational form LP model does not contain free variables. Then G has a zero-free diagonal, $D = G^{-1/2}$ and (6.6) can be reduced to normal equations

$$AD^2A^T\Delta\boldsymbol{y} = \boldsymbol{r}^b - AD^2\boldsymbol{r}^a =: \boldsymbol{r}.$$

Given a basis $\mathcal{B}$, this system is transformed into

$$C\Delta\boldsymbol{u} := (A_\mathcal{B}D_\mathcal{B})^{-1}AD^2A^T(A_\mathcal{B}D_\mathcal{B})^{-T}\Delta\boldsymbol{u} = (A_\mathcal{B}D_\mathcal{B})^{-1}\boldsymbol{r} \tag{6.12}$$

and solved by the CR method for $\Delta\boldsymbol{u}$. The transformation is algebraically equivalent to applying the preconditioned CR method (Algorithm 8) with $M = A_\mathcal{B}D_\mathcal{B}^2A_\mathcal{B}^T$ to the normal equations, but iterating on $\Delta\boldsymbol{u}$ rather than $\Delta\boldsymbol{y}$ has shown to be numerically more stable when $A_\mathcal{B}D_\mathcal{B}$ is ill conditioned. From an approximate solution for $\Delta\boldsymbol{u}$ a solution to the KKT system is recovered from

$$\begin{aligned}
\Delta\boldsymbol{y} &= (A_\mathcal{B}D_\mathcal{B})^{-T}\Delta\boldsymbol{u},\\
\Delta\boldsymbol{x}_\mathcal{N} &= D_\mathcal{N}^2(\boldsymbol{r}_\mathcal{N}^a + A_\mathcal{N}^T\Delta\boldsymbol{y}),\\
\Delta\boldsymbol{x}_\mathcal{B} &= A_\mathcal{B}^{-1}(\boldsymbol{r}^b - A_\mathcal{N}\Delta\boldsymbol{x}_\mathcal{N}).
\end{aligned}$$

name	ndense	ipiter	diagonal		augmented	
			iter	time	iter	time
Linf_bts4	2	6	1547	2.17	996	1.61
Linf_five20b	2	4	617	0.92	404	0.69
jendrec1	58	6	1132	0.21	154	0.06
scfxm1-2r-256	54	7	2733	1.39	1326	0.79
stormg2-125	119	28	14164	15.75	986	1.43
stormg2_1000	119	9	4334	44.61	202	2.57
app1-2	265	12	2711	2.40	876	1.20
buildingenergy	19	2	331	1.30	69	0.36
dano3mip	1	6	595	0.13	586	0.13
in	1475	7	817	35.68	892	46.70
neos-506428	22	8	2001	2.50	660	0.92
neos-631710	556	9	512	1.87	174	0.96
neos-738098	1	4	935	0.27	1169	0.36
ns1663818	3	1	958	38.04	964	38.85
ns1758913	265	18	4407	11.28	198	0.71
ns1853823	656	9	1533	7.44	398	2.85
ns1854840	474	17	3518	10.48	1684	7.11
ns1905797	348	5	819	0.53	47	0.06
ns2017839	6	12	3949	3.86	1776	2.11
ns2118727	1248	10	3033	8.94	2988	17.05
ns2137859	320	19	3180	6.70	1731	5.31
opm2-z11-s8	8	12	2271	2.25	2175	2.44
opm2-z12-s7	8	12	2305	3.30	1426	2.33
reblock420	20	17	2904	0.93	580	0.22
rmatr100-p5	100	9	2629	0.34	1144	0.20
rmatr200-p5	200	8	3337	1.93	900	0.72
rmine14	28	14	3927	6.64	1636	3.35
rmine21	42	8	1721	16.28	367	4.31
wnq-n100-mw99-14	200	15	3225	9.49	208	0.79
neos1	1	8	1100	0.92	1146	0.97
neos3	1	7	1260	3.50	842	2.38
ns1687037	874	8	1322	7.65	1328	13.67
ns1688926	105	5	1005	2.21	198	0.92
nug30	900	11	1702	7.43	987	5.63

Table 5: Number of CR iterations and computation time (in seconds) for the diagonal and augmented diagonal preconditioner.

By definition of $\Delta\boldsymbol{x}$, the solution satisfies (6.8) with $\boldsymbol{\delta}_{\mathcal{N}} = \mathbf{0}$ and

$$\begin{aligned}
\boldsymbol{\delta}_{\mathcal{B}} &= D_{\mathcal{B}}^{-1}\Delta\boldsymbol{x}_{\mathcal{B}} - D_{\mathcal{B}}A_{\mathcal{B}}^{T}\Delta\boldsymbol{y} - D_{\mathcal{B}}\boldsymbol{r}_{\mathcal{B}}^{a} \\
&= D_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}(\boldsymbol{r}^{b} - A_{\mathcal{N}}\Delta\boldsymbol{x}_{\mathcal{N}}) - \Delta\boldsymbol{u} - D_{\mathcal{B}}\boldsymbol{r}_{\mathcal{B}}^{a} \\
&= D_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}\boldsymbol{r}^{b} - D_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}}^{2}(\boldsymbol{r}_{\mathcal{N}}^{a} + A_{\mathcal{N}}^{T}\Delta\boldsymbol{y}) - \Delta\boldsymbol{u} - D_{\mathcal{B}}\boldsymbol{r}_{\mathcal{B}}^{a} \\
&= (A_{\mathcal{B}}D_{\mathcal{B}})^{-1}\boldsymbol{r}^{b} - (A_{\mathcal{B}}D_{\mathcal{B}})^{-1}AD^{2}\boldsymbol{r}^{a} - \left(I_m + (A_{\mathcal{B}}D_{\mathcal{B}})^{-1}A_{\mathcal{N}}D_{\mathcal{N}}^{2}A_{\mathcal{N}}^{T}(A_{\mathcal{B}}D_{\mathcal{B}})^{-T}\right)\Delta\boldsymbol{u} \\
&= (A_{\mathcal{B}}D_{\mathcal{B}})^{-1}\boldsymbol{r} - C\Delta\boldsymbol{u},
\end{aligned}$$

which is the residual in (6.12). To satisfy the accuracy requirement (6.9), the iterative method is terminated when the infinity norm of the residual in (6.12) becomes smaller than τ. By solving the transformed system, the residual for the termination test is readily available.

When G has zeros on the diagonal, the KKT system can still be reduced to normal equations if all free variables are basic. This requirement is satisfied by the initial basis in IPX (see Section 6.6) and is maintained throughout. Hence let $\mathcal{B} = \mathcal{B}_0 \cup \mathcal{B}_1$, where $\mathcal{B}_0$ are the indices of all free variables. Then (6.7) can be permuted to

$$\begin{bmatrix} I_{\mathcal{N}} & & D_{\mathcal{N}}A_{\mathcal{N}}^{T} \\ & \begin{bmatrix} I_{\mathcal{B}_1} & \\ & 0 \end{bmatrix} & D_{\mathcal{B}}A_{\mathcal{B}}^{T} \\ A_{\mathcal{N}}D_{\mathcal{N}} & A_{\mathcal{B}}D_{\mathcal{B}} & 0 \end{bmatrix} \begin{pmatrix} D_{\mathcal{N}}^{-1}\Delta\boldsymbol{x}_{\mathcal{N}} \\ D_{\mathcal{B}}^{-1}\Delta\boldsymbol{x}_{\mathcal{B}} \\ -\Delta\boldsymbol{y} \end{pmatrix} = \begin{pmatrix} D_{\mathcal{N}}\boldsymbol{r}_{\mathcal{N}}^{a} \\ D_{\mathcal{B}}\boldsymbol{r}_{\mathcal{B}}^{a} \\ \boldsymbol{r}^{b} \end{pmatrix}.$$

Transforming this system using the scaled basis matrix yields

$$\begin{bmatrix} I_{\mathcal{N}} & & D_{\mathcal{N}}A_{\mathcal{N}}^{T}A_{\mathcal{B}}^{-T}D_{\mathcal{B}}^{-1} \\ & \begin{bmatrix} I_{\mathcal{B}_1} & \\ & 0 \end{bmatrix} & I_{\mathcal{B}} \\ D_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}} & I_{\mathcal{B}} & 0 \end{bmatrix} \begin{pmatrix} D_{\mathcal{N}}^{-1}\Delta\boldsymbol{x}_{\mathcal{N}} \\ D_{\mathcal{B}}^{-1}\Delta\boldsymbol{x}_{\mathcal{B}} \\ -\Delta\boldsymbol{u} \end{pmatrix} = \begin{pmatrix} D_{\mathcal{N}}\boldsymbol{r}_{\mathcal{N}}^{a} \\ D_{\mathcal{B}}\boldsymbol{r}_{\mathcal{B}}^{a} \\ D_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}\boldsymbol{r}^{b} \end{pmatrix},$$

where $\Delta\boldsymbol{u} = D_{\mathcal{B}}A_{\mathcal{B}}^{T}\Delta\boldsymbol{y}$ as before. It follows from the second block equation and $D_{\mathcal{B}_0}$ being the identity matrix that the components of $-\Delta\boldsymbol{u}$ corresponding to free variables are given by $\boldsymbol{r}_{\mathcal{B}_0}^{a}$ and can be eliminated. Also, the components of $\Delta\boldsymbol{x}_{\mathcal{B}}$ corresponding to free variables can be removed from the linear system and can be computed from the third block equation once the remaining components of $D^{-1}\Delta\boldsymbol{x}$ are known. Let $P = \begin{bmatrix} I_{\mathcal{B}_1} & 0 \end{bmatrix}$ have m columns. The resulting linear system is

$$\begin{bmatrix} I_{\mathcal{N}} & & D_{\mathcal{N}}A_{\mathcal{N}}^{T}A_{\mathcal{B}}^{-T}D_{\mathcal{B}}^{-1}P^{T} \\ & I_{\mathcal{B}_1} & I_{\mathcal{B}_1} \\ PD_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}} & I_{\mathcal{B}_1} & 0 \end{bmatrix} \begin{pmatrix} D_{\mathcal{N}}^{-1}\Delta\boldsymbol{x}_{\mathcal{N}} \\ D_{\mathcal{B}_1}^{-1}\Delta\boldsymbol{x}_{\mathcal{B}_1} \\ -P\Delta\boldsymbol{u} \end{pmatrix} = \begin{pmatrix} D_{\mathcal{N}}\boldsymbol{r}_{\mathcal{N}}^{a} \\ D_{\mathcal{B}_1}\boldsymbol{r}_{\mathcal{B}_1}^{a} \\ PD_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}\boldsymbol{r}^{b} \end{pmatrix} - \begin{pmatrix} \boldsymbol{\xi} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix},$$

where

$$\boldsymbol{\xi} = D_{\mathcal{N}}A_{\mathcal{N}}^{T}A_{\mathcal{B}}^{-T}D_{\mathcal{B}}^{-1}\begin{pmatrix} \mathbf{0} \\ \boldsymbol{r}_{\mathcal{B}_0}^{a} \end{pmatrix} = D_{\mathcal{N}}A_{\mathcal{N}}^{T}A_{\mathcal{B}}^{-T}\begin{pmatrix} \mathbf{0} \\ \boldsymbol{r}_{\mathcal{B}_0}^{a} \end{pmatrix}.$$

Because the upper left 2×2 block now has a zero-free diagonal, the system can be reduced to normal equations

$$\begin{aligned}
\left(I_{\mathcal{B}_1} + PD_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}}^{2}A_{\mathcal{N}}^{T}A_{\mathcal{B}}^{-T}D_{\mathcal{B}}^{-1}P^{T}\right)(P\Delta\boldsymbol{u}) = \\
- PD_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}A_{\mathcal{N}}D_{\mathcal{N}}(D_{\mathcal{N}}\boldsymbol{r}_{\mathcal{N}}^{a} - \boldsymbol{\xi}) - D_{\mathcal{B}_1}\boldsymbol{r}_{\mathcal{B}_1}^{a} + PD_{\mathcal{B}}^{-1}A_{\mathcal{B}}^{-1}\boldsymbol{r}^{b}, \quad (6.13)
\end{aligned}$$

and can be solved for $P\Delta\boldsymbol{u}$. The solution for $\Delta\boldsymbol{u}$ is completed by inserting the components corresponding to free variables given by $-\boldsymbol{r}_{\mathcal{B}_0}^{a}$.

The dimension of (6.13) is m minus the number of free variables. In the IPX implementation the CR method always operates on vectors of dimension m, in which components corresponding to free variables are initialized to zero and are reset to zero after each matrix-vector product with C defined in (6.12). The only actual change to the code was to replace the right-hand side of (6.12) by that of (6.13), scattered into a full size vector. The result is a concise handling of free variables, which implicitly reduces the dimension of the linear systems.

Computing matrix-vector products with the transformed normal matrix

$$C = I_m + (A_{\mathcal{B}} D_{\mathcal{B}})^{-1} A_{\mathcal{N}} D_{\mathcal{N}}^2 A_{\mathcal{N}}^T (A_{\mathcal{B}} D_{\mathcal{B}})^{-T} \tag{6.14}$$

is by far the dominating cost of one CR iteration. They are implemented through a matrix-vector product with $A_{\mathcal{N}} D_{\mathcal{N}}^2 A_{\mathcal{N}}^T$ using the one-pass method, and two linear system solves with an LU factorization of $A_{\mathcal{B}} D_{\mathcal{B}}$. The row and column permutation from the LU factorization are applied a priori to $A_{\mathcal{N}}$ and to the vectors occurring in the CR method, respectively, so that no permutations are needed in the CR iterations.

6.5 Maintaining a Basis Matrix

The idea for the preconditioner in IPX is to use a ρ-maximum volume basis of AD, given some parameter $\rho \geq 1$. As discussed in Chapter 2, finding such a basis requires an iterative update procedure. The difficulty in implementing an update method is finding pivot elements efficiently. Because the scaled tableau matrix $D_{\mathcal{B}}^{-1} A_{\mathcal{B}}^{-1} A_{\mathcal{N}} D_{\mathcal{N}}$ is only available implicitly through operations with $A_{\mathcal{B}}^{-1}$, searching systematically for an entry that is larger than ρ in absolute value is expensive, as shown by the experiments in Section 2.3. For large-scale problems it is generally impractical to compute all entries of the tableau, so that we cannot even test if a given basis has ρ-maximum volume.

To make the approach practical, the target must therefore be relaxed, keeping in mind that the ultimate goal is solving the linear systems efficiently. The author's idea was to update the basis as long as pivot elements which increase the volume are readily found, and to stop updating when the pivot search becomes costly. The key component of such a method is the rule for selecting candidate rows or columns of the scaled tableau matrix to be "scanned" for pivot elements.

A first approach was to choose a nonbasic variable as candidate to enter the basis if its scaling factor increases significantly, or likewise to choose a basic variable as candidate to leave the basis when its scaling factor decreases significantly. Computational results were disappointing. The issue was that when the LP model is primal (or dual) degenerate, there exist basic variables whose scaling factor tends to zero (or nonbasic variables whose scaling factor tends to infinity) as the IPM converges. These variables were tried repeatedly to leave (or to enter) the basis, without finding a sufficiently large entry in the corresponding row (or column) of the scaled tableau matrix. Because in practice LP models are often highly primal and dual degenerate, choosing candidate variables by looking only at their scaling factor is not effective.

Better results were obtained by computing a weight for each nonbasic variable based on the entries in its scaled tableau column, and choosing the variable with maximum weight as candidate to enter the basis. Such a procedure is stated in Algorithm 9.

Assume first that the parameter *nslices* is 1. Then w_j is the sum of entries in column j of the scaled tableau matrix if $j \in \mathcal{N}$. The algorithm chooses the index with maximum such weight as candidate and computes the corresponding column of the scaled tableau matrix. If its maximum entry is larger than ρ in absolute value, the basis is updated. In this case the tableau row needs to be computed as well for updating the column sums (line 14). If a column does not contain a pivot element, it is not search again in that interior point iteration. After computing $maxskip + 1$ columns without finding a pivot, the algorithm terminates.

Algorithm 9 Maxvolume Heuristic

Input: basis $\mathcal{B}$, parameters $(\rho, nslices, maxskip)$ with $\rho \geq 1$, $1 \leq nslices \leq m$, $maxskip \geq 0$.

for $s = 0$ to $nslices - 1$ **do**
  Let $\boldsymbol{u} \in \mathbb{R}^m$, $\boldsymbol{w} \in \mathbb{R}^{n+m}$.
  for $i = 1$ to m **do** // set row mask
    $u_i = 1$ if $\text{mod}(i, nslices) = s$
    $u_i = 0$ if $\text{mod}(i, nslices) \neq s$
  $\boldsymbol{w}_{\mathcal{N}}^T = \boldsymbol{u}^T D_{\mathcal{B}}^{-1} A_{\mathcal{B}}^{-1} A_{\mathcal{N}} D_{\mathcal{N}}$, $\boldsymbol{w}_{\mathcal{B}} = \boldsymbol{0}$ // initialize column weights
  $nskipped = 0$ // count "skipped" columns
  while $nskipped \leq maxskip$ **do**
    $j = \arg\max_k |w_k|$ // choose candidate column
    Compute $\boldsymbol{v} = D_{\mathcal{B}}^{-1} A_{\mathcal{B}}^{-1} A_j d_j$.
    if $\|\boldsymbol{v}\|_\infty > \rho$ **then**
      $p = \arg\max_i |v_i|$
      $\mathcal{B}_p = j$
      Update $\boldsymbol{w}$.
    else // no pivot found in column j
      $nskipped = nskipped + 1$
      Flag column j to be excluded from pivot search.

When $nslices > 1$, then in each iteration of the outer loop only a subset of the rows of the tableau matrix contributes to the column weights. This slicing of the tableau matrix decreases the chance that when a column has large positive and negative entries that these cancel out in the column sum; and it makes the algorithm adaptive to the number of rows of A.

IPX runs Algorithm Maxvolume Heuristic at the beginning of each interior point iteration with the new scaling matrix D and the basis from the previous iteration as starting basis. Free and fixed variables are excluded from the pivot search and remain basic and nonbasic, respectively. The default parameters are $nslices = 5 + \lfloor m/10000 \rfloor$ and $maxskip = 10$. Assuming that the IPM performs no more than 100 iterations, the total number of skipped columns is bounded by $5000 + m/10$. The specific parameter values have shown little effect on the number of basis updates and the quality of the basis preconditioner.

To investigate the number of basis updates and CR iterations, we consider two LP models, `pds-20` and `nug20`, which have similar dimensions (see Table 6) but different characteristics. IPX was run on these models twice, once using Algorithm Maxvolume Heuristic for updating the basis, and once using Algorithm Maxvolume Sequential from Section 2.3, which yields a (true) ρ-maximum volume basis. Each update method was run with parameter $\rho \in \{1.1, 2.0, 4.0, 10.0\}$, giving a total of 8 IPM solves per model. We note that computing the ρ-maximum volume basis through Algorithm Maxvolume Sequential was a very expensive task for `nug20` and is not an option in practice. For the comparisons, IPX was run without fixing variables (Section 6.7) and without crossover (Section 6.8); the remaining parameters were defaults. The initial IPM iterations and the starting basis were the same for all runs on the same model.

