Electronic Transactions on Numerical Analysis.
Volume 40, pp. 187-203, 2013.

ISSN 1068-9613.

ETNA
Kent State University
http://etna.math.kent.edu

VERIFIED STABILITY ANALYSIS USING THE LYAPUNOV MATRIX EQUATION*

ANDREAS FROMMER† AND BEHNAM HASHEMI‡§

Abstract. The Lyapunov matrix equation $AX + XA^* = C$ arises in many applications, particularly in the context of stability of matrices or solutions of ordinary differential equations. In this paper we present a method, based on interval arithmetic, which computes with mathematical rigor an interval matrix containing the exact solution of the Lyapunov equation. We work out two options which can be used to verify, again with mathematical certainty, that the exact solution of the equation is positive definite. This allows to prove stability of the (non-Hermitian) matrix A if we chose C as a negative definite Hermitian matrix. Our algorithm has computational cost comparable to that of a state-of-the art algorithm for computing a floating point approximation of the solution because we can cast almost all operations as matrix-matrix operations for which interval arithmetic can be implemented very efficiently.

Key words. stability analysis, Lyapunov matrix equation, interval arithmetic, Krawczyk's method, verified computation

AMS subject classifications. 65F05, 65G20

1. Introduction.

Let A and C be two given (real or) complex matrices of size $n \times n$. The equation

$$AX + XA^* = C, \tag{1.1}$$

which is linear in the entries of X, is called the Lyapunov matrix equation. It is well-known that this equation has a unique solution if and only if $\lambda_i + \overline{\lambda_j} \neq 0$ for all i and j where $\lambda_1, \lambda_2, \cdots, \lambda_n$ are the eigenvalues of A; see, e.g., [20]. The Lyapunov matrix equation is of interest in control and system theory especially in controllability and observability Gramians, balancing transformation, stability robustness to parameter variations, robust stability and performance study of large scale systems, reduced-order modeling and control filtering with singular measurement noise [13]. Of particular interest is the case in which the right-hand side matrix C is Hermitian [4, 16, 42] for which, if unique, the solution X is also Hermitian.

The matrix A is called stable (also negative stable or Hurwitz stable in the literature), if all its eigenvalues have negative real parts. Negative (positive) definite matrices are a special case of stable (positive stable) matrices [20]. An important tool for checking stability of a given matrix A is to solve the Lyapunov matrix equation (1.1) with C chosen to be a negative definite matrix and then check X for positive definiteness, because the following theorem holds.

THEOREM 1.1. *A matrix $A \in \mathbb{C}^{n\times n}$ is stable if and only if there exists a positive definite solution to the Lyapunov equation (1.1) where C is Hermitian negative definite.*

A proof of this basic theorem can be found in, e.g., [11, Thm 4.4] or [20, Thm 2.2.1]; see also [26, Thm 13.24]. Checking the stability of a matrix based on Theorem 1.1 has the advantage that the question of stability for an arbitrary matrix $A \in \mathbb{C}^{n\times n}$ is transferred to the simpler question of checking positive definiteness for a Hermitian matrix X [20].

To further highlight the importance of the concept of matrix stability consider a nonlinear system of n first-order differential equations

$$\dot{x} = f(x), \tag{1.2}$$

*Received September 7, 2012. Accepted for publication February 12, 2013. Published online July 10, 2013. Recommended by P. Van Dooren. This work was completed while the second author was visiting University of Wuppertal under support of grant A/12/06039 by the German Academic Exchange Service (DAAD).

†Department of Mathematics, University of Wuppertal, 42097 Wuppertal, Germany (frommer@math.uni-wuppertal.de).

‡Department of Mathematics, Faculty of Basic Sciences, Shiraz University of Technology, Modarres Boulevared, Shiraz 71555-313, Iran (hashemi@sutech.ac.ir, hoseynhashemi@gmail.com).

ETNA
Kent State University
http://etna.math.kent.edu

with $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}^n$. A vector $\hat{x}$ is called an *equilibrium point* of (1.2) if $f(\hat{x}) = 0$, i.e., the constant function $x(t) = \hat{x}$ is a solution. An equilibrium point $\hat{x}$ is called *asymptotically stable* if there is a neighborhood N of $\hat{x}$, such that for every solution of (1.2) for which $x(t) \in N$ for some t, we have $\lim_{t\to\infty} x(t) = \hat{x}$ [44, pp. 298-299]. Physically this means that a system whose state is perturbed slightly from an equilibrium point will return to that equilibrium point. The stability of each equilibrium point can be analyzed by linearizing (1.2) about that point. Specifically, for x near $\hat{x}$ the solutions of (1.2) are approximated well by solutions of

$$\dot{x} = \frac{\partial f}{\partial x}(\hat{x})(x - \hat{x}), \tag{1.3}$$

which is a linear system of differential equations with coefficient matrix $\frac{\partial f}{\partial x}$. It can be shown that all solutions of (1.3) satisfy $x - \hat{x} \to 0$ as $t \to \infty$ if the Jacobian $\frac{\partial f}{\partial x}$ at $\hat{x}$ is stable; see, e.g., [44, pp. 291-298]. To put it another way: $\hat{x}$ is an asymptotically stable equilibrium point of (1.2) if all the eigenvalues of the Jacobian at $\hat{x}$ have negative real parts. We refer to [20, Ch. 2] for more details.

1.1. Verified stability analysis. In this paper we consider the Lyapunov matrix equation (1.1) where C is Hermitian. Our goal is to present a *verified* numerical algorithm, i.e., an algorithm whose output will be exactly one of the two following statements, where the first one is correct with mathematical certainty:

1. (1.1) has a Hermitian positive definite solution. The algorithm then also provides correct (and tight) lower and upper bounds for each entry of the solution X.
2. Failure, i.e., we do not obtain any information on whether X is positive definite or not.

In the case that the algorithm outputs the first statement we have thus proved mathematically that A is stable. The major ingredient in our algorithm is its use of (machine) interval arithmetic to fully control rounding of floating point operations which—starting from a given approximate solution $\check{X}$ to the Lyapunov equation—allows to compute enclosing intervals for all entries of the $n \times n$ solution matrix X, i.e., a matrix $\boldsymbol{X}$ whose entries are compact intervals which have been proven to contain the corresponding entries of the exact solution X by the algorithm. The aim is to compute narrow intervals for all of these entries so that we have high chances of success for a subsequent, final step which proves that all Hermitian matrices contained in $\boldsymbol{X}$ are positive definite using an approach by Rump [40]. Our algorithm will also be computationally efficient in theory and practice: In many cases its total cost is of the same order as the cost for obtaining the approximate solution $\check{X}$, i.e., $\mathcal{O}(n^3)$. Moreover, the algorithm almost exclusively uses matrix-matrix operations, a crucial feature for time efficient machine interval arithmetic since it avoids most of the otherwise very costly switchings of rounding modes.

Verified stability analysis based on the Lyapunov matrix equation $AX + XA^* = -I$ has already been considered in [8, Ch. 4] and [9, 33]. The details on how the Lyapunov equation is solved are not available in [8, 9, 33], the reported numerical examples only include 2×2 matrices. Moreover, in these publications 2^{n-1} "corner" matrices had to be used to verify positive definiteness of interval matrices, employing a method by Alefeld [1].

Another approach to verify stability of a matrix based on interval arithmetic was pursued by Gross [15]; see also the comments in [22]. Here, the problem of verified stability analysis of A is firstly converted into the problem of verified Schur stability (all eigenvalues have modulus less than 1) of an interval matrix enclosing the Möbius transform $S(A) = I+2(A-I)^{-1}$. The so-called Cordes algorithm [7] is then applied to the enclosure of $S(A)$. The Cordes algorithm checks whether for a certain exponent $m \in \mathbb{N}$ the spectral radius of an interval matrix

ETNA
Kent State University
http://etna.math.kent.edu

containing the set $\{Y \,|\, Y = \tilde{A}^m, \tilde{A} \in \boldsymbol{S}(A)\}$ is less than 1, using Geršgorin's theorem, e.g., [43]. It is important to avoid the wrapping effect as much as possible which is why the Cordes algorithm considers only exponents m which are powers of two in a recursive manner and, in addition, uses Lohner's idea [27] of representing parallelepipeds in a factorized form. This approach has also been extended for an interval input matrix $\boldsymbol{A}$ by applying a partitioning process on non-degenerate interval elements of $\boldsymbol{A}$ at the risk of ending up with exponential complexity.

1.2. Interval arithmetic and Intlab. (Machine) interval arithmetic is the basic mathematical tool in verified numerical computing [17]. Detailed investigations of the mathematical properties of interval arithmetic can be found in, e.g., [2, 29, 30]. Here, we only review those basic properties needed in our methods. A real compact interval can be represented via its two endpoints or via its midpoint and radius. Generalizing this to the set of complex numbers yields two different concepts of complex intervals: rectangular intervals, characterized by a lower left and upper right corner in the complex plane, and circular intervals, characterized again by midpoint and radius. More precisely, a rectangular complex interval $\boldsymbol{a}$ is the closed rectangle $\boldsymbol{a} = \{x + iy : \ x \in \boldsymbol{x}, y \in \boldsymbol{y}\}$ with $\boldsymbol{x}$ and $\boldsymbol{y}$ real compact intervals, also denoted $\boldsymbol{a} = \boldsymbol{x} + i\boldsymbol{y}$, while a circular complex interval $\boldsymbol{a}$ is a closed circular disk of radius $\operatorname{rad}(\boldsymbol{a})$ and center $\operatorname{mid}(\boldsymbol{a})$, i.e., $\boldsymbol{a} = \langle \operatorname{mid}(\boldsymbol{a}), \operatorname{rad}(\boldsymbol{a}) \rangle = \{z \in \mathbb{C} : \ |z - \operatorname{mid}(\boldsymbol{a})| \leq \operatorname{rad}(\boldsymbol{a})\}$.

