Incomplete LU preconditioner based on max-plus approximation of LU factorization

Hook, James and Tisseur, Francoise

2016

MIMS EPrint: **2016.47**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

INCOMPLETE LU PRECONDITIONER BASED ON MAX-PLUS APPROXIMATION OF LU FACTORIZATION *

JAMES HOOK† AND FRANÇOISE TISSEUR‡

Abstract. We present a new method for the a priori approximation of the orders of magnitude of the entries in the LU factors of a complex or real matrix A. This approximation can be used to quickly determine the positions of the largest entries in the LU factors of A and these positions can then be used as the sparsity pattern for an incomplete LU factorization preconditioner. Our method uses max-plus algebra and is based solely on the moduli of the entries of A. We also present techniques for predicting which permutation matrices will be chosen by Gaussian elimination with partial pivoting. We exploit the strong connection between the field of Puiseux series and the max-plus semiring to prove properties of the max-plus LU factors. Experiments with a set of test matrices from the University of Florida sparse matrix collection show that our max-plus LU preconditioners outperform traditional level of fill methods and have similar performance to those preconditioners computed with more expensive threshold-based methods.

Key words. max-plus algebra, LU factorization, Hungarian scaling, linear systems of equations, sparse matrices, incomplete LU factorization, preconditioning.

AMS subject classifications. 65F08, 65F30, 15A23, 15A80.

1. Introduction.

Max-plus algebra is the analog of linear algebra developed for the binary operations max and plus over the real numbers together with $-\infty$, the latter playing the role of additive identity. Max-plus algebraic techniques have already been used in numerical linear algebra to, for example, approximate the orders of magnitude of the roots of scalar polynomials [17], to approximate the moduli of the eigenvalues of matrix polynomials [1, 9, 13], and to approximate singular values [8]. These approximations have been used as starting points for iterative schemes or in the design of preprocessing steps to improve the numerical stability of standard algorithms [3, 6, 13]. Our aim is to show how max-plus algebra can be used to approximate the sizes of the entries in the LU factors of a complex or real matrix A and how these approximations can subsequently be used in the construction of an incomplete LU (ILU) factorization preconditioner for A.

In order to be able to apply max-plus techniques to the matrix $A \in \mathbb{C}^{n\times n}$ we must first transform it into a max-plus matrix. We do this using the valuation map

$$\mathcal{V}_c : \mathbb{C} \to \overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty\}, \quad \mathcal{V}_c(x) = \log|x|, \quad (\log 0 = -\infty). \tag{1.1}$$

The valuation map is applied to matrices componentwise so that $\mathcal{V}_c(A) \in \overline{\mathbb{R}}^{n\times n}$ is a max-plus matrix. Note that for $x, y \in \mathbb{C}$, $\mathcal{V}_c(xy) = \mathcal{V}_c(x) + \mathcal{V}_c(y)$, and when $|x| \gg |y|$ or $|x| \ll |y|$ then $\mathcal{V}_c(x+y) \approx \max\{\mathcal{V}_c(x), \mathcal{V}_c(y)\}$. This suggests using the operations max and plus, which we denote by $\oplus$ and $\otimes$, respectively, in place of the classical addition and multiplication once we have applied the map $\mathcal{V}_c$.

The fundamental basis for our approximation of the magnitude of the entries of the LU factors of $A \in \mathbb{C}^{n\times n}$ is

*Version of December 6, 2016. This work was supported by Engineering and Physical Sciences Research Council grant EP/I005293. The work of the second author was also supported by a Royal Society-Wolfson Research Merit Award.

†Bath Institute for Mathematical Innovation, University of Bath, Bath, BA2 7AY, UK (j.l.hook@bath.ac.uk).

‡School of Mathematics, The University of Manchester, Manchester, M13 9PL, UK. (francoise.tisseur@manchester.ac.uk).

(a) the fact that the entries in the lower triangle of L and the upper triangle of U can be expressed explicitly in terms of determinants of submatrices S of A, and
(b) the heuristic that, when the matrix S has large variation in the size of its entries, $\mathcal{V}_c\big(\det(S)\big) \approx \operatorname{perm}\big(\mathcal{V}_c(S)\big)$, where perm is the max-plus permanent.

We use (a) and (b) to define a lower triangular max-plus matrix $\mathcal{L}$ and an upper triangular max-plus matrix $\mathcal{U}$ such that

$$\mathcal{V}_c(L) \approx \mathcal{L}, \qquad \mathcal{V}_c(U) \approx \mathcal{U}, \tag{1.2}$$

and refer to $\mathcal{L}$ and $\mathcal{U}$ as the max-plus LU factors of $\mathcal{A} := \mathcal{V}_c(A) \in \overline{\mathbb{R}}^{n\times n}$. The approximation (1.2) is a heuristic which only aims to capture the order of magnitude of the entries of L and U. One way to think about the max-plus LU approximation of the LU factors of A is as an intermediate between the true LU factors of A and a symbolic or boolean factorization which, based purely on the pattern of nonzero entries in A, predicts the nonzero patterns of the LU factors. We show that the matrix-matrix product $\mathcal{L} \otimes \mathcal{U}$ is usually not a factorization of $\mathcal{A}$ but that it "balances" $\mathcal{A}$.

In order for the max-plus approximation to be useful in practice, it is essential that the cost of computing it is less than the cost of computing the LU factorization exactly. We show that the max-plus LU factors can be computed by solving maximally weighted tree problems. As a result we provide an algorithm for computing the LU approximation of $A \in \mathbb{C}^{n\times n}$ with worst case cost $O\big(n\tau + n^2 \log n\big)$, where τ is the number of nonzero entries in A. Note that this cost depends on the number of nonzero entries in A and not on the number of nonzero entries in the LU factors of A. Thus while the approximate LU factors will exhibit fill-in just as in the exact case, the cost of computing the approximation is not affected by fill-in and will therefore be less than computing the exact LU factors. If the matrix A is first reordered according to its optimal assignment, so that the product of the moduli of the entries on its diagonal is maximized, then our approximation of the LU factors can be computed in parallel by n separate computations, each of individual cost $O\big(\tau + n \log n\big)$. If we seek only the positions and values of the k largest entries in each row of U and column of L, or if we seek only the position and values of the entries that are greater in modulus than some threshold, then this cost can be reduced further.

An approximation of the size of the entries in the LU factors of a sparse matrix A can be used to help construct an ILU preconditioner for solving $Ax = b$, that is, a pair of sparse lower and upper triangular matrices L, U such that the preconditioned matrix $AU^{-1}L^{-1}$ is more amenable to iterative methods such as GMRES [16]. Two classes of ILU preconditioners are threshold ILU and ILU(k). In threshold ILU, Gaussian elimination is applied to A but any computed element with modulus less than some threshold value is set to zero. Threshold ILU can produce effective preconditioners, but it can be quite slow. This is because there is a lot of work spent computing values that turn out to be less than the threshold and also because the sparse data structures that store the matrix entries are constantly being updated to accommodate the larger entries that are to be saved. For ILU(k) preconditioners, a sparsity pattern for the incomplete LU factors is first computed from a symbolic factorization that determines the level of fill-in of each fill-in entry of A [16, Sec. 10.3]. A fill-in entry is dropped when its level of fill is above k and the corresponding entry in the sparsity pattern matrix is set zero. The ILU factors are then computed using a variant of Gaussian elimination restricted to the sparsity pattern such as that provided in [16, Alg. 10.3]. The ILU(k) preconditioners can be computed quickly (for small k) but they do not

reliably result in effective preconditioners as they do not consider numerical values. Our max-plus LU approximation enables us to take a hybrid approach that offers the best of both of these methods as it uses the max-plus LU factors to define the sparsity pattern of the ILU preconditioners. Provided the entries of $\mathcal{L}$ and $\mathcal{U}$ give good approximations of the size of the true LU entries, our approach results in an ILU pair very close to the one obtained through standard threshold ILU, but our ILU pair can be computed considerably faster than the threshold ILU pair using the techniques for computing ILU(k) factors.

The remainder of this paper is organized as follows. In Section 2 we introduce the max-plus permanent and discuss how it can be used to approximate the order of magnitude of the determinant of a complex matrix. This approximation forms the basis of our LU factor approximation. In Section 3 we define the max-plus LU factors of a max-plus matrix and argue that they can be used to approximate the orders of magnitude of the entries in the LU factors of a complex matrix. We also show how our max-plus LU factorization can be adapted to include pivoting and examine the special case of Hungarian scaled matrices. In Section 4 we examine the connection between max-plus LU factors and the LU decomposition of matrices of Puiseux series, and use this connection to prove several of the theoretical results that are stated earlier in the paper. In Section 5 we give a derivation of our different max-plus LU algorithms and describe our max-plus ILU preconditioner. In Section 6 we apply our max-plus LU approximation and ILU preconditioning technique to a set of test problems from real life scientific computing problems.

Throughout this paper, complex matrices will be denoted by capital letters with their entries denoted by the corresponding lower case letter in the usual way $A = (a_{ij}) \in \mathbb{C}^{n\times n}$. Matrices of complex Puiseux series will be denoted by capital letters with a tilde and their entries by the corresponding lower case letter also with a tilde $\widetilde{A} = (\tilde{a}_{ij}) \in \mathbb{C}\{\{z\}\}^{n\times n}$, where $\mathbb{C}\{\{z\}\}$ denotes the field of Puiseux series. Max-plus matrices will be denoted by calligraphic capital letters and their entries by the corresponding lower case calligraphic letter $\mathcal{A} = (\mathcal{a}_{ij}) \in \overline{\mathbb{R}}^{n\times n}$. Since the most important results of this paper are the heuristic max-plus approximations, we will present these results in the style of theorems with a justification following each heuristic in lieu of a proof.

2. Heuristic approximation of the determinant. If we replace the sum by a maximum and the product by a summation in the Leibniz formula for the determinant of $A \in \mathbb{C}^{n\times n}$,

$$\det(A) = \sum_{\pi\in\Pi(n)} \operatorname{sgn}(\pi) \prod_{i=1}^{n} a_{i,\pi(i)},$$

where $\Pi(n)$ is the set of all permutations on $\{1,\ldots,n\}$, and replace the complex scalars $a_{i,\pi(i)}$ by scalars $\mathcal{a}_{i,\pi(i)} \in \overline{\mathbb{R}}$, we obtain the formula for the max-plus permanent of $\mathcal{A} = (\mathcal{a}_{ij}) \in \overline{\mathbb{R}}^{n\times n}$,

$$\operatorname{perm}(\mathcal{A}) = \max_{\pi\in\Pi(n)} \sum_{i=1}^{n} \mathcal{a}_{i,\pi(i)} = \bigoplus_{\pi\in\Pi(n)} \bigotimes_{i=1}^{n} \mathcal{a}_{i,\pi(i)}. \tag{2.1}$$

The following heuristic is fundamental to our max-plus LU approximation.

HEURISTIC 2.1. *Let $A \in \mathbb{C}^{n\times n}$ be sparse with nonzero entries that vary widely in magnitude and let $\mathcal{V}_c$ be as in* (1.1)*. Then*

$$\mathcal{V}_c\big(\det(A)\big) \approx \operatorname{perm}\big(\mathcal{V}_c(A)\big). \tag{2.2}$$

Justification. The determinant of A is a sum of terms the vast majority of which are zero (due to sparsity) and the remainder of which vary widely in order of magnitude (due to the wide variation in entry magnitude). The order of magnitude of the sum of a small number of terms of widely varying magnitude can then be approximated by the order of magnitude of the greatest of those terms, which is precisely $\operatorname{perm}(\mathcal{V}_c(A))$. □

We show in Section 4 that the permanent can also be used to calculate the exact asymptotic growth rate of the determinant of a generic matrix of Puiseux series, which provides some additional support for Heuristic 2.1. In the meantime let us look at a few examples.

EXAMPLE 2.2. *We use the logarithm in base 10 for $\mathcal{V}_c$ and consider*

$$A = \begin{bmatrix} 10 & 0 & 1000 \\ 1 & 10 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \qquad \mathcal{A} = \mathcal{V}_c(A) = \begin{bmatrix} 1 & -\infty & 3 \\ 0 & 1 & -\infty \\ -\infty & 0 & 0 \end{bmatrix},$$

For this example, $\operatorname{perm}(\mathcal{A}) = 3$*, which provides an order of magnitude approximation of* $\det(A) = -900$ *since* $\log|\det(A)| \approx 2.95$.

Of course we can easily find counter examples where the approximation in (2.2) is very poor. However, we can think of these matrices as occupying a set of small measure, so that the order of magnitude of the determinant of a "typical" complex matrix will be well approximated.

EXAMPLE 2.3. *Consider $A \in \left[\begin{smallmatrix} \omega_1 & \omega_2 \\ \omega_3 & \omega_4 \end{smallmatrix}\right] \in \mathbb{C}^{2\times 2}$ with $|\omega_i| = 1$, $i = 1, \ldots, 4$ so that $\mathcal{V}_c(A) = \left[\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right]$, where $\mathcal{V}_c(x) := \log_{10}|x|$. Choosing $\omega_i = 1$, $i = 1, \ldots, 4$ yields a singular A and* $\log|\det(A)| = -\infty$*, which is not detected by the max-plus approximation since* $\operatorname{perm}(\mathcal{V}_c(A)) = 0$*. Likewise whenever* $\det(A)$ *is close to zero the max-plus approximation will not be accurate. However for most choices of ω the approximation will capture the order of magnitude of* $\det(A)$*. Indeed if each ω_i is an independent random variable uniformly distributed on the unit circle then* $|\det(A)|$ *has expected value* $\mathbb{E}\big(|\det(A)|\big) = 4/\pi \approx 1$*, and for small $\epsilon > 0$, the probability* $\mathbb{P}\big(|\det(A)| \le \epsilon\big) \approx \epsilon/\pi$*. Thus the choices of ω_i for which the max-plus approximation fails to capture the order of magnitude of* $\det(A)$ *represent a set of small measure.*

3. Max-plus LU factors. An LU decomposition of $A \in \mathbb{C}^{n\times n}$ is a factorization of A into two factors, a unit lower triangular matrix denoted by L and an upper triangular matrix denoted by U such that $A = LU$. The entries of the L and U factors can be given explicitly in terms of determinants of submatrices of A (see [5, p. 35] or [10, p.11]) by

$$l_{ik} = \det\big(A([1:k-1,i],1:k)\big)/\det\big(A(1:k,1:k)\big), \quad i \ge k, \tag{3.1}$$

$$u_{kj} = \det\big(A(1:k,[1:k-1,j])/\det\big(A(1:k-1,1:k-1)\big), \quad j \ge k, \tag{3.2}$$

and $l_{ik} = u_{kj} = 0$ for $i, j < k$, where $A(i\colon j, k\colon \ell)$ denotes the submatrix of A formed by the intersection of the rows i to j and columns k to ℓ. If both the numerator and denominator in (3.1)–(3.2) are zero then we use the convention $0/0 = 0$. If the

denominator is equal to zero but the numerator is not, then we say that A does not admit an LU decomposition. If all of the denominators in (3.1)–(3.2) are nonzero then $A = LU$ is the unique LU decomposition of A.