The number of basis updates and CR iterations per interior point iteration are plotted in Figures 3 and 4, and their grand totals (excluding initial IPM iterations) are given in Table 6. Regarding the total number of basis updates, there is little difference between Algorithm Maxvolume Heuristic and Algorithm Maxvolume Sequential, except for $\rho = 1.1$. This means that if there would exist an efficient method for finding pivot elements, then maintaining a (true) ρ-maximum volume basis for $\rho \geq 2$, say, would be computationally feasible. Regarding the plots of the CR iteration counts, it is seen that for both models and all values of ρ the curves closely resemble each other, showing that the heuristically found basis yields an equally good

name (dimension)	ρ	Maxvolume Heuristic		Maxvolume Sequential	
		updates	CR iter	updates	CR iter
`pds-20`	1.1	2708	3700	3546	3554
($m = 12081$,	2.0	1899	4370	1981	4109
$n = 81163$)	4.0	1654	5738	1648	5564
	10.0	1492	9225	1493	8869
`nug20`	1.1	15051	22028	29184	17559
($m = 14098$,	2.0	10383	25654	11993	22033
$n = 72546$)	4.0	8455	34299	9075	29854
	10.0	7137	58039	7396	49747

Table 6: Total operation counts.

preconditioner as a maximum volume basis. The curves also illustrate the typical behaviour of the CR iteration counts during the course of the IPM: one or more peaks usually occur after the switch to basis preconditioning, and the iterations level out toward the end of the interior point solve.

On the two example models, a smaller ρ systematically reduced the number of CR iterations. The behaviour is not always that clear, however. As discussed in Section 3.1, sparsity in the tableau matrix can also have a large effect on the spectrum of C and thereby the CR iteration count. This is seen quite impressively by comparing the hypersparse `pds-20` to the not-hypersparse `nug20`. Despite their similar dimensions, the average number of CR iterations per linear system (not shown) was about 10 times higher for `nug20`. When the density of the tableau matrix varies significantly between different bases, it can happen that a smaller ρ leads to more CR iterations. Such a behaviour has been observed on a number of LP models. It is not obvious how to choose a basis that preserves sparsity as far as possible while targeting the maximum volume criterion. In the LP context this remains an important topic for further investigation.

When the correlations are as clear as in Table 6, then the optimal ρ in terms of total computation time must trade-off the cost for updating the basis to the reduction of the time spent on the CR method. A larger set of models has shown that values between 2.0 and 4.0 lead to similar and close-to-optimal total runtime. Therefore IPX uses a problem independent default of $\rho = 2.0$.

6.6 Crash Basis

At the switch to basis preconditioning a starting basis must be determined, given the current interior point iterate and its associated scaling matrix D. To reduce the number of basis updates in the first run of Algorithm Maxvolume Heuristic and to make the linear algebra operations fast, the basis should have the following (often competing) properties:

(i) The basis matrix is well conditioned.

(ii) The basis matrix and ideally the tableau matrix are sparse.

(iii) The basis is close to a maximum volume basis for the current D.

(iv) All free variables are basic.

(v) All fixed variables are nonbasic.

Making all free variables basic is required for the reduction of the KKT system to normal equations (Section 6.4) and is always achievable. If the columns corresponding to free variables

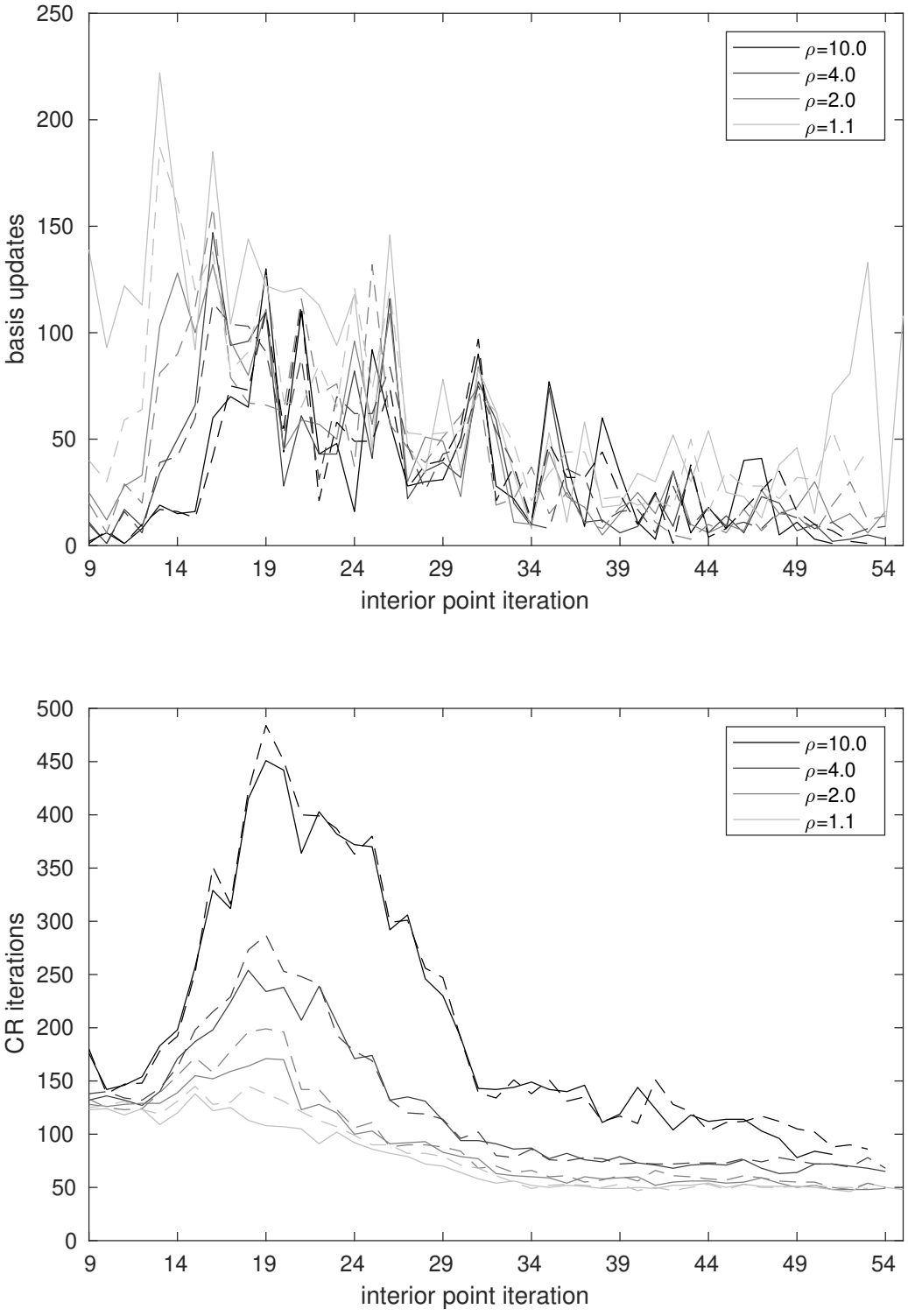


Figure 3: LP model `pds-20` with ρ-maximum volume basis (solid line) and heuristically computed basis (dashed line).

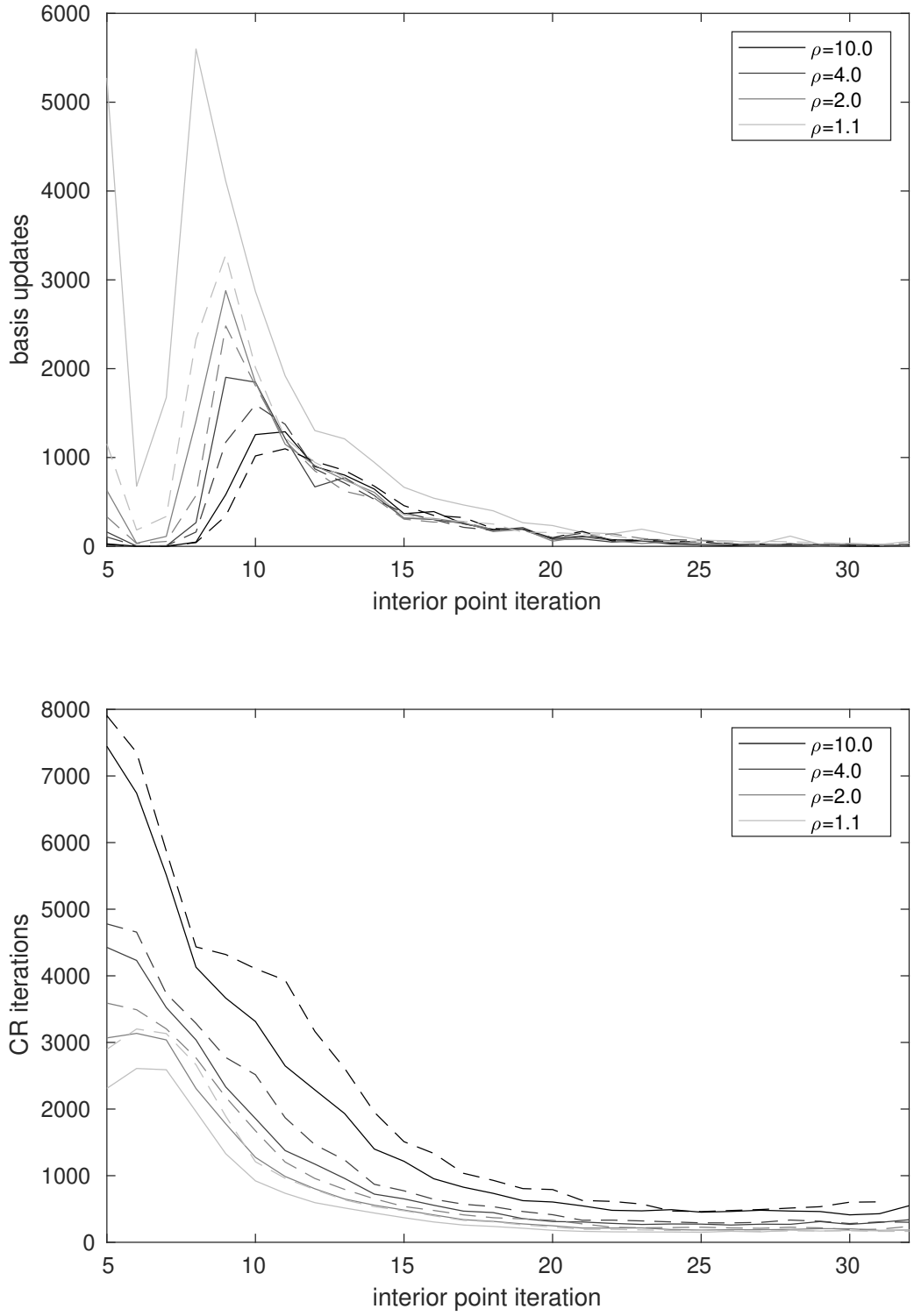


Figure 4: LP model `nug20` with ρ-maximum volume basis (solid line) and heuristically computed basis (dashed line).

are linearly dependent, then the model is either dual infeasible or some of the variables are redundant and can be fixed at an arbitrary value. The analogue requirement is to make all fixed variables nonbasic, which allows them to be removed from the optimization entirely. Again, this is always achievable, for if a fixed slack variable cannot be replaced in the basis, then either is the corresponding row of A redundant and can be removed, or the model is primal infeasible. In theory, such removals could happen during preprocessing. The computational cost for these operations can be high, however, as it requires finding a basis that contains the maximum number of free variables and the minimum number of fixed variables. Finding such a basis is a nontrivial task in general. It is therefore preferable to construct a starting basis only once at the switch to basis preconditioning.

The obvious method for finding a basis that satisfies (v) is by computing a rectangular LU factorization of A after removing columns corresponding to fixed slack variables. This would be unacceptably expensive for many large-scale problems, however. Instead, IPX "crashes" a starting basis through the following steps:

Initial guess A set $\mathcal{J}$ of m column indices is constructed as follows:

(i) If the LP model contains free variables, the first step is computing an incomplete left-looking LU factorization of the corresponding columns of A. In the left-looking method L starts out to be the identity matrix and its strictly lower triangular part is computed one column at a time. Let j be the next free variable, $\hat{\boldsymbol{a}} = L^{-1}A_j$ and i be such that $|\hat{a}_i|$ is maximum among all entries of $|\hat{\boldsymbol{a}}|$ whose row has not been pivotal. If $|\hat{a}_i| > 10^{-3}$, variable j is added to $\mathcal{J}$ and the next column of L is composed from the entries of $\hat{\boldsymbol{a}}/\hat{a}_i$ that are nonzeros in A_j and have not been pivotal. (Restricting the column of L to the nonzero pattern of A_j makes the factorization incomplete.) After processing all free variables, L is discarded but the index set $\mathcal{I}$ of pivot rows is kept.

(ii) In the second and third step columns of A corresponding to fixed and free variables are treated as vacant; i. e. they are treated as being structurally zero in A and AD. The second step adds singleton columns to $\mathcal{J}$ if their entry in AD is sufficiently large. For each $i \notin \mathcal{I}$ the maximum entry in row i of $|AD|$ and the maximum entry that lies in a singleton column are determined. If the singleton entry is nonzero and not smaller than 0.5 times the maximum of the row, its column is added to $\mathcal{J}$ and i is added to $\mathcal{I}$.

(iii) Let A_{33} be the submatrix of A composed of rows $i \notin \mathcal{I}$ and columns $j \notin \mathcal{J}$. The third step extends $\mathcal{I}$ and $\mathcal{J}$ by choosing a structurally independent subset of the columns of A_{33}. Processing in decreasing order of the d_j, the next column from A_{33} is tested for being structurally dependent on the columns from A_{33} already chosen by computing an alternating augmenting path, see Duff [15]. If the candidate column is independent, its index is added to $\mathcal{J}$ and the row that was newly matched by the augmenting path is added to $\mathcal{I}$. The method stops at the latest when $|\mathcal{J}| = m$ or when all columns of A_{33} have been processed. It is stopped before if too many candidate columns were structurally dependent, ensuring that the computation time is small.

(iv) Finally $\mathcal{J}$ is completed by adding slack variables for $i \notin \mathcal{I}$.

Initial factorization The task is to find a well conditioned basis matrix that contains as many columns of $\mathcal{J}$ as possible. A right-looking Markowitz LU factorization of $A_{\mathcal{J}}$ is computed with the usual columnwise threshold pivoting. If during the course of the factorization all entries in a column of the active submatrix become smaller than 10^{-3}, the column is immediately removed without choosing a pivot. After completion, the LU factors are padded with unit columns

whose row has not been pivotal. Accordingly a preliminary basis $\mathcal{B}$ is obtained composed of indices of $\mathcal{J}$ and indices of slack variables.

Basis repair The resulting basis matrix $A_{\mathcal{B}}$ would be nonsingular in exact arithmetic. It is well known, however, that $A_{\mathcal{B}}$ can have tiny singular values even if no small pivots occur in the LU factorization, and such cases actually happen in practice. It is therefore essential to control the condition number of $A_{\mathcal{B}}$ and to repair numerical singularities if necessary. Because the model is scaled during preprocessing so that the maximum entry of A is bounded by a moderate number, a high condition number of $A_{\mathcal{B}}$ can only be caused by large entries in $A_{\mathcal{B}}^{-1}$. As described by Higham and Relton [35] a rook search is performed for estimating the maximum absolute entry of $A_{\mathcal{B}}^{-1}$ along with its position (p, i) in the matrix. If the entry is larger than 10^5, then $\mathcal{B}_p$ is replaced by the i-th slack variable and the rook search is repeated. During the basis repair $A_{\mathcal{B}}$ typically requires frequent refactorization due to numerical instability in the LU update. If the basis matrix is refactorized, columns are dropped from the active submatrix and replaced by unit columns as in the initial factorization.

Handling free and fixed variables The obtained basis may contain fixed slack variables and may not contain all free variables. These "defects" are corrected by basis updates. For each free variable that is nonbasic a column of the tableau matrix is computed and the variable is pivoted into the basis if it can replace a non-free basic variable. If not, the model is either declared dual infeasible or the free variable is fixed at zero. Likewise, for each fixed (slack) variable in the basis a row of the tableau matrix is computed and searched for an exchange column. If the tableau row is numerically zero, the model is either declared primal infeasible or the constraint corresponding to the slack variable is removed. After correcting all defects, the remaining fixed variables (which are nonbasic now) are excluded from the IPM solve.

The described procedure has been developed through extensive testing and works efficiently on most real-world problems. Steps (i)–(iii) of the initial guess are designed to be fast and to yield a column subset of close-to full rank. Depending on the characteristics of the LP model, each of the three steps may add the majority of columns to $\mathcal{J}$. Computing the starting basis typically accounts for less than 5% of the total IPX runtime. In the cases where the computation time is significant, the last step of the above procedure often dominates due to the number of pivots for fixed variables. Processing these variables as described is advantageous later, however, because otherwise fixed variables can lead to small step sizes in the IPM and dependent equality constraints can cause dual variables to blow up.

The crash bases are surprisingly close to the bases at the end of the interior point solve. In Table 7 the number of basis updates is reported in relation to m. Note that a value 0.25, for example, means that at least 75% of the starting basis must be final.

6.7 Fixing Variables

Let us say that the scaling factor d_j is primal degenerate if $d_j \to 0$ and $j \in \mathcal{B}$, and dual degenerate if $d_j \to \infty$ and $j \in \mathcal{N}$, when $\mathcal{B}$ is a basis of maximum cardinality (see Section 5.1). Degenerate scaling factors correspond to degeneracies of the LP model and are a source of numerical difficulties in the IPM. They eventually lead to multiplications with large quantities and to divisions by small quantities for composing the right-hand side of (6.12), recovering the solution to the KKT system and computing matrix-vector products with (6.14), even if the entries in the scaled tableau matrix are bounded. The resulting round-off errors limit the accuracy that can be achieved by the linear algebra. For that reason several models in our test set could not be solved to 8-digit accuracy by an early version of IPX.

basis updates / m	instances
0.000–0.010	24
0.010–0.025	8
0.025–0.050	6
0.050–0.100	16
0.100–0.250	46
0.250–0.500	22
0.500–1.000	38
1.000–3.350	5

Table 7: Number of basis updates in the IPM starting from crash basis (165 models in total).

A related issue are unbounded solution sets, in which case the IPM tends to push variables toward infinity. An unbounded dual solution requires primal degeneracy of the LP model and often manifests in some x_j^l or x_j^u for $j \in \mathcal{B}$ to converge to zero preliminarily. Consequently some primal degenerate scaling factors already approach zero in early iterations. Both the degenerate scaling factors and the large dual variables can severely limit the accuracy of the step directions and prevent the IPM from reaching its termination criterion. The analogue can happen for an unbounded primal solution set, but these occur less often because many LP models have finite lower and upper bounds on the variables.

A robust solution to these issues is fixing degenerate variables during the optimization process. The theoretical foundation for this was laid in Section 5.5. Its implementation in IPX is stated in Algorithm 10, where for simplicity it is assumed that $u_j = \infty$ for all j. In the actual implementation the smaller of x_j^l and x_j^u is considered in the loop over $\mathcal{B}$ and the larger of z_j^l and z_j^u is considered in the loop over $\mathcal{N}$. The procedure is executed in each IPM iteration with basis preconditioning prior to updating the basis by the maximum volume algorithm.

Assume the parameters ε_p and ε_d to be small enough such that

$$x_j^l \le \varepsilon_p \implies (x_j^l)^* = 0 \quad \text{and} \quad z_j^l \le \varepsilon_d \implies (z_j^l)^* = 0 \quad \text{for all } j, \tag{6.15}$$

where $\left(\boldsymbol{x}, \boldsymbol{x}^l, \boldsymbol{x}^u, \boldsymbol{y}, \boldsymbol{z}^l, \boldsymbol{z}^u\right)^*$ is a strictly complementary solution. If no basis exchange happens in the loop over $\mathcal{B}$, variable j is considered primal degenerate; i. e. its position in the basis can only be filled by a variable whose scaling factors becomes zero. Similarly, if no exchange happens in the loop over $\mathcal{N}$, variable j is considered dual degenerate; i. e. it can only replace basic variables whose scaling factor tends to infinity. The correctness of the procedure was proved in Section 5.5 under the assumption that (6.15) holds and that the values of the fixed variables are in the interior of the solution set. Both conditions will be satisfied for ε_p and ε_d sufficiently small. If the conditions do not hold, then either the modified LP model will be infeasible or the obtained "solution" will not be an optimal solution for the original problem. As seen from line 10 of Algorithm 10, postprocessing a fixed dual variable increases the primal residual if x_j was not at its supposed bound. Similarly, postprocessing a fixed primal variable in line 19 increases the dual residual if $A_j^T \boldsymbol{y} = c_j$ was not satisfied for the column treated as absent.