An arithmetic operation $\circ \in \{+, -, *, /\}$ between two intervals can, in principle, be defined in the set theoretic sense. For reasons of a practical implementation, however, one wants these operations to be closed in the given interval format, and the characterizing parameters should be easy to compute. So, for example, the standard arithmetic for circular complex interval arguments $\boldsymbol{a}, \boldsymbol{b}$ is defined as follows:

$$
\begin{aligned}
\langle \operatorname{mid}(\boldsymbol{a}), \operatorname{rad}(\boldsymbol{a}) \rangle \pm \langle \operatorname{mid}(\boldsymbol{b}), \operatorname{rad}(\boldsymbol{b}) \rangle &= \langle \operatorname{mid}(\boldsymbol{a}) \pm \operatorname{mid}(\boldsymbol{b}), \operatorname{rad}(\boldsymbol{a}) + \operatorname{rad}(\boldsymbol{b}) \rangle, \\
\langle \operatorname{mid}(\boldsymbol{a}), \operatorname{rad}(\boldsymbol{a}) \rangle * \langle \operatorname{mid}(\boldsymbol{b}), \operatorname{rad}(\boldsymbol{b}) \rangle &= \\
\langle \operatorname{mid}(\boldsymbol{a})\operatorname{mid}(\boldsymbol{b}), |\operatorname{mid}(\boldsymbol{a})|\operatorname{rad}(\boldsymbol{b}) + |\operatorname{mid}(\boldsymbol{b})|\operatorname{rad}(\boldsymbol{a}) + \operatorname{rad}(\boldsymbol{a})\operatorname{rad}(\boldsymbol{b}) \rangle, & \\
1/\langle \operatorname{mid}(\boldsymbol{a}), \operatorname{rad}(\boldsymbol{a}) \rangle &= \langle 1/\operatorname{mid}(\boldsymbol{a}), 1/(|\operatorname{mid}(\boldsymbol{a})| - \operatorname{rad}(\boldsymbol{a})) \rangle, \\
\langle \operatorname{mid}(\boldsymbol{a}), \operatorname{rad}(\boldsymbol{a}) \rangle / \langle \operatorname{mid}(\boldsymbol{b}), \operatorname{rad}(\boldsymbol{b}) \rangle &= (1/\langle \operatorname{mid}(\boldsymbol{b}), \operatorname{rad}(\boldsymbol{b}) \rangle) * \langle \operatorname{mid}(\boldsymbol{a}), \operatorname{rad}(\boldsymbol{a}) \rangle.
\end{aligned}
$$

Herein, the operations $+$ and $-$ coincide with the set theoretic definition; see, e.g., [2] for details. For all operations $\circ$ we have the fundamental enclosure relation

$$
\boldsymbol{a} \circ \boldsymbol{b} \supseteq \{a \circ b : a \in \boldsymbol{a}, b \in \boldsymbol{b}\}. \tag{1.4}
$$

The enclosure property (1.4) carries over to expressions: If $r(x_1, \ldots, x_n)$ is an arithmetic expression in the variables $x_1, \ldots, x_n$, then its interval arithmetic evaluation $r(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, an interval, contains the range of r for $x_1 \in \boldsymbol{x}_1, \ldots, x_n \in \boldsymbol{x}_n$.

When interval arithmetic is implemented on a computer, the parameters defining the result interval are computed in floating point arithmetic from the parameters defining the interval operands. For the enclosure property to hold for such a *machine interval arithmetic* it is mandatory to use directed roundings appropriately; see, e.g., [31]. Intlab [39] is an open source Matlab toolbox that provides such a reliable machine interval arithmetic. It is freely available for non-commercial use from `http://www.ti3.tu-harburg.de/~rump/intlab/`. A crucial ingredient to the efficiency of Intlab is the fact that it allows to use implementations of (interval) matrix-matrix and matrix-vector operations in the midpoint-radius format in a way that the number of switches of the rounding mode is independent of the dimension of the matrices/vectors. On today's computer architectures this allows much (up to 1000 times) faster execution times than an interval code in which rounding modes

ETNA
Kent State University
http://etna.math.kent.edu

would be switched anew for each operation with scalar operands [38]. A similar approach is also available for the C++ package C-XSC; see [19, 23].

We used Intlab[1] [39] to implement the algorithms for verifying stability of a matrix developed in the present paper. In order to make these implementations efficient, we seek to formulate as much of the computational work as possible in terms of matrix-matrix operations.

We end this introduction by explaining some of our notation. For a complex $n \times n$ matrix N, the matrix $M := N^T$ represents the transpose of N. The notation N^* for the Hermitian transpose $N^* = \overline{N}^T$ was already used earlier; see (1.1).

For two matrices $A \in \mathbb{C}^{m\times m}$ and $B \in \mathbb{C}^{n\times n}$, $A \otimes B$ (see, e.g., [20]) denotes the Kronecker product of A and B, so $A \otimes B$ is a matrix of size $mn \times mn$. By vec we denote the operation of stacking the columns of a matrix in order to obtain one long vector. So $\mathrm{vec}(A)$ is a vector of length m^2. For $d = [d_1, \ldots, d_n]^T \in \mathbb{C}^n$, the matrix $\mathrm{Diag}\,(d)$ denotes the diagonal matrix in $\mathbb{C}^{n\times n}$ whose i-th diagonal entry is d_i. We extend this to matrices: For $D \in \mathbb{C}^{n\times m}$ we put $\mathrm{Diag}\,(D) = \mathrm{Diag}\,(\mathrm{vec}(D)) \in \mathbb{C}^{nm\times nm}$. By $./$ we mean the Hadamard (pointwise) division.

The following lemma will turn out to be useful. For part a), see, e.g., [20]; part b) is trivial.

LEMMA 1.2. *For any three (real or) complex matrices A, B, and C with compatible sizes we have*

a) $\mathrm{vec}(ABC) = (C^T \otimes A)\mathrm{vec}(B)$.

b) $\mathrm{Diag}\,(A)^{-1}\mathrm{vec}(B) = \mathrm{vec}(B./A)$.

The remaining part of this paper is organized as follows. In Section 2, we give the details of our algorithms for verified stability analysis. Section 3 contains the results of a series of numerical experiments, and some conclusions are given in Section 4.

2. Verified solution of the Lyapunov equation via (block) diagonalization. In this section we describe in detail our verification algorithm for computing enclosures for the solution of a Lyapunov equation along with the test for positive definiteness. The approach relies on a Krawzcyk-type method which we present first.

Krawczyk's method is a classical method for computing an enclosure for the solution of a general, unstructured, non-singular linear system

$$\mathcal{P}x = c, \quad \mathcal{P} \in \mathbb{C}^{m\times m}, \quad x, c \in \mathbb{C}^m. \tag{2.1}$$

Given an approximate solution $\check{x}$ of the linear system, computed by some floating point linear system solver, and given an approximate inverse $\mathcal{R}$ of $\mathcal{P}$, again computed by some floating point algorithm, Krawczyk's method [24] in its improved version by Rump [37] uses machine interval arithmetic (including outward rounding) to check whether

$$\boldsymbol{k} := \mathcal{R}(c - \mathcal{P}\check{x}) + (I_m - \mathcal{R}\mathcal{P})\boldsymbol{z} \subseteq \mathrm{int}\,(\boldsymbol{z}). \tag{2.2}$$

Here, I_m denotes the identity in $\mathbb{C}^m$ and $\mathrm{int}\,(\boldsymbol{z})$ denotes the topological interior of the (closed) interval vector $\boldsymbol{z}$. If (2.2) holds we know that the solution of (2.1) is contained in $\check{x} + \boldsymbol{z}$ due to the following result from [37]; see also [12].

THEOREM 2.1. *[12, 37] Let $\boldsymbol{z}$ be an interval vector. If*

$$\mathcal{K} := \{\mathcal{R}(c - \mathcal{P}\check{x}) + (I_m - \mathcal{R}\mathcal{P})z : \ z \in \boldsymbol{z}\} \subseteq \mathrm{int}\,(\boldsymbol{z}), \tag{2.3}$$

[1] More precisely, we used Intlab V6. The latest (end 2012) release Intlab V7 represents a major update where several algorithms have been modified.

ETNA
Kent State University
http://etna.math.kent.edu

then $\mathcal{P}$ and $\mathcal{R}$ are non-singular, and the solution of (2.1) is contained in $\check{x} + \mathcal{K} \subseteq \check{x} + \boldsymbol{z}$.

Note that $\mathcal{K} \subseteq \boldsymbol{k}$ due to the enclosure property of interval arithmetic. The enclosure method we will present for the Lyapunov equations also relies on Theorem 2.1, but it uses interval arithmetic in a different manner than in (2.2) to compute an enclosure for $\mathcal{K}$.