Based on these formulae we define the *max-plus LU factors* of $\mathcal{A} \in \overline{\mathbb{R}}^{n\times n}$ to be the unit lower triangular max-plus matrix $\mathcal{L}$ and the upper triangular max-plus matrix $\mathcal{U}$ with entries given by

$$\mathcal{l}_{ik} = \operatorname{perm}\big(\mathcal{A}([1:k-1,i],1:k)\big) - \operatorname{perm}\big(\mathcal{A}(1:k,1:k)\big),\ i>k,\ \mathcal{l}_{ii}=0, \qquad (3.3)$$

$$u_{kj} = \operatorname{perm}\big(\mathcal{A}(1:k,[1:k-1,j]) - \operatorname{perm}\big(\mathcal{A}(1:k-1,1:k-1)\big),\ j\geq k, \qquad (3.4)$$

and $\mathcal{l}_{ik} = u_{kj} = -\infty$ if $i,j<k$. If the two terms on the right hand side of (3.3) or (3.4) are $-\infty$ then we use the convention $-\infty-(-\infty)=-\infty$. If the second term is $-\infty$ but the first is not, then we say that $\mathcal{A}$ does not admit max-plus LU factors.

HEURISTIC 3.1. *Let $A \in \mathbb{C}^{n\times n}$ and suppose that $\mathcal{V}_c(A)$ admits max-plus LU factors $\mathcal{L},\mathcal{U} \in \overline{\mathbb{R}}^{n\times n}$. Then A admits an LU decomposition $A = LU$ with*

$$\mathcal{V}_c(L) \approx \mathcal{L}, \qquad \mathcal{V}_c(U) \approx \mathcal{U}.$$

Justification. From Heuristic 2.1, we expect that the determinant of a submatrix of A is zero if and only if the permanent of the corresponding submatrix of $\mathcal{V}_c(A)$ is minus infinity. Therefore if $\mathcal{V}_c(A)$ admits max-plus LU factors then A admits an LU factorization $A = LU$, where the LU factors are as in (3.1)–(3.2). Taking the valuation of these expressions, applying Heuristic 2.1 and comparing to (3.3)–(3.4) gives the required result. □

EXAMPLE 3.2. *The matrix A of Example 2.2 has LU factorization*

$$A = \begin{bmatrix} 10 & 0 & 1000 \\ 1 & 10 & 0 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0.1 & 1 & 0 \\ 0 & 0.1 & 1 \end{bmatrix} \begin{bmatrix} 10 & 0 & 1000 \\ 0 & 10 & -100 \\ 0 & 0 & 11 \end{bmatrix} = LU,$$

and max-plus LU factors

$$\mathcal{L} = \begin{bmatrix} 0 & -\infty & -\infty \\ -1 & 0 & -\infty \\ -\infty & -1 & 0 \end{bmatrix}, \qquad \mathcal{U} = \begin{bmatrix} 1 & -\infty & 3 \\ -\infty & 1 & 2 \\ -\infty & -\infty & 1 \end{bmatrix},$$

which provide good approximations of the orders of magnitude of the entries in L, U.

EXAMPLE 3.3. *The LU factorization of the matrix A of Example 2.3 with $|\omega_i| = 1$ is given by*

$$A = \begin{bmatrix} \omega_1 & \omega_2 \\ \omega_3 & \omega_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{\omega_3}{\omega_1} & 1 \end{bmatrix} \begin{bmatrix} \omega_1 & \omega_2 \\ 0 & \omega_4 - \frac{\omega_2\omega_3}{\omega_1} \end{bmatrix} = LU$$

and the max-plus LU factors of $\mathcal{V}_c(A)$ are given by

$$\mathcal{L} = \begin{bmatrix} 0 & -\infty \\ 0 & 0 \end{bmatrix}, \qquad \mathcal{U} = \begin{bmatrix} 0 & 0 \\ -\infty & 0 \end{bmatrix}.$$

Since $\mathcal{V}_c(A)$ is independent of the choice of the ω_i since $|\omega_i| = 1$ so are its max-plus LU factors. The $(2,2)$ entry of U is the only entry where the max-plus approximation

is not guaranteed to be perfectly accurate but for most choices of the ω_i, the max-plus approximation captures the order of magnitude of the entries of L and U. There is, however, a small set of parameter values of small measure for which the max-plus approximation is not accurate.

Our definition of the max-plus LU factors of a max-plus matrix was chosen so that that we could use it to approximate the orders of magnitude of the entries in the LU factors of a complex matrix. But what do the max-plus LU factors of a max-plus matrix $\mathcal{A} \in \overline{\mathbb{R}}^{n\times n}$ tell us about $\mathcal{A}$?

THEOREM 3.4. *Suppose that $\mathcal{A} = (a_{ij}) \in \overline{\mathbb{R}}^{n\times n}$ has max-plus LU factors $\mathcal{L}$,$\mathcal{U} \in \overline{\mathbb{R}}^{n\times n}$. Then for each $i, j = 1, \ldots, n$ either*

$$(\mathcal{L} \otimes \mathcal{U})_{ij} := \max_{1\le k\le n} (\ell_{ik} + u_{kj}) > a_{ij}, \tag{3.5}$$

where the maximum is attained by at least two different values of k or

$$(\mathcal{L} \otimes \mathcal{U})_{ij} = a_{ij}. \tag{3.6}$$

The proof of Theorem 3.4 is provided in Section 4. We say that the max-plus matrix product $\mathcal{L} \otimes \mathcal{U}$ *balances* $\mathcal{A}$.

3.1. Pivoting. After k steps of Gaussian elimination applied to $A \in \mathbb{C}^{n\times n}$, the matrix A is reduced to

$$M_k \cdots M_1 A = U^{(k)} = \begin{bmatrix} U_{11}^{(k)} & U_{12}^{(k)} \\ 0 & U_{22}^{(k)} \end{bmatrix}, \tag{3.7}$$

where the M_i are Gauss transforms and $U_{11}^{(k)} \in \mathbb{C}^{k\times k}$ is upper triangular. Like the LU factors themselves, the entries of $U^{(k)}$ can be expressed in terms of determinants of submatrices of A as the next lemma shows.

LEMMA 3.5. *Let $A \in \mathbb{C}^{n\times n}$ have an LU factorization and let $U^{(k)}$ be as in* (3.7). *Then*

$$u_{ij}^{(k)} = \begin{cases} \det\big(A([1\colon k, i], [1\colon k, j])\big) / \det\big(A(1\colon k, 1\colon k)\big), & i, j > k, \\ u_{ij} & \textit{otherwise,} \end{cases} \tag{3.8}$$

where $U = U^{(n-1)}$ is the upper triangular factor in the LU factorization of A.

Proof. Suppose that $i, j > k$. Let R_i and C_j be elementary matrices swapping rows $k+1$ and i, and columns $k+1$ and j, respectively. Define $A' := R_i A C_j$ and let $U'^{(k)}$ be the matrix obtained after performing k steps of Gaussian elimination on A'. Then $U'^{(k)} = R_i U^{(k)} C_j$ and in particular $u'^{(k)}_{k+1,k+1} = u_{ij}^{(k)}$. The Gauss transform M'_{k+1} at step $k+1$ has the form $I + m_{k+1} e_{k+1}^T$, where $e_i^T m_{k+1} = 0$ for $i = 1, \ldots, k+1$ so that the $(k+1, k+1)$ entries of $U'^{(k)}$ and $M'_{k+1} U'^{(k)} = U'^{(k+1)}$ are the same, that is, $u'^{(k)}_{k+1,k+1} = u'^{(k+1)}_{k+1,k+1}$. But $u'^{(k+1)}_{k+1,k+1} = u'_{k+1,k+1}$ and by (3.2),

$$\begin{aligned} u'_{k+1,k+1} &= \det\big(A'(1:k+1, [1:k+1])\big) / \det\big(A'(1:k, 1:k)\big) \\ &= \det\big(A([1\colon k, i], [1\colon k, j])\big) / \det\big(A(1\colon k, 1\colon k)\big). \end{aligned}$$

The next steps of Gaussian elimination leave the (i, j) entries of $U^{(k)}$ with $\min\{i, j\} \le k$ unchanged so that $u_{ij}^{(k)} = u_{ij}$ for $\min\{i, j\} \le k$. ∎

We say that $A \in \mathbb{C}^{n\times n}$ is *partial pivoting free* if

$$|u_{k+1,k+1}^{(k)}| = \max_{k+1\le i\le n} |u_{i,k+1}^{(k)}|, \quad k = 0,\dots,n-1, \tag{3.9}$$

where $U^{(0)} = A$. If the matrix A is partial pivoting free then it is possible to apply Gaussian elimination with partial pivoting to A without the need for any row interchanges.

Let $A = LU$ be the LU decomposition of A and suppose that we compute an approximate LU pair $\widehat{L},\widehat{U} \in \mathbb{C}^{n\times n}$ using Gaussian elimination. The *backward error* of these approximate LU factors is equal to the perturbation $\Delta A \in \mathbb{C}^{n\times n}$ such that $A + \Delta A = \widehat{L}\widehat{U}$ and is known to satisfy [7, Lem. 9.6]

$$\|\Delta A\|_\infty \le \frac{nu}{1-nu}\||L||U|\|_\infty \le \gamma_n\big(1 + 2(n^2-n)\rho_n(A)\big)\|A\|_\infty,$$

where u is the unit roundoff and $\rho_n(A)$ is the *growth factor* for A defined by

$$\rho_n(A) = \frac{\max\limits_{0\le k\le n-1}\Big(\max\limits_{k\le i,j\le n}|u_{ij}^{(k)}|\Big)}{\max\limits_{i,j}|a_{ij}|}. \tag{3.10}$$

Thus if $\|\Delta A\|_\infty$ is small relative to $\|A\|_\infty$, which certainly happens when $\rho_n(A)$ is small, then the factorization is stable, otherwise it it unstable [7, Sec. 9.3].

In analogy to (3.9) we say that the max-plus matrix $\mathcal{A} \in \overline{\mathbb{R}}^{n\times n}$ is *partial pivoting free* if

$$\mathcal{u}_{k+1,k+1}^{(k)} = \max_{k+1\le i\le n} \mathcal{u}_{i,k+1}^{(k)}, \quad k = 0,\dots,n-1, \tag{3.11}$$

where $\mathcal{u}_{ij}^{(0)} := \mathcal{a}_{ij}$ and

$$\mathcal{u}_{ij}^{(k)} := \begin{cases} \operatorname{perm}\big(\mathcal{A}([1:k,i],[1:k,j])\big) - \operatorname{perm}\big(\mathcal{A}(1:k,1:k)\big), & i,j > k, \\ \mathcal{u}_{ij} & \text{otherwise.} \end{cases} \tag{3.12}$$

Also, in analogy to (3.10) we define the *max-plus growth factor* of $\mathcal{A} \in \overline{\mathbb{R}}^{n\times n}$ by

$$\varrho_n(\mathcal{A}) = \max_{0\le k\le n-1}\Big(\max_{k\le i,j\le n}\mathcal{u}_{ij}^{(k)}\Big) - \max_{1\le i,j\le n}\mathcal{a}_{ij} \ge 0. \tag{3.13}$$

THEOREM 3.6. *If $\mathcal{A} \in \overline{\mathbb{R}}^{n\times n}$ is partial pivoting free then $\varrho_n(\mathcal{A}) = 0$.*

The proof of Theorem 3.6 is deferred to Section 4.

HEURISTIC 3.7. *For $A \in \mathbb{C}^{n\times n}$ we have $\mathcal{V}_c\big(\rho_n(A)\big) \approx \varrho_n\big(\mathcal{V}_c(A)\big)$.*

Justification. From Lemma 3.5 and Heuristic 2.1 we have $\mathcal{V}_c(u_{ij}^{(k)}) \approx \mathcal{u}_{ij}^{(k)}$. The result then follows from the comparison of (3.10) and (3.13). ∎

If $\mathcal{V}_c(A)$ is partial pivoting free then it follows from Theorem 3.6 that $\varrho_n\big(\mathcal{V}_c(A)\big) = 0$ so that, based on Heuristic 3.7, the growth factor $\rho_n(A)$ should to be of order one, implying a backward stable LU factorization. As before this is a heuristic and it is not difficult to construct counterexample matrices A for which $\mathcal{V}_c(A)$ is partial or full pivoting free but that cannot be factorized in a stable way without further pivoting.

Theorem 3.6 and Heuristic 3.7 suggest applying a permutation P to a given A such that $\mathcal{V}_c(PA)$ is partial pivoting free. We show in Section 5.2 how to update

our max-plus LU algorithm to include partial pivoting. Another option is to apply Hungarian scaling, which is a two-sided diagonal scaling applied to $A \in \mathbb{C}^{n\times n}$ along with a permutation P that maximizes the product of the moduli of the diagonal entries of the matrix

$$H = PD_1AD_2, \tag{3.14}$$

where $D_1, D_2 \in \mathbb{R}^{n\times n}$ are nonsingular and diagonal, and such that H's entries satisfy

$$|h_{ij}| \le 1, \quad |h_{ii}| = 1 \quad i,j = 1,\dots,n. \tag{3.15}$$

We refer to any complex matrix satisfying (3.15) as a *Hungarian matrix*. The max-plus matrix $\mathcal{H} = \mathcal{V}_c(H) \in \overline{\mathbb{R}}^{n\times n}$ is such that $\hbar_{ij} \le 0$, $\hbar_{ii} = 0$, $i,j = 1,\dots,n$ and is referred to as a *max-plus Hungarian* matrix.

THEOREM 3.8. *Max-plus Hungarian matrices always admit max-plus LU factors.*

Proof. Suppose $\mathcal{H} \in \mathbb{R}_{\max}^{n\times n}$ is Hungarian. From (3.3) and (3.4) we have that

$$\operatorname{perm}\big(\mathcal{H}(1:k,1:k)\big) \ne -\infty, \quad k = 1,\dots,n,$$

is a sufficient condition for $\mathcal{H}$ to admit max-plus LU factors. Since $\hbar_{ii} = 0$ for $i = 1,\dots,n$, we have $\operatorname{perm}\big(\mathcal{H}(1:k,1:k)\big) \ge 0$ for $k = 1,\dots,n$. □

THEOREM 3.9. *A max-plus Hungarian matrix is partial pivoting free.*

Proof. It follows from (3.11) and (3.12) that $\mathcal{H}$ is partial pivoting free if

$$\operatorname{perm}\big(\mathcal{H}(1:k+1,1:k+1)\big) \ge \operatorname{perm}\big(\mathcal{H}([1:k,i],[1:k+1])\big) \tag{3.16}$$

for all $i = k+1,\dots,n$ and for all $k = 0,\dots,n-1$. But since $\hbar_{ij} \le 0$ for all i,j, the permanent of any submatrix of $\mathcal{H}$ must be nonpositive. Hence the right hand side of the inequality in (3.16) must be less than or equal to zero. Also, since $\mathcal{H}$ has zero diagonal entries, the permanent of any principal leading submatrix of $\mathcal{H}$ is equal to zero. Therefore the inequality in (3.16) must have left hand side equal to zero so that $\mathcal{H}$ is partial pivoting free. □

Therefore given $A \in \mathbb{C}^{n\times n}$ we apply Hungarian scaling to obtain $H = PD_1AD_2$ and from Theorems 3.6 and 3.9, and Heuristic 3.7, we expect that it should be possible to factorize H in a stable way without any need for interchange. This preprocessing technique was originally suggested by Olschowka and Neumaier in [14]. They prove that Hungarian scaling removes the need for interchange in Gaussian elimination for some special classes of matrices. Whilst our results do not constitute a definite theorem they provide some intuitive explanation for the widely observed fact that Hungarian scaling significantly reduces the need for pivoting (see Section 6).