The default parameters in IPX are $\varepsilon_p = \varepsilon_d = 10^{-9}$. Fixing variables resolved the linear algebra issues caused by degeneracy and the blow-up of variables so that the IPM reached its termination criterion on almost all models in our test set. The few problems that could not be solved failed for other reasons (see Section 8.2). The IPM termination criterion requires in addition to (6.5) that the relative primal and dual residuals are $\le 10^{-6}$ (by default). The termination condition is tested again after postprocessing fixed variables and rescaling the solution to the user data. If it is not satisfied, then the final solution is declared "imprecise"

Algorithm 10 Fixing Degenerate Variables

Input: $\varepsilon_p > 0$, $\varepsilon_d > 0$.

for each $j \in \mathcal{B}$ **do** // Scan for primal degenerate variables
  if $x_j^l \le \min(0.01 z_j^l, \varepsilon_p)$ **then**
    Let p be the position of j in $\mathcal{B}$ and compute $\boldsymbol{r}^T = \boldsymbol{e}_p^T D_{\mathcal{B}}^{-1} A_{\mathcal{B}}^{-1} A_{\mathcal{N}} D_{\mathcal{N}}$.
    if $\|\boldsymbol{r}\|_\infty > 2$ **then**
      Replace j by $j' \in \mathcal{N}$ corresponds to the maximum entry in $|\boldsymbol{r}|$.
    else
      // Fix dual variable:
      Store $\bar{z}_j^l = z_j^l$.
      Set $c_j = c_j - z_j^l$, $z_j^l = 0$, $x_j^l = \infty$ and treat l_j as $-\infty$ in subsequent iterations.
      After termination of the IPM restore $z_j^l = \bar{z}_j^l$ and set $x_j = l_j$, $x_j^l = 0$.
for each $j \in \mathcal{N}$ **do** // Scan for dual degenerate variables
  if $z_j^l \le \min(0.01 x_j^l, \varepsilon_d)$ **then**
    Compute $\boldsymbol{v} = D_{\mathcal{B}}^{-1} A_{\mathcal{B}}^{-1} A_j d_j$.
    if $\|\boldsymbol{v}\|_\infty > 2$ **then**
      Replace j' by j in the basis, where $j' \in \mathcal{B}$ corresponds to the maximum entry in $|\boldsymbol{v}|$.
    else
      // Fix primal variable:
      Set $\boldsymbol{b} = \boldsymbol{b} - A_j x_j$ and treat column j as absent in subsequent iterations.
      After termination of the IPM set $x_j^l = x_j - l_j$ and $z_j^l = 0$.

and requires to be cleaned up by crossover and possibly a simplex run. This happened on 7 models in our test set.

Figure 5 visualizes the number of primal variables in relation to n and the number of dual variables in relation to m that were fixed by the above procedure when running the IPM to 8-digit accuracy. Each cross in the plot represents one model from our test set, counting 23 instances for which only primal variables became fixed, 50 instances for which only dual variables became fixed, and 53 instances for which at least one primal and one dual variable became fixed. On the majority of models a significant fraction of the variables (say $> 1\%$) was degenerate and close to a bound (primal) or close to zero (dual) when the IPM reached its standard termination criterion. Regarding that ε_p and ε_d were chosen quite small, it is not surprising that numerical difficulties can occur at this stage in the IPM.

6.8 Crossover

For the crossover it is convenient to combine the dual slack variables into $\boldsymbol{z} = \boldsymbol{z}^l - \boldsymbol{z}^u$. A primal-dual solution $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ then satisfies

$$A\boldsymbol{x} = \boldsymbol{b}, \quad A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c}, \tag{6.16a}$$

$$\boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}, \tag{6.16b}$$

$$z_j \le 0 \text{ if } x_j > l_j \quad \text{for all } j, \tag{6.16c}$$

$$z_j \ge 0 \text{ if } x_j < u_j \quad \text{for all } j. \tag{6.16d}$$

The crossover method in IPX starts by "dropping" the final IPM iterate to an $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ that satisfies (6.16b)–(6.16d); that means, for each variable either x_j is set to a bound or z_j is set

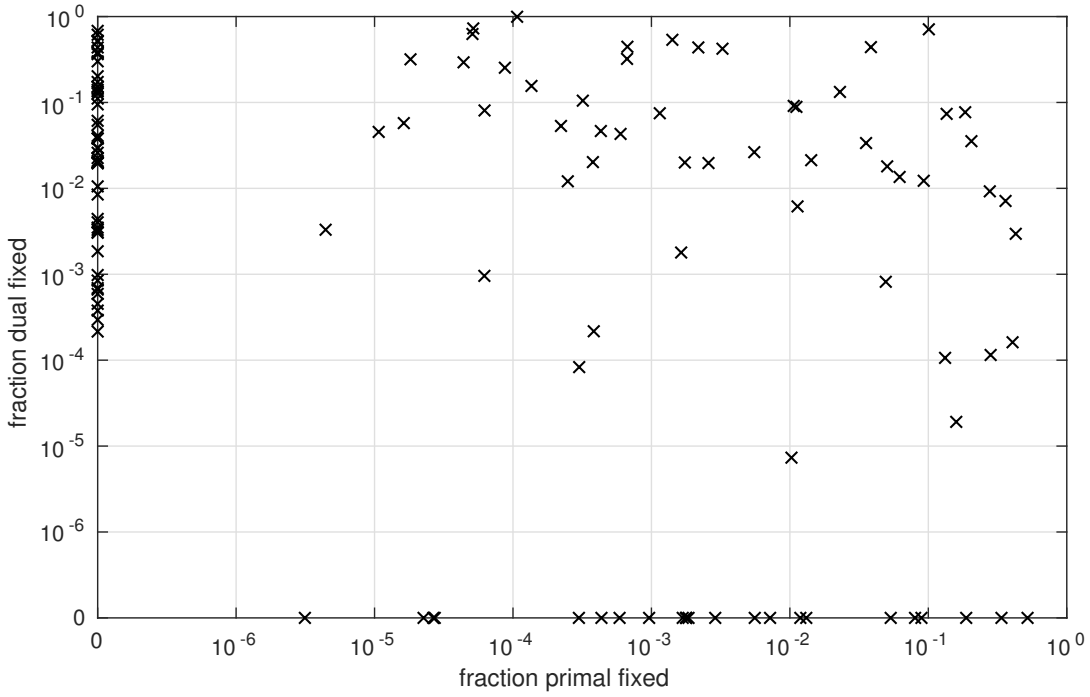


Figure 5: LP models scattered according to the number of variables fixed by the IPM.

to zero, depending on which perturbation is smaller. Dropping usually increases violation of (6.16a). The basis from the preconditioner then serves as starting basis for the push phases described in Section 5.4, where the superbasic variables are now $\mathcal{N}^+ = \{j \in \mathcal{N} \mid l_j < x_j < u_j\}$ and $\mathcal{B}^+ = \{j \in \mathcal{B} \mid z_j \neq 0\}$.

IPX always executes the dual push phase first. A dual push is blocked if the update to $\boldsymbol{z}_\mathcal{N}$ would violate (6.16c)–(6.16d). Notice that in the form (6.16) the complementarity requirement has been incorporated into the dual feasibility condition. Hence complementarity is maintained even if the starting basis is not of maximum cardinality. In this case it can happen that an update to z_j is nonzero for some $j \in \mathcal{N}^+$, causing a block immediately. The required pivot that exchanges $i \in \mathcal{B}^+$ by $j \in \mathcal{N}^+$ is called a "wrong pivot" because it would not occur if the IPM had been run to sufficiently high accuracy. Wrong pivots occasionally appear in practice and are treated as any other blocking in the dual push phase.

In the primal push phase superbasic variables can be moved toward any finite bound, and the nearer one is chosen in IPX. A push is blocked if the update to $\boldsymbol{x}_\mathcal{B}$ would violate (6.16b). Because there exist no dual superbasic variables, wrong pivots do not occur.

The push phases process primal and dual superbasic variables in decreasing and increasing order, respectively, of the scaling factors from the final IPM iterate. Empirical tests did not show a systematic difference in the number of pivot operations compared to other orderings, and considering the scaling factors is the most natural choice that makes the computations invariant to a column permutation of A.

In the implementation care must be taken with small pivot elements to prevent the basis matrix from becoming too ill conditioned for finite precision arithmetic. A blocking variable is eligible as pivot only if the pivot element is larger than 10^{-5} in absolute value. Among all candidates the exchange variable is chosen by the two-pass ratio test proposed by Harris [34] for the simplex method, which allows larger pivot elements by exploiting a feasibility tolerance

for (6.16b)–(6.16d) with default value 10^{-7}. The combination of the two measures made the implementation robust against failure due to a numerically singular basis matrix. The modified pivot search proposed in Section 5.4 is not used in the final version of the code. It alone did not prevent the basis matrix from becoming numerically singular in some cases, and in combination with the two-pass ratio test it did not provide a significant improvement to the conditioning of basis matrices.

Throughout the push operations the conditions (6.16b)–(6.16d) are maintained exactly by truncating the update to a variable if necessary. Along with the initial dropping to complementarity and the fact that the primal and dual residuals of the final IPM iterate are not exactly zero, this results in a violation of (6.16a). At the end of the push phases $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ is therefore a basic solution to the LP model with perturbed right-hand side and objective, and $\mathcal{B}$ may or may not be an optimal basis for the original problem. If it is not, reoptimization with the simplex method is required.

The main reason for a non-optimal final basis is that the pairwise complementarity products $x_j^l z_j^l$ and $x_j^u z_j^u$ can be quite large when the IPM reaches the standard termination criterion (6.5), resulting in a large perturbation. To reduce the number of simplex iterations in the clean-up, IPX uses a more stringent termination criterion for the IPM when crossover is requested. In addition to (6.5) it is required that

$$\max_j \|\delta x_j A_j\|_\infty \le 10^{-8}(1 + \|(\boldsymbol{b}, \boldsymbol{l}_\mathcal{L}, \boldsymbol{u}_\mathcal{U})\|_\infty), \tag{6.17a}$$

$$\max_j |\delta z_j| \le 10^{-8}(1 + \|\boldsymbol{c}\|_\infty), \tag{6.17b}$$

where δx_j and δz_j are the perturbations to drop the iterate to complementarity. (For each j either δx_j or δz_j is zero.) The reason behind the criterion is to control the (relative) primal and dual residuals that are introduced by dropping any variable to complementarity. These residuals are decisive for the cost of the simplex clean-up, so that using (6.17) is more appropriate than controlling the complementarity measure (6.3).

On 60% of the models in our test set the stricter condition forced at least one extra interior point iteration, and except for some badly scaled models no more than 6 iterations were needed. Hence the additional computation time for the IPM is moderate, in particular because the final iterations are typically fast with basis preconditioning. Using the stricter termination criterion for the IPM, the basis after the push phases was optimal in 150 out of 165 cases.

An extreme example is the highly degenerate LP model `nug30`. Without imposing (6.17) the vertex solution defined by the basis at the end of the push phases violated the bound constraints by $7 \cdot 10^{-4}$ and the dual sign condition by $5 \cdot 10^{-5}$. Neither the primal nor dual simplex from the commercial software Gurobi 7.0 finished within 36,000 seconds. By imposing (6.17) the IPM performed 4 additional iterations and the primal infeasibility of the final vertex decreased to $9 \cdot 10^{-11}$. The primal simplex then solved the model to optimality in one iteration.

A conventional IPM implementation would not be able to achieve (6.17) reliably because the linear systems would eventually become too ill conditioned to be solved to sufficient accuracy. Imposing the stringent termination criterion in IPX became possible by the dynamic elimination of degenerate variables described in the previous section.

7 Computing and Updating Sparse LU Factors

Computing an inverse representation of a basis matrix and maintaining it after column changes are the key operations for preparing a basis preconditioner in the IPM. The same type of matrices and operations occur in the revised simplex method, for which tailored linear algebra kernels have been developed over the past decades. This chapter reviews the two standard components in modern simplex codes: the Markowitz LU factorization and the update method of Forrest and Tomlin [18]. An extension to the update is presented that avoids its algebraic operations whenever a "spiked matrix" can be permuted to triangular form. The implementation of the factorization and update in a software package called BASICLU is described. The code was written by the author to provide the linear algebra operations for IPX.

Throughout the chapter permutation matrices and permutation vectors are used. When $\boldsymbol{p}$ and $\boldsymbol{q}$ are permutation vectors, $B(\boldsymbol{p},\cdot)$ is the matrix whose i-th row is the p_i-th row of B and $B(\cdot,\boldsymbol{q})$ is the matrix whose j-th column is the q_j-th column of B. The corresponding permutation matrices P and Q are permutations of the identity matrix such that $B(\boldsymbol{p},\boldsymbol{q}) = PBQ^T$. A square matrix B is said to be "symmetrically permuted triangular" if there exists a permutation matrix P such that PBP^T is upper triangular.

7.1 Factorizing LP Basis Matrices

The INVERT procedure in simplex codes decomposes a basis matrix B into $PBQ^T = LU$, where L is unit lower triangular, U is upper triangular and the permutations P and Q are chosen to preserve sparsity and numerical stability.

A large part of LP basis matrices can often be brought to triangular form by permutation. The first step in the basis factorization is therefore to permute B into

$$\tilde{P}B\tilde{Q}^T = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ & B_{22} & \\ & B_{32} & B_{33} \end{bmatrix} = \begin{bmatrix} I & & \\ & B_{22} & \\ & B_{32} & I \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ & I & \\ & & B_{33} \end{bmatrix}, \tag{7.1}$$

where B_{11} and B_{22} are upper and lower triangular, respectively, and the "bump" B_{33} contains no further singletons. In the form (7.1) column singletons have been eliminated before row singletons, resulting in the B_{12} block to appear in U. This is desired because keeping L sparse leads to a sparse spike column in the update. Finding the initial triangular form requires no sparse matrix manipulations; it is done in BASICLU by a method of J. R. Gilbert, explained in [13, exercise 3.7].

For the bump factorization a right-looking method with dynamic Markowitz ordering has become standard in modern simplex codes. Each pivot step of a right-looking factorization updates the "active submatrix", which consists of the rows and columns that have not been pivotal; see Duff et al. [16] for an introduction to sparse matrix factorizations. The Markowitz ordering chooses the pivot element by means of a small Markowitz count $(r_i - 1)(c_j - 1)$, where r_i and c_j are the number of nonzeros in row i and column j of the active submatrix. The Markowitz count is an upper bound on the fill-in created by the pivot operation. To ensure numerical stability, an entry of the active submatrix is eligible as pivot only if it is not smaller than a factor $0 < u \leq 1$ times the maximum in its column. Values between $u = 0.01$ and $u = 0.1$ are commonly used in practice.

The implementation of the bump factorization in BASICLU follows mostly the description of Suhl and Suhl [74]. The active submatrix is held columnwise and additionally its pattern rowwise. A "file" data structure is used that allows fill-in by moving lines to the end of the file when necessary. The pivot element is chosen by a truncated Markowitz search that scans a limited number of rows and columns with a small nonzero count. In the elimination step the

pivot row and column are removed from the file and stored in the final factors, and the rank-1 outer product is added to the remaining submatrix. For the latter operation some implementation details have been adopted from the COIN-OR codes [11] `CoinFactorization[1-4].cpp` due to J. J. H. Forrest.

The pivot tolerance u is a user parameter. Because it is not known a priori how small u can be chosen, the stability of the factorization is tested a posteriori based on the method described in [16, Section 4.7]. For certain right-hand sides $\boldsymbol{b}$ and $\boldsymbol{c}$ the solutions to $B\boldsymbol{x} = \boldsymbol{b}$ and $B^T\boldsymbol{y} = \boldsymbol{c}$ are computed from the LU factors and the maximum of the scaled residuals

$$r_1 = \frac{\|\boldsymbol{b} - B\boldsymbol{x}\|_1}{\|\boldsymbol{b}\|_1 + \|B\|_1 \|\boldsymbol{x}\|_1}, \quad r_2 = \frac{\left\|\boldsymbol{c} - B^T\boldsymbol{y}\right\|_1}{\|\boldsymbol{c}\|_1 + \|B^T\|_1 \|\boldsymbol{y}\|_1}$$

is reported to the user. In general, a factorization is considered numerically stable if the scaled residuals are in order of machine precision for any right-hand sides (see Duff et al. [16, Chapter 4]). In practice the test can only be evaluated for one or two vectors. The vector $\boldsymbol{b}$ is constructed from L as in the LINPACK condition number estimate [16, Section 4.12]: it has components ± 1 and the signs are chosen during the course of solving $L\tilde{\boldsymbol{x}} = \boldsymbol{b}$ with the aim of making $\|\tilde{\boldsymbol{x}}\|_1$ large. The vector $\boldsymbol{c}$ is constructed likewise from $U^T\tilde{\boldsymbol{y}} = \boldsymbol{c}$ to make $\|\tilde{\boldsymbol{y}}\|_1$ large. The hope is that an unstable factorization will be revealed by the residual of either $\boldsymbol{b}$ or $\boldsymbol{c}$.

By default IPX sets the initial pivot tolerance to 0.0625. If the stability measure $\max(r_1, r_2)$ of a basis factorization is larger than 10^{-12}, the pivot tolerance for all subsequent factorizations is adjusted to 0.3 and the factorization is repeated. This occurs rarely in practice, but for some LP models it was necessary to obtain linear system solutions of sufficient accuracy.