The Lyapunov equation (1.1) can be written using the vec and $\otimes$ notations in the form of the following system of linear equations [20, 45]

$$\mathcal{P}x = c, \text{ where } \mathcal{P} = I_n \otimes A + \overline{A} \otimes I_n,\ x = \mathrm{vec}(X) \text{ and } c = \mathrm{vec}(C). \tag{2.4}$$

Note that by Lemma 1.2 we have $\mathrm{vec}(XA^*) = (\overline{A} \otimes I_n)\mathrm{vec}(X)$.

In order to apply Krawczyk's method to (2.4), we need to compute an approximate inverse $\mathcal{R}$,

$$\mathcal{R} \approx \mathcal{P}^{-1} = (I_n \otimes A + \overline{A} \otimes I_n)^{-1} \in \mathbb{C}^{n^2 \times n^2},$$

which is an $\mathcal{O}(n^6)$ process if we use Gaussian elimination and do not try to take advantage of the sparsity structure of $\mathcal{P}$. But even if $\mathcal{R}$ were computed more efficiently, we still need to compute $I - \mathcal{R}\mathcal{P}$. If A is a dense (non-sparse) matrix we must expect $\mathcal{R}$ to be a dense matrix, too. Since each column of $\mathcal{P}$ then contains (at most) $2n - 1$ nonzero entries and $\mathcal{R}$ is an $n^2 \times n^2$ matrix, computing $\mathcal{R}\mathcal{P}$ has a complexity $\mathcal{O}(n^5)$. Therefore, the cost for computing $\boldsymbol{k}$ in (2.2) is at least $\mathcal{O}(n^5)$ which is prohibitively high for larger values of n.

The Lyapunov equation is a special case of the Sylvester matrix equation $AX + XB = C$ with $B = A^*$. We have shown in [12] how the complexity of an enclosure method for the solution of the Sylvester equation can be reduced to $\mathcal{O}(n^3)$ if A and B are diagonalizable. While [12] expicitly relies on interval arithmetic, the recent paper [28] extends the approach of [12] in a way that it is sufficient to check certain inequalities using (IEEE standard compliant) floating point operations. We now recall the approach of [12] specialized to the case of the Lyapunov equation. To expose the central idea on how to reduce the complexity to $\mathcal{O}(n^3)$, we first discuss the idealized situation where all arithmetic operations are done exactly. The modification to be applied in practice using machine interval arithmetic to completely control all roundings will be discussed thereafter.

If A is diagonalizable we have the (exact) spectral decomposition

$$VA = DV \quad \text{with} \quad V, D \in \mathbb{C}^{n \times n}, D = \mathrm{diag}(\lambda_1, \ldots, \lambda_n). \tag{2.5}$$

Here, V is a matrix of left eigenvectors of A. So X is an exact solution of the Lyapunov equation (1.1) if and only if

$$(VAV^{-1})(VXV^*) + (VXV^*)(VAV^{-1})^* = VCV^*.$$

Therefore, $Y = VXV^*$ is the solution of the linearly transformed Lyapunov equation

$$DY + YD^* = G, \tag{2.6}$$

where $G = VCV^*$. The Lyapunov equation (2.6) is equivalent to the linear system of equations

$$\mathcal{Q}\, y = g, \tag{2.7}$$

where

$$\begin{aligned} \mathcal{Q} &= I_n \otimes (VAV^{-1}) + \overline{(VAV^{-1})} \otimes I_n, \\ y &= (\overline{V} \otimes V)x, \\ g &= (\overline{V} \otimes V)c. \end{aligned} \tag{2.8}$$

ETNA
Kent State University
http://etna.math.kent.edu

Since we temporarily assume that V is a matrix of exact eigenvectors, we have, of course, that $VAV^{-1} = D$, which shows that $\mathcal{Q}$ is diagonal. This means that an approximate inverse $\mathcal{R}$ for $\mathcal{Q}$ can be computed very cheaply at cost $\mathcal{O}(n^2)$, and the same holds for the product $\mathcal{R}\mathcal{Q}$. This shows that diagonalization bears the potential of substantially reducing the computational cost in a Krawczyk-type enclosure method.

Let us now turn to the realistic setting, where instead of exact arithmetic we use floating point and machine interval arithmetic. We assume that V and D in the diagonalization of A are computed via a floating point algorithm. Then (2.5) will hold only approximately, but we still have that the solution Y of the transformed Lyapunov equation (2.6) is related to the solution X of the original Lyapunov equation (1.1) via $Y = VXV^*$ if in (2.6) we replace D with the matrix VAV^{-1}, which is now only approximately diagonal. But since VAV^{-1} is close to D, we also have that the diagonal matrix

$$\Delta = I \otimes D + \bar{D} \otimes I, \tag{2.9}$$

is close to the matrix $\mathcal{Q}$ from (2.8) (which as VAV^{-1} is not exactly diagonal any more). This shows that the inverse of the diagonal matrix Δ can be taken as the approximate inverse $\mathcal{R}$ in a Krawczyk-type enclosure method. This is crucial to the efficiency of a Krawczyk-type method, since because Δ is diagonal the computation of $\Delta^{-1}\mathcal{Q}$ has complexity $\mathcal{O}(n^3)$ only, given that $\mathcal{Q}$ has (at most) $2n-1$ non-zero entries in each of its n^2 rows.

While the matrices V and D are available as the result of a floating point algorithm, the matrix V^{-1} is not. Working just with an approximation for V^{-1}, obtained by some floating point algorithm, is not sufficient for our purposes, because then the relation $Y = VXV^*$ between the original solution X and the solution Y of the transformed system (2.7) would hold only approximately. We therefore work with an enclosure $\boldsymbol{I}_V$ for V^{-1}, i.e., with an interval matrix $\boldsymbol{I}_V$ known to contain the exact inverse V^{-1}. Such an interval matrix $\boldsymbol{I}_V$ can be obtained using Krawczyk's method for the function $VX - I$. An implementation is available through the Intlab function `verifylss`.

Let now $\check{X}$ be an approximate solution for the Lyapunov equation (1.1), obtained by some floating point algorithm and let $S := A\check{X} + \check{X}A^* - C$ be its residual. Then the error $T := X - \check{X}$ with respect to the exact solution solves

$$AT + TA^* = -S.$$

The idea is now to use the transformations described so far to obtain an efficient Krawczyk-type method to compute an enclosure for T.

Let $\boldsymbol{B}$ and $\boldsymbol{F}$ be interval matrices that contain VAV^{-1} and $V(A\check{X} + \check{X}A^* - C)V^*$, respectively. If we can compute an interval matrix $\boldsymbol{E}$ that contains all solutions of all equations

$$BE + EB^* = -F, \tag{2.10}$$

for every $B \in \boldsymbol{B}$ and every $F \in \boldsymbol{F}$, then $\check{X} + \boldsymbol{I}_V \boldsymbol{E} \boldsymbol{I}_V^*$ will contain the exact solution $X = \check{X} + T$ of the original Lyapunov equation (1.1). To obtain such $\boldsymbol{E}$, we wish to apply a Krawczyk-type method based on Theorem 2.1. Converting matrices into vectors using the vec operator, this means that we have to compute an enclosure for the set

$$\mathcal{K}(\boldsymbol{e}) := \{\Delta^{-1}(-f + (\Delta - (I_n \otimes B + \overline{B} \otimes I_n))e,\ f \in \boldsymbol{f},\ B \in \boldsymbol{B},\ e \in \boldsymbol{e}\}, \tag{2.11}$$

where $\boldsymbol{f} = \mathrm{vec}(\boldsymbol{F})$, $\boldsymbol{e} = \mathrm{vec}(\boldsymbol{E})$. Using machine interval arithmetic to evaluate the expression defining F, an enclosure for $\{-f + (\Delta - (I_n \otimes B + \overline{B} \otimes I_n))e, f \in \boldsymbol{f},\ B \in \boldsymbol{B}, e \in \boldsymbol{e}\}$ can be computed in matrix terms as $\mathrm{vec}(\boldsymbol{G})$ where

$$\boldsymbol{G} := -\boldsymbol{F} + (D - \boldsymbol{B})\boldsymbol{E} + \boldsymbol{E}(D - \boldsymbol{B})^*.$$

ETNA
Kent State University
http://etna.math.kent.edu

Also, in matrix instead of vector terms, the multiplication with the diagonal matrix Δ^{-1} is a pointwise division by the matrix $d \cdot l^* + l \cdot d^*$, where $d \in \mathbb{C}^{n\times 1}$ is a vector containing the diagonal entries of D and $l^* = [1, \cdots, 1] \in \mathbb{R}^{1\times n}$. Note that the (i,j)-entry $d_i + \bar{d}_j$ of this matrix cannot be computed exactly in floating point arithmetic, but machine interval arithmetic yields a matrix $\boldsymbol{L} \supseteq d \cdot l^* + l \cdot d^*$. Using Theorem 2.1 and writing everything in matrix terms we have that if

$$\boldsymbol{K}(\boldsymbol{E}) = (-\boldsymbol{F} + (D - \boldsymbol{B})\boldsymbol{E} + \boldsymbol{E}(D - \boldsymbol{B})^*)./\boldsymbol{L} \subseteq \operatorname{int}(\boldsymbol{E}),$$

then $\boldsymbol{K} := \boldsymbol{K}(\boldsymbol{E})$ contains all solutions of (2.10), and the exact solution X of the original Lyapunov equation (1.1) is contained in $\check{X} + \boldsymbol{I}_V \boldsymbol{K} \boldsymbol{I}_V^*$.