EXAMPLE 3.10. *Let* $A = \left[\begin{smallmatrix} 1 & 10^{-3} \\ 10 & 1 \end{smallmatrix}\right]$. *We have that* $\mathcal{V}_c(A) = \left[\begin{smallmatrix} 0 & -3 \\ 1 & 0 \end{smallmatrix}\right]$ *is not partial pivoting free since* $\mathcal{U}^{(0)} = \mathcal{V}_c(A)$ *is such that* $u_{21}^{(0)} = 1 > u_{11}^{(0)} = 0$. *Similarly* A *is not partial pivoting free. It is easy to check that the matrices* PA *and* $\mathcal{V}_c(PA)$ *with* $P = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ *are both partial pivoting free. Now a Hungarian scaling for* A *with* $P = I$, $D_1 = \operatorname{diag}(1, 10^{-2})$, *and* $D_2 = \operatorname{diag}(1, 10^2)$ *is given by* $H = PD_1AD_2 = \left[\begin{smallmatrix} 1 & 10^{-1} \\ 10^{-1} & 1 \end{smallmatrix}\right]$ *so that* $\mathcal{V}_c(H) = \left[\begin{smallmatrix} 0 & -1 \\ -1 & 0 \end{smallmatrix}\right]$. *Theorem* 3.9 *guarantees that* $\mathcal{V}_c(H)$ *is partial pivoting free and it is easy to check that* H *is also partial pivoting free.*

4. Puiseux series. There is a stronger connection between the field of complex Puiseux series and the semiring $\mathbb{R}_{\max} = (\overline{\mathbb{R}}, \oplus, \otimes)$ than between the field of complex numbers and $\mathbb{R}_{\max}$, which we now exploit to prove properties of the max-plus LU factors as well as Theorems 3.4 and 3.6, and to provide further justification of Heuristic 2.1. This section is not needed for the derivation on the max-plus LU algorithms presented in Section 5.

Complex Puiseux series

$$f(z) = \sum_{i=k}^{\infty} c_i z^{\frac{i}{m}}, \tag{4.1}$$

with $m \in \mathbb{N}$, $k \in \mathbb{Z}$, $c_i \in \mathbb{C}$, $i \geq k$, and $c_k \neq 0$ form an algebraically closed field under addition and multiplication denoted by $\mathbb{C}\{\{z\}\}$. On that field, we define the valuation

$$\mathcal{V}_p : \mathbb{C}\{\{z\}\} \mapsto \overline{\mathbb{R}}, \quad \mathcal{V}_p(f) = -k/m, \tag{4.2}$$

that is, the valuation of a Puiseux series is minus the degree of its lowest order term. This valuation provides a near homeomorphism between $\mathbb{C}\{\{z\}\}$ and $\mathbb{R}_{\max}$,

$$\begin{aligned} \mathcal{V}_p(fg) &= \mathcal{V}_p(f) \otimes \mathcal{V}_p(g), \quad \text{for all } f, g \in \mathbb{C}\{\{z\}\}, \\ \mathcal{V}_p(f+g) &\leq \mathcal{V}_p(f) \oplus \mathcal{V}_p(g), \quad \text{for all } f, g \in \mathbb{C}\{\{z\}\}, \\ \mathcal{V}_p(f+g) &= \mathcal{V}_p(f) \oplus \mathcal{V}_p(g), \quad \text{for almost all } f, g \in \mathbb{C}\{\{z\}\}, \end{aligned} \tag{4.3}$$

where the third relation holds except for when $\mathcal{V}_p(f) = \mathcal{V}_p(g)$ and the coefficient of the lowest order term of f is equal to minus that of g. As for complex matrices, the valuation $\mathcal{V}_p$ is applied componentwise to matrices with Puiseux series entries. We decorate matrices in $\mathbb{C}\{\{z\}\}^{n\times n}$ with a tilde to distinguish them from matrices in $\mathbb{C}^{n\times n}$.

Any entry of $\widetilde{A} \in \mathbb{C}\{\{z\}\}^{n\times n}$ can be written as

$$\tilde{a}_{ij} = c_{ij} z^{-\mathcal{V}_p(\tilde{a}_{ij})} + \text{higher order terms},$$

where $C = (c_{ij}) =: \mathcal{L}(\widetilde{A}) \in \mathbb{C}^{n\times n}$ is the matrix of lowest order term coefficients of $\widetilde{A}$ with $\mathcal{L} : \mathbb{C}\{\{z\}\}^{n\times n} \mapsto \mathbb{C}^{n\times n}$. For a set of permutations $\Phi \subset \Pi(n)$, we define the map $g_\Phi : \mathbb{C}^{n\times n} \mapsto \mathbb{C}$ by

$$g_\Phi(C) = \sum_{\pi\in\Phi} \operatorname{sign}(\pi) \prod_{i=1}^{n} c_{i\pi(i)}. \tag{4.4}$$

Note that $g_{\Pi(n)}(C) = \det(C)$. For $\mathcal{A} \in \overline{\mathbb{R}}^{n\times n}$ such that $\operatorname{perm}(\mathcal{A}) \neq -\infty$ we denote by

$$\operatorname{ap}(\mathcal{A}) = \Big\{ \pi \in \Pi(n) : \sum_{i=1}^{n} \boldsymbol{a}_{i\pi(i)} = \operatorname{perm}(\mathcal{A}) \Big\}$$

the set of optimal assignments for $\mathcal{A}$.

The next lemma identifies the set of matrices with Puiseux series entries such that the valuation of the determinant is exactly the permanent of the valuation (see Heuristic 2.1 for matrices with complex entries).

LEMMA 4.1. *Let* $\widetilde{A} \in \mathbb{C}\{\{z\}\}^{n\times n}$ *and suppose that* $g_{\mathrm{ap}(\mathcal{V}_p(\widetilde{A}))}\big(\mathcal{L}(\widetilde{A})\big) \neq 0$, *where* g, ap, *and* $\mathcal{L}$ *are defined above. Then* $\mathcal{V}_p\big(\det(\widetilde{A})\big) = \mathrm{perm}\big(\mathcal{V}_p(\widetilde{A})\big)$.

Proof. Let $\mathcal{A} = \mathcal{V}_p(\widetilde{A}) \in \overline{\mathbb{R}}^{n\times n}$. First suppose that $\mathrm{perm}(\mathcal{A}) = -\infty$. Then for each permutation $\pi \in \Pi(n)$ there exists i such that $a_{i\pi(i)} = -\infty$ so that $\tilde{a}_{i\pi(i)} = 0$. Thus $\det(\widetilde{A}) = \sum_{\pi\in\Pi(n)} \mathrm{sign}(\pi) \prod_{i=1}^n \tilde{a}_{i\pi(i)} = 0$ and $\mathcal{V}_p\big(\det(\widetilde{A})\big) = \mathrm{perm}(\mathcal{A})$.

Now suppose that $\mathrm{perm}(\mathcal{A}) \neq -\infty$ and let $C = \mathcal{L}(\widetilde{A})$. Then

$$\det(\widetilde{A}) = \sum_{\pi\in\Pi(n)} \mathrm{sign}(\pi) \prod_{i=1}^n \tilde{a}_{i\pi(i)} = \sum_{\pi\in\Pi(n)} \mathrm{sign}(\pi)\Big(z^{-\sum_{i=1}^n a_{i\pi(i)}} \prod_{i=1}^n c_{i\pi(i)} + \text{h.o.t.}\Big),$$

where h.o.t. stands for higher order terms. We break the sum into two parts, one over $\mathrm{ap}(\mathcal{A})$ and one over $\Pi(n) \setminus \mathrm{ap}(\mathcal{A})$. We have that

$$\begin{aligned}\sum_{\pi\in\mathrm{ap}(\mathcal{A})} \mathrm{sign}(\pi)\Big(z^{-\sum_{i=1}^n a_{i\pi(i)}} \prod_{i=1}^n c_{i\pi(i)} + \text{h.o.t.}\Big) &= z^{-\mathrm{perm}(\mathcal{A})} \sum_{\pi\in\mathrm{ap}(\mathcal{A})} \mathrm{sign}(\pi) \prod_{i=1}^n c_{i\pi(i)} + \\ &\qquad \text{h.o.t.} \\ &= z^{-\mathrm{perm}(\mathcal{A})} g_{\mathrm{ap}(\mathcal{A})}(C) + \text{h.o.t.},\end{aligned}$$

where $g_{\mathrm{ap}(\mathcal{A})}(C)$ is defined in (4.5). Since for $\pi \in \Pi(n)\setminus\mathrm{ap}(\mathcal{A})$, $\sum_{i=1}^n a_{i\pi(i)} < \mathrm{perm}(\mathcal{A})$, $z^{-\sum_{i=1}^n a_{i\pi(i)}}$ is higher order than $z^{-\mathrm{perm}(\mathcal{A})}$ and so is

$$\sum_{\pi\in\Pi(n)\setminus\mathrm{ap}(\mathcal{A})} \mathrm{sign}(\pi)\Big(z^{-\sum_{i=1}^n a_{i\pi(i)}} \prod_{i=1}^n c_{i\pi(i)} + \text{h.o.t.}\Big).$$

Hence, $\det(\widetilde{A}) = z^{-\mathrm{perm}(\mathcal{A})} g_{\mathrm{ap}(\mathcal{A})}(C) + \text{h.o.t.}$ and $\mathcal{V}_p\big(\det(\widetilde{A})\big) = \mathrm{perm}(\mathcal{A})$ since $g_{\mathrm{ap}(\mathcal{A})}(C) \neq 0$. ∎

The next lemma will be useful to show that $\mathcal{V}_p\big(\det(\widetilde{A})\big) = \mathrm{perm}\big(\mathcal{V}_p(\widetilde{A})\big)$ holds for generic matrices $\widetilde{A} \in \mathbb{C}\{\{z\}\}^{n\times n}$ but also to explain what we mean by generic in this context.

LEMMA 4.2. *Let* g_Φ *be as in* (4.4). *Then the set*

$$\mathcal{G}_n = \{C \in \mathbb{C}^{n\times n} : \ g_\Phi(C) \neq 0 \text{ for all nonempty } \Phi \subset \Pi(n)\} \tag{4.5}$$

is a generic (*open and dense*) *subset of* $\mathbb{C}^{n\times n}$.

Proof. For each $\Phi \subset \Pi(n)$, $g_\Phi(C)$ is a polynomial in the coefficients of C. A polynomial is either identically equal to zero or only zero on some low dimensional subset. Therefore

$$V(g_\phi) = \{C \in \mathbb{C}^{n\times n} \ : \ g_\Phi(C) = 0\},$$

is either the whole of $\mathbb{C}^{n\times n}$ or it is a lower dimensional subset of $\mathbb{C}^{n\times n}$. Choose some permutation $\pi \in \Phi$ and define $C_\pi \in \mathbb{C}^{n\times n}$ by $c_{ij} = 1$ if $j = \pi(i)$ and $c_{ij} = 0$ otherwise. By construction we have $g_\Phi(C_\pi) = \mathrm{sign}(\pi) = \pm 1 \neq 0$ and therefore $C_\pi \notin V(g_\phi)$. Therefore $V(g_\phi)$ cannot be the whole of $\mathbb{C}^{n\times n}$ and must instead be a lower dimensional subset. Thus $\mathbb{C}^{n\times n} \setminus V(g_\phi)$ is a generic subset of $\mathbb{C}^{n\times n}$. Finally note that

$$\mathcal{G}_n = \bigcap_{\phi\subset\Pi(n)} \big\{\mathbb{C}^{n\times n} \setminus V(g_\phi)\big\},$$

is a finite intersection of generic subsets and is therefore generic. ∎

Now if $\widetilde{A} \in \mathbb{C}\{\{z\}\}^{n\times n}$ is such that $\mathcal{L}(\widetilde{A}) \in \mathcal{G}_n$ then $g_{\mathrm{ap}(\mathcal{V}_p(\widetilde{A}))}\big(\mathcal{L}(\widetilde{A})\big) \neq 0$. Lemma 4.2 states that $\mathcal{G}_n$ is a generic set, so that the property $g_{\mathrm{ap}(\mathcal{V}_p(\widetilde{A}))}\big(\mathcal{L}(\widetilde{A})\big) \neq 0$ is a generic property for $\widetilde{A} \in \mathbb{C}\{\{z\}\}^{n\times n}$ with respect to the topology induced by the map $\mathcal{L} : \mathbb{C}\{\{z\}\}^{n\times n} \mapsto \mathbb{C}^{n\times n}$. A more intuitive way of understanding this result is that, if we have a matrix $\widetilde{A}$ where the leading order coefficients $\mathcal{L}(\widetilde{A})$ have been chosen at random, according to a continuous distribution, then with probability one $g_{\mathrm{ap}(\mathcal{V}_p(\widetilde{A}))}\big(\mathcal{L}(\widetilde{A})\big) \neq 0$ will hold. We then say that $\mathcal{V}_p\big(\det(\widetilde{A})\big) = \mathrm{perm}\big(\mathcal{V}_p(\widetilde{A})\big)$ holds for "almost all" $\widetilde{A} \in \mathbb{C}\{\{z\}\}^{n\times n}$.

EXAMPLE 4.3. *Consider*

$$\widetilde{A} = \begin{bmatrix} z^{-1} & 0 & z^{-3} \\ 1 & z^{-1} & 0 \\ 0 & 1 & 1 \end{bmatrix}, \quad \mathcal{V}_p(\widetilde{A}) = \begin{bmatrix} 1 & -\infty & 3 \\ 0 & 1 & -\infty \\ -\infty & 0 & 0 \end{bmatrix}, \quad \mathcal{L}(\widetilde{A}) = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

It is easy to check that $\mathcal{L}(\widetilde{A}) \in \mathcal{G}_3$, $\det(\widetilde{A}) = -z^{-3} + z^{-2}$, *and that* $\mathcal{V}_p\big(\det(\widetilde{A})\big) = \mathrm{perm}\big(\mathcal{V}_p(\widetilde{A})\big) = 3$ *as expected from Lemma* 4.1.