7.2 The Forrest-Tomlin Update

Assume that $B = LU$ is the factorization into a unit lower triangular matrix L and an upper triangular matrix U, and let $\bar{B} = B + (\boldsymbol{a} - B\boldsymbol{e}_j)\boldsymbol{e}_j^T$ be the basis matrix after a column change. The Forrest-Tomlin (FT) update [18] computes a row eta matrix R and a symmetrically permuted triangular matrix $\bar{U}$ such that $\bar{B} = LR\bar{U}$, where a row eta matrix has the form

$$R = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ r_{j1} & \cdots & 1 & \cdots & r_{jm} & \\ & & & \ddots & & \\ & & & & 1 & \end{bmatrix}. \tag{7.2}$$

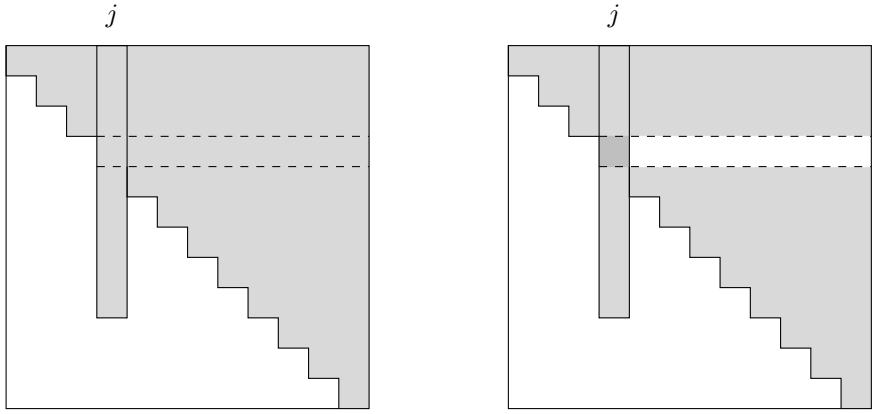
To derive formulas for R and $\bar{U}$, the expression for $\bar{B}$ is manipulated as follows (from [38]):

$$\begin{aligned} \bar{B} &= B + (\boldsymbol{a} - B\boldsymbol{e}_j)\boldsymbol{e}_j^T \\ \Longrightarrow\ L^{-1}\bar{B} &= U + \left(L^{-1}\boldsymbol{a} - U\boldsymbol{e}_j\right)\boldsymbol{e}_j^T \\ &=: U + (\hat{\boldsymbol{a}} - U_j)\,\boldsymbol{e}_j^T =: \hat{U}. \end{aligned}$$

$\hat{U}$ is a spiked upper triangular matrix as illustrated in Figure 6(a). The FT update eliminates the off-diagonal entries in row j of $\hat{U}$ by a row transformation. Let $\boldsymbol{w}^T$ be the row vector corresponding to row j of U without the diagonal entry and let $\boldsymbol{r}^T = \boldsymbol{w}^T U^{-1}$. Because $r_j = 0$, the matrix $R = I + \boldsymbol{e}_j\boldsymbol{r}^T$ has the form (7.2). Computing

$$R^{-1}\hat{U} = \left(I - \boldsymbol{e}_j\boldsymbol{r}^T\right)\hat{U} = \bar{U}$$

Figure 6:



(a) spike column in position j (b) after eliminating off-diagonals

yields a matrix $\bar{U}$ in which row j is a singleton row with diagonal entry $\bar{u}_{jj} = \hat{a}_j - \boldsymbol{r}^T\hat{\boldsymbol{a}}$. Assuming that $\bar{B}$ is nonsingular, $\bar{u}_{jj} \neq 0$. $\bar{U}$ is illustrated in Figure 6(b). It can be brought to upper triangular form by a cyclic permutation of rows and columns j to m.

Let B^r be the basis matrix after r column changes to B. Applying the FT update r times yields the factorization

$$B^r = LR^1 \cdots R^r U^r, \tag{7.3}$$

where each of the R^k is a row eta matrix and U^r is symmetrically permuted triangular. The permutation $\boldsymbol{p}$ that makes $U^r(\boldsymbol{p}, \boldsymbol{p})$ upper triangular is updated with each FT update by moving the index of the spike column to the end. The form (7.3) allows solving linear systems with B^r and its transpose by forward and backward substitution with L and U^r and application of the easily computed $(R^k)^{-1}$.

When the sequence of eta matrices becomes uneconomically long, usually after between 200 and 2000 updates (see Huangfu and Hall [38]), the basis matrix is refactorized. Because the FT update always pivots on the diagonal of U for eliminating off-diagonal entries in row j, the entries in the row eta matrices can become large. If this leads to numerical instability, refactorization is required earlier. Controlling the factorization frequency for stability and speed is discussed by Koberstein [43, Section 6.2.3] and Huangfu [37, Section 2.2.4].

7.3 Permutation to Triangular Form

While updating very sparse LU factors it frequently happens that the spiked matrix $\hat{U}$ is already permuted triangular. In this case the row operations of the FT update are unnecessary to restore triangularity. It will be shown how to efficiently find a permutation of $\hat{U}$ to triangular form if it exists. The method presented here was developed by author and described in [71]. The same idea was investigated at about the same time but independently by Fukasawa and Poirrier [20].

Some background from graph theory is required. The sparsity pattern of an $m \times m$ matrix B defines a directed graph $G = (V, E)$ with nodes $V = \{1, \ldots, m\}$ and edges $(i, j) \in E$ when $B_{i,j} \neq 0$. Self-edges for nonzero diagonal entries are included in E. A path $j_0 \rightsquigarrow j_k$ is a sequence of nodes $(j_0, \ldots, j_k)$ where an edge exists from each node to the next in the sequence. The nodes in the path other than j_0 and j_k are called "intermediate nodes". A cycle is a path $j \rightsquigarrow j$ with at least one intermediate node. (A self-edge is not a cycle.) $\text{Reach}_B(j)$ is the set of

all nodes i for which there exists a path $j \rightsquigarrow i$ in the directed graph of B. Note that j might not be contained in $\text{Reach}_B(j)$ if the self-edge is not present. A symmetric permutation of matrix B corresponds to a renumbering of the nodes in its graph G. B is symmetrically permuted triangular if and only if G is acyclic.

The following lemma shows how a column permutation changes the directed graph of a matrix.

Lemma 7.1. *Let B be an $m \times m$ matrix and Q be the permutation matrix that reorders columns $j_0, \ldots, j_n$ in a cycle; i. e. column j_{k+1} of B becomes column j_k of BQ for $0 \le k \le n$, where $j_{n+1} := j_0$. Denote $G = (V, E)$ the directed graph of B and $G_Q = (V, E_Q)$ the directed graph of BQ. Then for any node $i \in V$*

$$\begin{aligned} (i, j_k) \in E_Q &\iff (i, j_{k+1}) \in E \quad \text{for } 0 \le k \le n, \\ (i, j) \in E_Q &\iff (i, j) \in E \qquad \text{for } j \notin \{j_0, \ldots, j_n\}. \end{aligned}$$

For finding a permutation of a spiked matrix to triangular form, we have to distinguish whether or not the diagonal entry of the spike column is zero. Let $\hat{U}$ be obtained by inserting the vector $\hat{\boldsymbol{a}}$ into column j of the matrix U. For ease of notation it is assumed that U is upper triangular, but the methods also work if it is symmetrically permuted triangular.

7.3.1 Spike has a nonzero diagonal entry

We first consider the case $\hat{a}_j \neq 0$. Because U and $\hat{U}$ differ only in column j, and because entry (j, j) is nonzero in both matrices, $\text{Reach}_U(j) = \text{Reach}_{\hat{U}}(j)$. An example for this is shown in Figure 7.

In the following $\mathcal{U}_j$ denotes the set of indices of nonzeros in column j of U.

Theorem 7.2. *Let $\hat{U}$ have a zero-free diagonal and be upper triangular except for column j. Then $\hat{U}$ is permuted triangular if and only if*

$$\hat{\mathcal{U}}_j \cap \text{Reach}_{\hat{U}}(j) = \{j\}. \tag{7.4}$$

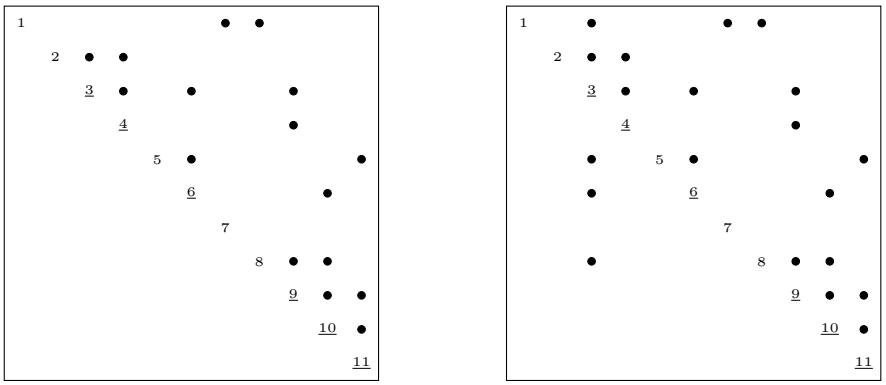
The permutation to triangular form is symmetric.

Proof. It is first shown that any permutation of a matrix with a zero-free diagonal to triangular form is symmetric. The proof is by contradiction. Assume that U has a zero-free diagonal and $U(\boldsymbol{p}, \boldsymbol{q})$ is upper triangular, where $\boldsymbol{p} \neq \boldsymbol{q}$ are permutation vectors. Without loss of generality let $p_1 \neq q_1$ (otherwise consider a submatrix of U). Because $U(\boldsymbol{p}, \boldsymbol{q})$ is upper triangular, column q_1 of U is a singleton. However, because U and $U(\boldsymbol{p}, \boldsymbol{q})$ both have a zero-free diagonal, the column has nonzeros in positions q_1 and p_1, which yields the contradiction.

Secondly it is shown that $\hat{U}$ is permuted triangular if and only if (7.4) holds. Let $G = (V, E)$ be the directed graph of $\hat{U}$. Then $\hat{U}$ is permuted triangular if and only if G is acyclic. Because only column j of $\hat{U}$ has entries below the diagonal, any cycle in G must contain node j. Such a cycle exists if and only if $i \in \text{Reach}_G(j)$ for some $i \neq j$ and $(i, j) \in E$. □

Figure 7(b) illustrates the theorem. Because the intersection of $\hat{\mathcal{U}}_3 = \{1, 2, 3, 5, 6, 8\}$ and $\text{Reach}(3) = \{3, 4, 6, 9, 10, 11\}$ is $\{3, 6\}$, $\hat{U}$ is not permuted triangular. Indeed, its graph contains the cycle $3 \rightsquigarrow 6 \rightsquigarrow 3$. On the other hand, if the entry in row 6 of the spike was not present, $\hat{U}$ would be permuted triangular.

To implement the triangularity test, $\text{Reach}_U(j)$ must be computed, usually by a graph traversal of U starting from node j (see, for example, [13, Section 3.2]). In some applications this can be omitted. When $\boldsymbol{e}_j^T B^{-1}$ has been computed before the update, then the (structural) nonzero pattern of $\boldsymbol{e}_j^T U^{-1}$ has been determined, which is $\text{Reach}_U(j)$. Therefore $\hat{\mathcal{U}}_j$ and



(a) original matrix U (b) $\hat{U}$ with spike in column 3

Figure 7: Example matrix U and spiked matrix $\hat{U}$. The dark circles represent off-diagonal nonzeros. The diagonal is assumed zero-free and numbered for easy reference. Nodes in Reach(3) are underlined.

$\text{Reach}_U(j)$ are available anyway and condition (7.4) can be tested at negligible cost. This applies, for example, to the revised simplex method.

When $\hat{U}$ is permuted triangular, the permutation to triangular form is obtained as follows.

Lemma 7.3. *Let $\hat{U}$ have a zero-free diagonal and be upper triangular except for column j. If $\hat{U}$ is permuted triangular, $\hat{U}(\boldsymbol{p}, \boldsymbol{p})$ is upper triangular, where $\boldsymbol{p}$ is obtained from the identity permutation $(1, \ldots, m)$ by removing the indices in $\text{Reach}_{\hat{U}}(j)$ and appending them in topological order to the end.*

Proof. A matrix is upper triangular if the nodes in its graph $G = (V, E)$ are numbered in topological order; i. e. if $(i, k) \in E$ only if $i \leq k$.

Let $\boldsymbol{p} = (p_1, \ldots, p_s, j, p_{s+2}, \ldots, p_m)$, where the sequence $(j, p_{s+2}, \ldots, p_m) = \text{Reach}_{\hat{U}}(j)$ is in topological order. Then the following holds true:

(i) The leading $s \times s$ block of $\hat{U}(\boldsymbol{p}, \boldsymbol{p})$ is upper triangular because it is the remainder after removing rows and columns from the spiked upper triangular matrix $\hat{U}$, including the spike column.

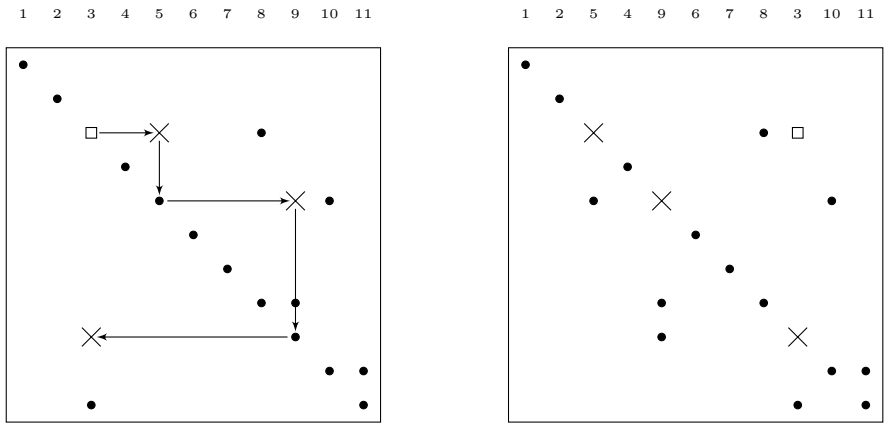
(ii) The trailing $(m-s)\times(m-s)$ block of $\hat{U}(\boldsymbol{p}, \boldsymbol{p})$ is upper triangular because $(j, p_{s+2}, \ldots, p_m)$ is in topological order with respect to $\hat{U}$.

(iii) There are no edges (p_l, p_k) in the graph of $\hat{U}$ for $1 \leq k \leq s$ and $s + 1 \leq l \leq m$, because otherwise p_k would be in $\text{Reach}_{\hat{U}}(j)$. Hence $\hat{U}(\boldsymbol{p}, \boldsymbol{p})$ has a zero block of dimension $(m - s) \times s$ below the diagonal.

Consequently, $\hat{U}(\boldsymbol{p}, \boldsymbol{p})$ is upper triangular. □

7.3.2 Spike has a zero diagonal entry

Secondly, we consider the case $\hat{a}_j = 0$. Provided that $\hat{U}$ is nonsingular, it cannot be the symmetric permutation of a triangular matrix (a symmetric permutation does not permute off-diagonal entries on the diagonal). It can be an unsymmetric permutation of a triangular matrix, however.



(a) augmenting path in $\hat{U}$ (b) columns $(3, 5, 9)$ cycled

Figure 8: Example matrix $\hat{U}$ and permutated to zero-free diagonal. Dark circles and crosses are nonzeros, the white square is a zero. Columns are numbered by their index in $\hat{U}$.

Lemma 7.4. *Let Q be a permutation matrix such that $\hat{U}Q$ has a zero-free diagonal. If $\hat{U}$ is the permutation of an upper triangular matrix U, then $\hat{U}Q$ is a symmetric permutation of U.*

Proof. Assume that $P_1\hat{U}P_2 = U$ for permutation matrices P_1, P_2. Then $P_1(\hat{U}Q)Q^TP_2 = U$ and because $\hat{U}Q$ has a zero-free diagonal, it follows from the proof of Theorem 7.2 that $P_1^T = Q^TP_2$. Hence $\hat{U}Q$ is indeed a symmetric permutation of U. □

It follows that testing if $\hat{U}$ is permuted triangular can be broken down into two steps. The first step is to find a column permutation Q that makes the diagonal of $\hat{U}Q$ zero-free. Such a permutation exists when $\hat{U}$ has full structural rank. The second step is to test if $\hat{U}Q$ is symmetrically permuted triangular. By Lemma 7.4 the result of step two is independent of the specific choice of Q.

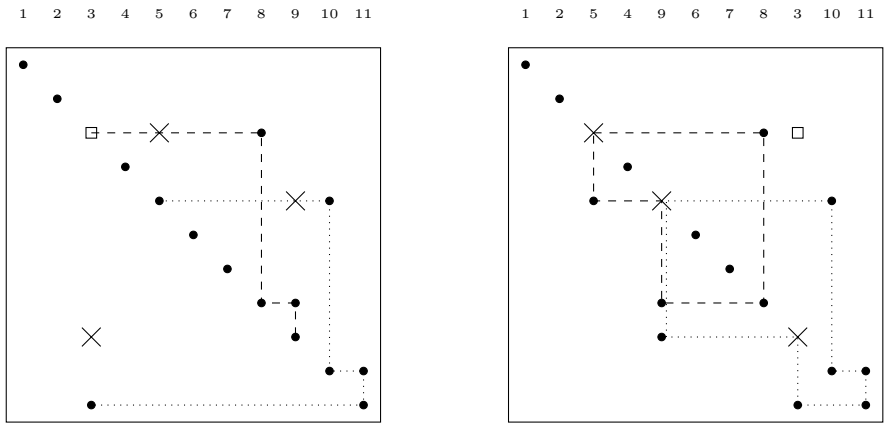
A column permutation Q that makes the diagonal of $\hat{U}Q$ zero-free is found by computing a maximum matching. A maximum matching pairs each row i with a column j such that entry (i, j) of $\hat{U}$ is nonzero and no row or column is paired twice. The diagonal of $\hat{U}$ already defines a partial matching in which each row i is matched to column i except for row j. This matching is increased by 1 by finding an alternating augmenting path [15]. In our setting, an alternating augmenting path is defined by a sequence of column indices $(j_0, j_1, \ldots, j_n)$, where $j_0 = j$ and entries (j_k, j_{k+1}) for $0 \leq k < n$ as well as (j_n, j_0) of $\hat{U}$ are nonzero. The permutation matrix Q which cycles columns $j_0, \ldots, j_n$ makes the diagonal of $\hat{U}Q$ zero-free. An alternating augmenting path is found, for example, by a depth-first search with look-ahead as described by Duff [15].

Figure 8(a) shows a spiked upper triangular matrix in which the spike has a zero diagonal entry. The arrows illustrate an alternating augmenting path through columns $(3, 5, 9)$. Figure 8(b) shows the matrix after a cyclic permutation of these columns.

The column permutation Q defined by the augmenting path $(j_0, \ldots, j_n)$ makes $\hat{U}Q$ a spiked upper triangular matrix with spikes in columns $j_0, \ldots, j_n$, see Figure 8(b). Theorem 7.2 easily adapts to a matrix with multiple spike columns:

Theorem 7.5. *Let $\bar{U}$ have a zero-free diagonal and be upper triangular except for columns $j_0, \ldots, j_n$. Then $\bar{U}$ is permuted triangular if and only if*

$$\bar{\mathcal{U}}_{j_k} \cap \mathrm{Reach}_{\bar{U}}(j_k) = \{j_k\} \quad \textit{for } 0 \leq k \leq n.$$



(a) example matrix $\hat{U}$ (b) columns $(3, 5, 9)$ cycled

Figure 9: Both paths in $\hat{U}$ become cycles after the column permutation.

Proof. This is shown analogously to the proof of Theorem 7.2 by using that any cycle in the directed graph of $\bar{U}$ must contain at least one of the nodes $j_0, \ldots, j_n$. □

Theorem 7.5 gives a simple characterization of whether $\hat{U}Q$ (and hence $\hat{U}$) is permuted triangular. It is inconvenient to implement, however, because computing $\mathrm{Reach}_{\bar{U}}(j_k)$ requires the pattern of the column permuted matrix $\bar{U} = \hat{U}Q$. In the implementation we would like to apply the column permutation only for an update, i. e. after determining that $\hat{U}$ is permuted triangular.

There is another characterization of whether $\hat{U}$ is permuted triangular given an augmenting path. Stated in the following theorem, this result requires more effort to derive, but is convenient to implement.

Theorem 7.6. *Let $\hat{U}$ be upper triangular except for column j_0, which has a zero diagonal entry and one or more nonzeros below. Let $(j_0, j_1, \ldots, j_n)$ be an augmenting path in $\hat{U}$ and $j_{n+1} := j_0$. Let $G' = (V, E')$ be the directed graph of $\hat{U}$ without the edges (j_k, j_{k+1}) for $0 \le k \le n$. Then $\hat{U}$ is permuted triangular if and only if*

$$j_{k+1}, \ldots, j_n \notin \mathrm{Reach}_{G'}(j_k) \quad \text{for } 0 \le k \le n-1, \tag{7.5a}$$

$$\mathrm{Reach}_{G'}(\{j_0, \ldots, j_n\}) \cap \hat{\mathcal{U}}_{j_0} = \{j_n\}. \tag{7.5b}$$

Before proving the theorem, it should be illustrated at an example. For the matrix in Figure 8(a) G' is the graph given by the pattern of dark circles. The crosses are the edges in the augmenting path which are excluded from G'. The matrix is plotted again in Figure 9. The dashed line in part (a) is a path $3 \rightsquigarrow 9$, meaning that $j_2 \in \mathrm{Reach}_{G'}(j_0)$. Hence condition (7.5a) is violated. It is seen in part (b) how this path together with the self-edges of diagonal entries of $\hat{U}$ becomes a cycle in the column permuted matrix, confirming that $\hat{U}$ is indeed not permuted triangular. Condition (7.5b) is also violated because $11 \in \mathrm{Reach}_{G'}(5)$ and $11 \in \hat{\mathcal{U}}_3$. The dotted line in Figure 9 shows how that path in G' becomes a cycle in the column permuted matrix.