In Algorithm 1, we explicitly "symmetrize" a computed interval matrix $\boldsymbol{A}$ when we know that the exact, non-interval matrices of interest contained in $\boldsymbol{A}$ are Hermitian. In principle, symmetrization would mean that we replace an entry $\boldsymbol{a}_{ij}$ of $\boldsymbol{A}$ by $\boldsymbol{a}_{ij} \cap \bar{\boldsymbol{a}_{ji}}$, showing that this operation works towards making the entries of $\boldsymbol{A}$ more narrow. In midpoint-radius format, the intersection of two intervals is not a disk any more. Intlab therefore has to use a relatively sophisticated algorithm to obtain a disk containing this intersection. Interestingly, the Intlab `intersect` operator does *not* fulfill a commutativity relation of the kind $\texttt{intersect}(\boldsymbol{a}, \boldsymbol{b}) = \texttt{intersect}(\bar{\boldsymbol{a}}, \bar{\boldsymbol{b}})$. As a result, an interval matrix computed as the entrywise intersection of $\boldsymbol{A}$ and $\boldsymbol{A}^*$ is not necessarily "Hermitian". Instead of $\texttt{intersect}(A, A^*)$ we therefore use a symmetrization operator H implemented via the following Matlab-Intlab commands:

```
H=intersect(A, A*);
H(A) := tril(H,-1)+diag(real(diag(H)))+tril(H,-1)*;.
```

Here, `tril(H,-1)` is the Matlab command which returns the strictly lower triangular part of `H`. Note that $H(A)$ contains indeed all Hermitian (point) matrices $A \in \boldsymbol{A}$.

The interval hull operator $\square(0, \boldsymbol{U})$ used in Step 10 of Algorithm 1, produces an interval matrix each entry of which is the smallest compact interval containing 0 and the respective entry of the interval matrix $\boldsymbol{U}$.

2.1. Checking positive definiteness of interval matrices. Determining the positive definiteness of symmetric interval matrices plays an important role in several applications ranging from stability analysis of matrices, global optimization problems and solution of linear interval equations over semi-definite programming problems, to the representation theory of Lie groups [10, 41]. Rohn [35] showed that the problem of determining the positive definiteness of a real symmetric interval matrix is NP-hard. Shao and Hou [41] proved that an $n \times n$ Hermitian interval matrix $\boldsymbol{A}$ is positive definite if and only if $4^{n-1}(n-1)!$ specially chosen Hermitian vertex matrices are positive definite; see also [18, 36]. Rump [40] presented a computationally simple and fast sufficient criterion implying positive definiteness of a symmetric or Hermitian interval matrix. His method is based on a single floating-point Cholesky decomposition of the midpoint matrix, its backward stability analysis and a perturbation result. More recently, Domes and Neumaier [10] proposed a so-called directed Cholesky factorization that can also be used for verifying positive definiteness.

In Step 17 of Algorithm 1 we need to check the positive definiteness of the exact solution X of the Lyapunov equation (1.1). We did so using the Intlab function `isspd` which is based on [40]. We also tested the alternative approach to prove positive definiteness from [10], where the code was kindly made available to us by the authors and observed very similar results. We thus stayed with `isspd` from Intlab in the present paper.

Since the exact solution X is contained in $\boldsymbol{X} = \check{X} + \boldsymbol{I}_V \boldsymbol{K} \boldsymbol{I}_V^*$, our first option to be implemented in Step 17 is as follows

Option 1: Apply `isspd` to $H(\boldsymbol{X})$, i.e., the Hermitian part of $\boldsymbol{X}$.

ETNA
Kent State University
http://etna.math.kent.edu

Algorithm 1 If successful this algorithm obtains an interval matrix $\boldsymbol{W}$ such that $\check{X} + \boldsymbol{W}$ contains the unique Hermitian solution X of the Lyapunov equation (1.1) and checks the positive definiteness of X.

1: Use a floating point algorithm to get an approximate solution $\check{X}$ of the Lyapunov equation (1.1). Then, replace $\check{X}$ by its Hermitian part $H(\check{X})$.
2: Compute V and D in the spectral decomposition (2.5) using a floating point algorithm.
3: Compute an $n \times n$ interval matrix $\boldsymbol{L}$ s.t. $\boldsymbol{L} \supseteq d \cdot l^* + l \cdot d^*$. {$\boldsymbol{L}$ is an interval matrix due to outward rounding}
4: Let $\boldsymbol{L} = H(\boldsymbol{L})$. {$\mathrm{vec}(\boldsymbol{L}) \supseteq \mathrm{diag}(\Delta)$ from (2.9)}
5: Compute an interval matrix $\boldsymbol{I}_V$ containing V^{-1}.
6: Compute $\boldsymbol{F} = V \cdot (A\check{X} + (A\check{X})^* - C) \cdot V^*$ using interval arithmetic and let $\boldsymbol{F} = H(\boldsymbol{F})$.
7: Compute $\boldsymbol{B} = (VA)\boldsymbol{I}_V$ using interval arithmetic everywhere.
8: Compute $\boldsymbol{E} = -\boldsymbol{F}./\boldsymbol{L}$, put $j = 1$. {Prepare loop}
9: **repeat**
10: Put $\boldsymbol{E} = \square(0, \boldsymbol{E} \cdot [1-\varepsilon, 1+\varepsilon])$, increment j. {ε-inflation}
11: Compute $\boldsymbol{N} = (D - \boldsymbol{B})\boldsymbol{E}$.
12: Compute $\boldsymbol{M} = -\boldsymbol{F} + (\boldsymbol{N} + \boldsymbol{N}^*)$. {$\boldsymbol{M}$ is Hermitian}
13: Compute $\boldsymbol{K} = \boldsymbol{M}./\boldsymbol{L}$. {$\boldsymbol{K}$ is Hermitian}
14: **until** ($\boldsymbol{K} \subseteq \mathrm{int}\,\boldsymbol{E}$ or $j = j_{max}$)
15: **if** $\boldsymbol{K} \subseteq \mathrm{int}\,\boldsymbol{E}$ **then** {successful termination}
16: Compute $\boldsymbol{W} = (\boldsymbol{I}_V \boldsymbol{K})\boldsymbol{I}_V^*$. {solution X of (1.1) is in $H(\check{X} + \boldsymbol{W})$}
17: Check for positive definiteness of $H(\check{X} + \boldsymbol{W})$. {2 options available}
18: **else**
19: Output "verification not successful".
20: **end if**

On the other hand, we know that

$$X = \check{X} + V^{-1}EV^{-*} = V^{-1}(\check{Y} + E)V^{-*}, \text{ with } \check{Y} = V\check{X}V^* \text{ and } V^{-*} = (V^{-1})^*.$$

So, the matrix X is positive definite if and only if $Y = \check{Y} + E$ is positive definite. We know that Y is contained in $\check{Y} + \boldsymbol{K}$ where $\check{Y} = (V\check{X})V^*$. Therefore, an alternative approach is to show that every Hermitian matrix contained in an interval matrix $\boldsymbol{Y} \supseteq \check{Y} + \boldsymbol{K}$ is positive definite. Our second option is therefore as follows

Option 2: Apply `isspd` to $H(\boldsymbol{Y})$, where $\boldsymbol{Y}$ is given as $(V\check{X})V^* + \boldsymbol{K}$ with $(V\check{X})V^*$ computed using machine interval arithmetic to account for floating point roundings in the evaluation of the products.

Option 2 is likely to be superior to option 1 since the interval entries of $\boldsymbol{Y}$ can be expected to be narrower than those of $\boldsymbol{X}$ and $\boldsymbol{Y}$ is often (slightly) better conditioned than $\boldsymbol{X}$.

2.2. Block diagonalization. The approach presented may fail at two places: The variant of Krawczyk's method may fail to verify the crucial condition $\boldsymbol{K} \subseteq \mathrm{int}\,\boldsymbol{E}$ in Step 14 of Algorithm 1, or the test for positive definiteness may fail in Step 17. The chances for failures increase as the condition number of the left eigenvector matrix V increases. Due to the wrapping effect, the radii of the entries of the various interval matrices to be computed in Algorithm 1 will then tend to become very large so that the condition $\boldsymbol{K} \subseteq \mathrm{int}\,\boldsymbol{E}$ in Step 14 will not hold any more.

We therefore also present a variant of our algorithm where we use the *block diagonalization* of Bavely and Stewart [3] to control the condition number of V at the expense of having

ETNA
Kent State University
http://etna.math.kent.edu

D with (hopefully small) blocks along the diagonal. The block diagonal factorization can be written as

$$A = V^{-1}DV, \tag{2.12}$$

where D is *block* diagonal with each diagonal block being triangular.

We now formally define $\mathcal{Q}$ exactly as in (2.8) and its approximation Δ as in (2.9), i.e., $\Delta = I_n \otimes D + \overline{D} \otimes I_n$. Of course, Δ is not diagonal any more. It is block diagonal with upper triangular diagonal blocks. We refer to [12] for further details.