We now show how to use Puiseux series to further justify Heuristic 2.1.

Justification of Heuristic 2.1. For $f(z) = cz^{-\mathcal{V}_p(f)} + \text{h.o.t.} \in \mathbb{C}\{\{z\}\}$ with $c \neq 0$, we have that $\log_{|z|} |f(z)| \to -\mathcal{V}_p(f)$ as $|z| \to 0$. Therefore, for z_0 in the domain of the asymptotic regime of f, we have that $\log_{|z_0|} |f(z_0)| \approx -\mathcal{V}_p(f)$.

Now suppose that $x_0 \in \mathbb{C}$ is some value of interest and that we know $\mathcal{V}_p(f)$ but not f, where $f \in \mathbb{C}\{\{z\}\}$ is a Puiseux series with $f(\check{z}_0) = x_0$ for some $\check{z}_0 \in \mathbb{C}$. Then, assuming $\check{z}_0$ is small enough so that it is in the domain of the asymptotic regime of f we have

$$\mathcal{V}_c(x_0) = \log|x_0| = \log|f(\check{z}_0)| = \log|\check{z}_0| \log_{|\check{z}_0|} |f(\check{z}_0)| \approx -\log|\check{z}_0| \mathcal{V}_p(f). \tag{4.6}$$

This approximation falls short of being a theorem because we have no way of guaranteeing that $\check{z}_0$ is in the domain of the asymptotic regime of f. In other words there is no uniform scale for determining what constitutes a small value of $z \in \mathbb{C}$.

We can apply the same idea to approximate the determinant of $A \in \mathbb{C}^{n\times n}$. Suppose that we know $\mathcal{V}_p(\widetilde{A})$ with $\widetilde{A} \in \mathbb{C}\{\{z\}\}^{n\times n}$ such that $\widetilde{A}(\check{z}_0) = A$ for some $\check{z}_0 \in \mathbb{C}$ and $\mathcal{L}(\widetilde{A}) \in \mathcal{G}_n$. Then, assuming $\check{z}_0$ is in the domain of the asymptotic regime of $\widetilde{A}$, if follows from (4.6) that

$$\mathcal{V}_c(A) \approx -\log|\check{z}_0| \mathcal{V}_p(\widetilde{A}). \tag{4.7}$$

Since $\widetilde{A}(\check{z}_0) = A$ we have $\det(A) = f(\check{z}_0)$, where $f = \det(\widetilde{A}) \in \mathbb{C}\{\{z\}\}$. Assuming that $\check{z}_0$ is in the domain of the asymptotic regime of f and applying (4.6) we have

$$\mathcal{V}_c\big(\det(A)\big) \approx -\log|\check{z}_0| \mathcal{V}_p(\det(\widetilde{A})).$$

Using Lemma 4.1 and (4.7) we obtain that $\mathcal{V}_c\big(\det(A)\big) \approx \mathrm{perm}\big(\mathcal{V}_c(A)\big)$, which provides another justification for Heuristic 2.1. ∎

We will need the next lemma.

LEMMA 4.4. *Let* $C \in \mathbb{C}^{n\times n} \setminus \mathcal{G}_n$ *with* $\mathcal{G}_n$ *as in* (4.5). *Then any* $k \times k$ *submatrix of* C *is in* $\mathbb{C}^{k\times k} \setminus \mathcal{G}_k$.

Proof. If $P, Q \in \mathbb{C}^{n\times n}$ are permutation matrices then $C \in \mathbb{C}^{n\times n} \setminus \mathcal{G}_n$ if and only if $PCQ \in \mathbb{C}^{n\times n} \setminus \mathcal{G}_n$ so it suffices to prove the result for the principal submatrix C of order k, which we denote by S. For any $\Psi \subset \Pi(k)$, we construct $\Phi \subset \Pi(n)$ by setting $\Phi = \{\pi[\varphi] \in \Pi(n) : \varphi \in \Psi\}$, where

$$\pi[\varphi](i) = \begin{cases} \varphi(i), & 1 \le i \le k, \\ i, & k+1 \le i \le n. \end{cases}$$

Then

$$g_\Phi(C) = \sum_{\pi\in\Phi} \operatorname{sign}(\pi) \prod_{i=1}^{n} c_{i\pi(i)} = \sum_{\varpi\in\Psi} \operatorname{sign}(\varpi) \prod_{i=1}^{k} s_{i\varpi(i)} \prod_{i=k+1}^{n} c_{ii} = g_\Psi(S) \prod_{i=k+1}^{n} c_{ii}$$

so that $g_\Psi(S) = 0$ if and only if $g_\Phi(C) = 0$. Hence $C \in \mathbb{C}^{n\times n} \setminus \mathcal{G}_n$ if and only if $S \in \mathbb{C}^{k\times k} \setminus \mathcal{G}_k$. ∎

As for complex matrices, an LU factorization of $\widetilde{A} \in \mathbb{C}\{\{z\}\}^{n\times n}$ is a factorization of $\widetilde{A}$ into a lower triangular matrix $\widetilde{L} \in \mathbb{C}\{\{z\}\}^{n\times n}$ with ones on the diagonal and an upper triangular matrix $\widetilde{U} \in \mathbb{C}\{\{z\}\}^{n\times n}$ such that $\widetilde{A} = \widetilde{L}\widetilde{U}$. When the factorization exists, the nonzero entries of $\widetilde{L}$ and $\widetilde{U}$ can be defined as in (3.1)–(3.2) with $\widetilde{A}$ in place of A. The next result should be compared to Heuristic 3.1.

THEOREM 4.5. *Let $\widetilde{A} \in \mathbb{C}\{\{z\}\}^{n\times n}$ be such that $\mathcal{L}(\widetilde{A}) \in \mathcal{G}_n$ and suppose that $\mathcal{V}_p(\widetilde{A}) \in \overline{\mathbb{R}}^{n\times n}$ admits max-plus LU factors $\mathcal{L}$,$\mathcal{U} \in \overline{\mathbb{R}}^{n\times n}$. Then $\widetilde{A}$ admits an LU factorization $\widetilde{A} = \widetilde{L}\widetilde{U}$, where*

$$\mathcal{V}_p(\widetilde{L}) = \mathcal{L}, \quad \mathcal{V}_p(\widetilde{U}) = \mathcal{U}.$$

If for $\widetilde{A} \in \mathbb{C}\{\{z\}\}^{n\times n}$, $\mathcal{V}_p(\widetilde{A})$ does not admit max-plus LU factors then $\widetilde{A}$ does not admit an LU factorization.

Proof. Let $\mathcal{A} = \mathcal{V}_p(\widetilde{A})$. From Lemmas 4.1 and 4.4 we have

$$\mathcal{V}_p\big(\det\big(\widetilde{A}([i_1,\dots,i_k],[j_1,\dots,j_k])\big)\big) = \operatorname{perm}\big(\mathcal{A}([i_1,\dots,i_k],[j_1,\dots,j_k])\big),$$

for all submatrices of $\widetilde{A}$ since $\mathcal{L}(\widetilde{A}) \in \mathcal{G}_n$. Therefore a submatrix of $\widetilde{A}$ has zero determinant if and only if the corresponding submatrix of $\mathcal{V}_p(\widetilde{A})$ has permanent equal to minus infinity. Thus if $\mathcal{A}$ admits max-plus LU factors then an LU factorization of $\widetilde{A}$ exists with entries given by (3.1)–(3.2).

If $\mathcal{V}_p(\widetilde{A})$ does not have max-plus LU factors then this means that for some i, j, k, the first term on the right hand side of (3.3) or (3.4) is equal to $-\infty$ but the second term is not. As a result the denominator on the right hand side of (3.1) or (3.2) is equal to 0 but numerator is not so $\widetilde{A}$ does not have an LU factorization. ∎

Recall from Section 3 that for $\mathcal{A}, \mathcal{B}, \mathcal{C} \in \overline{\mathbb{R}}^{n\times n}$, the product $\mathcal{A}\otimes\mathcal{B}$ balances $\mathcal{C}$ if for every $i, j = 1, \dots, n$ either $(\mathcal{A}\otimes\mathcal{B})_{ij} = c_{ij}$ or $(\mathcal{A}\otimes\mathcal{B})_{ij} = \max_{1\le k\le n}(a_{ik} + b_{kj}) > c_{ij}$, where the maximum must be attained by at least two different values of k.

LEMMA 4.6. *Let $\widetilde{A}, \widetilde{B} \in \mathbb{C}\{\{z\}\}^{n\times n}$. Then the product $\mathcal{V}_p(\widetilde{A}) \otimes \mathcal{V}_p(\widetilde{B})$ balances $\mathcal{V}_p(\widetilde{A}\widetilde{B})$.*

Proof. We have $(\widetilde{A}\widetilde{B})_{ij} = cz^{-\max_{1\le k\le n}(\mathcal{V}_p(\tilde{a}_{ik})+\mathcal{V}_p(\tilde{b}_{kj}))} + \text{h. o. t.}$, where $c \in \mathbb{C}$ is the coefficient of the lowest order term in the sum. Therefore

$$\mathcal{V}_p(\widetilde{A}\widetilde{B})_{ij} = \max_{1\le k\le n} (\mathcal{V}_p(\tilde{a}_{ik}) + \mathcal{V}_p(\tilde{b}_{kj})) = \big(\mathcal{V}_p(\widetilde{A}) \otimes \mathcal{V}_p(\widetilde{B})\big)_{ij}, \tag{4.8}$$

unless $c = 0$, which is only possible if the maximum in (4.8) is attained more than once, in which case $\mathcal{V}_p(\widetilde{A}\widetilde{B})_{ij} < \max_{1\le k\le n}\big(\mathcal{V}_p(\tilde{a}_{ik}) + \mathcal{V}_p(\tilde{b}_{kj})\big) = \big(\mathcal{V}_p(\widetilde{A}) \otimes \mathcal{V}_p(\widetilde{B})\big)_{ij}$. ∎

We are now ready to prove Theorems 3.4 and 3.6.

Proof of Theorem 3.4. Suppose that $\mathcal{A} = \mathcal{V}_p(\widetilde{A}) \in \overline{\mathbb{R}}^{n\times n}$ admits max-plus LU factors $\mathcal{L},\mathcal{U} \in \overline{\mathbb{R}}^{n\times n}$, where $\widetilde{A} \in \mathbb{C}\{\{z\}\}^{n\times n}$ is such that $\mathcal{L}(\widetilde{A}) \in \mathcal{G}_n$. Then by Theorem 4.5, $\widetilde{A}$ has LU factorization $\widetilde{A} = \widetilde{L}\widetilde{U}$ with $\mathcal{V}_p(\widetilde{L}) = \mathcal{L}$ and $\mathcal{V}_p(\widetilde{U}) = \mathcal{U}$ and by Lemma 4.6 the product $\mathcal{L} \otimes \mathcal{U}$ balances $\mathcal{A} = \mathcal{V}_p(\widetilde{A})$. ∎

Proof of Theorem 3.6. Let $\widetilde{A} \in \mathbb{C}\{\{z\}\}^{n\times n}$ satisfy the conditions in the statement of Theorem 4.5. Now let $\widetilde{U}^{(k)} = \widetilde{M}_k \cdots \widetilde{M}_1 \widetilde{A} = \widetilde{M}_k \widetilde{U}^{(k-1)}$ be the matrix obtained after k steps of Gaussian elimination applied to $\widetilde{A} = \widetilde{U}^{(0)}$, where $\widetilde{M}_k = I - \widetilde{m}_k e_k^T$, with e_k the kth unit vector, $\widetilde{m}_k = [0,\dots,0,\widetilde{m}_{k+1,k},\dots,\widetilde{m}_{n,k}]^T$, and $\widetilde{m}_{ik} = \widetilde{u}_{ik}^{(k-1)}/\widetilde{u}_{kk}^{(k-1)}$. By Lemma 4.6, the product $\mathcal{V}_p(\widetilde{M}_k) \otimes \mathcal{V}_p(\widetilde{U}^{(k-1)})$ balances $\mathcal{V}_p(\widetilde{U}^{(k)})$, which yields

$$\mathcal{V}_p(\widetilde{u}_{ij}^{(k)}) =: u_{ij}^{(k)} \le \max\{u_{ij}^{(k-1)}, u_{ik}^{(k-1)} + u_{kj}^{(k-1)} - u_{kk}^{(k-1)}\}, \quad i,j \ge k. \tag{4.9}$$

Next, we show by induction on k that $u_{ij}^{(k)} \le \max_{p,q} a_{p,q}$ for all i,j, and k. Since $\widetilde{U}^{(0)} = \widetilde{A}$, $\mathcal{V}_p(\widetilde{U}^{(0)}) = \mathcal{V}_p(\widetilde{A})$ so that $u_{ij}^{(0)} \le \max_{p,q} a_{p,q}$ for all i,j. Assume that $u_{ij}^{(\ell)} \le \max_{p,q} a_{p,q}$ for all i,j and $\ell \le k-1$. Since $\mathcal{A} = \mathcal{V}_p(\widetilde{A})$ is partial pivoting free, $u_{kk}^{(k-1)} \ge u_{ik}^{(k-1)}$, which combined with (4.9) and the induction hypothesis gives

$$u_{ij}^{(k)} \le \max\{u_{ij}^{(k-1)}, u_{kj}^{(k-1)}\} \le \max_{p,q} a_{p,q}$$

for all i,j. Hence $\varrho_n(\mathcal{A}) = \max_{0\le k\le n-1}\Big(\max_{k\le i,j\le n} u_{ij}^{(k)}\Big) - \max_{1\le i,j\le n} a_{ij} \le 0$. But, by definition, $\varrho_n(\mathcal{A}) \ge 0$ so $\varrho_n(\mathcal{A}) = 0$. ∎

5. Max-plus LU algorithm. Computing the max-plus LU factors directly from the formulae (3.3)–(3.4) is computationally expensive as each entry in either the lower part of $\mathcal{L}$ or the upper part of $\mathcal{U}$ requires the computation of two max-plus permanents, or in other words, the solution of two optimal assignment problems. The best known algorithms for computing an optimal assignment of $\mathcal{A} \in \mathbb{R}_{\max}^{n\times n}$ have worst case cost $O\big(n\tau + n^2\log n\big)$, where τ is the number of nonzeros in $\mathcal{A}$. So the computation of the LU factors using (3.3)–(3.4) can cost as much as $O\big(n^2\tau + n^3\log n\big)$ operations. We now describe a more efficient approach, which consists of simultaneously computing all the permanents needed for all the entries in a row of $\mathcal{U}$ or a column of $\mathcal{L}$, while sharing some of the computation along the way. The bipartite graph set-up underpinning our method will be familiar to readers who already have some knowledge of primal dual optimal assignment solvers such as the Hungarian algorithm.