For the following proof of Theorem 7.6 some quantities are fixed: Q is the permutation matrix corresponding to the augmenting path $(j_0, \ldots, j_n)$, $j_{n+1} = j_0$, $G_Q = (V, E_Q)$ is the directed graph of $\hat{U}Q$, and $G' = (V, E')$ is defined in Theorem 7.6.

Lemma 7.7. *For $0 \le k < l \le n$ there exists a path $j_l \rightsquigarrow j_k$ in G_Q.*

Proof. For $0 \leq k \leq n-1$, (j_{k+1}, j_{k+1}) is a self-edge in $\hat{U}$. By Lemma 7.1 (j_{k+1}, j_k) is an edge in G_Q. Consequently, for $0 \leq k < l \leq n$ there is a path $j_l \rightsquigarrow j_k$ in G_Q. □

Lemma 7.8. *If there exists a path $j_k \rightsquigarrow j_l$ in G' for some $k \neq l$, then G_Q contains a cycle.*

Proof. Assume that for some $k \neq l$ there is a path $j_k \rightsquigarrow j_l$ in G'. We can assume without loss of generality that the intermediate nodes do not contain any node of $\{j_0, \ldots, j_n\}$.

If $l = k+1$, the path contains at least one intermediate node j because (j_k, j_{k+1}) is not an edge in G'. Hence $j_k \rightsquigarrow j \rightsquigarrow j_{k+1}$ is a path in G'. It follows from Lemma 7.1 that these edges form paths $j_k \rightsquigarrow j$ and $j \rightsquigarrow j_k$ in G_Q, so that G_Q contains a cycle.

If $l \neq k+1$, we can say that $k+2 \leq l \leq n+1$ because $\hat{U}$ is upper triangular except for column j_{n+1}. It follows from Lemma 7.1 that $j_k \rightsquigarrow j_{l-1}$ is a path in G_Q. Because by Lemma 7.7 $j_{l-1} \rightsquigarrow j_k$ is also a path in G_Q, G_Q contains a cycle. □

Proof of Theorem 7.6. Because the diagonal of $\hat{U}Q$ is zero-free, $\hat{U}$ is permuted triangular if and only if G_Q is acyclic. It will be shown that G_Q is acyclic if and only if (7.5a) and (7.5b) hold.

"$\Longrightarrow$" For the "only if" implication assume that (7.5a) or (7.5b) are violated. If (7.5a) is violated, then there exists a path $j_k \rightsquigarrow j_l$ in G' for some $l > k$. By Lemma 7.8 G_Q contains a cycle.

Now assume that (7.5b) is violated. Then there exists an index $j \neq j_n$, $j \in \hat{\mathcal{U}}_{j_0}$, and some $0 \leq k \leq n$ such that $j \in \mathrm{Reach}_{G'}(j_k)$. We have to distinguish two cases:

(i) If $j = j_k$, then $(j_k, j_0) \in E'$ and by Lemma 7.1 $(j_k, j_n) \in E_Q$. Because $j_n \rightsquigarrow j_k$ is a path in G_Q by Lemma 7.7, G_Q contains a cycle.

(ii) If $j \neq j_k$, we can assume that $j \notin \{j_0, \ldots, j_n\}$ because otherwise Lemma 7.8 applies. We can also assume without loss of generality that $j_k \rightsquigarrow j$ is a path in G' whose intermediate nodes do not contain any node of $\{j_0, \ldots, j_n\}$. By Lemma 7.1 these edges form a path $j_k \rightsquigarrow j$ in G_Q. Because $(j, j_0) \in E'$ we have that $(j, j_n) \in E_Q$ and therefore $j_k \rightsquigarrow j \rightsquigarrow j_n$ is a path in G_Q. On the other hand $j_n \rightsquigarrow j_k$ is also a path in G_Q by Lemma 7.7. Hence G_Q contains a cycle.

"$\Longleftarrow$" It is shown that if G_Q contains a cycle, then (7.5a) or (7.5b) are violated. It must be distinguished whether or not j_n is in the cycle.

(i) Assume that there is a cycle in G_Q containing node j_n and let (j, j_n) be an edge in the cycle. Then $j \in \mathrm{Reach}_{G_Q}(j_n)$. Because

$$\mathrm{Reach}_{G_Q}(j_n) \overset{Lemma\ 7.7}{=} \mathrm{Reach}_{G_Q}(\{j_0, \ldots, j_n\}) = \mathrm{Reach}_{G'}(\{j_0, \ldots, j_n\}),$$

we have that $j \in \mathrm{Reach}_{G'}(\{j_0, \ldots, j_n\})$. On the other hand, because (j, j_n) is a nonzero entry in $\hat{U}Q$, we have that $j \in \hat{\mathcal{U}}_{j_0}$. It follows that (7.5b) is violated.

(ii) Assume that there is a cycle in G_Q not containing node j_n. Let $0 \leq k \leq n-1$ be the smallest k such that node j_k is in the cycle.

If another node j_l is in the cycle, we can assume without loss of generality that $k+1 \leq l \leq n-1$ is such that the intermediate nodes in the path $j_k \rightsquigarrow j_l$ do not contain any node of $\{j_0, \ldots, j_n\}$. By Lemma 7.1 $j_k \rightsquigarrow j_{l+1}$ is a path in G', violating (7.5a).

On the other hand, assume that there is no node of $\{j_0, \ldots, j_n\}$ in the cycle other than j_k. Let (j, j_k) be an edge in the cycle. Hence $j_k \rightsquigarrow j \rightsquigarrow j_k$ is a path in G_Q. By Lemma 7.1 these edges form a path $j_k \rightsquigarrow j \rightsquigarrow j_{k+1}$ in G', violating (7.5a).

□

When $\hat{U}$ is permuted triangular, the permutation to triangular form is obtained as follows.

Lemma 7.9. *Let $\hat{U}Q$ have a zero-free diagonal and be upper triangular except for columns $j_0, \ldots, j_n$. Let G_Q be the directed graph of $\hat{U}Q$. When $\hat{U}Q$ is permuted triangular, $(\hat{U}Q)(\boldsymbol{p}, \boldsymbol{p})$ is upper triangular, where $\boldsymbol{p}$ is obtained from the identity permutation $(1, \ldots, m)$ by removing the indices in $\mathrm{Reach}_{G_Q}(\{j_0, \ldots, j_n\})$ and appending them in topological order to the end.*

Proof. Analogous to the proof of Lemma 7.3. □

7.4 Implementing the Update

BASICLU combines the permutation method from the previous section with the FT update to maintain the factorization

$$B = LR^1 \cdots R^r U \tag{7.6}$$

after column changes to B. Here L is the symmetric permutation of a unit lower triangular matrix, the R^k are row eta matrices of the form (7.2) and U is permuted triangular. Note that the columns of L are indexed such that column i was computed in the elimination step of the LU factorization in which row i was pivot row. Accordingly, the row indices of U are (unpermuted) row indices of B. This form is preferred over using permuted indices because updating U by an unsymmetric permutation changes the row-column pairings defined by the pivot elements.

The factorization is stored by the following combination of data structures. Names starting with capitals are variables of a composed data type, whereas lower case names denote arrays.

`pmap, qmap` are integer arrays of length m such that $i = \texttt{pmap}[j]$ and $j = \texttt{qmap}[i]$ if entry (i, j) of U is a pivot element. Consequently $U(\texttt{pmap}, \cdot)$ and $U(\cdot, \texttt{qmap})$ are symmetrically permuted triangular.

`Lcols, Lrows` hold the matrix L in compressed column and compressed row format, respectively. The unit diagonal entries are not stored. The data structures are set up after factorization and are not modified by updates.

`Rrows` stores for each eta matrix R^k the index of the nontrivial row and its off-diagonal entries. The data structure starts out empty after factorization and entries are accumulated by FT updates.

`Ucols` stores the matrix $U(\cdot, \texttt{qmap})$ columnwise without the diagonal entries. When a column of U is replaced, it is zeroed out in `Ucols` and the new column is stored at the end of the memory space.

`Urows` stores the matrix $U(\texttt{pmap}, \cdot)$ rowwise without the diagonal entries. The rows are held in a file data structure that allows replacing one column of U at a time.

`colpivots, rowpivots` are real arrays of length m that hold two copies of the pivot elements, one copy by column indices and one by row indices. That means, $\texttt{colpivots}[j]$ is entry $(\texttt{pmap}[j], j)$ of U and $\texttt{rowpivots}[i]$ is entry $(i, \texttt{qmap}[i])$ of U.

The rowwise and columnwise representation of L and U is required for solving all variants of sparse triangular systems, which is inevitable for hypersparse LP problems. Notice that the columns stored in `Ucols` are indexed by row indices of B, and the rows stored in `Urows` are indexed by column indices of B. The benefit of this seemingly odd idea is that solving

$U\boldsymbol{x} = \boldsymbol{b}$ and $U^T\boldsymbol{x} = \boldsymbol{b}$ for a sparse $\boldsymbol{b}$ by the method of Gilbert and Peierls [21] does not require indirection through `pmap` or `qmap` during the depth-first search.

The update of the factorization (7.6) is stated in Algorithm 11 without considering the manipulation of data structures except for `pmap` and `qmap`. The high-level description lets us focus on the method rather than technical details, but it is not difficult to see that the operations fit naturally to the data structures above.

A factorization update is called "symmetric" when the diagonal entry of the spike is nonzero. Lines 11–13 of Algorithm 11 implement condition (7.4) to test triangularity in the symmetric case. The implementation implicitly works on the matrix $U(\cdot, \texttt{qmap})$, which is symmetrically permuted triangular before inserting the spike into column i_0. It is exploited that $\boldsymbol{r}^T$ was computed before, whose nonzero pattern is $\text{Reach}(i_0) \setminus \{i_0\}$.

The triangularity test for the unsymmetric case is implemented in lines 18–28. It works on the matrix $U(\texttt{pmap}, \cdot)$, which is symmetrically permuted triangular before inserting the spike into column j_0. The rowwise storage of this matrix allows to traverse the out-adjacency list of each node in its graph. The augmenting path $(j_0, \ldots, j_n)$ is computed by a breadth-first search because it gives a path of minimum length and thereby reduces the work in the subsequent loop. The reaches in lines 21 and 24 are determined by depth-first searches. To operate on the graph G', the edges (j_k, j_{k+1}) are temporarily replaced by self-edges (j_k, j_k) in `Urows`.

The FT update and the update by symmetric permutation do not change `pmap` or `qmap`. In the unsymmetric case `pmap` is updated by a cyclic permutation of positions $(j_0, \ldots, j_n)$ and `qmap` is updated as its inverse permutation (lines 32–33 in Algorithm 11).

So far, we have ignored the permutations $\boldsymbol{p}$ and $\boldsymbol{q}$ that make $U(\boldsymbol{p}, \boldsymbol{q})$ upper triangular. Notice that $\boldsymbol{p}$ and $\boldsymbol{q}$ are not needed for updating the factorization or solving sparse triangular systems. They are required, however, for solving $U\boldsymbol{x} = \boldsymbol{b}$ or $U^T\boldsymbol{x} = \boldsymbol{b}$ for a dense $\boldsymbol{b}$, since they determine in which order the columns or rows of U must be processed. Because an LU package for general-purpose linear programming needs to allow efficient solution of linear systems with a dense right-hand side, it is necessary to maintain $\boldsymbol{p}$ and $\boldsymbol{q}$.

After a fresh factorization $\boldsymbol{p}$ and $\boldsymbol{q}$ are stored as vectors `p` and `q` of length m. As explained in Section 7.3, each update requires moving some indices to the end of the vectors. Because shifting indices cannot be done efficiently, the implementation only appends these indices to the end of `p` and `q`, thereby generating duplicates and increasing the vector length. When the length becomes too large, say $2m$, garbage collection is performed.

Solving $U\boldsymbol{x} = \boldsymbol{b}$ for a dense $\boldsymbol{b}$ is outlined in Algorithm 12. The solve is carried out in a work array of length m that initially holds $\boldsymbol{b}$. When an index is retrieved from `p` for the first time, the computation of its solution component is completed by dividing by the pivot element. The work array is updated by subtracting a multiple of the corresponding column of `Ucols` (which are indexed by row indices), and the solution value is stored in an output array. Its position in the work array is set to zero. If an index occurs multiple times in `p`, then only at the first occurence is its value in the work array (possibly) nonzero. In all later occurences, the component of the work array is zero and the loop iteration is skipped.

Solving $U^T\boldsymbol{x} = \boldsymbol{b}$ proceeds similarly using `Urows`, `colpivots` and by traversing `q` from the end. For triangular solves with L and a dense right-hand side, the permutation that brings L to triangular form is available from the factorization. Note that in any case the columns or rows of L or U are not accessed in memory order. Hence the solves are slower than with matrix factors stored with permuted indices.

BASICLU enables solving each of the systems $L\boldsymbol{x} = \boldsymbol{b}$, $L^T\boldsymbol{x} = \boldsymbol{b}$, $U\boldsymbol{x} = \boldsymbol{b}$ and $U^T\boldsymbol{x} = \boldsymbol{b}$ with a dense or sparse right-hand side. When the user requests solving $B\boldsymbol{x} = \boldsymbol{b}$ or $B^T\boldsymbol{x} = \boldsymbol{b}$ with a dense right-hand side, then both triangular solves are performed dense. When the user inputs a sparse right-hand side, then the first triangular solve is performed sparse, and the second triangular solve is performed either sparse or dense, depending on if the (intermediate)

Algorithm 11 Combined LU update

Input: $B = LR^1 \cdots R^r U$, `pmap`, `qmap`, vector $\boldsymbol{a}$ to replace column j_0 of B

Prepare update
$i_0 = \texttt{pmap}[j_0]$
Let $\boldsymbol{w}^T$ be row i_0 of U without the entry in position j_0.
Compute $\boldsymbol{r}^T = \boldsymbol{w}^T U^{-1}$.
Compute $\hat{\boldsymbol{a}} = (R^r)^{-1} \cdots (R^1)^{-1} L^{-1} \boldsymbol{a}$.
Replace column j_0 of U by $\hat{\boldsymbol{a}}$.
Test triangularity
$\texttt{istriangular} = true$
if $\hat{a}_{i_0} \neq 0$ **then** // spike has nonzero diagonal entry
    $\texttt{issymmetric} = true$
    for all i for which $\hat{a}_i \neq 0$ **do**
        if $r_i \neq 0$ **then**
            $\texttt{istriangular} = false$
else // spike has zero diagonal entry
    $\texttt{issymmetric} = false$
    Find augmenting path $(j_0, \ldots, j_n)$ in $U(\texttt{pmap}, \cdot)$.
    $j_{n+1} = j_0$
    Let G' be the directed graph of $U(\texttt{pmap}, \cdot)$ without the edges (j_k, j_{k+1}), $0 \leq k \leq n$.
    Assume that all nodes are unmarked.
    for $k = 0$ to $n - 1$ **do**
        Mark nodes in $\text{Reach}_{G'}(j_k)$.
        if node j_{k+1} is marked **then**
            $\texttt{istriangular} = false$
    Mark nodes in $\text{Reach}_{G'}(j_n)$.
    for all i for which $\hat{a}_i \neq 0$ **do**
        $j = \texttt{qmap}[i]$
        if $j \neq j_n$ and node j is marked **then**
            $\texttt{istriangular} = false$
Apply update
if `istriangular` **then** // spiked matrix is permuted triangular
    if not `issymmetric` **then**
        $\texttt{pmap}[j_{k+1}] = \texttt{pmap}[j_k]$ for $0 \leq k \leq n$. // simultaneous assignment
        $\texttt{qmap}[\texttt{pmap}[j_k]] = j_k$ for $0 \leq k \leq n$.
else // FT update
    Compute pivot element $\alpha = \hat{a}_{i_0} - \boldsymbol{r}^T \hat{\boldsymbol{a}}$.
    Replace row i_0 of U by a singleton row with nonzero α in position j_0.
    Append row eta matrix R^{r+1} with off-diagonals in row i_0 given by $\boldsymbol{r}^T$.

Algorithm 12 Solving $U\boldsymbol{x} = \boldsymbol{b}$ for a dense $\boldsymbol{b}$

Require real arrays `x` and `w` of size m;
`x` is for output, `w` is workspace.
1: Copy $\boldsymbol{b}$ into `w` and initialize `x` to zero.
2: **for** $k = \mathit{length}(\mathtt{p})$ to 1 backwards **do**
3: $\quad ipivot = \mathtt{p}[k]$
4: $\quad$ **if** $\mathtt{w}[ipivot] \neq 0$ **then**
5: $\quad\quad \mathtt{w}[ipivot] = \mathtt{w}[ipivot]/\mathtt{rowpivots}[ipivot]$
6: $\quad\quad$ **for each** i for which $\mathtt{Ucols}[i, ipivot] \neq 0$ **do**
7: $\quad\quad\quad \mathtt{w}[i] = \mathtt{w}[i] - \mathtt{Ucols}[i, ipivot] \cdot \mathtt{w}[ipivot]$
8: $\quad\quad \mathtt{x}[\mathtt{qmap}[ipivot]] = \mathtt{w}[ipivot]$
9: $\quad\quad \mathtt{w}[ipivot] = 0$

right-hand side has more than $0.05m$ nonzeros.

7.5 Benchmark on Simplex Bases

The combined update method from Algorithm 11 was compared to the pure FT update on sequences of simplex bases. The test set were the 30 LP models used by Huangfu and Hall [38], which represent a wide range of structural properties. For each model the pivot sequence generated by the dual simplex method of CPLEX 12.7 [12] was recorded. CPLEX was run without presolve, starting from the slack basis and using dual steepest edge pricing; the other parameters were defaults. Problem `Linf_520c` was removed from the test set because it was not solved within 12 hours. The remaining problems are listed in Table 8 with the dimension of the basis matrices ("rows") and their average bump size ("bump").

For each basis matrix B in the sequence, BASICLU had to solve the forward and backward systems

$$B\boldsymbol{x} = \boldsymbol{a}, \quad B^T\boldsymbol{y} = \boldsymbol{e}_j,$$

where $\boldsymbol{a}$ was the vector that would replace column j of B. The routine `solve_for_update` was used, which performs the linear system solve and stores the spike or row eta in preparation for the update. To obtain results for the pure FT update, the triangularity test in Algorithm 11 was simply skipped. In both cases the refactorization frequency was chosen as suggested by BASICLU, which measures the cost of operations with the accumulated row eta matrices relative to the cost of the last factorization. Refactorization due to numerical instability was ignored because there should be no difference between the permuted update and the FT update.

The fraction of updates that could be permuted (symmetrically permuted) to triangular form is given in column "perm (sym)" of Table 8. The following two block columns show the number of factorizations (n_f), factorization time (t_f), solve time (t_s) and update time (t_u) (in seconds) for the combined update and the pure FT update. The last column shows the ratio of the total computation time using the combined update to that of the FT update; a value < 1 means that the combined update was faster.