The following modifications need to be applied to Algorithm 1. In Step 2, the matrices V and D will be obtained from the block diagonal factorization (2.12). Since the matrix D is not diagonal any more we have to modify Steps 8 and 13 where we used pointwise division by the interval matrix $\boldsymbol{L}$. For the block modification, Steps 3 and 4 will not be needed and instead of Step 8 we use forward substitution to solve $\Delta\mathrm{vec}(\boldsymbol{E}) = -\mathrm{vec}(\boldsymbol{F})$ for $\boldsymbol{E}$ and let $\boldsymbol{E} = H(\boldsymbol{E})$. Similarly, instead of Step 13, we solve $\Delta\mathrm{vec}(\boldsymbol{K}) = \mathrm{vec}(\boldsymbol{M})$ for $\boldsymbol{K}$ using forward substitution and let $\boldsymbol{K} = H(\boldsymbol{K})$. It must be noted that these two modifications destroy the matrix-matrix operation paradigm, so that the modified algorithm will be substantially more time consuming in Steps 8 and 13 as before. Note that this modified algorithm will have complexity $\mathcal{O}(n^3)$ only if the sizes of the blocks in D are bounded by a constant.

We refer to [12] for a discussion of why a standard Schur decomposition of A, i.e., an orthogonal reduction to (full) triangular form, is not a viable approach for an enclosure method based on machine interval arithmetic due to the exponentially growing accumulation of outward roundings.

3. Numerical results. All our numerical examples focus on proving the stability of a given matrix A. So, in view of Theorem 1.1, we apply Algorithm 1 to compute an enclosing interval matrix for the solution of the Lyapunov matrix equation

$$AX + XA^* = -I, \tag{3.1}$$

and then run `isspd` on the thus computed enclosure for X (option 1) or $Y = VXV^*$ (option 2).

In our tables, we wish to report indicators on the quality of the computed enclosure matrices. These indicators will be based on the *relative precision* of an interval $\boldsymbol{a}$ given as

$$\mathrm{rp}(\boldsymbol{a}) := \min(\texttt{relerr}(\boldsymbol{a}), 1),$$

where `relerr` is defined as

$$\texttt{relerr}(\boldsymbol{a}) = \begin{cases} \frac{\mathrm{rad}\,(\boldsymbol{a})}{|\mathrm{mid}\,(\boldsymbol{a})|}, & 0 \notin \boldsymbol{a}, \\ \mathrm{rad}\,(\boldsymbol{a}), & 0 \in \boldsymbol{a}, \end{cases}$$

available as an Intlab function. Loosely speaking, $-\log_{10}(\mathrm{rp}(\boldsymbol{a}))$ can be regarded as the number of known correct digits of an "exact" value contained in $\boldsymbol{a}$. For an interval matrix $\boldsymbol{X}$ we will report two indicators mrp and arp based on the relative precision $\mathrm{rp}(\boldsymbol{X}_{ij})$ of its entries $\boldsymbol{X}_{ij}$, defined as

$$\mathrm{mrp}(\boldsymbol{X}) := \max\{\mathrm{rp}(\boldsymbol{X}_{ij}) \mid i,j = 1,\ldots,n\}, \tag{3.2}$$

$$\mathrm{arp}(\boldsymbol{X}) := \left(\prod_{i,j=1,n} (\mathrm{rp}(\boldsymbol{X}_{ij}))\right)^{1/n^2}. \tag{3.3}$$

ETNA
Kent State University
http://etna.math.kent.edu

So $-\log_{10}(\mathrm{mrp}(\boldsymbol{X}))$ and $-\log_{10}(\mathrm{arp}(\boldsymbol{X}))$ represent the minimum and average number of known correct digits, respectively.

The following notation is used in all our tables: Under the headline "problem info" we summarize basic information about the Lyapunov equation considered. The first column refers to the name under which the test matrix is known from the literature or to our choice of parameters if A is from a parametrized family of test matrices. The second column gives important basic information about A, namely, the ℓ_2-condition number of its (approximate) eigenvector matrix V from (2.5). For smaller problem sizes we also report $\kappa(\mathcal{P})$, the ℓ_2-condition number of the matrix $\mathcal{P}$ from (2.4) and the separation "sep" of A and $-A^*$, i.e., the smallest singular value of $\mathcal{P}$. As was suggested by one of the referees, when A is the Jacobian of a system of ODEs, the quotient of the largest and smallest modulus of the eigenvalues of A is a customary measure for the stiffness of the system of ODEs; see, e.g., [21, pp. 56]. We therefore also report this *stiffness ratio* r_s for all our examples.

The third column contains information about the computation of the floating point approximation $\check{X}$. This matrix is obtained using the command `lyap` from the Matlab Control System Toolbox. We report the time t_{fl} required by `lyap` in seconds as well as the quality of the approximate solution $\check{X}$ as given by the Frobenius norm of the residual matrix $A\check{X} + \check{X}A^* + I$.

We report numerical results for both options available for Algorithm 1. Here, $\boldsymbol{X}$ and $\boldsymbol{Y}$ stand for the (symmetrized) interval matrices computed by Algorithm 1 with option 1 and 2, respectively. The columns named $\mathrm{spd}(\boldsymbol{X})$ and $\mathrm{spd}(\boldsymbol{Y})$ show the result of Intlab's `isspd` function for checking positive definiteness. Here, 1 means that every Hermitian matrix $X \in \boldsymbol{X}$ has been verified by `isspd` to be positive definite, while 0 means that no verified result could be obtained by `isspd`, and similarly for $\boldsymbol{Y}$. In a second column we give information on the quality of the computed enclosing matrix by reporting the values of the indicators mrp and arp defined in (3.2) and (3.3). The value k is the number of sweeps through the repeat loop of Algorithm 1. We set $j_{max} = 9$. If the algorithm is successful in computing an enclosure, it very often is so in the first sweep, $k = 1$. In the fourth column, "time" stands for the total time required by Algorithm 1 followed by one call to `isspd`, and $\kappa(\mathrm{mid}\,(\boldsymbol{X}))$ and $\kappa(\mathrm{mid}\,(\boldsymbol{Y}))$ stand for the condition number of the midpoint of the computed enclosures as a hint on the condition of the respective exact solution.

It remains to explain the meaning of the parameter "res. prec.". The success of a Krawczyk type method crucially depends on the precision we obtain when evaluating the residual, i.e., the radii of the components of the computed interval matrix $\boldsymbol{F} \ni V \cdot (A\check{X} + \check{X}A^* - C) \cdot V^*$. Note that $\boldsymbol{F}$ is an interval matrix since we have to account for all floating point roundings when evaluating the residual. In our tables, "res. prec." refers to the effort we invested into getting "tighter" enclosures for the residual. More precisely, "res. prec. = double" in the fourth column corresponding to option 1 means that no special effort is made, i.e., $\boldsymbol{F}$ is obtained by performing all matrix operations using the standard interval arithmetic operations (based on rounded IEEE double precision operations), starting with the point matrices $V, A, \check{X}$ and C. Similarly, for option 2 "res. prec. = double" means that we compute an enclosure for $\tilde{\boldsymbol{Y}} = (V\check{X})V^*$ in $\boldsymbol{Y}$ using standard interval arithmetic with the point matrices V and $\check{X}$. If the algorithm is not successful with these standard choices, we use an improved method from [32] for computing an interval enclosure $\boldsymbol{B}$ for the product $A\check{X}$. We then use standard double precision to evaluate the sum $\boldsymbol{B} + \boldsymbol{B}^* - C$ and the subsequent multiplications with V and V^*. This option is indicated as "res. prec. = impr." in the fourth column corresponding to option 1, and similarly in option 2 where it means that we use this improved multiplication just for the factor $V\check{X}$ when computing $\tilde{\boldsymbol{Y}} = V\check{X}V^*$.

If the algorithm is still not successful, we turn to "res. prec. = quad.", where now the ma-

ETNA
Kent State University
http://etna.math.kent.edu

trix product $A\check{X}$ is evaluated using simulated quadruple precision based on error-free transformations. This option is available as a choice in Intlab, but it should be noted that its computation speed is orders of magnitudes slower, since now rounding modes are switched for each scalar operation. For this option, as a side effect, we actually also obtain a "tighter" enclosure $\boldsymbol{I}_V$ for the inverse of the matrix V (see line 5 of Algorithm 1). We always obtain $\boldsymbol{V}$ via the Intlab function `verifylss`. This function computes an interval enclosure for a linear system and allows for block right hand sides (rhs), so that an enclosure for the inverse is obtained when choosing rhs as the identity matrix. When Intlab is required to work with simulated quadruple precision, `verifylss` for a block rhs actually does not use (simulated) quadruple precision everywhere, but rather uses just an improved precision implementation of the function `lssresidual` which is called from `verifylss`.

Our first set of tests is from Example 4.1 of the CTLEX benchmark [25]. This is an academic example which has the advantage that we can chose several parameters to control the conditioning of $\mathcal{P}$ as well as the dimension n, thus enabling, in particular, a scaling study. We report our numerical results in Tables 3.1 and 3.2. n, r, s are parameters in Example 4.1 from the CTLEX benchmark [25]. For the smaller examples in Table 3.1 we also report results obtained with the function `VERMATREQN` of the software package Versoft [34] for comparison. Versoft is a collection of Intlab programs, computing enclosures for the solutions of various numerical problems. For the Lyapunov equation, `VERMATREQN` basically uses Krawczyk's method for the large system (2.4). It has the advantage of not having to diagonalize or block diagonalize A at the disadvantage of a computational complexity beyond $\mathcal{O}(n^3)$.