To $\mathcal{A} = (a_{ij}) \in \overline{\mathbb{R}}^{n\times n}$, we associate a bipartite graph $G = (X, Y; E)$ with left vertices $X = \{x(1),\dots,x(n)\}$, right vertices $Y = \{y(1),\dots,y(n)\}$, and edge set E. We include an edge e_{ij} in E from $x(i)$ to $y(j)$ with weight $w(e_{ij}) = a_{ij}$ whenever $a_{ij} \ne -\infty$. Thus the edges out of a left vertex $x(i)$ represent the finite entries in the ith row of $\mathcal{A}$ (see Figure 5.1(a)).

A *matching* M is a subset of E with the property that no vertex in M is incident to more than one edge. Vertices which are incident to edges in M are said to be

$$\mathcal{A} = \begin{bmatrix} a & b & -\infty \\ c & d & e \\ -\infty & f & -\infty \end{bmatrix}$$

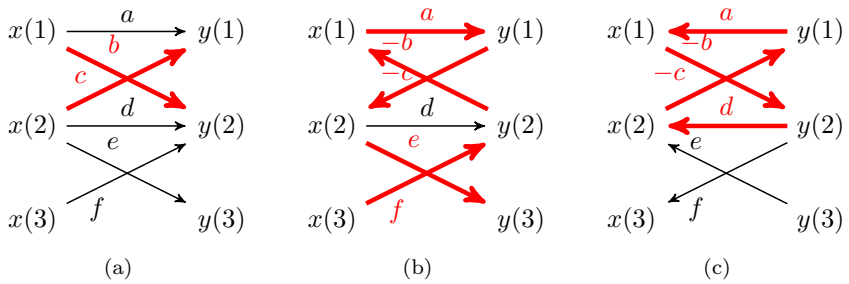


FIG. 5.1. (a) *Bipartite graph G of $\mathcal{A}$ with matching $M = \{e_{12}, e_{21}\}$ highlighted with thicker lines.* (b) *Residual graph $R_G(M)$ with directed path $\sigma = \{e_{32}, e_{12}, e_{11}, e_{21}, e_{23}\}$ highlighted with thicker lines.* (c) *Transpose residual graph $R_G(M)$ with cycle $c = \{e_{12}, e_{22}, e_{21}, e_{11}\}$ highlighted with thicker lines.*

matched. The *weight* of a matching $w(M)$ is the sum of its edge weights. Given a matching M we define the *residual graph* $R_G(M)$ to be the bipartite graph obtained from G by reversing the direction of all of the edges in M and changing the sign of the edges' weight (see Figure 5.1(b)). Note that we maintain the labelling of edges even when they are reversed. Thus e_{ij} labels either the forwards edge from $x(i)$ to $y(j)$ or the backwards edge from $y(j)$ to $x(i)$, depending on wether or not is has been reversed. We do not switch to labelling this edge as e_{ji}. We define the weight $w(\sigma)$ of a directed path or cycle σ through $R_G(M)$ to be equal to the sum of the weights of its constitute edges in $R_G(M)$. For the directed path $\sigma = \{e_{32}, e_{12}, e_{11}, e_{21}, e_{23}\}$ through $R_G(M)$ in Figure 5.1(b), $w(\sigma) = f - b + a - c + e$. We denote by $R^T(M)$ the *transpose residual graph* obtained from $R(M)$ by reversing the direction of all edges (see Figure 5.1(c)).

Given a subset S of the edges of $R_G(M)$ we *augment M according to S*, written as $M \triangle S$, by taking all the edges that appear in either M or S but not both, that is,

$$M \triangle S := \{M \cup S\} \setminus \{M \cap S\}.$$

When we augment with respect to a path/cycle trough $R_G(M)$, we treat the path/cycle as a set of edges. For the path $\sigma = \{e_{32}, e_{12}, e_{11}, e_{21}, e_{23}\}$ in Figure 5.1(b), we have that $M \triangle \sigma = \{e_{11}, e_{23}, e_{32}\}$, which is a matching between the left vertices $\{x(1), x(2), x(3)\}$ and the right vertices $\{y(1), y(2), y(3)\}$ with weight $w(M \triangle \sigma) = a + e + f = w(M) + w(\sigma)$.

A maximally weighted spanning tree T through $R(M)$ rooted at $x(k)$ consists of the maximally weighted paths from $x(k)$ to every reachable left and right vertex. The *depth* of a reachable vertex is the weight of the corresponding maximally weighted path in T. If T does not reach a vertex then this vertex has depth $-\infty$.

Our max-plus LU algorithm relies on the following result.

PROPOSITION 5.1. *Let $\mathcal{A} \in \overline{\mathbb{R}}^{n \times n}$ have max-plus LU factors $\mathcal{L} = (\ell_{ij})$ and $\mathcal{U} = (u_{ij})$, and let G be the bipartite graph associated with $\mathcal{A}$ with left vertices $X = \{x(1), \ldots, x(n)\}$ and right vertices $Y = \{y(1), \ldots, y(n)\}$. Let M_ℓ be a maximally*

weighted matching between the left vertices $\{x(1), \ldots, x(\ell)\}$ *and right vertices* $\{y(1), \ldots, y(\ell)\}$. *Then*

- u_{kj} *is either the weight of the maximally weighted path through* $R(M_{k-1})$ *from* $x(k)$ *to* $y(j)$ *for* $j \geq k$, *or* $-\infty$ *if there is no such path,*
- ℓ_{ik}*is either the weight of the maximally weighted path through* $R^T(M_k)$ *from* $x(k)$ *to* $x(i)$ *for* $i > k$, *or* $-\infty$ *if there is no such path.*

The proof of Proposition 5.1 is technical and is left to Appendix A.

The sequence of maximally weighted matchings $M_1, \ldots, M_n$ can be obtained iteratively starting with $M_0 = \emptyset$. At step $k > 0$, M_{k-1} is augmented with respect to a maximally weighted path through $R_G(M_{k-1})$ from $x(k)$ to $y(k)$ to form M_k. The $n-k$ maximally weighted paths needed to calculate the kth row of $\mathcal{U}$ can be obtained by solving a single maximally weighted spanning tree problem rooted at $x(k)$. The kth column of $\mathcal{L}$ can be obtained in a similar way. These yield the following algorithm.

ALGORITHM 5.2 (Max-plus LU). *Given* $\mathcal{A} \in \overline{\mathbb{R}}^{n\times n}$ *this algorithm returns a unit lower triangular matrix* $\mathcal{L}$ *and an upper triangular matrix* $\mathcal{U}$ *such that* $\mathcal{L}$,$\mathcal{U}$ *are max-plus LU factors for* $\mathcal{A}$.

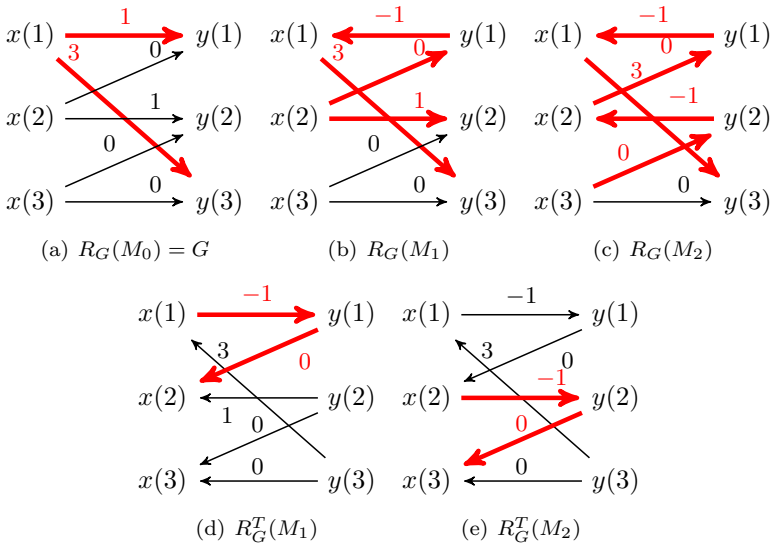
```
   % G denotes the bipartite graph associated with A with left vertices
   % X = {x(1),...,x(n)} and right vertices Y = {y(1),...,y(n)}.
1  Set the lower part of U and strictly upper part of L to −∞, and the diagonal
   entries of L to 0.
2  Set M_0 = ∅.
3  for k = 1 : n
4        Compute the maximally weighted spanning tree T through R_G(M_{k−1})
         rooted at x(k).
5        for j = k : n
6              u_kj = depth of y(j) in T.
7        end
8        M_k = M_{k−1} △ σ, where σ is the maximally weighted path through
         R_G(M_{k−1}) from x(k) to y(k).
9        Compute the maximally weighted spanning tree T' through R_G^T(M_k)
         rooted at x(k).
10       for i = k + 1 : n
11             ℓ_ik = depth of x(k) in T'.
12       end
13 end
```

If $\mathcal{A}$ does not admit max-plus LU factors then at some step k of Algorithm 5.2, $y(k)$ will have depth $-\infty$ and there will be no path from $x(k)$ to $y(k)$ so the algorithm will not be able to augment the matching M_{k-1} in line 8.

EXAMPLE 5.3. *We apply Algorithm* 5.2 *to compute the max-plus LU factors for*

$$\mathcal{A} = \begin{bmatrix} 1 & -\infty & 3 \\ 0 & 1 & -\infty \\ -\infty & 0 & 0 \end{bmatrix}.$$

$k = 1$. *The maximally weighted spanning tree* T *through* $R_G(M_0) = G$ *rooted at* $x(1)$ *is highlighted with thicker red lines in Figure* 5.2(a). *The depths of the* Y *vertices* (*i.e.,* 1 *for* $y(1)$, $-\infty$ *for* $y(2)$ *as it is not reached by the spanning tree, and* 3 *for* $y(3)$) *give the entries for the first row of* $\mathcal{U}$.



(a) $R_G(M_0) = G$ (b) $R_G(M_1)$ (c) $R_G(M_2)$

(d) $R_G^T(M_1)$ (e) $R_G^T(M_2)$

FIG. 5.2. *Bipartite graphs produced by Algorithm* 5.2 *for the matrix* $\mathcal{A}$ *of Example* 5.3. *Maximally weighted spanning trees and paths are highlighted in thicker red lines.*

The maximally weighted path σ *through* $R_G(M_0)$ *from* $x(1)$ *to* $y(1)$ *consists of a single edge* $\sigma = \{e_{11}\}$ *so that* $M_1 = M_0 \triangle \sigma = \{e_{11}\}$ *yielding the residual graph* $R_G(M_1)$ *displayed in Figure* 5.2(b).

The maximally weighted spanning tree T' *through* $R_G^T(M_1)$ *rooted at* $x(1)$ *is highlighted with thicker red lines in Figure* 5.2(d). *The depths of the* X *vertices give the entries for the first column of* $\mathcal{L}$.

$k = 2$. *Figure* 5.2(b) *highlights the maximally weighted spanning tree* T *through the residual graph* $R_G(M(1))$ *rooted at* $x(2)$. *The depths of the* Y *vertices give the entries for the second row of* $\mathcal{U}$.

The maximally weighted path σ *through* $R_G(M_1)$ *from* $x(2)$ *to* $y(2)$ *consists of a single edge* $\sigma = \{e_{22}\}$ (*Figure* 5.2(b)) *so that* $M(2) = M(1) \triangle \sigma = \{e_{11}, e_{22}\}$. *The residual graph* $R_G(M(2))$ *is shown in Figure* 5.2(c). *The maximally weighted spanning tree* T' *through* $R_G^T(M(2))$ *rooted at* $x(2)$ *is highlighted in Figure* 5.2(e). *The depths of the* X *vertices give the entries for the second column of* $\mathcal{L}$ *below the diagonal.*

$k = 3$. *The maximally weighted spanning tree* T *through* $R_G(M_2)$ *rooted at* $x(3)$ *is highlighted in Figure* 5.2(c). *The depth of the* $y(3)$ *vertex gives the entry for the third row of* $\mathcal{U}$.

The algorithm returns the max-plus LU *factors*

$$\mathcal{L} = \begin{bmatrix} 0 & -\infty & -\infty \\ -1 & 0 & -\infty \\ -\infty & -1 & 0 \end{bmatrix}, \qquad \mathcal{U} = \begin{bmatrix} 1 & -\infty & 3 \\ -\infty & 1 & 2 \\ -\infty & -\infty & 1 \end{bmatrix}.$$

Algorithm 5.2 requires the solution of maximally weighted spanning tree problems in steps 4 and 9. Note that the spanning tree T in step 4 provides the maximally weighted path σ needed in step 8. To efficiently solve the maximally weighted spanning tree problems for a given bipartite graph $G = (X, Y; E)$, we follow an approach taken by Orlin and Lee for the optimal assignment problem [15]. Their approach consists

of adjusting the edge weights of G by defining a potential $\phi : X, Y \mapsto \mathbb{R}$ so that, for each edge $e \in E$ from a vertex a to a vertex b, the new edge weight is given by $w'(e) = w(e) - \phi(a) + \phi(b)$ with the property that $w'(e) \leq 0$. This leads to adjusted path weights $w'(\sigma) = w(\sigma) - \phi(a) + \phi(b)$, for a path σ from vertex a to vertex b. Hence if σ is a maximally weighted path from a to b for the original bipartite graph then it stays maximally weighted for the bipartite graph with adjusted weights, and vice versa. Now since all the adjusted edge weights are nonpositive, Dijkstra's algorithm can then be used to compute the maximally weighted spanning trees and the depth $w'(\sigma)$ of each of its maximally weighted paths σ. Then the depth of σ for the original graph G is given by $w(\sigma) = w'(\sigma) + \phi(a) - \phi(b)$.

The computational cost of adjusting the weights using the technique in [15] is done in $O(\tau)$ operations, where τ is the number of edges in G. Dijkstra's algorithm solves the maximally weighted spanning tree problem with worst case cost $O\big(\tau + n \log n\big)$ for a graph with n vertices and τ edges. Since we need to compute $2n$ of such spanning trees, our max-plus LU algorithm applied to $\mathcal{A} \in \overline{\mathbb{R}}^{n\times n}$ will have worst case cost $O\big(n\tau + n^2 \log n\big)$, where τ is the number of finite entries in $\mathcal{A}$.

5.1. Max-plus LU algorithm for Hungarian matrices. Algorithm 5.2 simplifies if we first apply a Hungarian scaling and an optimal assignment to $\mathcal{A}$ to produce a Hungarian scaled and reordered max-plus matrix $\mathcal{H}$. In particular, if $\mathcal{A} = \mathcal{V}_c(A)$ with $A \in \mathbb{C}^{n\times n}$, then $\mathcal{H} = \mathcal{V}_c(H)$ with H as in (3.14) is Hungarian. The next lemma shows that there is no need to compute the sequence of maximally weighted matchings $M_1, \ldots, M_n$ (see step 8 of Algorithm 5.2) as these come for free for Hungarian matrices.