Unsurprisingly, there is a clear correlation between the fraction of updates that could be permuted and a small bump size. This makes the permutation method particularly relevant for large-scale LP problems, where the bump is typically a small part of the basis matrix. It is also seen that unsymmetric permutations occur frequently, disqualifying the idea to exploit only symmetric permutations, which would allow a much simpler implementation.

A more visible illustration of the results is given in Figure 10. In both plots the abscissa holds the fraction of updates that could be permuted and each cross represents an LP model, with its ordinate as the problem dimension (upper plot) or the ratio from the last column of Table 8

(lower plot). For all of the large-scale problems (say $m > 10^5$) more than 80% of the updates could be permuted. In these cases the reduction of the computation time ranged between 25% and 70%. If only a minor fraction of updates could be permuted, the triangularity test caused up to 5% overhead. The latter figure should be considered with care, however, because in this case our implementation of the FT update is not efficient. Here better performance could be achieved by storing the factors with permuted indices, since then data structures are traversed in memory order during forward and backward substitution. Such an implementation is not possible in combination with the update by permutation, because an unsymmetric permutation changes the row–column mappings defined by the pivot elements (`pmap` and `qmap` above).

name	rows	bump	perm (sym)	combined				Forrest-Tomlin				ratio
				n_f	t_f	t_s	t_u	n_f	t_f	t_s	t_u	
cre-b	9648	137	0.94 (0.04)	34	0.04	0.21	0.03	49	0.05	0.31	0.02	0.74
dano3mip_lp	3202	1116	0.28 (0.01)	223	0.48	3.03	0.15	223	0.48	3.06	0.08	1.02
dcp2	32388	3331	0.41 (0.34)	39	0.33	1.57	0.06	41	0.34	1.66	0.05	0.96
dfl001	6071	2319	0.53 (0.16)	103	0.32	3.54	0.09	108	0.32	3.62	0.05	0.99
fome12	24284	9201	0.52 (0.15)	237	3.18	19.60	0.49	246	3.16	20.46	0.25	0.97
gen4	1537	243	0.02 (0.01)	7	0.13	0.40	0.05	7	0.14	0.41	0.02	1.05
ken-18	105127	5	0.99 (0.72)	14	0.26	0.84	0.24	143	2.42	2.25	0.08	0.28
l30	2701	2299	0.04 (0.03)	102	0.51	1.83	0.13	102	0.52	1.83	0.04	1.03
lp22	2958	1381	0.17 (0.06)	143	0.79	2.59	0.21	144	0.81	2.53	0.07	1.06
maros-r7	3136	765	0.38 (0.29)	31	0.10	0.37	0.03	36	0.11	0.38	0.02	0.98
mod2	34774	2841	0.67 (0.28)	118	0.88	11.30	0.19	123	0.85	12.12	0.09	0.95
ns1688926	32768	70	0.51 (0.48)	18	0.29	6.49	0.21	18	0.29	6.43	1.33	0.87
osa-60	10280	4	0.96 (0.02)	16	0.01	0.15	0.01	23	0.02	0.31	0.04	0.49
pds-20	33874	146	0.97 (0.42)	21	0.08	0.45	0.06	76	0.24	0.56	0.02	0.73
pds-40	66844	536	0.96 (0.42)	50	0.49	3.11	0.23	152	1.24	4.05	0.07	0.71
pds-100	156243	654	0.97 (0.41)	90	2.32	8.72	0.91	337	7.31	10.80	0.20	0.65
pilot87	2030	991	0.13 (0.10)	84	0.89	2.10	0.24	84	0.88	2.12	0.22	1.01
qap12	3192	2708	0.05 (0.00)	1170	29.70	60.53	5.41	1176	30.39	62.30	1.86	1.01
self	960	790	0.02 (0.01)	136	13.68	4.40	0.94	138	14.08	4.96	1.04	0.95
sgpf5y6	246077	275	0.99 (0.17)	38	1.57	2.41	0.40	182	6.69	6.92	0.15	0.32
stat96v1	5995	4875	0.14 (0.13)	99	0.63	4.68	0.13	100	0.62	4.60	0.06	1.03
stat96v4	3173	2764	0.19 (0.18)	333	2.26	9.23	0.36	335	2.27	9.23	0.26	1.01
stormg2-125	66185	529	0.94 (0.26)	45	0.44	0.42	0.08	103	0.98	0.86	0.05	0.50
stormg2_1000	528185	3646	0.95 (0.22)	127	14.78	18.34	1.01	308	33.39	31.61	0.70	0.52
stp3d	159488	3302	0.84 (0.31)	120	3.26	45.10	0.74	129	3.26	53.10	0.43	0.86
truss	1000	700	0.29 (0.10)	181	0.11	0.47	0.04	186	0.12	0.46	0.03	1.02
watson_1	201155	1904	0.93 (0.53)	114	4.28	2.62	0.22	156	5.67	4.67	0.18	0.68
watson_2	352013	2575	0.94 (0.58)	129	8.71	4.36	0.32	176	11.37	8.04	0.26	0.68
world	34506	2467	0.70 (0.28)	127	0.86	10.23	0.17	134	0.86	11.77	0.10	0.89

Table 8: Comparison of LU update methods.

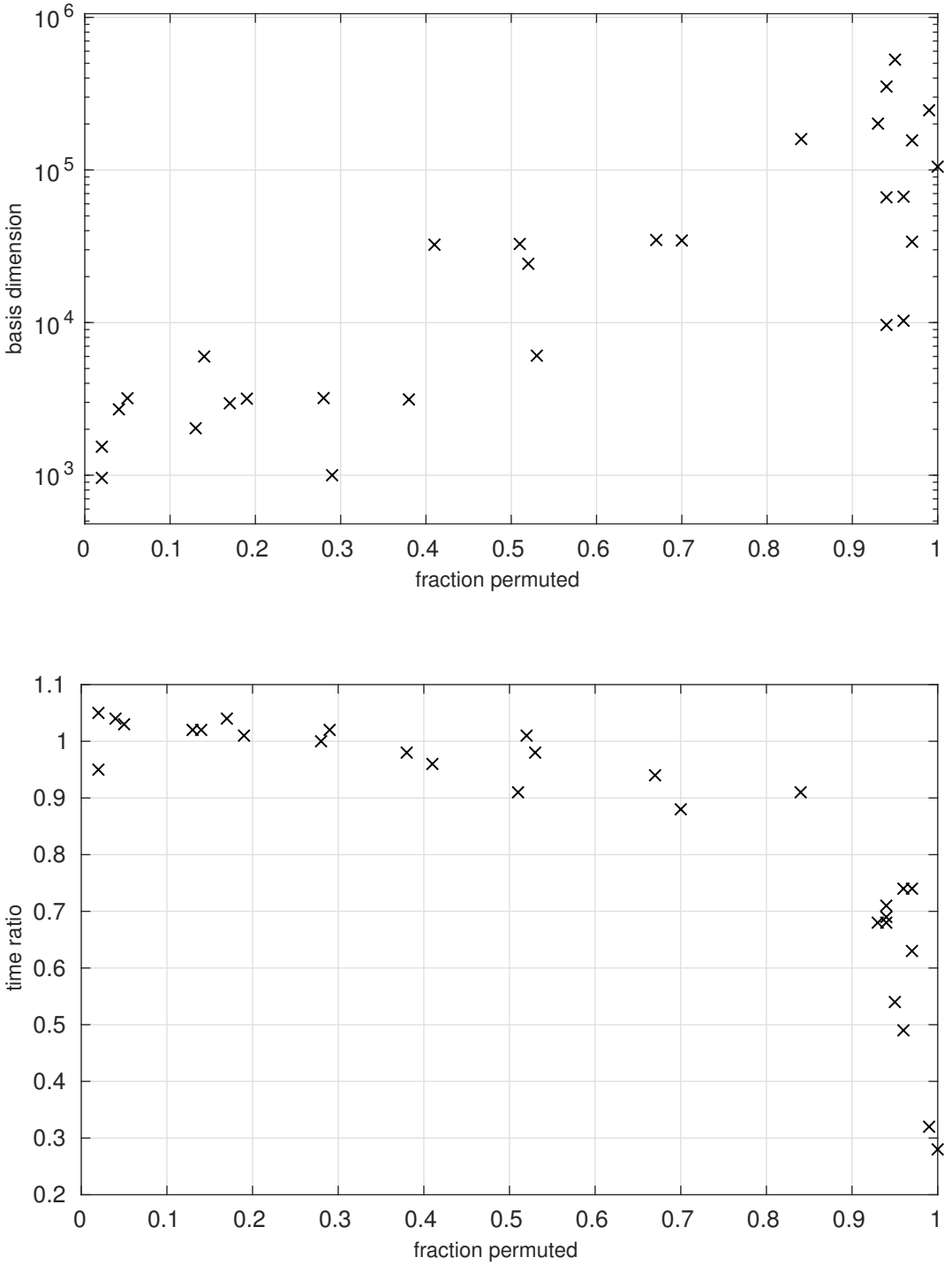


Figure 10: Visualization of data from Table 8.

8 Computational Results

The goal of this chapter is to provide a comprehensive comparison of the newly developed LP solver to the current state-of-the-art. In the first part the performance and robustness of IPX are evaluated on a diverse set of LP models, using a Cholesky-based IPM and a dual simplex as reference. In the second part the advantage of the new approach on two specific problem classes is discussed.

8.1 Test Environment

The computational results were obtained from IPX version 1.0, which was the code version at the time of submitting this thesis. BASICLU version 2.0 was used for the basis factorizations and updates. Both packages are publicly available from `https://www.maths.ed.ac.uk/ERGO/software.html`. For comparison the interior point ("barrier") and dual simplex solver of the commercial software Gurobi [32] version 7.0 were used.

The tests were run on a desktop computer with an Intel i5-6500 CPU (4 cores, 3.2GHz, 6MB L3 cache) and 8GB physical memory. While it is clear that the Cholesky factorization would have benefitted from hardware with higher floating point capability more than the iterative method, the setup was chosen because it is frequently used by practitioners. IPX and BASICLU were compiled with GCC version 4.8.5 and optimization flag `-O2`. IPX requires the LAPACK [48] routines `dpotrf` and `dpotrs` for the dense Cholesky factorization and solves, which were used with the OpenBLAS library [63].

8.2 A Diverse Problem Set

To compose a test set that represents a wide range of applications, all LP models from the sources listed in Appendix A were collected. For the mixed-integer problems the canonical LP relaxation was built, and a presolved version was generated for each model by the Gurobi LP presolve. Instances for which the resulting $\bar{m} \times \bar{n}$ constraint matrix was such that $\min(\bar{m}, \bar{n}) \leq 1000$ were removed, because for these models the normal equations (possibly after dualization) can be solved efficiently by dense linear algebra routines.

Both interior point solvers were then applied with default parameters and a time limit of $36,000$ seconds. IPX was run on the presolved models, whereas Gurobi was given the original models and its presolve time was subtracted from its total runtime. Infeasible and unbounded models, as well as models for which the Cholesky factorization required more than the 8GB physical memory were removed from the set. Models were also removed if they were solved by both methods within 1 second. The test set was finally cleaned by keeping only 1 or 2 instances of the same model (for example, from the 12 `pds` instances only `pds-60` and `pds-100` were kept). The resulting set contained 170 LP models that are listed with their solution times in Appendix A. An overview of their dimensions is given in Table 9.

IPX does not provide a simplex implementation for cleaning up the basic solution after the crossover push phases. For the study the final basis was used as starting basis for the Gurobi primal or dual simplex, depending on which infeasibility was smaller. In 150 cases Gurobi decided the initial solution to be optimal within its default tolerances. In 15 cases a simplex run was necessary and the time was added to the total IPX runtime.

All models in the test set were solved by either IPX or Gurobi barrier to basic solution. The Gurobi crossover reached time limit on 1 instance (`nug30`), whereas the IPX interior point method failed on 5 instances:

- On `stormg2_1000`, `ns2122603` and `ns1688926` the IPM stopped after no progress was achieved over a number of iterations. The issue seems to be solvable by a refined IPM

$\min(\bar{m}, \bar{n})$	instances
1,036–2,499	16
2,500–4,999	17
5,000–9,999	27
10,000–24,999	41
25,000–49,999	31
50,000–99,999	11
100,000–249,999	19
250,000–499,999	4
500,000–999,999	3
1,000,000–1,439,571	1

Table 9: Row and column dimensions $(\bar{m}, \bar{n})$ of 170 test problems after presolve.

subset	instances	IPX/Gurobi	IPX faster	Gurobi faster
>1s	164	3.67	22	142
>10s	89	3.94	16	73
>100s	41	6.29	6	35
>1s	102	4.93	7	95
>10s	59	5.85	7	52
>100s	35	7.95	4	31

Table 10: Runtime comparison on LP models that were solved by IPX and Gurobi barrier. The lower half excludes models for which the Gurobi dual simplex was faster than the Gurobi barrier.

implementation (for example using centrality correctors or a more conservative choice of step sizes). The models are questionable numerically, however, due to a wide range of entries in the problem data.

- On `cont1_l` and `cont11_l` the initial LU factorization ran out of memory. It turned out that the large fill-in was caused by the default pivot tolerance of 0.0625 being too small. For the related but smaller instances `cont1` and `cont11` IPX detected the initial LU factorization to be unstable and tightened the pivot tolerance to 0.3. In the repeated factorization the fill-in decreased by about a factor 4. After setting the initial LU pivot tolerance to 0.3, `cont1_l` was solved to an optimal basic solution in 3155 seconds (Gurobi required 1951 seconds); `cont11_l` reached time limit after 3 interior point iterations with basis preconditioning.

In the upper half of Table 10 the runtimes of IPX and Gurobi barrier are compared on the 164 models that were solved by both methods with default parameters. The column "IPX/Gurobi" shows the geometric mean of the runtime ratios, a value >1.0 meaning that IPX was by that factor slower. The subset ">10s" consists of the models for which at least one solver required more than 10 seconds. The last subset should be considered with care because 41 instances are insufficient to draw a conclusion. In the lower half of the table the comparison is repeated after removing the 62 models for which the Gurobi dual simplex was faster than the Gurobi barrier.

The results prove the robustness of the method implemented in IPX. It solved the vast majority of test problems and its average performance was in the same order of magnitude as that of the Cholesky-based solver. For large instances, say $m > 100,000$, the irregular memory access of the basis updates and the iterative linear solver became decisive. Here the Cholesky factorization often performed better as long as its floating point work was not excessive. This

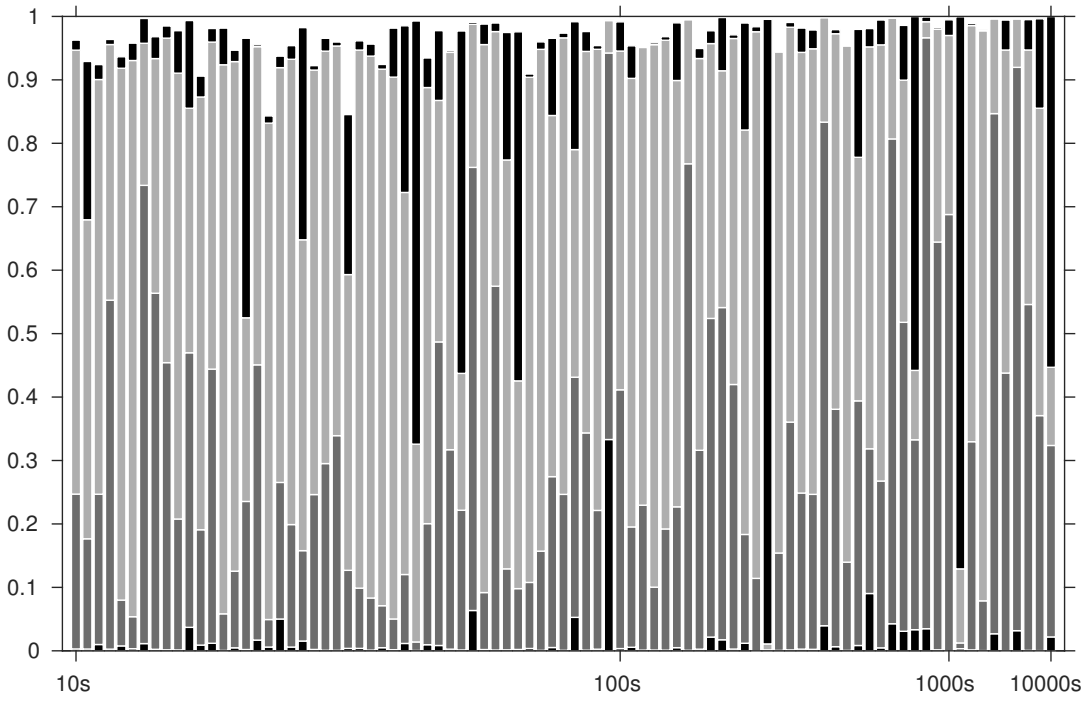


Figure 11: Fraction of total runtime spent for computing the crash basis (black bar at the bottom), for updating the basis during the interior point solve (dark grey), for running the linear solver (light grey), for crossover (black bar at the top) and for the remaining interior point algorithm (white areas). The 87 LP models are ordered on the x-axis by total runtime.

explains the increase in the performance ratios toward higher solution times. By comparing the two ">10s" subsets it is seen that about half of the models where IPX performed better than Gurobi barrier were solved more efficiently by the dual simplex; yet there remain relevant problems on which the new approach was superior to both conventional methods.

The breakdown of the total IPX runtime into different parts of the algorithm is illustrated in Figure 11 for the 87 models that IPX solved successfully but took longer than 10 seconds. Computing the starting basis was inexpensive except for `cont1`, where it accounted for one third of the total time. Here the issue was the large fill-in in the first *LU* factorization before tightening the pivot tolerance. On average 15% of the time for running the linear solver was spent in the initial IPM iterations (not shown separately). Taking geometric means, preparing the preconditioner and running the iterative solver accounted for 22% and 45% of the total time, respectively. Crossover took 2% on average, but dominated the IPM runtime on 7 models.

8.3 Specific Problem Classes

The results from the previous section indicated that the method implemented in IPX can be superior to both a Cholesky-based IPM and a dual simplex. Two LP models where this was the case and where instances of different dimensions were available should be discussed.

The first example is a planning model from the petroleum industry that was formulated by Alabi and Castro [5]. Table 11 provides the dimensions of the presolved models for 60, 120, 180, 240 and 300 planning days, and statistics from the three solvers. The models are hypersparse and `srd300` can be considered large-scale by today's standards.

name	$\bar{m}$	$\bar{n}$	nnz($\bar{A}$)	Cholesky	simplex	IPX
srd060	93,200	178,510	2,796,210	400MB	198,614	70,150
srd120	186,440	357,070	9,804,510	1.5GB	411,601	139,966
srd180	279,680	535,630	21,024,810	3.0GB	667,702	207,292
srd240	372,920	714,190	36,457,110	5.0GB	957,958	296,348
srd300	466,160	892,750	56,101,410	8.0GB	1,182,170	366,304
nug12	2,794	8,771	33,443	16MB	35,033	1,995
nug15	5,698	22,230	85,425	60MB	168,806	6,334
nug20	14,098	72,546	281,906	300MB	563,761	14,128
nug30	52,260	379,350	1,567,800	4.0GB	$> 171,866$	30,759

Table 11: Dimensions of presolved models, factor size of the Gurobi Cholesky factorization, iteration count of the Gurobi dual simplex, and number of basis updates in IPX with crossover.

name	IPX			Gurobi barrier			simplex
	total	IPM	basis	total	IPM	basis	
srd060	74.2	73.5	0.7	59.8	57.5	2.3	27.7
srd120	329.5	327.4	2.1	496.5	484.9	11.6	1048.3
srd180	891.4	886.8	4.6	1459.9	1435.8	24.1	3199.7
srd240	1883.3	1877.2	6.1	3615.5	3577.2	38.3	8483.8
srd300	3477.3	3467.1	10.2	t	t		14465.7
nug12	2.4	2.2	0.2	0.6	0.5	0.2	16.4
nug15	15.1	13.3	1.8	4.0	2.8	1.2	238.5
nug20	192.3	176.1	16.1	435.5	37.5	398.0	3515.0
nug30	11473.2	5125.2	6348.0	t	1400.5	t	t

Table 12: Computation times in seconds. "t" means 36,000 seconds time limit reached.