TABLE 3.1
Numerical results for different tests from Example 4.1 of the CTLEX benchmark [25].

problem info.			Versoft [34]			Alg. 1 with option 1				Alg. 1 with option 2		
n	$\kappa(V)$	t_{fl}	spd	mrp($\boldsymbol{X}$)	time	spd	mrp($\boldsymbol{X}$)	k	time	spd	mrp($\boldsymbol{Y}$)	time
r	$\kappa(\mathcal{P})$	r_s	($\boldsymbol{X}$)	arp($\boldsymbol{X}$)		($\boldsymbol{X}$)	arp($\boldsymbol{X}$)		res. prec.	($\boldsymbol{Y}$)	arp($\boldsymbol{Y}$)	prec. ($\tilde{\boldsymbol{Y}}$)
s	sep	‖res‖							$\kappa(\text{mid}\,\boldsymbol{X})$			$\kappa(\text{mid}\,\boldsymbol{Y})$
10	3.1E3	1.9E-3	0	9.2E-5	3.1E-2	0	3.4E-4	1	1.5E-2	1	7.6E-4	2.4E-2
3.1	3.4E12	2.6E4		9.1E-5			2.4E-4		double		1.1E-4	double
2.5	6.5E-6	1.4E-3							3.9E10			1.2E5
10		1.5E-3			3.3E-2	1	8.7E-11	1	4.5E-2	1	4.7E-7	3.7E-2
3.1							6.1E-11		impr.		8.8E-9	double
2.5												
50	1.2E2	1.1E-2	0	8.6E-1	3.2E2	0	1.2E-2	8	2.7E-1	1	4.1E-2	2.9E-1
1.8	3.2E15	3.2E12		1.5E-2			2.5E-5		quad.		2.6E-6	double
1.1	1.6E-2	9.5E-2							2.4E15			1.1E13

The following observations can be made from Tables 3.1 and 3.2: Option 2 in Algorithm 1 allows to prove stability more often than option 1. This can be attributed to the fact that the condition number of the matrix $\text{mid}\,(\boldsymbol{Y})$ is often significantly smaller than that of $\text{mid}\,(\boldsymbol{X})$ and that, at least on the average, the relative width of the interval entries of $\boldsymbol{Y}$ from option 2 is smaller than for $\boldsymbol{X}$ from option 1. The right part of Figure 3.1 depicts this fact graphically.

When the condition number of $\mathcal{P}$ is very high (beyond inverse machine precision), standard double precision arithmetic is not sufficient. However, switching to improved or quadruple precision often helps. For $n = 50$, the Versoft function `VERMATREQN` already needs almost a factor of 1000 more time than Algorithm 1, and `isspd` is never successful on the results computed with Versoft. It should be noted, however, that we could not adapt the precision in Versoft as we did in Algorithm 1 to be successful for the problems considered in

ETNA
Kent State University
http://etna.math.kent.edu

TABLE 3.2
Numerical results for larger tests from Example 4.1 of the CTLEX benchmark [25].

problem info.			Alg. 1 with option 1				Alg. 1 with option 2		
n	$\kappa(V)$	t_{fl}	spd	$\mathrm{mrp}(\boldsymbol{X})$	k	time	spd	$\mathrm{mrp}(\boldsymbol{Y})$	time
r	$\kappa(\mathcal{P})$	r_s	$(\boldsymbol{X})$	$\mathrm{arp}(\boldsymbol{X})$		res. prec.	$(\boldsymbol{Y})$	$\mathrm{arp}(\boldsymbol{Y})$	prec. $(\tilde{\boldsymbol{Y}})$
s	sep	$\|\|\mathrm{res}\|\|$				$\kappa(\mathrm{mid}\,\boldsymbol{X})$			$\kappa(\mathrm{mid}\,\boldsymbol{Y})$
70	7.7E2	1.7E-2	0	2.0E-1	9	3.9E-1	0	2.5E-1	4.1E-1
1.5	1.5E18	1.4E12		2.0E-4		quad.		1.4E-5	double
1.1	5.4E-4	2.1E1				2.9E16			7.2E12
70		1.6E-2	0	2.2E-1	3	1.5E-1	1	1.9E-3	2.0E-1
1.5				2.2E-4		impr.		3.3E-6	impr.
1.1									
250	1.9E1	5.0E-1	0	4.6E-1	1	1.9	1	5.2E-1	1.9
1.1	-	2.0E10		8.8E-5		double		2.4E-5	double
1.01	-	3.9E-5				4.5E11			1.3E11
500	3.5E2	4.6	0	1.0	1	1.6E1	1	8.4E-1	1.6E1
1.05	-	3.7E10		2.5E-3		double		1.3E-4	double
1.01	-	1.9E-3				2.9E13			1.4E12
700	2.7E3	1.4E1	1	4.5E-4	1	4.5E1	1	1.4E-6	4.4E1
1.005	-	3.3E1		5.8E-10		double		2.8E-12	double
1.01	-	1.2E-9				4.2E5			1.8E5
1000	5.1E4	4.3E1	0	1.2E-2	1	1.4E2	1	3.9E-3	1.4E2
1.005	-	1.5E2		1.6E-7		double		3.6E-10	double
1.01	-	1.2E-6				6.6E8			5.9E7

Table 3.1.

The results from Table 3.2 also illustrate the scaling behavior of Algorithm 1. Remarkably, the computation of the interval enclosure and the test for positive definiteness, i.e., the total run time of Algorithm 1 is consistently only about 3 to 4 times as large as the time spent in `lyap`, i.e. the time needed to obtain the approximate solution. Graphically, this fact is reported in the left part of Figure 3.1, thus illustrating its $\mathcal{O}(n^3)$ complexity as well as the efficiency of Intlab and of the matrix-matrix operation approach of Algorithm 1.

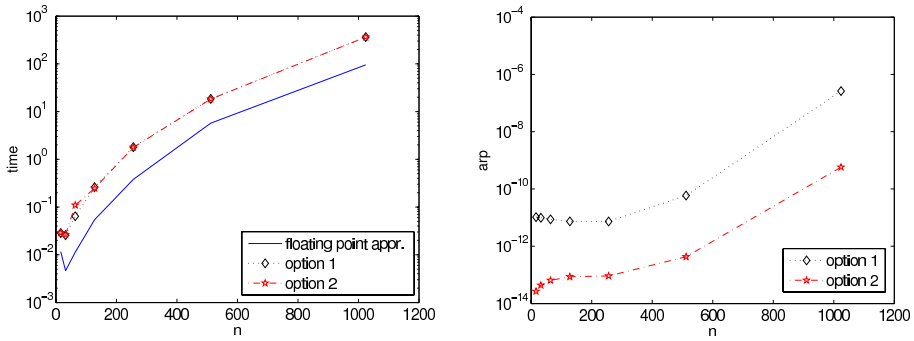

FIG. 3.1. *Time versus dimension (left) and the average relative precision(*arp*) versus dimension (right) for different tests from Example 4.1 of CTLEX with* $r = 1.005$, $s = 1.01$

Table 3.3 reports results for some "real world examples" taken from [6]. The `CDplayer` example refers to the problem of finding a low-cost controller that can make the servo-system of a CD player faster and less sensitive to external shocks [6]. The corresponding model con-

ETNA
Kent State University
http://etna.math.kent.edu

tains 60 vibration modes and both options of Algorithm 1 were successful to verify stability of the matrix A. The `heat-cont` example comes from a dynamical system corresponding to the heat diffusion equation [6] and we were again successful in proving stability of the matrix A. The `iss` example is a structural model of component 1r (Russian service module) of the International Space Station (ISS). Here, both our approaches were successful to verify stability of the matrix A. The `beam` example is the clamped beam model obtained by spatial discretization of a partial differential equation [6]. The `eady` example comes from a model of atmospheric storm track. We refer to [5] for details. Note that the condition number of the eigenvector matrix V for the `eady` example has a condition number of approximately 10^{+9}.

In the examples `CDplayer` and `heatcont` the matrix A is actually normal (with non-real eigenvalues), so that the condition of the eigenvector matrix V is 1. In the other examples, the matrix A is non-normal and we see that the "difficulty" to prove stability increases with the condition of V. Example `eady` is particularly interesting because it is the only example in which option 2 failed while option 1 was successful (using simulated quadruple precision). We attribute this to the high condition number of V which affects the width of the computed interval enclosure for $V\check{X}V^*$. Indeed, this is the only example where the average precision in $\boldsymbol{Y}$ is less than that in $\boldsymbol{X}$.

It can also be noted that in these examples the execution time for the whole verified computation can be up to 100 times more than for the floating point computation of the approximate solution $\check{X}$. There are two main reasons for this deterioration as compared to the examples of Table 3.3. On the one hand our Algorithm 1 sometimes needs more than one sweep through the repeat loop. On the other hand, the floating point computation of $\check{X}$ via Matlab's function `lyap` can take advantage of sparsity of the matrix A, whereas our Algorithm 1 always works with dense matrices. This applies particularly to the examples `CDplayer` and `iss` in which the matrix A is sparse, so that the computation of $\check{X}$ is orders of magnitude faster than it would be with a dense matrix A of the same size.