LEMMA 5.4. *Let $\mathcal{H} \in \overline{\mathbb{R}}^{n\times n}$ be a Hungarian matrix (i.e., $\hbar_{ij} \leq 0$, $\hbar_{ii} = 0$, $1 \leq i, j \leq n$) and let $G = (X, Y; E)$ be the corresponding bipartite graph. Then the sets of edges $M_k = \{e_{11}, \ldots, e_{kk}\}$, $k = 1, \ldots, n$ are maximally weighted matchings between the left vertices $\{x(1), \ldots, x(k)\}$ and the right vertices $\{y(1), \ldots, y(k)\}$ for $k = 1, \ldots, n$.*

Proof. We note that every principal submatrix $\mathcal{H}_k$ of order k of a Hungarian matrix $\mathcal{H}$ is Hungarian. Since $\mathcal{H}_k$ has nonpositive entries, $\sum_{i=1}^{k} \hbar_{i,\pi(i)} \leq 0$ for any $\pi \in \Pi(k)$. Now for $\pi = \mathrm{id}$, the identity permutation, $\sum_{i=1}^{k} \hbar_{ii} = 0$ so that $\pi = \mathrm{id}$ is an optimal assignment for $\mathcal{H}_k$. But an optimal assignment corresponds to a permutation representing a maximally weighted perfect matching between the left and right vertices of the bipartite graph associated with $\mathcal{H}_k$, in other words, $M_k = \{e_{11}, \ldots, e_{kk}\}$ is a maximally weighted matching between $\{x(1), \ldots, x(k)\}$ and $\{y(1), \ldots, y(k)\}$. ∎

Knowing this sequence of maximally weighted matchings *a priori* enables us to parallelize Algorithm 5.2. We no longer need to compute the maximally weighted paths through $R(M_k)$ before we can form M_{k+1} and compute the maximally weighted paths through $R(M_{k+1})$. Instead we can treat each $R(M_k)$ separately, computing the kth row of $\mathcal{U}$ and $(k-1)$th column of $\mathcal{L}$ in parallel.

ALGORITHM 5.5 (Max-plus LU for Hungarian matrix). *Given a Hungarian matrix $\mathcal{H} \in \overline{\mathbb{R}}^{n\times n}$, this algorithm returns a unit lower triangular matrix $\mathcal{L}$ and an upper triangular matrix $\mathcal{U}$ such that $\mathcal{L}$,$\mathcal{U}$ are max-plus LU factors for $\mathcal{H}$.*

% G denotes the bipartite graph associated with $\mathcal{H}$ with left vertices
% $X = \{x(1), \ldots, x(n)\}$ and right vertices $Y = \{y(1), \ldots, y(n)\}$.

1 Set the lower part of $\mathcal{U}$ and strictly upper part of $\mathcal{L}$ to $-\infty$, and the diagonal entries of $\mathcal{L}$ to 0.

2 Set $M_0 = \emptyset$.

3 for $k = 1 : n$
4 Set $M_k = \{e_{11}, \ldots, e_{kk}\}$.
5 Compute the maximally weighted spanning tree T through $R_G(M_{k-1})$ rooted at $x(k)$.
6 for $j = k : n$
7 u_{kj} = depth of $y(j)$ in T.
8 end
9 Compute the maximally weighted spanning tree T' through $R_G^T(M_k)$ rooted at $x(k)$.
10 for $i = k + 1 : n$
11 ℓ_{ik} = depth of $x(k)$ in T'.
12 end
13 end

Because the entries of $\mathcal{H}$ are nonpositive, we can use Dijkstra's algorithm to compute the maximally weighted spanning trees in steps 4 and 8. Thus our max-plus LU algorithm applied to $\mathcal{H} \in \mathbb{R}_{\max}^{n\times n}$ will have worse case cost $O\big(n\tau + n^2 \log n\big)$, where τ is the number of finite entries in $\mathcal{H}$.

Dijkstra's algorithm permanently labels vertices in decreasing order of their depth in the tree. This means that when we run Dijkstra's algorithm to compute the kth row of $\mathcal{U}$ or kth column of $\mathcal{L}$, we are given the position and value of the entries one at a time in decreasing order of their value. If we are only interested in entries that are greater than some threshold, or if we only want to know the m largest entries in each row then we can stop Dijkstra's algorithm earlier and reduce the cost considerably. The exact cost of this implementation will depend heavily on the particular details of the problem matrix. This approach will not work for a non-Hungarian scaled matrix as while Dijkstra's algorithm will always label vertices in decreasing order, it will be in the order of their adjusted depths. So we can not be sure that we have found the m largest entries until we have computed all of the adjusted depths and then converted them back into their true depths using the potential.

5.2. Max-plus LU algorithm with partial pivoting. At each step k of the max-plus algorithm, the unmatched left vertices can be permuted to maximize the weight of the augmenting path. For this, we find a maximally weighted path through $R_G(M_{k-1})$ from $\{x(k), \ldots, x(n)\}$ to $y(k)$. If this maximally weighted path roots at $x(i)$ then we swap $x(i)$ with $x(k)$ and perform step k of Algorithm 5.2. Note that a maximally weighted path through $R_G(M_{k-1})$ from $\{x(k), \ldots, x(n)\}$ to $y(k)$ or more generally to $\{y(k), \ldots, y(n)\}$ can be obtained by solving one maximally weighted spanning tree problem. This is done by adding a root vertex r and connecting it to each unmatched left vertex $x(j)$, $j = k, \ldots, n$ by an edge of weight zero (see Figure 5.3(a)). We then compute the maximally weighted spanning tree through $R_G(M_{k-1}) \cup \{r\}$ rooted at r.

ALGORITHM 5.6 (Max-plus LU with partial pivoting). *Given $\mathcal{A} \in \overline{\mathbb{R}}^{n\times n}$, this algorithm returns a permutation π, a unit lower triangular matrix $\mathcal{L}$ and an upper triangular matrix $\mathcal{U}$ such that $\mathcal{L}$,$\mathcal{U}$ are max-plus LU factors for the partial pivoting free matrix $\mathcal{P}_\pi \otimes \mathcal{A}$, where $(\mathcal{P}_\pi)_{ij} = 0$ for $j = \pi(i)$ and $(\mathcal{P}_\pi)_{ij} = -\infty$ otherwise.*

% G_π denotes the bipartite graph associated with $\mathcal{A}$ with left vertices
% $X_\pi = \{x\big(\pi(1)\big), \ldots, x\big(\pi(n)\big)\}$ and right vertices $Y = \{y(1), \ldots, y(n)\}$
% for some permutation π.
1 Set the lower part of $\mathcal{U}$ and strictly upper part of $\mathcal{L}$ to $-\infty$, and the diagonal

entries of $\mathcal{L}$ to 0.
2 Set $M_0 = \emptyset$ and $\pi = [1,\ 2,\ \ldots\ , n]$.
3 for $k = 1 : n$
4 Add a root vertex r and connect it to each left vertex $x\big(\pi(j)\big)$, $j = k, \ldots, n$ by an edge of weight zero.
5 Compute the maximally weighted spanning tree T through $R_{G_\pi}(M_{k-1})$ rooted at r.
6 Swap $\pi(k)$ with $\pi(i)$, where $x\big(\pi(i)\big)$ is the 2nd vertex on the maximally weighted path from r to $y(k)$.
7 for$j = k, \ldots, n$
8 u_{kj} = depth of $y(j)$ in T
9 end
10 $M_k = M_{k-1} \triangle \sigma$, where σ is the maximally weighted path through $R_{G_\pi}(M_{k-1})$ from $x\big(\pi(k)\big)$ to $y(k)$.
11 Compute the maximally weighted spanning tree T' through $R^T_{G_\pi}(M_k)$ rooted at $x\big(\pi(k)\big)$.
12 for $i = k + 1 : n$
13 ℓ_{ik} = depth of $x\big(\pi(k)\big)$ in T'
14 end
15 end

Line 6 in Algorithm 5.6 is the interchanging step. It is not difficult to see that $\mathcal{P}_\pi \otimes \mathcal{A}$ is partial pivoting free.

EXAMPLE 5.7. *Let us apply Algorithm* 5.6 *to* $\mathcal{A} = \left[\begin{smallmatrix} 1 & 2 \\ 3 & 5 \end{smallmatrix}\right]$. *Let* $\pi = [1, 2]$.

$k = 1$. *We adjoin a root vertex* r *to the bipartite graph* $R_{G_\pi}(M_0) = G_\pi$ *and connect it to the left vertices of* $x(\pi(1))$, $x(\pi(2))$ *by an edge of weight zero. The maximally weighted spanning tree* T *through* $R_{G_\pi}(M_0) \cup \{r\}$ *rooted at* r *is shown with thicker lines in Figure* 5.3(a). *The depths of the* Y *vertices (i.e.,* 3 *for* $y(1)$ *and* 5 *for* $y(2)$) *define the entries of the first row of* $\mathcal{U}$. *Since the maximally weighted path is the one from* $x(\pi(2))$, *we we swap* $\pi(1)$ *with* $\pi(2)$ *so that* $\pi = [2\ 1]$. *Then the maximally weighted path* σ *through* $R_{G_\pi}(M_0)$ *from* $x(\pi(1))$ *to* $y(1)$ *consists of a single edge* $\{e_{\pi(1),1}\} = \{e_{21}\}$ *so that* $M_1 = M_0 \triangle \{e_{21}\} = \{e_{21}\}$ *yielding the residual graph* $R_{G_\pi}(M_1)$ *displayed in Figure* 5.3(b). *The maximally weighted spanning tree* T' *through* $R^T_{G_\pi}(M_1)$ *rooted at* $x(\pi(1)) = x(2)$ *is highlighted with thicker red lines in Figure* 5.3(c). *The depths of the path from* $x(\pi(1))$ *to* $x(\pi(2))$, *i.e.,* -2, *defines* ℓ_{21}.

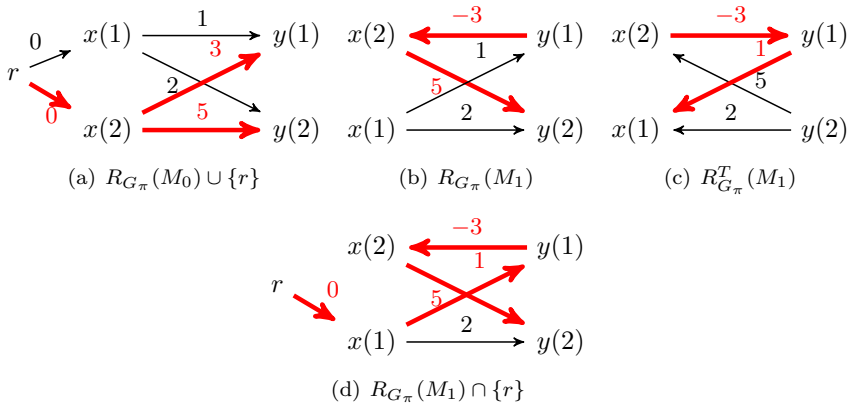
$k = 2$. *We adjoin a root vertex* r *to the bipartite graph* $R_{G_\pi}(M_1)$. *Figure* 5.3(d) *highlights the maximally weighted spanning tree* T *through* $R_{G_\pi}(M_1) \cup \{r\}$ *rooted at* r. *The depths of the* Y *vertices define the last row of* $\mathcal{U}$.

Hence we obtain

$$P_\pi \otimes \mathcal{A} = \begin{bmatrix} -\infty & 0 \\ 0 & -\infty \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix} = \begin{bmatrix} 3 & 5 \\ 1 & 2 \end{bmatrix}, \quad \mathcal{L} = \begin{bmatrix} 0 & -\infty \\ -2 & 0 \end{bmatrix}, \quad \mathcal{U} = \begin{bmatrix} 3 & 5 \\ -\infty & 3 \end{bmatrix}.$$

5.3. Max-plus incomplete LU preconditioner. Given the max-plus LU factors $\mathcal{L}$ and $\mathcal{U}$ of $\mathcal{V}_c(A)$ we define the max-plus ILU preconditioner as follows. For a threshold t we store

$$S_{ij} = \begin{cases} 1 & \text{if } \ell_{ij} \geq \log t + \max_k \log |a_{ik}| \text{ or } u_{ij} \geq \log t + \max_k \log |a_{ik}|, \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$



(a) $R_{G_\pi}(M_0) \cup \{r\}$ (b) $R_{G_\pi}(M_1)$ (c) $R^T_{G_\pi}(M_1)$

(d) $R_{G_\pi}(M_1) \cap \{r\}$

FIG. 5.3. *Bipartite raphs produced by Algorithm* 5.6 *applied to* $\mathcal{A} = \left[\begin{smallmatrix} 1 & 2 \\ 3 & 5 \end{smallmatrix}\right]$.

We then compute the incomplete LU factors for A restricted to positions where S is nonzero using, for example, the general static pattern ILU algorithm describe in [16, Alg. 10.1] or the more practical variant [16, Alg. 10.3].

6. Numerical experiments. For our numerical experiments, we select all real nonsymmetric matrices in the University of Florida sparse matrix collection [4] of size $100 \leq n \leq 5000$ that have numeric value symmetry no greater than 0.9, and that are structurally nonsingular. When two matrices from the same group have size n and number of nonzero entries within 1% of each other then we consider these matrices as duplicate and remove one of them. This leaves us with a total of 260 matrices. We used MATLAB to perform the computations. Note that our max-plus LU algorithms are implemented as research codes rather than efficient implementations and, for this reason, we only work with matrices of moderate sizes. For the valuation $\mathcal{V}_c$, we use the logarithm to base 10.