The second example are LP relaxations of quadratic assignment problems (QAPs), which are commonly used for modelling facility location problems. The nug models in Table 11 originated from the binary LP formulations of Resende et al. [67] of the facility location problems from Nugent et al. [60] with 12, 15, 20 and 30 facilities. Although their dimensions are moderate, the models are known to be challenging for both Cholesky factorization and simplex.

The solution times are given in Table 12. On the srd models, excluding the smallest one, IPX was about 4 times faster than the dual simplex. This ratio is surprising because the difference in the number of basis updates was only a factor 3 and we would have expected the simplex to be more efficient in terms of time per basis update. The Gurobi barrier performed somewhere between the other two solvers except for the largest instance, where the use of swap memory degraded its floating point performance. The fill-in in the Cholesky factorization was about a factor 10 in all cases.

On the nug models the number of dual simplex iterations and the time per iteration increased rapidly with the problem dimension. This also applied for the simplex clean-up in the Gurobi crossover. For nug30 Gurobi required about 3400 seconds for constructing a crossover starting basis, followed by 12500 seconds for the push phases. The resulting basis was largely primal and dual infeasible and the simplex clean-up reached time limit after $190,000$ iterations. The basis returned by the IPX crossover was almost optimal and was cleaned up by the primal simplex in 1 iteration. On these models the ability of IPX to compute an interior solution that is close-to complementary is decisive.

On both models and all instances IPX required roughly $\bar{m}$ basis updates including crossover, which is considerably smaller than the common $3\bar{m}$ target for the simplex.

9 Conclusions

The computational results demonstrated that the developed method is a robust, general-purpose LP solver, that on average performs in the same order of magnitude as the best Cholesky-based codes. To the author's knowledge this was not achieved by previous approaches of iterative linear algebra in IPMs. The key ingredients in the present work were the maximum volume criterion for choosing a basis preconditioner and the update scheme for maintaining such a basis at reasonable cost.

Regarding performance of the IPM, it is clear that the new approach is not a replacement for direct linear algebra. On the majority of LP models from a diverse test set the Cholesky factorization was faster, in particular for large-scale instances. However, there remained relevant problems on which IPX performed better than a Cholesky-based IPM and a dual simplex. For such applications the new solver is a true alternative to conventional methods. It might also be the preferred method when memory resources are limited or on computing hardware with low floating point capability. Its advantage over the simplex method is to require almost constantly no more than m basis updates in the IPM, where m is the number of rows of the LP model.

Basis preconditioning offers an advantage over other linear algebra kernels in combination with crossover. By eliminating degenerate variables during the interior point solve, it enables computing highly accurate solutions, which can be dropped to complementarity without causing a large primal or dual residual. In most cases the crossover push phases then yield an optimal basic solution. On the largest QAP model in our test set IPX was the only solver capable of computing a basic solution in acceptable time. The technique can also be added to conventional IPM implementations. Here, additional IPM iterations with basis preconditioning can be performed conveniently after constructing a crossover starting basis but before executing the push phases.

While the computational part of the thesis focussed on linear programming, the theoretical discussions indicated that basis preconditioning is suitable for solving least squares problems in general. Features like controlling the relative solution error, adding columns after an initial solve, detecting the numerical rank of a matrix (through a maximum volume basis) and computing the minimum norm least squares solution make it attractive in areas like parameter estimation and regression analysis. Because in these applications typically a single linear system needs to be solved and there are no scaling factors involved, constructing a basis preconditioner may require a different approach than in the IPM.

There are several questions about basis preconditioning in IPMs that deserve further investigation. During the development of IPX it was observed that different ρ-maximum volume bases can lead to very different iteration counts of the linear solver. This was attributed to sparsity in the tableau matrix. It seems possible to derive an algorithm for choosing the basis such that the Frobenius norm of the scaled tableau matrix is minimum among all neighbouring bases. This is likely to yield a sparse tableau, if it exists, and a more favourable spectrum of the preconditioned matrix. The linear algebra operations would be more expensive than for finding a maximum volume basis, but can still be economical, in particular for hypersparse problems.

Another question is extending the method to convex quadratic programming (QP) problems. For a QP model the KKT matrix can have a non-diagonal $(1,1)$-block, so that it cannot be scaled to least squares form. Because direct factorization of the KKT matrix can be much more expensive for QP than for LP models, developing a robust IPM with an iterative linear algebra kernel would be an improvement for many practitioners.

A Test Set and Solution Times

The LP models have been obtained from the following sources:

(1) J. Castro: `http://www-eio.upc.es/~jcastro/`;
(a) huge CTA instances, (b) integrated refinery problems, (c) `L1.zip`, (d) `Linf.zip`

(2) J. A. J. Hall: `http://www.maths.ed.ac.uk/hall/PublicLP/`

(3) Kennington collection: `http://www.netlib.org/lp/data/kennington/index.html`

(4) C. Mészáros: `http://old.sztaki.hu/~meszaros/public_ftp/lptestset/`;
(a) `misc`, (b) `New`, (c) `problematic`, (d) `stochlp`

(5) MIPLIB 2010: `http://miplib.zib.de/download/miplib2010-complete.tgz`

(6) H. Mittelmann: `http://plato.asu.edu/ftp/lptestset/`;
(a) `fome`, (b) `misc`, (c) `nug`, (d) `pds`, (e) `rail`

(7) Netlib collection: `http://www.netlib.org/lp/data/index.html`

The following table lists the computation times in seconds, excluding presolve, obtained on an Intel i5-6500 CPU (4 cores, 3.2GHz, 6MB L3 cache) and 8GB of physical memory. "simplex" means the Gurobi dual simplex solver. $\bar{m}$, $\bar{n}$ and nnz are the row and column dimension and the number of nonzero entries of the constraint matrix (without slack variables to inequality constraints) after the Gurobi presolve.

src	name	$\bar{m}$	$\bar{n}$	nnz	IPX				Gurobi barrier			simplex
					total	IPM	push	cleanup	total	IPM	crossover	
1a	L1_sixm250obs	154168	308284	639901	180.1	176.5	3.6		43.5	42.1	1.4	179.8
1a	L1_sixm500obs	283055	565680	1196860	553.8	506.0	47.8		471.8	468.7	3.1	803.0
1b	srd120	186440	357070	9804510	329.5	327.4	2.1		496.5	484.9	11.6	1048.3
1b	srd240	372920	714190	36457110	1883.3	1877.2	6.1		3615.5	3577.2	38.3	8483.8
1c	L1_bts4	25611	69900	181413	5.7	5.6	0.1		1.4	1.3	0.2	0.8
1c	L1_five20b	28342	62986	187981	47.4	47.2	0.1		6.5	3.7	2.8	251.9
1d	Linf_bts4	61465	70128	287954	11.1	10.8r	0.3	0.1	49.3	48.7	0.6	22.6
1d	Linf_five20b	60911	63458	285418	243.5	64.6r	16.3	162.6	25.1	10.8	14.3	1239.2
2	dcp2	13927	25376	281550	5.7	5.7r	0.0		0.7	0.6	0.1	1.5
3	cre-b	5213	31720	107039	2.3	2.3	0.0		0.4	0.4	0.1	0.6
3	ken-18	39855	89346	204936	5.5	5.5	0.1		1.3	1.1	0.1	1.1
3	osa-60	10209	224125	584253	5.7	5.6	0.1		1.1	1.0	0.1	0.5
4a	bas1lp	5409	4443	582390	4.0	3.8	0.2		1.2	1.0	0.1	0.2
4a	baxter	18867	10742	78412	2.5	2.4	0.1		1.9	1.9	0.1	0.3
4a	co9	6784	10675	81809	2.1	2.1	0.0		0.4	0.3	0.1	2.3
4a	dbic1	33598	140205	781668	58.4	46.7	11.7		12.0	2.8	9.2	23.3
4a	dbir1	7150	24862	994828	17.3	16.3	1.0		1.6	1.4	0.2	0.1
4a	e18	24598	14211	131991	12.6	12.3	0.2		63.2	63.1	0.2	1.0
4a	ex3sta1	12602	12675	54377	3.5	3.4	0.0		0.4	0.2	0.2	10.0
4a	jendrec1	2109	3535	88915	0.9	0.9	0.0		1.1	0.7r	0.4	0.4
4a	lpl1	32427	82733	252587	21.9	21.4	0.5		4.1	1.8	2.3	5.1
4a	mod2	23706	24005	114068	6.1	6.1	0.0		1.5	1.2	0.2	8.4
4a	model10	2961	10373	94372	1.6	1.5	0.0	0.0	0.3	0.2	0.1	4.2
4a	nemsemm1	2829	36052	495916	5.8	5.8	0.0		0.6	0.6	0.1	0.2
4a	nl	5290	7997	36524	1.0	1.0	0.0		0.4	0.4	0.0	0.5
4a	nsct1	7696	11297	589578	7.7	7.5	0.2		3.5	3.4	0.1	0.1
4a	p010	8907	17770	60150	4.3	4.2	0.0		0.1	0.1	0.0	0.1
4a	rat7a	2152	6772	156249	2.0	2.0	0.0		0.5	0.3	0.2	2.6
4a	route	20779	23923	184576	2.4	2.3	0.1		0.4	0.4	0.1	0.1
4a	stat96v1	4624	187892	562110	21.7	21.2r	0.4	0.1	20.8	4.9r	15.8	22.5
4a	stat96v3	26274	1090091	3250147	1423.9	470.1r	14.0	939.9	8063.5	5.2r	8058.3	7116.0
4a	ulevimin	4594	41225	135479	3.9	3.9	0.0		0.6	0.5	0.1	4.7
4a	world	23640	25839	114431	6.7	6.7	0.0		1.7	1.4	0.3	10.3
4b	degme	185501	659415	8127528	2496.8	2495.9	0.9		166.4	79.8	86.6	t
4b	karted	46501	133114	1770336	499.6	499.4	0.2		144.9	27.2	117.7	t
4b	tp-6	142752	1014301	11537419	2820.9	2820.0	0.8		66.0	42.0	24.0	t
4b	ts-palko	22002	47235	1076903	314.7	314.5	0.2		79.2	8.6	70.6	14865.8
4c	gen4	1475	4173	104236	12.3	11.8r	0.5		1.8	0.5	1.3	0.3
4c	l30	2698	15360	51093	1.8	1.6	0.1	0.1	4.0	0.8	3.2	2.8
4d	fxm3_16	32946	57391	315606	7.2	7.2	0.1		0.9	0.8	0.2	1.2
4d	pltexpa4_6	13693	24221	70946	3.0	3.0	0.1		1.3	1.2	0.1	0.2

src	name	$\bar{m}$	$\bar{n}$	nnz	IPX				Gurobi barrier			simplex
					total	IPM	push	cleanup	total	IPM	crossover	
4d	scfxm1-2r-256	25922	45663	158006	8.1	8.1	0.1		1.6	1.0	0.5	2.5
4d	stormg2-125	47536	129869	358498	28.3	28.1	0.2		2.9	2.5	0.4	1.2
4d	stormg2_1000	380036	1038119	2865373		f			29.3	24.1	5.2	53.1
5	30_70_45_095_100	12515	10967	46614	2.4	1.0	1.5		1.2	0.3	0.9	1.4
5	app1-2	52261	26265	194705	4.5	4.4	0.1		1.0	0.9	0.2	1.3
5	atlanta-ip	19835	17484	182879	10.2	10.0	0.2		4.8	4.5	0.3	6.5
5	bab3	22478	393457	3097799	120.1	119.8	0.3		6.2	5.9	0.3	125.0
5	bley_xl1	175178	5724	867393	10.4	7.8	2.6		13.5	13.2	0.3	14.1
5	buildingenergy	225031	128695	683844	262.5	260.5	2.0		6.1	5.7	0.5	7.9
5	circ10-3	42620	2700	307320	1.2	1.0	0.3		0.6	0.4	0.2	0.8
5	core4872-1529	4607	24098	184762	4.2	4.1	0.1		1.6	1.3	0.3	27.0
5	dano3mip	3150	13837	79530	1.7	1.7	0.0		1.1	0.9	0.1	4.8
5	datt256	9863	196147	1124622	1218.3	158.1	1060.2		718.3	1.0	717.3	308.7
5	dc1l	1650	35496	424338	5.9	5.8	0.1		1.1	1.0	0.1	5.8
5	dolom1	1802	10825	176976	2.4	2.4	0.0		0.6	0.5	0.0	2.6
5	ds-big	1042	173029	4573582	196.9	195.4	1.6		6.0	5.0	1.0	195.7
5	ex10	62932	15896	1031940	38.7	12.9	25.8		121.8	114.4	7.4	141.4
5	f2000	10495	3995	29490	3.6	2.3	1.3		2.8	0.9	1.9	46.5
5	germanrr	5524	10650	159548	2.0	1.9	0.0	0.0	0.3	0.3	0.0	0.2
5	gmut-75-50	2499	35915	569806	6.0	5.9	0.0	0.0	0.8	0.7	0.1	0.8
5	in	1495549	1439571	6696217	5381.1	4622.9	758.2		101.2	82.7	18.4	t
5	ivu06-big	1177	2197774	22556378	2067.8	2064.9	2.9		34.3	30.7	3.6	179.5
5	ivu52	2116	157543	2178871	69.2	68.4	0.8		3.5	3.2	0.3	30.9
5	map06	37504	18973	81653	11.3	11.2	0.1		3.5	3.2	0.3	2.2
5	mining	661094	348958	2754430	772.0	770.0	2.0		46.6	36.8	9.8	2253.4
5	momentum3	56421	13334	566199	25.7	25.6	0.2		22.1	18.0	4.0	18.2
5	msc98-ip	15293	12823	81666	4.1	3.6	0.3	0.2	3.0	2.6	0.4	1.1
5	mspp16	524814	4081	27555511	65.3	65.0	0.3		57.4	52.7	4.7	3.6
5	mzzv11	9316	9902	133424	2.3	2.1	0.2		1.6	1.5r	0.1	1.1
5	n15-3	28860	153140	575246	19.0	18.6	0.3		2.4	2.1	0.3	9.6
5	n3seq24	5950	119856	2404844	27.0	26.5	0.5		4.1	3.7	0.3	1.9
5	nb10tb	94742	49744	802185	92.8	78.2	0.6	14.0	53.6	41.9	11.7	115.7
5	neos-1140050	3795	39075	806835	6.8	6.8	0.0		3.9	3.3	0.6	16.3
5	neos-1429212	29416	64404	1222493	34.1	31.4	2.6		3.3	1.9	1.4	3.1
5	neos-1605075	3464	4085	91314	1.3	0.9	0.3		1.0	0.9	0.1	1.3
5	neos-476283	4312	6528	3924503	23.8	23.3	0.5		20.0	19.8	0.3	0.6
5	neos-506428	129925	42981	343466	7.5	4.4	3.1		1.3	1.0	0.3	0.1
5	neos-520729	17391	64464	175131	12.1	11.8	0.3		0.4	0.3	0.1	4.4
5	neos-631710	169576	167056	834166	220.2	3.4	216.8		8.7	1.2	7.5	1350.3
5	neos-738098	25736	8981	100842	2.6	1.5	1.1		0.6	0.5	0.1	1.7
5	neos-799711	12660	11615	40152	1.4	1.3	0.1		0.5	0.4	0.1	0.1

src	name	$\bar{m}$	$\bar{n}$	nnz	IPX				Gurobi barrier			simplex
					total	IPM	push	cleanup	total	IPM	crossover	
5	neos-824661	18804	29920	122230	2.0	1.7	0.3		0.3	0.2	0.1	0.2
5	neos-826694	6730	15210	52020	1.2	0.8	0.3		0.2	0.1	0.1	0.2
5	neos-933638	9642	8887	54484	1.9	1.1	0.9		0.6	0.4	0.3	5.1
5	neos-941313	13099	116670	386400	8.4	6.9	1.6		1.1	0.4	0.7	23.2
5	neos-948126	7208	7777	37891	1.6	1.1	0.5		0.4	0.2	0.2	12.6
5	neos-957389	5114	6034	355369	4.4	4.3	0.1		0.6	0.5	0.1	0.1
5	neos-984165	6921	7315	36573	1.7	1.2	0.5		0.4	0.2	0.2	10.2
5	neos6	1036	8563	251723	1.1	1.1	0.0		0.5	0.4	0.0	0.1
5	neos808444	17729	19330	119357	2.8	1.2	1.6		0.6	0.4	0.3	0.2
5	net12	13975	14115	80108	2.8	2.8	0.0		4.4	4.4	0.1	0.3
5	netdiversion	99581	128968	495878	110.3	105.2	5.1		14.4	4.2	10.3	3.5
5	npmv07	60835	162180	378798	23.1	22.9	0.2		3.7	3.5	0.2	0.6
5	ns1111636	13453	76462	283284	5.3	5.1	0.2		0.5	0.4	0.1	1.5
5	ns1116954	131865	11928	409850	7.0	2.1	4.9		12.2	11.7	0.5	5.7
5	ns1631475	24074	22485	100023	4.6	4.4	0.3		1.0	0.6	0.5	16.4
5	ns1644855	30597	30199	2100495	49.5	48.0	1.6		131.6	131.2	0.5	55.4
5	ns1663818	167047	123027	20177986	1105.0	1077.8r	27.3		208.3	197.3r	10.9	1.3
5	ns1685374	43195	9174	206101	9.9	9.9	0.0		11.9	1.6	10.4	161.8
5	ns1696083	10513	7748	371757	4.9	4.9	0.1		1.7	1.6	0.1	0.0
5	ns1758913	615190	17824	1265492	26.2	19.6	6.6		21.0	3.9	17.1	5.5
5	ns1853823	223144	213176	1346924	492.6	473.3	19.3		32.7	20.8r	11.9	2333.3
5	ns1854840	143236	135754	844834	136.5	124.1	12.4		6.2	4.7	1.6	1935.7
5	ns1904248	146398	38262	371070	6.6	1.7	4.9		0.8	0.7	0.2	25.6
5	ns1905797	51876	18188	239156	1.6	1.4	0.1		11.4	11.3	0.1	0.3
5	ns2017839	49884	53364	298482	20.1	20.1	0.1	0.0	7.3	3.2r	4.1	3.8
5	ns2118727	160408	164350	636614	42.0	41.9	0.1		13.8	13.5	0.3	41.4
5	ns2122603	19036	16556	64357		f			4.5	1.6	2.9	1.2
5	ns2124243	32277	48893	192513	4.8	4.4	0.4		0.6	0.3	0.3	4.5
5	ns2137859	99385	99462	593332	11.6	11.4	0.2		2.7	2.1	0.5	0.4
5	ns894244	9479	16399	68362	3.1	2.6	0.5		0.8	0.6	0.2	11.5
5	ns930473	22846	33136	139974	7.2	6.8	0.4		1.1	0.8	0.4	0.7
5	nsr8k	6283	38264	370206	16.8	16.4	0.4		4.7	3.9	0.7	133.1
5	ofi	105107	222217	848037	135.5	134.7	0.8	0.1	29.0	28.5	0.5	45.5
5	opm2-z11-s8	205936	7653	473063	19.4	10.9	8.6		22.4	15.0	7.4	29.2
5	opm2-z12-s7	297892	10336	678441	44.4	20.4	23.9		37.8	23.2	14.6	59.6
5	pb-simp-nonunif	123532	11798	298082	1.3	1.0	0.3		1.3	0.7	0.5	6.9
5	rail02	54524	192618	599436	71.0	62.3	8.6		45.2	42.0	3.2	685.1
5	rail03	129647	567095	1349518	399.2	318.7	80.5		55.4	48.6	6.8	1160.7
5	ramos3	2187	2187	32805	4.6	1.0	3.6		4.4	0.3	4.1	23.2
5	reblock420	62800	4200	138670	1.9	1.2	0.8		0.8	0.6	0.3	2.2
5	rmatr100-p5	8685	8784	26152	3.9	3.8	0.0		9.1	9.1	0.0	0.3