Our final numerical results deal with situations where instead of a full diagonalization a block diagonalization should be performed. Our first test comes from Example 4.2 in the CTLEX benchmark [25]. This is a 45×45 matrix having just one Jordan block. So the matrix is not exactly diagonalizable, and the computed eigenvector matrix V has a condition number of 10^{17}, approximately. Here, Algorithm 1 fails because it was impossible to obtain the matrix $\boldsymbol{I}_V$, an interval enclosure for V^{-1}. Using `bdschur` to obtain a block diagonalization with a requested bound of 10^8 for the condition of V results in just one block of size 45, i.e., we have the classical reduction to Schur form. Our algorithm with block diagonalization, termed Algorithm 2 in Table 3.4, is now successful. This is actually an exceptionally lucky situation to be attributed to the fact that all elements in the triangular matrix have the same sign so that the accumulation of outward roundings does not cause too much harm when we perform forward substitution in interval arithmetic. The execution time increases substantially due to the fact that the backward substitution for the triangular matrix Δ cannot be cast into matrix-matrix operations.

We note that for this example the function `VERMATREQN` from Versoft is successful in computing an enclosure, which, in addition, is verified to be positive definite by `isspd`. The computed enclosure $\boldsymbol{X}$ by Versoft was obtained after 160s with mrp and arp equal to $5 \cdot 10^{-11}$ and $1.4 \cdot 10^{-13}$, respectively.

The matrix A in our second test comes from Example 5.27 in [14, p. 110]. We multiply the matrix A given there by -1 to make it stable. The matrix is 10×10 and is both, defective and derogatory. Algorithm 1 again fails because it attempts to diagonalize A, obtaining a matrix V with condition number $\kappa(V) = 2.5 \cdot 10^{18}$ so that it is impossible to compute an interval enclosure for its inverse. On the other hand, a block diagonal factorization of the

ETNA
Kent State University
http://etna.math.kent.edu

TABLE 3.3
Numerical results for tests from [6].

problem info.			Alg. 1 with option 1				Alg. 1 with option 2		
n	$\kappa(V)$	t_{fl}	spd	$\mathrm{mrp}(\boldsymbol{X})$	k	time	spd	$\mathrm{mrp}(\boldsymbol{Y})$	time
name	$\kappa(\mathcal{P})$	r_s	$(\boldsymbol{X})$	$\mathrm{arp}(\boldsymbol{X})$		res. prec.	$(\boldsymbol{Y})$	$\mathrm{arp}(\boldsymbol{Y})$	prec. $(\tilde{\boldsymbol{Y}})$
	sep	$\|\|res\|\|$				$\kappa(\mathrm{mid}\,\boldsymbol{X})$			$\kappa(\mathrm{mid}\,\boldsymbol{Y})$
120	1	4.0E-2	1	1.5E-13	3	3.6	1	2.9E-12	2.9
`CDplayer`	1.8E6	3.3E4		5.5E-15		double		1.4E-14	double
	4.9E-2	2.4E-14				3.3E4			3.3E4
200	1	1.6E-1	1	2.0E-8	1	6.8E-1	1	1.0	6.4E-1
`heat-cont`	2.4E4	1.6E4		2.5E-11		double		2.5E-12	double
	1.9E-1	6.3E-11				2.0E+4			1.6E4
270	6.1E1	1.8E-1	1	5.6-9	3	2.6E1	1	3.1E-13	1.9E1
`iss`	2.3E7	9.8E1		3.5E-12		double		3.7E-14	double
	3.3E-4	8.4E-12				3.8E3			9.9E1
348	3.6E2	9.4E-1	0	8.0E-1	1	1.1E1	1	8.3E-1	1.1E1
`beam`	-	1.0E5		1.3E-5		double		5.6E-5	double
	-	2.0E-5				1.4E10			5.1E5
598	1.1E9	4.8	0	1.0	1	5.3E1	0	2.9E-5	5.4E1
`eady`	-	4.2E1		6.7E-4		double		1.2E-10	double
	-	1.8E-9				4.6E5			1.1E18
598		4.7	1	1.5E-3	3	2.9E2	0	3.6E-6	2.9E2
`eady`				1.7E-12		quad.		7.6E-12	double
		1.8E-9				4.6E5			7.4E17

matrix A computed with `bdschur` results in one block of size 5, one block of size 4 and one block of size 1 with a condition number for V of approximately 10^2. The second row of Table 3.4 contains results for this example with this block-diagonalization, where now our algorithm is again successful. As before, `Versoft` is also successful for this example. It takes 0.5s and obtains an enclosure for X with mrp and arp equal to $1.2 \cdot 10^{-11}$ and $3.3 \cdot 10^{-12}$, respectively.

TABLE 3.4
Numerical results using "Alg. 2.", i.e., the variant of Algorithm 1 which uses block diagonalization. The first test is from Example 4.2 from the CTLEX benchmark [25], while the second is from Example 5.27 in [14].

problem info.			Alg. 2 with option 1				Alg. 2 with option 2		
n	$\kappa(V)$	t_{fl}	spd	$\mathrm{mrp}(\boldsymbol{X})$	k	time	spd	$\mathrm{mrp}(\boldsymbol{Y})$	time
λ	$\kappa(\mathcal{P})$	r_s	$(\boldsymbol{X})$	$\mathrm{arp}(\boldsymbol{X})$		res. prec.	$(\boldsymbol{Y})$	$\mathrm{arp}(\boldsymbol{Y})$	prec. $(\tilde{\boldsymbol{Y}})$
s	sep	$\|\|\mathrm{res}\|\|$				$\kappa(\mathrm{mid}\,\boldsymbol{X})$			$\kappa(\mathrm{mid}\,\boldsymbol{Y})$
45	1.0	6.9E-3	1	6.1E-3	1	5.8E1	1	1.4E-6	5.8E1
-1.1	3.4E5	2.3		9.1E-6		double		3.7E-11	double
1.1	4.1E-17	8.0E-12				2.1E4			1.9E4
10	1.8E2	2.1E-1	1	3.9E-10	1	6.7E-1	1	7.1E-12	6.7E-1
-	5.3E4	3.0		3.3E-11		double		3.3E-12	double
-	5.6E-16	2.8E-11				7.2E3			9.4E4

We conclude this section with some more general comments. Algorithm 1 consists of three critical parts: The computation of the floating point appoximation $\check{X}$, which should be accurate, the computation of the enclosure $\boldsymbol{K}$ which should be narrow, and the check for positive definiteness, which should be successful. For the first part, we can take the

ETNA
Kent State University
http://etna.math.kent.edu

best floating point algorithm available. Obtaining a good floating point approximation will be particularly hard if the problem is very stiff, i.e., if the value of r_s reported in all our tables is large. Computing the enclosure $\boldsymbol{K}$ crucially depends on the condition number of the eigenvector matrix V. If this condition number is too large, we will not succeed in computing an enclosure, and we then have to use the block method (which tries to bound the condition number of V) instead. If a small condition number for V can only be obtained with relatively large blocks, the approach of (the block version of) Algorithm 1 will fail as a whole because the interval quantities computed in the forward substitution process induced by the blocks will yield interval quantities which become too large. Computing $\boldsymbol{K}$ can also fail just because $\check{X}$ is not accurate enough. Finally, the success of checking positive definiteness using `isspd` on $H(\boldsymbol{X})$ or $H(\boldsymbol{Y})$ depends on the condition of $\mathrm{mid}\,(X(\boldsymbol{X}))$ or $\mathrm{mid}\,(H(\boldsymbol{Y}))$, respectively, and the radii of the entries of $\boldsymbol{X}$ and $\boldsymbol{Y}$. The method is more likely to succeed when the interval entries are narrow, i.e. when we have tight enclosures, and when $\mathrm{mid}\,(H(\boldsymbol{X}))$ or $\mathrm{mid}\,(H(\boldsymbol{Y}))$ is well conditioned. The latter property, in principle, depends on the choice of the matrix C, which we always took to be $-I$ in our examples. As a rule, we would also expect the matrices to be less well conditioned when the stiffness ratio r_s of the matrix A is large.

Note that we can also fail to obtain an enclosure because the computed approximate solution $\check{X}$ is not accurate enough.

4. Conclusions. We presented a verified numerical method to prove stability of matrices by computing interval enclosures for the solution of a Lyapunov equation and subsequently showing that this solution is positive definite. If our algorithm is successful, it is proved in a mathematically rigorous manner that the matrix is stable. If the algorithm is not successful, we do not have a mathematically rigorous result, i.e., we do not know whether the matrix is stable or not. We presented two options for the task of proving the positive definiteness, where the one which works with the interval enclosure for the transformed solution $Y = VXV^*$ usually yields better enclosures and is successful in more cases. Due to an implementation oriented towards matrix-matrix operations, the algorithm is time efficient when implemented in Intlab. In its basic version our method requires the matrix to be diagonalized numerically, but it can be generalized to use a block diagonalization in cases where the eigenvector matrix is too ill conditioned. Larger blocks, however, will usually prevent our algorithm from being successful since we then suffer from the accumulation effect in outward roundings during the forward substitution process.