6.1. Stability of Gaussian elimination with no pivoting. The aim of our first set of experiments is to compare the numerical stability of Gaussian elimination with no pivoting applied to Hungarian scaled and reordered matrices H in (3.14), and to reordered matrices $P_\pi A$, where π is the permutation returned by Algorithm 5.6. For each matrix A in the test set we construct H using the HSL code MC64 [11] and $P_\pi A$ using our MATLAB implementation of Algorithm 5.6. Theorem 3.6 together with Heuristic 3.7 suggest that the growth factors for both H and $P_\pi A$ are of order one since both $\mathcal{V}_c(H)$ and $\mathcal{V}_c(P_\pi A)$ are partial pivoting free. Although this is just a heuristic, we expect the LU factorization with no pivoting of H and $P_\pi A$ to have better numerical stability than for A. To examine the stability of Gaussian elimination on these two classes of matrices we compute the relative backward errors

$$\eta_X = \frac{\|X - \widehat{L}\widehat{U}\|_F}{\|X\|_F}$$

for $X = H$ and $X = P_\pi A$, where $\widehat{L}$ and $\widehat{U}$ are the computed LU factors of the LU factorization of X. We also use Gaussian elimination with no pivoting to compute the LU factorizations of the original matrices A. For each class of matrices, i.e., $X = A, H$ and $P_\pi A$, we plot in Figure 6.1 the proportion of problems for which we

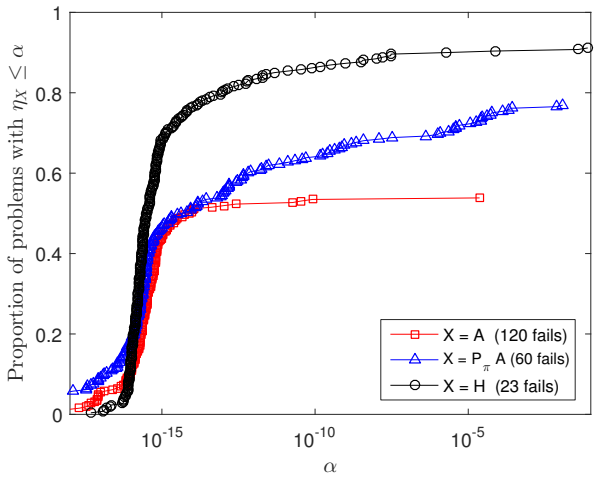


FIG. 6.1. *Proportion of problems with relative backward error $\eta_X \leq \alpha$ for $X = A, H, P_\pi A$.*

are able to compute LU factors without breakdown and with $\eta_X \leq \alpha$ against α. If the factorization breaks down or if $\eta_X \geq 10^{-1}$, we record a fail. Without pivoting or scaling, the LU factorization fails for almost half of the test matrices A and $\eta_A \leq 10^{-10}$ for 53% of the test matrices. After applying the max-plus LU permutation P_π to A, the number of failed LU factorizations falls from 120 to 60, and $\eta_{P_\pi A} \leq 10^{-10}$ for 64% of the test matrices. The number of failed LU factorizations is lower for Hungarian scaled matrices H (only 23 fails) and $\eta_H \leq 10^{-10}$ for 86% of the test matrices. Since our aim is to build a new class of incomplete LU preconditioners and

- for the vast majority of matrices in the test set a reasonably stable LU factorization with no pivoting is possible if A is Hungarian scaled and reordered into H,
- Hungarian scaling has been shown experimentally to be beneficial for iterative methods [2],
- the max-plus LU algorithm is easier to implement for Hungarian scaled matrices,

from here on we will work with the subset of test problems for which the Hungarian scaled and reordered matrix H can be factorized with no pivoting and with $\eta_H < 0.1$. This results in a test subset of 233 matrices.

6.2. Max-plus LU approximation. The max-plus LU approximation can assist in the computation of an ILU preconditioner by providing a prediction of the positions of larger entries in the LU factors of H. One way of measuring the quality of this prediction is to treat the max-plus LU approximation as a binary classifier. For the LU factorization $H = LU$ of a matrix H from the test set and its max-plus LU factors $\mathcal{L}$,$\mathcal{U}$, we predict that $|l_{ij}| \geq 10^{-t}$ for $i > j$ if and only if $\mathcal{l}_{ij} \geq -t$, and likewise for the entries of U. The entries of L and U are then labeled as true positive or true negative according to the scheme displayed in Figure 6.2(a) (for example, for a given t, the (i, j) entry of $\mathcal{L}$ is true positive if $\mathcal{l}_{ij} \geq -t$ and $\log_{10} |l_{ij}| \geq -t$). The accuracy of the classifier is defined by

$$\text{accuracy} = \frac{\text{number of true positives and true negatives}}{\text{number of nonzeros in } L \text{ and } U},$$

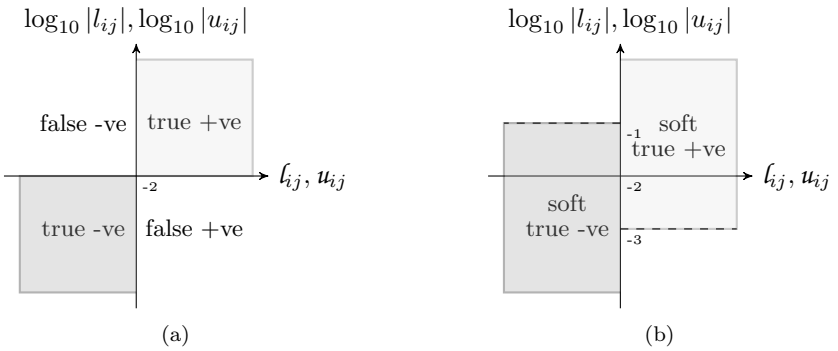


FIG. 6.2. *Labelling of the nonzero entries in the L and U factors for calculating (soft) accuracy and (soft) precision for $t = 2$. The axes cross at $(-2, -2)$.*

TABLE 6.1
Proportion of test problems with with (soft) accuracy and (soft) precision greater than p for $t = 2$.

p	0.8	0.85	0.9	0.95
$A(p)$	85%	83%	79%	71%
$SA(p)$	97%	93%	90%	85%
$P(p)$	86%	83%	80%	59%
$SP(p)$	93%	91%	89%	72%

and the precision is defined by

$$\text{precision} = \frac{\text{number of true positives}}{\text{number of entries in } L \text{ and } U \text{ such that } |l_{ij}| \geq 10^{-t} \text{ and } |u_{ij}| \geq 10^{-t}}.$$

We also define the soft accuracy and soft precision, which are calculated in the same way using the number of entries in L and U that are soft true positive and soft true negative, where the labelling "soft true positive" and "soft true negative" is done according to the scheme displayed in Figure 6.2(b). We record in Table 6.1 the following proportions:

$$\begin{aligned} A(p) &= \text{proportion of test examples with accuracy} \geq p, \\ SA(p) &= \text{proportion of test examples with soft accuracy} \geq p, \\ P(p) &= \text{proportion of test examples with precision} \geq p, \\ SP(p) &= \text{proportion of test examples with soft precision} \geq p, \end{aligned}$$

for $p = 0.80, 0.85, 0.9, 0.95$ and $t = 2$. The scores in Table 6.1 are quite high and show that, for our test set, the max-plus LU factors provide a good prediction of the position of the larger entries in L and U.

6.3. Behaviour of max-plus ILU preconditioning. In this section, we examine the behaviour of our max-plus ILU preconditioner on our test set of 233 Hungarian matrices H. We apply GMRES with a right ILU preconditioner to the systems $Hx = b$. For the ILU preconditioner, we use

1. threshold ILU as implemented in MATLAB's `ilu` function with the options
   `setup.type = 'crout'; setup.milu = 'off';`

TABLE 6.2
Number of test matrices N for which ILU(k) is used with level k.

Level k	0	1	2	3	4	5	≥ 6
# of test matrices	104	45	28	10	8	4	34

2. ILU(k) through the MATLAB function `iluk` from [12] and level of fill in k,
3. ILU(0) with zero level of fill in,
4. max-plus ILU by forming the pattern matrix S in (5.1) and by calling `iluk` with input parameters set up to bypass the symbolic factorization.

For both threshold ILU and max-plus ILU, we used 10^{-2} as the drop-off tolerance. The level k for ILU(k) is chosen as the smallest integer such that the resulting ILU factors are denser than those obtained by the max-plus method. We justify this choice at the end of this section. The distribution of the levels k is shown in Table 6.2.

We use unrestarted GMRES with tolerance set to 10^{-5}. A test matrix is marked as a fail if GMRES fails to converge within `maxit` iterations. For each preconditioned system, we record the number of GMRES iterations required for convergence multiplied by the sum of the number of nonzero entries in H and the number of nonzero entries in the ILU factors. For the unpreconditioned systems we record the number of GMRES iterations multiplied by the number of nonzero entries in H. These measures give an approximation of the cost of the GMRES solves but do not include the cost of constructing the preconditioner. The left plot in Figure 6.3 is a performance profile that compares this cost measure over the different ILU strategies. For the unpreconditioned systems, GMRES fails to converge in less than `maxit`=100 iterations for 144 out of 233 problems and has clearly the worse performance. Systems preconditioned by ILU(k) have a cost greater than double that of the best method about 50% of the problems. The figure shows that the performance of the max-plus ILU preconditioned systems is close to that of the standard threshold ILU preconditioned systems. For about 80% of problems the cost of the max-plus method is within a factor of 2 of the best method. We performed the same set of tests but with BICGSTAB in place of GMRES for the iterative solver. The corresponding performance profile (see right plot in Figure 6.3) shows that the different ILU preconditoners exhibit the same behaviour as for GMRES.

The cost measure that we have used tends to decrease with the number of nonzero positions included in the ILU factors for all three different ILU techniques. As a result, our choice for the level k is slightly generous to the ILU(k) method. Of course the total cost of solving the linear system should also include the cost of computing the ILU factors, which increases with the number of nonzero positions, so that in practice choosing denser ILU factors is not an advantage. We use this cost measure here for its simplicity. We note that the cost of computing the max-plus ILU preconditioner is smaller than that of the standard threshold ILU preconditioner and roughly the same as computing the ILU(k) preconditioner.

7. Conclusion. We presented a new method for approximating the order of magnitude of the entries in the LU factors of a matrix $A \in \mathbb{C}^{n\times n}$. This method uses max-plus algebra and is based solely on the moduli of the entries in A. If the matrix A is first Hungarian scaled and reordered then this LU approximation can be computed in parallel with n independent computations of cost $O\big(\tau + log(n)\big)$. If we seek only the positions and values of the largest entries in the LU factors then this cost can be reduced further.

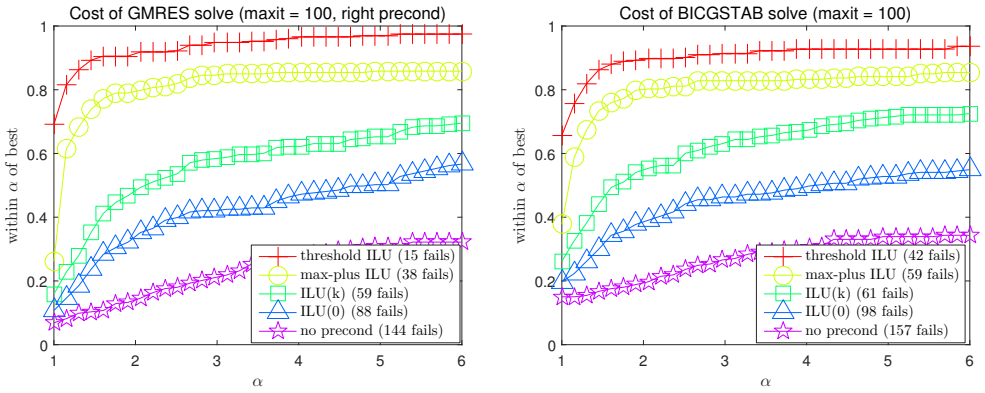


FIG. 6.3. *Performance profile comparing relative costs of GMRES (left plot) and BICGSTAB (right plot) solves for different ILU preconditioning strategies.*

We have shown that this approximation can be used to help compute an ILU preconditioner for A. First we reorder and rescale A to obtain a Hungarian matrix H, then we compute the positions of the largest entries in the LU factors of H and finally we use these positions as the sparsity pattern for an ILU preconditioner. The resulting preconditioner tends to outperform the comparable ILU(k) preconditioner and have performance very close to a comparable threshold ILU preconditioner.

The numerical examples presented in this paper represent a proof of principal that the max-plus ILU technique can be advantageous in the solution of sparse linear systems.

REFERENCES

[1] M. Akian, S. Gaubert, and A. Marchesini. Tropical bounds for eigenvalues of matrices. *Linear Algebra Appl.*, 446:281–303, 2014.

[2] M. Benzi, J. C. Haws, and M. Tůma. Preconditioning highly indefinite and nonsymmetric matrices. *SIAM J. Sci. Comput.*, 22(4):1333–1353, 2000.

[3] D. A. Bini and V. Noferini. Solving polynomial eigenvalue problems by means of the Ehrlich-Aberth method. *Linear Algebra Appl.*, 439(4):1130–1149, 2013.

[4] T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Trans. Math. Software*, 38(1):1:1–1:25, 2011.

[5] F. R. Gantmacher. *The Theory of Matrices*, volume one. Chelsea, New York, 1959. ISBN 0-8284-0131-4. x+374 pp.

[6] S. Gaubert and M. Sharify. Tropical scaling of polynomial matrices. In *Positive systems*, volume 389 of *Lecture Notes in Control and Information Sciences*, pages 291–303. Springer-Verlag, Berlin, 2009.

[7] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2002. ISBN 0-89871-521-0. xxviii+680 pp.

[8] J. Hook. Max-plus singular values. *Linear Algebra Appl.*, 486:419–442, 2015.

[9] J. Hook and F. Tisseur. Max-plus eigenvalues and singular values: a useful tool in numerical linear algebra, 2016. In preparation.

[10] A. S. Householder. *The Theory of Matrices in Numerical Analysis*. Blaisdell, New York, 1964. ISBN 0-486-61781-5. xi+257 pp. Reprinted by Dover, New York, 1975.

[11] HSL. A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk/.

[12] K. Miller. ILU(k) Preconditioner. `https://uk.mathworks.com/matlabcentral/fileexchange/48320-ilu-k--preconditioner`.

[13] V. Noferini, M. Sharify, and F. Tisseur. Tropical roots as approximations to eigenvalues of matrix polynomials. *SIAM J. Matrix Anal. Appl.*, 36(1):138–157, 2015.

[14] M. Olschowka and A. Neumaier. A new pivoting strategy for Gaussian elimination. *Linear Algebra Appl.*, 240:131–151, 1996.

[15] J. B. Orlin and Y. Lee. Quickmatch–a very fast algorithm for the assignment problem. Working papers 3547-93, Massachusetts Institute of Technology (MIT), Sloan School of Management, 1993.

[16] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2003. ISBN 0-89871-534-2. xviii+528 pp.

[17] M. Sharify. *Scaling Algorithms and Tropical Methods in Numerical Matrix Analysis: Application to the Optimal Assignment Problem and to the Accurate Computation of Eigenvalues*. PhD thesis, Ecole Polytechnique, Palaiseau, France, Sept. 2011. Available from `http://hal.archives-ouvertes.fr/docs/00/64/38/36/PDF/thesis.pdf`.

Appendix A.

This appendix presents several technical results needed to prove Proposition 5.1. We refer to Section 5 for notation and definitions.