src	name	$\bar{m}$	$\bar{n}$	nnz	IPX				Gurobi barrier			simplex
					total	IPM	push	cleanup	total	IPM	crossover	
5	rmatr200-p5	37617	37816	113048	56.5	55.7	0.8		1.3	1.2	0.1	3.5
5	rmine14	268474	32144	659980	83.0	80.4	2.6		48.6	20.8	27.8	508.0
5	rmine21	1441506	162402	3514014	2707.8	2580.2	127.7		1620.0	694.8	925.2	t
5	rocII-9-11	42412	20557	503343	6.6	6.3	0.3		1.1	0.7	0.3	0.1
5	satellites3-40-fs	28171	54578	199798	22.0	14.7	7.4		6.2	3.1	3.1	7.5
5	satellites3-40	37422	54578	588630	34.4	25.4	9.0		61.2	56.6	4.6	2.0
5	sct1	9171	14514	85676	4.2	4.1	0.1		1.3	1.2	0.1	1.9
5	shs1023	126657	432068	1013029	173.5	170.8	2.7		14.4	12.8	1.6	128.0
5	siena1	2211	13482	254310	3.9	3.8	0.1		1.1	1.0	0.1	8.0
5	sing161	150806	464036	1459828	281.9	273.6	8.3		13.6	12.1	1.5	296.0
5	sing359	153253	428748	1408257	268.7	258.4	10.3		11.9	10.1	1.8	235.4
5	sp97ar	1692	14100	281317	1.4	1.4	0.0		0.2	0.2	0.0	0.2
5	splan1	521819	1253087	5088694	5242.2	4989.8	250.4	1.9	1775.7	1587.6	188.1	t
5	stockholm	36517	16622	111003	5.6	5.5	0.1		0.7	0.6	0.1	0.8
5	stp3d	97936	137646	500753	196.5	163.4	33.1		11.2	9.4	1.8	138.2
5	tanglegram1	68210	34501	204158	7.3	5.8	1.5		7.7	0.3	7.4	0.1
5	triptim3	14121	22982	494503	14.2	13.3	0.9		5.3	4.8	0.5	280.6
5	uc-case3	34288	27194	230068	7.2	7.0	0.2		1.1	0.9	0.2	0.8
5	unitcal_7	43301	25767	109859	12.5	12.1	0.4		1.1	0.9	0.2	0.3
5	van	22016	7360	481408	2.7	2.7	0.0		7.5	7.3	0.1	1.9
5	vpphard	39140	43395	332904	8.2	7.2	1.0		3.7	3.4	0.3	1.0
5	vpphard2	136399	139234	561958	26.2	25.8	0.4		36.6	36.3	0.3	0.8
5	wnq-n100-mw99-14	656900	10000	1333400	20.3	20.0	0.2		39.4	39.0	0.3	1.7
6a	fome13	34600	76249	246895	40.5	36.0	4.5		5.8	4.6	1.2	78.3
6a	fome21	24173	162336	356706	15.5	15.0	0.5		3.5	3.1	0.4	2.6
6b	cont1	120395	40398	359593	88.5	88.4	0.1		18.8	2.0	16.8	85.5
6b	cont11	120395	80396	359593	668.6	641.7	4.8	22.1	250.0	1.9	248.1	1120.3
6b	cont11_l	1468599	981396	4403001		m			3256.3	32.1	3224.2	t
6b	cont1_l	1918399	641598	5752001		m			1951.3	50.3	1900.9	t
6b	neos	419478	41140	911651	118.8	118.6	0.2		8.1	7.8	0.3	10.7
6b	neos1	131581	1892	468009	7.9	7.8	0.1		1.4	1.3	0.1	3.9
6b	neos3	512209	6624	1542816	60.9	27.4	33.5		38.6	11.4	27.2	16.1
6b	ns1687037	36080	30955	1377655	145.8	145.8	0.1		24.5	15.4	9.1	778.1
6b	ns1688926	24576	16489	901120		f			28.9	22.5 r	6.4	7.4
6b	sgpf5y6	19499	39020	109247	3.4	3.3	0.1		1.7	1.4	0.3	0.2
6b	watson_1	107522	216565	658933	85.7	85.2	0.5		4.5	3.3	1.3	7.2
6b	watson_2	185474	378986	1040238	193.8	192.7	1.1		6.5	3.5	3.0	12.4
6c	nug08-3rd	18270	20448	128547	573.8	253.8	320.1		91.2	18.8	72.4	330.9
6c	nug20	14098	72546	281906	192.3	176.1	16.1		435.4	37.5	398.0	3515.0
6c	nug30	52260	379350	1567800	11473.2	5125.2	6337.0	11.1	Inf	1400.5	Inf	t
6d	pds-100	94994	433867	933313	110.6	104.9	5.7		30.9	28.1	2.7	15.4

src	name	$\bar{m}$	$\bar{n}$	nnz	IPX				Gurobi barrier			simplex
					total	IPM	push	cleanup	total	IPM	crossover	
6d	pds-60	54289	285112	617497	39.5	37.7	1.8		16.5	15.4	1.1	10.1
6e	rail2586	2463	909940	7901059	259.0	258.6	0.4		15.3	12.7	2.7	19.9
6e	rail4284	4176	1090526	11174639	370.7	369.9	0.8		32.9	29.7	3.2	34.2
7	dfl001	4325	9511	30833	2.2	2.0	0.1		0.6	0.5	0.1	5.7
7	pilot87	1815	4420	70035	1.6	1.6	0.0		0.4	0.3	0.2	3.7
7	qap15	5698	22218	85413	15.4	13.3	2.1		4.3	2.8	1.5	361.1

f: failed, t: time limit, r: IPM solution reported not optimal

References

[1] T. Achterberg. Exploiting degeneracy in MIP. `http://www.iasi.cnr.it/aussois/web/uploads/2018/slides/achterbergt.pdf`. Accessed: Sep 13, 2018.

[2] G. Al-Jeiroudi. *On inexact Newton directions in interior point methods for linear optimization*. PhD thesis, University of Edinburgh, 2009.

[3] G. Al-Jeiroudi and J. Gondzio. Convergence analysis of the inexact infeasible interior-point method for linear optimization. *J. Optim. Theory Appl.*, 141(2):231–247, 2009.

[4] G. Al-Jeiroudi, J. Gondzio, and J. A. J. Hall. Preconditioning indefinite systems in interior point methods for large scale linear optimisation. *Optim. Methods Softw.*, 23(3):345–363, 2008.

[5] A. Alabi and J. Castro. Dantzig-Wolfe and block coordinate-descent decomposition in large-scale integrated refinery-planning. *Comput. Oper. Res.*, 36(8):2472–2483, 2009.

[6] E. D. Andersen, J. Gondzio, C. Mészáros, and X. Xu. Implementation of interior-point methods for large scale linear programs. In *Interior point methods of mathematical programming*, volume 5 of *Appl. Optim.*, pages 189–252. Kluwer Acad. Publ., Dordrecht, 1996.

[7] M. Arioli and I. S. Duff. Preconditioning linear least-squares problems by identifying a basis matrix. *SIAM J. Sci. Comput.*, 37(5):S544–S561, 2015.

[8] A. Ben-Israel. A volume associated with $m \times n$ matrices. *Linear Algebra Appl.*, 167:87–111, 1992.

[9] A. Ben-Israel and T. N. E. Greville. *Generalized inverses*, volume 15 of *CMS Books in Mathematics/Ouvrages de Mathématiques de la SMC*. Springer-Verlag, New York, second edition, 2003.

[10] R. E. Bixby and M. J. Saltzman. Recovering an optimal LP basis from an interior point solution. *Oper. Res. Lett.*, 15(4):169–178, 1994.

[11] COIN-OR Utils. `https://projects.coin-or.org/CoinUtils`. Accessed: Feb 10, 2017.

[12] IBM ILOG CPLEX Optimization Studio. `https://www.ibm.com/us-en/marketplace/ibm-ilog-cplex`. Accessed: Jun 23, 2017.

[13] T. A. Davis. *Direct methods for sparse linear systems*, volume 2 of *Fundamentals of Algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006.

[14] T. A. Davis and W. W. Hager. Row modifications of a sparse Cholesky factorization. *SIAM J. Matrix Anal. Appl.*, 26(3):621–639, 2005.

[15] I. S. Duff. On algorithms for obtaining a maximum transversal. *ACM Trans. Math. Softw.*, 7(3):315–330, September 1981.

[16] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct methods for sparse matrices*. Numerical Mathematics and Scientific Computation. Oxford University Press, New York, second edition, 2017.

[17] R. Fletcher and S. P. J. Matthews. Stable modification of explicit *LU* factors for simplex updates. *Math. Programming*, 30(3):267–284, 1984.

[18] J. J. H. Forrest and J. A. Tomlin. Updated triangular factors of the basis to maintain sparsity in the product form simplex method. *Math. Programming*, 2:263–278, 1972.

[19] L. Foster. San Jose State University singular matrix database. `http://www.math.sjsu.edu/singular/matrices/`. Accessed: Jan 17, 2018.

[20] R. Fukasawa and L. Poirrier. Permutations in the factorization of simplex bases. `http://www.optimization-online.org/DB\_HTML/2016/12/5770.html`. submitted Dec 13, 2016; accessed May 8, 2017.

[21] J. R. Gilbert and T. Peierls. Sparse partial pivoting in time proportional to arithmetic operations. *SIAM J. Sci. Statist. Comput.*, 9(5):862–874, 1988.

[22] A. J. Goldman and A. W. Tucker. Theory of linear programming. In H. W. Kuhn and A. W. Tucker, editors, *Linear Inequalities and Related Systems*, volume 38 of *Annals of Mathematics Studies*, pages 53–98. Princeton University Press, Princeton, NJ, 1956.

[23] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[24] J. Gondzio. Multiple centrality corrections in a primal-dual method for linear programming. *Comput. Optim. Appl.*, 6(2):137–156, 1996.

[25] J. Gondzio. Convergence analysis of an inexact feasible interior point method for convex quadratic programming. *SIAM J. Optim.*, 23(3):1510–1527, 2013.

[26] Z. Gong, M. Aldeen, and L. Elsner. A note on a generalized Cramer's rule. *Linear Algebra Appl.*, 340:253–254, 2002.

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

[28] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. How to find a good submatrix. In *Matrix methods: theory, algorithms and applications*, pages 247–256. World Sci. Publ., Hackensack, NJ, 2010.

[29] S. A. Goreinov and E. E. Tyrtyshnikov. The maximal-volume concept in approximation by low-rank matrices. In *Structured matrices in mathematics, computer science, and engineering, I (Boulder, CO, 1999)*, volume 280 of *Contemp. Math.*, pages 47–51. Amer. Math. Soc., Providence, RI, 2001.

[30] I. S. Gradshteyn and I. M. Ryzhik. *Table of integrals, series, and products*. Elsevier/Academic Press, Amsterdam, seventh edition, 2007.

[31] M. Gu and S. C. Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM J. Sci. Comput.*, 17(4):848–869, 1996.

[32] Gurobi Optimization. `http://www.gurobi.com`. Accessed: Jan 24, 2017.

[33] J. A. J. Hall and K. I. M. McKinnon. Hyper-sparsity in the revised simplex method and how to exploit it. *Comput. Optim. Appl.*, 32(3):259–283, 2005.

[34] P. M. J. Harris. Pivot selection methods of the Devex LP code. *Math. Programming*, 5:1–28, 1973.

[35] N. J. Higham and S. D. Relton. Estimating the largest elements of a matrix. *SIAM J. Sci. Comput.*, 38(5):C584–C601, 2016.

[36] Y. P. Hong and C.-T. Pan. Rank-revealing QR factorizations and the singular value decomposition. *Math. Comp.*, 58(197):213–232, 1992.

[37] Q. Huangfu. *High performance simplex solver*. PhD thesis, University of Edinburgh, 2013.

[38] Q. Huangfu and J. A. J. Hall. Novel update techniques for the revised simplex method. *Comput. Optim. Appl.*, 60(3):587–608, 2015.

[39] C. H. Hung and T. L. Markham. The Moore-Penrose inverse of a partitioned matrix $M = \left(\begin{smallmatrix} A & D \\ B & C \end{smallmatrix}\right)$. *Linear Algebra and Appl.*, 11:73–86, 1975.

[40] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[41] P. A. Knight, D. Ruiz, and B. Uçar. A symmetry preserving algorithm for matrix scaling. *SIAM J. Matrix Anal. Appl.*, 35(3):931–955, 2014.

[42] D. E. Knuth. Semioptimal bases for linear dependencies. *Linear and Multilinear Algebra*, 17(1):1–4, 1985.

[43] A. Koberstein. *The Dual Simplex Method, Techniques for a fast and stable implementation*. PhD thesis, Paderborn University, 2005.

[44] M. Kojima, N. Megiddo, and S. Mizuno. A primal-dual infeasible-interior-point algorithm for linear programming. *Math. Programming*, 61(3, Ser. A):263–280, 1993.

[45] M. Kojima, S. Mizuno, and A. Yoshise. A polynomial-time algorithm for a class of linear complementarity problems. *Math. Programming*, 44(1, (Ser. A)):1–26, 1989.

[46] M. Kojima, S. Mizuno, and A. Yoshise. A primal-dual interior point algorithm for linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior-Point and Related Methods*, chapter 2, pages 29–47. Springer, New York, NY, 1989.

[47] M. Kojima, S. Mizuno, and A. Yoshise. An $O(\sqrt{n}\,L)$ iteration potential reduction algorithm for linear complementarity problems. *Math. Programming*, 50(3, (Ser. A)):331–342, 1991.

[48] LAPACK. `http://www.netlib.org/lapack/`. Accessed: Sep 2, 2018.

[49] M. Lubin, J. A. J. Hall, C. G. Petra, and M. Anitescu. Parallel distributed-memory simplex for large-scale stochastic LP problems. *Comput. Optim. Appl.*, 55(3):571–596, 2013.

[50] D. G. Luenberger. The conjugate residual method for constrained minimization problems. *SIAM J. Numer. Anal.*, 7:390–398, 1970.

[51] N. Megiddo. Pathways to the optimal set in linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior-Point and Related Methods*, chapter 8, pages 131–158. Springer, New York, NY, 1989.

[52] N. Megiddo. On finding primal- and dual-optimal bases. *ORSA J. Comput.*, 3(1):63–65, 1991.

[53] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM J. Optim.*, 2(4):575–601, 1992.

[54] S. Mizuno and F. Jarre. Global and polynomial-time convergence of an infeasible-interior-point algorithm using inexact computation. *Math. Program.*, 84(1, Ser. A):105–122, 1999.

[55] S. Mizuno, M. Kojima, and M. J. Todd. Infeasible-interior-point primal-dual potential-reduction algorithms for linear programming. *SIAM J. Optim.*, 5(1):52–67, 1995.

[56] R. D. C. Monteiro and I. Adler. Interior path following primal-dual algorithms. part I: Linear programming. *Math. Programming*, 44(1, (Ser. A)):27–41, 1989.

[57] R. D. C. Monteiro and J. W. O'Neal. Convergence analysis of a long-step primal-dual infeasible interior-point LP algorithm based on iterative linear solvers. Technical report, School of ISyE, Georgia Tech, USA, 2003.

[58] R. D. C. Monteiro, J. W. O'Neal, and T. Tsuchiya. Uniform boundedness of a preconditioned normal matrix used in interior-point methods. *SIAM J. Optim.*, 15(1):96–100, 2004.

[59] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006.

[60] C. E. Nugent, T. E. Vollman, and J. Ruml. An experimental comparison of techniques for the assignment of facilities to locations. *Oper. Res.*, 16:150–173, 1968.

[61] A. R. L. Oliveira. *A New Class of Preconditioners for Large-Scale Linear Systems from Interior Point Methods for Linear Programming*. PhD thesis, Rice University, 1997.

[62] A. R. L. Oliveira and D. C. Sorensen. A new class of preconditioners for large-scale linear systems from interior point methods for linear programming. *Linear Algebra Appl.*, 394:1–24, 2005.

[63] OpenBLAS. `https://www.openblas.net/`. Accessed: Sep 2, 2018.

[64] C.-T. Pan. On the existence and computation of rank-revealing *LU* factorizations. *Linear Algebra Appl.*, 316(1-3):199–222, 2000.

[65] C. H. Papadimitriou. The largest subdeterminant of a matrix. *Bull. Soc. Math. Grèce (N.S.)*, 25:95–105, 1984.

[66] G. Peters and J. H. Wilkinson. The least squares problem and pseudo-inverses. *The Computer Journal*, 13:309–316, 1970.

[67] M. G. C. Resende, K. G. Ramakrishnan, and Z. Drezner. Computing lower bounds for the quadratic assignment problem with an interior point algorithm for linear programming. *Oper. Res.*, 43(5):781–791, 1995.

[68] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.

[69] L. Schork and J. Gondzio. An inexact potential reduction method for linear programming. Technical Report ERGO-16-005, University of Edinburgh, 2016.

[70] L. Schork and J. Gondzio. Maintaining a basis matrix in the linear programming interior point method. Technical Report ERGO-17-009, University of Edinburgh, 2017.

[71] L. Schork and J. Gondzio. Permuting spiked matrices to triangular form and its application to the Forrest-Tomlin update. Technical Report ERGO-17-002, University of Edinburgh, 2017.

[72] L. Schork and J. Gondzio. Implementation of an interior point method with basis preconditioning. Technical Report ERGO-18-014, University of Edinburgh, 2018.

[73] L. Schork and J. Gondzio. Rank revealing Gaussian elimination by the maximum volume concept. Technical Report ERGO-18-002, School of Mathematics, University of Edinburgh, 2018.

[74] U. H. Suhl and L. M. Suhl. Computing sparse *LU* factorizations for large-scale linear programming bases. *ORSA Journal on Computing*, 2(4):325–335, 1990.

[75] K. Tanabe. Centered Newton method for mathematical programming. In *System modelling and optimization (Tokyo, 1987)*, volume 113 of *Lect. Notes Control Inf. Sci.*, pages 197–206. Springer, Berlin, 1988.

[76] R. A. Tapia, Y. Zhang, and Y. Ye. On the convergence of the iteration sequence in primal-dual interior-point methods. *Math. Programming*, 68(2, Ser. A):141–154, 1995.

[77] M. J. Todd. Detecting infeasibility in infeasible-interior-point methods for optimization. In *Foundations of computational mathematics: Minneapolis, 2002*, volume 312 of *London Math. Soc. Lecture Note Ser.*, pages 157–192. Cambridge Univ. Press, Cambridge, 2004.

[78] M. J. Todd and Y. Ye. A centered projective algorithm for linear programming. *Math. Oper. Res.*, 15(3):508–529, 1990.

[79] R. J. Vanderbei. *Linear programming: foundations and extensions*, volume 4 of *International Series in Operations Research & Management Science*. Kluwer Academic Publishers, Boston, MA, 1996.

[80] S. J. Wright. *Primal-dual interior-point methods*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[81] R. Wunderling. *Paralleler und Objektorientierter Simplex*. PhD thesis, Technical University of Berlin, 1996.