Let us finally note that Algorithm 1 can be adapted to the case where the input matrix A is an interval matrix $\boldsymbol{A}$. This situation arises when one wants to model uncertainties in the input data A. In Algorithm 1 we then compute the approximate solution $\check{X}$ and V, D with respect to the midpoint of $\boldsymbol{A}$, while all other occurencies of A in Algorithm 1 have to be replaced by $\boldsymbol{A}$. Numerical tests show that this approach gives good enclosures as long as $\boldsymbol{A}$ has narrow interval entries. As is to be expected, this approach is faster than `VERMATREQN` from the `VERSOFT` library while the enclosures obtained are (slightly) larger than those from `VERMATREQN`.

REFERENCES

[1] G. ALEFELD, *Inclusion methods for systems of nonlinear equations—the interval Newton method and modifications*, in Topics in Validated Computations (Oldenburg, 1993), J. Herzberger, ed., vol. 5 of Stud. Comput. Math., North-Holland, Amsterdam, 1994, pp. 7–26.

[2] G. ALEFELD AND J. HERZBERGER, *Introduction to Interval Computations*, Academic Press, New York, 1983.

ETNA
Kent State University
http://etna.math.kent.edu

[3] A. BAVELY AND G. STEWART, *An algorithm for computing reducing subspaces by block diagonalization*, SIAM J. Numer. Anal., 16 (1979), pp. 359–367.
[4] P. BENNER AND T. DAMM, *Lyapunov equations, energy functionals, and model order reduction of bilinear and stochastic systems*, SIAM J. Control Optim., 49 (2011), pp. 686–711.
[5] Y. CHAHLAOUI AND P. VAN DOOREN, *A collection of benchmark examples for model reduction of linear time invariant dynamical systems*, Tech. Rep. SLICOT Working Note 2002-2., 2002.
http://www.icm.tu-bs.de/NICONET/REPORTS/SLWN2002-2.ps.gz
[6] ———, *Benchmark examples for model reduction of linear time-invariant dynamical systems*, in Dimension Reduction of Large-Scale Systems, P. Benner, D. C. Sorensen, and V. Mehrmann, eds., vol. 45 of Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2005, pp. 379–392.
[7] D. CORDES, *Verifizierter Stabilitätsnachweis für Lösungen von Systemen periodischer Differentialgleichungen auf dem Rechner mit Anwendungen.*, Ph.D. Thesis, Department of Mathematics, University of Karsruhe, 1987.
[8] N. DELANOUE, *Algorithmes numériques pour l'analyse topologique: Analyse par intervalles et théorie des graphes*, PhD thesis, Laboratoire d'Ingénierie des Systèmes Automatisés, Université d'Angers, 2006.
http://lisa.univ-angers.fr/THESES/TheseDelanoueNicolas.zip
[9] N. DELANOUE, L. JAULIN, AND B. COTTENCEAU, *Stability analysis of a nonlinear system using interval analysis*, preprint, Laboratoire d'Ingénierie des Systèmes Automatisés, Université d'Angers, 2006.
193.49.146.171/~delanoue/article/delanoue.pdf
[10] F. DOMES AND A. NEUMAIER, *Rigorous enclosures of ellipsoids and directed Cholesky factorizations*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 262–285.
[11] F. W. FAIRMAN, *Linear Control Theory: The State Space Approach*, John Wiley & Sons, New York, 1998.
[12] A. FROMMER AND B. HASHEMI, *Verified error bounds for solutions of Sylvester matrix equations*, Linear Algebra Appl., 436 (2012), pp. 405–420.
[13] Z. GAJIĆ AND M. QURESHI, *Lyapunov Matrix Equation in System Stability and Control*, Dover, New York, 2008.
[14] R. GREGORY AND D. KARNEY, *A Collection of Matrices for Testing Computational Algorithms*, Wiley-Interscience, New York, 1969.
[15] B. GROSS, *Verification of asymptotic stability for interval matrices and applications in control theory*, in Scientific Computing with Automatic Result Verification, E. Adams and U. Kulisch, eds., Academic Press, New York, 1993, pp. 357–395.
[16] S. J. HAMMARLING, *Numerical solution of the stable, nonnegative definite Lyapunov equation*, IMA J. Numer. Anal., 2 (1982), pp. 303–323.
[17] R. HAMMER, M. HOCKS, U. KULISCH, AND D. RATZ, *Numerical Toolbox for Verified Computing. Volume I: Basic Numerical Problems. Theory, Algorithms, and Pascal-XSC Programs.*, Springer, Berlin, 1993.
[18] D. HERTZ, *The extreme eigenvalues and stability of Hermitian interval matrices*, IEEE Trans. Circuits Systems-I: Fund. Theory Appl., 39 (1992), pp. 463–466.
[19] W. HOFSCHUSTER AND W. KRÄMER, *C-XSC 2.0 – a C++ library for extended scientific computing*, in Numerical Software with Result Verification, R. Alt, A. Frommer, R. Kearfott, and W. Luther, eds., vol. 2991 of Lecture Notes in Computer Science, Springer, Berlin, 2004, pp. 15–35.
[20] R. A. HORN AND C. R. JOHNSON, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, 1994.
[21] A. ISERLES, *A First Course in the Numerical Analysis of Differential Equations*, Cambridge University Press, New York, 1996.
[22] R. B. KEARFOTT, *Interval computations: Introduction, uses, and resources*, Euromath Bulletin, 2 (1996), pp. 95–112.
[23] R. KLATTE, U. W. KULISCH, A. WIETHOFF, C. LAWO, AND M. RAUCH, *C-XSC. A C++ Class Library for Extended Scientific Computing*, Springer, Berlin, 1993.
[24] R. KRAWCZYK, *Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken*, Computing, 4 (1969), pp. 187–201.
[25] D. KRESSNER, V. MEHRMANN, AND T. PENZL, *CTLEX - A collection of benchmark examples for continuous-time Lyapunov equations*, Tech. Rep. SLICOT Working Note 1999-6., 1999.
http://www.slicot.org/REPORTS/SLWN1999-6.ps.gz
[26] A. J. LAUB, *Matrix Analysis for Scientists & Engineers*, SIAM, Philadelphia, PA, 2005.
[27] R. LOHNER, *Einschliessung der Lösung gewöhnlicher Anfangs-und Randwertaufgaben und Anwendungen*, PhD thesis, Universität Karlsruhe, 1988.
[28] S. MIYAJIMA, *Fast enclosure for solutions of Sylvester equations*, Linear Algebra Appl., to appear, 2012.
[29] R. E. MOORE, R. B. KEARFOTT, AND M. J. CLOUD, *Introduction to Interval Analysis*, SIAM, Philadelphia, 2009.
[30] A. NEUMAIER, *Interval Methods for Systems of Equations*, no. 37 in Encyclopedia of Mathematics and its Applications, Cambridge University Press, Cambridge, 1990.
[31] M. L. OVERTON, *Numerical Computing with IEEE Floating Point Arithmetic*, SIAM, Philadelphia, 2001.

ETNA
Kent State University
http://etna.math.kent.edu

[32] K. OZAKI, T. OGITA, AND S. OISHI, *Tight and efficient enclosure of matrix multiplication by using optimized BLAS*, Numer. Linear Algebra Appl., 18 (2011), pp. 237–248.
[33] A. RAUH, J. MINISINI, E. HOFER, AND H. ASCHEMANN, *Robust and optimal control of uncertain dynamical systems with state-dependent switchings using interval arithmetic*, Reliab. Comput., 15 (2011), pp. 333–344.
[34] J. ROHN, *VERSOFT: Verification software in MATLAB/INTLAB*. http://uivtx.cs.cas.cz/~rohn/matlab
[35] ———, *Checking positive definiteness or stability of symmetric interval matrices is NP-hard*, Comment. Math. Univ. Carolin., 35 (1994), pp. 795–797.
[36] ———, *Positive definiteness and stability of interval matrices*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 175–184.
[37] S. M. RUMP, *Solving algebraic problems with high accuracy*, in A New Approach to Scientific Computation, W. Miranker and E. Kaucher, eds., vol. 7 of Comput. Sci. Appl. Math., Academic Press, New York, 1983, pp. 51–120.
[38] ———, *Fast and parallel interval arithmetic*, BIT, 39 (1999), pp. 534–554.
[39] ———, *INTLAB – INTerval LABoratory*, in Developments in Reliable Computing, T. Csendes, ed., Kluwer Academic Publishers, Dordrecht, 1999, pp. 77–104.
[40] ———, *Verification of positive definiteness*, BIT, 46 (2006), pp. 433–452.
[41] J. SHAO AND X. HOU, *Positive definiteness of Hermitian interval matrices*, Linear Algebra Appl., 432 (2010), pp. 970–979.
[42] H. L. STALFORD AND C. H. CHAO, *A necessary and sufficient condition in Lyapunov robust control*, J. Optim. Theory Appl., 63 (1989), pp. 191–204.
[43] R. S. VARGA, *Geršgorin and His Circles*, Springer, Berlin, 2004.
[44] D. S. WATKINS, *Fundamentals of Matrix Computations*, Second ed., Pure and Applied Mathematics, Wiley-Interscience, New York, 2002.
[45] A. WEINMANN, *Uncertain Models and Robust Control*, Springer, Vienna, 1991.