LEMMA A.1. *Let G be a bipartite graph with left vertices $X = \{x(1), \dots, x(n)\}$ and right vertices $Y = \{y(1), \dots, y(n)\}$ and let M be a matching between $\{x(1), \dots, x(k)\}$ and $\{y(1), \dots, y(k)\}$. The following statements hold.*

(i) *If σ is a direct path through the residual graph $R_G(M)$ from the unmatched left vertex $x(k+1)$ to the unmatched right vertex $y(k+1)$ then $M \triangle \sigma$ is a matching between the left vertices $\{x(1), \dots, x(k), x(k+1)\}$ and the right vertices $\{y(1), \dots, y(k), y(k+1)\}$, with weight $w(M \triangle \sigma) = w(M) + w(\sigma)$.*

(ii) *If C is a cycle through the residual graph $R_G(M)$ then $M \triangle C$ is a matching between the same vertices as M with weight $w(M \triangle C) = w(M) + w(C)$.*

(iii) *If σ is a direct path through the residual graph $R_G(M)$ from the unmatched left vertex $x(k+1)$ to the matched left vertex $x(k)$ then $M \triangle \sigma$ is a matching between the left vertices $\{x(1), \dots, x(k+1)\} \setminus \{x(k)\}$ and the right vertices $\{y(1), \dots, y(k)\}$ with weight $w(M \triangle \sigma) = w(M) + w(\sigma)$.*

(iv) *If S_1 and S_2 are disjoint subsets of edges, each either a path or a cycle in $R_G(M)$, then $w\big(M \triangle (S_1 \cup S_2)\big) = w(M) + w(S_1) + w(S_2)$.*

Proof. (i) There are no edges into any unmatched left vertices or out of any unmatched right vertices, so that σ can only visit its origin vertex $x(k+1)$, destination vertex $y(k+1)$ as well as the vertices matched by M. Since σ is a path from $x(k+1)$ it must include exactly one edge out of this vertex and since this vertex is unmatched in M there can be no edge out of it in M. Thus $M \triangle \sigma$ contains exactly one edge incident to the origin vertex $x(k+1)$. Likewise for the destination vertex, $M \triangle \sigma$ contains exactly one edge incident to $y(k+1)$.

Let u be a matched left vertex visited by σ. Then σ must include an edge into u, which being a right-to-left edge must be an edge in M, σ must also include an edge out of u, which being a left-to-right edge must not be in M. Thus $M \triangle \sigma$ contains exactly one edge incident to u. Likewise if v is a matched right vertex visited by σ then $M \triangle \sigma$ contains exactly one edge incident to v.

Matched vertices that are not visited by σ are unaffected, likewise unmatched vertices are unaffected. Thus $M \triangle \sigma$ is a subset of E, with exactly one edge incident to each of the left vertices $\{x(1), \dots, x(k), x(k+1)\}$ and each of the right vertices $\{y(1), \dots, y(k), y(k+1)\}$, and no edge incident to any other vertices.

The weight of $M \triangle \sigma$ is equal to $w(M)$ plus the weight of any edges in σ but not in M minus the weight of any edges in M and σ. Since any edge in $M \cap \sigma$ is a backward edge with a extra minus sign and any edge in σ / M is a forwards edge without an extra minus sign, we have $w(M \triangle \sigma) = w(M) + w(\sigma)$.

(ii) The proof of (ii) is the same as for (i) but without the origin or destination

vertices.

(iii) The argument is the same as (i) except for the destination vertex $x(k)$. This vertex is incident to exactly one edge in M, which is a right-to-left vertex, since σ ends at this vertex it must also contain this edge and therefore $M\triangle\sigma$ does not contain an edge incident to $x(k)$.

(iv) Since $S_1 \cup S_2 = \emptyset$ we have $M\triangle(S_1 \cup S_2) = (M\triangle S_1)\triangle S_2$. Hence,

$$w\big((M\triangle S_1)\triangle S_2\big) = w(M\triangle S_1) + \hat{w}(S_2),$$

where $\hat{w}(S_2)$ is the weight of the edge set S_2 in the residual graph $R_G(M\triangle S_1)$. However since S_1 and S_2 are disjoint we have $\hat{w}(S_2) = w(S_2)$, where $w(S_2)$ is the weight of the edge set S_2 in the residual graph $R_G(M)$. (this follows because none of the edges affected by augmenting with respect to S_1 are in the set S_2). Therefore we have

$$\begin{aligned} w\big(M\triangle(S_1 \cup S_2)\big) &= w\big((M\triangle S_1)\triangle S_2\big) \\ &= w(M\triangle S_1) + w(S_2) \\ &= w(M) + w(S_1) + w(S_2). \qquad \square \end{aligned}$$

For the residual graph $R_G(M)$ in Figure 5.1(b) and

(i) the path $\sigma = \{e_{32}, e_{12}, e_{11}, e_{21}, e_{23}\}$, we have that $M\triangle\sigma = \{e_{11}, e_{23}, e_{32}\}$, which is a matching between the left vertices $\{x(1), x(2), x(3)\}$ and the right vertices $\{y(1), y(2), y(3)\}$ with weight $w(M\triangle\sigma) = a + e + f = w(M) + w(\sigma)$,

(ii) the cycle $c = \{e_{22}, e_{12}, e_{11}, e_{21}\}$ through $R_G(M)$ with weight $w(\sigma) = d - b + a - c$ we have that $M\triangle c = \{e_{11}, e_{22}\}$, which is a matching between the left vertices $\{x(1), x(2)\}$ and the right vertices $\{y(1), y(2)\}$ with weight $w(M\triangle c) = a + d = w(M) + w(c)$,

(iii) the path $\sigma = \{e_{32}, e_{12}, e_{11}, e_{21}\}$ through $R_G(M)$ with weight $w(\sigma) = f - b + a - c$, we have that $M\triangle\sigma = \{e_{11}, e_{32}\}$, which is a matching between the left vertices $\{x(1), x(3)\}$ and the right vertices $\{y(1), y(2)\}$, with weight $w(M\triangle\sigma) = a + f = w(M) + w(\sigma)$.

LEMMA A.2. *Let G be the bipartite graph associated with $\mathcal{A} \in \overline{\mathbb{R}}^{n\times n}$ with left vertices $\{x(1), \ldots, x(n)\}$ and right vertices $\{y(1), \ldots, y(n)\}$. Then* $\mathrm{perm}(\mathcal{A}([i_1, \ldots, i_k], [j_1, \ldots, j_k])$ *is the weight of the maximally weighted matching between the left vertices $\{x(i_1), \ldots, x(i_k)\}$ and the right vertices $\{y(i_1), \ldots, y(i_k)\}$.*

Proof. Any matching M between the left vertices $\{x(1), \ldots, x(k)\}$ and the right vertices $\{y(1), \ldots, y(k)\}$ can be represented by a unique permutation $\pi \in \Pi(k)$ so that M_π is the matching that matches $x(i)$ to $y\big(\pi(i)\big)$. Now consider $w(M_\pi) = \sum_{i=1}^{k} a_{x(i)y(\pi(i))}$ so that the weight of the maximally weighted matching is given by

$$\max_{\pi\in\Pi(k)} w(M_\pi) = \max_{\pi\in\Pi(k)} \sum_{i=1}^{k} a_{x(i)y(\pi(i))} = \mathrm{perm}(\mathcal{A}([i_1, \ldots, i_k],\ [j_1, \ldots, j_k]). \qquad \square$$

LEMMA A.3. *Let $G = (X, Y; E)$ be a bipartite graph and let $M_\pi \subset E$, $M_\omega \subset E$ be matchings defined by the permutations $\pi \in \Pi(k)$ and $\omega \in \Pi(k+1)$, respectively. Then there exists a path σ through $R_G(M)$ from $x(k+1)$ to $y(k+1)$ as well as disjoint cycles $C_1, \ldots, C_m$ through $R_G(M)$, such that $M_\omega = M_\pi\triangle\big(\sigma \cup C_1 \cup \cdots \cup C_m\big)$.*

Proof. We will prove the lemma by constructing the path and cycles as follows. Set $\sigma(1) = x(k+1)$, then set

$$\begin{aligned}\sigma(2) &= y\big(\omega(k+1)\big), & \sigma(3) &= x\big(\pi^{-1}\omega(k+1)\big),\\ \sigma(4) &= y\big(\omega\pi^{-1}\omega(k+1)\big), & \sigma(5) &= x\big(\pi^{-1}\omega\pi^{-1}\omega(k+1)\big),\ \ldots.\end{aligned}$$

There is no right vertex $y(j)$ with $\pi^{-1}(j) = k+1$ since the domain of π is $\{1,\ldots,k\}$. Also all subsequent vertices visited by the constructed path can only have one predecessor as π and ω are permutations. Therefore σ cannot contain any cycle and must terminate since there are only finitely many vertices that it can visit without repetition. The only way that the path can terminate is by reaching a vertex where the next step is not well defined and the only such vertex is $y(k+1)$. The constructed path therefore terminates at $\sigma(2\ell) = y(k+1)$.

Next we pick any left vertex x matched by M_π, which is not visited by σ. We construct a new path starting from x with the same rule that we used for constructing σ. Since there are no possible termination point, where π^{-1} or ω are not defined, this new path must be cyclic. If the constructed cycle is of length 2 then we discount the cycle but still record the constituent vertices as having been visited. We construct further cycles $C_1,\ldots,C_m$ until all matched vertices have been visited.

By construction each vertex matched by M_π either has the same matching under M_ω or is incident to two edges in $\sigma \cup C_1 \cup \cdots \cup C_m$. One edge from M_π and one from M_ω. When we augment by taking the symmetric difference, the edge from M_π is replaced by the one from M_ω. Likewise the origin and destination vertices are each incident to an edge which is in M_ω but not in M_π, so these edges are also included when we augment. □

The following theorem shows us how we can obtain a sequence of maximally weighted matchings by augmenting with respect to maximally weighted paths through the residual graph. This is the mechanism by which we will compute all of the entries in the max-plus LU factors.

THEOREM A.4. *Let $G = (X, Y; E)$ be a bipartite graph and let M be a maximally weighted matching between the left vertices $\{x(1),\ldots,x(k)\}$ and the right vertices $\{y(1),\ldots,y(k)\}$. The following hold.*

(i) *If σ is a maximally weighted path trough $R_G(M)$ from the unmatched left vertex $x(k+1)$ to the unmatched right vertex $y(k+1)$ then $M\triangle\sigma$ is a maximally weighted matching between the left vertices $\{x(1),\ldots,x(k+1)\}$ and the right vertices $\{y(1),\ldots,y(k+1)\}$.*

(ii) *If σ be a maximally weighted path through $R_G(M)$ from the unmatched left vertex $x(k+1)$ to the matched left vertex $x(k)$ then $M\triangle\sigma$ is a maximally weighted matching between the left vertices $\{x(1),\ldots,x(k-1),\, x(k+1)\}$ and the right vertices $\{y(k),\ldots,y(k)\}$.*

Proof. (i) Let σ be a path through $R_G(M)$ and let $C_1,\ldots,C_m$ be disjoint cycles in $R_G(M)$. Then by Lemma A.1(iv) $w\big(M\triangle(\sigma\cup C_1\cup\cdots\cup C_m)\big) = w(M) + w(\sigma) + w(C_1) + \cdots + w(C_m)$. It follows from Lemma A.1(ii) that if C is single cycle then $M\triangle C$ is a matching between the same vertices as M, and since M is the maximally weighted matching on its matched vertices, $w(M\triangle C) = w(M) + w(C) \le w(M)$ so that $w(C) \le 0$ showing that any cycle in $R_G(M)$ must have nonpositive weight.

Now consider $\max_{M'} w(M')$, where the maximum is taken over all matchings from $\{x(1),\ldots,x(k+1)\}$ to $\{y(k),\ldots,y(k+1)\}$. Lemma A.3 tells us that every matching M' from $\{x(1),\ldots,x(k+1)\}$ to $\{y(k),\ldots,y(k+1)\}$ can be written as the

augmentation of the matching M with respect to some path and cycles. Therefore

$$\max_{M'} w(M') = w(M) + \max_{\sigma, C_1, \ldots, C_m} w(\sigma) + w(C_1) + \cdots + w(C_m),$$

where the maximum is taken over all paths through $R_G(M)$ from $x(k+1)$ to $y(k+1)$ and disjoint cycles $C_1, \ldots, C_m$ in $R_G(M)$. Since the cycle weights are all nonpositive we have

$$\max_{M'} w(M') \leq w(M) + \max_{\sigma} w(\sigma)$$

and since the upper bound is attained by the matching $M' = M \triangle \arg\max_\sigma w(\sigma)$, where $\arg\max_\sigma w(\sigma)$ is a maximally weighted path through $R_G(M)$ from $x(k+1)$ to $y(k+1)$, we have

$$\max_{M'} w(M') = w\big(M \triangle \arg\max_{\sigma} w(\sigma)\big).$$

Hence $M \triangle \arg\max_\sigma w(\sigma)$ is a maximally weighted matching between the left vertices $\{x(1), \ldots, x(k+1)\}$ and the right vertices $\{y(1), \ldots, y(k+1)\}$.

(ii) This follows from Lemmas A.1 and A.3 in analogy to Theorem A.4. □

We are now ready to prove Proposition 5.1.

Proof. [Proof of Proposition 5.1]

Let M_k be a maximally weighted matching between the left vertices $\{x(1), \ldots, x(k)\}$ and the right vertices $\{y(1), \ldots, y(k)\}$ and let σ be a maximally weighted path through $R(M)$ from the unmatched left vertex $x(i)$ with $i > k$ to the matched left vertex $x(k)$. Then from Lemma A.2 we have $\operatorname{perm}\big(\mathcal{A}(1:k, 1:k)\big) = w(M)$. From Theorem A.4 we have that $M \triangle \sigma$ is the maximally weighted matching between the left vertices $\{x(1), \ldots, x(k-1), x(i)\}$ and the right vertices $\{y(1), \ldots, y(k)\}$. So that from Lemma A.2 we have $\operatorname{perm}\big(\mathcal{A}([1:k-1, i], 1:k)\big) = w(M \triangle \sigma)$. Finally using Lemma A.1 we have $w(M \triangle \sigma) = w(M) + w(\sigma)$ and using the expression for ℓ_{ik} for $i > k$ in (3.3), we have that,

$$\ell_{ik} = \operatorname{perm}\big(\mathcal{A}([1:k-1, i], 1:k)\big) - \operatorname{perm}\big(\mathcal{A}(1:k, 1:k)\big) = w(M) + w(\sigma) - w(M) = w(\sigma).$$

Similarly for the upper factor. Let M_{k-1} be a maximally weighted matching between the left vertices $\{x(1), \ldots, x(k-1)\}$ and the right vertices $\{y(1), \ldots, y(k-1)\}$ and let σ be a maximally weighted path through $R(M)$ from the unmatched left vertex $x(k)$ to the unmatched right vertex $y(j)$, for $j \geq k$. Then from Lemma A.2 we have

$$\operatorname{perm}\big(\mathcal{A}(1:k-1, 1:k-1)\big) = w(M).$$

From Theorem A.4 we have that $M \triangle \sigma$ is the maximally weighted matching between the left vertices $\{x(1), \ldots, x(k)\}$ and the right vertices $\{y(1), \ldots, y(k-1), y(j)\}$. So that from Lemma A.2 we have

$$\operatorname{perm}\big(\mathcal{A}(1:k, [1:k-1, j])\big) = w(M \triangle \sigma).$$

Finally using Lemma A.1 we have $w(M \triangle \sigma) = w(M) + w(\sigma)$ and using the expression for u_{kj} for $j \geq k$ in (3.3), we have that,

$$\begin{aligned} u_{kj} &= \operatorname{perm}\big(\mathcal{A}(1:k, [1:k-1, j])\big) - \operatorname{perm}\big(\mathcal{A}(1:k-1, 1:k-1)\big) \\ &= w(M) + w(\sigma) - w(M) = w(\sigma). \end{aligned}$$ □